# 文档概述

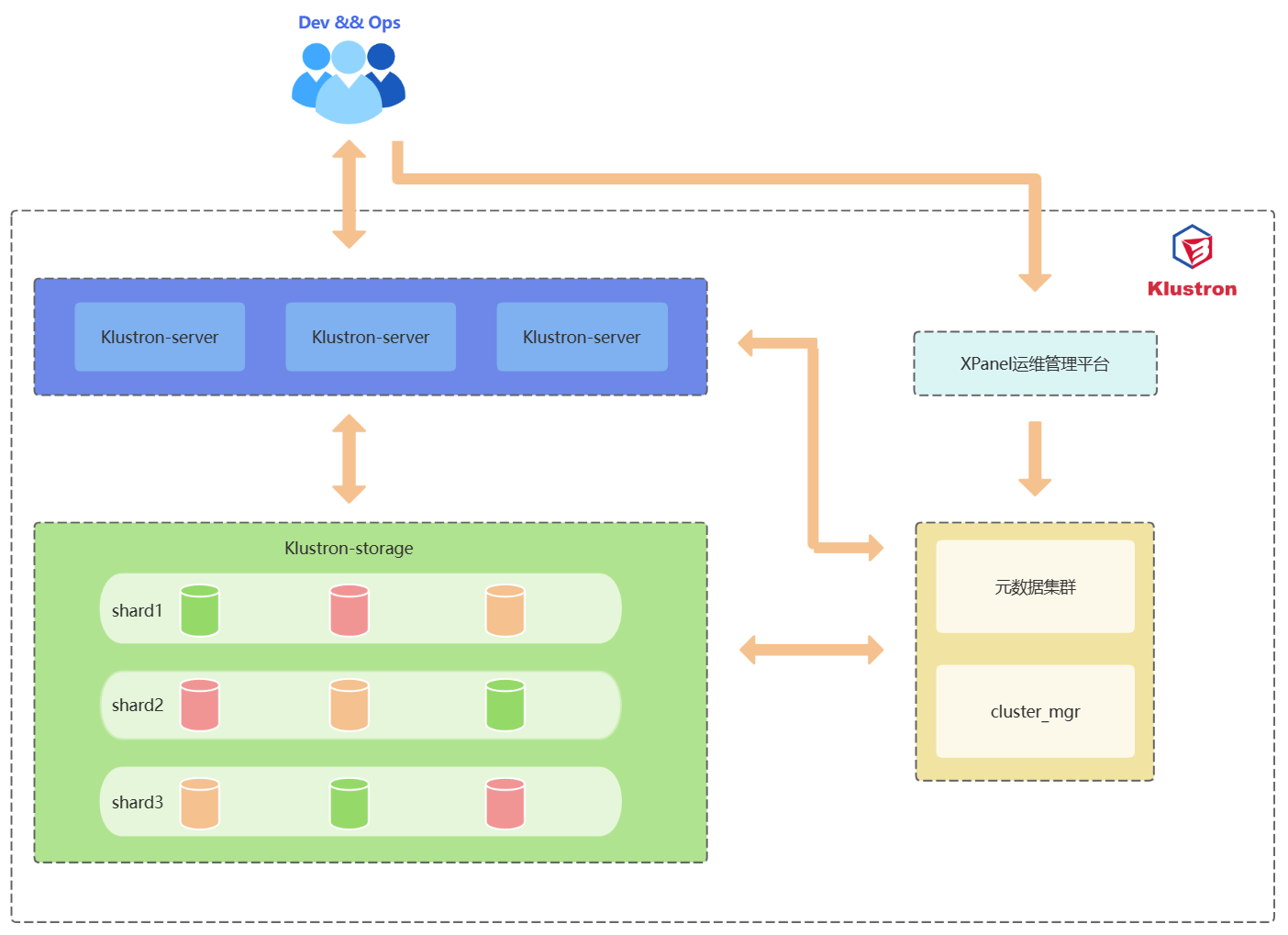
## 文档概述

本文档描述了Klustron的系统构架、组件介绍、部署，再到描述了基本的运维和集群管理操作，以及相关Klustron相关工具的使用和Klustron相关内容的介绍，最后还会对常见问题进行解答。帮助用户从0到1，快速的使用Klustron，再到熟悉的运维和管理Klustron。

# 架构介绍和术语说明

## 架构介绍

**Klustron系统架构：**



## 组件介绍

|  |  |
| --- | --- |
| **Klustron 集 群 组 件 介 绍** | |
| **组件名称** | **功能** |
| Klustron-server | 计算节点，负责接收应用客户端请求，通过合并处理存储节点返回的结果，最终返回客户端结果。计算节点是无状态节点，常规使用方式是在负载均衡中配置计算节点，当有计算节点故障，负载均衡层会将请求发送到其他正常的计算节点，继续提供来自应用客户端请求服务。 |
| Klustron-storage | 存储集群，负责实现数据持久化、金融级数据一致性。存储集群由2个或以上shard组成，每一个shard是3副本，1个master和2个slave。主节点接受来自计算节点的读写请求，执行请求并返回结果给计算节点。 |
| 元数据集群 | 负责存储Klustron集群的拓扑结构、节点连接信息、DDL日志，commit log，和其他集群管理日志。 |
| cluster\_mgr | 负责维护正确的集群和节点状态，实现集群管理、集群逻辑备份和恢复, 集群物理备份和恢复、水平弹性伸缩等功能。 |
| XPanel | GUI 工具软件，提供网页集群管理平台，让DBA通过点击鼠标就可以轻松完成绝大部分的数据库运维管理工作。 |
| node\_mgr | node\_mgr负责将cluster\_mgr下发的操作在对应的机器节点上执行命令，以及定时上报集群主节点的心跳信息。每一台机器节点都存在一个node\_mgr。 |

## 术语说明

|  |  |
| --- | --- |
| **Klustron 常 用 术 语 介 绍** | |
| **术语** | **解释** |
| 集群  /Cluster | 在Klustron中，集群由若干个计算节点和若干个存储集群shard组成，例如A集群由3个计算节点和2个存储集群shard组成。 |
| 计算节点/Klustron-server  /CN(Computing Node) | 负责接收应用客户端请求，通过合并处理存储节点返回的结果，最终返回客户端结果。每个计算节点都是无状态且相互独立的，集群中有多个计算节点，且接入了负载均衡层，当单个计算节点故障时，不会对集群有影响，集群仍然持续提供读写服务。 |
| 存储集群  /Klustron-storage  /Shard | 一个 Klustron 集群包含一个或者多个 shard, 每个 shard 存储一部分用户数据，不同 shard 存储的用户数据没有交集。每个 shard 有一个主节点和多个备机节点，主备节点通过 binlog 复制实现高可用。 |
| 存储节点  /Storage Node | 用于存储用户业务数据，是一个 kunlun-storage 组件的运行实例。若干个存储节点组成一个shard，在shard中作为主节点或者备节点运行。 |
| 元数据集群  /Metadata shard | 由多个存储节点组成的复制组，用于管理Klustron集群的元数据，不用于存储用户业务数据，元数据集群在运行时是一个复制组。可以服务多个Klustron集群。 |
| cluster\_mgr  /集群管理器  /Cluster Manager | 该节点负责接收客户端的Klustron集群操作和管理请求，通过将实际动作发送给各个节点管理器，来完成请求。 |
| XPanel | 一个 Web 应用程序，运行在 docker 镜像中，用户可以使用浏览器连接 XPanel 来执行集群管理操作。 |
| node\_mgr  /节点管理器  /Node Manager | 每台工作机器都需要部署的一个的本地命令执行节点，该节点将接收来自集群管理器的请求，针对存储节点、计算节点、以及Klustron在本机的其他组件执行各种操作，包括但不限于安装，删除，停止，备份等。 |

# 标准集群部署

## 部署前置要求

### 操作系统和CPU架构

|  |  |
| --- | --- |
| **操作系统版本** | **CPU的架构（x86\_64/AMD64/ARM64）** |
| CentOS 7.5及以上 | x86\_64  ARM64 |
| CentOS 8 | x86\_64  ARM64 |
| 统信服务器操作系统V20（1050a） | AMD64  ARM64 |
| 统信服务器操作系统V20（1050e） | AMD64  ARM64 |
| openSUSE 15 | x86\_64 |
| Ubuntu 20.04 | x86\_64 |
| Kylin v10 | x86\_64  ARM64 |

### 服务器配置

#### 测试和开发环境配置建议

对于开发和测试环境，建议至少使用三台机器，每台机器上可以部署多种类型的组件, 以下是各种组件的具体要求:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **组 件** | **CPU** | **内存** | **磁盘类型及可用空间** | **网络** | **部署个数** | **部署说明** |
| Klustron-server | 4 | 8G+ | SAS，20G+ | 千兆网卡 | 1 | 可以与存储集群混合部署 |
| Klustron-storage | 4 | 8G+ | SAS，20G+ | 千兆网卡 | 1 | 1个存储集群至少包含3个节点，即一主两备 |
| 元数据集群 | 4 | 8G+ | SAS,20G+ | 千兆网卡 | 1 | 可以与存储集群混合部署 |
| cluster\_mgr | 2 | 4G+ | SAS,20G+ | 千兆网卡 | 1 | 可以与存储集群混合部署 |
| XPanel | 2 | 4G+ | SAS,10G+ | 千兆网卡 | 1 | 可以与存储集群混合部署 |
| node\_mgr | 1 | 2G+ | SAS,10G+ | 千兆网卡 | 1 | 每台机器都会自动部署 |

#### 生产环境配置要求

对于生产环境，至少需要6台机器；机器资源足够时，建议使用9台机器，3台计算节点+6台存储节点，其他组件可以与存储节点混合部署；以下是各种组件的具体要求:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **组 件** | **CPU** | **内存** | **磁盘类型及可用空间** | **网络** | **部署个数** | **部署说明** |
| Klustron-server | 16 | 32G+ | SSD，20G+ | 万兆网卡 | 3 | 独立部署 |
| Klustron-storage | 8 | 32G+ | SSD，200G+ | 万兆网卡 | 3 | 1个存储集群至少包含3个节点，即一主两备 |
| 元数据集群 | 4 | 8G+ | SSD,50G+ | 万兆网卡 | 1 | 可以与存储集群混合部署 |
| cluster\_mgr | 4 | 8G+ | SSD,20G+ | 万兆网卡 | 1 | 可以与存储集群混合部署 |
| XPanel | 4 | 8G+ | SSD,10G+ | 万兆网卡 | 1 | 可以与计算节点混合部署 |
| node\_mgr | 1 | 2G+ | SSD,10G+ | 万兆网卡 | 1 | 每台机器都会自动部署 |

### 网络配置

|  |  |  |
| --- | --- | --- |
| **组 件** | **默认端口/端口范围** | **端口说明** |
| Klustron-server | 47000-48000 | 计算节点提供的pg或mysql客户端连接访问通信端口范围 |
| Klustron-storage | 57000-58000 | 存储节点通信端口范围 |
| 元数据集群 | 6001 | 元数据集群访问通信端口 |
| cluster\_mgr | 58000 | cluster\_mgr的raft协议端口 |
| cluster\_mgr | 58001 | cluster\_mgr的http协议端口 |
| cluster\_mgr | 59010 | cluster\_mgr 默认的 prometheus 端口 |
| XPanel | 18080 | XPanel运管平台服务访问端口 |
| node\_mgr | 58002 | node\_mgr默认的http端口 |
| node\_mgr | 58003 | node\_mgr默认的tcp端口 |
| node\_mgr | 58010 | node\_mgr默认的 prometheus 端口 |

## 在线部署

### 部署前准备工作-所有机器操作

**该步骤的准备工作需要在每台机器上进行操作。**

#### 清理系统安装自带的MariaDB

使用root用户运行下面的命令：

yum remove postfix

rpm -e mariadb-libs

#### 关闭防火墙

在root用户下执行命令：

systemctl stop firewalld

systemctl disable firewalld

#### 创建kunlun组和kunlun用户

使用root执行命令并设置用户密码：

useradd --create-home --shell /bin/bash kunlun

passwd kunlun

#### 创建相关目录并设置所属组

创建软件安装目录 /data1/kunlun，可根据用户定义的规范进行创建即可，这里以磁盘挂载到/data1目录为例，并将目录的属组设置为 kunlun:kunlun。

使用 root 执行下面的命令：

mkdir -p /data1/kunlun /data1/softwares

# **/data1/kunlun为各组件安装完成后的目录** **/data1/softwares为部署前所存放安装包、相关脚本和文件的目录**

chown -R kunlun:kunlun /data1/kunlun /data1/softwares

#### 修改系统文件句柄数

使用root执行下面的命令：

vim /etc/security/limits.conf

#在文件最后一行的下面加入：

\* soft nofile 65536

\* hard nofile 200000

#### 设置kunlun用户sudo免密执行root命令

使用root执行下面的命令：

visudo

#在最后加上下面的内容：

kunlun ALL=(ALL) NOPASSWD: ALL

### 部署前准备工作-中控机操作

**该步骤的准备工作只需要在中控机上进行操作。**

#### 中控机与其他机器配置SSH互信

以kunlun用户在中控机执行下面的命令：

ssh-keygen -t rsa

运行该命令后，在所有需要输入的地方直接回车就可以了。成功后会在/home/kunlun/目录下的.ssh目录中生成 id\_rsa 和 id\_rsa.pub 两个文件，其中 id\_rsa 是存放私钥的文件，而 id\_rsa.pub 则为存放公钥的文件。

将中控机的公钥复制到其他服务器上，包括中控机本机：

ssh-copy-id -i ~/.ssh/id\_rsa.pub 192.168.40.151

ssh-copy-id -i ~/.ssh/id\_rsa.pub 192.168.40.152

ssh-copy-id -i ~/.ssh/id\_rsa.pub 192.168.40.153

验证中控机上执行命令验证kunlun用户到其他机器上的免密登陆:

ssh 192.168.40.151 date

ssh 192.168.40.152 date

ssh 192.168.40.153 date

#### 安装python2,git和wget等工具

使用root执行下面的命令：

yum -y install python git wget yum-utils

#### 部署XPanel

由于计划在中控上部署XPanel，需要提前安装和部署docker。

使用root执行下面的命令：

清除系统自带的docker相关的rpm：

yum remove docker \

docker-client \

docker-client-latest \

docker-common \

docker-latest \

docker-latest-logrotate \

docker-logrotate \

docker-engine

切换阿里镜像：

yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

安装docker社区版：

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

启动docker，并设置其随操作系统自启动：

systemctl start docker

systemctl enable docker

#### 下载最新部署工具

将部署工具下载到/data1/softwares目录下，使用root用户执行：

cd /data1/softwares

git clone -b 1.1 https://gitee.com/zettadb/cloudnative.git

#### 下载安装需要的软件包

使用root执行下面的命令：

cd /data1/softwares/setuptools-36.5.0/cloudnative/cluster/clustermgr

wget http://zettatech.tpddns.cn:14000/thirdparty/hadoop-3.3.1.tar.gz

wget http://zettatech.tpddns.cn:14000/thirdparty/jdk-8u131-linux-x64.tar.gz

wget http://zettatech.tpddns.cn:14000/thirdparty/mysql-connector-python-2.1.3.tar.gz

wget http://zettatech.tpddns.cn:14000/thirdparty/prometheus.tgz

wget http://zettatech.tpddns.cn:14000/thirdparty/haproxy-2.5.0-bin.tar.gz

wget http://zettatech.tpddns.cn:14000/thirdparty/efk/filebeat-7.10.1-linux-x86\_64.tar.gz

VERSION=1.1.2

wget http://downloads.Klustron.com//releases/$VERSION/release-binaries/kunlun-cluster-manager-$VERSION.tgz

wget http://downloads.Klustron.com//releases/$VERSION/release-binaries/kunlun-node-manager-$VERSION.tgz

wget http://downloads.Klustron.com//releases/$VERSION/release-binaries/kunlun-server-$VERSION.tgz

wget http://downloads.Klustron.com//releases/$VERSION/release-binaries/kunlun-storage-$VERSION.tgz

wget http://downloads.Klustron.com//releases/$VERSION/release-binaries/kunlun-proxysql-$VERSION.tgz

chown -R kunlun:kunlun /data1/softwares/

### 部署安装

#### 编辑部署拓扑json文件

中控机上使用kunlun用户修改文件/data1/softwares/setuptools-36.5.0/cloudnative/cluster/cluster\_and\_node\_mgr.json ，加入相关的配置信息，这里以192.168.40.151（中控机），192.168.40.152，192.168.40.153三台机器为例。

{

"machines":[

{

"ip":"192.168.40.151",

"basedir":"/data1/kunlun",

"user":"kunlun"

},

{

"ip":"192.168.40.152",

"basedir":"/data1/kunlun",

"user":"kunlun"

},

{

"ip":"192.168.40.153",

"basedir":"/data1/kunlun",

"user":"kunlun"

}

],

"meta":{

"ha\_mode": "rbr",

"nodes":[

{

"ip":"192.168.40.151"

},

{

"ip":"192.168.40.152"

},

{

"ip":"192.168.40.153"

}

]

},

"cluster\_manager": {

"nodes": [

{

"ip": "192.168.40.151"

},

{

"ip": "192.168.40.152"

},

{

"ip": "192.168.40.153"

}

]

},

"node\_manager": {

"nodes": [

{

"ip": "192.168.40.151"

},

{

"ip": "192.168.40.152"

},

{

"ip": "192.168.40.153"

}

]

},

"xpanel": {

"ip": "192.168.40.151",

"image": "registry.cn-hangzhou.aliyuncs.com/kunlundb/kunlun-xpanel:1.1.2"

}

}

#### 执行安装

生成安装脚本

中控机使用kunlun用户执行下面的命令：

cd /data1/softwares/setuptools-36.5.0/cloudnative/cluster

sudo python2 setup\_cluster\_manager.py --autostart --config=cluster\_and\_node\_mgr.json --product\_version=$version --action=install

# **$version** 是指需要安装的集群版本号，例如：1.1.2

示例：

sudo python2 setup\_cluster\_manager.py --autostart --config=cluster\_and\_node\_mgr.json --product\_version=1.1.2 --action=install

运行安装脚本

中控机使用kunlun用户执行下面的命令：

bash -e /data1/softwares/setuptools-36.5.0/cloudnative/cluster/clustermgr/install.sh

#### 登陆XPanel进行集群创建

浏览器上登陆XPanel控制台，在能访问192.168.40.151的机器上打开浏览器，输入地址：http://192.168.40.151:18080/KunlunXPanel

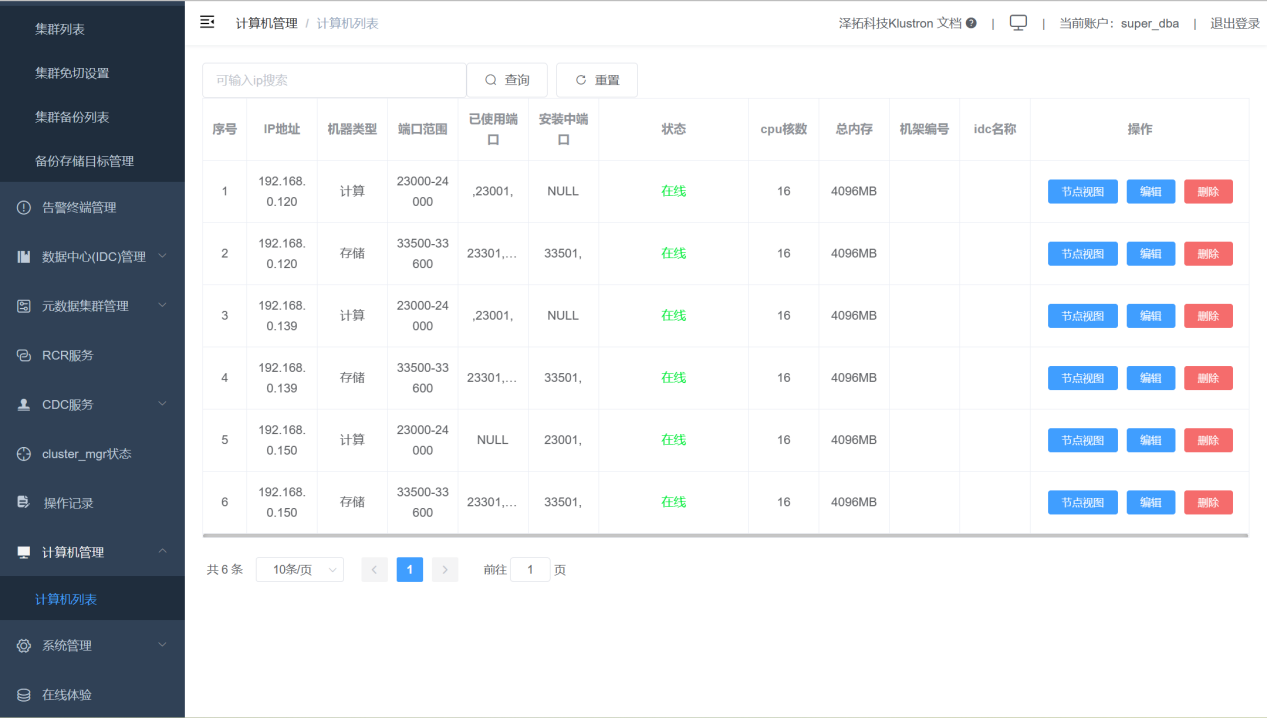
初次登录用户名和密码是：super\_dba/super\_dba，初次登陆需要修改super\_dba密码。

登录后，首页显示如下：



点击左边控制栏“计算机管理”“计算机列表”，显示下面的界面。

检查所有的机器处于在线状态：



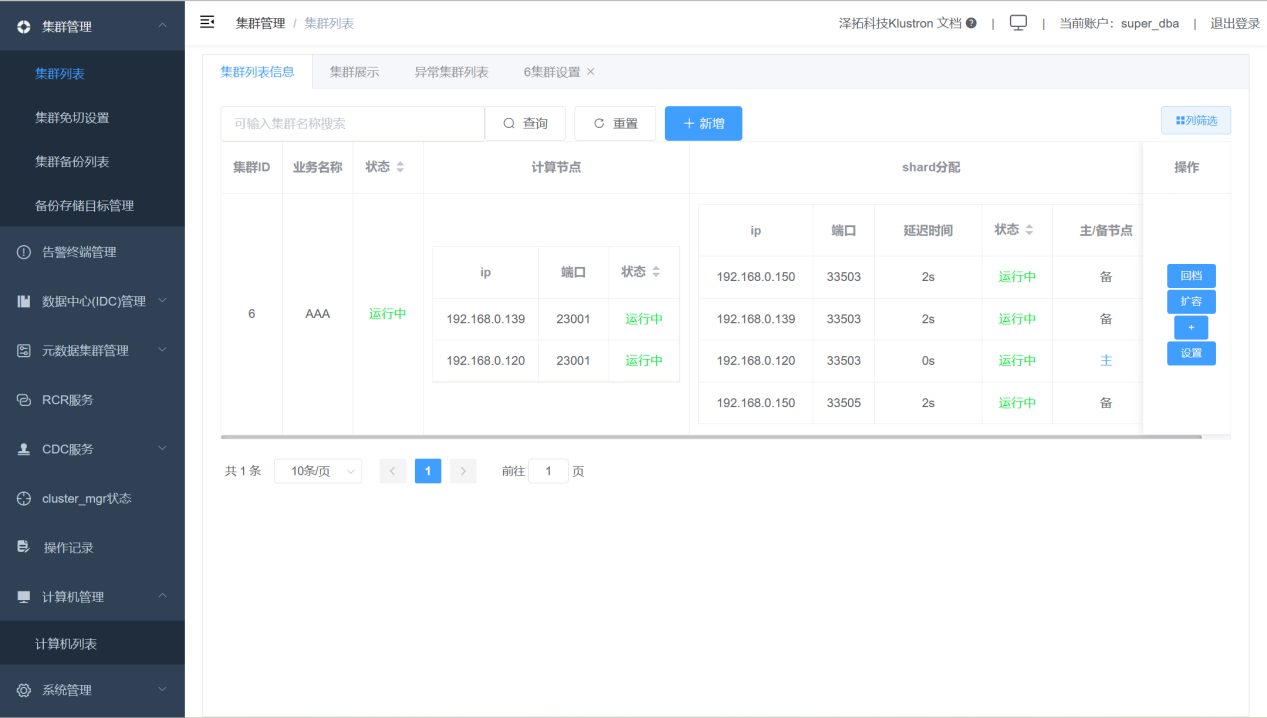
增加集群。

点击“集群管理”，“集群列表”，再点击“新增”按钮



添加集群参数如下图：





完成集群安装后，通过连接计算节点的ip和port，默认用户名和密码都是abc，就可以使用Klustron Cluster了，登录后用户根据需求进行用户的创建和密码的修改。

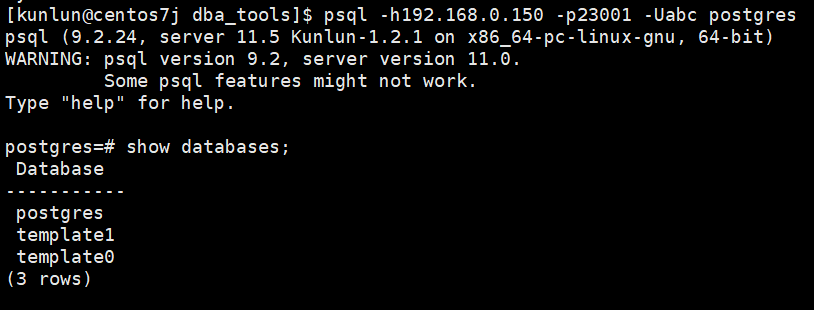
#### pg和mysql协议登录验证

**Tips：postgresql协议和mysql协议登录访问的端口号不同，具体见XPanel中【集群列表】->【设置】->【计算节点列表】。**

Postgresql登录验证：

psql -h192.168.0.150 -p23001 -Uabc postgres

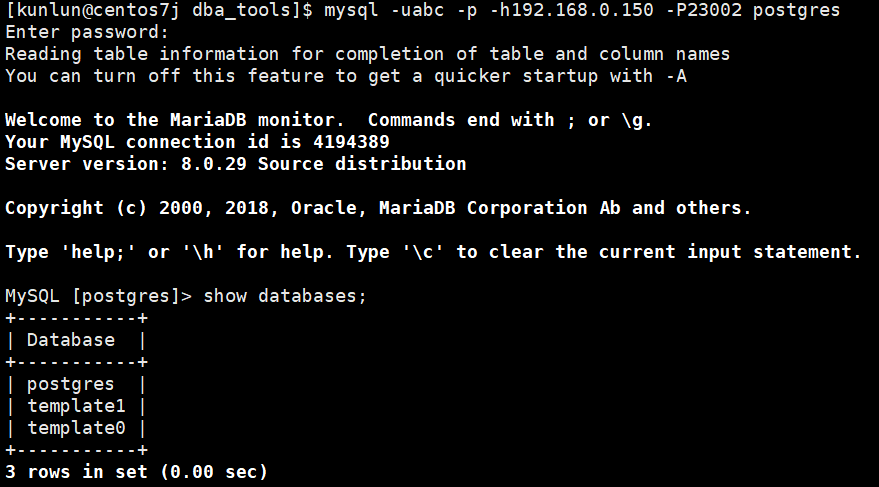
如图：



mysql登录验证：

mysql -uabc -p -h192.168.0.150 -P23002 postgres

如图：



## 离线部署

### 部署前准备工作-所有机器操作

**该步骤的准备工作需要在每台机器上进行操作。**

#### 清理系统安装自带的MariaDB

使用root用户运行下面的命令：

yum remove postfix

rpm -e mariadb-libs

#### 关闭防火墙

在root用户下执行命令：

systemctl stop firewalld

systemctl disable firewalld

#### 创建kunlun组和kunlun用户

使用root执行命令并设置用户密码：

useradd --create-home --shell /bin/bash kunlun

passwd kunlun

#### 创建相关目录并设置所属组

创建软件安装目录 /data1/kunlun，可根据用户定义的规范进行创建即可，这里以磁盘挂载到/data1目录为例，并将目录的属组设置为 kunlun:kunlun。

使用 root 执行下面的命令：

mkdir -p /data1/kunlun /data1/softwares

# **/data1/kunlun为各组件安装完成后的目录** **/data1/softwares为部署前所存放安装包、相关脚本和文件的目录**

chown -R kunlun:kunlun /data1/kunlun /data1/softwares

#### 修改系统文件句柄数

使用root执行下面的命令：

vim /etc/security/limits.conf

#在文件最后一行的下面加入：

\* soft nofile 65536

\* hard nofile 200000

#### 设置kunlun用户sudo免密执行root命令

使用root执行下面的命令：

visudo

#在最后加上下面的内容：

kunlun ALL=(ALL) NOPASSWD: ALL

### 部署前准备工作-中控机操作

**该步骤的准备工作只需要在中控机上进行操作。**

#### 中控机与其他机器配置SSH互信

以kunlun用户在中控机执行下面的命令：

ssh-keygen -t rsa

运行该命令后，在所有需要输入的地方直接回车就可以了。成功后会在/home/kunlun/目录下的.ssh目录中生成 id\_rsa 和 id\_rsa.pub 两个文件，其中 id\_rsa 是存放私钥的文件，而 id\_rsa.pub 则为存放公钥的文件。

将中控机的公钥复制到其他服务器上，包括中控机本机：

ssh-copy-id -i ~/.ssh/id\_rsa.pub 192.168.40.151

ssh-copy-id -i ~/.ssh/id\_rsa.pub 192.168.40.152

ssh-copy-id -i ~/.ssh/id\_rsa.pub 192.168.40.153

验证中控机上执行命令验证kunlun用户到其他机器上的免密登陆:

ssh 192.168.40.151 date

ssh 192.168.40.152 date

ssh 192.168.40.153 date

#### 安装python2,git和wget等工具

使用root执行下面的命令：

yum -y install python git wget yum-utils

#### 安装docker服务

由于计划在中控上部署XPanel，需要提前安装和部署docker。

使用root执行下面的命令：

清除系统自带的docker相关的rpm：

yum remove docker \

docker-client \

docker-client-latest \

docker-common \

docker-latest \

docker-latest-logrotate \

docker-logrotate \

docker-engine

安装docker社区版：

yum -y install docker

启动docker，并设置其随操作系统自启动：

systemctl start docker

systemctl enable docker

#### 上传安装工具及安装包到服务器

**通过在线环境下载并打包最新部署工具和安装包，或者通过Klustron团队咨询后下载对应的离线部署包**

在线下载并打包（**这里以v1.2.2版本为例，对应的版本下载以实际的为准**）：

将部署工具下载到/data1/softwares目录下，使用root用户执行：

cd /data1/softwares

git clone -b 1.1 https://gitee.com/zettadb/cloudnative.git

#### 下载安装需要的软件包

使用root执行下面的命令：

cd /data1/softwares/setuptools-36.5.0/cloudnative/cluster/clustermgr

wget http://zettatech.tpddns.cn:14000/thirdparty/hadoop-3.3.1.tar.gz

wget http://zettatech.tpddns.cn:14000/thirdparty/jdk-8u131-linux-x64.tar.gz

wget http://zettatech.tpddns.cn:14000/thirdparty/mysql-connector-python-2.1.3.tar.gz

wget http://zettatech.tpddns.cn:14000/thirdparty/prometheus.tgz

wget http://zettatech.tpddns.cn:14000/thirdparty/haproxy-2.5.0-bin.tar.gz

wget http://zettatech.tpddns.cn:14000/thirdparty/efk/filebeat-7.10.1-linux-x86\_64.tar.gz

VERSION=1.2.2

wget http://downloads.Klustron.com//releases/$VERSION/release-binaries/kunlun-cluster-manager-$VERSION.tgz

wget http://downloads.Klustron.com//releases/$VERSION/release-binaries/kunlun-node-manager-$VERSION.tgz

wget http://downloads.Klustron.com//releases/$VERSION/release-binaries/kunlun-server-$VERSION.tgz

wget http://downloads.Klustron.com//releases/$VERSION/release-binaries/kunlun-storage-$VERSION.tgz

wget http://downloads.Klustron.com//releases/$VERSION/release-binaries/kunlun-proxysql-$VERSION.tgz

chown -R kunlun:kunlun /data1/softwares/

### 部署安装

#### 编辑部署拓扑json文件

中控机上使用kunlun用户修改文件/data1/softwares/setuptools-36.5.0/cloudnative/cluster/cluster\_and\_node\_mgr.json ，加入相关的配置信息，这里以192.168.40.151（中控机），192.168.40.152，192.168.40.153三台机器为例。

与在线部署的不同之处在于XPanel部分，离线部署没有"image"属性，只有"imageType": "file"属性。

{

"machines":[

{

"ip":"192.168.40.151",

"basedir":"/data1/kunlun",

"user":"kunlun"

},

{

"ip":"192.168.40.152",

"basedir":"/data1/kunlun",

"user":"kunlun"

},

{

"ip":"192.168.40.153",

"basedir":"/data1/kunlun",

"user":"kunlun"

}

],

"meta":{

"ha\_mode": "rbr",

"nodes":[

{

"ip":"192.168.40.151"

},

{

"ip":"192.168.40.152"

},

{

"ip":"192.168.40.153"

}

]

},

"cluster\_manager": {

"nodes": [

{

"ip": "192.168.40.151"

},

{

"ip": "192.168.40.152"

},

{

"ip": "192.168.40.153"

}

]

},

"node\_manager": {

"nodes": [

{

"ip": "192.168.40.151"

},

{

"ip": "192.168.40.152"

},

{

"ip": "192.168.40.153"

}

]

},

"xpanel": {

"ip": "192.168.40.151",

"imageType": "file"

}

}

#### 执行安装

生成安装脚本

中控机使用kunlun用户执行下面的命令：

cd /data1/softwares/setuptools-36.5.0/cloudnative/cluster

sudo python2 setup\_cluster\_manager.py --autostart --config=cluster\_and\_node\_mgr.json --product\_version=$version --action=install

# **$version** 是指需要安装的集群版本号，例如：1.2.2

示例：

sudo python2 setup\_cluster\_manager.py --autostart --config=cluster\_and\_node\_mgr.json --product\_version=1.2.2 --action=install

运行安装脚本

中控机使用kunlun用户执行下面的命令：

bash -e /data1/softwares/setuptools-36.5.0/cloudnative/cluster/clustermgr/install.sh

#### 登陆XPanel进行集群创建

浏览器上登陆XPanel控制台，在能访问192.168.40.151的机器上打开浏览器，输入地址：http://192.168.40.151:18080/KunlunXPanel

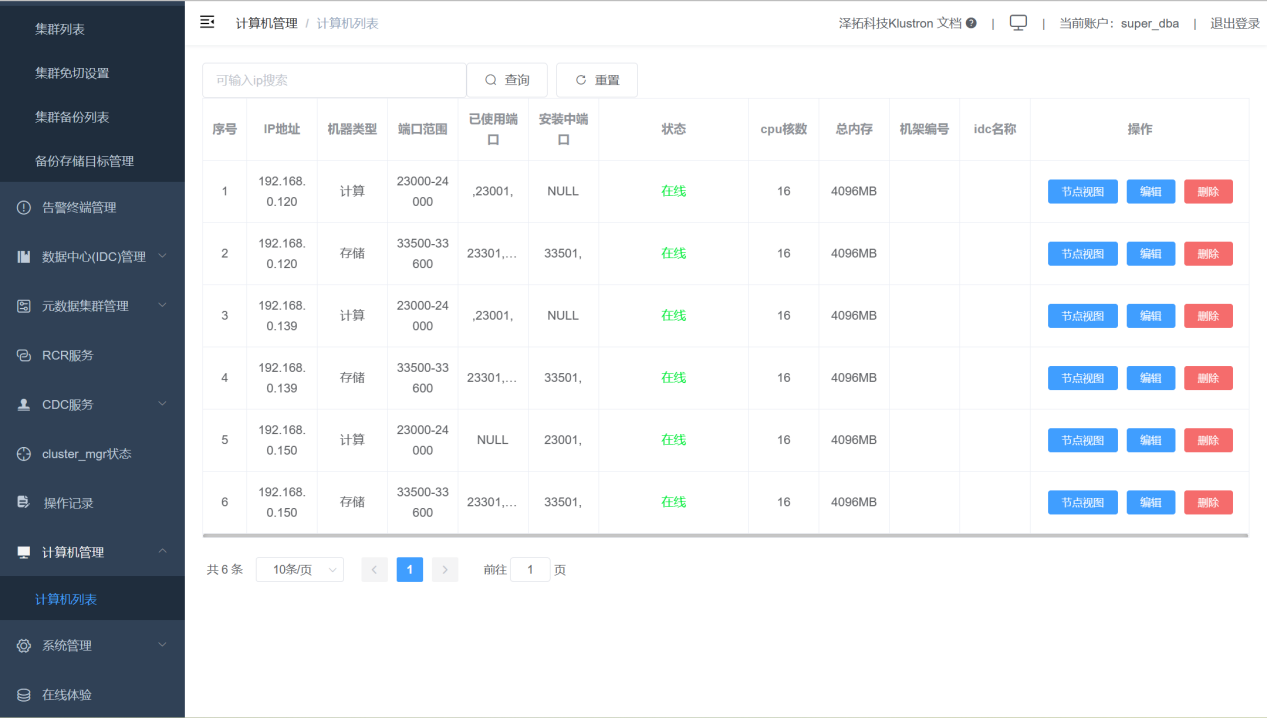
初次登录用户名和密码是：super\_dba/super\_dba，初次登陆需要修改super\_dba密码。

登录后，首页显示如下：



点击左边控制栏“计算机管理”“计算机列表”，显示下面的界面。

检查所有的机器处于在线状态：



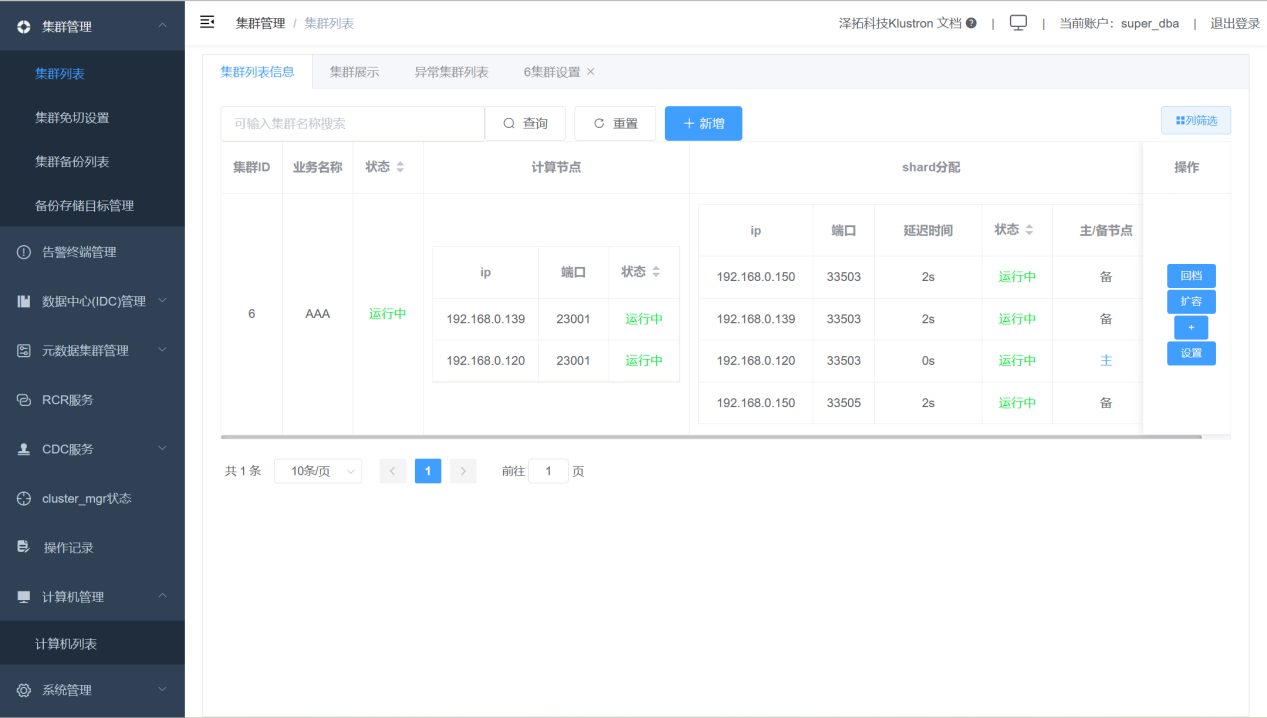
增加集群。

点击“集群管理”，“集群列表”，再点击“新增”按钮



添加集群参数如下图：





完成集群安装后，通过连接计算节点的ip和port，默认用户名和密码都是abc，就可以使用Klustron Cluster了，登录后用户根据需求进行用户的创建和密码的修改。

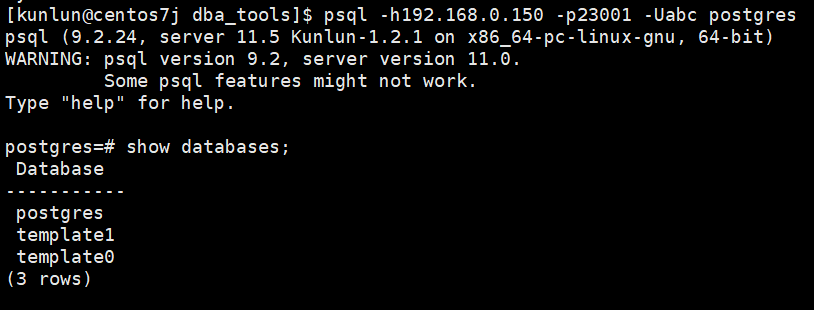
#### pg和mysql协议登录验证

**Tips：postgresql协议和mysql协议登录访问的端口号不同，具体见XPanel中【集群列表】->【设置】->【计算节点列表】。**

Postgresql登录验证：

psql -h192.168.0.150 -p23001 -Uabc postgres

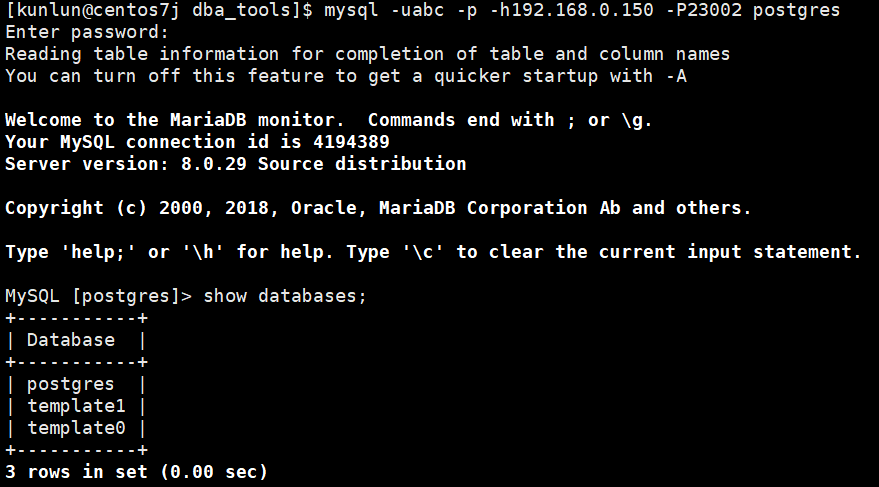
如图：



mysql登录验证：

mysql -uabc -p -h192.168.0.150 -P23002 postgres

如图：



## 其他组件部署

### HDFS部署

#### HDFS单节点部署

#### HDFS集群部署

# 集群运维操作

## 集群各组件启停

### 计算节点启停

查看计算节点进程：

ps -ef|grep $port # **$port**是指**计算节点**的端口号

示例：

ps -ef|grep 23001

到计算节点的安装目录中与bin目录同级的scripts目录下：

cd /$install\_dir/instance\_binaries/computer/$port/kunlun-server-$version/scripts

# **$install\_dir**是指计算节点安装目录

# **$port**是指计算节点端口号

# **$version** 是指计算节点的版本号，例如：1.2.1

示例：

cd /data/ha/kunlun/instance\_binaries/computer/23001/kunlun-server-1.2.1/scripts

启动：

python2 start\_pg.py --port=$port

示例：

python2 start\_pg.py --port=23001

停止：

python2 stop\_pg.py --port=$port

示例：

python2 stop\_pg.py --port=23001

### 存储节点启停

查看存储节点进程：

ps -ef|grep $port # **$port**是指**存储节点**的端口号

示例：

ps -ef|grep 33503

到存储节点的安装目录中与bin目录同级的scripts目录下：

cd/$install\_dir/instance\_binaries/storage/$port/kunlun-storage-$version/dba\_tools

# **$install\_dir**是指存储节点安装目录

# **$port**是指存储节点端口号

# **$version** 是指存储节点的版本号，例如：1.2.1

示例：

cd /data/ha/kunlun/instance\_binaries/storage/33503/kunlun-storage-1.2.1/dba\_tools

启动：

./startmysql.sh $port

示例：

./startmysql.sh 33503

停止：

./stopmysql.sh $port

示例：

./stopmysql.sh 33503

### cluster\_mgr节点启停

查看cluster\_mgr进程：

ps -ef|grep cluster\_mgr

到cluster\_mgr的安装目录中与conf目录同级的bin目录下：

cd /$install\_dir/kunlun-cluster-manager-$version/conf/cluster\_mgr.cnf

# **$install\_dir**是指cluster\_mgr安装目录

# **$version** 是指计算节点的版本号，例如：1.2.1

示例：

cd /data/ha/kunlun/kunlun-cluster-manager-1.2.1/bin

启动：

./start\_cluster\_mgr.sh

停止：

./stop\_cluster\_mgr.sh

### node\_mgr节点启停

查看node\_mgr进程：

ps -ef|grep node\_mgr

到node\_mgr的安装目录中与conf目录同级的bin目录下：

cd /$install\_dir/kunlun-node-manager-$version/conf/

# **$install\_dir**是指node\_mgr安装目录

# **$version** 是指计算节点的版本号，例如：1.2.1

示例：

cd /data/ha/kunlun/kunlun-node-manager-1.2.1/bin

启动：

./start\_node\_mgr.sh

停止：

./stop\_node\_mgr.sh

### 元数据节点启停

**Tips：元数据集群的启停方式和存储节点的启停方式相同，但需要注意修改对应安装目录和对应的端口号。**

### XPanel启停

查看xpanel进程：

sudo docker ps -a

**Tips:**查看xpanel进程的状态：**Exited**表示进程停止，**Up**表示进程存活。

启动：

sudo docker container start $xpanel\_service\_name

# **$xpanel\_service\_name** 是指xpanel服务名，由xpanel\_**$xpanel-service-p**# **ort**组成，例如xpanel\_10024

示例：

sudo docker container start xpanel\_10024

停止：

sudo docker container stop $xpanel\_service\_name

示例：

sudo docker container stop xpanel\_10024

### 扩缩机器节点

#### 扩容机器节点

扩容机器节点用作于什么组件，由配置参数nodetype决定，nodetype: 用于指定机器的用途，字符串类型，当前有 4 个可选值 'none', 'storage', 'server', 'both', 分别表示该机器将不部署任何节点，用于部署存储节点，用于部署计算节点，以及用于部署存储和计算节点。值默认为 both。

编辑新的json文件，例如add\_node.json文件，在中控机/data1/softwares/setuptools-36.5.0/cloudnative/cluster目录下编辑文件，配置的内容样例如下：

{

"machines":[

{

"ip":"192.168.0.132",

"basedir":"/data1/kunlun",

"user":"kunlun"

},

{

"ip":"192.168.0.134",

"basedir":"/data1/kunlun",

"user":"kunlun"

}

],

"meta":{

"group\_seeds": "192.168.0.110:6001,192.168.0.111:6002,192.168.0.100:6003"

},

"node\_manager": {

"nodes": [

{

"ip": "192.168.0.132"

},

{

"ip": "192.168.0.134"

}

]

}

}

以上配置样例中，machines表示新增扩容的机器信息；meta表示当前元数据集群ip和port；node\_manager表示执行cluster\_mgr下发命令的组件，每台机器都需要部署。

在机器扩容前，新增的机器需要按照3.2.1章节“部署前准备工作-所有机器操作”和3.2.2章节中的第一个步骤“中控机与其他机器配置SSH互信”。

执行机器扩容

原集群中控机上，kunlun用户执行命令：

cd /data1/softwares/setuptools-36.5.0/cloudnative/cluster

sudo python2 setup\_cluster\_manager.py --config=add\_node.json --action=install

bash -e clustermgr/install.sh

扩容机器完成后，可以在XPanel的“计算机管理”->“计算机列表”查看到新扩容的机器。如下图片中红框的就是新增的机器。



#### 缩容机器节点

### 摧毁所有集群

**高危警告！！！再三确认所有集群可以摧毁后，方可执行，此操作不可逆！！！**

中控机上执行摧毁所有集群命令：

cd /data1/softwares/setuptools-36.5.0/cloudnative/cluster

python setup\_cluster\_manager.py --autostart --config=$config\_file --product\_version=$version --action=clean

bash /data1/softwares/setuptools-36.5.0/cloudnative/cluster/clustermgr/clean.sh

# **$config\_file** 是指Klurton Cluster的拓扑文件，文件以.json格式结尾。

# **$version**是指当前集群的版本号，例如：1.2.1

示例：

python setup\_cluster\_manager.py --autostart --config=cluster\_and\_node\_mgr.json --product\_version=1.2.1 --action=clean

bash /data1/softwares/setuptools-36.5.0/cloudnative/cluster/clustermgr/clean.sh

## 各组件日志目录和配置文件目录

### 日志目录

**Tips：klustron集群中各个模块的端口号都可以在XPanel上进行查看。**

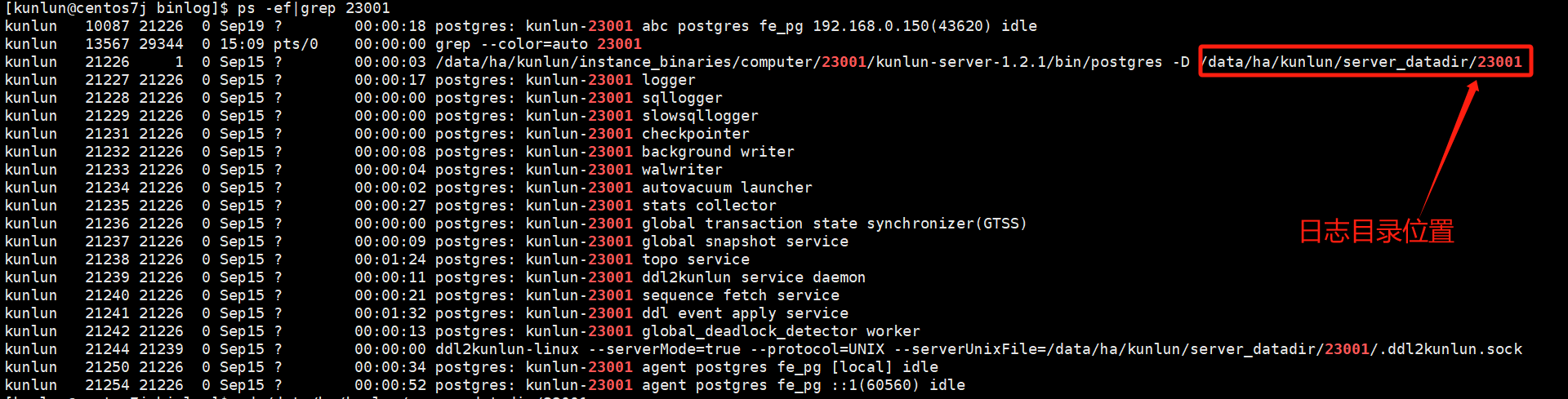
* 计算节点kunlun-server

通过如下命令查看计算节点kunlun-server日志目录：

ps -ef|grep 23001

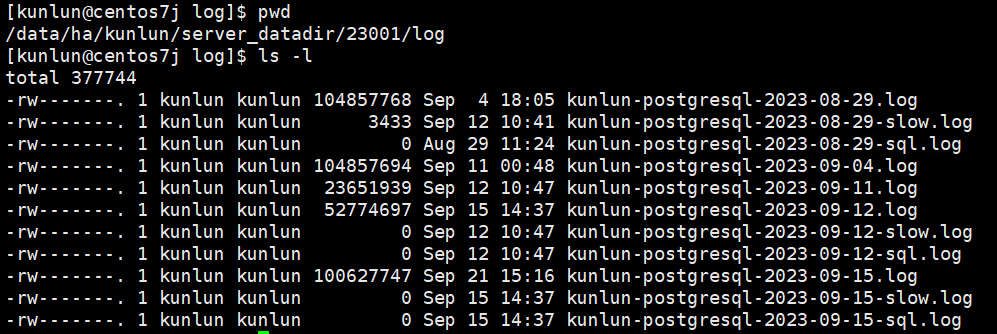
#这里以23001计算节点端口号为例

如图：



cd /data/ha/kunlun/server\_datadir/23001/log

如图：



日志文件分类：

* kunlun-postgresql-\*.log 记录的是计算节点运行日志文件，记录计算节点运行期间所接收到SQL语句与计算节点程序本身的日志、ERROR、WARNING等信息。
* kunlun-postgresql-\*-slow.log 记录的是计算节点出现的慢SQL日志，计算节点上慢SQL由变量log\_min\_duration\_statement控制阈值大小，默认值10，单位s，超过10s的sql会记录到该日志文件中。
* kunlun-postgresql-\*-sql.log 记录的是从计算节点发给存储节点的SQL语句，由变量enable\_sql\_log控制打开或者关闭该日志，默认值off，处于关闭。
* 存储节点storage-server

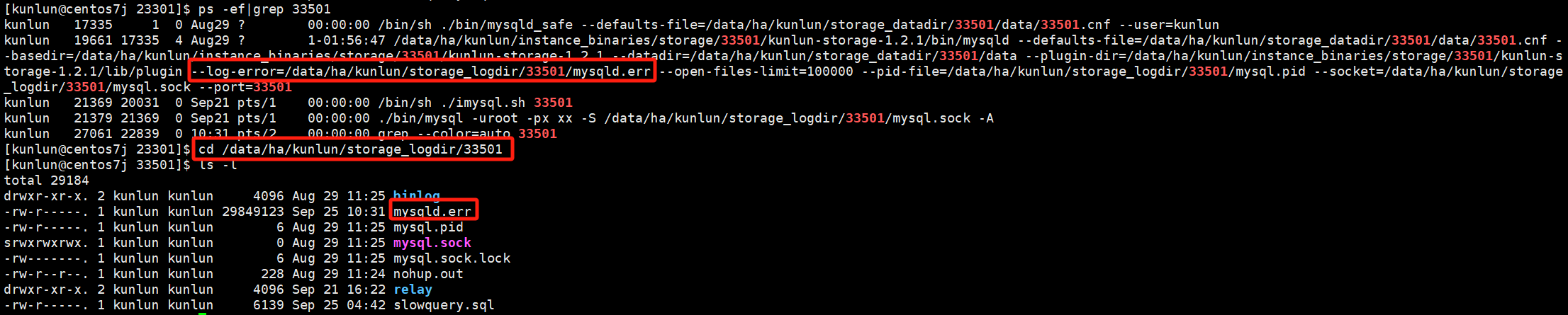
通过如下命令查看存储节点storage-server日志目录：

ps -ef|grep 33501

#这里以33501存储节点端口号为例

cd /data/ha/kunlun/storage\_logdir/33501

如图：



主要查看的是mysqld.err日志，该日志记录存储节点运行信息、Warning、ERROR等信息。

* cluster\_mgr节点

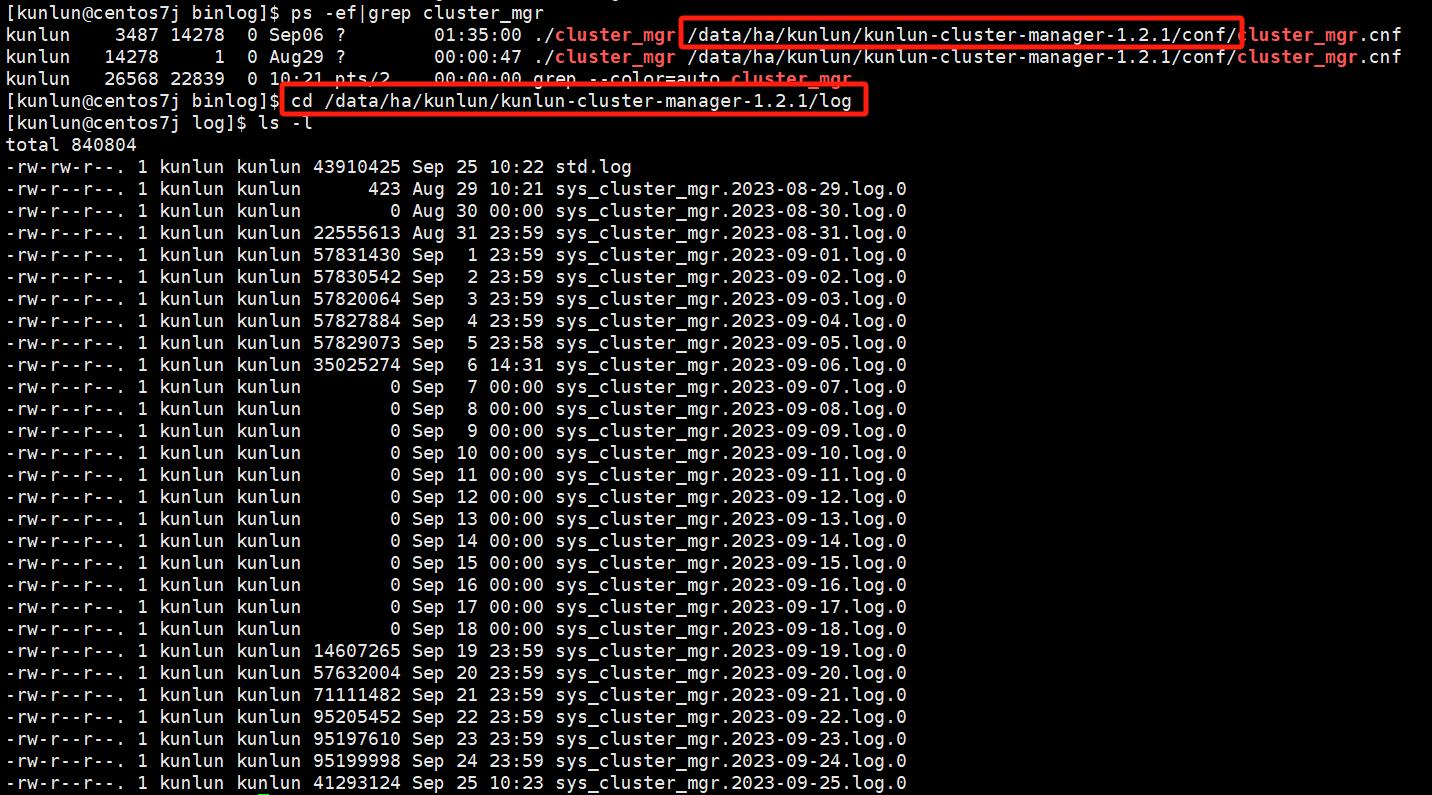
通过如下命令查看cluster\_mgr节点日志目录：

ps -ef|grep cluster\_mgr

log目录与conf目录同级

cd /data/ha/kunlun/kunlun-cluster-manager-1.2.1/log

如图：



主要查看的是sys\_cluster\_mgr\*.log.\*日志，记录的是cluster\_mgr运行期间打印的任务，管理调度，错误等信息。

* 元数据集群

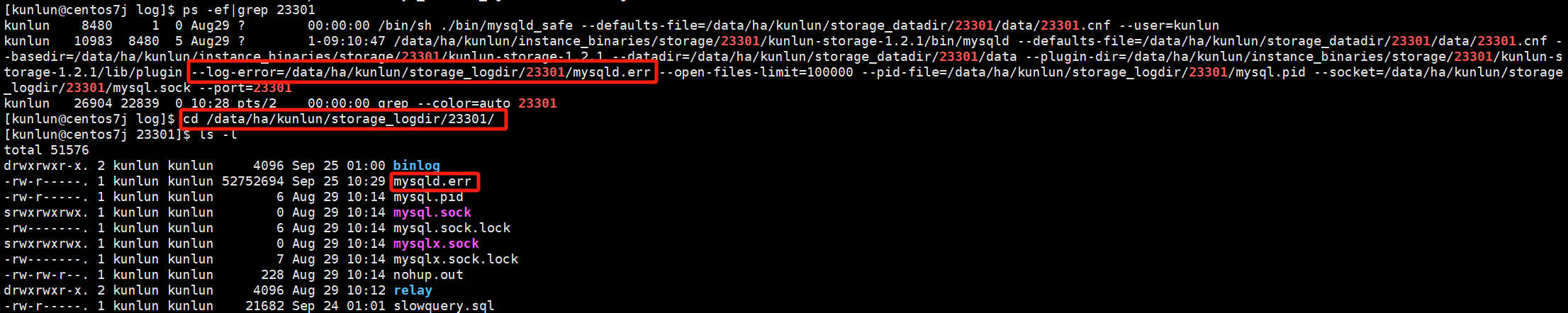
通过如下命令查看元数据节点日志目录：

ps -ef|grep 23301

#这里以23301元数据节点端口号为例

cd /data/ha/kunlun/storage\_logdir/23301/

如图：



主要查看的是mysqld.err日志，该日志记录元数据节点运行信息、Warning、ERROR等信息。

* node\_mgr节点

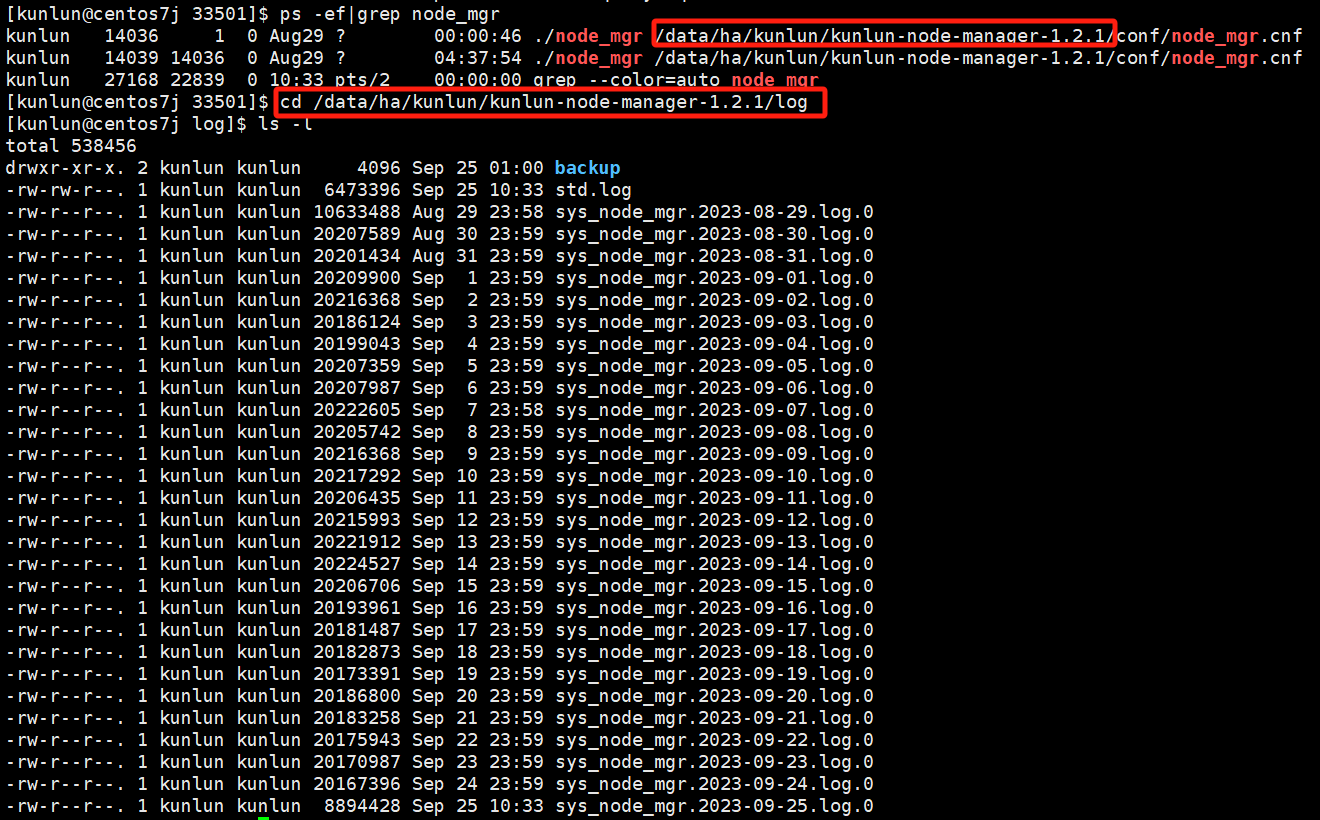
通过如下命令查看node\_mgr节点日志目录：

ps -ef|grep node\_mgr

log目录与conf目录同级

cd /data/ha/kunlun/kunlun-node-manager-1.2.1/log

如图：



主要查看的是sys\_node\_mgr.\*.log.\*日志，该日志记录着cluster\_mgr下发的任务让node\_mgr执行的相关shell命令，例如：集群安装，扩缩容节点，备份恢复，主备切换等任务对应的shell命令。

### 修改配置模板

在创建集群前如有特殊需求，可以在配置模板中修改对应的配置，最后再创建集群。

* **计算节点**
* **环境部署前**

修改步骤如下：

到部署工具目录下，以Klustron1.2.1集群版本命令为例：

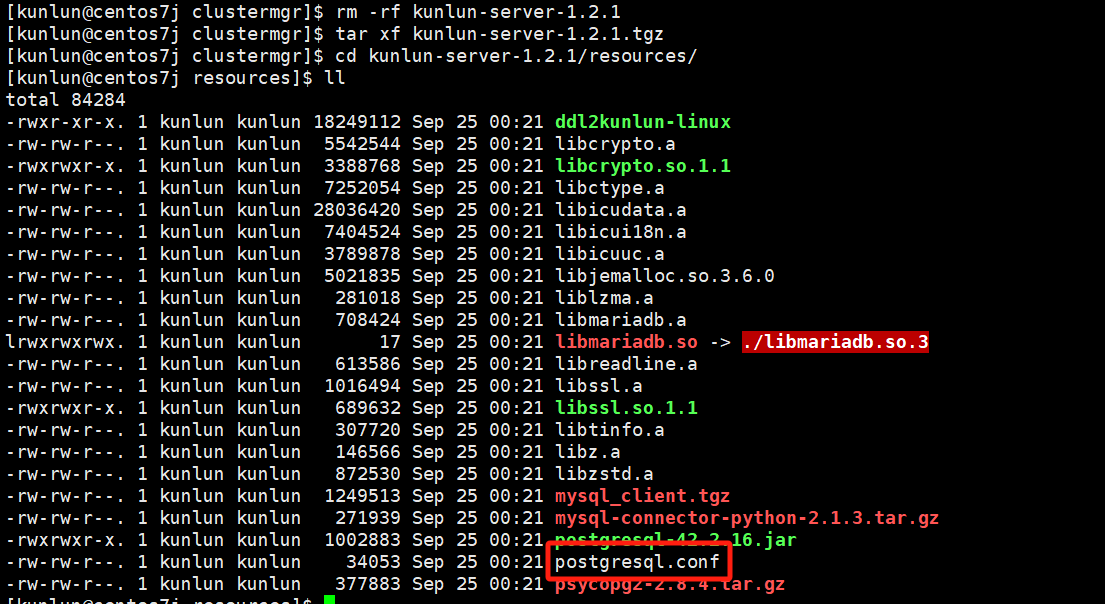
cd setuptools-36.5.0/cloudnative/cluster/clustermgr

tar xf kunlun-server-1.2.1.tgz

cd kunlun-server-1.2.1/resources

vim postgresql.conf

如图：



编辑配置模板文件，修改对应的配置值即可。**请注意！！！修改配置值之前可以与Klustron数据库团队确认是否可以更改，以免配置失败影响使用。**

最后打包压缩：

mv kunlun-server-1.2.1.tgz kunlun-server-1.2.1\_bak.tgz

tar czf kunlun-server-1.2.1.tgz kunlun-server-1.2.1/

再按照部署步骤完成部署。

* **环境部署后**

Klustron的环境已经部署完成，使再次创建集群时按照新的配置文件进行配置。

修改步骤如下：

cd $base\_dir/program\_binaries/kunlun-server-1.2.1/resources

vim postgresql.conf

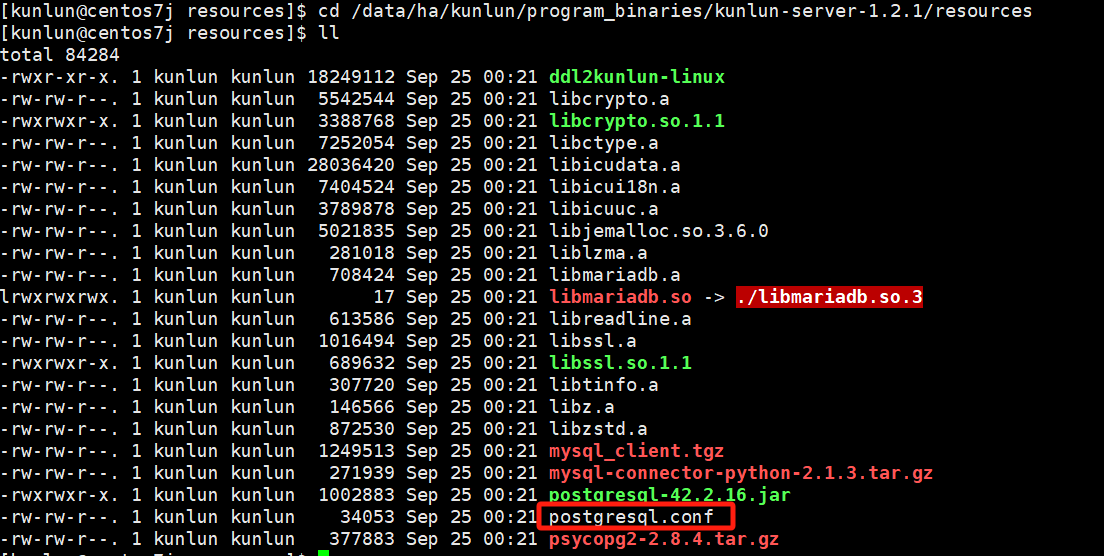
$base\_dir是指安装基础目录，可以在部署时的拓扑文件.json中查看。

以Klustron1.2.1集群版本命令为例：

cd /data/ha/kunlun/program\_binaries/kunlun-server-1.2.1/resources

vim postgresql.conf

如图：



编辑配置模板文件，修改对应的配置值即可。**请注意！！！修改配置值之前可以与Klustron数据库团队确认是否可以更改，以免配置失败影响使用。**

* **存储节点**
* **环境部署前**

修改步骤：

到部署工具目录下，以Klustron1.2.1集群版本命令为例：

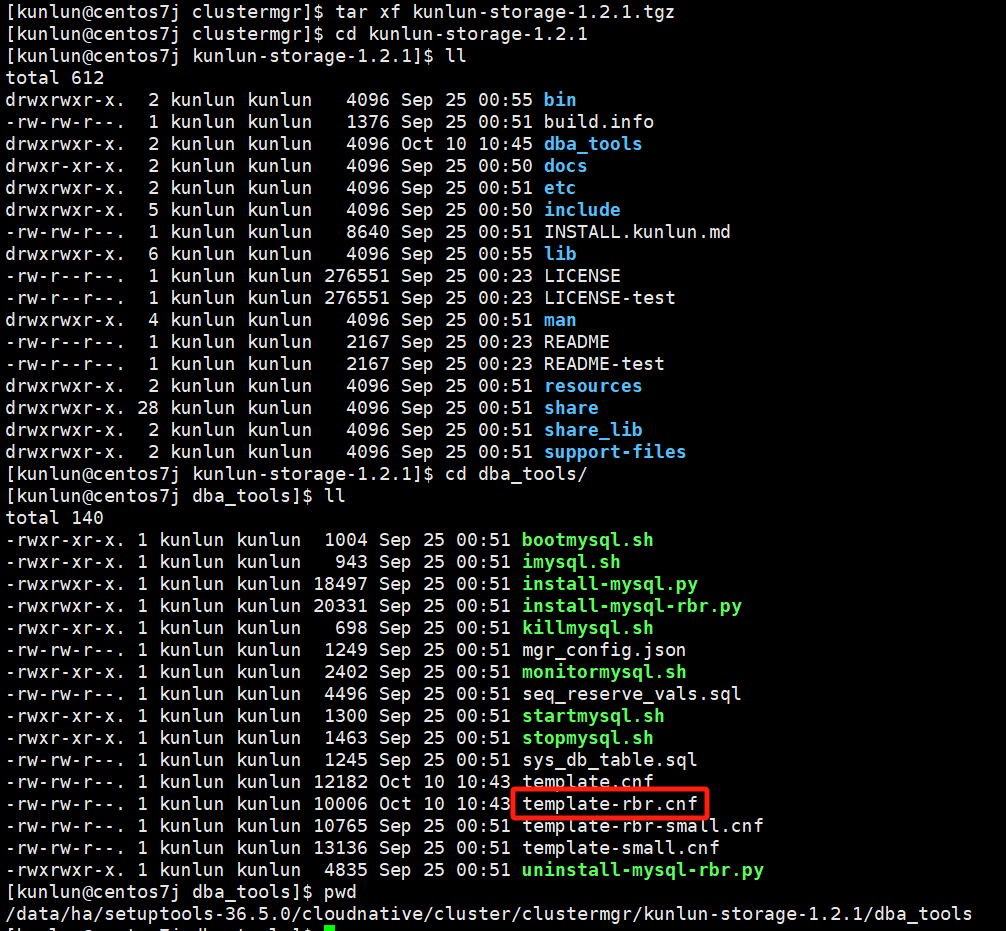
cd setuptools-36.5.0/cloudnative/cluster/clustermgr

tar xf kunlun-storage-1.2.1.tgz

cd kunlun-storage-1.2.1/dba\_tools

vim template-rbr.cnf

如图：



编辑配置模板文件，修改对应的配置值即可。**请注意！！！修改配置值之前可以与Klustron数据库团队确认是否可以更改，以免配置失败影响使用。**

最后打包压缩：

mv kunlun-storage-1.2.1.tgz kunlun-storage-1.2.1\_bak.tgz

tar czf kunlun-storage-1.2.1.tgz kunlun-storage-1.2.1/

* **环境部署后**

Klustron的环境已经部署完成，使再次创建集群时按照新的配置文件进行配置。

修改步骤

cd $base\_dir/program\_binaries/kunlun-storage-1.2.1/dba\_tools

vim template-rbr.cnf

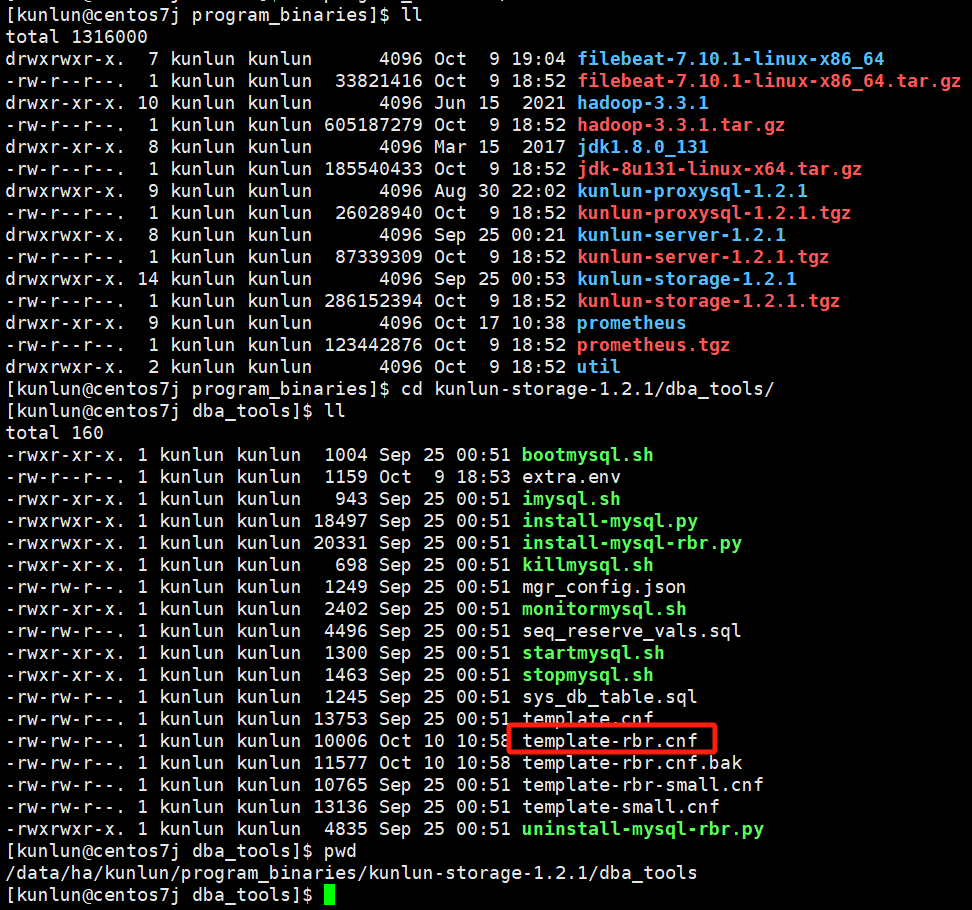
$base\_dir是指安装基础目录，可以在部署时的拓扑文件.json中查看。

以Klustron1.2.1集群版本命令为例：

cd /data/ha/kunlun/program\_binaries/kunlun-storage-1.2.1/dba\_tools

vim template-rbr.cnf

如图：



编辑配置模板文件，修改对应的配置值即可。**请注意！！！修改配置值之前可以与Klustron数据库团队确认是否可以更改，以免配置失败影响使用。**

### 配置文件目录与生效

**注意：众多情况下，计算节点、存储节点、元数据集群、cluster\_mgr、node\_mgr的配置文件一般情况下用户不需要去修改，除特殊情况外，这里只是介绍配置文件所在路径，以供特殊情况使用！！！**

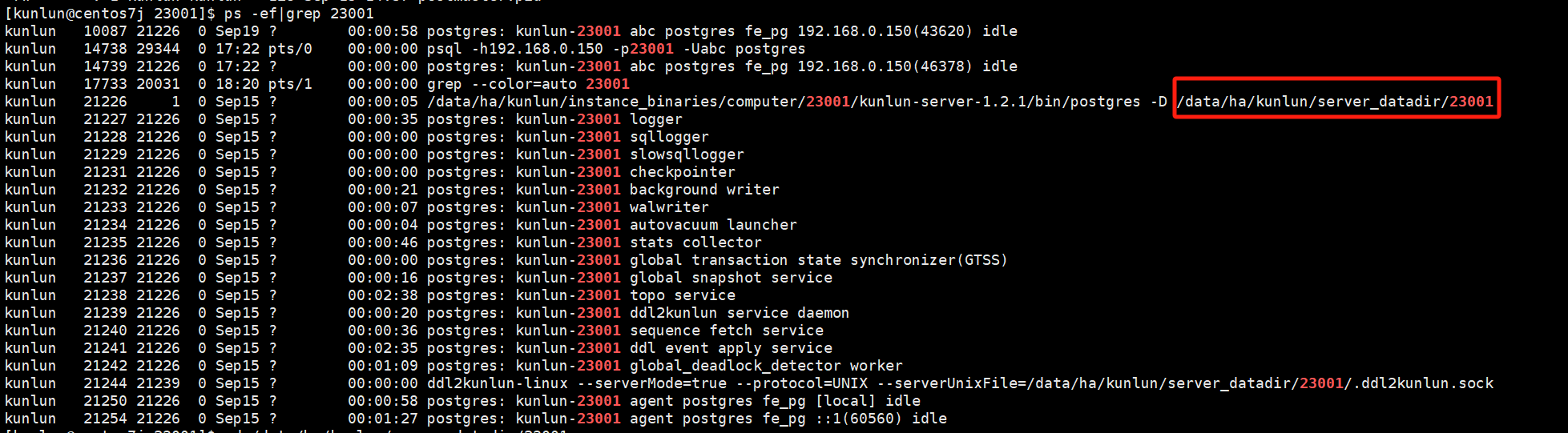
* 计算节点配置文件目录

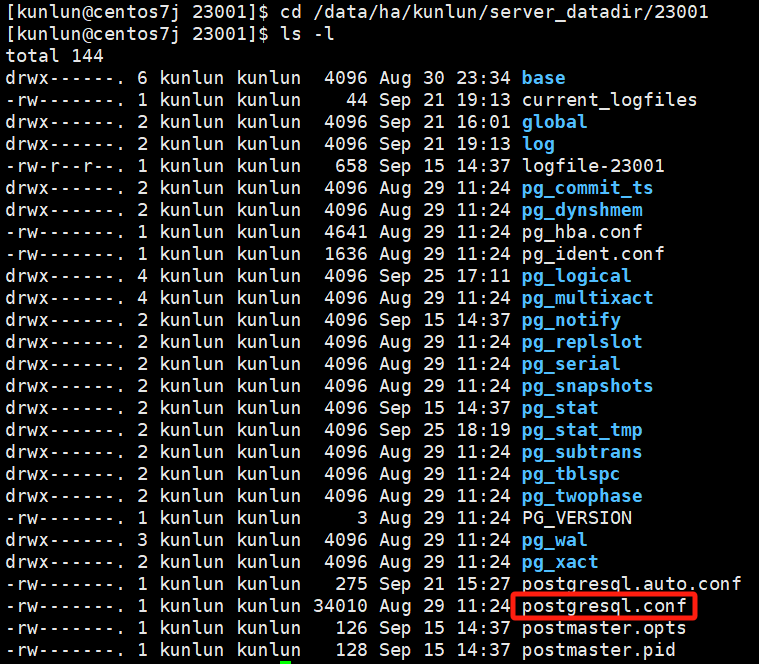
通过如下命令查看计算节点kunlun-server日志目录：

ps -ef|grep 23001

#这里以23001计算节点端口号为例

如图：





在计算节点中分为多种类型的系统变量，不同的系统变量生效的方式不同，如需修改系统变量并使其生效，见如下链接中计算节点部分内容：

<http://doc.klustron.com/zh/Klustron_system_variable.html>

* 存储节点配置文件

通过如下命令查看存储节点storage-server日志目录：

ps -ef|grep 33501

#这里以33501存储节点端口号为例

如图：



存储节点中如需修改系统变量并使其生效，见如下链接中存储节点部分内容：

<http://doc.klustron.com/zh/Klustron_system_variable.html>

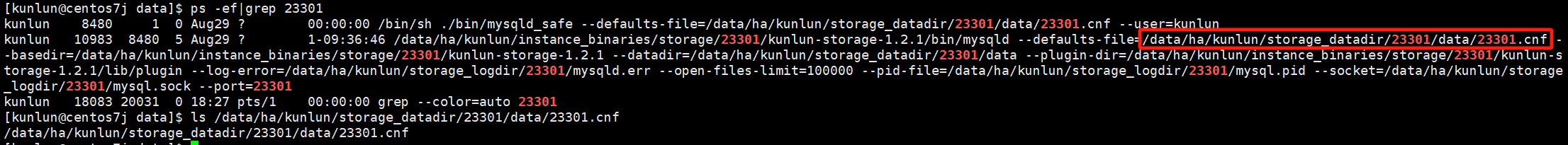
* 元数据节点配置文件

通过如下命令查看元数据节点日志目录：

ps -ef|grep 23301

#这里以23301元数据节点端口号为例

如图：

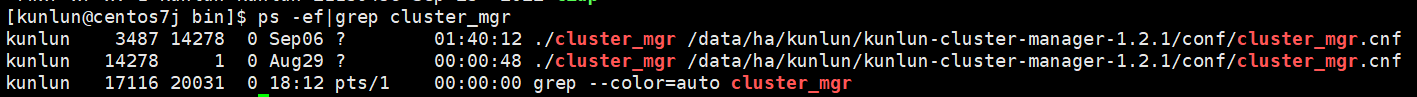


* cluster\_mgr节点配置文件目录

通过如下命令查看cluster\_mgr节点配置文件目录：

ps -ef|grep cluster\_mgr

如图：



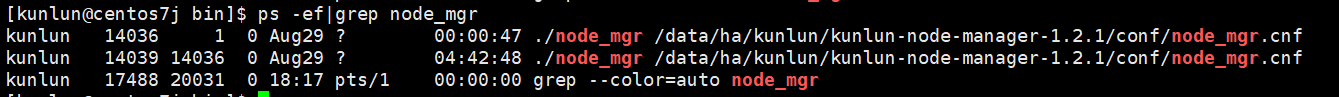
修改配置文件后，生效方式需要重启cluster\_mgr，参考[4.1.3章节](#_cluster_mgr节点启停)内容。

* node\_mgr节点配置文件目录

通过如下命令查看node\_mgr节点日志目录：

ps -ef|grep node\_mgr

如图：



### 服务器角色等信息

通过登录元数据集群获取相关信息

## 负载均衡配置

Klustron不提供负载均衡组件，由用户自己按需提供使用。负载均衡配置方法如下：在集群创建完成后，用户通过将计算节点的ip和port配置到负载均衡中，即可完成并可以连接负载均衡提供的ip进行访问Klustron。

## 导入导出工具

Klustron支持copy命令进行导入导出，也支持pg\_dump和pg\_restore，还有myloader和mydumper工具进行数据导入导出。

#### COPY命令

* copy命令方式

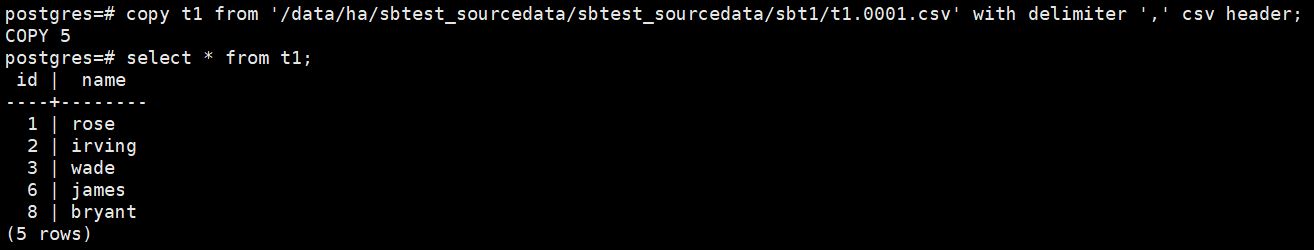
通过postgresql客户端登录后，执行命令。

**Tips：导入之前需要已有表结构或者先导入表结构。**

**导入：忽略首行，列间隔用“,”表示的csv文件**

copy t1 from '/data/ha/sbtest\_sourcedata/sbtest\_sourcedata/sbt1/t1.0001.csv' with delimiter ',' csv header;

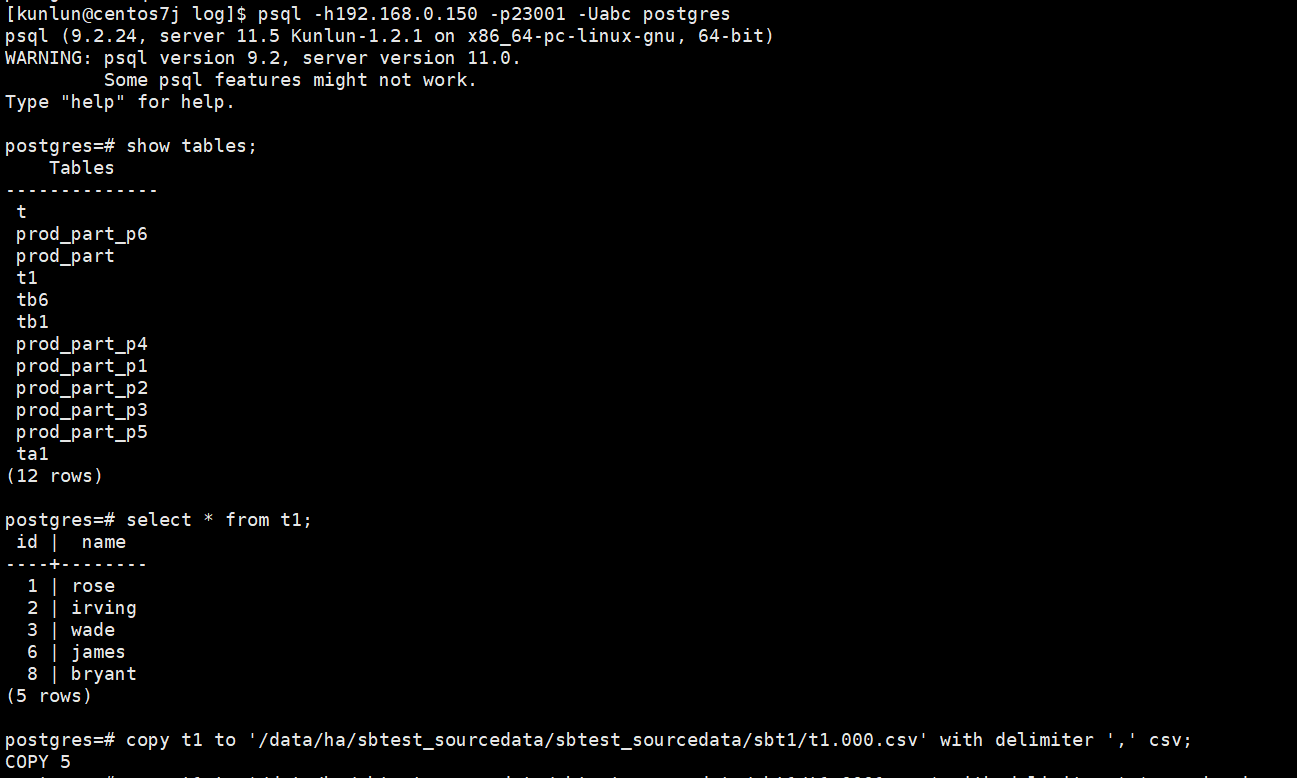
如图：



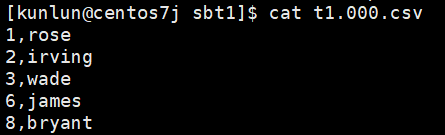
**导出：整表数据，不带表头**

copy t1 to '/data/ha/sbtest\_sourcedata/sbtest\_sourcedata/sbt1/t1.000.csv' with delimiter ',' csv;

如图：



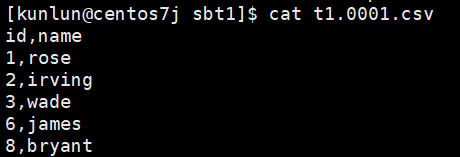
导出的CSV文件示例：



**导出：整表数据，带表头**

copy t1 to '/data/ha/sbtest\_sourcedata/sbtest\_sourcedata/sbt1/t1.0001.csv' with delimiter ',' csv header;

导出的CSV文件示例：



Copy命令常用参数说明：

**csv ：**表示文件格式

**header ：** 表示文件第一行为字段名，导入数据库时忽略首行

**delimiter** **：**表示字段分割方式，示例是以逗号分割

**encoding ：**表示文件编码方式，示例是 GBK 编码，（默认为utf-8）

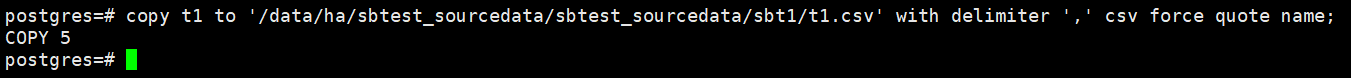
**quote ：**指定需要用什么符号来将指定列进行定界符，指定一个数据值被引用时使用的引用字符。默认是双引号。 这必须是一个单一的单字节字符。只有使用 CSV格式时才允许这个选项。

**force quote ：**根据指定的列进行定界符，使用时force quote不需要加“\_”。强制必须对每个指定列中的所有非NULL值使用引用。 NULL输出不会被引用。如果指定了\*， 所有列的非NULL值都将被引用。只有在 COPY TO中使用CSV格式时才允许这个选项。

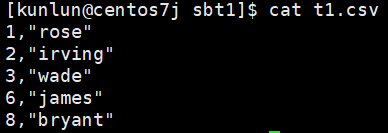
**quote**使用示例：

copy t1 to '/data/ha/sbtest\_sourcedata/sbtest\_sourcedata/sbt1/t1.csv' with delimiter ',' csv force quote name;

如图：



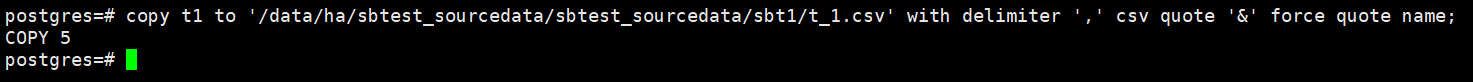
导出的CSV文件示例：



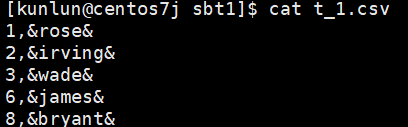
**force quote**使用示例：

copy t1 to '/data/ha/sbtest\_sourcedata/sbtest\_sourcedata/sbt1/t\_1.csv' with delimiter ',' csv quote '&' force quote name;

如图：



导出的CSV文件示例：



#### pg\_dump和pg\_restore

关于pg\_dump和pg\_restore命令参数的介绍见官网文档：

pg\_dump：

<http://www.postgres.cn/docs/11/app-pgdump.html>

pg\_restore：

<http://www.postgres.cn/docs/11/app-pgrestore.html>

如下是常用的pg\_dump和pg\_restore命令的介绍

* pg\_dump导出

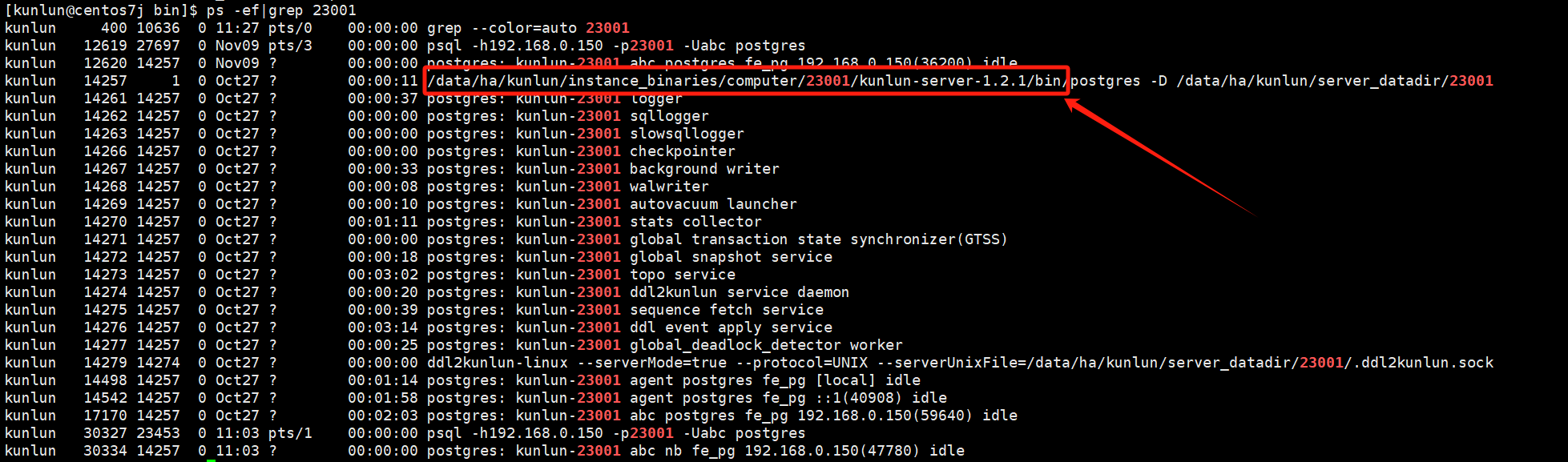
由于导入导出工具客户端需要与服务端版本匹配的原因，需要使用配套工具操作。

pg\_dump导入和pg\_restore导出工具的存放目录位于计算节点如下目录：

这里以kunlun-server的端口号**23001**为例：

ps -ef|grep 23001

图片示例：

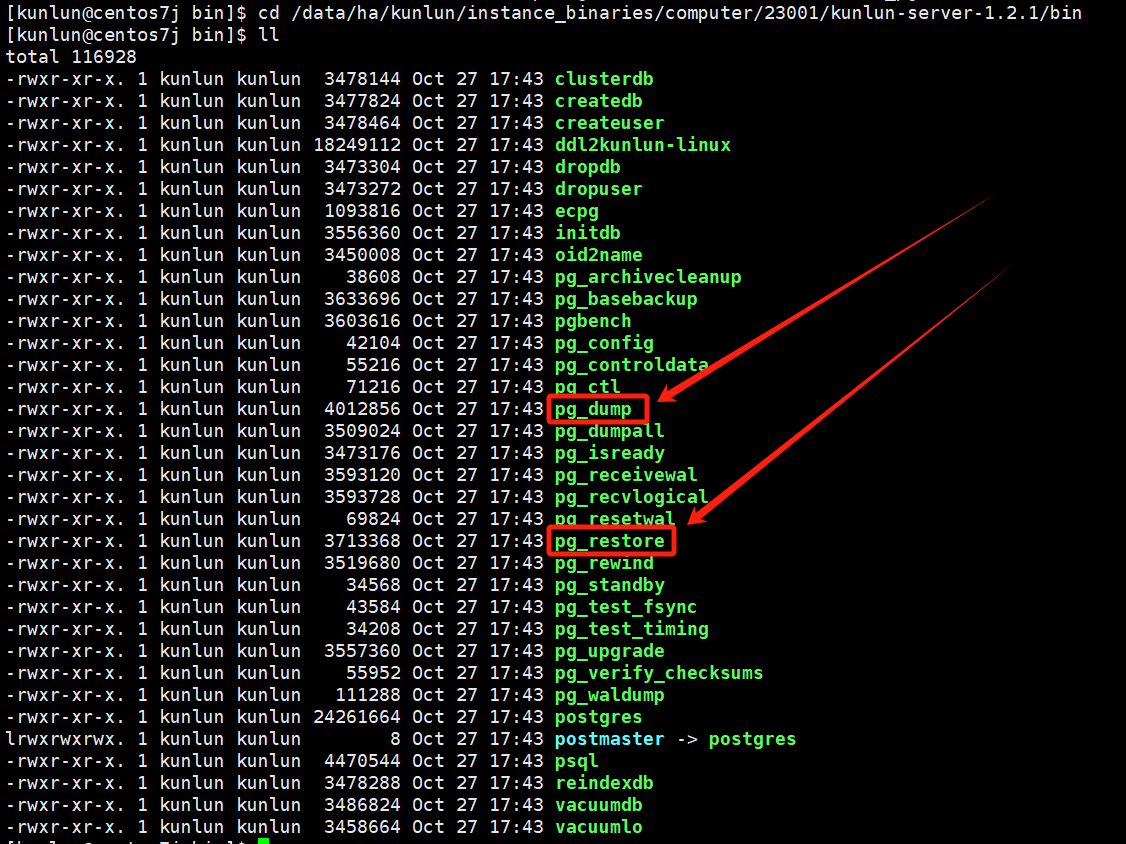


查看pg\_dump和pg\_restore所在的目录：

cd /data/ha/kunlun/instance\_binaries/computer/23001/kunlun-server-1.2.1/bin

ls -l

图片示例：



导出指定库(这里以pgdb库为例)，导出的文件为sql文本：

./pg\_dump -h192.168.0.150 -p23001 -Uabc -d pgdb > pgdb.sql

导出指定表(这里以n1表为例)，导出的文件为sql文本：

./pg\_dump -h192.168.0.150 -p23001 -Uabc -d pgdb -t n1 > n1.sql

导出指定表结构(这里以n1表为例)，导出的文件为sql文本：

./pg\_dump -h192.168.0.150 -p23001 -Uabc -d pgdb -t n1 -s > n1\_s.sql

导出指定表数据(这里以n1表为例)，导出的文件为sql文本：

./pg\_dump -h192.168.0.150 -p23001 -Uabc -d pgdb -t n1 -a > n1\_data.sql

导出多个表(这里以t1和t2表为例)，导出的文件为sql文本：

./pg\_dump -h192.168.0.150 -p23001 -Uabc -d pgdb -t t1 -t t2 > t.sql

**注意：以下格式不同于文本格式的sql文件：**

导出指定库（转储一个数据库到一个自定义格式归档文件，以nb库为例）：

./pg\_dump -h192.168.0.150 -p23001 -Uabc -d nb -Fc > nb.dump

导出指定表（转储一个数据库到一个自定义格式归档文件，以t1表为例）：

./pg\_dump -h192.168.0.150 -p23001 -Uabc -d postgres -t t1 -Fc > t1.dump

* psql导入

对于sql文本文件可以使用psql进行导入：

**注意：psql导入的前置条件**

前置条件：nb库下的表没有与sql文件中的表冲突。

导入库（以nb库为例）：

psql -h192.168.0.150 -p23001 -Uabc nb < nb.sql

前置条件：n1表不存在。

导入表（以n1表为例）：

psql -h192.168.0.150 -p23001 -Uabc nb < n1.sql

前置条件：n1表不存在

导入表结构（以n1表为例）：

psql -h192.168.0.150 -p23001 -Uabc nb < n1.sql

前置条件：n2表存在，用户决定n2表是否有约束。

导入表数据（以n2表为例）：

psql -h192.168.0.150 -p23001 -Uabc nb < n2.sql

* pg\_restore导入

**对于自定义格式归档文件，可以使用pg\_restore**

前置条件：nb库已经存在

导入库（以nb库为例，-c参数会删除nb库下面的表，然后再进行导入）：

./pg\_restore -h192.168.0.150 -p23001 -Uabc -d nb -c -e nb.dump

前置条件：nb库不存在

导入库（以nb库为例，-C参数在导入前创建好nb库）：

./pg\_restore -h192.168.0.150 -p23001 -Uabc -d postgres -C nb.dump

前置条件：在nb库中n2表不存在。

导入表结构（以n2表为例）：

./pg\_restore -h192.168.0.150 -p23001 -Uabc -d nb -e n2.dump

前置条件：n2表存在，由用户决定n2表是否有约束。

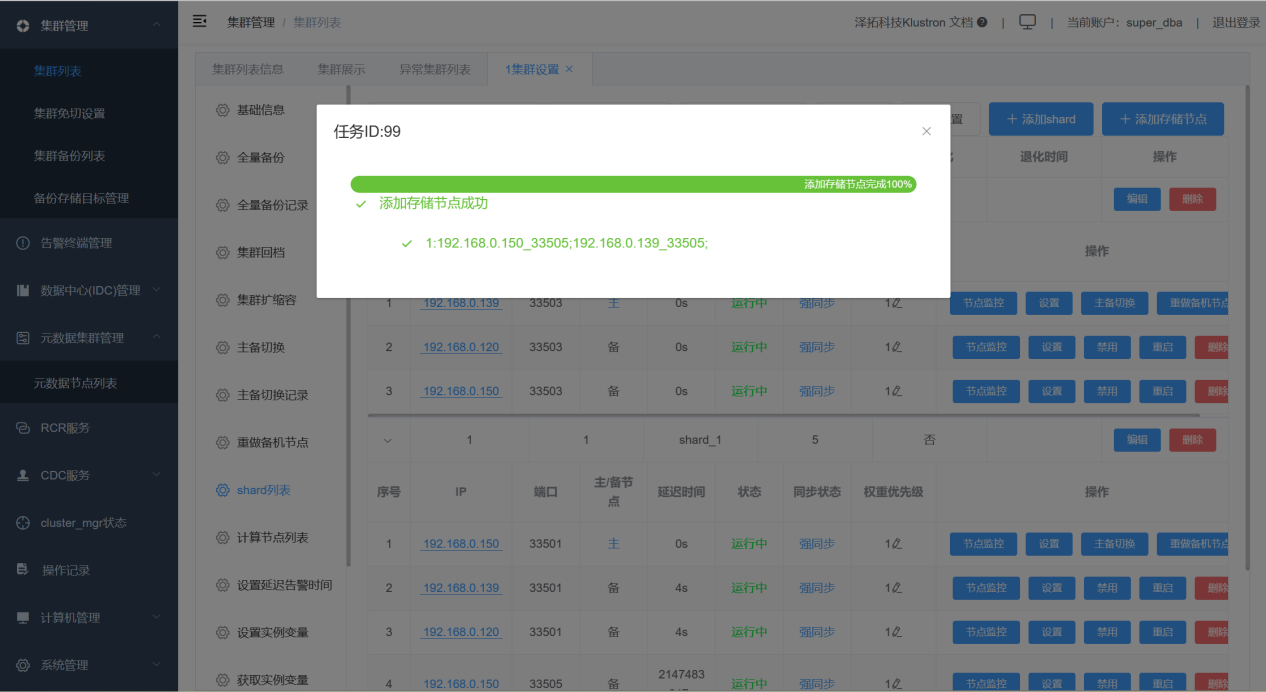
导入表数据（以n2表为例）

./pg\_restore -h192.168.0.150 -p23001 -Uabc -d nb -a -e n2.dump

* myloader导出
* mydumper导入

## 添加shard中备节点

通过XPanel上新增shard的备节点：



## 系统变量管理

如需查看Klustron特有的系统变量介绍见链接：

<http://doc.klustron.com/zh/Klustron_system_variable.html>

### enable\_global\_mvcc

在修改enable\_global\_mvcc参数时，需要一定的时间窗口进行变更操作，在变更窗口中将所有的计算节点都修改并重启后使该参数生效，验证成功后方可恢复应用访问Klustron。

修改的步骤如下：

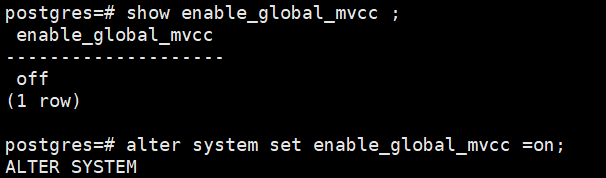
* **修改enable\_global\_mvcc参数**

使用postgresql协议登录计算节点，然后执行如下命令：

show enable\_global\_mvcc ;

alter system set enable\_global\_mvcc =on;

如图：



* **重启计算节点**

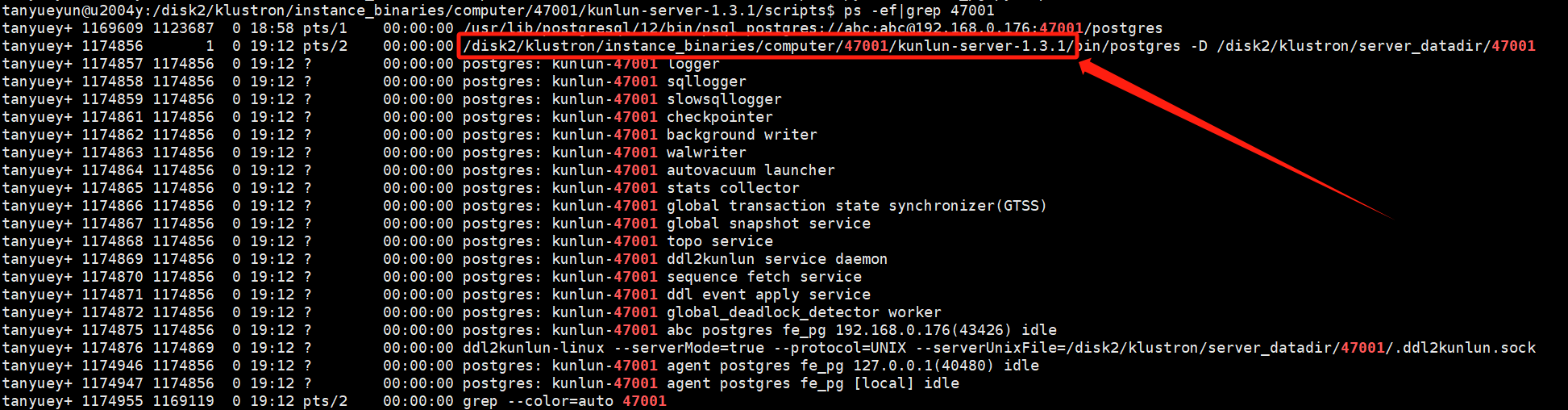
首先查看计算节点进程：

ps -ef|grep $port # **$port**是指**计算节点**的端口号

示例：

ps -ef|grep 47001

如图：



到计算节点的安装目录中与bin目录同级的scripts目录下：

cd /$install\_dir/instance\_binaries/computer/$port/kunlun-server-$version/scripts

# **$install\_dir**是指计算节点安装目录

# **$port**是指计算节点端口号

# **$version** 是指计算节点的版本号，例如：1.3.1

示例：

cd /disk2/klustron/instance\_binaries/computer/47001/kunlun-server-1.3.1/scripts

停止：

python2 stop\_pg.py --port=$port

示例：

python2 stop\_pg.py --port=47001

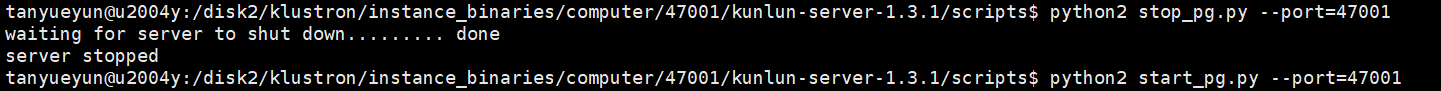
启动：

python2 start\_pg.py --port=$port

示例：

python2 start\_pg.py --port=47001

如图：

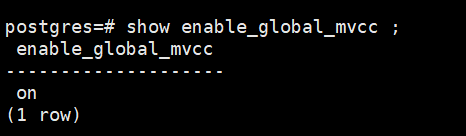


* **验证修改成功**

使用postgresql协议登录计算节点，然后执行如下命令：

show enable\_global\_mvcc ;

如图：



**注意：其余计算节点遵循如上操作步骤进行修改！！！**

## 命令行保底操作

添加计算节点

添加shard

添加备机节点

### 手动重做备机

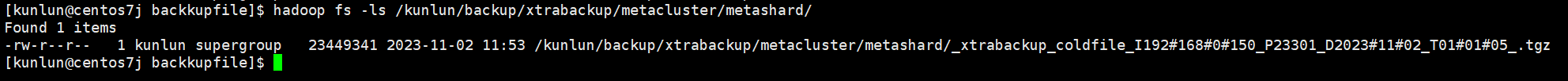
当出现XPanel无法使用，又急于故障恢复备机增加高可用性时，可以使用手动的方式进行重做备机。

如下为操作步骤，以重做元数据集群的一个备机为例。

将HDFS上的元数据集群的备份下载到本地，拷贝到对应需要重做的机器上。

查看：

hadoop fs -ls /kunlun/backup/xtrabackup/metacluster/metashard/



从HDFS上下载下来的全备文件存放到已经创建好的目录上：

示例：

mkdir -p /data/ha/backupfile/

#这里以/data/ha/backupfile/为例，用户可自定义目录名。

**特别注意：如下涉及的操作命令以实际的目录和端口号为准！！！变量只做介绍，以用户环境实际情况为准！！！**

将全备文件下载后并解压（**注意：需要将全备文件拷贝到重做备机的机器上**）：

hadoop fs -get /kunlun/backup/xtrabackup/metacluster/metashard/**$coldfile**.tgz /data/ha/backupfile/

# **$coldfile**为HDFS上的全备文件，**以实际的为准**。

示例：

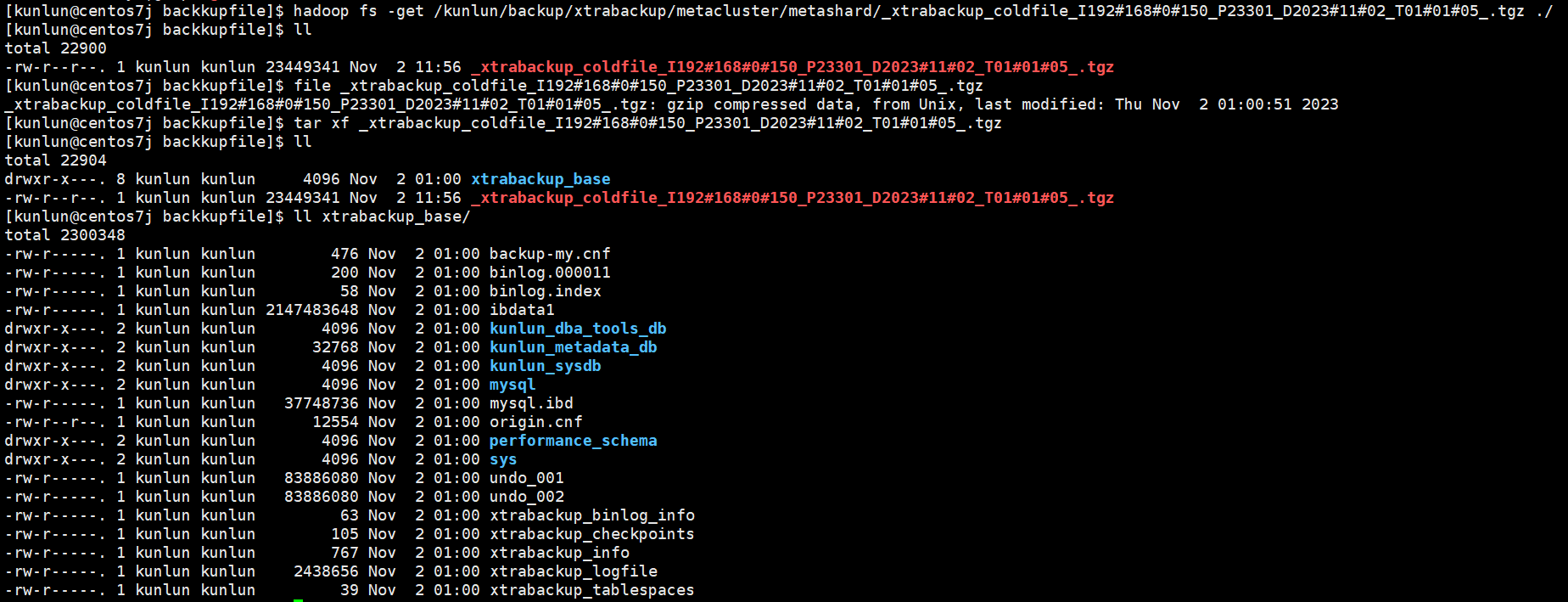
hadoop fs -get /kunlun/backup/xtrabackup/metacluster/metashard/\_xtrabackup\_coldfile\_I192#168#0#150\_P23301\_D2023#11#02\_T01#01#05\_.tgz /data/ha/backupfile/

tar xf **$coldfile**.tgz

示例：

tar xf \_xtrabackup\_coldfile\_I192#168#0#150\_P23301\_D2023#11#02\_T01#01#05\_.tgz

如图：



解压后进行恢复，步骤如下：

* apply redo日志，完成数据一致性

cd /**$install\_dir**/kunlun-node-manager-**$version**/bin/

**# $install\_dir是指node\_mgr安装目录**

**# $version 是指部署的版本号，例如：1.2.1**

示例：

cd /data/ha/kunlun/kunlun-node-manager-1.2.1/bin/

./util/xtrabackup --prepare --apply-log-only --target-dir=**$fullbackup\_base** >> /data/ha/log/rebuild\_node\_tool\_37.log 2>&1

**# $fullbackup\_base** 是指解压后的全备目录

# /data/ha/log/rebuild\_node\_tool\_37.log可自定义一个日志文件

示例：

./util/xtrabackup --prepare --apply-log-only --target-dir=/data/ha/backupfile/xtrabackup\_base >> ../log/rebuild\_node\_tool\_37.log 2>&1

* 备份配置文件，后续安装实例会用到

cp -a **$install\_dir**/storage\_datadir/**$port**/data/**$port**.cnf **$install\_dir**/kunlun/storage\_datadir/**$port**/

**# $port**是指存储节点端口号

示例，以23301端口为例：

cp -a /data/ha/kunlun/storage\_datadir/23301/data/23301.cnf /data/ha/kunlun/storage\_datadir/23301/

* 停止故障mysqld进程

**注意：node\_mgr会拉起mysqld进程，进行删除文件的速度要快，保证先停止mysqld进程再删除数据和日志文件。**

示例：

cd /data/ha/kunlun/instance\_binaries/storage/23301/kunlun-storage-1.2.1/dba\_tools; ./stopmysql.sh 23301

* 删除数据文件和日志文件

**注意：以实际的安装目录和端口号为准！！！**

rm -rf **$install\_dir**/storage\_datadir/**$port**/data/\*

rm -rf **$install\_dir**/storage\_datadir/**$port**/data/.\*

rm -rf **$install\_dir**/storage\_waldir/**$port**/redolog/\*

rm -rf **$install\_dir**/storage\_logdir/**$port**/relay/\*

rm -rf **$install\_dir**/storage\_logdir/**$port**/binlog/\*

示例：

rm -rf /data/ha/kunlun/storage\_datadir/23301/data/\*

rm -rf /data/ha/kunlun/storage\_datadir/23301/data/.\*

rm -rf /data/ha/kunlun/storage\_waldir/23301/redolog/\*

rm -rf /data/ha/kunlun/storage\_logdir/23301/relay/\*

rm -rf /data/ha/kunlun/storage\_logdir/23301/binlog/\*

* 把完成apply的全备拷贝到对应的数据目录

cd **$install\_dir**/kunlun-node-manager-**$version**/bin/

示例：

cd /data/ha/kunlun/kunlun-node-manager-1.2.1/bin/

./util/xtrabackup --defaults-file=**$install\_dir**/storage\_datadir/**$port**/**$port**.cnf --user=agent --pagent\_pwd --copy-back --target-dir=**$fullbackup\_base** >> ../log/rebuild\_node\_tool\_37.log 2>&1

示例：

./util/xtrabackup --defaults-file=/data/ha/kunlun/storage\_datadir/23301/23301.cnf --user=agent --pagent\_pwd --copy-back --target-dir=/data/ha/kunlun/storage\_datadir/23301/xtrabackup\_tmp >> ../log/rebuild\_node\_tool\_37.log 2>&1

* 删除存放全备目录下的所有文件

rm -rf **$fullbackup\_base**/\*

rm -rf **$fullbackup\_base**/.rocksdb

示例：

rm -rf /data/ha/backupfile/xtrabackup\_base/\*

rm -rf /data/ha/backupfile/xtrabackup\_base/.rocksdb

* 拷贝配置文件到对应数据目录和启动实例

cp -a **$install\_dir**/storage\_datadir/**$port**/**$port**.cnf **$install\_dir**/storage\_datadir/**$port**/data/

示例：

cp -a /data/ha/kunlun/storage\_datadir/23301/23301.cnf /data/ha/kunlun/storage\_datadir/23301/data/

启动mysql示例：

cd **$install\_dir**/instance\_binaries/storage/**$port**/kunlun-storage-**$version**/dba\_tools; ./startmysql.sh **$port**

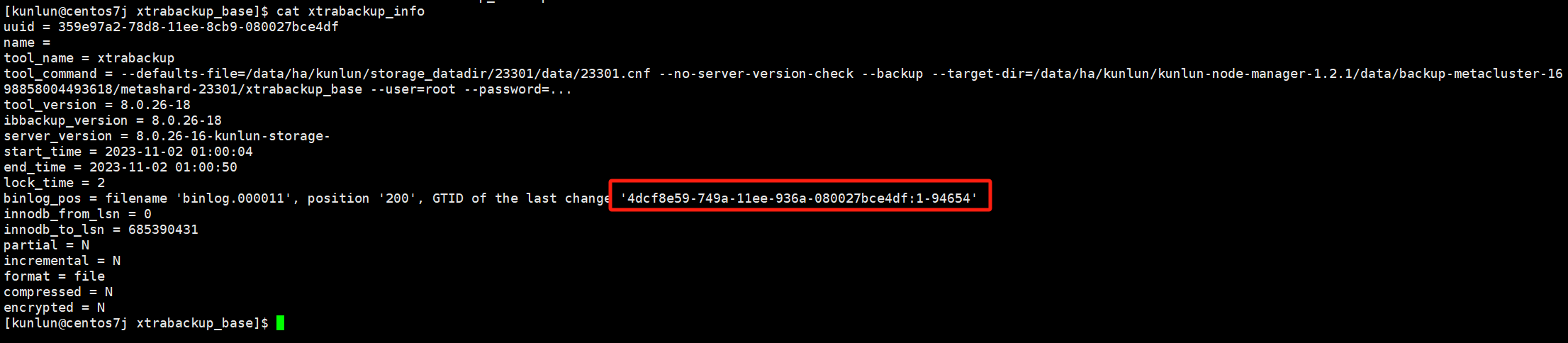
示例：

cd /data/ha/kunlun/instance\_binaries/storage/23301/kunlun-storage-1.2.1/dba\_tools; ./startmysql.sh 23301

* 根据全备信息xtrabackup\_info文件中的gtid，重新建立主备关系

全备相关的信息和GTID可以查看xtrabackup\_info文件

如图为GTID：



cd **$fullbackup\_base**

cat xtrabackup\_info

reset master; set global gtid\_purged='**$GTID**'; stop slave; reset slave all; change master to MASTER\_AUTO\_POSITION = 1, MASTER\_HOST='**$master\_host**' , MASTER\_PORT=**$master\_port**, MASTER\_USER='repl' , MASTER\_PASSWORD='repl\_pwd',MASTER\_CONNECT\_RETRY=1 ,MASTER\_RETRY\_COUNT=1000, MASTER\_DELAY=0, MASTER\_HEARTBEAT\_PERIOD=10 for CHANNEL 'kunlun\_repl'; start slave for CHANNEL 'kunlun\_repl';

**# $GTID** 记录在全备目录xtrabackup\_info文件

**# $master\_host** 当前集群的主节点ip

**# $master\_port** 当前集群的主节点端口号

示例：

reset master; set global gtid\_purged='4dcf8e59-749a-11ee-936a-080027bce4df:1-82129'; stop slave; reset slave all; change master to MASTER\_AUTO\_POSITION = 1, MASTER\_HOST='192.168.0.150' , MASTER\_PORT=23301, MASTER\_USER='repl' , MASTER\_PASSWORD='repl\_pwd',MASTER\_CONNECT\_RETRY=1 ,MASTER\_RETRY\_COUNT=1000, MASTER\_DELAY=0, MASTER\_HEARTBEAT\_PERIOD=10 for CHANNEL 'kunlun\_repl'; start slave for CHANNEL 'kunlun\_repl';

* 验证主备关系是否正常

ps -ef|grep **$port**

示例：

ps -ef|grep 23301

登录重做完成的备节点进行查看主备关系：

cd **$install\_dir**/instance\_binaries/storage/**$port**/kunlun-storage-**$version**/dba\_tools/

示例：

cd /data/ha/kunlun/instance\_binaries/storage/23301/kunlun-storage-1.2.1/dba\_tools/

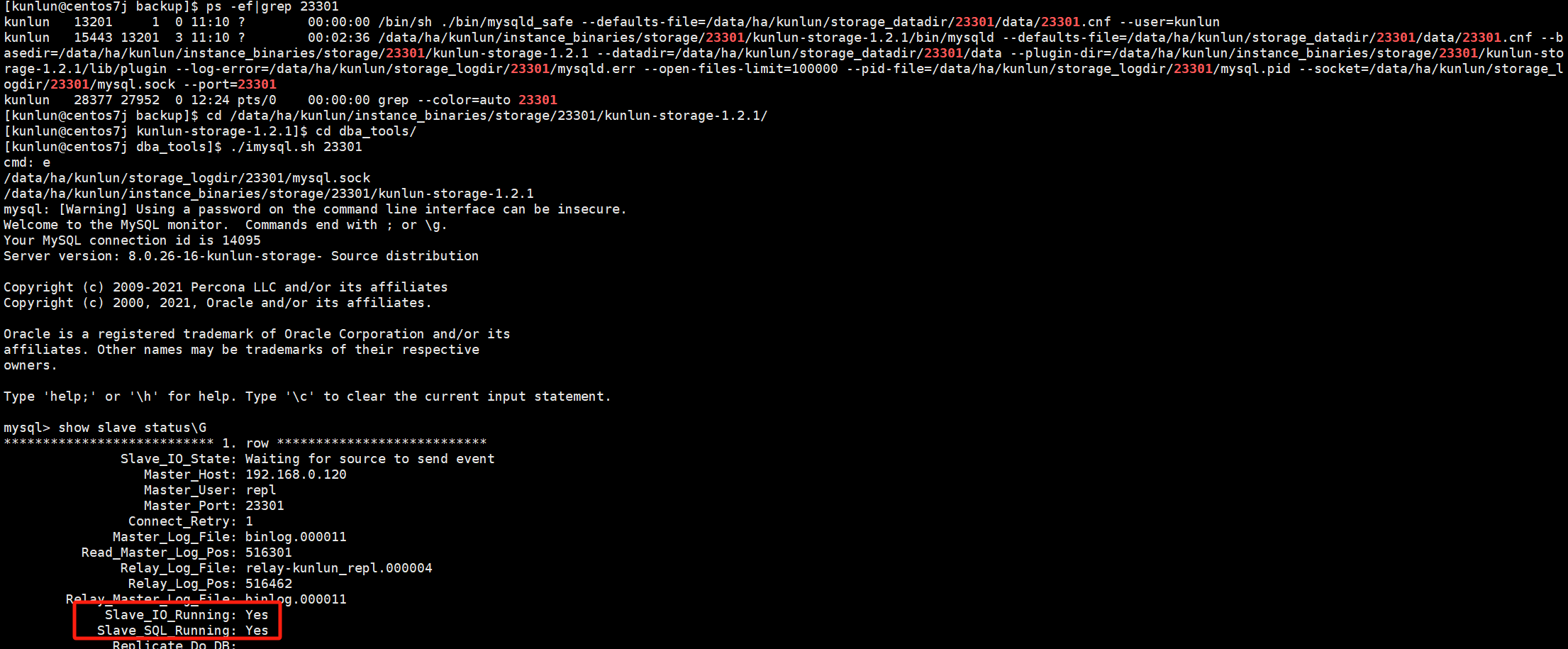
./imysql.sh **$port**

示例：

./imysql.sh 23301

show slave status\G

如图：



重做备机后，主备关系正常。

## 计算节点访问权限控制

通过修改每个计算节点的pg\_hba.conf文件，进行访问权限控制。

操作步骤，**每个计算节点操作：**

* 修改pg\_hba.conf：

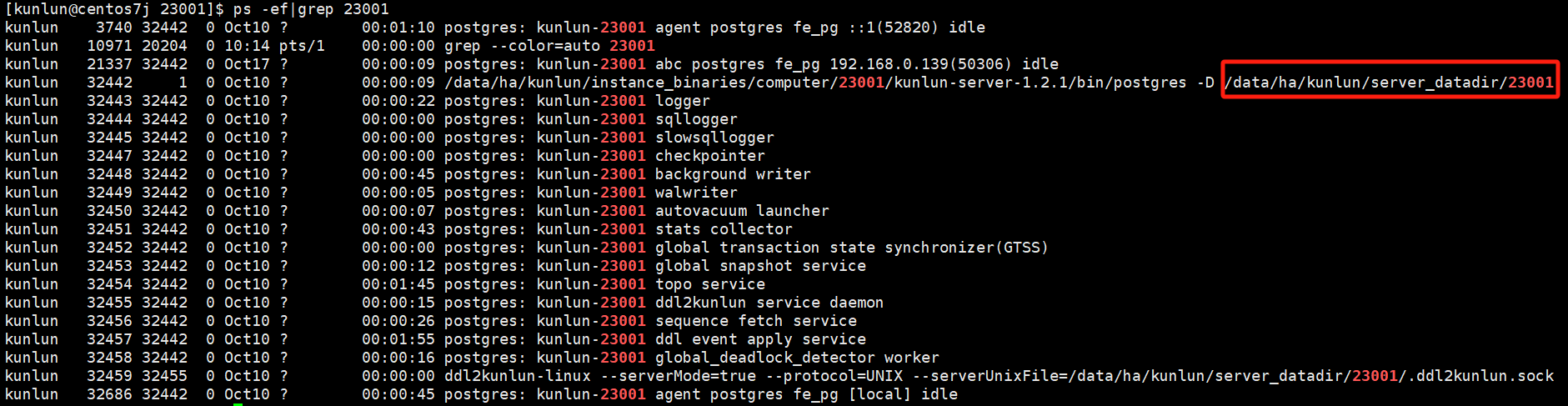
查看文件所在目录：

ps -ef|grep $port # **$port**是指**计算节点**的端口号

示例：

ps -ef|grep 23001

如图：



cd /$install\_dir/server\_datadir/$port

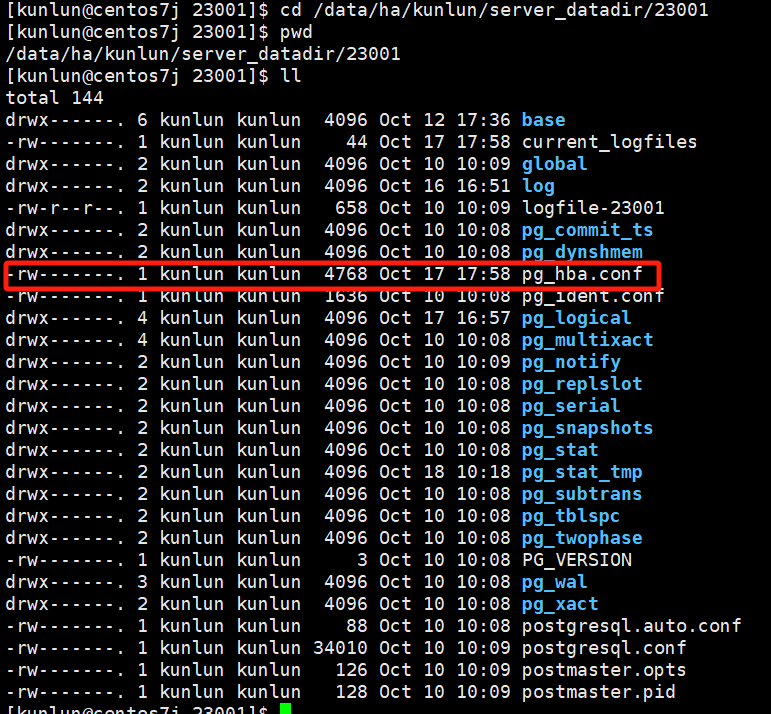
# **$install\_dir**是指计算节点安装目录

# **$port**是指计算节点端口号

示例：

cd /data/ha/kunlun/server\_datadir/23001

如图：



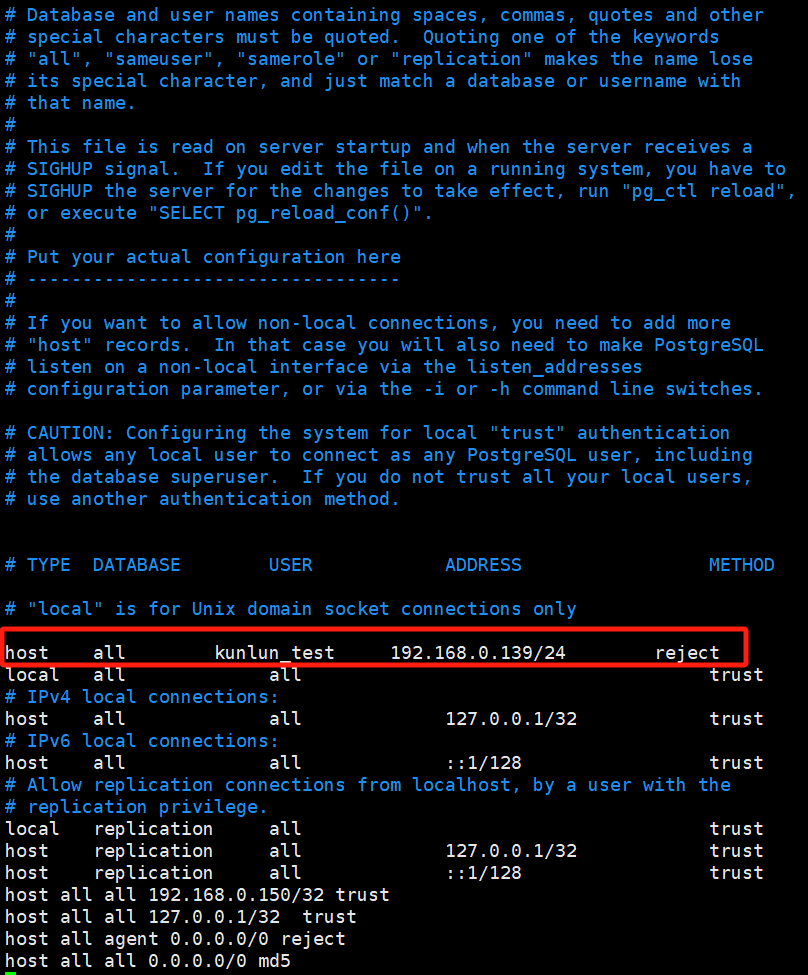
在配置文件的**第一行**加入如下内容：

host all kunlun\_test 192.168.0.139/24 reject

该内容的具体含义是拒绝所有来自192.168.0.139的kunlun\_test用户的TCP/IP连接访问。**在前面第一行加入的原因为前面的配置会覆盖后面的配置，如果前面有相关比较大的允许访问权限配置，那么后面即使设置了拒绝也无法生效。**具体的pg\_hba.conf文件参数配置介绍见PG官网链接：

<https://www.postgresql.org/docs/current/auth-pg-hba-conf.html>

如图：



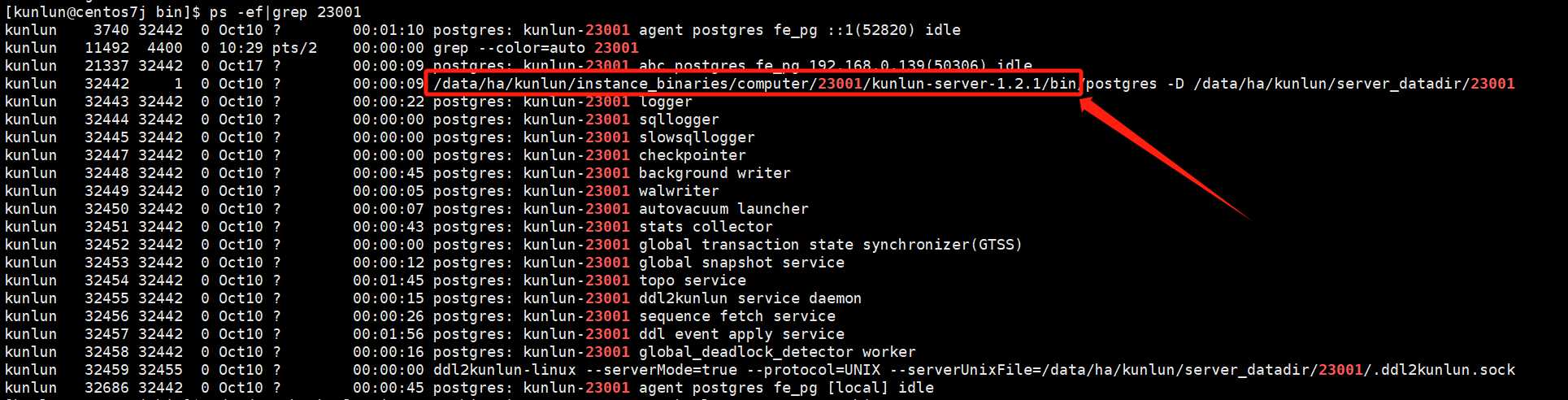
* 使配置生效：

查看kunlun-server进程，确认pg\_ctl工具所在目录

这里以kunlun-server的端口号**23001**为例：

ps -ef|grep 23001

图片示例：

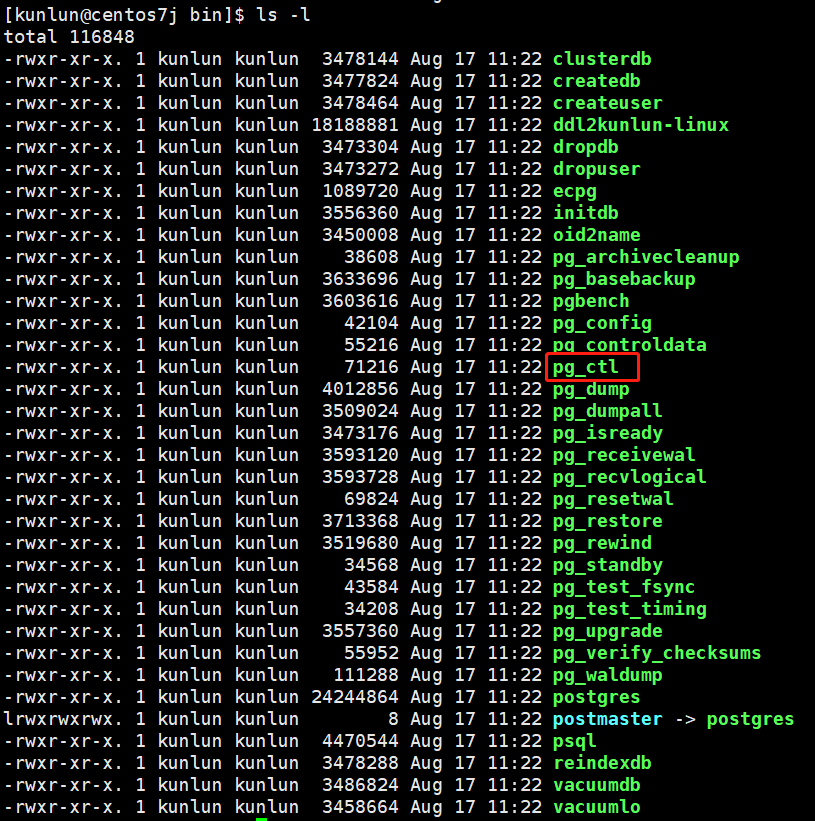


查看pg\_ctl所在的目录：

cd /data/ha/kunlun/instance\_binaries/computer/23001/kunlun-server-1.2.1/bin

ls -l

图片示例：



执行reload：

./pg\_ctl -D /data/ha/kunlun/server\_datadir/23001 reload

图片示例：



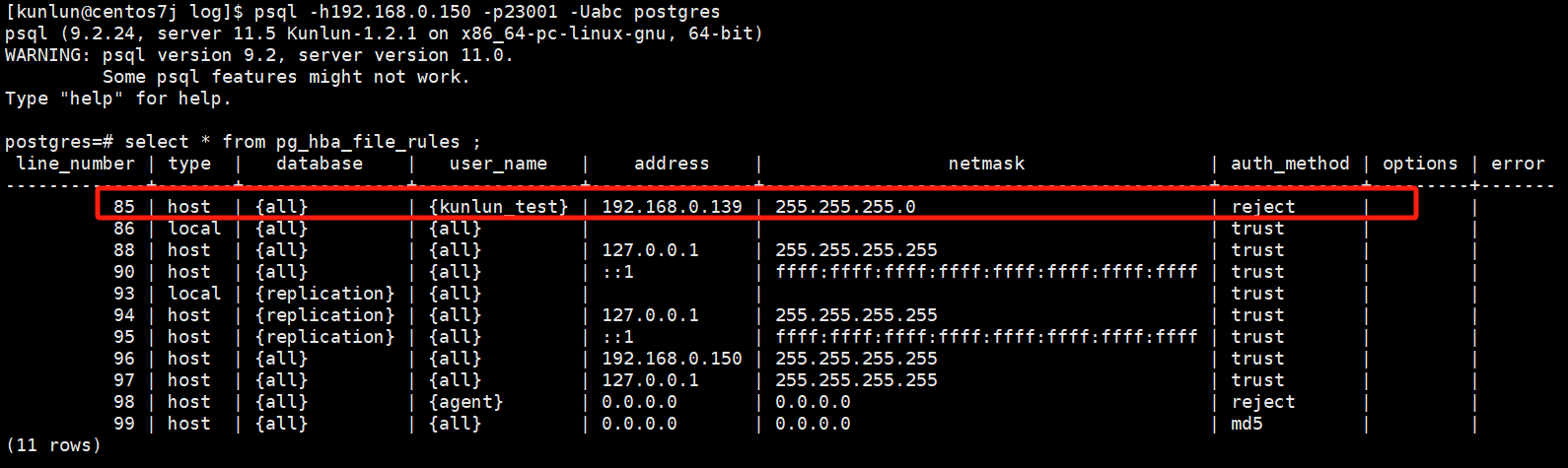
验证生效：

登录计算节点：

psql -h192.168.0.150 -p23001 -Uabc postgres

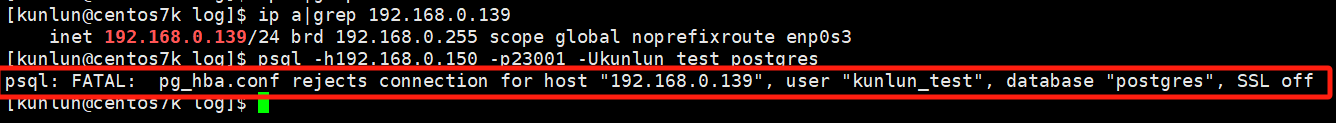
执行sql：

select \* from pg\_hba\_file\_rules ;



* 登录验证

192.168.0.139机器上执行登录：



登录失败，报错信息描述pg\_hba.conf配置文件拒绝了连接。

## 手动配置cgroup实现CPU硬隔离

**cgroup实现CPU硬隔离在创建集群会自动执行，不需要手动设置，这里只是说明如需故障排查或者其他问题需要设置cgroup时，操作使用说明。**

* 存储节点

执行命令：

到kunlun2cgroup工具目录下，该工具在node\_mgr的bin/util目录下：

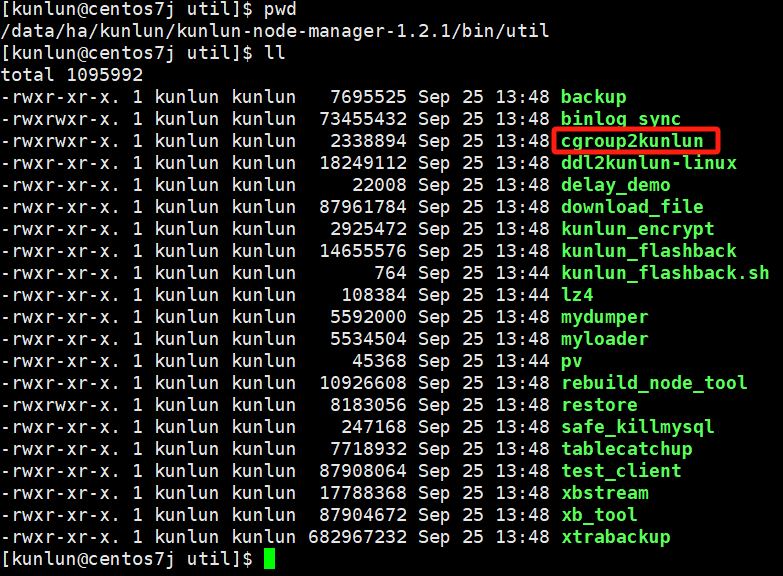
cd $base\_dir/kunlun-node-manager-1.2.1/bin/util

$base\_dir是指安装基础目录，可以在部署时的拓扑文件.json中查看。

示例：

cd /data/ha/kunlun/kunlun-node-manager-1.2.1/bin/util

如图：



cgroup2kunlun工具参数解析：

Usage of ./cgroup2kunlun:

-action string

'add' or 'delete' cgroup related to the port

-control\_mode string

resource control type, share or quota (default "quota")

-cpu\_nums int

cpu resources reserving account, 1 means one cpu cores (default 1)

-pidFile string

instance pid file path

-port string

instance port

-resource\_type string

resource controller type (default "cpu")

添加cgroup命令，示例：

./cgroup2kunlun -resource\_type=cpu -port=33503 -cpu\_nums=8 -control\_mode=quota -pidFile=/data/ha/kunlun/storage\_logdir/33503/mysql.pid -action=add

-resource\_type 参数为资源控制类型，默认为cpu

-port参数为存储节点端口号

-cpu\_nums 参数为存储节点使用的vCPU核数

-control\_mode 参数为控制模式，默认为独占限制

-pidFile 参数为存储节点mysql.pid文件的绝对路径

-action 参数为配置cgroup的行为，分为add添加和delete删除

示例：

添加存储节点33503端口，8vCPU的限制：

./cgroup2kunlun -resource\_type=cpu -port=33503 -cpu\_nums=8 -control\_mode=quota -pidFile=/data/ha/kunlun/storage\_logdir/33503/mysql.pid -action=add

删除cgroup配置：

./cgroup2kunlun -resource\_type=cpu -port=33503 -cpu\_nums=8 -control\_mode=quota -pidFile=/data/ha/kunlun/storage\_logdir/33503/mysql.pid -action=delete

* 计算节点

示例：

添加计算节点23001端口，8vCPU的限制：

./util/cgroup2kunlun -resource\_type=cpu -port=23001 -cpu\_nums=8 -control\_mode=quota -pidFile=/data/ha/kunlun/server\_datadir/23001/postmaster.pid -action=add

删除cgroup配置：

./util/cgroup2kunlun -resource\_type=cpu -port=23001 -cpu\_nums=8 -control\_mode=quota -pidFile=/data/ha/kunlun/server\_datadir/23001/postmaster.pid -action=delete

## 扩展管理extension

### extension管理

* 计算节点添加扩展（**注意：集群只有一个计算节点时！！！**）

以添加**uuid-ossp**扩展为例

安装相关依赖

sudo yum install -y e2fsprogs-devel uuid-devel libuuid-devel zlib-devel zlib-static

下载pg11.2版本进行源码编译

wget https://ftp.postgresql.org/pub/source/v11.20/postgresql-11.20.tar.gz

Tips：如果wget命令下载速度过慢，可以尝试用电脑访问下载后传到服务器上。

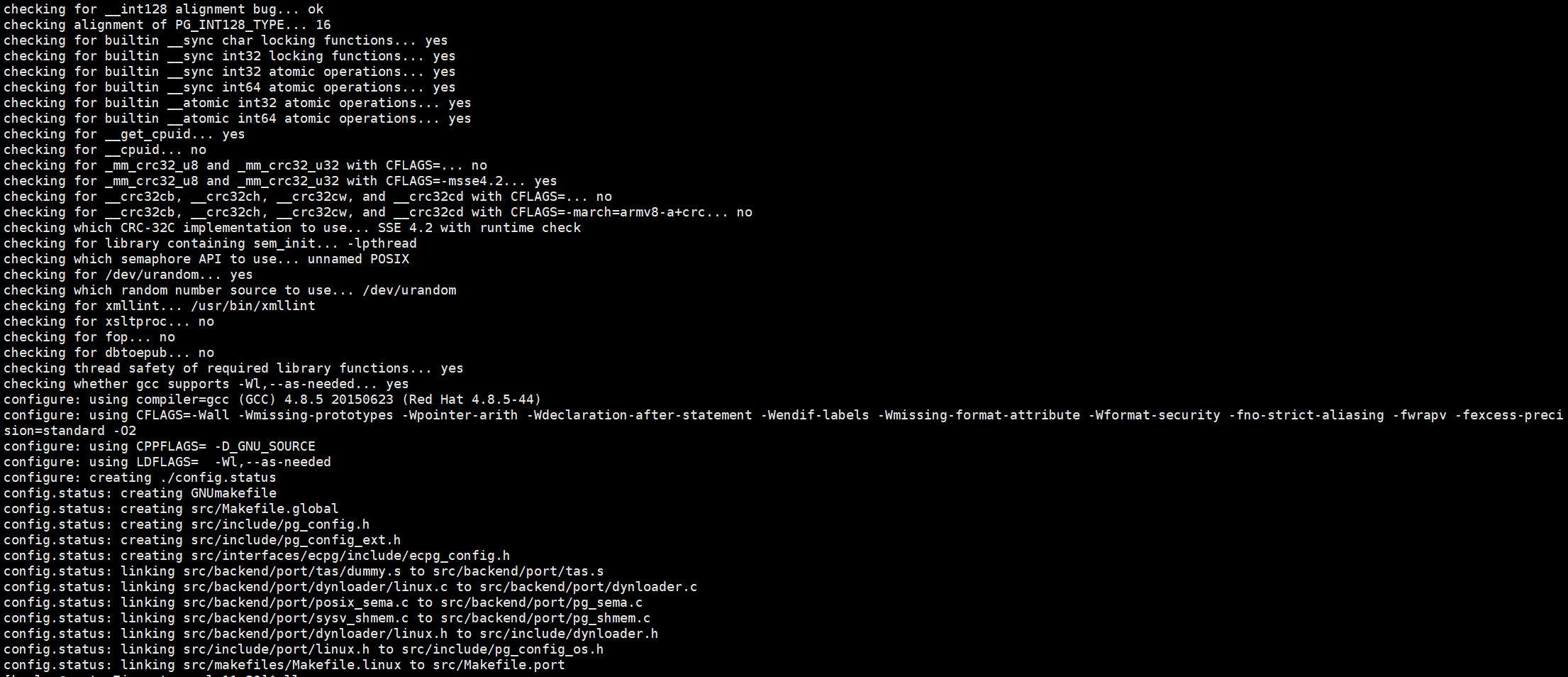
解压后编译

Tar xf postgresql-11.20.tar.gz

cd postgresql-11.20

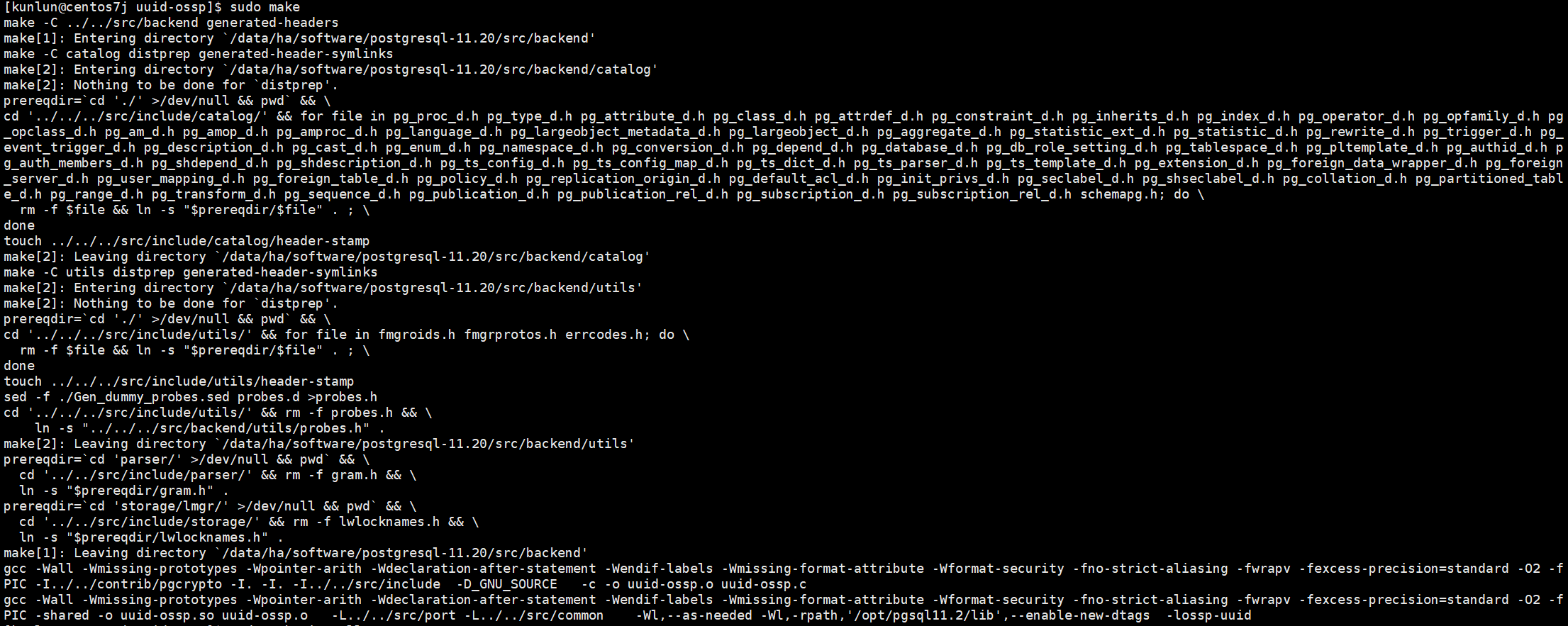
./configure --prefix=/opt/pgsql11.2 --with-uuid=ossp

执行后结果如下图，无报错：



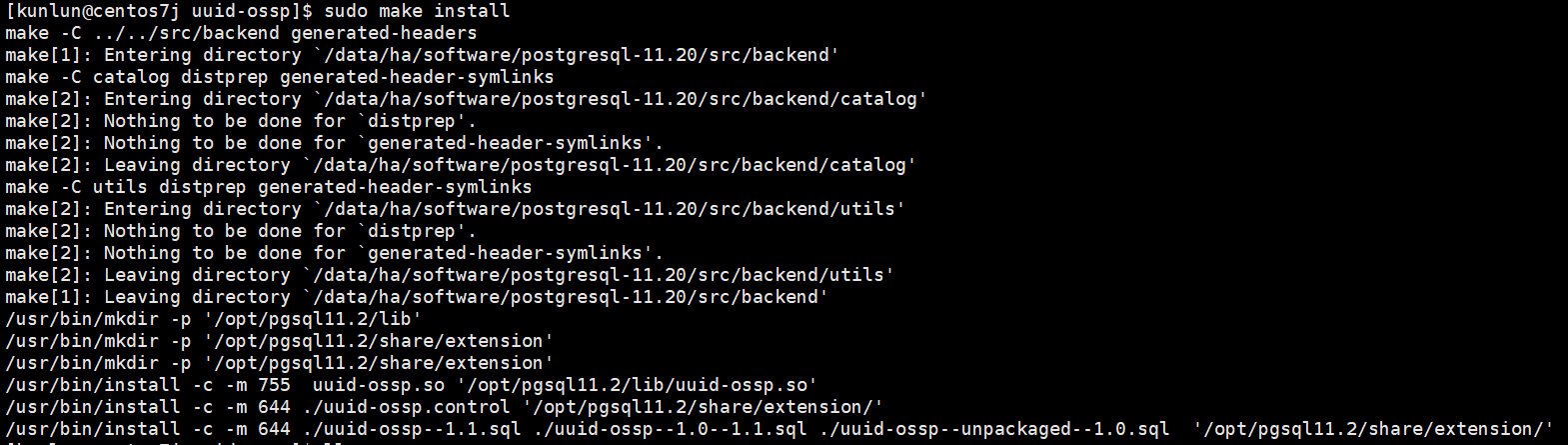
sudo make

执行命令后如图：



sudo make install

执行命令后如图：



将编译好的扩展相关文件拷贝到对应的目录

cd postgresql-11.20 && sudo chown kunlun:kunlun -R \*

cd postgresql-11.20/contrib/uuid-ossp/

cp -a uuid-ossp.so **/data/ha/kunlun**/instance\_binaries/computer/**23001**/**kunlun-server-1.2.1**/lib/postgresql/

cp -a uuid-ossp--1.0--1.1.sql uuid-ossp--1.1.sql uuid-ossp--unpackaged--1.0.sql uuid-ossp.control **/data/ha/kunlun**/instance\_binaries/computer/**23001**/**kunlun-server-1.2.1**/share/postgresql/extension/

目录介绍：

**/data/ha/kunlun**为**$base\_dir**是指安装基础目录，可以在部署时的拓扑文件.json中查看。

**23001**为计算节点端口号，以实际的为准。

**kunlun-server-1.2.1**为计算节点安装的目录，一般以kunlun-server-版本号定义。

登录计算节点后创建uuid-ossp扩展

先查看已经存在的扩展

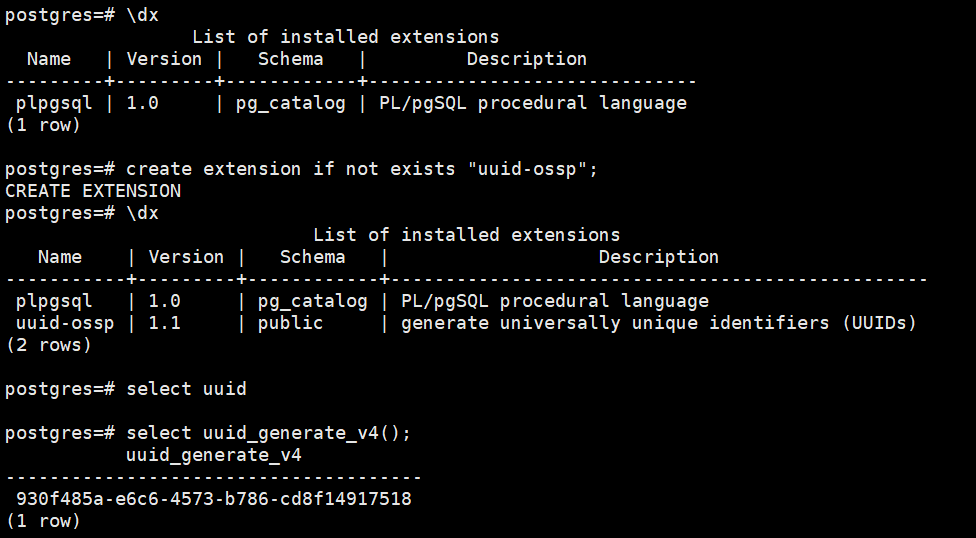
\dx

create extension if not exists "uuid-ossp";

\dx

select uuid\_generate\_v4();

如图所示：



* 计算节点添加扩展（**注意：集群有多个计算节点时！！！**）

**同集群内其他计算节点添加扩展**，需要yum安装所需依赖

sudo yum install -y e2fsprogs-devel uuid-devel libuuid-devel zlib-devel zlib-static

省去编译步骤，见单个计算节点的步骤进行操作。

然后在所有的计算节点上将编译好的扩展文件拷贝到对应的目录。

例如本实例中uuid-ossp扩展，只需要编译好的

uuid-ossp.so文件拷贝到

**/data/ha/kunlun**/instance\_binaries/computer/**23001**/**kunlun-server-1.2.1**/lib/postgresql/目录下；

uuid-ossp--1.0--1.1.sql

uuid-ossp--1.1.sql

uuid-ossp--unpackaged--1.0.sql

uuid-ossp.control这四个文件拷贝到

**/data/ha/kunlun**/instance\_binaries/computer/**23001**/**kunlun-server-1.2.1**/share/postgresql/extension/

最后在一个计算节点上进行操作加载extension即可，其他的计算节点就可以看到对应的extension。

登录计算节点后创建uuid-ossp扩展

先查看已经存在的扩展

\dx

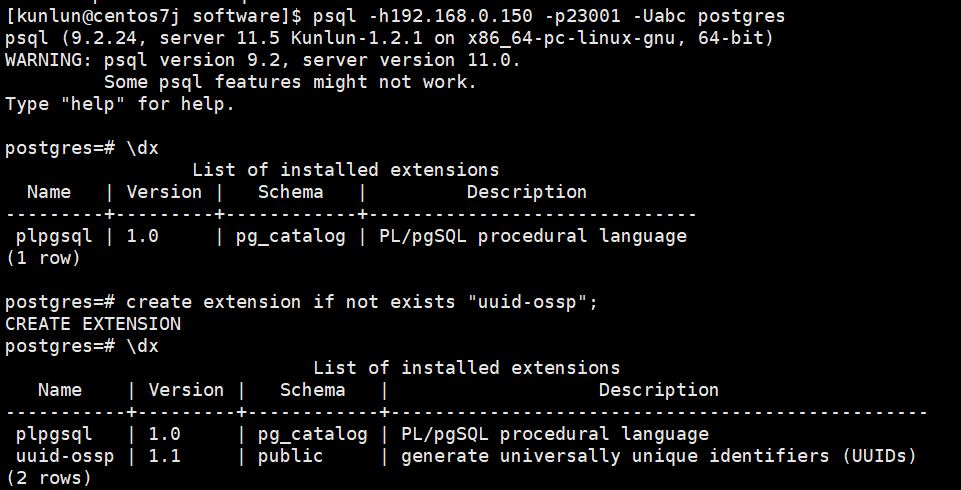
create extension if not exists "uuid-ossp";

\dx

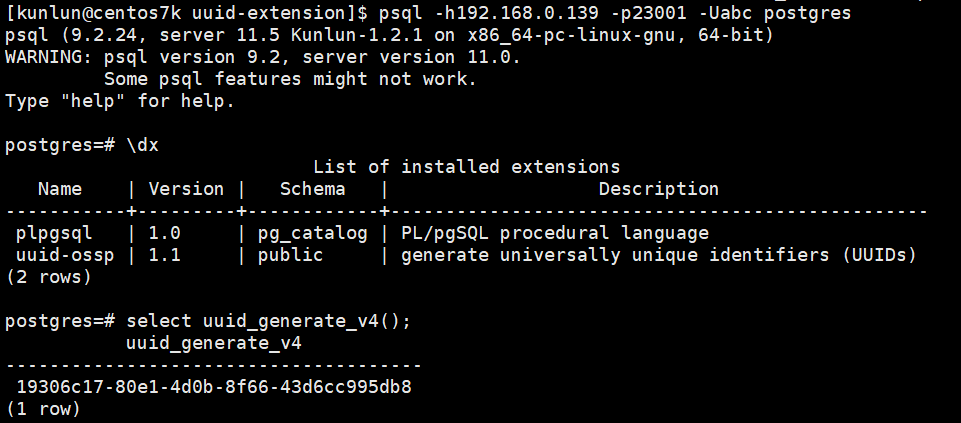
select uuid\_generate\_v4();

如图所示：

计算节点1：



计算节点2：



* 将扩展文件添加到公共目录

为了往后**新增计算节点**或者**创建新集群**，不需要重复操作添加扩展，先在各个计划新增计算机器节点上安装依赖

sudo yum install -y e2fsprogs-devel uuid-devel libuuid-devel zlib-devel zlib-static

然后将扩展文件拷贝到对应的公共目录即可。

cd postgresql-11.20/contrib/uuid-ossp/

cp -a uuid-ossp.so $base\_dir/program\_binaries/**kunlun-server-1.2.1**/lib/postgresql/

cp -a uuid-ossp--1.0--1.1.sql uuid-ossp--1.1.sql uuid-ossp--unpackaged--1.0.sql uuid-ossp.control $base\_dir/program\_binaries/**kunlun-server-1.2.1**/share/postgresql/extension/

XPanel发起新增机器节点或者创建集群即可看到扩展。

### Postgres\_fdw扩展使用

postgres\_fdw扩展为postgesql提供，无须手动下载后安装，直接加载extension即可。关于postgres\_fdw介绍及操作示例可以先参考官网链接：

<http://postgres.cn/docs/11/postgres-fdw.html>

，了解后再进行操作。

环境介绍：

外部server和表等相关信息：

host：192.168.0.150

post：23007

user：abc

password：省略

db\_name：klustron，

schema\_name：public，

table\_name：k1。

内部映射相关信息：

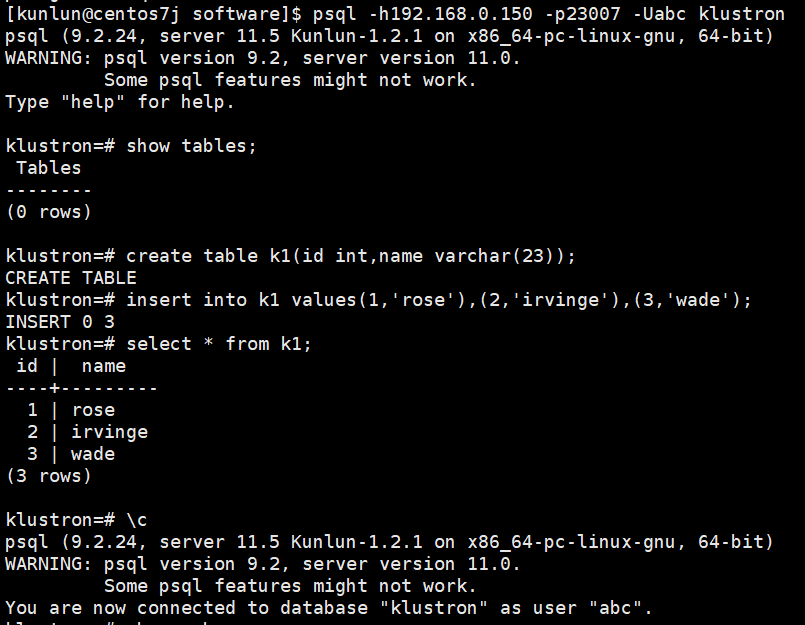
server：foreign\_server

用户映射：abc

映射表：foreign\_table1

如下为实际的操作步骤：

* 外部表k1信息



* 内部映射等配置相关操作

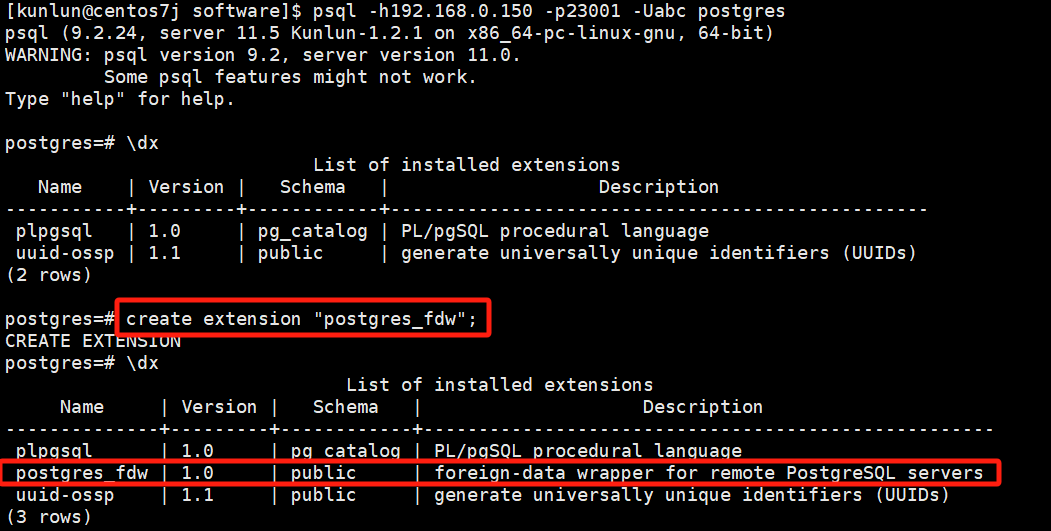
(1)创建extension

\dx

create extension "postgres\_fdw";

\dx

如图：



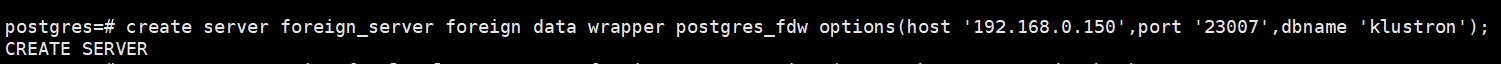
1. create server，create mapping，create foreign table

create server foreign\_server foreign data wrapper postgres\_fdw options(host '192.168.0.150',port '23007',dbname 'klustron');

create user mapping for user server foreign\_server options(user 'abc', password 'abc');

create foreign table foreign\_table1(id int,name varchar(23)) server foreign\_server options(schema\_name 'public',table\_name 'k1');

如图：







(3)验证和操作外部表

select \* from foreign\_table1;

insert into foreign\_table1 values(6,'james'),(7,'durant'),(8,'bryant');

select \* from foreign\_table1;

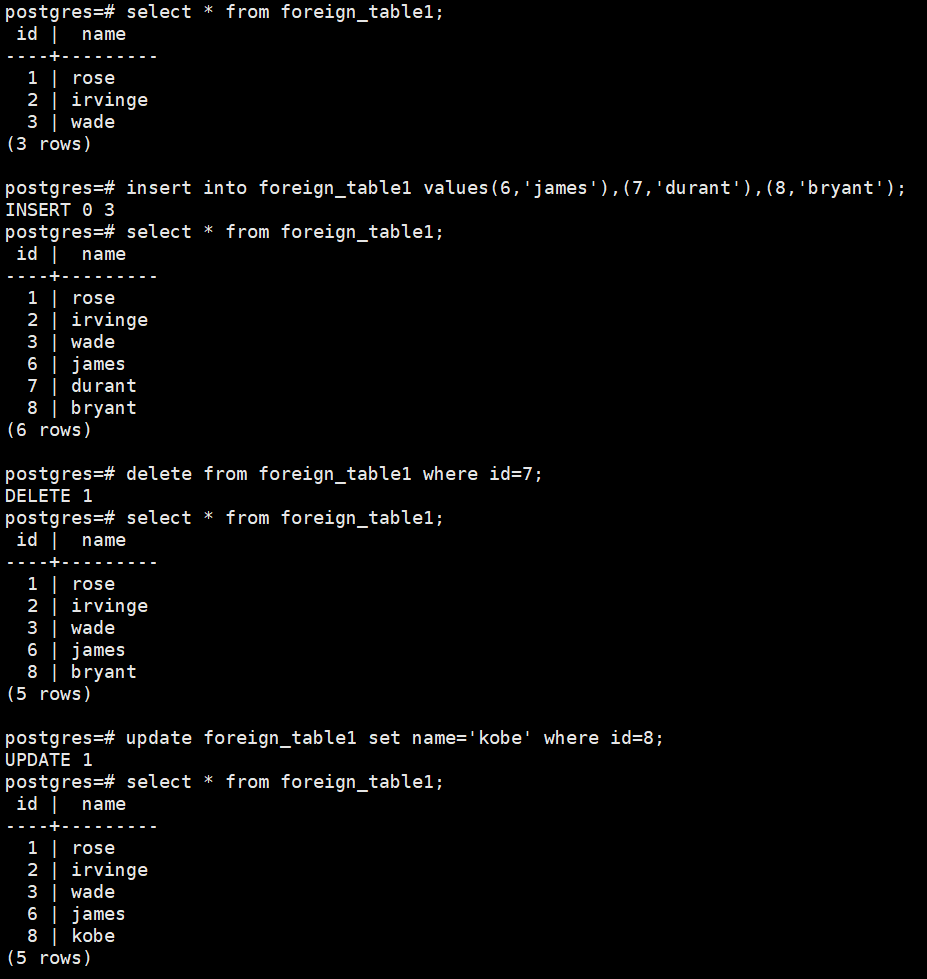
delete from foreign\_table1 where id=7;

select \* from foreign\_table1;

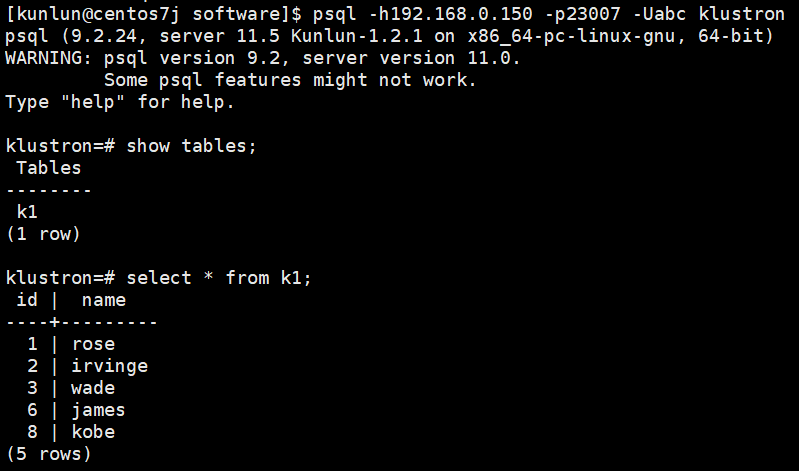
update foreign\_table1 set name='kobe' where id=8;

select \* from foreign\_table1;

如图：



外部表登录验证：



清理环境：

drop foreign table foreign\_table1;

drop user mapping for user server foreign\_server;

drop server foreign\_server;

drop extension postgres\_fdw;

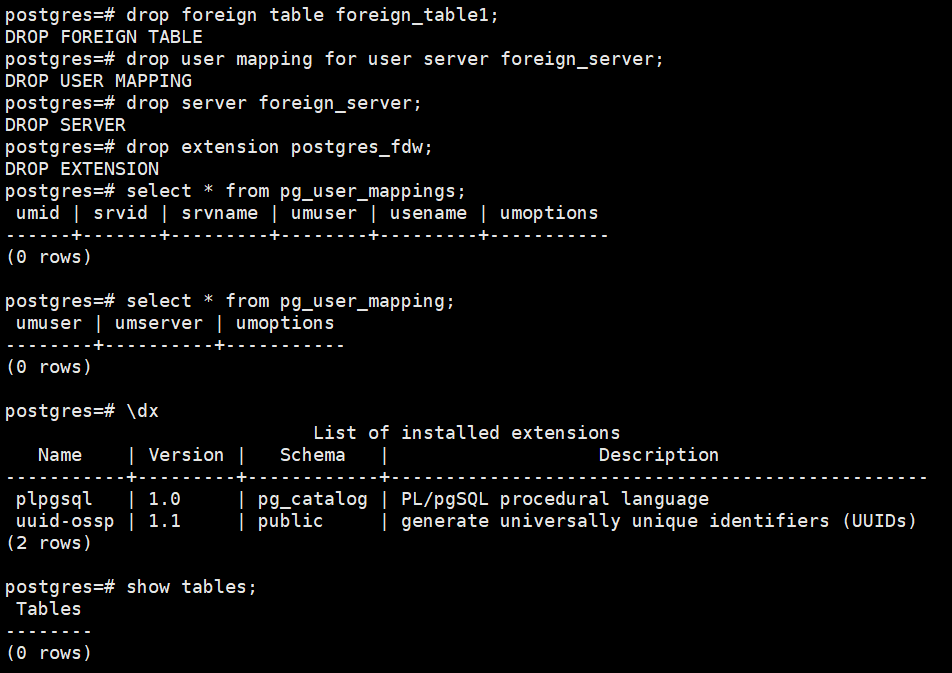
select \* from pg\_user\_mappings;

select \* from pg\_user\_mapping;

\dx

show tables;

如图：



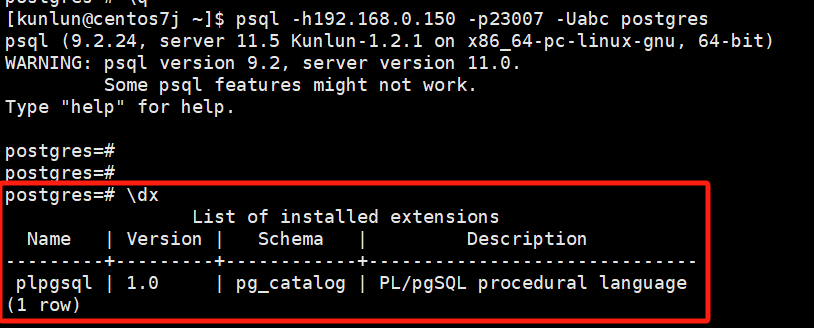
## SSL认证

### 单向开启SSL

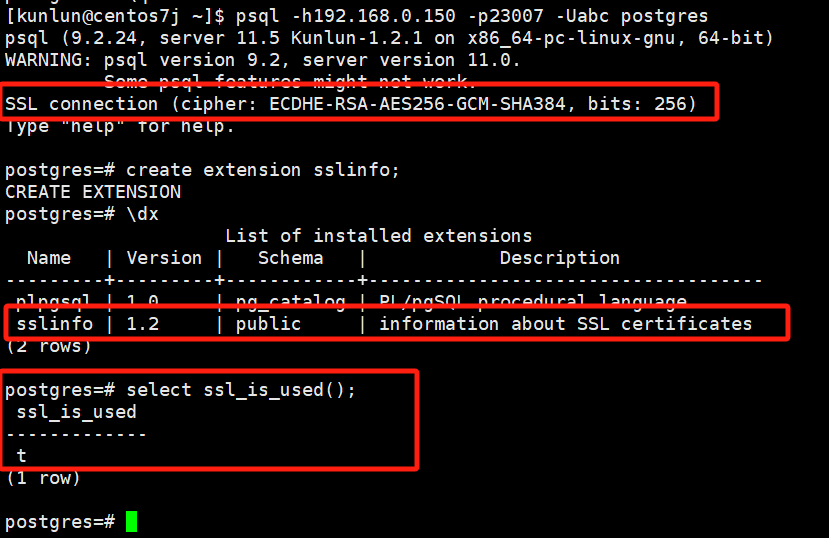
环境准备：

yum install -y openssl-devel openssl perl-ExtUtils\*

未开启SSL前，登录计算节点没有SSL的信息：



开启后SSL后，登录计算节点有SSL信息：



开启步骤如下：

* 生成证书和私钥文件

mkdir openssl

cd openssl/

openssl req -new -x509 -days 365 -nodes -text -out server.crt -keyout server.key -subj "/CN=centos7j"

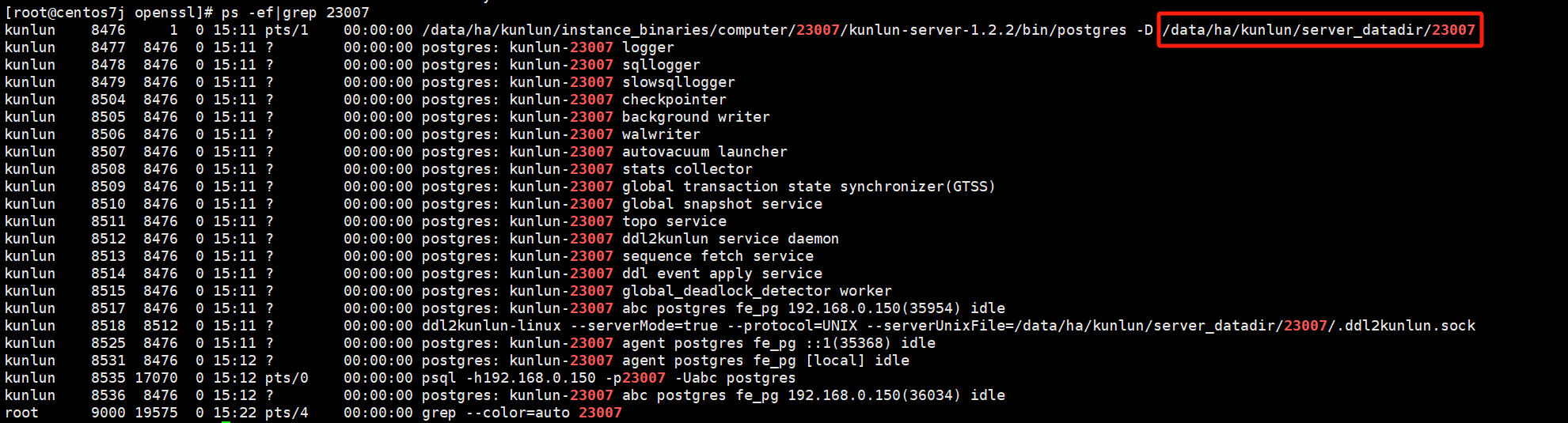
# /CN=后跟主机名，可以使用hostname命令得出

修改server.key文件权限

chmod 600 server.key

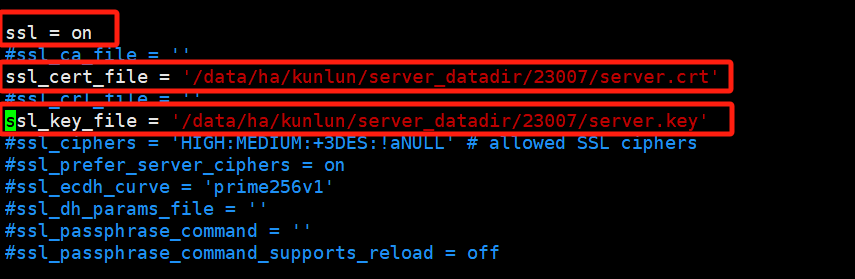
拷贝到对应目录下，这里以计算节点的23007端口号为例：

cp -a server.\* /data/ha/kunlun/server\_datadir/23007/ #具体目录以实际为准，可执行ps -ef|grep $port $port代表计算节点端口号。



* 修改配置文件

vim postgresql.conf



根据具体的文件绝对路径填写，ssl\_cert\_file和ssl\_key\_file，并开启ssl。

* 重启计算节点

查看计算节点进程：

ps -ef|grep $port # **$port**是指**计算节点**的端口号

示例：

ps -ef|grep 23007

到计算节点的安装目录中与bin目录同级的scripts目录下：

cd /$install\_dir/instance\_binaries/computer/$port/kunlun-server-$version/scripts

# **$install\_dir**是指计算节点安装目录

# **$port**是指计算节点端口号

# **$version** 是指计算节点的版本号，例如：1.2.2

示例：

cd /data/ha/kunlun/instance\_binaries/computer/23007/kunlun-server-1.2.2/scripts

启动：

python2 start\_pg.py --port=$port

示例：

python2 start\_pg.py --port=23007

停止：

python2 stop\_pg.py --port=$port

示例：

python2 stop\_pg.py --port=23007

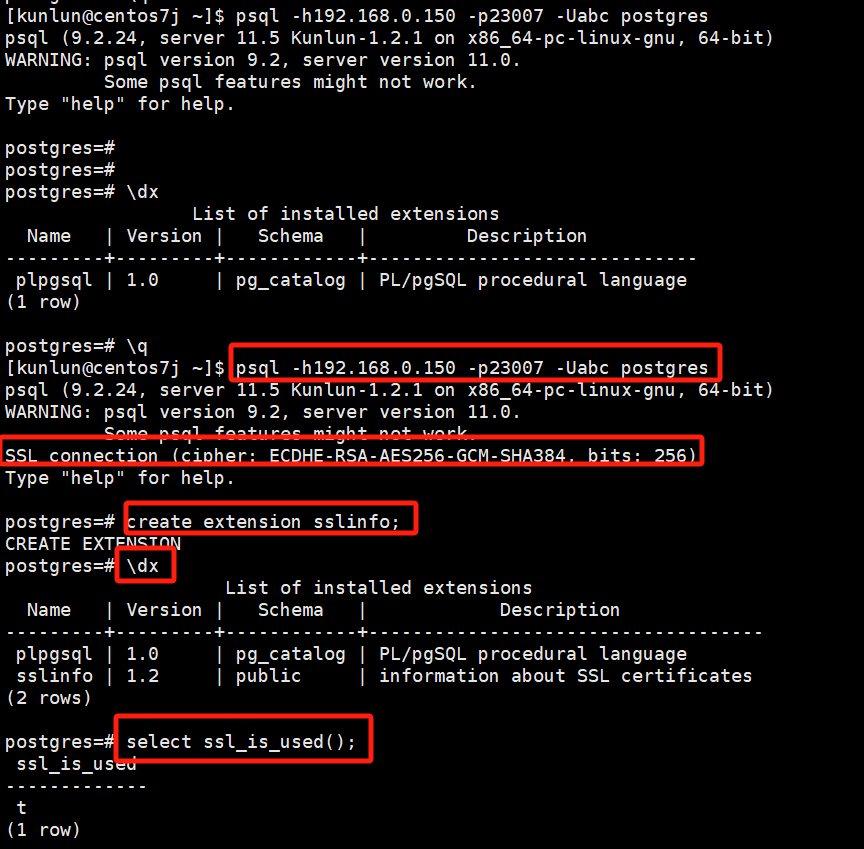
* 登录计算节点创建sslinfo扩展和验证

psql -h192.168.0.150 -p23007 -Uabc postgres

create extension sslinfo;

\dx

select ssl\_is\_used();

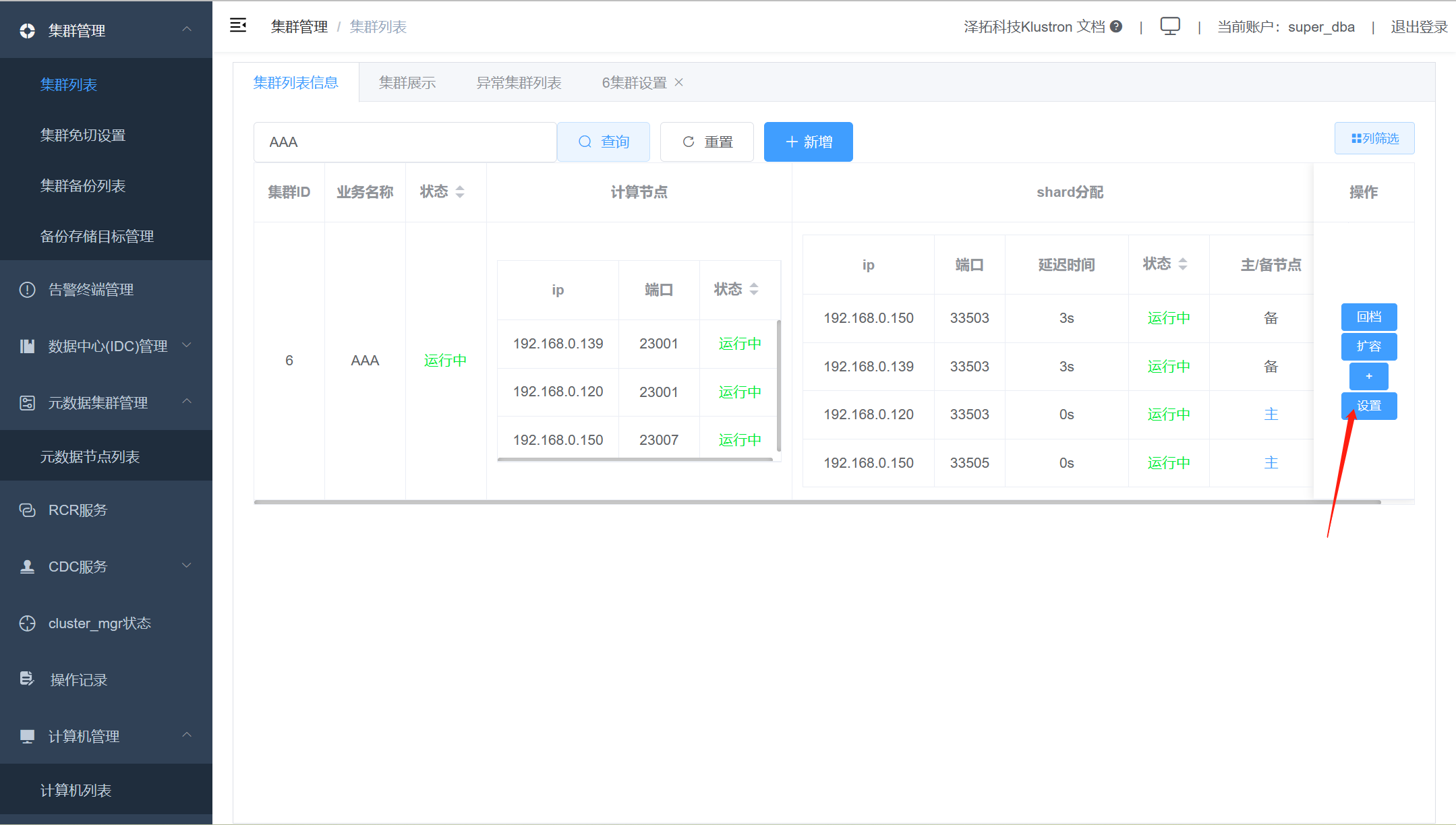


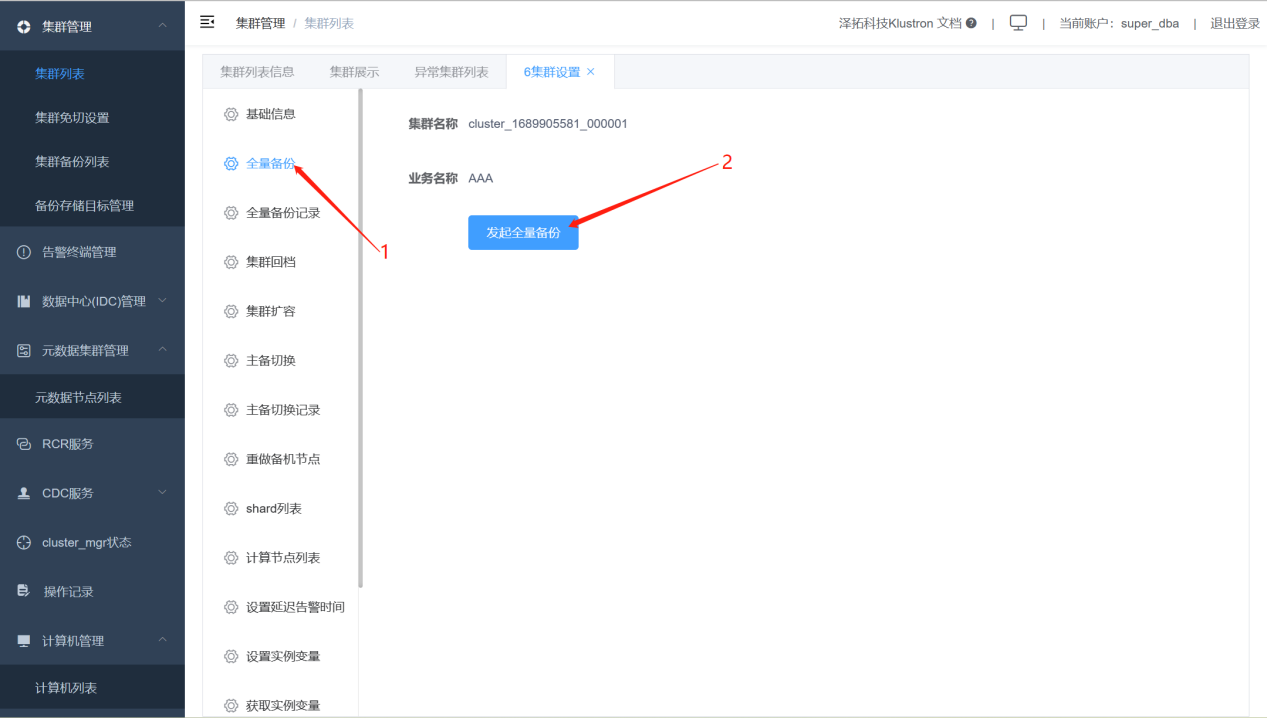
# 备份与恢复

## 物理备份

注意：发起物理备份前，请确认HDFS已经部署成功，并在XPanel的【集群管理】->【备份存储目标管理】中添加成功。

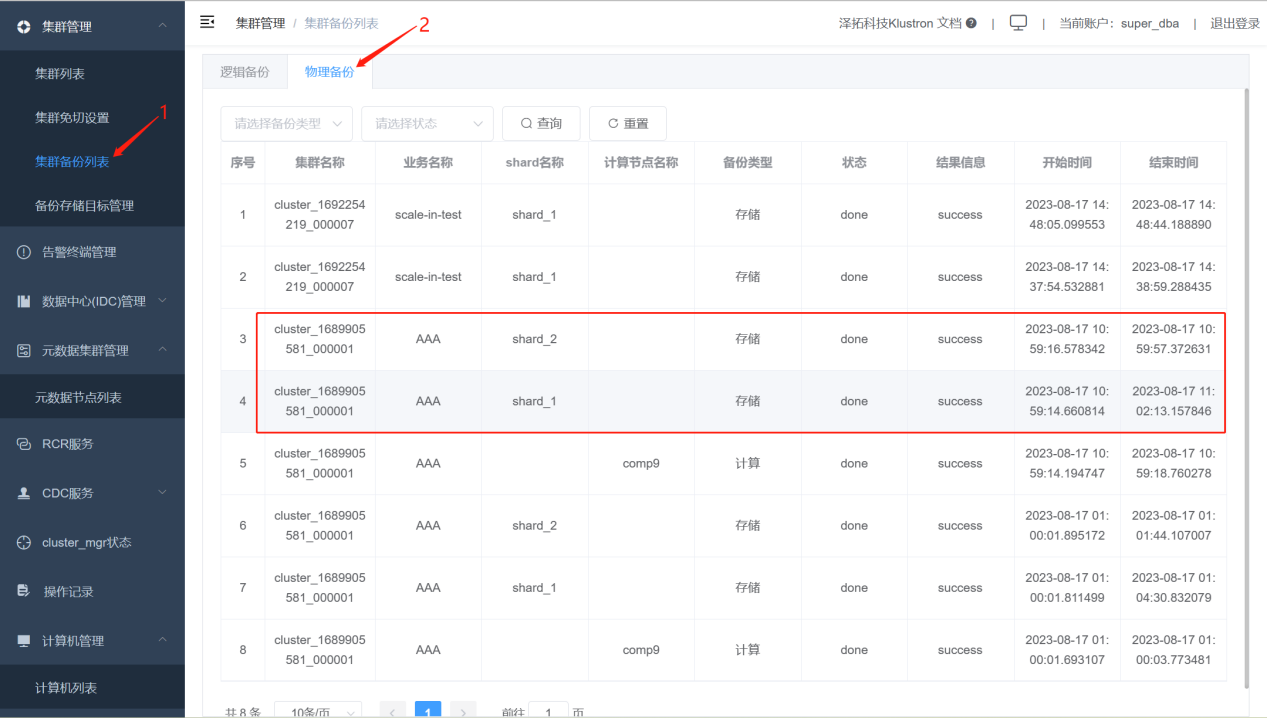
通过XPanel点击【发起全量备份】，即刻开始进行集群物理备份任务。





验证物理备份进度

（1）XPanel点【集群备份列表】->【物理备份】，可以查看到发起物理备份的任务状态。



1. 到HDFS上进行校验，是否备份成功。

Hdfs机器上执行命令：

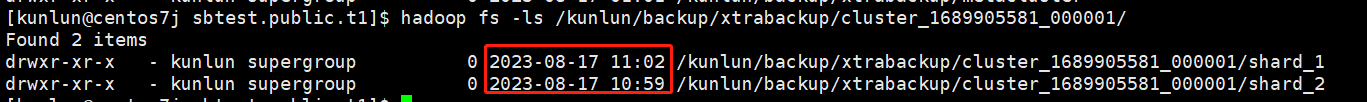
hadoop fs -ls /kunlun/backup/xtrabackup/$cluster\_name/

# **$cluster\_name** 是指集群名称，例如cluster\_1689905581\_000001，可以在第一个步骤中拷贝得到。

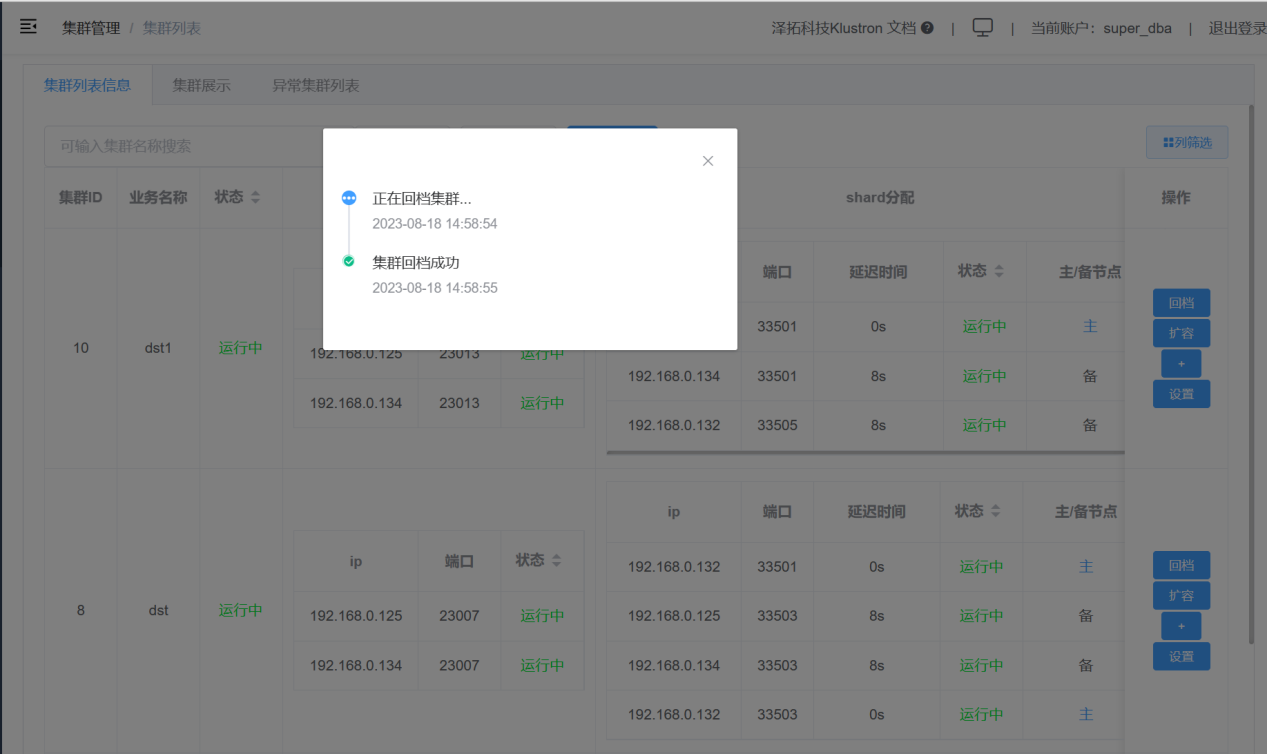
示例：

hadoop fs -ls /kunlun/backup/xtrabackup/cluster\_1689905581\_000001/

如图：



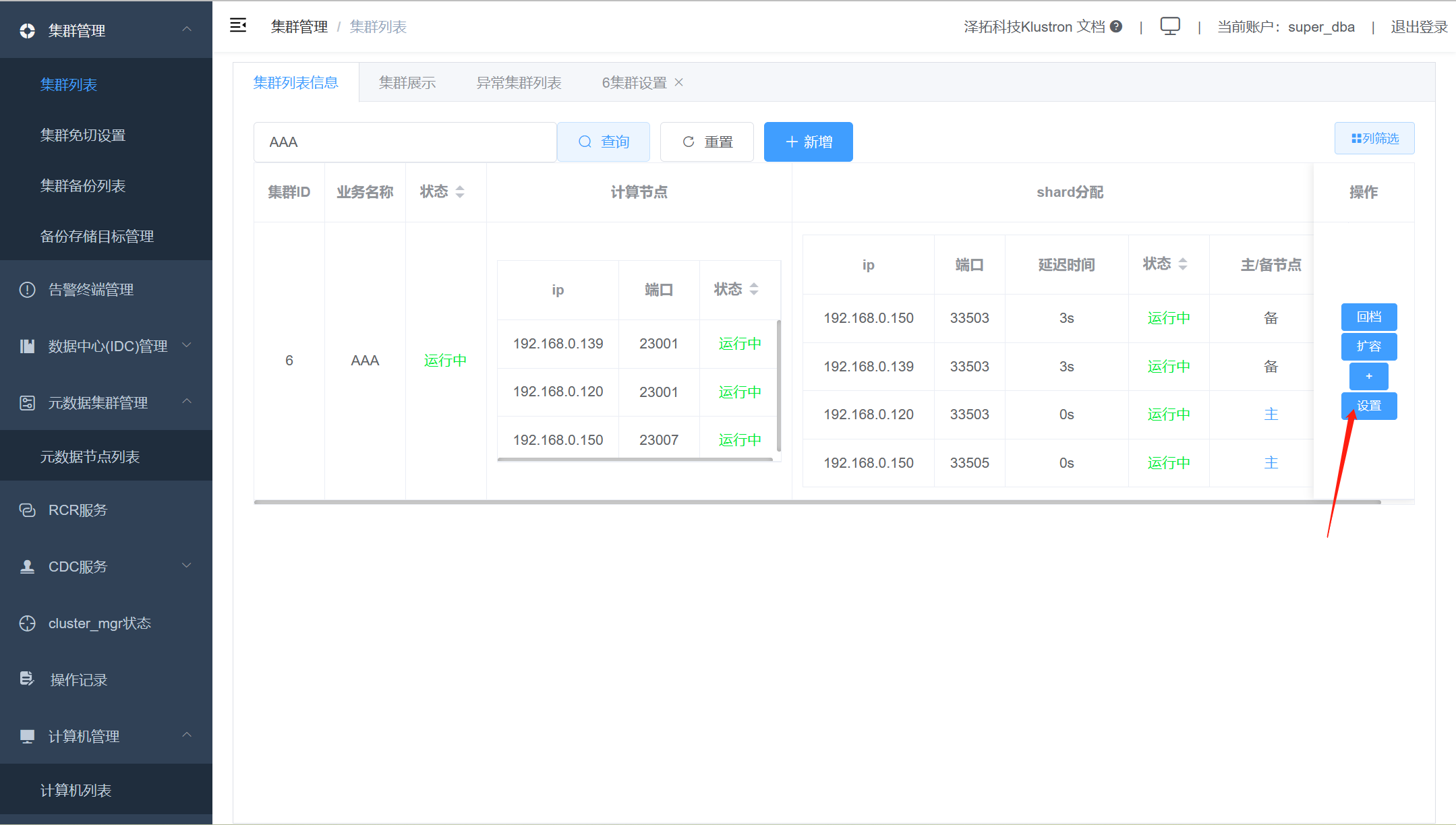
## 物理回档



## 逻辑备份

注意：发起逻辑备份前，请确认HDFS已经部署成功，并在XPanel的【集群管理】->【备份存储目标管理】中添加成功。

通过XPanel中选择需要备份的库或者表，然后选择【备份时间范围】，点击【保存】，即刻开始进行该集群下指定的库或者表逻辑备份。





验证逻辑备份进度

（1）XPanel点【集群备份列表】->【逻辑备份】，可以查看到发起逻辑备份的任务状态。



1. 到HDFS上进行校验，是否备份成功。

Hdfs机器上执行命令：

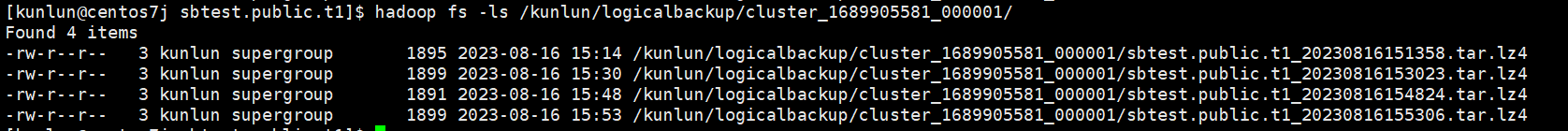
hadoop fs -ls /kunlun/logicalbackup/$cluster\_name/

# **$cluster\_name** 是指集群名称，例如cluster\_1689905581\_000001，可以在第一个步骤中拷贝得到。

示例：

hadoop fs -ls /kunlun/logicalbackup/cluster\_1689905581\_000001/

如图：



## 逻辑恢复

# 集群指引参考

## 错误码介绍

这里只对Klustron特有的错误码进行介绍，pg和mysql常见错误见如下链接：

Pg错误码：

<https://www.postgresql.org/docs/11/errcodes-appendix.html>

MySQL错误码：

<https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html>

存储节点错误码：

* Error Code：9000

强同步等待超时，具体强同步介绍见链接：

<https://doc.kunlunbase.com/zh/Klustron_Storage_Cluster_Fullsync.html>

* Error Code：9001

V1.2.1或以后版本已经废弃

* Error Code：9002

SEQUENCE序列已经存在。

* Error Code：9003

数据库中的序列通过直接插入kunlun\_sysdb.sequences系统表而错误创建，且未缓存。mysqld重启后可能可以使用或者根本无法使用。

* Error Code：9004

SEQUENCE序列不存在。

* Error Code：9005

SEQUENCE序列的可用值达到最大值后，不循环。

* Error Code：9006

指定了错误的序列参数值。

* Error Code：9007

SEQUENCE序列因使用在使用中不能删除或修改。

* Error Code：9008

ENTITY实体类型未找到。

## 元数据表说明

### pg\_cluster\_meta

字段说明:

[comp\_node\_id]

- 当前集群的当前计算节点的唯一ID，由与`cluster\_id`关联的`cluster\_name`指定

[cluster\_id]

- `cluster\_name` 指定的当前集群的唯一 ID

[cluster\_master\_id]

- 元数据集群主实例的唯一ID，代表MGR的读写节点。 元数据集群是一个 MySQL MGR 集群。

[cluster\_name]

- 当前集群的名称。

[comp\_node\_name]

- 当前计算节点的名称。

### pg\_cluster\_meta\_nodes

字段说明:

[server\_id]

- mysql server\_id 变量值。用作主键来识别行。

[cluster\_id]

- 集群的id，对于一个此类表的所有行都是相同的

[is\_master]

- 该节点是否为主

[port]

- 用于连接的端口号

[hostaddr]

- 用于连接的 ip 或 dns 信息

[user\_name]

- 用于连接的用户名

[passwd]

- 用于连接的密码

### pg\_computing\_node\_stat

字段说明:

[comp\_node\_id]

- 当前集群的当前计算节点的唯一ID，由与`cluster\_id`关联的`cluster\_name`指定

[num\_queries]

- 当前计算节点处理的查询数

[num\_1shard\_queries]

- 发送到单个后端分片的查询数量

[num\_slow\_queries]

- 慢查询数量

[num\_rejected\_queries]

- 计算节点拒绝处理的查询数量

[num\_illegal\_queries]

- 非法查询的数量，i.e.访问权限问题等

[num\_ro\_queries]

- 只读查询数

[num\_sent\_queries]

- 发送到后端存储分片的查询总量

[num\_recv\_res]

- 从存储分片收到的查询结果总量（以字节为单位）

[num\_txns]

- 处理的事务总量，包括已提交、已中止、2PC 或 2PC One-phase

[num\_1shard\_txns]

- 其操作仅在单个存储分片中执行的事务数

[num\_ro\_txns]

- 只读事务数

[num\_rb\_txns]

- 回滚事务数

[num\_rb\_txns\_deadlock]

- 死锁导致的回滚事务数。

[accumulated\_since]

- 该节点的所有统计数据都是自该时间戳以来累积的

### pg\_ddl\_log\_progress

字段说明:

[dbid]

- kunlun\_metadata\_db.db\_cluster中每个簇有一行，“dbid”指主键

[ddl\_op\_id]

- 按当前计算节点重放ddl日志的位置

[max\_op\_id\_done\_local]

- ddl 首次在本地完成的ddl日志的最大位置，用于分布式DDL崩溃安全

### pg\_shard

字段说明:

[name]

- 存储分片名称

[id]

- 每个存储分片的唯一ID

[master\_node\_id]

- 每个存储分片的主节点id

[num\_nodes]

- 每个存储分片的节点数

[space\_volumn]

- 表占用文件的大小

[num\_tablets]

- 表数量

[db\_cluster\_id]

- 当前集群的id指kunlun\_metadata\_db.db\_cluster的主键

[when\_created]

- 表创建时间

### pg\_shard\_node

字段说明:

[id]

- 与 kunlun\_metadata\_db.shard\_nodes.id关联的当前分片节点的唯一id

[port]

- 端口分片节点监听

[shard\_id]

- 与kunlun\_metadata\_db.shard.id关联的分片的唯一id

[svr\_node\_id]

- 服务器嵌套的机器的ID

[ro\_weight]

- 当将只读语句选择分派到备节点时，选择具有最大分片ro\_weight的分片。 将其设置为 0 以禁用备节点读取节点

[hostaddr]

- 用于连接的IP信息

[user\_name]

- 用于连接的用户名

[passwd]

- 用于连接的密码信息

[when\_created]

- 创建时间

# 集群常见问题

## 产品介绍及使用

**Q：Klustron支持云上购买集群吗？**

A：支持，Klustron目前已经在阿里云和AWS云上线，登录阿里云上搜索kunlunbase，可以进行规格查看以及购买使用，目前支持体验免费两天集群的使用。登录AWS云上搜索kunlunbase，可以进行规格查看以及购买使用。

* 阿里云上的Klustron快速体验链接：

[KunlunBase HTAP 分布式数据库【最新版】\_Linux\_LAMP\_MySQL-云市场-阿里云 (aliyun.com)](https://market.aliyun.com/products/56024006/cmgj00061010.html?spm=5176.21213303.J_6704733920.25.130a53c9CG0ETL&scm=20140722.S_market@@%E8%BD%AF%E4%BB%B6%E5%B8%82%E5%9C%BA@@cmgj00061010._.ID_cmgj00061010-RL_kunlunbase-LOC_search~UND~ad-OR_ser-V_3-P0_0" \l "sku=yuncode5501000001)

* AWS云上的Klustron快速体验链接：

[亚马逊云科技 Marketplace: KunlunBase (amazonaws.cn)](https://awsmarketplace.amazonaws.cn/marketplace/pp/prodview-l323th5kfolrm?sr=0-1&ref_=beagle&applicationId=AWSMPContessa)

## 部署安装

**Q：Klustron能否部署在云虚拟机，或者自建机房环境中？**

A：Klustron支持两种部署方式：第一种，在线部署，在云虚拟机上通过在线部署的方式进行部署Klustron。第二种，离线部署，通过提前下载并上传安装包到自建机房的服务器上，然后进行部署。详情见[第三章](#_标准集群部署)节。

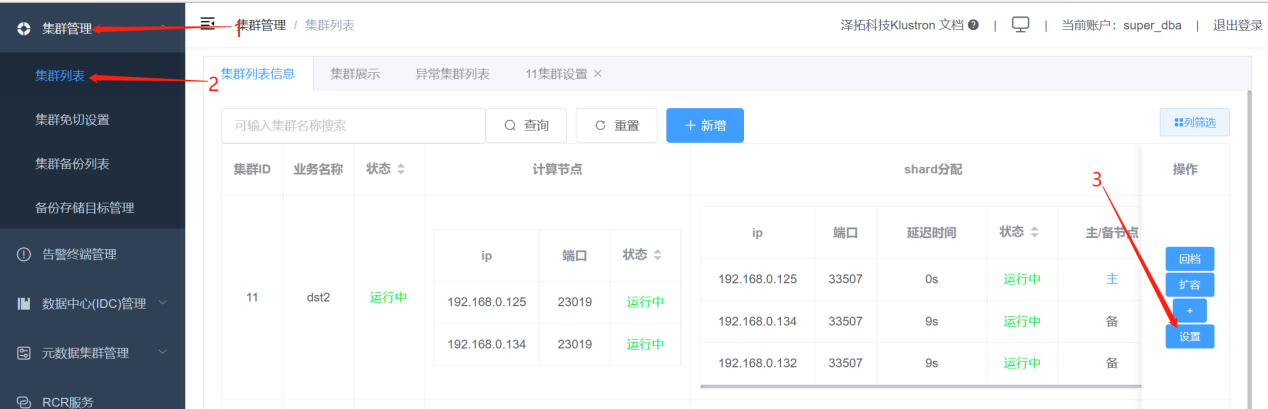
## 集群高可用

**Q：当集群中有shard的备节点故障后迟迟没有恢复正常状态，或者一直处于高延迟情况，该如何快速的恢复备节点？**

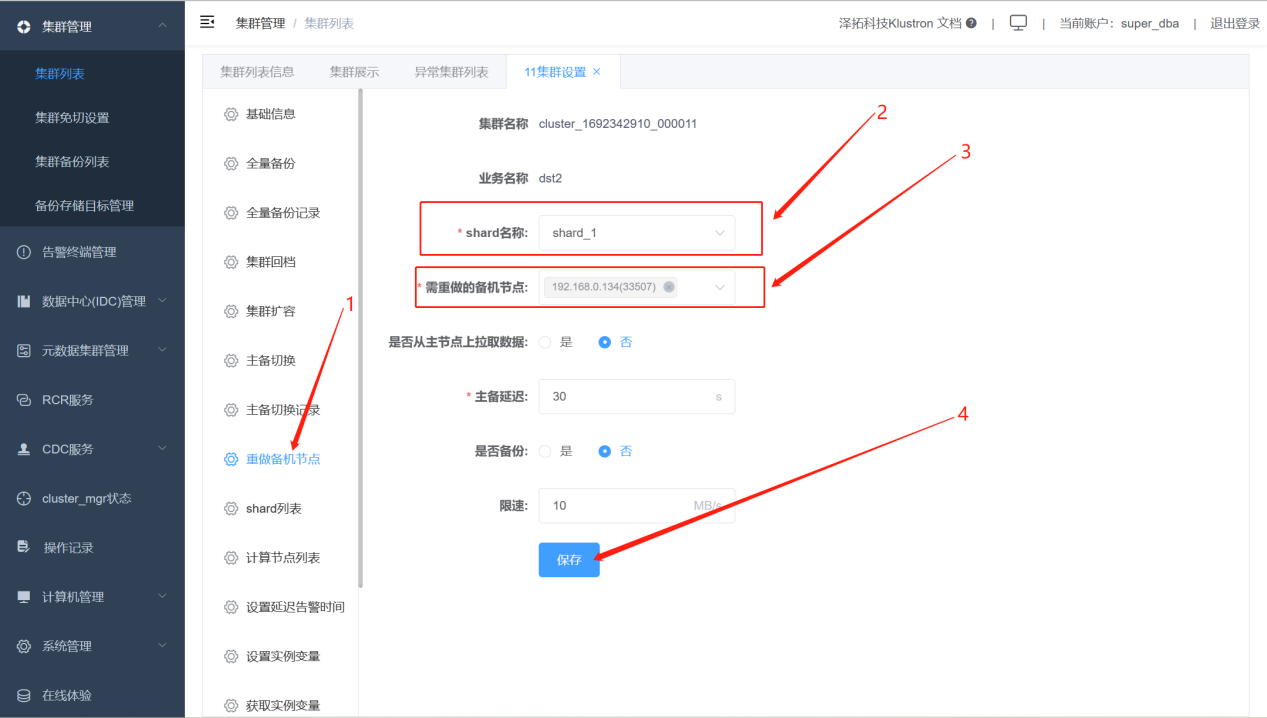
A：可以通过XPanel上【集群管理】->【集群列表】->【设置】->【重做备机节点】，选择对应的“shard名称”和“需重做的备机节点”，其他选项保持默认即可，点击【保存】，即可发起重做备机任务流程。

步骤如下图：

（1）【集群管理】->【集群列表】->【设置】



（2）【重做备机节点】->“shard名称”，“需重做的备机节点” ->【保存】



## 备份恢复

## SQL语法相关

## 监控告警

## 数据迁移