

解决方案实践

# 云基华海政务大数据解决方案实践

文档版本 1.0  
发布日期 2023-12-08



版权所有 © 华为技术有限公司 2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## 商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

## 注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

# 安全声明

## 漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：

<https://www.huawei.com/cn/psirt/vul-response-process>

如企业客户须获取漏洞信息，请参见如下网址：

<https://securitybulletin.huawei.com/enterprise/cn/security-advisory>

# 目录

<b>1 方案概述</b> .....	<b>1</b>
<b>2 资源和成本规划</b> .....	<b>4</b>
<b>3 实施步骤</b> .....	<b>5</b>
3.1 Redis 的部署.....	5
3.2 mysql 的部署.....	8
3.3 nginx 的部署.....	12
3.4 activemq 安装.....	13
3.5 MongoDB 安装.....	15
3.6 rocketmq 部署.....	16
3.7 Kafka 的安装.....	18
3.8 其他组件(华为提供).....	23
3.9 数据集成平台应用部署.....	23
3.10 态势感知平台应用部署.....	24
3.11 组织关联平台应用部署.....	25
3.12 数据治理平台部署.....	27
3.13 共享交换平台的部署.....	29
<b>4 修订记录</b> .....	<b>32</b>

# 1 方案概述

## 应用场景

- 业务挑战

通常政府各部门数据及业务现状不清楚，跨部门、跨区域、跨业务数据难共享，存在数据烟囱、信息孤岛问题，数据缺少科学分类和统一管理。

系统建设中需要完成各类数据资源的接入，实现分散的业务数据归集，但跨平台、跨业务缺少能支撑多种采集方式的数据采集工具，实现多源异构数据的归集。

通常政务数据呈现不完整、不规范、不准确等数据质量问题，整合海量数据，如何快速支撑各类应用调用，如何形成数据资产，保证数据服务泄密等安全问题。

关键KPI指标完成情况无法落实到人、随时跟踪、及时发现问题。业务难以围绕区域、行业、业务、时间等多维度深度分析，发现变化趋势，深度剖析问题，领导决策难以做到看图说话、读数决策。

- 解决方案

云基华海电子政务大数据平台，主要以各政府部门业务信息资源为核心基础，依托国家电子政务网，借助现有的各级政府数据共享交换平台，运用大数据、云计算、人工智能等高新技术手段，建设电子政务“数据资源集成+数据仓库构建+数据资源治理+大数据分析应用”的面向政务流域的全流程、全级次、全方位的大数据平台，为构建集约化、高效化、透明化的政府治理与运行模式提供支撑。

- 方案价值

整合各类数据资源，打通各个业务系统单独建设形成的数据孤岛，建成覆盖各部门系统“全业务”的信息系统以及各类数据的集中、共享与交换，更好地满足公众服务和行业管理需要；

积极推进政府各项业务工作跨部门、跨地区、跨层级协同，实现横向、纵向可通过信息化方式进行高效的业务协同、监管协同，从而提高整体工作效能。

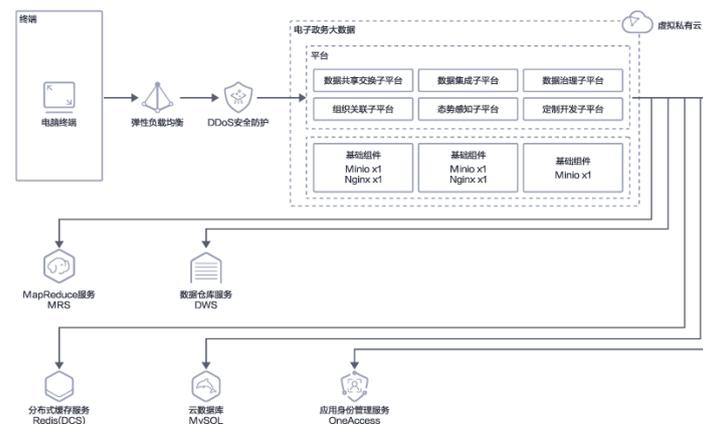
构建形成公共基础数据库和各类主题数据库为基础，推动政务数据迁移集聚，发挥政务大数据协同服务的作用，简化办事流程，打造公共服务和社会治理的新模式，带动地方经济发展，支撑政府管理研判和决策机制。

## 方案架构

方案主要由“政务数据集成共享+数仓构建+数据治理服务+数据智能化产品体系”形成面向政务领域全流程、全级次、全方位的政务大数据解决方案：

1. 通过数据集成平台，实现对政务数据的统一汇聚；
2. 通过数据共享交换子平台，实现政务数据内部共享和对外开放；
3. 提供政务元数据管理、数据标准管理、数据规范建模、数据质量管理、政务数据开发和数据资产管理等政务数据治理服务；
4. 利用组织关联和知识图谱技术，为政务部门提供业务画像、时序图谱等智能化应用场景分析；
5. 凭借人工智能和数据可视化技术，对政务数据进行智能化挖掘和分析，提供运行监测、产业大脑、应急预警和分析研判等智慧化决策支撑。

图 1-1 云基华海政务大数据解决方案整体架构



### 部署架构说明：

- 云基础设施：基于华为云基础设施底座承载。主要涉及的云服务有弹性云主机、虚拟私有云、弹性负载均衡以及安全服务Anti-DDos等；

- 云PaaS服务：数据主要采用数据湖服务MRS、数据仓库服务DWS、云数据库服务【存储】RDB进行存储和计算，采用分布式缓存服务DCS应对平台数据访问；
- 应用软件层：应用层软件基于弹性云服务器部署，单独划分虚拟私有云；

## 方案优势

- 产品体系完备且优势明显  
自主研发共享交换、数据集成、数据治理、组织关联和态势感知等完善的系列数据产品，组织关联产品在行业内拥有先发优势。
- 行业方案成熟度高  
长期专注于政务数据全景建设并深耕十年。曾经承接九个省部级政务大数据平台建设以及服务三个省级以上大型项目数据业务，政务领域实践经验丰富。
- 政务服务处于领导地位  
参与了数据开放、政务大数据以及城市数据平台等多项国家（行业）标准的制定。与多所高校和研究机构成立研究中心和联合实验室。

# 2 资源和成本规划

表 2-1 资源和成本规划

序号	资源名称	配置规格	命名示范	Region	数量	目录价 (包月)
1	云容器引擎	集群规格: cce.s2.small 50 节点	yunpi_c loud1	华北-北京 四, 通用可用区	1	1262.40
2	弹性云服务器	( CCE 节点3台, EMQ 2 台 ) 通用计算增强型   c6s.xlarge.2   8vCPUs   32GiB   系统盘40G  数据 盘500G	yunpi_t est1	华北-北京 四, 通用可用区	3	1142.70
3	弹性云服务器	通用计算增强型   c6s.xlarge.2  16vCPUs   64GiB  系统盘40G  数据 盘1 T	yunpi_h test1	华北-北京 四, 通用可用区	6	2142.30
4	云数据库RDS	主备 8 vCPUs   32 GB	yunpi_r ds	华北-北京 四, 通用可用区	1	2064
5	文档数据库 DDS ( Mon gdb )	副本集 8 vCPUs   32 GB 1T	yunpi_ mdb	华北-北京 四, 通用可用区	1	6010
6	缓存 Redis	主备 redis.ha.xu1.large.r2.1 1G	yunpi_r edis	华北-北京 四, 通用可用区	1	67.50

# 3 实施步骤

- 3.1 Redis的部署
- 3.2 mysql的部署
- 3.3 nginx的部署
- 3.4 activemq安装
- 3.5 MongoDB安装
- 3.6 rocketmq部署
- 3.7 Kafka的安装
- 3.8 其他组件(华为提供)
- 3.9 数据集成平台应用部署
- 3.10 态势感知平台应用部署
- 3.11 组织关联平台应用部署
- 3.12 数据治理平台部署
- 3.13 共享交换平台的部署

## 3.1 Redis 的部署

### redis服务安装:

- 上传redis安装包  
将安装包上传至服务器: 如/opt下  
服务名称: redis  
程序安装路径:/usr/local/redis
- 安装步骤
  - a. 进入/opt/(上传安装包路径)  
命令行界面输入:  

```
cd /opt/
```

图 3-1 进入

```
[root@yjhh opt]# cd /opt/
```

- b. 解压redis安装包

命令行界面输入:

```
tar -zxvf redis-5.0.0.tar.gz
```

图 3-2 解压

```
[root@gx01 opt]# tar -zxvf redis-4.0.7.tar.gz
```

- c. 移动至/usr/local/redis (程序安装路径)

命令行界面输入:

```
mv redis-5.0.0 /usr/local/redis
```

图 3-3 移动

```
[root@gx01 opt]# mv redis-4.0.7 /usr/local/redis
```

- d. 进入redis目录

命令行界面输入:

```
cd /usr/local/redis
```

图 3-4 进入 redis 目录

```
[root@yjhh opt]# cd /usr/local/redis
```

- e. 编译安装redis

命令行界面输入:

```
make && make install
```

图 3-5 编译

```
[root@yjhh redis]# make && make install
```

- f. 打开redis配置文件

命令行界面输入:

```
vim /usr/local/redis/redis.conf
```

图 3-6 打开

```
[root@yjhh redis]# vim /usr/local/redis/redis.conf
```

- g. 修改配置文件

注销掉bind 127.0.0.1此行!

图 3-7 修改 1

```
# JUST COMMENT THE FOLLOWING LINE.  
# ~~~~~  
#bind 127.0.0.1  
# Protected mode is a layer of secu
```

修改protected-mode yes为protected-mode no

图 3-8 修改 2

```
# even if no authentication is configured
# are explicitly listed using the 'requirepass'
protected-mode no

# Accept connections on the specified port:
# port 0 means no port.
# Accept connections on the specified IP address:
```

修改timeout 0为timeout 5000

图 3-9 修改 3

```
# Close the connection after a client is idle for
# timeout seconds. A value of 0 means to wait until
# the client disconnects.
timeout 5000

# TCP keepalive.
#
```

修改daemonize no为daemonize yes

图 3-10 修改 4

```
# By default Redis does not run as a daemon.
# Note that Redis will write a pidfile in /var/run/redis.pid
daemonize yes

# If you run Redis from upstart or systemd:
```

保存退出!

- redis启动  
命令行界面输入:  
首先进入到cd /usr/local/redis 在执行:  
/usr/local/redis/src/redis-server redis.conf  
另一种启动: ./redis-server ../redis.conf

图 3-11 redis 启动

```
[root@yjh redis]# /usr/local/redis/src/redis-server redis.conf
8844:C 16 Jan 2019 16:42:55.188 # oO00o000o000o Redis is starting oO00o000o000o
8844:C 16 Jan 2019 16:42:55.188 # Redis version=5.0.0, bits=64, commit=00000000, modified=0, pid=8844, just started
8844:C 16 Jan 2019 16:42:55.188 # Configuration loaded
```

## 3.2 mysql 的部署

### mysql服务安装:

- 上传mysql安装包

表 3-1 上传 mysql 安装包

安装服务器	gx01节点
程序名称	Mysql
上传路径	/opt
程序安装路径	/usr/local/mysql

- 安装步骤
  - a. 进入/opt/(上传安装包路径)

命令行界面输入:

```
cd /opt/
```

图 3-12 进入

```
[root@yjhh opt]# cd /opt/
```

- b. 解压mysql安装包

命令行界面输入:

```
tar -zxvf mysql-5.6.42-linux-glibc2.12-x86_64.tar.gz
```

图 3-13 解压

```
[root@yjhh opt]# tar -zxvf mysql-5.6.41-linux-glibc2.12-x86_64.tar.gz
```

- c. 移动至/usr/local/mysql (程序安装路径)

命令行界面输入:

```
mv mysql-5.6.42-linux-glibc2.12-x86_64 /usr/local/mysql
```

图 3-14 移动

```
[root@yjhh opt]# mv mysql-5.6.41-linux-glibc2.12-x86_64 /usr/local/mysql
```

- d. 创建mysql用户

命令行界面输入:

```
useradd -s /sbin/nologin mysql
```

图 3-15 创建

```
[root@yjhh opt]# useradd -s /sbin/nologin mysql
```

- e. 进入mysql目录

命令行界面输入:

```
cd /usr/local/mysql
```

图 3-16 进入 mysql 目录

```
[root@gx01 mysql]# cd /usr/local/mysql
```

- f. 建立mysql储存路径并增加mysql用户使用权限

命令行界面输入：

```
mkdir -p /data/mysql ; chown -R mysql:mysql /data/mysql
```

图 3-17 建立

```
[root@yjhh mysql]# mkdir -p /data/mysql ; chown -R mysql:mysql /data/mysql
```

- g. 初始化数据库

```
yum install libaio* -y
```

```
yum -y install numactl.x86_64
```

```
yum -y install perl
```

```
yum -y install perl-devel
```

```
yum -y install perl-Data-Dumper
```

命令行界面输入：

```
./scripts/mysql_install_db --user=mysql --datadir=/data/mysql
```

图 3-18 初始化数据库

```
[root@yjhh mysql]# ./scripts/mysql_install_db --user=mysql --datadir=/data/mysql
```

- h. 复制mysql配置文件

命令行界面输入：

```
cp support-files/my-default.cnf /etc/my.cnf
```

图 3-19 复制

```
[root@yjhh mysql]# cp support-files/my-default.cnf /etc/my.cnf
```

- i. 修改mysql配置文件

命令行界面输入：

```
vim /etc/my.cnf
```

将下面代码加入mysql配置文件的[mysqld]下并保存！

```
basedir=/usr/local/mysql  
datadir=/data/mysql  
socket=/tmp/mysql.sock  
user=mysql  
character-set-server=utf8  
lower_case_table_names=1
```

图 3-20 修改

```
[mysqld]
# sort_buffer_size = 2M
# read_rnd_buffer_size = 2M
basedir=/usr/local/mysql
datadir=/data/mysql
socket=/tmp/mysql.sock
user=mysql
character-set-server=utf8
lower_case_table_names=1
```

- j. 复制mysql启动配置文件

命令行界面输入:

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

图 3-21 复制

```
[root@yjhh mysql]# cp support-files/mysql.server /etc/init.d/mysql
```

- k. 修改mysql启动配置文件

命令行界面输入:

```
vim /etc/init.d/mysql
```

图 3-22 修改

```
[root@yjhh mysql]# vim /etc/init.d/mysql
```

将启动配置文件中代码basedir和datadir修改为以下内容并保存!

```
basedir=/usr/local/mysql
datadir=/data/mysql
```

图 3-23 保存

```
basedir=/usr/local/mysql
datadir=/data/mysql
```

- mysql启动

命令行界面输入:

```
service mysqld start
```

图 3-24 mysql 启动

```
[root@yjhh mysql]# service mysqld start
Starting MySQL. [ 确定 ]
```

- 配置环境变量
  - a. 打开环境变量配置文件

命令行界面输入：

```
vim /etc/profile
```

图 3-25 打开

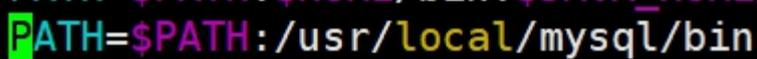


```
[root@yjhh opt]# vim /etc/profile
```

- b. 添加配置文件  
在配置末尾处加入下列代码并保存!  
代码如下：

```
PATH=$PATH:/usr/local/mysql/bin
```

图 3-26 添加配置文件

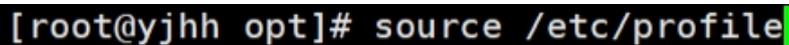


```
PATH=$PATH:/usr/local/mysql/bin
```

- c. 刷新配置文件，使其配置文件生效  
命令行界面输入：

```
source /etc/profile
```

图 3-27 刷新配置文件



```
[root@yjhh opt]# source /etc/profile
```

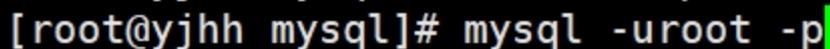
- mysql密码修改和远程访问开启

- a. 登录mysql

命令行界面输入：

```
mysql -uroot -p(mysql5.6初始密码为空，输入后直接回车)
```

图 3-28 登录



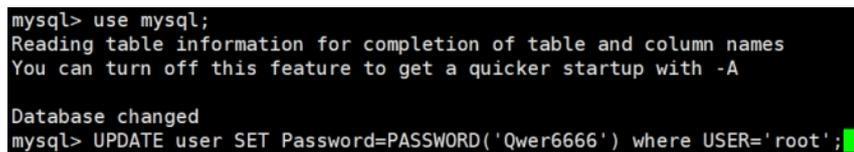
```
[root@yjhh mysql]# mysql -uroot -p
```

- b. 修改mysql密码（下方括号内为所设置的密码）

命令行界面输入：

```
use mysql;  
UPDATE user SET Password=PASSWORD('Qwer6666') where USER='root';
```

图 3-29 修改 mysql 密码



```
mysql> use mysql;  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
  
Database changed  
mysql> UPDATE user SET Password=PASSWORD('Qwer6666') where USER='root';
```

- c. 开启远程访问  
命令行界面输入：

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'Qwer6666' WITH GRANT OPTION;  
FLUSH PRIVILEGES;
```

图 3-30 开启远程访问

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'Qwer6666' WITH GRANT OPTION;
```

- 设置mysql开机启动
  - a. 设置mysql开机启动  
命令行界面输入：  
chkconfig mysqld on
  - b. 验证开机启动是否设置成功  
命令行界面输入：  
chkconfig --list | grep mysqld (如2, 3, 4, 5为开则表示设置成功)

图 3-31 验证

```
[root@yjhh mysql]# chkconfig --list | grep mysqld  
  
注：该输出结果只显示 SysV 服务，并不包含  
原生 systemd 服务。SysV 配置数据  
可能被原生 systemd 配置覆盖。  
  
要列出 systemd 服务，请执行 'systemctl list-unit-files'。  
查看在具体 target 启用的服务请执行  
'systemctl list-dependencies [target]'。  
  
mysqld          0:关    1:关    2:开    3:开    4:开    5:开    6:关
```

## 3.3 nginx 的部署

### Nginx的服务安装：

- 安装步骤  
上传包到/opt/下  
第一步：安装依赖  
yum install -y gcc-c++  
yum install -y pcre pcre-devel  
yum install -y zlib zlib-devel  
yum install -y openssl openssl-devel  
下载nginx安装包：<http://nginx.org/download/>  
将下载的nginx源码包上传到linux服务器上，解压

图 3-32 下载

```
total 868  
drwxr-xr-x. 2 root root    6 Apr 11 2018 bin  
drwxr-xr-x. 2 root root    6 Apr 11 2018 etc  
drwxr-xr-x. 2 root root    6 Apr 11 2018 games  
drwxr-xr-x. 2 root root    6 Apr 11 2018 include  
drwxr-xr-x. 2 root root    6 Apr 11 2018 lib  
drwxr-xr-x. 2 root root    6 Apr 11 2018 lib64  
drwxr-xr-x. 2 root root    6 Apr 11 2018 libexec  
drwxr-xr-x. 9 1001 1001 186 Mar  5 22:01 nginx-1.9.9  
-rw-r--r--. 1 root root 887908 Mar  5 22:00 nginx-1.9.9.tar.gz  
drwxr-xr-x. 2 root root    6 Apr 11 2018 sbin  
drwxr-xr-x. 5 root root    49 Mar 16 2019 share  
drwxr-xr-x. 3 root root    60 Mar 16 2019 src  
[root@mysql local]#
```

- Nginx配置

```
cd nginx-1.9.9  
./configure --prefix=/usr/local/nginx
```

configure完成之后，会有如下信息，诸如日志文件，配置文件啥的

图 3-33 Nginx 配置

```
Configuration summary  
+ using system PCRE library  
+ OpenSSL library is not used  
+ md5: using system crypto library  
+ sha1: using system crypto library  
+ using system zlib library  
  
nginx path prefix: "/usr/local/nginx"  
nginx binary file: "/usr/local/nginx/sbin/nginx"  
nginx configuration prefix: "/usr/local/nginx/conf"  
nginx configuration file: "/usr/local/nginx/conf/nginx.conf"  
nginx pid file: "/usr/local/nginx/logs/nginx.pid"  
nginx error log file: "/usr/local/nginx/logs/error.log"  
nginx http access log file: "/usr/local/nginx/logs/access.log"  
nginx http client request body temporary files: "client_body_temp"  
nginx http proxy temporary files: "proxy_temp"  
nginx http fastcgi temporary files: "fastcgi_temp"  
nginx http uwsgi temporary files: "uwsgi_temp"  
nginx http scgi temporary files: "scgi_temp"
```

- 编译安装

```
make && make install
```

nginx安装成功

图 3-34 编译安装

```
total 600  
drwxr-xr-x. 2 root root    6 Apr 11 2018 bin  
drwxr-xr-x. 2 root root    6 Apr 11 2018 etc  
drwxr-xr-x. 2 root root    6 Apr 11 2018 games  
drwxr-xr-x. 2 root root    6 Apr 11 2018 include  
drwxr-xr-x. 2 root root    6 Apr 11 2018 lib  
drwxr-xr-x. 2 root root    6 Apr 11 2018 lib64  
drwxr-xr-x. 2 root root    6 Apr 11 2018 libexec  
drwxr-xr-x. 6 root root  54 Mar  5 22:11 nginx  
-rw-r--r--. 1 root root 887908 Mar  5 22:00 nginx-1.9.9.tar.gz  
drwxr-xr-x. 2 root root    6 Apr 11 2018 sbin  
drwxr-xr-x. 5 root root   49 Mar 16 2019 share  
drwxr-xr-x. 3 root root   60 Mar 16 2019 src  
[root@mysql local]#
```

- 启动nginx

先检测nginx的配置是否正确

```
./nginx/sbin/nginx -t
```

图 3-35 启动 nginx

```
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok  
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
```

如果出现上面两句话，说明nginx配置ok，可以启动。

## 3.4 activemq 安装

- 上传activemq安装包

表 3-2 上传 activemq 安装包

安装服务器	gx01节点
程序名称	activemq
上传路径	/opt/
程序安装路径	usr/local/activemq

- 安装步骤

在gx01节点、gx02和gx03三个节点对应的服务器分别执行下列步骤。

- a. 进入opt目录，命令行界面输入：

```
cd /opt/
```

- b. 解压apache-activemq-5.15.8-bin.tar.gz安装包

```
tar -zxvf apache-activemq-5.15.8-bin.tar.gz
```

- c. 移动至/usr/local/activemq

```
mv apache-activemq-5.15.8 /usr/local/activemq
```

图 3-36 移动

```
[root@slave1 opt]# mv apache-activemq-5.15.8 /usr/local/activemq
```

- activemq启动

进入activemq的bin目录，执行启动命令！

命令行界面输入：

```
cd /usr/local/activemq/bin
```

图 3-37 进入

```
[root@slave1 bin]# cd /usr/local/activemq/bin
```

```
./activemq start
```

图 3-38 启动

```
[root@master bin]# ./activemq start
INFO: Loading '/usr/local/activemq/bin/env'
INFO: Using java '/usr/local/jdk/bin/java'
INFO: Starting - inspect logfiles specified in logging.properties and log4j.properties to get details
INFO: pidfile created: '/usr/local/activemq/data/activemq.pid' (pid '22319')
```

- activemq状态查看

进入activemq的bin目录，执行启动命令！

命令行界面输入：

```
cd /usr/local/activemq/bin
./activemq status
```

- activemq关闭

进入activemq的bin目录，执行启动命令！

命令行界面输入：

```
cd /usr/local/activemq/bin  
./activemq stop
```

## 3.5 MongoDB 安装

- 上传MongoDB安装包

表 3-3 上传 MongoDB 安装包

安装服务器	gx01节点、gx02节点、gx03节点
程序名称	MongoDB
上传路径	/opt/
程序安装路径	/usr/local/mongo

- 安装步骤

在gx01节点、gx02和gx03三个节点对应的服务器分别执行下列步骤。

- a. 进入opt目录，命令行界面输入：  

```
cd /opt/
```

图 3-39 进入 opt 目录

```
[root@gx01 ~]# cd /opt/
```

- b. 解压mongodb-linux-x86\_64-4.0.8.tgz安装包；命令行界面输入：  

```
tar -zxvf mongodb-linux-x86_64-4.0.8.tgz
```

图 3-40 解压

```
[root@master opt]# tar -zxvf mongodb-linux-x86_64-4.0.8.tgz
```

- c. 移动至/usr/local/mongodb（程序安装路径）；命令行界面输入：  

```
mv mongodb-linux-x86_64-4.0.8 /usr/local/mongodb
```

图 3-41 移动

```
[root@gx01 opt]# mv mongodb-linux-x86_64-4.0.8 /usr/local/mongodb
```

- d. 创建data目录和logs目录；命令行界面输入：  

```
mkdir -p /usr/local/mongodb/{data,logs}
```

图 3-42 创建

```
[root@slave1 opt]# mkdir -p /usr/local/mongo/config/{data,logs}
```

- e. 配置config server

在/usr/local/mongodb目录下新建config实例的启动参数文件,并启动实例；  
命令行界面输入：

```
cd /usr/local/mongodb
```

图 3-43 配置 1

```
[root@gx01 mongodb]# cd /usr/local/mongodb
```

命令行界面输入：

```
vim mongodb.config
```

图 3-44 配置 2

```
[root@wlsj1 mongodb]# vim mongodb.config
```

将下面代码分别加入3个mongodb配置文件下并保存

```
dbpath=/usr/local/mongodb/data  
logpath=/usr/local/mongodb/logs/mongodb.log  
port=27017  
fork=true  
replSet=configRS  
configsvr=true  
bind_ip=0.0.0.0
```

- 启动实例

命令行界面输入：

```
cd /usr/local/mongodb/bin  
./mongod -f ../mongodb.config
```

- MongoDB启动

命令行界面输入：

```
cd /usr/local/mongodb/bin  
./mongod -f ../mongodb.config
```

- MongoDB状态查看

查看MongoDB服务运行状态

```
ps -ef |grep mongo
```

- MongoDB关闭

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

## 3.6 rocketmq 部署

- 上传rocketmq包

表 3-4 上传 rocketmq 包

安装服务器	slaver1 节点, slaver2 节点
程序名称	rocketmq-all-4.3.2-bin-release.zip
上传路径	/opt/
程序安装路径	/usr/local/rocketmq

- 安装步骤

a. 首先将rocketmq安装包上传到服务器/opt目录，并解压，slaver1 节点，slaver2 节点操作；命令行界面输入：

```
unzip rocketmq-all-4.3.2-bin-release.zip
```

b. 将解压后的文件移动至/usr/local目录下，slaver1 节点，slaver2 节点操作；命令行界面输入：

```
mv rocketmq-all-4.3.2-bin-release /usr/local/rocketmq
```

c. 上传修改broker-a.conf配置文件，slaver1节点

图 3-45 节点 1

```
# limitations under the License.
brokerClusterName=rocketmq-master
brokerName=broker-a
brokerId=0
namesrvAddr=192.168.32.131:9876;192.168.32.130:9876
brokerIP1=192.168.32.131
deleteWhen=04
fileReservedTime=48
storePathRootDir=/usr/local/rocketmq/store
storePathCommitLog=/usr/local/rocketmq/store/commitlog
storePathConsumeQueue=/usr/local/rocketmq/store/consumequeue
storePathIndex=/usr/local/rocketmq/store/index
storeCheckpoint=/usr/local/rocketmq/store/checkpoint
abortFile=/usr/local/rocketmq/store/abort
brokerRole=ASYNC_MASTER
flushDiskType=ASYNC_FLUSH
#是否允许 Broker 自动创建Topic，建议线下开启，线上关闭
autoCreateTopicEnable=true
#是否允许 Broker 自动创建订阅组，建议线下开启，线上关闭
autoCreateSubscriptionGroup=true
#Broker 对外服务的监听端口
listenPort=10911
```

Slaver2节点:

图 3-46 节点 2

```
# See the License for the specific language governing permissions and
# limitations under the License.
brokerClusterName=rocketmq-master
brokerName=broker-b
brokerId=1
namesrvAddr=192.168.32.131:9876;192.168.32.130:9876
brokerIP1=192.168.32.130
deleteWhen=04
fileReservedTime=48
storePathRootDir=/usr/local/rocketmq/store
storePathCommitLog=/usr/local/rocketmq/store/commitlog
storePathConsumeQueue=/usr/local/rocketmq/store/consumequeue
storePathIndex=/usr/local/rocketmq/store/index
storeCheckpoint=/usr/local/rocketmq/store/checkpoint
abortFile=/usr/local/rocketmq/store/abort
brokerRole=ASYNC_MASTER
flushDiskType=ASYNC_FLUSH
#是否允许 Broker 自动创建Topic，建议线下开启，线上关闭
autoCreateTopicEnable=true
#是否允许 Broker 自动创建订阅组，建议线下开启，线上关闭
autoCreateSubscriptionGroup=true
#Broker 对外服务的监听端口
listenPort=10911
```

d. 配置rocketmq 的环境变量，slaver1 节点，slaver2 节点操作  
vim /etc/profile

在后面添加

```
#设置rocketmq的环境变量
export ROCKETMQ_HOME=/usr/local/rocketmq
export PATH=$JAVA_HOME/bin:$ROCKETMQ_HOME/bin:$PATH
```

按esc+!wq保存

使rocketmq的配置生效

```
source /etc/profile
```

- rocketmq停止

命令行界面输入:

```
nohup sh mqshutdown namesrv
nohup sh mqshutdown broker
```

- rocketmq启动

到你的安装路径下启动两台机器的NameServer：先启动两台机器的NameServer，再启动两台机器的Borker，关机的时候顺序相反，先关闭两台机器的Broker，再关闭两台机器的Nameserver。

命令行输入：

```
cd /usr/local/rocketmq/
```

启动name server，执行

```
nohup sh bin/mqnamesrv &
```

日志打印：

```
tail -f nohup.out
```

图 3-47 rocketmq 启动

```
[root@slaver1 rocketmq]# nohup sh bin/mqnamesrv &
[+] 4918
[root@slaver1 rocketmq]# nohup: ignoring input and appending output to 'nohup.out'
[root@slaver1 rocketmq]# tail -f nohup.out
Java HotSpot(TM) 64-bit Server VM warning: using the DefNew young collector with the CMS collector is deprecated and will likely be removed in a future release
Java HotSpot(TM) 64-bit Server VM warning: useCMSCompactAtFullCollection is deprecated and will likely be removed in a future release.
The Name Server boot success. serializeType=300
```

启动broker-a.properties 执行：

```
nohup sh /usr/local/rocketmq/bin/mqbroker -c /usr/local/rocketmq/conf/2m-noslave/broker-a.properties &
```

启动broker-b.properties 执行：

```
nohup sh /usr/local/rocketmq/bin/mqbroker -c /usr/local/rocketmq/conf/2m-noslave/broker-b.properties &
```

## 3.7 Kafka 的安装

### Kafka 集群安装

- 上传confluent安装包

表 3-5 上传 confluent 安装包

安装服务器	gx01节点、gx02节点、gx03节点
程序名称	Kafka
上传路径	/opt/
程序安装路径	/usr/local/confluent

- 安装步骤

在gx01节点、gx02和gx03三个节点对应的服务器分别执行第一步至第六步的步骤。

- a. 进入confluent-5.0.0.tar安装包的/opt/目录；命令行界面输入：  

```
cd /opt/
```

图 3-48 进入

```
[root@gx01 ~]# cd /opt/
```

- b. 解压confluent-5.0.0.tar安装包；命令行界面输入：  

```
tar -xvf confluent-5.0.0.tar
```

图 3-49 解压

```
[root@gx01 opt]# tar -xvf confluent-5.0.0.tar
```

- c. 将解压后的文件移动至/usr/local/confluent目录下；命令行界面输入：

```
mv confluent-5.0.0 /usr/local/confluent
```

图 3-50 移动

```
[root@gx01 opt]# mv confluent-5.0.0 /usr/local/confluent
```

- d. 进入/usr/local/confluent目录下；命令行界面输入：  
cd /usr/local/confluent/

图 3-51 进入

```
[root@gx01 local]# cd /usr/local/confluent/  
[root@gx01 confluent]# ls  
bdc h.json bin etc lib logs README share src test
```

- e. zookeeper集群配置，编辑zookeeper.properties配置文件；命令行界面输入：  
cd etc/kafka/  
vim zookeeper.properties

图 3-52 配置文件 1

```
[root@gx01 confluent]# cd etc/kafka/
```

图 3-53 配置文件 2

```
[root@gx01 kafka]# vim zookeeper.properties
```

- f. 更改zookeeper.properties配置文件的参数；命令行界面输入：  
server.1=IP1:2888:3888  
server.2=IP2:2888:3888  
server.3=IP3:2888:3888

此处配置的IP1、IP2、IP3分别为gx01、gx02、gx03三个节点的服务器地址。

图 3-54 配置文件参数

```
server.1=10.168.1.120:2888:3888  
server.2=10.168.1.126:2888:3888  
server.3=10.168.1.103:2888:3888
```

- g. kafka集群配置，编辑server.properties配置文件；命令行界面输入：  
vim server.properties

图 3-55 编辑

```
[root@gx01 kafka]# vim server.properties
```

- h. 更改server.properties配置文件的参数  
**1# 集群中各节点broker.id必须唯一**  
broker.id=0

图 3-56 更改 1

```
#gx01节点      #gx02节点      #gx03节点  
broker.id=0    broker.id=1    broker.id=2
```

**2# host.name和 advertised.listeners为各节点的主机IP**

```
host.name=【本机IP】  
advertised.listeners=PLAINTEXT://【本机IP】:9092
```

图 3-57 修改 2

```
host.name=10.168.1.120  
advertised.listeners=PLAINTEXT://10.168.1.120:9092
```

图 3-58 修改 3

```
host.name=10.168.1.126  
advertised.listeners=PLAINTEXT://10.168.1.126:9092
```

图 3-59 修改 4

```
host.name=10.168.1.103  
advertised.listeners=PLAINTEXT://10.168.1.103:9092
```

### 3# 修改zookeeper.connect为zookeeper集群配置地址，三个节点修改的一致

```
zookeeper.connect=IP1:2181,IP2:2181,IP3:2181
```

图 3-60 修改 5

```
zookeeper.connect=10.168.1.120:2181,10.168.1.126:2181,10.168.1.103:2181
```

:wq保存退出!

#### i. schema-registry 集群配置；命令行界面输入：

```
cd ../../etc/schema-registry/  
vim schema-registry.properties
```

图 3-61 配置

```
[root@qx01 kafka]# cd ../../etc/schema-registry/  
[root@qx01 schema-registry]# ls  
connect-avro-distributed.properties connect-avro-standalone.properties log4j.properties schema-registry.properties  
[root@qx01 schema-registry]# vim schema-registry.properties
```

#### j. 更改schema-registry.properties配置文件

##### 1# 修改listeners为各节点主机IP

```
listeners=http://【本机IP】:8086
```

图 3-62 更改 1

```
listeners=http://10.168.1.120:8086
```

```
listeners=http://10.168.1.126:8086
```

```
listeners=http://10.168.1.103:8086
```

##### 2# 修改kafkastore.connection.url为三个节点的主机IP，三个节点修改的一致

```
kafkastore.connection.url=IP1:2181,IP2:2181,IP3:2181
```

图 3-63 更改 2

```
kafkastore.connection.url=10.168.1.120:2181,10.168.1.126:2181,10.168.1.103:2181
```

:wq保存退出!

- k. kafka connect集群配置；命令行界面输入：

```
vim connect-avro-distributed.properties
```

图 3-64 更改 3

```
[root@gx01 schema-registry]# vim connect-avro-distributed.properties
```

- l. 更改connect-avro-distributed.properties配置文件

1# 修改bootstrap.servers为三个节点的主机IP，三个节点修改的一致

```
bootstrap.servers=IP1:9092,IP2:9092,IP3:9092
```

图 3-65 主机 IP1

```
bootstrap.servers=10.168.1.120:9092,10.168.1.126:9092,10.168.1.103:9092
```

2# 修改key.converter.schema.registry.url为三个节点的主机IP，三个节点修改的一致

```
key.converter.schema.registry.url=http://IP1:8086,http://IP2:8086,http://IP3:8086
```

图 3-66 主机 IP2

```
key.converter.schema.registry.url=http://10.168.1.120:8086,http://10.168.1.126:8086,http://10.168.1.103:8086
```

3# 修改value.converter.schema.registry.url为三个节点的主机IP，三个节点修改的一致

```
value.converter.schema.registry.url=http://IP1:8086,http://IP2:8086,http://IP3:8086
```

图 3-67 主机 IP3

```
value.converter.schema.registry.url=http://10.168.1.120:8086,http://10.168.1.126:8086,http://10.168.1.103:8086
```

4# 修改rest.host.name为各节点的主机IP

```
rest.host.name=【本机IP】
```

图 3-68 主机 IP4

```
rest.host.name=10.168.1.120
```

```
rest.host.name=10.168.1.126
```

```
rest.host.name=10.168.1.103
```

:wq保存退出！

- m. kafka rest 配置

命令行界面输入：

```
cd ../kafka-rest/  
vim kafka-rest.properties
```

图 3-69 配置

```
[root@gx01 schema-registry]# cd ../kafka-rest/  
[root@gx01 kafka-rest]# ls  
kafka-rest.properties log4j.properties  
[root@gx01 kafka-rest]# vim kafka-rest.properties
```

n. 更改kafka-rest.properties配置文件

1# 修改host.name为各节点主机IP

```
host.name= [ 本机IP ]
```

图 3-70 更改主机 IP1

```
host.name=10.168.1.120  
host.name=10.168.1.126  
host.name=10.168.1.103
```

2# 修改schema.registry.url为三个节点的主机IP，三个节点修改的一致

```
schema.registry.url=http://IP1:8086,http://IP2:8086,http://IP3:8086
```

图 3-71 更改主机 IP2

```
schema.registry.url=http://10.168.1.120:8086,http://10.168.1.126:8086,http://10.168.1.103:8086
```

3# 修改bootstrap.servers为三个节点的主机IP，三个节点修改的一致

```
bootstrap.servers=IP1:9092,IP2:9092,IP3:9092
```

图 3-72 更改主机 IP3

```
bootstrap.servers=10.168.1.120:9092,10.168.1.126:9092,10.168.1.103:9092
```

:wq保存退出!

o. 创建myid文件

在三个节点/home/confluent/confluent.data/zookeeper/data目录下，创建myid文件并写入对应编号；命令行界面输入：

```
--没有路径创建路径；mkdir -p /home/confluent/confluent.data/zookeeper/data  
cd /home/confluent/confluent.data/zookeeper/data/
```

执行：touch myid

```
vim myid
```

图 3-73 创建 myid 文件

```
[root@gx01 data]# cd /home/confluent/confluent.data/zookeeper/data/  
[root@gx01 data]# cat myid  
1  
  
[root@gx02 bin]# cd /home/confluent/confluent.data/zookeeper/data/  
[root@gx02 data]# cat myid  
2  
  
[root@gx03 bin]# cd /home/confluent/confluent.data/zookeeper/data/  
[root@gx03 data]# cat myid  
3
```

• Confluent启动

分别进入三个节点主机的 cd /usr/local/confluent/bin/目录下；命令行界面输入：

```
./confluent start schema-registry
```

图 3-74 Confluent 启动

```
[root@gx01 bin]# ./confluent start schema-registry  
[root@gx02 bin]# ./confluent start schema-registry  
[root@gx03 bin]# ./confluent start schema-registry
```

- Confluent状态查看  
分别进入三个节点主机的/usr/local/confluent/bin/目录下；命令行界面输入：  
./confluent status  
启动命令  
./confluent start schema-registry  
5.5 版本启动  
./confluent local start schema-registry
- Confluent关闭  
分别进入三个节点主机的/usr/local/confluent/bin/目录下；命令行界面输入：  
./confluent stop schema-registry

### 3.8 其他组件(华为提供)

表 3-6 其他组件

文档数据库服务DDS
对象存储服务OBS
弹性文件服务SFS
MRS服务
分布式消息服务 DMS Kafka版
数据仓库服务DWS

### 3.9 数据集成平台应用部署

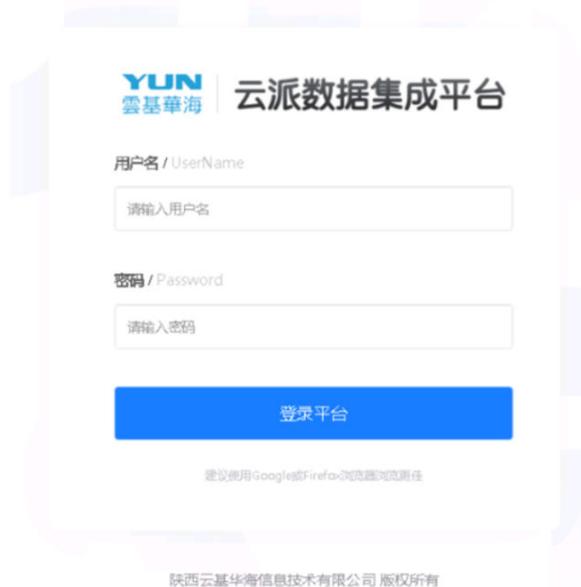
1. 将部署文件上传到服务器/usr/local/下，解压target4.0.tar  
tar -zxvf target4.0.tar  
说明：需要java环境  
检查环境：java -vrsion

图 3-75 解压

```
[root@redis bin]# java -version  
java version "1.8.0_191"  
Java(TM) SE Runtime Environment (build 1.8.0_191-b12)  
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)  
[root@redis bin]#
```

2. 进入/usr/local/target4.0/bin下启动服务；启动命令：  
nohup ./streamsets dc
3. 访问界面如下图所示，http://ip:18630

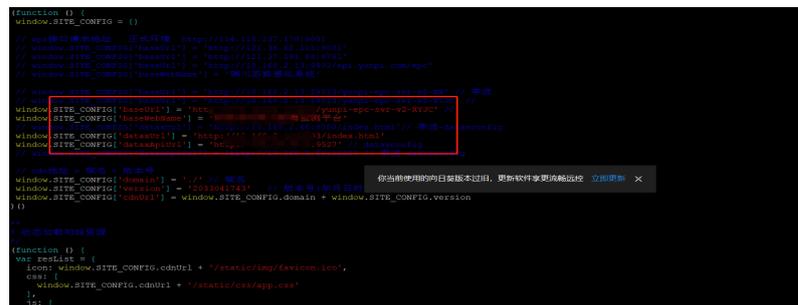
图 3-76 访问



### 3.10 态势感知平台应用部署

1. 前端应用部署：  
将前端包上传至服务器  
/usr/local/tsgz
2. 修改配置文件进入config下修改配置文件  
vim index.js

图 3-77 修改配置文件 1



后端包的配置与修改共4个后端jar服务器包。

- sx-credit-monitor-msvs.jar
- tsgz-gate-msvs.jar
- tsgz-reg-msvs.jar
- yunpi-epc-svr.jar

通过修改四个包的配置文件如：vim application-xyjc.yml将连接数据库信息进行修改置后完成。

图 3-78 修改配置文件 2

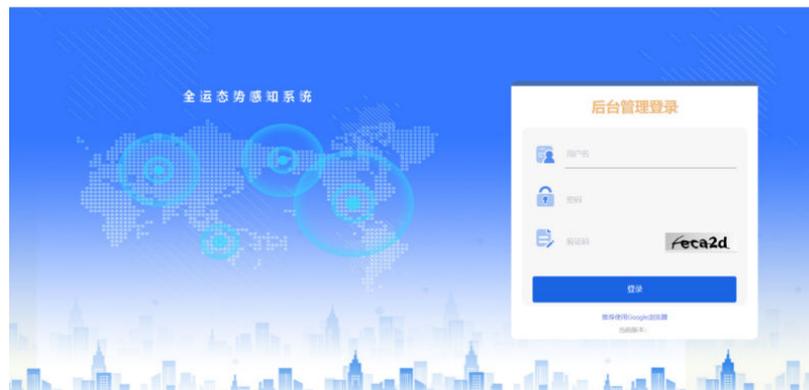
```
多数据源的配置
datasource:
  type: com.alibaba.druid.pool.DruidDataSource
  druid:
    first:
      driver-class-name: com.mysql.cj.jdbc.Driver
      url: jdbc:mysql://10.10.10.3306/tsgz_v2_xyjc?useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai
      username: root
      password: root
    second:
      url: jdbc:postgresql://10.10.10.32/dws_sx_ioc_v2.0?useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai
      driver-class-name: org.postgresql.Driver
      username: postgres
      password: postgres
    third:
      url: jdbc:postgresql://10.10.10.33/dws_sxjdsj_ioc_v3.0?useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai
      driver-class-name: org.postgresql.Driver
      username: postgres
      password: postgres
    four:
```

服务启动:

```
Nohup java -jar yunpi-epc-svr.jar --spring.config.location=application-xyjc.yml
Nohup java -jar sx-credit-monitor-msvs.jar --spring.config.location=application-sxprod.yml
Nohup java -jar tsgz-reg-msvs.jar --spring.profiles.active=tsgz
Nohup java -jar tsgz-gate-msvs.jar --spring.profiles.active=tsgz
```

访问界面: http:ip:8083进行访问

图 3-79 访问



### 3.11 组织关联平台应用部署

1. 将前端上传至服务器/usr/local/web下前端包  
yunpi-base-mgr  
修改配置信息将请求后端网关地址进行修改  
Vim config.js
2. 将后端包上传至服务器/usr/local/web下

图 3-80 上传

```
/*M
 * @Description: ^M
 * @Author: wyb^M
 * @Date: 2021-05-21 09:58:35^M
 * @LastEditTime: 2021-09-03 15:00:42^M
 */^M
/**^M
 * 系统全局配置文件^M
 * @prama {String} apiUrl 后台网管地址^M
 * @prama {String} wsUrl 消息接口地址^M
 * @prama {String} govern_web_url 跳转数据治理的data-go项目部署地址^M
 * @prama {String} log_web_url 跳转日志管理log-manage项目部署地址^M
 */^M
(function () {^M
// 开发环境参数配置^M
window.dev = {^M
// apiUrl: "http://124.70.75.233:9026",
apiUrl: "http://10.168.2.39:9088",
// apiUrl: "http://10.168.2.39:9088",
wsUrl: "http://10.168.2.39:9402/websocket/",
govern_web_url: "http://10.168.2.39:8082/",
GX_web_url: "http://10.168.2.11:8085/"
};^M
^M
window.config = {^M
logoTitle: "云派数据治理平台", // 登录标题^M
headerTitle: '基础管理系统', //基础管理头部标题^M
headerLogo: 'iconYUNPI-LOGO' //系统logo
};^M
})();^M
^M
```

- 3. 服务的配置修改及启动  
修改所有jar服务连接nacos配置的信息如下标红处。

图 3-81 修改 1

```
rw-r--r-- 2 root root 22 4月 27 2022 log
rw-r--r-- 3 root root 55 4月 27 2022 logs
rw-r--r-- 7 root root 96 4月 25 2022 nacos
rw-r--r-- 1 root root 771 10月 28 2022 run-jar.sh
rw-r--r-- 1 root root 82597574 4月 25 2022 yunpi-base-auth.jar
rw-r--r-- 1 root root 651264 4月 20 09:21 yunpi-base-auth.log
rw-r--r-- 1 root root 86806967 9月 29 2022 yunpi-base-gateway.jar
rw-r--r-- 1 root root 24576 6月 15 08:52 yunpi-base-gateway.log
rw-r--r-- 3 root root 24 4月 27 2022 yunpi-base-mge
rw-r--r-- 1 root root 92928161 4月 25 2022 yunpi-base-system.jar
rw-r--r-- 1 root root 989312 4月 28 09:20 yunpi-base-system.log
rw-r--r-- 1 root root 1 5月 19 2022 yunpi-base-system.log
rw-r--r-- 1 root root 28486356 10月 25 2022 yunpi-bdb-msvs.log
rw-r--r-- 1 root root 77196531 4月 25 2022 yunpi-bde-msvc.jar
rw-r--r-- 1 root root 17574 3月 22 11:00 yunpi-bde-msvc.log
rw-r--r-- 1 root root 3649536 10月 25 2022 yunpi-daq-msvs.log
rw-r--r-- 1 root root 373384 10月 11 2022 yunpi-dg-msvs.log
rw-r--r-- 1 root root 45983 10月 25 2022 yunpi-dp-msvs.log
rw-r--r-- 1 root root 64355 10月 11 2022 yunpi-dqd-msvs.log
rw-r--r-- 1 root root 118776585 4月 25 2022 yunpi-dsa-msvs.jar
rw-r--r-- 1 root root 35958592 6月 15 08:52 yunpi-dsa-msvs.log
rw-r--r-- 1 root root 135155891 4月 25 2022 yunpi-dwh-msvs.jar
rw-r--r-- 1 root root 45856 4月 28 09:00 yunpi-dwh-msvs.log
rw-r--r-- 1 root root 138176687 4月 25 2022 yunpi-mdm-msvs.jar
rw-r--r-- 1 root root 1589248 6月 15 00:00 yunpi-mdm-msvs.log
rw-r--r-- 1 root root 146888265 4月 25 2022 yunpi-md-msvs.jar
rw-r--r-- 1 root root 31629312 6月 15 00:00 yunpi-md-msvs.log
```

图 3-82 修改 2

```
server:
  port: 5002

productName: S3ZL
#系统提供给用户的重置密码
userResetPwd: YunSP19

# Spring
spring:
  application:
    # 应用名称
    name: yunpi-system
  profiles:
    # 环境配置
    active: dev
  cloud:
    nacos:
      discovery:
        namespace: ${productName}
        group: ${spring.profiles.active}
        # 服务注册地址
        #server-addr: ${NACOS_HOST1:10.168.2.11}:${NACOS_PORT:8848},${NACOS_HOST2:10.168.2.12}:${NACOS_PORT:8848},${NACOS_HOST3:10.168.2.13}:${NACOS_PORT:8848}
        server-addr: 10.168.2.39:8848
        config:
          namespace: ${productName}
          # ${spring.profiles.active}
          # 配置中心地址
          #server-addr: ${NACOS_HOST1:10.168.2.11}:${NACOS_PORT:8848},${NACOS_HOST2:10.168.2.12}:${NACOS_PORT:8848},${NACOS_HOST3:10.168.2.13}:${NACOS_PORT:8848}
          server-addr: 10.168.2.39:8848
          # 配置文件格式
          file-extension: yaml
          # 共享配置
          shared-configs:
            - group: ${spring.profiles.active}
              dataId: application-base-${spring.profiles.active}.${spring.cloud.nacos.config.file-extension}
            - refresh:

```

- 4. 服务的启动  
nohup java -jar yunpi-base-gateway.jar >> yunpi-base-gateway.log  
nohup java -jar yunpi-base-auth.jar >> yunpi-auth.log  
nohup java -jar yunpi-base-system.jar>> yunpi-base.log

```
nohup java -jar yunpi-bde-msvc.jar >> yunpi-bde.log  
nohup java -jar yunpi-xxl.jar >> yunpi-xxl.log  
nohup java -jar yunpi-dsm-msvs.jar >> yunpi-dsm.log  
nohup java -jar yunpi-dwh-msvs.jar >> yunpi-dwh.log  
nohup java -jar yunpi-mdm-msvs.jar>> yunpi-mdm.log  
nohup java -jar yunpi-md-msvs.jar >> yunpi-md.log  
nohup java -jar yunpi-file-msvs.jar>> yunpi-file.log  
nohup java -jar yunpi-datax-msvs.jar>> yunpi-datax.log
```

5. 访问界面: <http://ip:8084>

图 3-83 访问界面



6. 服务的停止:  
Ps -ef|grep jar服务包名称  
Kill -9 进程号

## 3.12 数据治理平台部署

1. 将前端上传至服务器/usr/local/web下两个前端包  
yunpi-base-mgr yunpi-data-govern
2. 将后端包上传至服务器  
修改配置信息将请求后端网关地址进行修改  
Vim config.js

图 3-84 上传

```
/*M
 * @Description: ^M
 * @Author: wyb^M
 * @Date: 2021-05-21 09:58:35^M
 * @LastEditTime: 2021-09-03 15:00:42^M
 */^M
/**^M
 * 系统全局配置文件^M
 * @prama {String} apiUrl 后台网管地址^M
 * @prama {String} wsUrl 消息接口地址^M
 * @prama {String} govern_web_url 跳转数据治理的data-go项目部署地址^M
 * @prama {String} log_web_url 跳转日志管理log-manage项目部署地址^M
 */^M
(function () {^M
 // 开发环境参数配置^M
 window.dev = {^M
 // apiUrl: "http://124.70.75.233:9026",
 apiUrl: "http://10.10.10.1:39:9088",
 // apiUrl: "http://10.10.10.1:39:9088",
 wsUrl: "http://10.10.10.1:39:9402/websocket/",
 govern_web_url: "http://10.10.10.1:39:8082/",
 GX_web_url: "http://10.10.10.1:211:8085/"
 };^M
 ^M
 window.config = {^M
 logoTitle: "云派数据治理平台", //登录标题^M
 headerTitle: '基础管理系统', //基础管理头部标题^M
 headerLogo: 'iconYUNPI-LOGO' //系统Logo
 };^M
 })(); ^M
 ^M
```

3. /usr/usr/local/jar下修改jar服务配置文件

图 3-85 修改配置文件

```
rwxf--xf-x 2 root root 22 4月 27 2022 Log
rwxf--xf-x 3 root root 55 4月 27 2022 logs
rwxf--xf-x 7 root root 96 4月 25 2022 nacos
rwxf--xf-x 1 root root 771 10月 28 2022 run-jar.sh
rw-f--f-- 1 root root 82597574 4月 25 2022 yunpi-base-auth.jar
rw-f--f-- 1 root root 651264 4月 20 09:21 yunpi-base-auth.log
rw-f--f-- 1 root root 86806967 9月 29 2022 yunpi-base-gateway.jar
rw-f--f-- 1 root root 24576 6月 15 08:52 yunpi-base-gateway.log
rwxf--xf-x 3 root root 24 4月 27 2022 yunpi-base-mge
rw-f--f-- 1 root root 92928161 4月 25 2022 yunpi-base-system.jar
rw-f--f-- 1 root root 909312 4月 20 09:20 yunpi-base-system.log
rw-f--f-- 1 root root 1 5月 19 2022 yunpi-base-system.loh
rw-f--f-- 1 root root 28406356 10月 25 2022 yunpi-bdb-msvs.log
rw-f--f-- 1 root root 77196531 4月 25 2022 yunpi-bde-msvc.jar
rw-f--f-- 1 root root 17574 3月 22 11:00 yunpi-bde-msvc.log
rw-f--f-- 1 root root 3649536 10月 25 2022 yunpi-dq-msvs.log
rw-f--f-- 1 root root 373384 10月 11 2022 yunpi-dq-msvs.log
rw-f--f-- 1 root root 45963 10月 25 2022 yunpi-dp-msvs.log
rw-f--f-- 1 root root 64355 10月 11 2022 yunpi-dq-d-msvs.log
rw-f--f-- 1 root root 118776585 4月 25 2022 yunpi-dsm-msvs.jar
rw-f--f-- 1 root root 35950592 6月 15 08:52 yunpi-dsm-msvs.log
rw-f--f-- 1 root root 135155091 4月 25 2022 yunpi-dwh-msvs.jar
rw-f--f-- 1 root root 45966 4月 20 09:00 yunpi-dwh-msvs.log
rw-f--f-- 1 root root 138176667 4月 25 2022 yunpi-mdm-msvs.jar
rw-f--f-- 1 root root 1589248 6月 15 00:00 yunpi-mdm-msvs.log
rw-f--f-- 1 root root 146808265 4月 25 2022 yunpi-md-msvs.jar
rw-f--f-- 1 root root 31629312 6月 15 00:00 yunpi-md-msvs.log
```

4. 服务的配置修改及启动

修改所有jar服务连接nacos配置的信息如下标红处。

图 3-86 修改及启动

```
server:
  port: 9202

productName: SJZL
#系统提供给用户的重置密码
userResetPwd: YunSP199

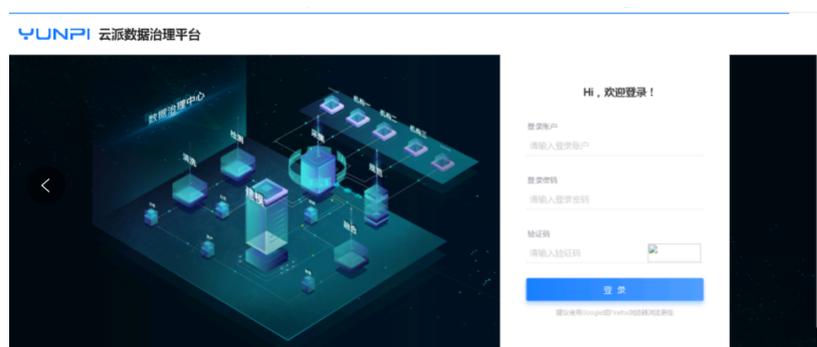
# Spring
spring:
  application:
    # 应用名称
    name: yunpi-system
  profiles:
    # 环境配置
    active: dev
  cloud:
    nacos:
      discovery:
        namespace: ${productName}
        group: ${spring.profiles.active}
        # 服务注册地址
        #server-addr: ${NACOS_HOST1:10.168.2.11}:${NACOS_PORT:8848},${NACOS_HOST2:10.168.2.12}:${NACOS_PORT:8848},${NACOS_HOST3:10.168.2.13}:${NACOS_PORT:8848}
        server-addr: 10.168.2.39:8848
        config:
          namespace: ${productName}
          group: ${spring.profiles.active}
          # 配置中心地址
          #server-addr: ${NACOS_HOST1:10.168.2.11}:${NACOS_PORT:8848},${NACOS_HOST2:10.168.2.12}:${NACOS_PORT:8848},${NACOS_HOST3:10.168.2.13}:${NACOS_PORT:8848}
          server-addr: 10.168.2.39:8848
        # 配置文件格式
        file-extension: yaml
        # 共享配置
        shared-configs:
          - group: ${spring.profiles.active}
            dataId: application-base-${spring.profiles.active}.${spring.cloud.nacos.config.file-extension}
            refresh: true
zipfile:usr/local/jar/sjzl/jar/yunpi-base-system.jar:8001-INF/classes/bootstrap.yml
```

5. 所有服务的启动

```
nohup java -jar yunpi-base-gateway.jar >> yunpi-base-gateway.log
nohup java -jar yunpi-base-auth.jar >> yunpi-auth.log
nohup java -jar yunpi-base-system.jar>> yunpi-base.log
nohup java -jar yunpi-bde-msvc.jar >> yunpi-bde.log
nohup java -jar yunpi-xxl.jar >> yunpi-xxl.log
nohup java -jar yunpi-dsm-msvs.jar >> yunpi-dsm.log
nohup java -jar yunpi-dwh-msvs.jar >> yunpi-dwh.log
nohup java -jar yunpi-mdm-msvs.jar>> yunpi-mdm.log
nohup java -jar yunpi-md-msvs.jar >> yunpi-md.log
nohup java -jar yunpi-file-msvs.jar>> yunpi-file.log
```

6. 访问界面: http://ip:8083

图 3-87 访问界面



7. 服务的停止:

```
Ps -ef|grep jar服务包名称
Kill -9 进程号
```

### 3.13 共享交换平台的部署

1. 将前端上传至服务器/usr/local/web下前端包  
包名为: yunpi-web
2. 将前端包上传至服务器/usr/local/web下  
修改yunpi-web/config文件夹下的config.js  
Vim config.js 修改网关地址

图 3-88 上传

```
window.baseUrl = {
  prod: 'http://10.168.2.238:9002/', // 项目部署地址
  fileDownload: {
    fileName: '系统帮助.docx', // 下载的文件名称
    fileUrl: '/systemHelp.docx' // 本地文件的路径
  }
}
window.sysConfig = {
  CN: '政务信息共享平台', // 平台中文名
  EN: 'Government Information Sharing Platform', // 平台英文名
  loginTitle: '数据信息共享平台', // 登录的主标题
  loginSubtitle: '便捷为民的高效信息共享网络工具', // 登录副标题
  copyright: 'Copyright © 2013-2022 陕西云基华海信息技术有限公司 All Rights Reserved.', // 版权
  copyCode: '', // 备案号
  browser: '(建议使用IE7、8、9、10、1024*768以上分辨率浏览本站)', // 浏览器版本
  skillSupport: '技术支持：云基华海信息技术有限公司', // 技术支持单位
  trustedUrl: 'http://10.168.2.246:20011' // 跳转至联邦学习基础路径
}
}
```

3. 后台服务的启动与配置。

配置jar服务：

Jar包为：dce-admin.jar dce-apimrg.jar dce-front.jar dce-center.jar dce-monitor.jar dce-res.jar

修改：jar服务配置信息：

vim dce-res.jar 修改数据库连接信息及redis连接信息，其他jar服务修改配置信息一样。

图 3-89 启动与配置

```
redis:
  database: 2
  host: 10.168.2.238
  port: 6379
  password: yjhh2022.com
  timeout: 3600
  jedis:
    pool:
      max-idle: 8
      max-wait: 1
      min-idle: 0
datasource:
  type: com.zaxxer.hikari.HikariDataSource
  driver-class-name: com.mysql.cj.jdbc.Driver
  url: jdbc:mysql://10.168.2.238:3306/produ_?dce_dev_online?zeroDateTimeBehavior=CONVERT_TO_NULL&useUnicode=true&characterEncoding=utf8
  username: root
  password: Qwer6666
  hikari:
    auto-commit: false
    minimum-idle: 2
    maximum-pool-size: 10
    connection-timeout: 10000
    max-lifetime: 600000
    idle-timeout: 60000
    validation-timeout: 1000
    leak-detection-threshold: 30000
jpa:
  database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
  database: MYSQL
  open-in-view: false
  generate-ddl: false
  show-sql: true
```

4. 服务的启动：

```
nohup java -jar dce-res.jar >>dce-res.log
nohup java -jar dce-spring.jar >>dce-res.log
nohup java -jar dce-monitor.jar >>dce-res.log
nohup java -jar dce-admin.jar >>dce-res.log
nohup java -jar dce-center.jar >>dce-res.log
nohup java -jar dce-res.jar >>dce-res.log
```

5. 服务的停止：

```
Ps -ef|grep jar服务包名称
Kill -9 进程号
```

6. 页面访问

访问界面：http://ip:8080

图 3-90 访问页面



# 4 修订记录

表 4-1 修订记录

发布日期	修订记录
2023-11-02	第一次正式发布。