解决方案实践

丰图城市数字孪生平台解决方案实践

文档版本 1.0

发布日期 2024-04-19

HUAWEI

华为技术有限公司

商标声明


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

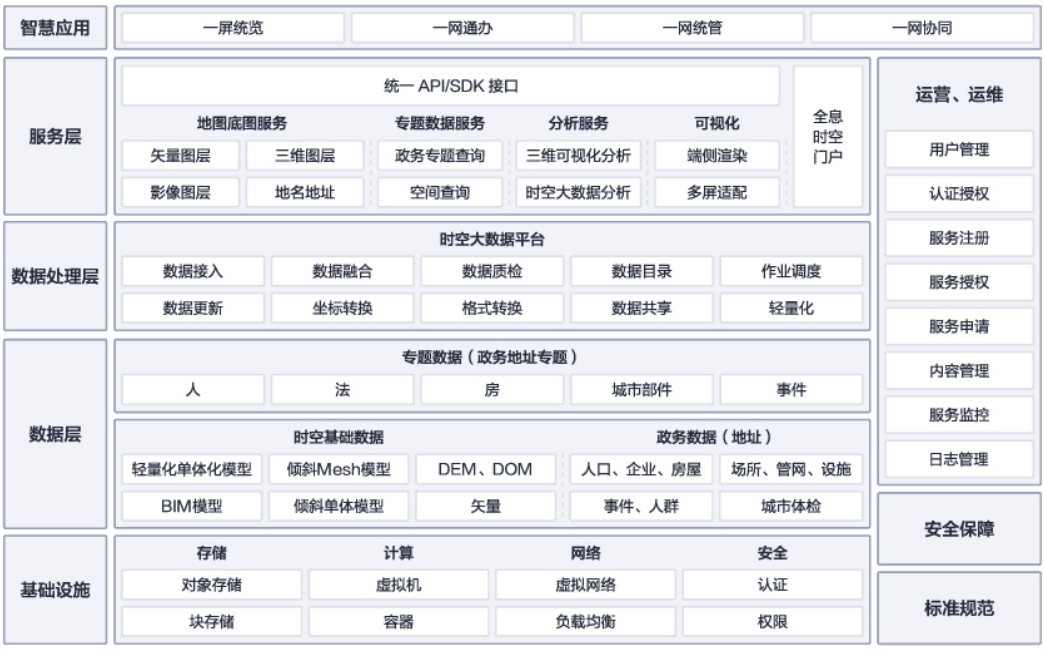
1 方案概述

应用场景

城市数字孪生平台作为城市级数字孪生建设的基础平台，通过城市过去、现在、将来的海量建筑、基础设施数据进行建模、管理和应用，汇聚并融合各委办局的业务数据，形成城市数字孪生底板。为智慧城市提供更丰富、细致、精确、直观的“时空可视化基础”，可支撑各类智慧应用系统的建设，主要包括数字孪生城市、新型智慧社区、应急管理、110实时警情定位平台、智慧交通、群租房位置预警系统、企业监管与服务、智能控税地图、统一地址服务服务、二维码门牌、事件智能分拨等。

方案架构

图 1-1 丰图城市数字孪生平台方案架构

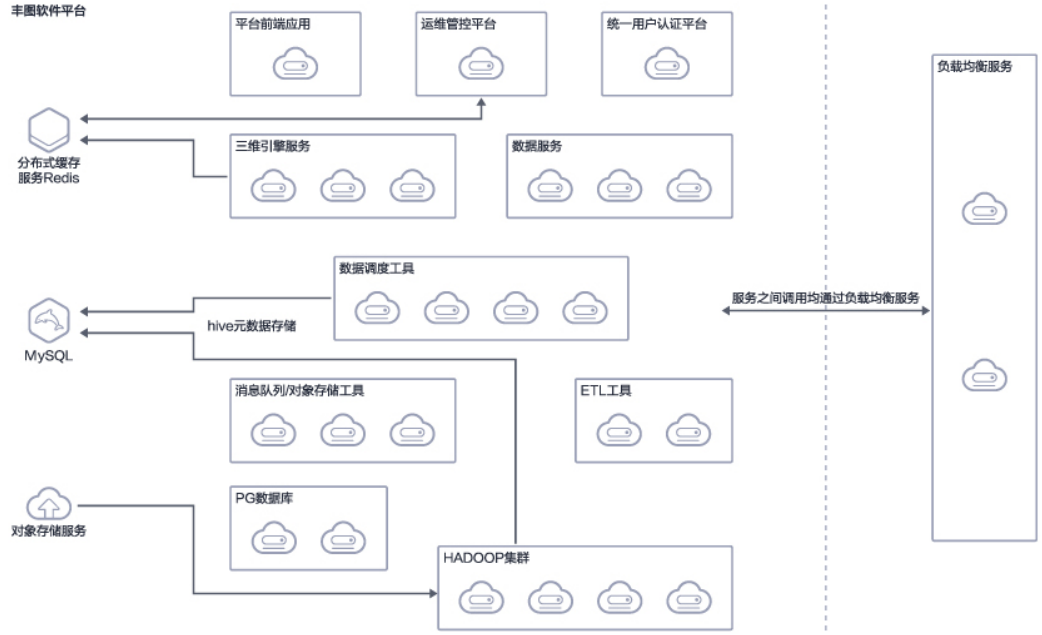


- 数据层提供CIM、BIM、矢量等时空基础数据、政务数据（地址相关）存储，其中政务数据地址标准化融合落图后以专题形式存储；
- 数据处理层提供二三维数据接入、格式与坐标转换、轻量化。政务数据接入、更新，地址标准化及与二三维地图语义化融合；

- 服务层以API/SDK形式对外提供二、三维、BIM地图底图服务、政务专题查询、空间查询、三维可视化及时空大数据分析，支持大中小屏可视化。全息时空门户提供运营、运维管理及CIM平台功能演示；

丰图城市数字孪生平台通常部署在客户侧政务云平台，部署环境设计如下图所示：

图 1-2 丰图城市数字孪生平台部署架构



架构描述：

1. 整个CIM平台依赖华为云HCS底座，包括ECS、Redis、MySQL、OBS等，由运维人员分配资源；
2. 数据层包括数据存储服务和大数据处理平台，数据通过ETL工具、数据调度工具等大数据处理平台组件处理后进行存储；
3. 平台对外以SDK/API的方式提供引擎服务能力和数据服务能力；
4. Portal和运管平台提供资源管控、系统运维能力；
5. 平台内服务之间调用通过负载均衡服务调用保障系统高可用；

方案优势

- 高逼真、大规模、高性能实景三维重建

  实景三维建模能力是在华为黎曼实验室的3D实景地图技术积累及Function Graph自适应调度能力基础上构建的：厘米级精细化还原建筑细节，高精度还原室内实景；同时大幅度提升经典建模流程效率，最高可支持60w+影像一次性整体处理。

- 提供PB级数据存储、极速地址数据查询服务

  丰图利用自身实践的时空大数据工程经验，基于华为云提供城市数字孪生平台底座能力，构建时空大数据平台产品，形成具有成熟的工业级时空数据生产、治理、分析服务体系。

- 独特数据资源及时空融合技术使数据更鲜活

为保障时空数据的鲜活性与准确性，丰图借助顺丰强大的线下资源和自动化的数据提取和更新技术，建立政府和企业协同的双循环的数据更新体系，降低数据更新成本提高更新效率。

2 资源和成本规划

表 2-1 资源和成本规划

云资源	规格	数量	每月费用（元）
弹性云服务器ECS	ARM | 8核 | 16GB 镜像：Euler2.8 存储：100GB	6	3876
弹性云服务器ECS	ARM | 16核 | 32GB 镜像：Euler2.8 存储：2000GB	3	7756.8
弹性云服务器ECS	ARM | 16核 | 32GB 镜像：Euler2.8 存储：1000GB	7	13081.6
弹性云服务器ECS	ARM | 16核 | 32GB 镜像：Euler2.8 存储：200GB	1	1292
弹性云服务器ECS	ARM | 16核 | 32GB 镜像：Euler2.8 存储：100GB	3	3666
弹性云服务器ECS	ARM | 32核 | 64GB 镜像：Euler2.8 存储：1000GB	9	27036
对象存储服务OBS	10T	1	912
分布式缓存服务Redis版	dcs.arm.master_standby_32G	1	2230.4
云数据库RDS	ARM 主备版 | 32核 | 64GB 存储：1TB	1	12892

云资源	规格	数量	每月费用（元）
总计：72742.8（*仅供参考，实际成本需要以具体项目为准*）			

3 实施步骤

3.1 数据库及中间件服务部署

3.1.1 服务器基础配置

本方案基于HCS底座部署，华为云的欧拉系统：基于稳定的Linux内核研发出面向企业级的通用服务器架构平台——Euler OS（Open Euler OS 2.8开源欧拉操作系统），支持ARM64鲲鹏处理器和容器虚拟化技术。

由HCS底座运维人员提供相关的操作系统、镜像和云服务器。

步骤1 挂载数据盘：/app

```
lsblk
fdisk -l
df -hT
```

步骤2 关闭防火墙和SELINUX

```
systemctl stop firewalld
systemctl disable firewalld
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config ; setenforce 0
```

步骤3 配置history命令显示时间

编辑文件/etc/bashrc，在最后添加如下内容：

```
vim /etc/bashrc
HISTFILESIZE=2000
HISTSIZE=2000
export HISTTIMEFORMAT="%Y-%m-%d_%H:%M:%S "
```

步骤4 设置主机名【可选】

1. 在机器上使用hostnamectl命令设置Hostname：
```
hostnamectl set-hostname sfmap1
hostname -f
```
2. 在机器上配置主机名解析，让相互之间可以主机名访问，在/etc/hosts中：
```
Vim +  /etc/hostname
127.0.0.1   localhost
```

步骤5 设置服务器的时间同步

配置内网或公网的时间服务器，所有的机器校准时间。例如：

```
ntpdate 10.10.10.10
```

步骤6 设置yum源

```
vim EulerOS-2.8.repo
[base]
name=EulerOS-2.0SP8 base
baseurl=http://repo.huaweicloud.com/euler/2.8/os/aarch64/
enabled=1
gpgcheck=0
gpgkey=http://repo.huaweicloud.com/euler/2.8/os/RPM-GPG-KEY-Euler
```

----结束

3.1.2 RDS 部署

RDS由HCS底座运维人员分配。

表 3-1 RDS 部署

服务	服务器	端口	账号/密码
RDS（华为云数据库）	192.168.x.x	3306	root/******

3.1.3 安装 PostgreSQL

1. 服务信息如下：

表 3-2 服务信息 2

服务名	服务器	安装目录	端口	账号/密码
Postgresql-9.6.8	10.190.x.x（主库） 10.190.x.x（从库）	/app/postgresql	5432	Postgres/******

2. 安装PG
   a. 准备安装包

      将编译好的在ARM架构欧拉系统编译通过的PostgreSQL+**PostGIS**软件包放到服务器/app目录下并解压：

```
tar xf postgresql_9.6.8_arm_eulerOS.tar.gz
```

b. 添加用户并授权

```
groupadd  postgres
useradd postgres -g postgres -m -d /home/postgres
chown -R postgres:postgres /app/postgresql
```

c. 切换用户，启动PG

```
su - postgres
cd /app/postgresql
./bin/pg_ctl -D data -l logfile start
```

d. 配置主从同步

以172.16.1.22（主），172.16.1.21（从）为例，编辑主节点配置

```
vi /app/pg/data/pg_hba.conf   #如下：
host    all     all       slave1ip/32       trust   #允许连接到主服务器
host   replication    replica     slave1ip/32      md5   #允许用replica复制
```

这里在最下面添加两行：

```
host    all             all             172.16.1.21/32          trust
host    replication     replica         172.16.1.21/32          trust

vi /app/postgresql/data/postgresql.conf     #如下：
data_directory = '/app/postgresql/data'             #自定义data目录
listen_addresses = '*'                          #监听所有ip
archive_mode = on                               #允许归档
archive_command = 'cp %p /app/postgresql/data/pg_archive/%f'   #使用命令归档
wal_level = replica                             #选择热备replica或logical
max_wal_senders = 16                              #最多多少个流复制链接
wal_keep_segments = 256                           #流复制保留最多的xlog数
wal_sender_timeout = 60s                          #流复制主机发送数据超时时间
max_connections = 5000                  #从库的max_connections必须大于主库的
```

e. 创建用户replica进行主从同步，并赋予登录和复制的权限

- 登录到数据库里(主节点)

```
su postgres
cd  /app/postgresql/
./bin/psql
CREATE ROLE replica login replication encrypted password 'replica';
./bin/pg_ctl -D data -l logfile restart
```

- slave1部分：#先备份数据，再同步数据

```
./bin/pg_ctl -D data -l logfile stop   ## 保持服务处于关闭状态
#自定义存档目录，先把旧的data全部移走
mkdir /home/postgres/pg_archive/
mv  /app/postgresql/data/  /home/postgres/pg_archive/
chmod 700 pg_archive && chown postgres:postgres pg_archive/
su  postgres
rm -rf  /app/postgresql/data/*   #先将data目录下的数据都清空
#为空的情况下，把主节点的数据用pg_basebackup同步到从节点
cd /app/postgresql/bin
pg_basebackup -P -h masterip -U replica -D /app/postgresql/data -X stream
./pg_basebackup -P -h 172.16.1.22 -U replica -D /app/postgresql/data -X stream
#配置recovery.conf，最底下添加三行。
cp /app/postgresql/share/recovery.conf.sample /app/postgresql/data/recovery.conf
vi /app/postgresql/data/recovery.conf
standby_mode = on         #该节点为从
primary_conninfo = 'host=$masterip port=5432 user=replica password=replica'
                        #主服务器的ip、user
recovery_target_timeline = 'latest'
##trigger_file = '/tmp/trigger_file0'
#配置postgresql.conf   ## 添加到75行位置，其余内容不需修改。
vi /app/postgresql/data/postgresql.conf
max_connections = 5500     #尽量大于主连接数的10%
max_standby_streaming_delay = 30s
wal_receiver_status_interval = 10s
hot_standby = on    ##从节点默认用off , 配置on启动psql可查询数据。
hot_standby_feedback = on  #出现错误复制，向主机反馈
#开启从数据库.
```

```
./bin/pg_ctl -D data -l logfile start
netstat  -tlnp
```

f. 查看复制状态（主库172.16.1.22）

```
./bin/psql
select client_addr,sync_state from pg_stat_replication;
```

图 3-1 查看复制状态

```
postgres=# select client_addr,sync_state from pg_stat_replication;
 client_addr | sync_state
-------------+------------
 172.16.1.21 | async
(1 row)
```

```
##主从节点的进程多了wal进程：
ps -ef | grep postgres
postgres: wal sender process replica 172.16.1.21
postgres: wal receiver process   streaming
#用pg_controldata命令查询主从集群运行状态
[postgres@host-172-16-1-22 bin]$ ./pg_controldata /app/postgresql/data/
pg_control version number:            960
Catalog version number:               201608131
Database system identifier:           7127200572656879006
Database cluster state:               in production
[postgres@host-172-16-1-21 bin]$ ./pg_controldata /app/postgresql/data/
pg_control version number:            960
Catalog version number:               201608131
Database system identifier:           7127200572656879006
Database cluster state:               in archive recovery
```

主从搭建成功后，主库的集群状态是in production，从库是in archive recovery，当主库崩溃，可以切换从库为主库。这时候主库状态是shut down，而从库是in production。

3.1.4 Kafka 集群部署

1. 服务信息如下：

表 3-3 服务信息 3

服务名	服务器	安装目录	端口
zookeeper	10.190.x.x 10.190.x.x	/app/zookeeper	2181、2888、3888
kafka	10.190.x.x	/app/kafka	9092

2. 安装JDK8

a. 准备安装包

可默认使用操作系统自带的openjdk

```
[root@vm001 ~]# rpm -qa|grep java
[root@vm001 ~]# rpm -qa|grep jdk
yum list java-1.8.0-openjdk*
（可选安装java-1.8.0-openjdk-devel）
yum -y install java-1.8.0-openjdk java-1.8.0-openjdk-devel
tar xf  jdk-8u341-linux-aarch64.tar.gz -C /usr/local/
```

b. 配置环境变量快捷命令

```
sed -i '$a export JAVA_HOME=/usr/local/jdk1.8.0_341\nexport JRE_HOME=$JAVA_HOME/jre
\nexport PATH=$JAVA_HOME/bin:JRE_HOME/bin:$PATH\nexport CLASSPATH=.:$JAVA_HOME/lib/
dt.jar:$JAVA_HOME/lib/tools.jar ' /etc/profile && source /etc/profile && java -version
```

3. 安装zookeeper
   a. 准备安装包

      三节点部署zookeeper，使用zookeeper-3.4.14.tar.gz，解压到/app目录，下载地址：https://www.apache.org/dyn/closer.lua/zookeeper/zookeeper-3.4.14/zookeeper-3.4.14.tar.gz

      ```
      tar xf zookeeper-3.4.14.tar.gz  -C /app
      mv zookeeper-3.4.14  zookeeper
      ```

   b. 添加zookeeper到环境变量

      ```
      vim + /etc/profile
      export ZOOKEEPER_HOME=/app/zookeeper
      export PATH=$ZOOKEEPER_HOME/bin:$PATH
      #集成安装上面的jdk之后，环境变量PATH合并配置示例：
      export ZOOKEEPER_HOME=/app/zookeeper
      export JAVA_HOME=/usr/local/jdk1.8.0_241
      export JRE_HOME=$JAVA_HOME/jre
      export PATH=$ZOOKEEPER_HOME/bin:$JAVA_HOME/bin:JRE_HOME/bin:$PATH
      export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
      #使环境变量生效
      source /etc/profile
      ```

   c. 修改配置文件

      ```
      #进入ZooKeeper所在目录
      cd /app/zookeeper/conf
      #复制配置文件
      cp zoo_sample.cfg zoo.cfg
      #修改配置文件
      vim zoo.cfg
      #修改数据目录
      dataDir=/app/zookeeper/data
      #在最后添加如下代码，其中server.1-3是部署ZooKeeper的节点对应sfmap345
      server.1=sfmap3:2888:3888
      server.2=sfmap4:2888:3888
      server.3=sfmap5:2888:3888
      #创建data目录作数据目录
      mkdir /app/zookeeper/data
      #在data目录中创建一个空文件，并向该文件写入ID
      touch /app/zookeeper/data/myid
      echo 1 > /app/zookeeper/data/myid
      ```

   d. 同步配置到其它节点

      ```
      #将配置好的ZooKeeper复制到其它节点
      scp -rp /app/zookeeper-3.4.6 root@sfmap4:/app/
      scp -rp /app/zookeeper-3.4.6 root@sfmap5:/app/
      #登录sfmap4、sfmap5，修改myid内容
      echo 2 > /app/zookeeper/data/myid
      echo 3 > /app/zookeeper/data/myid
      ```

   e. 启动服务

      分别在sfmap3，sfmap4、sfmap5上启动ZooKeeper

      ```
      cd /app/zookeeper/bin
      ./zkServer.sh start
      #查看服务状态
      ./zkServer.sh status
      Mode: leader   #或者   Mode: follower
      ```

4. 安装kafka
   a. 准备安装包

      下载地址：https://www.apache.org/dyn/closer.cgi?path=/kafka/2.6.0/kafka_2.13-2.6.0.tgz

      ```
      tar xf kafka_2.13-2.6.0.tgz -C /app
      mv kafka_2.13-2.6.0 kafka
      ```

b. 修改配置文件server.properties

```
broker.id=1
listeners=PLAINTEXT://0.0.0.0:9092  ##0.0.0.0或本机IP
#advertised.listeners=PLAINTEXT://192.168.56.101:9092
advertised.listeners=PLAINTEXT://192.168.0.100:9092  ##本机或公网IP
……
##log.dirs=/tmp/kafka-logs ##或者使用默认配置
log.dirs=/app/kafka/kafka-logs
……
##zookeeper.connect=localhost:2181  ##加注释默认的自带的zoo
zookeeper.connect=192.168.56.101:2181,192.168.56.102:2181,192.168.56.103:2181
zookeeper.connection.timeout.ms=18000
group.initial.rebalance.delay.ms=0
```

c. 修改启动脚本

修改bin/kafka-run-class.sh 找到约257行：

```
KAFKA_JVM_PERFORMANCE_OPTS="-server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 ...
```

直接删掉这个的参数 -XX:+UseG1GC，保存退出。

d. 参考a、b、c完成节点2和节点3的安装和配置

e. 三节点分别创建数据存储目录

```
mkdir  -p /app/kafka/kafka-logs
```

f. 三节点分别启动服务

```
cd  bin/
./kafka-server-start.sh -daemon  ../config/server.properties
```

3.1.5 Minio 集群部署

1. 服务信息如下：

表 3-4 服务信息 4

服务名	服务器	安装目录	端口	账号/密码
Minio	10.190.x.x 10.190.x.x 10.190.x.x	/app/minio	9029	admin/******

2. 安装minio

a. 准备安装包

从官网获取Minio，选arm64版本：**https://dl.min.io/server/minio/release/linux-amd64/minio**

b. 创建目录并上传Minio到/app/minio/run

```
mkdir -p /app/minio/{run,data,data1,data2}
```

c. 配置集群启动文件

```
# vim /app/minio/run/run.sh
#!/bin/bash
export MINIO_ACCESS_KEY=admin
export MINIO_SECRET_KEY=Admin123@minio
/app/minio/run/minio server --config-dir /etc/minio \
--address :9029 \
http://192.168.1.1/app/minio/data1 http://192.168.1.1/app/minio/data2 \
http://192.168.1.2/app/minio/data1 http://192.168.1.2/app/minio/data2 \
http://192.168.1.3/app/minio/data1 http://192.168.1.3/app/minio/data2
```

Minio默认9000端口，在配置文件中加入–address “127.0.0.1:9029” 可更改端口

说明

MINIO_ACCESS_KEY：用户名，长度最小是5个字符

MINIO_SECRET_KEY：密码，密码不能设置过于简单，不然minio会启动失败，长度最小是8个字符

–config-dir：指定集群配置文件目录

d. 创建Minio.server

```
# vim /usr/lib/systemd/system/minio.service
[Unit]
Description=Minio service
Documentation=https://docs.minio.io/
[Service]
WorkingDirectory=/app/minio/run
ExecStart=/app/minio/run/run.sh
Restart=on-failure
RestartSec=5
[Install]
WantedBy=multi-user.target
```

e. 修改权限

```
# chmod +x /usr/lib/systemd/system/minio.service && chmod +x /app/minio/run/minio &&
chmod +x /app/minio/run/run.sh
```

f. 启动集群并查看集群状态

```
# systemctl daemon-reload
# systemctl start minio
# systemctl enable minio
# systemctl status minio.service
```

g. 使用 nginx 负载均衡

单独对每个节点进行访问无法保障可靠性，通过使用 nginx 代理，进行负载均衡很有必要。配置如下：

```
upstream http_minio {
    server 192.168.222.10:9001 max_fails=3 fail_timeout=5s;
    server 192.168.222.10:9002 max_fails=3 fail_timeout=5s;
    server 192.168.222.10:9003 max_fails=3 fail_timeout=5s;
    }
server{
    listen       8888;
    server_name  192.168.222.10;
    ignore_invalid_headers off;
    client_max_body_size 0;
    proxy_buffering off;
    location / {
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-Host  $host:$server_port;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto  $http_x_forwarded_proto;
        proxy_set_header   Host $http_host;
        proxy_connect_timeout 300;
        proxy_http_version 1.1;
        chunked_transfer_encoding off;
        proxy_ignore_client_abort on;
        proxy_pass http://http_minio;
    }
}
```

3.1.6 Redis 集群部署

1. 服务信息如下：

表 3-5 服务信息 5

服务名	服务器	安装目录	端口	密码
redis	10.190.x.x	/app/redis-6.2.6	6379、6380、6381	******

2. 安装redis

   a. 准备安装包

```
cd /app
wget https://download.redis.io/releases/redis-6.2.6.tar.gz
tar -zxvf redis-6.2.6.tar.gz
cd /app/redis-6.2.6
```

   b. 编译

```
yum -y install gcc-c++
yum -y install make
##首先进入deps目录，编译Redis依赖库
cd deps/
make -j4 hiredis lua jemalloc linenoise
##再进行编译Redis
cd ../redis-6.2.6/
yum provides */ld
yum install -y  binutils-2.31.1-13.h3.eulerosv2r8.aarch64
[root@host-172-16-1-173 redis-6.2.6]# which ld
/usr/bin/ld
make -j4
make install
```

   c. 修改6个实例的配置文件，端口分别为7000、7001、7002、7003、7004、7005

```
cp redis.conf redis-7000.conf
vim  redis-7000.conf
bind 0.0.0.0
protected-mode no
port 7000
daemonize yes   #守护进程
pidfile ./redis_7000.pid
Loglevel warning
logfile ./redis-7000.log
dbfilename "dump-7000.rdb"
maxmemory 8GB
maxmemory-policy allkeys-lru
appendonly yes  #持久化
cluster-enabled yes
cluster-config-file "node-7000.conf"
activedefrag yes #碎片整理
masterauth "redis123"
##user default on
requirepass ****** #服务连接密码
Ignore-warnings ARM64-COW-BUG   #配置ARM64内核bug问题选项
```

   d. 三节点分别启动服务

```
nohup ./src/redis-server redis-7000.conf &
nohup ./src/redis-server redis-7001.conf &
nohup ./src/redis-server redis-7002.conf &
nohup ./src/redis-server redis-7003.conf &
nohup ./src/redis-server redis-7004.conf &
nohup ./src/redis-server redis-7005.conf &
 cd  /opt/redis/redis-6.2.1/src/
./redis-cli --cluster create node1:7001 node2:7003 node3:7005 node2:7002 node3:7004
node1:7000 --cluster-replicas 1
[root@zh-prod-redis-1 src]# ./redis-cli --cluster create 172.16.19.25:7001 172.16.19.26:7003
172.16.19.18:7005 172.16.19.26:7002 172.16.19.18:7004 172.16.19.25:7000  --cluster-replicas 1 -
a 'redis123'
[OK] All 16384 slots covered.
```

e. 部署成redis一主两从三哨兵的集群模式

```
mkdir /app/redis-5.0.5
 mkdir /app/redis-5.0.5/logs/
[root@host-172-16-1-173 conf]# ls /app/redis-5.0.5
conf  dump.rdb  logs  start.sh  stop.sh
[root@host-172-16-1-173 conf]# ls
redis.conf  sentinel1.conf  sentinel2.conf  sentinel3.conf  slave1.conf  slave2.conf
```

f. 启动、停止

```
./start.sh
./stop.sh
```

3.1.7 DCS 部署

DCS由HCS底座运维人员分配。

表 3-6 DCS 部署

服务	服务器	端口	账号/密码
DCS（华为云）	192.168.x.x	3306	******

3.2 数据底座模块部署

3.2.1 服务信息如下：

表 3-7 服务信息

服务器							
弹性IP（10.190.x.）	170	171	172	173	174	175	176
内网IP（192.168.32.）	110	2	11	102	148	39	174
Hostname（cim-skdsjpt-hadoop-00）	08	09	10	11	12	13	14

3.2.2 账号密码

表 3-8 账号密码

服务	链接或IP端口	账号	密码
管理界面 manager	10.190.x.170:8180	admin	******

服务	链接或IP端口	账号	密码
服务部署账号	所有节点	tdh	******
guardian-server	-	admin	******
inceptor	jdbc:hive2://10.190.x.x:10000/	-	-

3.2.3 安装 Manager

数据管理模块的安装需要先解压安装包，然后运行Web Installer使用图形化界面安装。

1. 准备安装包

```
# 解压出安装目录
tar xvzf TDH-Platform-Transwarp-version.tar.gz
# 进入解压后的transwarp目录
cd transwarp
#执行install二进制文件
./install
```

2. 通过浏览器访问管理节点，进入Web Installer界面

图 3-2 进入 Web Installer 界面

3. 系统首先需要您阅读Java许可。阅读完毕，单击“同意”进入下一步

图 3-3 下一步

4. 系统将自动检查管理节点的环境配置，主要包括时间、日期、时区及主机名信息并显示在屏幕上，请确认

图 3-4 确认

5. 安装结束后，您会被要求设置Transwarp Manager端口，推荐默认端口“8180”

图 3-5 推荐默认端口

6. 等待安装

图 3-6 等待安装

7. 安装Manager需要一个包含对应版本操作系统的资源库（repo）

图 3-7 repo

a. 如果您选择“使用远程RPM库”，您需要输入已经创建好的资源库的URL：

b. 如果您选择“创建新的RPM库(DVD)”，您需要准备好对应版本操作系统的光盘。

c. 如果您选择“创建新的RPM库(ISO)”，您需要准备好对应版本操作系统的ISO文件。建议您采用ISO镜像方法生成资源库库包。

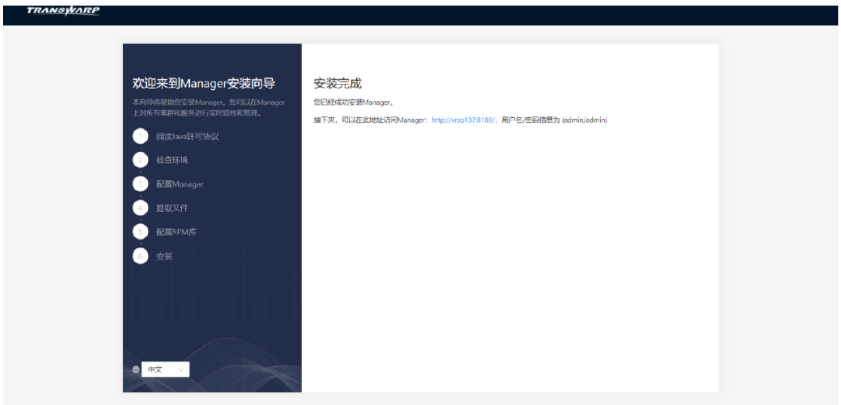
d. 如果您选择“后台手工配置”，您在该步骤无需进行其他操作。

选择后，系统会清理资源库缓存。

8. 资源库缓存清理完毕后，系统会自动开始安装和配置Transwarp Manager。安装程序会自动安装必需的软件包，全程静默安装，安装配置完成后自动跳转到下一步。

9. Manager安装完成，可以访问提示的安装地址并使用默认的用户名/密码（admin/******）去登录管理界面继续接下来的配置

图 3-8 安装完成



3.2.4 配置集群

1. 打开并登录Manager

   打开客户端浏览器（推荐使用Google Chrome浏览器），输入安装好的管理节点IP或DNS地址，比如http://172.16.3.108:8180/（172.16.3.108是管理节点的IP地址）。以admin的身份登录Manager

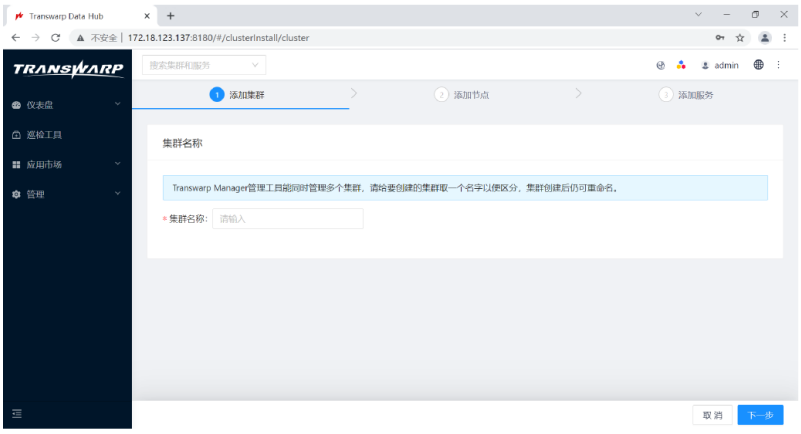
图 3-9 登录 Manager

2. 接受最终用户协议

图 3-10 接受

3. 设置集群名字，单击“下一步”

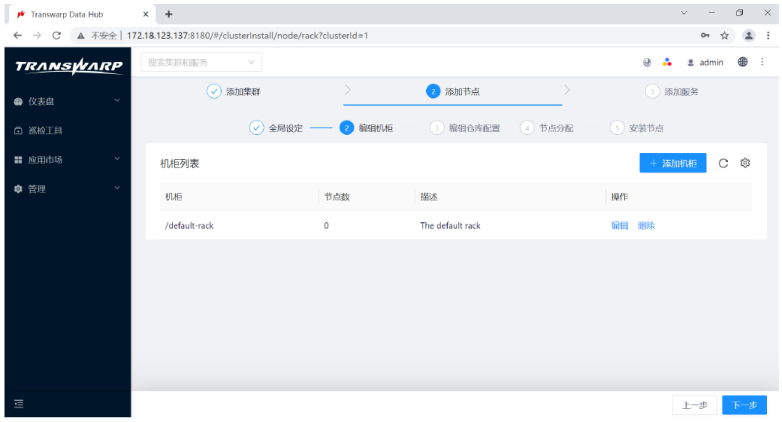
图 3-11 单击下一步



4. 添加集群服务器

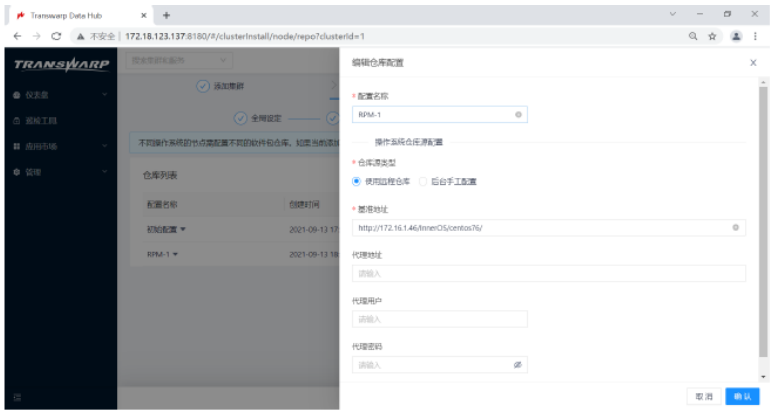
   单击机柜的名字和描述可以对它们进行编辑

图 3-12 添加集群服务器 1



配置RPM库

图 3-13 添加集群服务器 2



分配集群节点，并设置管理账号信息

图 3-14 添加集群服务器 3

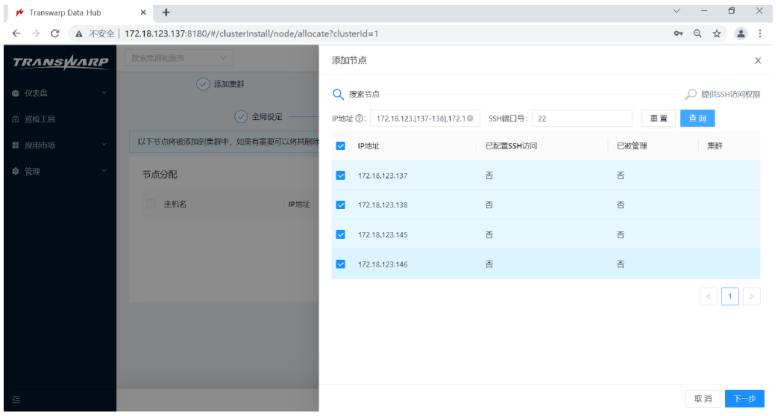
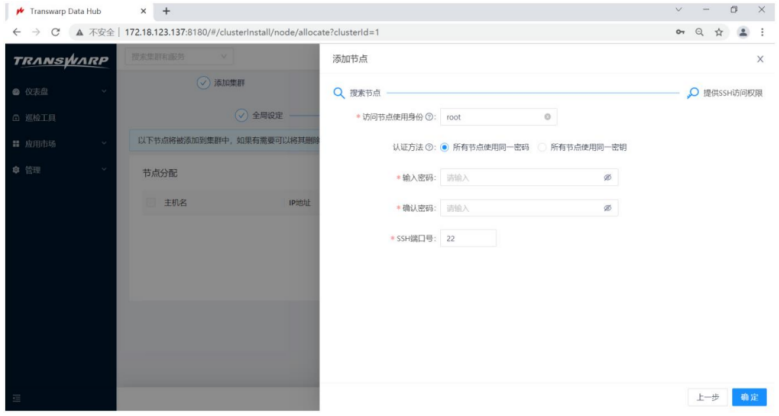
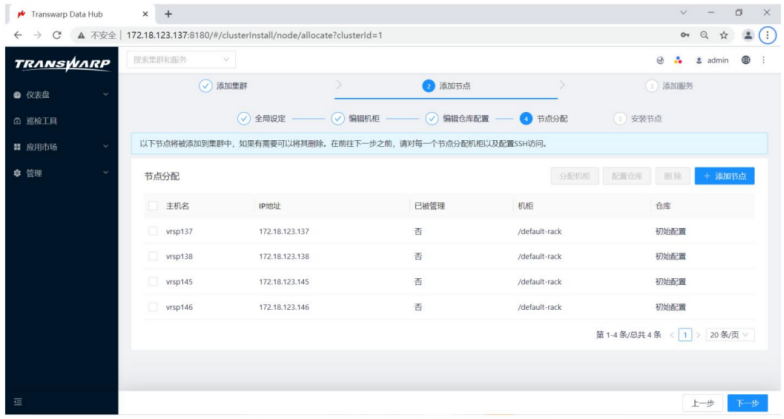


图 3-15 添加集群服务器 4



5. 配置完成

图 3-16 配置完成

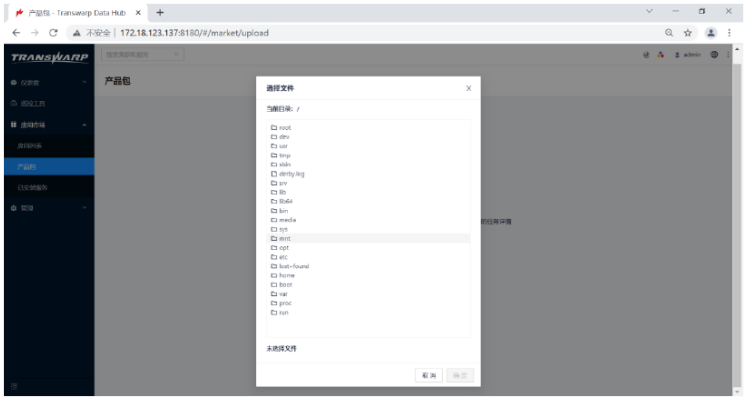


3.2.5 安装服务

1. 上传产品包

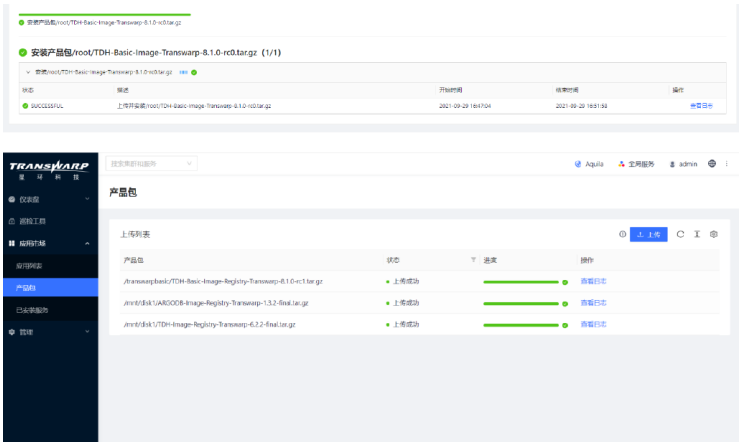
   从Manager菜单单击 应用市场，打开应用市场页面, 进入应用市场默认页面，从左侧导航栏单击上传产品包，并选择上传位置。

图 3-17 上传产品包 1



2. 上传成功后，相应的服务将会出现在安装服务时的选择服务页面

图 3-18 上传产品包 2



3.3 CIM 数据汇聚管理软件部署

3.3.1 spark 提交服务

1. 服务分布：

表 3-9 服务分布

服务名	服务器	安装目录	端口
data-spark-submit	10.190.x.x 10.190.x.x	/app/dt/data-spark-submit	8087

2. 安装spark提交服务
   a. 修改配置文件application-dev.yml
      - 修改stdms.callback.url中的ip和端口，配置为CIM数据汇聚管理平台的ip和端口
      - 当使用minio存储日志时，配置log.server.enable为true，并配置日志服务（data-log-manage）的ip和端口；

      当使用本地文件存储日志时，配置log.server.enable为false

```yml
# 服务端口
server:
  port: 8087
# 任务执行完成，回调CIM数据汇聚管理平台接口
stdms:
  callback:
    url: http://192.168.32.89:9090/stdms/feed-back/ch-task-status?objectId=%s&status=%d
# 是否将日志发送到日志服务并存储到minio
log:
  service:
    enable: false
    url: http://127.0.0.1:8089/save
# 通过minio中转传输脚本文件
minio:
  endpoint: http://192.168.32.227:9000
```

```
  access-key: admin
  secret-key: Admin123@minio
# 生产打开下面配置，使用jar同级config目录配置文件
logging:
  config: config/log4j2.xml
```

b. 启动、停止

- 启动：

  进入data-spark-submit.jar所在路径，执行

  ```shell
  nohup java -jar data-spark-submit.jar &
  ```

- 停止：

  ```
  Kill -9 进程号
  ```

c. 服务验证

- 服务启动验证与版本信息查询

  将以下地址中的ip和端口修改为实际部署ip和端口，请求接口，查看版本信息

  请求地址：http://localhost:8087/about

  请求方式：GET

- 服务功能验证

  任务提交验证：将以下地址中的ip和端口修改为实际部署ip和端口，在minio中创建桶data-mining-spark，将test文件夹中的test.py脚本上传至桶data-mining-spark的根路径下，请求以下接口

  请求地址：http://localhost:8087/spark-submit

  请求方式：POST

  请求参数体：

  ```json
  {
      "taskName": "app",
      "taskId": "1",
      "fileLocation": "minio",
      "master": "local",
      "deployMode": "client",
      "appResource": "test.py",
      "jars": "",
      "mainClass": "",
      "pyFiles": "",
      "verbose": false,
      "conf": {
          "spark.driver.memory": "1g",
          "spark.executor.memory": "1g",
          "spark.executor.cores": "2",
          "spark.kryoserializer.buffer.max": "1",
          "spark.default.parallelism": "100",
          "spark.shuffle.service.enabled": "false",
          "spark.dynamicAllocation.minExecutors": "1",
          "spark.dynamicAllocation.maxExecutors": "2",
          "spark.dynamicAllocation.enabled": "false"
      },
      "appArgs": []
  }
  ```

  返回结果示例：

  ```json
  {
  ```

```
    "success": true,
    "message": "提交spark任务成功",
    "content": {
        "taskId": "1",
        "taskName": "app",
        "createTime": "2022-08-16 18:27:24",
        "applicationId": "local-1660645647751"
    }
}
```

3.3.2 脚本执行服务

1. 服务分布：

表 3-10 服务分布

服务名	服务器	安装目录	端口
data-script-exec	10.190.x.x 10.190.x.x	/app/dt/data-script-exec	8090

2. 安装脚本执行服务

   a. 修改配置文件application-dev.yml

   - 修改stdms.callback.url中的ip和端口，配置为CIM数据汇聚管理平台的ip和端口
   - 当使用minio存储日志时，配置log.server.enable为true，并配置日志服务（data-log-manage）的ip和端口；

   当使用本地文件存储日志时，配置log.server.enable为false

```yml
# 服务端口
server:
  port: 8090
# 任务执行完成，回调CIM数据汇聚管理平台接口
stdms:
  callback:
    url: http://192.168.32.122:9090/stdms/feed-back/ch-task-status?objectId=%s
# 是否将日志发送到日志服务并存储到minio
log:
  service:
    enable: false
    url: http://127.0.0.1:8089/save
# 是否通过minio中转传输脚本文件
minio:
  enable: true
  endpoint: http://192.168.32.227:9000
  access-key: admin
  secret-key: Admin123@minio
# 生产打开下面配置，使用jar同级config目录配置文件
logging:
  config: config/log4j2.xml
```

   b. 启动、停止

   - 启动：

     进入data-script-exec.jar所在路径，执行

```shell
nohup java -jar data-script-exec.jar &
```

- 停止：
  Kill -9 进程号

c. 服务验证

i. 服务启动验证与版本信息查询

将以下地址中的ip和端口修改为实际部署ip和端口，请求接口，查看版本信息

请求地址：http://localhost:8090/about

请求方式：GET

ii. 服务功能验证

任务提交验证：将以下地址中的ip和端口修改为实际部署ip和端口，在minio中创建桶data-mining-shell，将test文件夹中的test.sh脚本上传至桶data-mining-shell的根路径下，请求以下接口

请求地址：http://localhost:8090/shell-exec

请求方式：POST

请求参数体：

```json
{
    "taskId": 1,
    "taskName": "test",
    "shellPath": "test.sh",
    "shellParam": "",
    "type": 0,
    "mainClass": "test.sh"
}
```

返回结果示例：

```json
{
    "success": true,
    "message": "提交shell任务成功",
    "content": {
        "taskId": "723",
        "taskName": "test",
        "createTime": "2022-08-16 16:22:42",
        "applicationId": "99e40788a4ae4205a4db097c84781066"
    }
}
```

3.3.3 查询服务

1. 服务分布：

表 3-11 服务分布

服务名	服务器	安装目录	端口
data-ad-hoc	10.190.x.x 10.190.x.x	/app/dt/data-ad-hoc	8086

2. 安装查询服务

a. 修改配置文件application-dev.yml

- 修改stdms.callback.url中的ip和端口，配置为CIM数据汇聚管理平台的ip和端口
- 当使用minio存储日志时，配置log.server.enable为true，并配置日志服务（data-log-manage）的ip和端口；

当使用本地文件存储日志时，配置log.server.enable为false

```yml
# 服务端口
server:
  port: 8086
# 任务执行完成，回调CIM数据汇聚管理平台接口
stdms:
  callback:
    url: http://192.168.32.89:9090/stdms/feed-back/ch-task-status?objectId=%s&status=%d
# 是否将日志发送到日志服务并存储到minio
log:
  service:
    enable: true
    url: http://127.0.0.1:8089/save
# 使用jar同级config目录配置文件
logging:
  config: config/log4j2.xml
```

b. 启动、停止

- 启动：

  进入data-ad-hoc.jar所在路径，执行

  ```shell
  nohup java -jar data-ad-hoc.jar &
  ```

- 停止：

  ```
  Kill -9 进程号
  ```

c. 服务验证

i. 服务启动验证与版本信息查询

将以下地址中的ip和端口修改为实际部署ip和端口，请求接口，查看版本信息

请求地址：http://localhost:8087/about

请求方式：GET

ii. 服务功能验证

任务提交验证：将以下地址中的ip和端口修改为实际部署ip和端口，配置正确的数据库连接信息，在数据库中创建user表，并填入任意测试数据，如下述案例表示查询本地mysql的demo库中的user表。

请求地址：http://localhost:8086/jdbc/execute

请求方式：POST

请求参数体：

```json
{
   "querySql": "aaaaaaaac2VsZWN0ICogZnJvbSB1c2Vy",
   "datasourceConfig": {
      "type": "MYSQL",
      "host": "localhost",
```

```
        "port": 3306,
        "database": "demo",
        "username": "root",
        "password": "123456"
    }
}
```

返回结果示例：

```json
{
    "success": true,
    "message": "提交任务成功",
    "content": {
        "createTime": "2022-08-16 11:39:25",
        "applicationId": "f4be5dbb6eaf4065a8ecc4ff6abc9529"
    }
}
```

3.3.4 日志服务

1. 服务分布：

表 3-12 服务分布

服务名	服务器	安装目录	端口
data-log-manager	10.190.x.x 10.190.x.x	/app/dt/data-log-manager	8089

2. 安装spark提交服务
   a. 修改配置文件application-dev.yml
      - 修改stdms.callback.url中的ip和端口，配置为CIM数据汇聚管理平台的ip和端口
      - 配置实时日志路径service-logs.path，默认路径为data-ad-hoc服务同级的logs路径
      - 当使用minio存储存档日志时，配置minio.enable为true，并配置日志服务（data-log-manage）的ip和端口；

      当使用本地文件存储日志时，配置log.server.enable为false

```yml
# 服务端口
server:
  port: 8089
# 实时日志路径
service-logs:
  path: "/app/appdeploy/logs"
# 日志存储到minio路径
minio:
  enable: false
  endpoint: http://127.0.0.1:9000
  access-key: admin
  secret-key: Admin123@minio
# 生产打开下面配置，使用jar同级config目录配置文件
logging:
  config: config/log4j2.xml
```

b. 启动、停止

- 启动：

  进入data-ad-hoc.jar所在路径，执行

  ```shell
  nohup java -jar data-log-manager.jar &
  ```

- 停止：

  ```
  Kill -9 进程号
  ```

c. 服务验证

服务启动验证与版本信息查询

将以下地址中的ip和端口修改为实际部署ip和端口，请求接口，查看版本信息

请求地址：http://localhost:8087/about

请求方式：GET

3.3.5 质检服务

1. 服务分布：

表 3-13 服务分布

服务名	服务器	安装目录	端口
data-quality-manager	10.190.x.x 10.190.x.x	/app/dt/data-quality-manager	8084

2. 安装spark提交服务

a. 修改配置文件application-dev.yml

i. 修改stdms.callback.url和stdms.result.url中的ip和端口，配置为CIM数据汇聚管理平台的ip和端口

ii. 采用本地模式运行，配置spark.master为local；采用集群模式运行，配置spark.master为yarn

```yml
# 服务端口
server:
  port: 8084
# spark任务提交模式
spark:
  master: local
# 任务执行完成，回调CIM数据汇聚管理平台接口，通知任务完成与质检数据回传
stdms:
  callback:
    url: http://192.168.32.89:9090/stdms/feed-back/ch-task-status
  result:
    url: http://192.168.32.89:9090/stdms/check-result/batch-save
# 生产打开下面配置，使用jar同级config目录配置文件
logging:
  config: config/log4j2.xml
```

b. 启动、停止

- 启动：

  进入data-quality-manager.jar所在路径，执行

```shell
nohup java -jar data-quality-manager.jar &
```

- 停止：
  Kill -9 进程号

c. 服务验证

i. 服务启动验证与版本信息查询

将以下地址中的ip和端口修改为实际部署ip和端口，请求接口，查看版本信息

请求地址：http://localhost:8087/about

请求方式：GET

ii. 服务功能验证

质检功能验证：将以下地址中的ip和端口修改为实际部署ip和端口，以下请求将对mysql库中的user表进行相关质检

请求地址：http://localhost:8089/dataquality/check

请求方式：POST

请求参数体：

```json
{
   "taskId": "7fdbcc680a4f4e929bc72d89856b00d3",
   "taskName": "",
   "taskTime": "2022-03-21 18:30:00",
   "nodes": [
      {
         "connectionInfo": {
            "datasourceId": 1,
            "type": "MYSQL",
            "host": "localhost",
            "port": 3306,
            "database": "demo",
            "username": "root",
            "password": "123456"
         },
         "dbRules": [
            ......
         ]
      }
   ]
}
```

3.3.6 算子服务

1. 服务分布：

表 3-14 服务分布

服务名	服务器	安装目录	端口
data-wrangling	10.190.x.x 10.190.x.x	/app/dt/data-wrangling	8085

2. 安装算子服务

a. 修改配置文件application-dev.yml

i. 修改stdms.callback.url中的ip和端口，配置为CIM数据汇聚管理平台的ip和端口

ii. 当使用minio存储日志时，配置log.server.enable为true，并配置日志服务（data-log-manage）的ip和端口；

当使用本地文件存储日志时，配置log.server.enable为false

iii. 当采用local模式运行时，data-wrangling-worker.cluster-mode配置为local；当采用集群模式运行时，data-wrangling-worker.cluster-mode配置为yarn，此时，需要将hadoop集群的core-site.xml、yarn-site.xml、hdfs-site.xml、mapred-site.xml等四个文件拷入config文件夹，同时，如果data-wrangling服务部署机器不在hadoop集群内的话，需要在部署机器上的/etc/hosts中配置集群服务器的ip和hostname

```yml
# 服务端口
server:
  port: 8085
# spark任务提交模式
data-wrangling-worker:
  cluster-mode: local
# 任务执行完成，回调CIM数据汇聚管理平台接口
stdms:
  callback:
    url: http://192.168.32.89:9090/stdms/feed-back/ch-task-status?objectId=%s&status=%d
# 是否将日志发送到日志服务并存储到minio
log:
  service:
    enable: true
    url: http://127.0.0.1:8089/save
```

b. 启动、停止

- 启动：

  进入bin路径，执行

  ```shell
  ./start.sh
  ```

- 停止：

  ```
  Kill -9 进程号
  ```

c. 服务验证

i. 服务启动验证与版本信息查询

将以下地址中的ip和端口修改为实际部署ip和端口，请求接口，查看版本信息

请求地址：http://localhost:8085/about

请求方式：GET

ii. 服务功能验证

任务提交验证：将以下地址中的ip和端口修改为实际部署ip和端口，配置正确的数据库连接信息，在数据库中创建user表，并填入任意测试数据，如下述案例表示查询本地mysql的demo库中的user表，并将user表中的数据同步到user1表。

请求地址：http://localhost:8085/data-wrangling

请求方式：POST

请求参数体：

```json
{
   "taskId": "7fdbcc680a4f4e929bc72d89856b00d3",
   "taskName": "lz-test",
   "nodes": [
      ......
   ],
   "lines": [
      {
         "from": "1",
         "to": "2"
      }
   ]
}
```

返回结果示例：

```json
{
   "success": true,
   "message": "提交spark任务成功",
   "content": {
      "taskId": "7fdbcc680a4f4e929bc72d89856b00d3",
      "taskName": "lz-test",
      "createTime": "2022-08-16 17:12:14",
      "level": "INFO",
      "applicationId": "local-1660641128001"
   }
}
```

3.3.7 网关服务

1. 服务信息如下：

表 3-15 服务信息

服务名	服务器	安装目录	端口
gis-gateway	10.190.x.x 10.190.x.x 10.190.x.x	/app/dt/gis-gateway	9999

2. 网关服务
   a. 修改配置文件application-dev.yml
      i. 配置redis连接，可以采用单节点模式或者哨兵模式
      ii. 配置kafka连接以及kafka主题

```yml
# 服务端口
server:
  port: 9999
spring:
  # redis配置，单节点模式配置host和port；哨兵模式配置sentinel
  redis:
    database: 9
    host: 192.168.32.80
    port: 6379
    lettuce:
      pool:
        max-active: 1000 # 连接池最大连接数
        max-wait: -1ms  # 连接池最大阻塞等待时间（使用负值表示没有限制）
```

```
      min-idle: 0 # 连接池中的最小空闲连接
      max-idle: 10 # 连接池中的最大空闲连接
  password: sf.rds.123
  timeout: 10000
kafka:
  bootstrap-servers: 192.168.32.227:9092,192.168.32.103:9092,192.168.32.171:9092
  producer:
    retries: 0
    batch-size: 1000
    buffer-memory: 10240000
    key-serializer: org.apache.kafka.common.serialization.StringSerializer
    value-serializer: org.apache.kafka.common.serialization.StringSerializer
  consumer:
    enable-auto-commit: true
    key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
    value-deserializer: org.apache.kafka.common.serialization.StringDeserializer
    group-id: gateWayGrop
# kafka消息主题
msg:
  topic: prod-log
```

b. 启动、停止

- 启动：

  进入gis-gateway.jar所在路径，执行

  ```shell
  nohup java -jar gis-gateway.jar &
  ```

- 停止：

  ```
  Kill -9 进程号
  ```

c. 服务验证

服务启动验证与版本信息查询

将以下地址中的ip和端口修改为实际部署ip和端口，请求接口，查看版本信息

请求地址：http://localhost:9999/about

请求方式：GET

3.3.8 Web 端二三维地图支撑软件集群

1. 服务信息如下：

表 3-16 服务信息

服务名	服务器	安装目录	端口
GeoServer	10.190.x.x 10.190.x.x 10.190.x.x	/app/apache-28080	admin/ geoserver

2. 软件安装

a. 安装NFS共享目录

```
yum install -y nfs-utils
mkdir -p /app/nfs
mkdir -pv /app/nfs/nfs-data
chmod 666 -R /app/nfs log:
#修改配置文件根据服务器所在网段修改
vim  /etc/exports
```

```
/app/nfs    192.168.32.0/24(rw,no_root_squash,no_all_squash,sync)
/usr/sbin/exportfs -r   ##加载nfs配置
#启动服务，顺序不能错
systemctl start rpcbind && systemctl  enable rpcbind
systemctl start nfs  && systemctl enable nfs
#验证
showmount -e localhost
#输出
Export list for localhost:
/app/nfs 10.190.0.0,10.0.0.0/8
```

三节点客户端挂载NFS：

```
#安装nfs
yum install -y nfs-utils
# 启动服务顺序不能错
systemctl start rpcbind && systemctl  enable rpcbind
systemctl start nfs  && systemctl enable nfs
# 查看服务器端共享文件
showmount -e NFS-IP
# 绑定服务器端共享目录
mount -t nfs NFS-IP:/app/nfs/nfs-data   /app/nfs-data
#查看绑定是否成功
df -h
```

图 3-19 图示 1

```
[root@host-172-16-1-177 logs]# df -h
Filesystem                      Size  Used Avail Use% Mounted on
devtmpfs                         16G     0   16G   0% /dev
tmpfs                            16G     0   16G   0% /dev/shm
tmpfs                            16G  4.5M   16G   1% /run
tmpfs                            16G     0   16G   0% /sys/fs/cgroup
/dev/mapper/euleros_host-root   7.3G  2.3G  4.6G  34% /
tmpfs                            16G   32K   16G   1% /tmp
/dev/vda2                       477M   51M  397M  12% /boot
/dev/vda1                       100M  5.8M   95M   6% /boot/efi
/dev/vdb1                       492G  346M  466G   1% /app
172.16.1.116:/app/nfs/nfs-data  492G  544M  466G   1% /app/nfs-data
tmpfs                           3.1G     0  3.1G   0% /run/user/0
[root@host-172-16-1-177 logs]#
```

设置开机自动挂载

```
vim /etc/fstab
NFS-IP:/app/nfs/nfs-data   /app/nfs-data   nfs  defaults  0  0
```

使用tomcat部署geoserver：

修改tomcat/conf/server.xml port = 28080

图 3-20 图示 2

```
    -->
    <Connector port="28080" protocol="org.apache.coyote.http11.Http11Nio2Protocol"
               maxThreads="1000"
               minSpareThreads="100"
               acceptCount="1500"
               disableUploadTimeout="true"
               maxPostSize="10240"
               maxHttpHeaderSize="8192"
               connectionTimeout="20000"
               redirectPort="8443" />
    <!-- A "Connector" using the shared thread pool-->
```

开启自动解压功能

图 3-21 图示 3

```
      <Host name="localhost"  appBase="webapps"
            unpackWARs="true" autoDeploy="true">
      <!--<Context path="/**" docBase="***" reloadable="false"
```

b. 下载geoserver部署包

https://build.geoserver.org/geoserver/2.20.x/geoserver-2.20.x-2022-09-01-war.zip

```
unzip  geoserver-2.20.x-2022-09-01-war.zip
```

c. 设置nfs共享目录

修改geoserver/tomcat-9091/webapps/geoserver/WEB-INF/web.xml

图 3-22 图示 4

```
<context-param>
   <param-name>GEOSERVER_DATA_DIR</param-name>
   <param-value>/app/nfs/nfs-data</param-value>
</context-param>
```

d. 安装插件

下载地址：https://build.geoserver.org/geoserver/2.20.x/community-2022-09-01/

- 根据版本选择下载geoserver-2.20-SNAPSHOT-jms-cluster-plugin.zip
- 将解压后的所有.jar包复制到tomcat中webapps/geoserver/WEB-INF/lib目录下
- 重启tomcat,等待geoserver启动成功
- 在共享目录/app/nfs/nfs-data目录下生成cluster目录，如果没有配置web.xml里面的共享目录默认在webapps/geoserver/data目录下生成cluster目录

e. 服务验证

浏览器访问ip:port//geoserver

3.3.9 CIM 数据汇聚管理软件

1. 服务信息如下：

表 3-17 服务信息

服务名	服务器	安装目录	端口
stdms-service-app	10.190.x.x 10.190.x.x 10.190.x.x	/app/stdms-server	8888
stdms	10.190.x.x 10.190.x.x	/app/stdms-tomcat-9.0.76	9090
xxl-job-admin	10.190.x.x 10.190.x.x	/app/stdms-xxljobs	8181

2. 安装stdms-service-app服务

config文件夹和 stdms-service-app-1.0.0-SNAPSHOT.jar 放在同级目录。

a. 修改配置文件application.yml，配置启动端口和数据源信息

```
server:
  port: 8888
……
  datasource:
    type: com.zaxxer.hikari.HikariDataSource
    driverClassName: org.postgresql.Driver
    url: jdbc:postgresql://192.168.32.44:5432/stdms_prd?
currentSchema=public&characterEncoding=UTF-8&reWriteBatchedInserts=true
    username: postgres
    password: ******
```

b. 修改redis-prod.properties

配置服务器redis的ip 端口用户名密码 redis.auth.database=11 建议保留默认

```
redis.session.fqn=gis-dsb-core-stdms-test
redis.session.database=4
redis.session.masterName=mymaster
redis.session.password=******
#哨兵配置，多个用逗号分隔
redis.session.sentinels=192.168.32.80:26379,192.168.32.80:26380,192.168.32.80:26381
```

c. 启动、停止

- 启动：

  进入stdms-service-app-1.0.0-SNAPSHOT.jar所在路径，执行

  ```shell
  nohup java -jar stdms-service-app-1.0.0-SNAPSHOT.jar &
  ```

- 停止：

  ```
  Kill -9 进程号
  ```

d. 服务验证

将以下地址中的ip和端口修改为实际部署ip和端口查看返回信息

请求地址：http://ip:8888/stdms/api/xml

3. 安装xxl-job-admin服务

application-config.yml 和 xxl-job-admin-2.3.0.jar放在同级目录。

a. 修改配置文件application-config.yml，配置启动端口和数据源信息

```
server:
  port: 8181
  servlet:
    context-path: /xxl-job-admin
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    type: DruidDataSource
    url: jdbc:mysql://192.168.32.3:3306/stdms_xxl_job_prd?
useUnicode=true&characterEncoding=UTF-8&autoReconnect=true&serverTimezone=Asia/
Shanghai
    username: root
    password: Fengtu12#$
```

b. 启动、停止

- 启动：

  进入xxl-job-admin-2.3.0.jar所在路径，执行

  ```shell
  nohup java -jar xxl-job-admin-2.3.0.jar &
  ```

- 停止：
  Kill -9 进程号

c. 服务验证

将以下地址中的ip和端口修改为实际部署ip和端口 登录系统

请求地址：http://ip:8181/xxl-job-admin

用户名密码 admin/******

3.4 平台运行维护软件部署

3.4.1 服务分布

表 3-18 服务分布

服务名	服务器	安装目录	端口
cas uniform-auth	10.190.x.x 10.190.x.x	/app/apache-tomcat-9.0.64_uniform_auth/	8001

3.4.2 服务部署

1. 安装包准备
   - 使用 Auto_Deploy 中的apache-tomcat-9.0.58.tar.gz包。
   - 将uniform-auth.war和cas.war 放到 webapps 目录下。
   - 将 application-config.yml 和 cas.properties 放到 config 目录下，config 和 webapps 同级目录，也和 conf 同级目录。
   - 将 uniform_auth.sql 导入到 mysql。
2. 配置数据库、服务 IP 和端口

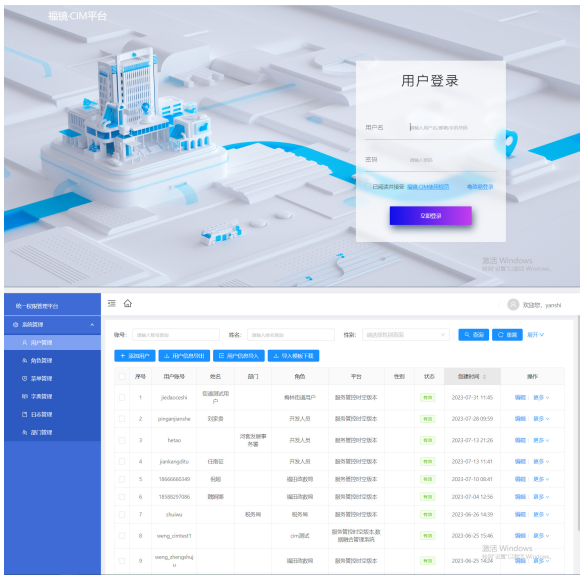
   cas 表名和字段需修改成 uniform_auth.sql 中 sys_user 表名和字段。

```
#修改数据库：
        url: jdbc:mysql://192.168.32.3:3306/uniform_auth?
autoReconnect=true&useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNul
l&useSSL=false&allowMultiQueries=true&rewriteBatchedStatements=true
        username: root
        password: Fengtu12#$
# cas相关
cas:
  # 是否启用cas，如果不启用则不做权限验证拦截，创建默认session，使用CasGuestUser作为默认session
用户（应对开放型系统的`偷懒`方案，不建议使用）
  enable: true
  host:
 # CAS服务器内网地址，用于服务端对服务端调用，例如验证ticket等（在办公环境可能和 client2-server
相同）
    server-2-server: http://192.168.32.61:8001/cas
    # CAS对用户访问地址，用于配置用户在客户端跳转的登录登出目的地址
    client-2-server: https://10.190.x.x/cas
logged.url 支持自定义跳转页面。访问 /cas/login，会跳转到 logged.url 页面。
访问 /cas/login?service=xx，会跳转到 xx 页面。
```

3. 页面访问

https://10.190.x.x/uniform-auth
预置账户：
admin/sf@xxx（管理员账号）
yanshi/xxxx（超管账号）

图 3-23 访问 a



3.5 二三维底板服务维护软件部署

3.5.1 服务分布

表 3-19 服务分布

服务名	服务器	安装目录	端口
cas uniform-auth	10.190.x.x 10.190.x.x	/app/apache-tomcat-9.0.64_uniform_auth/	8001

3.5.2 服务部署

1. 配置修改
   /app/appdeploy/portal/ser-portal/config/application-dev.yml

图 3-24 配置修改 1

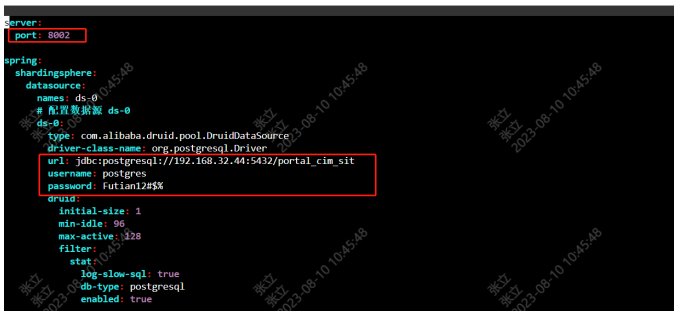


图 3-25 配置修改 2

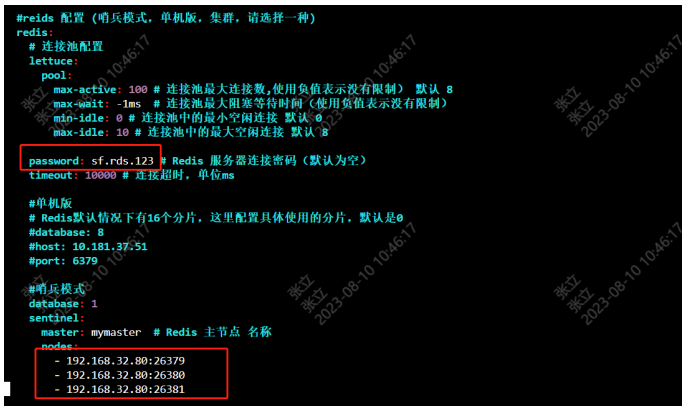


图 3-26 配置修改 3

图 3-27 配置修改 4

图 3-28 配置修改 5

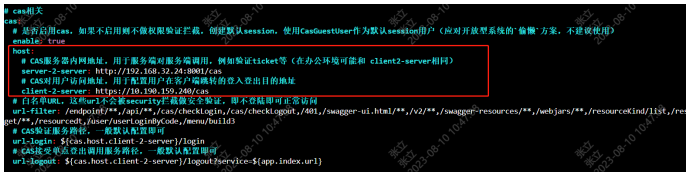
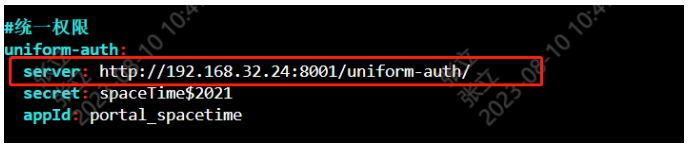
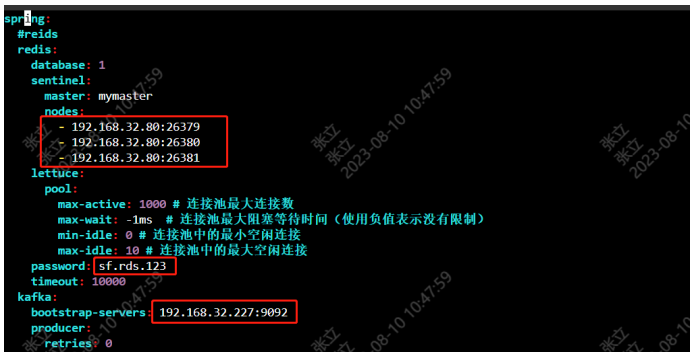


图 3-29 配置修改 6



/app/appdeploy/portal/ser-gateway/config/application-dev.yml

图 3-30 配置修改 7



其它，application.yml，tomcat 的 server.xml 等，端口视情况修改。

2. 页面访问

   https://10.190.x.x/

   使用 cas 账户：

   admin/sf@xxx（管理员账号）

   yanshi/xxxx（超管账号）

图 3-31 页面访问

3.6 CIM 全时空门户部署

3.6.1 服务分布

表 3-20 服务分布

服务名	服务器	安装目录	端口
ser-gateway	10.190.x.x 10.190.x.x	/app/appdeploy/portal/ser-gateway	8003
Ser-portal		/app/appdeploy/portal/ser-portal	8002

3.6.2 服务部署

1. 进入服务器，并移动到/app/appdeploy目录下
2. 启动&停止
   启动：nohup java -jar cim-1.0-SNAPSHOT.jar &
   停止：找到pid， 执行： kill -9 pid

3.7 二三维底板空间分析软件

3.7.1 服务分布

表 3-21 服务分布

服务名	服务器	安装目录	端口
Sf3d	10.190.x.x 10.190.x.x 10.190.x.x	/app/apache-tomcat-9.0.64_sf3d	8080
sfmap		/app/appdeploy/ sfmapTile_V6.0.SP2_arm	8006

3.7.2 sf3d 服务部署

1. 将webapps里面的sf3d文件夹复制到目标服务器的发布目录，即tomcat/webapps目录里面；
2. 在服务器上解压“/sf3d/编码”里面的压缩包，这些数据需要到前端web管理界面配置数据路径，例如dem数据“http://IP:8080/sf3d/440300/dem/layer.json”；
3. 修改sf3d/WEB-INF/classes下面的jdbc.properties里面
   test_db.url=jdbc:postgresql://192.168.32.44:5432/address_3d?characterEncoding=utf-8，test_db.username=* test_db.password=*；

4. 重启tomcat；
5. 在浏览器输入http://10.190.x.x/sf3d/area/list?adcode=当地编码，返回小区列表则服务OK。

3.7.3 sfmap 服务部署

1. PC矢量服务部署步骤
   a. 服务包部署
      tar -zxf sfmapTile_V6.0.tar.gz；
   b. 数据文件部署
      将矢量数据《城市拼音.dat》放入data/PCVectorTile目录下；
   c. 授权文件部署
      i. 授权文件放在renderlicense目录下；
      ii. 修改renderlicense/renderlicense.properties文件：
         1) 本地授权只需要将authorization.type=1，其他不用修改；
         2) 远程授权需要将authorization.type=2，且需要修改ip及端口。

      说明

      远程授权重启服务时需要等待10s再确认服务状态，否则会提示授权错误，10s之后正常。

      图 3-32 授权

```
#授权文件解密库名称
endecheck.name=libEndeCheck.so

#授权方式
#1：本地授权：采用授权文件授权，验证授权文件的ip和时间
#2：远程授权：采用授权文件和lc授权服务授权，验证授权文件时间
authorization.type=2

#---以下当使用远程授权时有效---

#lc授权服务ip
ip=10.82.244.165

#lc授权服务端口
port=7777
```

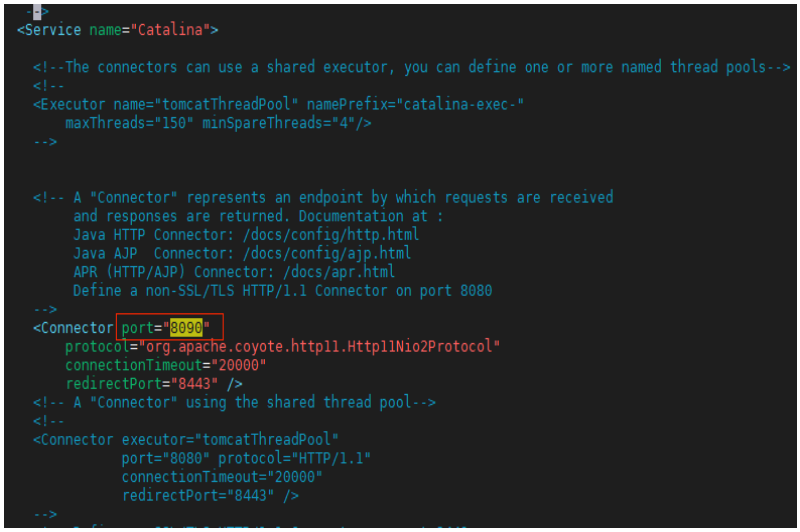
   d. 服务端口修改
      i. mre_server服务端口修改
         文件路径：/mre_server/conf/mre_server.properties

图 3-33 服务端口修改 1

ii. render_server服务端口修改

文件路径：/render_server/conf/server.xml

图 3-34 服务端口修改 2



e. 启动及停止服务

启动服务：

i. 进入bin目录（注意非服务下的bin目录）；

ii. sh start.sh 3（启动1：mre_server;2：render_server；3：两个服务都启动）

停止服务：

sh stop.sh 3（停止1：mre_server;2：render_server；3：两个服务都停止）

如果端口有修改，需要同步修改2个脚本中的port值。

2. 栅格服务部署

a. 服务包部署

tar -zxf sfmapTile_V6.0.tar.gz；

b. 数据文件部署

将栅格数据包拷到服务同一目录，并解压数据包，以3857投影数据为例：

tar -zxf 数据包名 -C ./data/PCRasterTile/3857

c. 授权文件部署

i. 将授权文件binary.dat放在renderlicense目录下；

ii. 修改renderlicense/renderlicense.properties文件：

- 本地授权只需要将authorization.type=1，其他不用修改
- 远程授权需要将authorization.type=2，且需要修改ip及端口

说明

远程授权重启服务时需要等待10s再确认服务状态，否则会提示授权错误，10s之后正常。

图 3-35 栅格服务部署

```
#授权文件解密库名称
endecheck.name=libEndeCheck.so

#授权方式
#1：本地授权：采用授权文件授权，验证授权文件的ip和时间
#2：远程授权：采用授权文件和lc授权服务授权，验证授权文件时间
authorization.type=2

#---以下当使用远程授权时有效---

#lc授权服务ip
ip=10.82.244.165

#lc授权服务端口
port=7777
```

d. 启动服务

i. 进入bin目录（注意非服务下的bin目录）；

ii. sh start.sh 3（启动1：mre_server;2：render_server；3：两个服务都启动）

停止服务：

sh stop.sh 3（停止1：mre_server;2：render_server；3：两个服务都停止）

如果端口有修改，需要同步修改2个脚本中的port值

e. 验证部署：获取栅格服务maptile（3857投影）

http://IP:8090/MapTileService/rt?x=421&y=193&z=9&data_name={adcode}，修改IP值和adcode值，可获取如下图片。

图 3-36 验证部署

4 修订记录

表 4-1 修订记录

发布日期	修订记录
2023-09-07	第一次正式发布。