大数据环境部署文档

1. 系统部署依据

- 大数据管理平台：DataSophon
- 大数据监控组件：Prometheus、Grafana、AlertManager
- 大数据组件：HDFS、YARN、Zookeeper、Kafka
- 计算组件：Hive、Flink、Spark3、Trino、Doris
- 搜索和分析引擎：Elasticsearch

1.1 名词解释

1. DataSophon：DataSophon是致力于自动化监控、运维、管理大数据基础组件和节点的，帮助您快速构建起稳定，高效的大数据集群服务。
2. Prometheus：Prometheus 是一个开源的服务监控系统和时间序列数据库。
3. Grafana：grafana是一个非常酷的数据可视化平台，常常应用于显示监控数据
4. AlertManager：Alertmanager 主要用于接收 Prometheus 发送的告警信息，它支持丰富的告警通知渠道，而且很容易做到告警信息进行去重，降噪，分组等，是一款前卫的告警通知系统
5. HDFS：HDFS是Hadoop使用的分布式文件系统，能存储和处理大规模数据。HDFS的设计目标是在标准硬件上运行，从而提供高容错性，并且能够处理已存储的大量数据。
6. YARN：Apache Hadoop YARN （Yet Another Resource Negotiator，另一种资源协调者）是一种新的 Hadoop 资源管理器，它是一个通用资源管理系统，可为上层应用提供统一的资源管理和调度，它的引入为集群在利用率、资源统一管理和数据共享等方面带来了巨大好处。
7. Zookeeper：zookeeper(以下简称ZK)是一个分布式的，开放源码的分布式应用程序协调服务，它是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、域名服

务、分布式同步、组服务等。ZK的目标就是封装好复杂易出错的关键服务，将简单易用的接口和性能高效、功能稳定的系统提供给用户。

8. Kafka：Kafka是Apache旗下的一款分布式流媒体平台，Kafka是一种高吞吐量、持久性、分布式的发布订阅的消息队列系统。它主要用于处理消费者规模网站中的所有动作流数据。动作指(网页浏览、搜索和其它用户行动所产生的数据)。

9. Hive：hive是基于Hadoop的一个数据仓库工具，用来进行数据提取、转化、加载，这是一种可以存储、查询和分析存储在Hadoop中的大规模数据的机制。

10. Flink：Apache Flink是由Apache软件基金会开发的开源流处理框架，其核心是用Java和Scala编写的分布式流数据流引擎。Flink以数据并行和流水线方式执行任意流数据程序，Flink的流水线运行时系统可以执行批处理和流处理程序。此外，Flink的运行时本身也支持迭代算法的执行。

11. Spark3：Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎，拥有Hadoop MapReduce所具有的优点；但不同于MapReduce的是——Job中间输出结果可以保存在内存中，从而不再需要读写HDFS，因此Spark能更好地适用于数据挖掘与机器学习等需要迭代的MapReduce的算法。

12. Trino：trino是一种分布式 SQL 查询引擎，旨在查询分布在一个或多个异构数据源上的大型数据集。它通过在整个集群的服务器上分配处理任务来实现横向扩展，基于这种架构，Trino查询引擎可以在集群内的计算节点并行处理海量数据的SQL查询。

13. Doris：Apache Doris 是一个基于 MPP 架构的高性能、实时的分析型数据库，以极速易用的特点被人们所熟知，仅需亚秒级响应时间即可返回海量数据下的查询结果，不仅可以支持高并发的点查询场景，也能支持高吞吐的复杂分析场景。

14. Elasticsearch：Elasticsearch 是一个分布式、RESTful 风格的搜索和数据分析引擎，能够解决不断涌现出的各种用例。

2. 准备环境

2.1 服务器规划

序号	服务器名称	服务器IP	服务器角色	服务器配置	服务器用途
1	ddp1	192.168.0.12	zk、nn、dn、yarn、hive、flink、spark，trino、mysql	16C64G	管理节点与Hadoop集群
2	ddp2	192.168.0.13	dn、yarn、kafka、trino	16C64G	Hadoop集群
3	ddp3	192.168.0.14	zk、dn、yarn、kafka、trino	16C64G	Hadoop集群
4	ddp4	192.168.0.15	zk、nn、dn、yarn、hive、trino	16C64G	Hadoop集群
5	ddp5	192.168.0.16	dn、yarn、kafka、trino	16C64G	Hadoop集群
6	esnode1	192.168.0.9	ESNode	16C64G	Elasticsearch集群
7	esnode2	192.168.0.10	ESNode	16C64G	Elasticsearch集群
8	esnode3	192.168.0.11	ESNode	16C64G	Elasticsearch集群
9	dorisnode1	192.168.0.4	Fe、Be、Broker	16C64G	Doris集群
10	dorisnode2	192.168.0.5	Fe、Be、Broker	16C64G	Doris集群
11	dorisnode3	192.168.0.6	Be、Broker	16C64G	Doris集群
12	dorisnode4	192.168.0.7	Be、Broker	16C64G	Doris集群
13	dorisnode5	192.168.0.8	Be、Broker	16C64G	Doris集群

2.2 按照上述规划修改服务器名称，重启后生效

```
hostnamectl set-hostname youhostname
```

2.3 修改主机hosts，每台都需要

```
vim /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.0.4 dorisnode1
192.168.0.5 dorisnode2
192.168.0.6 dorisnode3
192.168.0.7 dorisnode4
192.168.0.8 dorisnode5

192.168.0.9 esnode1
192.168.0.10 esnode2
192.168.0.11 esnode3

192.168.0.12 ddp1
192.168.0.13 ddp2
192.168.0.14 ddp3
192.168.0.15 ddp4
192.168.0.16 ddp5
```

2.4 修改打开文件数

```
echo '*     soft   nofile  1024000' >>/etc/security/limits.conf
echo '*     hard   nofile  1024000' >>/etc/security/limits.conf
```

2.5 关闭防火墙，关闭selinux

```
systemctl stop firewalld.service
systemctl disable firewalld.service
vim /etc/selinux/config
#修改
SELINUX=disabled
```

2.6 修改swappiness与默认值

```
echo 'vm.swappiness=0'>> /etc/sysctl.conf
vim   /etc/rc.local
echo never > /sys/kernel/mm/transparent_hugepage/defrag
echo never > /sys/kernel/mm/transparent_hugepage/enabled
sysctl -p
```

2.7 免密登录配置，需要与管理节点进行互信，主要本机也需要，以下命令只有管理节点执行

在ddp1服务上执行以下命令，一路回车,生成无密码的密钥对，并同步到集群

```
ssh-keygen -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.4
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.5
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.6
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.7
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.8
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.9
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.10
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.11
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.12
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.13
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.14
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.15
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.0.16
```

2.8 没有/data则创建/data，如果分系统盘和数据盘，则将/home改为/data

```
mkdir /data
#如果有/home则直接进入/etc/fstab，将/home改为/data
vim /etc/fstab

/dev/mapper/uos-root    /                       xfs     defaults        0 0
UUID=a57c6623-e92e-464e-bb56-c62b886b2949 /boot                   xfs     defaults
0 0
/dev/mapper/uos-home    /home(改为/data)                  xfs     defaults        0 0
/dev/mapper/uos-swap    none                    swap    defaults        0 0

#重启后生效
```

2.9 ntp时钟同步

每台都安装ntp

```
#先查看是否已经安装ntp
rpm -q ntp
#没有则安装
yum install ntp
#设置自启
systemctl enable ntpd.service
chkconfig --list ntpd
```

服务端设置

```
vim /etc/ntp.conf
driftfile /var/lib/ntp/drift
restrict default kod nomodify notrap nopeer noquery
restrict -6 default kod nomodify notrap nopeer noquery
restrict 127.0.0.1
restrict -6 ::1
restrict 192.168.0.0 mask 255.255.255.0 nomodify notrap
server 127.127.1.0
fudge  127.127.1.0 stratum 10
includefile /etc/ntp/crypto/pw
keys /etc/ntp/keys
```

客户端

```
vim /etc/ntp.conf
driftfile /var/lib/ntp/drift
restrict 127.0.0.1
restrict -6 ::1
 # 配置时间服务器为本地的时间服务器
server 192.168.0.128
 restrict 192.168.0.128 nomodify notrap noquery
 #server  127.127.1.0     # 客户端不需要同步自己
fudge   127.127.1.0 stratum 10
 includefile /etc/ntp/crypto/pw
 keys /etc/ntp/keys
```

启动或查看状态

```
systemctl start ntpd
systemctl status ntpd
systemctl enable ntpd
```

同步成功

```
[root@dorisnode2 ~]# ntpstat
synchronised to NTP server (10.1.187.1) at stratum 9
   time correct to within 4078 ms
   polling server every 1024 s
```

2.10 创建DDP部署包路径，并将DDP包放置进去

```
mkdir -p /opt/datasophon/DDP/packages
```

2.11 将datasophon-manager-1.1.1.tar.gz解压到/opt/datasophon路径下解压

```
tar -zxvf /opt/datasophon/DDP/packages/datasophon-manager-1.1.1.tar.gz -C
/opt/datasophon
```

```
[root@ddp1 ~]# ll /opt/datasophon/datasophon-manager-1.1.1/
总用量 44
drwxr-xr-x 2 root root    31  7月 12 16:44 bin
drwxr-xr-x 7 root root   169  7月 13 11:49 conf
drwxr-xr-x 2 root root    81 10月 22  2022 jmx
drwxr-xr-x 2 root root  8192  7月 12 16:44 lib
-rw-r--r-- 1 root root 11558 12月 21  2022 LICENSE
drwxr-xr-x 2 root root 12288 10月 17 10:00 logs
drwxr-xr-x 2 root root    21  7月 13 11:49 pid
-rw-r--r-- 1 root root  3264  6月 11 18:40 README.md
drwxr-xr-x 3 root root    43  6月 11 18:40 sql
```

2.12 管理服务器部署jdk，解压/opt/datasophon/DDP/packages中,设置环境变量，source后生效

```
tar -zxvf /opt/datasophon/DDP/packages/jdk-8u333-linux-x64.tar.gz -C /usr/local/
vim /etc/profile
export JAVA_HOME=/usr/local/jdk1.8.0_333
export JRE_HOME=$JAVA_HOME/jre
export CLASSPATH=.:$JAVA_HOME/lib:$JRE_HOME/lib:$CLASSPATH
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$PATH
```

2.13 重启服务器

```
reboot
```

2.14 安装mysql数据库，并关闭mysql ssl功能

查看是否开启ssl

```
SHOW VARIABLES LIKE '%ssl%';
```

如果看到have_ssl的值为YES，说明SSL已经开启

如果已经开启，可以修改配置文件my.cnf，追加参数

```
[mysqld]
skip_ssl
```

重启mysql服务后生效

执行以下Sql，并在datasophon数据库中执行文件datasophon.sql脚本

```
CREATE DATABASE IF NOT EXISTS datasophon DEFAULT CHARACTER SET utf8;
grant all privileges on *.* to datasophon@"%" identified by 'ic&Mr@qz9MK5Tun' with
grant option;
GRANT ALL PRIVILEGES ON *.* TO 'datasophon'@'%';

CREATE DATABASE IF NOT EXISTS hive DEFAULT CHARACTER SET utf8;
grant all privileges on *.* to hive@"%" identified by '@pVr*F0sI%)H$3B' with grant
option;
GRANT ALL PRIVILEGES ON *.* TO 'hive'@'%';
FLUSH PRIVILEGES;
```

3. 修改Datasophon配置

3.1 修改datasophon-manager中conf目录下的application.yml 配置文件中数据库链接配置

```
spring:
  datasource:
    type: com.alibaba.druid.pool.DruidDataSource
    url: jdbc:mysql:192.168.0.12:3306/datasophon?
useUnicode=true&characterEncoding=utf-8
    username: *******
    password: *******
    driver-class-name: com.mysql.jdbc.Driver
```

3.2 启动命令

```
启动：sh bin/datasophon-api.sh start api
停止：sh bin/datasophon-api.sh stop api
重启：sh bin/datasophon-api.sh restart api
```

worker类似

```
启动：sh /opt/datasophon/datasophon-worker/bin/datasophon-worker.sh start worker
停止：sh /opt/datasophon/datasophon-worker/bin/datasophon-worker.sh stop worker
重启：sh /opt/datasophon/datasophon-worker/bin/datasophon-worker.sh restart worker
```

3.3 访问管理界面，默认账号密码，admin:admin123

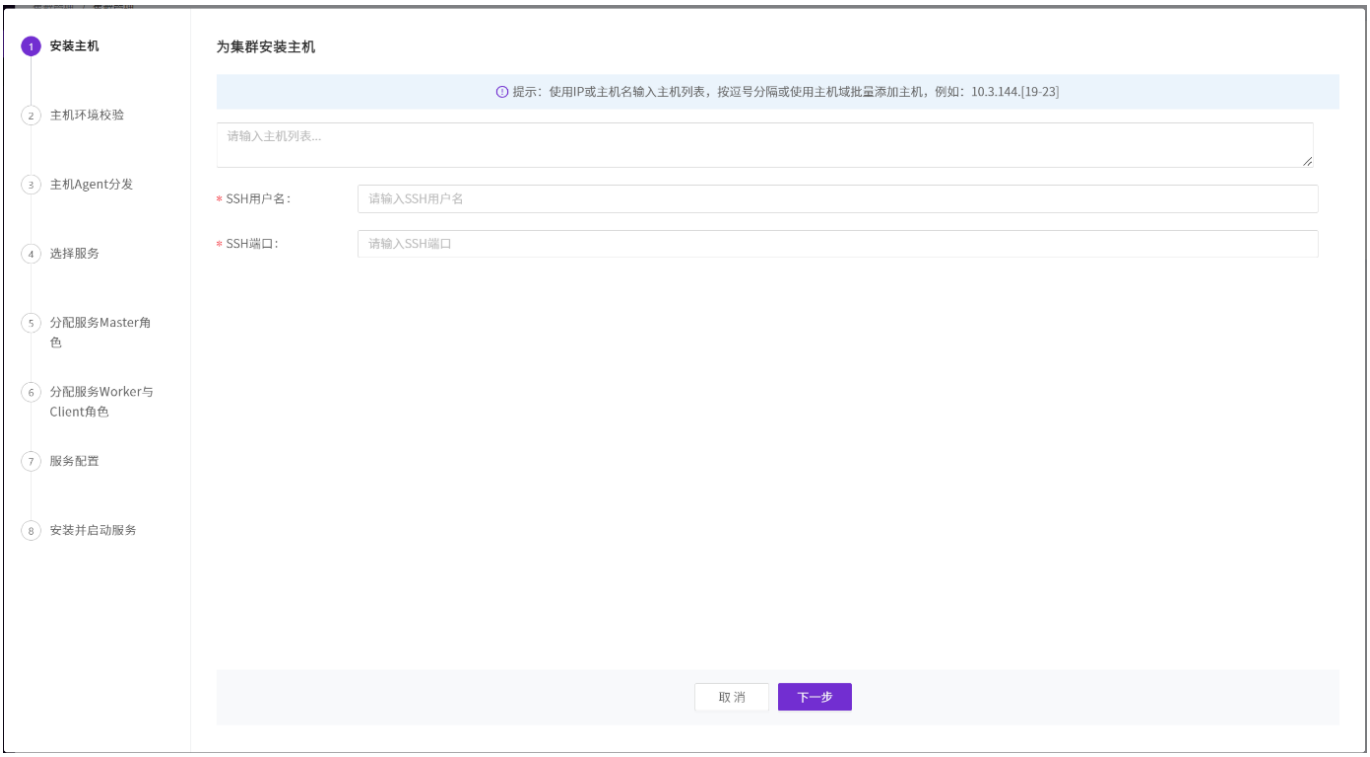
http://192.168.0.12:8081/ddh/#/login

4 使用Datasophon部署组件

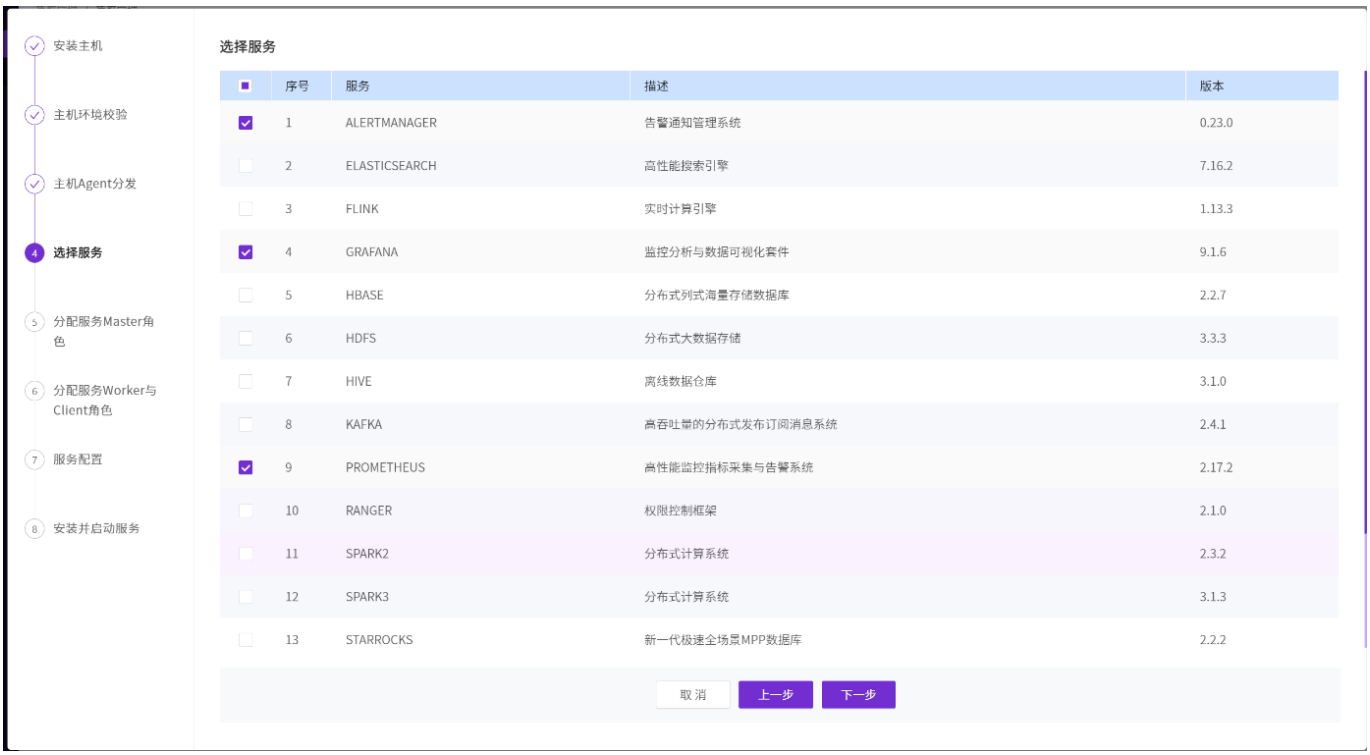
4.1 创建集群

输入集群名称、集群编码、集群框架

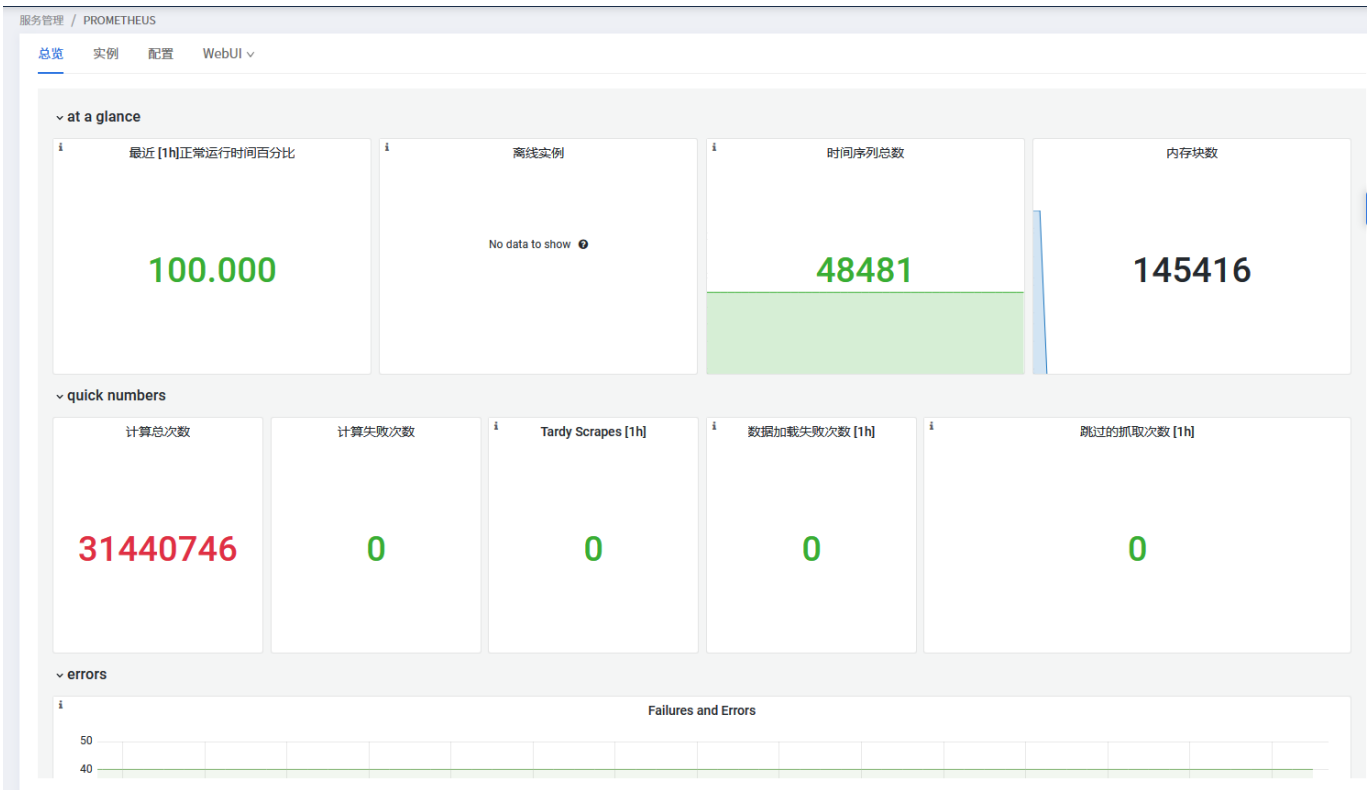
将主机列表输入、并设置ssh用户与密码等



主机校验成功后，初始化配置集群先选择部署AlertManager,Grafana和Prometheus三个组件。DataSophon依赖此三个组件实现系统监控告警管理。

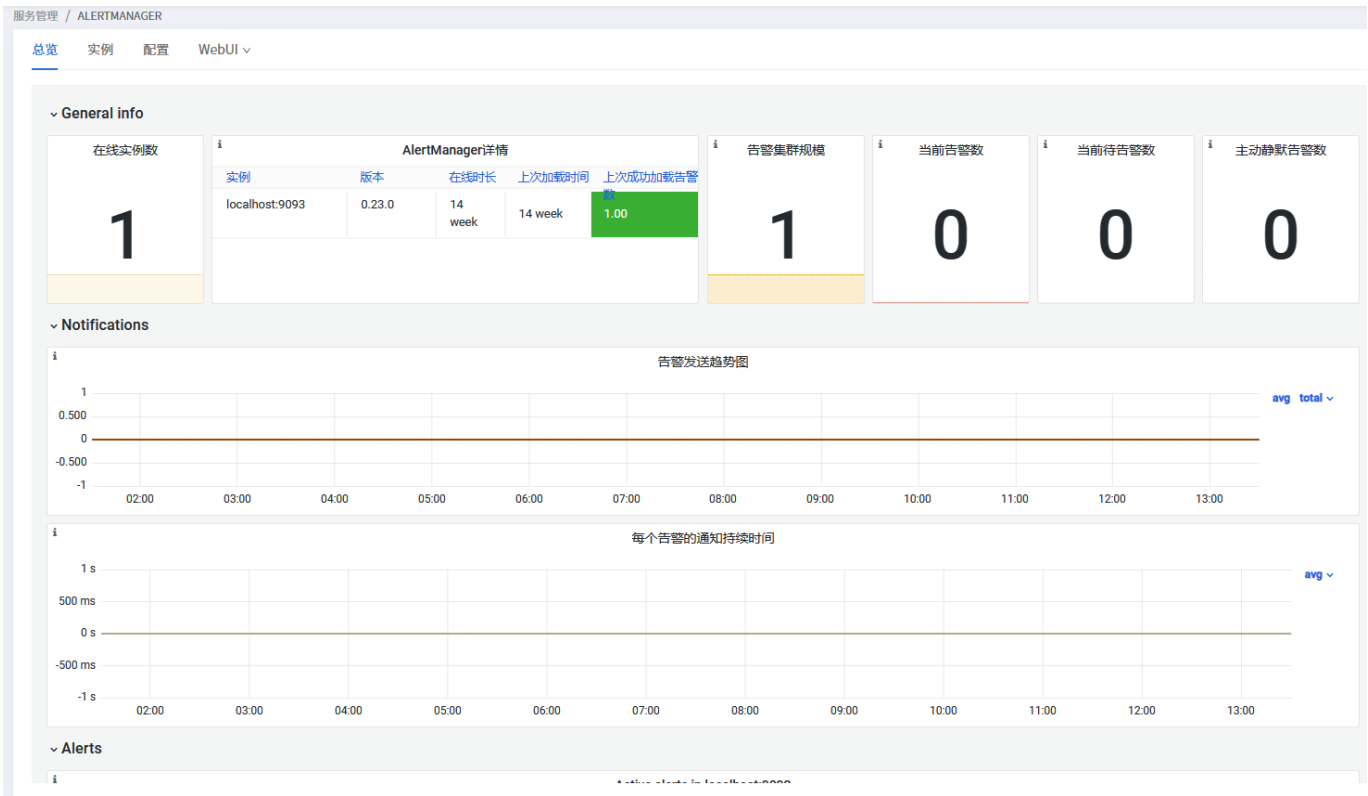


Prometheus



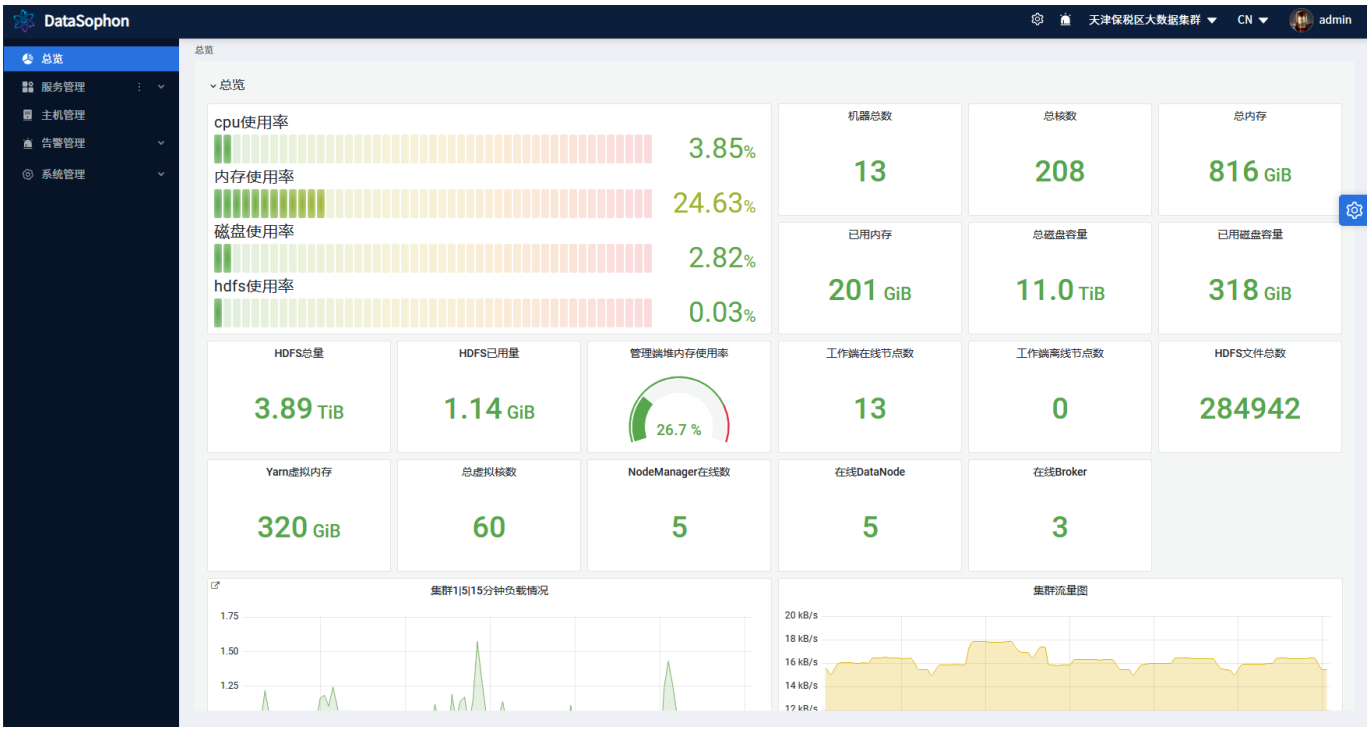
grafana

AlertManager



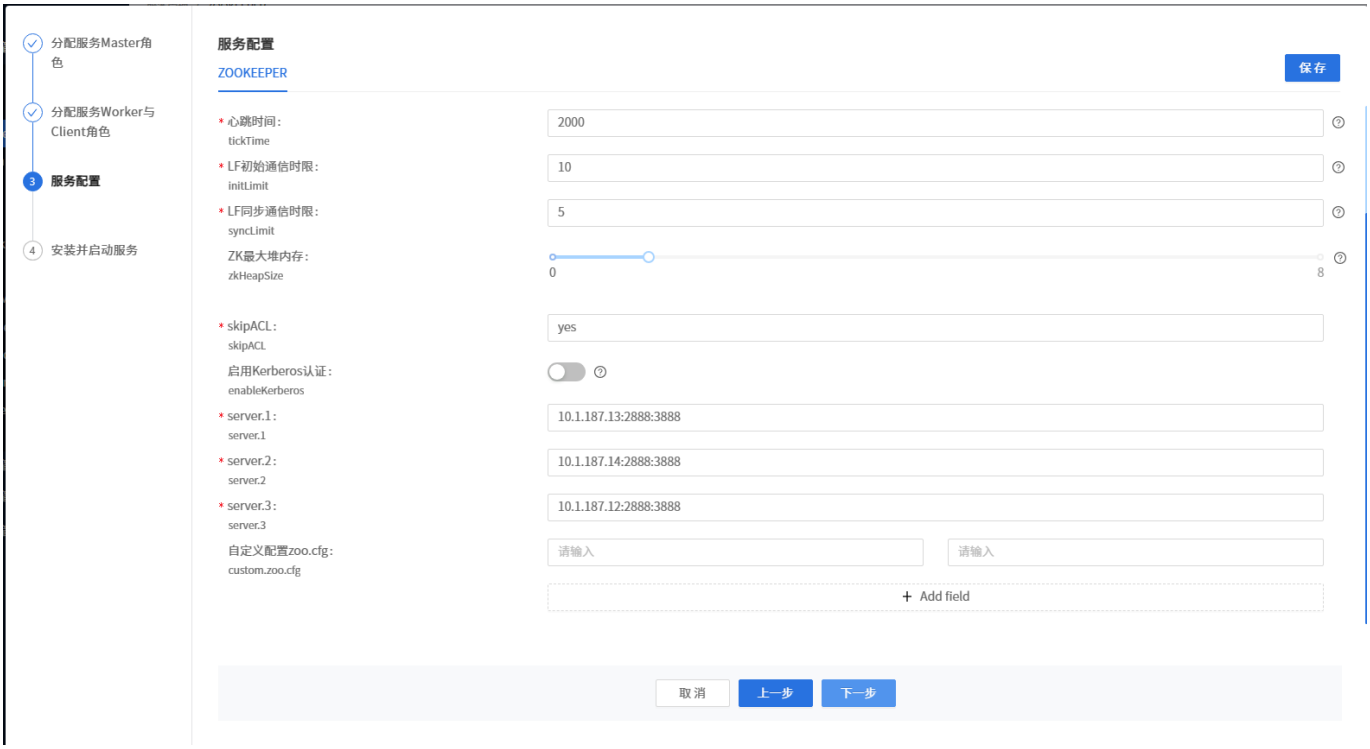
4.2 部署成功后，点击进入进入集群

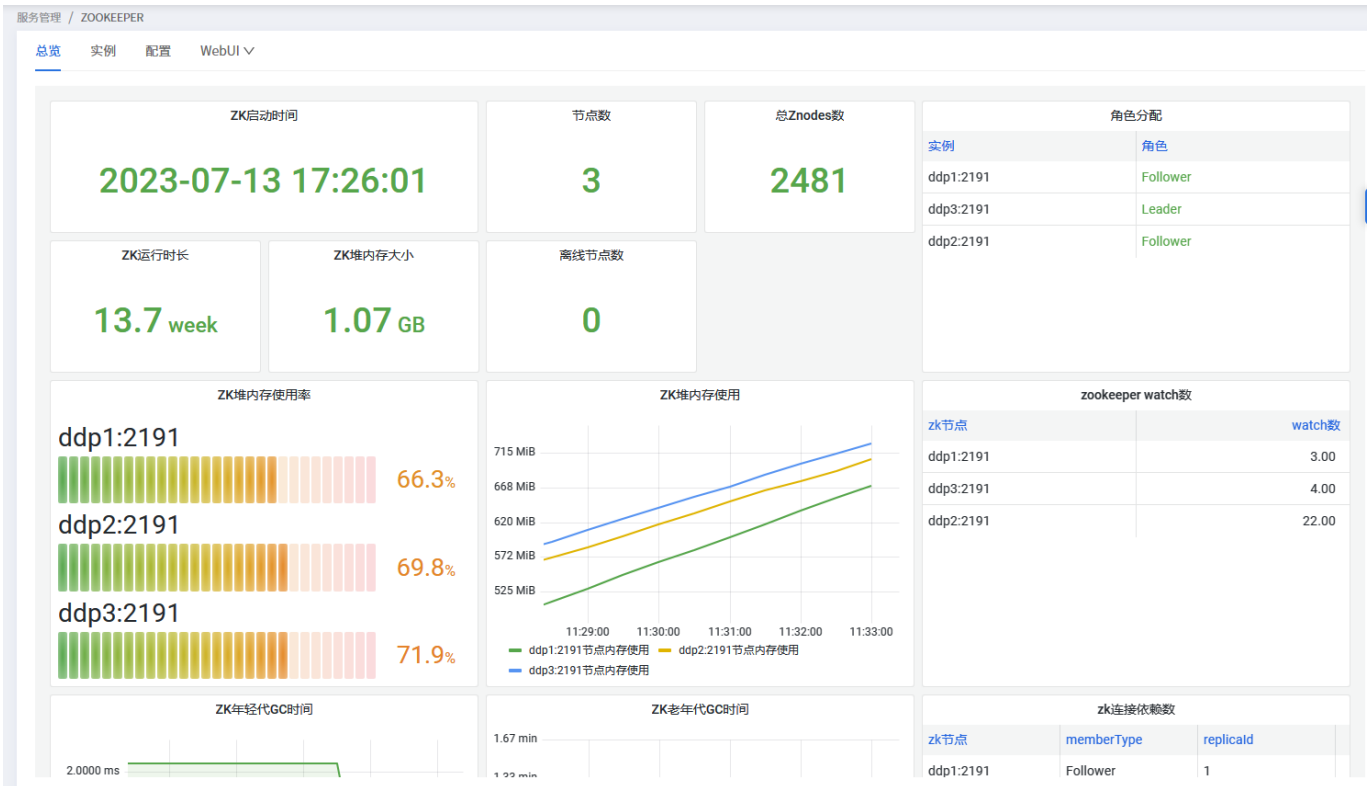
进入总览



点击主机管理，可以看到所有的服务器已经被管理

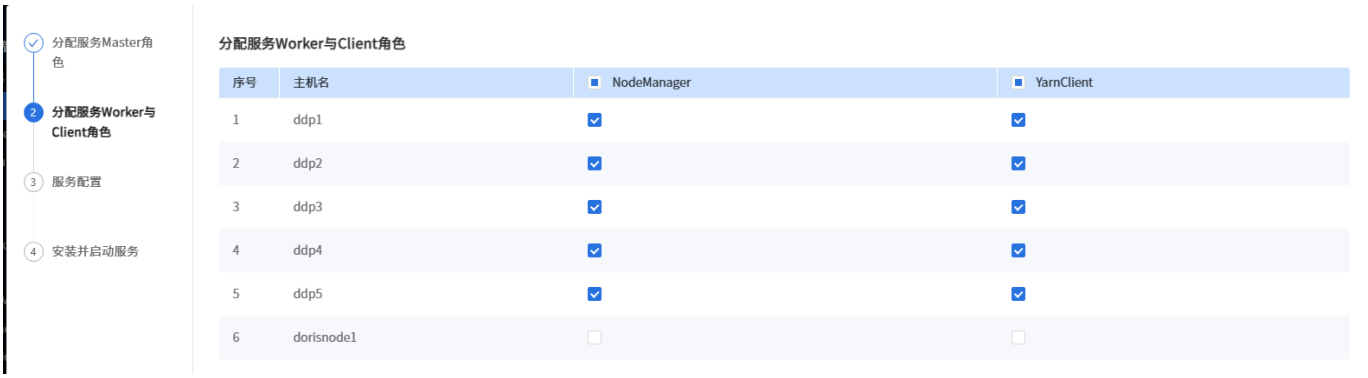
4.3 添加Zookeeper

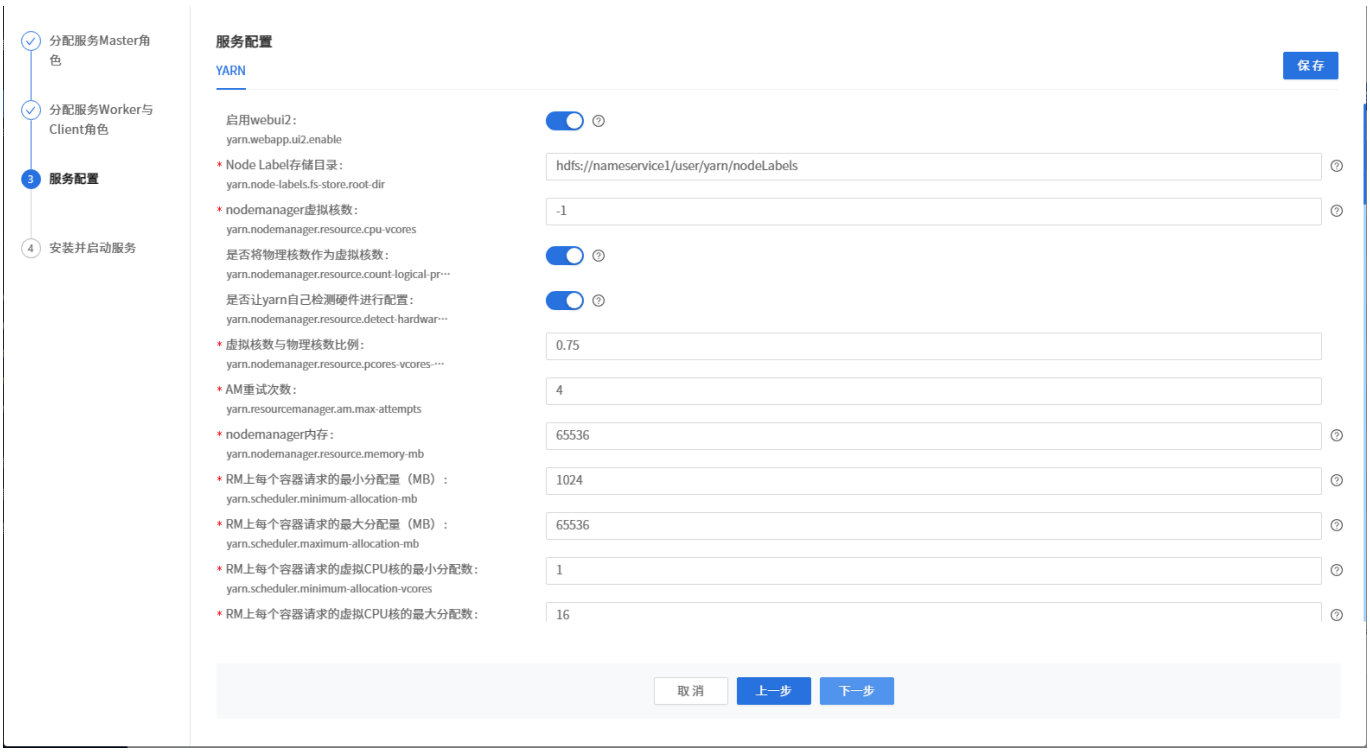

分配服务Master角色
分配服务Worker与Client角色
3 服务配置
4 安装并启动服务
服务配置
ZOOKEEPER
保存
* 心跳时间：
tickTime
2000
* LF初始通信时限：
initLimit
10
* LF同步通信时限：
syncLimit
5
ZK最大堆内存：
zkHeapSize
0
8
* skipACL：
skipACL
yes
启用Kerberos认证：
enableKerberos
* server.1：
server.1
10.1.187.13:2888:3888
* server.2：
server.2
10.1.187.14:2888:3888
* server.3：
server.3
10.1.187.12:2888:3888
自定义配置zoo.cfg：
custom.zoo.cfg
请输入
请输入
+ Add field
取 消
上一步
下一步

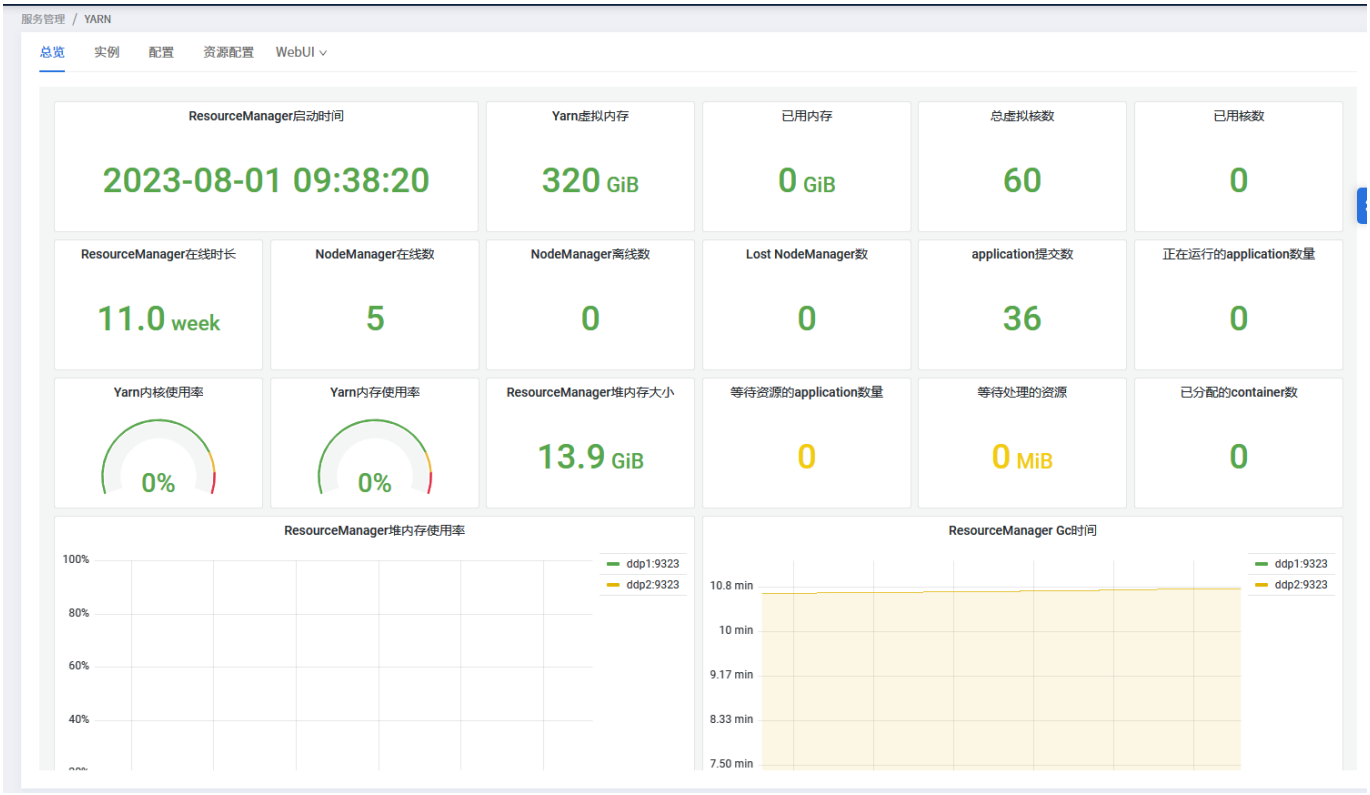

服务管理 / ZOOKEEPER
总览
实例
配置
WebUI
ZK启动时间
2023-07-13 17:26:01
节点数
3
总Znodes数
2481
角色分配
实例
角色
ddp1:2191
Follower
ddp3:2191
Leader
ddp2:2191
Follower
ZK运行时长
13.7 week
ZK堆内存大小
1.07 GB
离线节点数
0
ZK堆内存使用率
ddp1:2191
66.3%
ddp2:2191
69.8%
ddp3:2191
71.9%
ZK堆内存使用
715 MiB
668 MiB
620 MiB
572 MiB
525 MiB
11:29:00
11:30:00
11:31:00
11:32:00
11:33:00
ddp1:2191节点内存使用
ddp2:2191节点内存使用
ddp3:2191节点内存使用
zookeeper watch数
zk节点
watch数
ddp1:2191
3.00
ddp3:2191
4.00
ddp2:2191
22.00
ZK年轻代GC时间
2.0000 ms
ZK老年代GC时间
1.67 min
zk连接依赖数
zk节点
memberType
replicaId
ddp1:2191
Follower
1

4.4 添加Yarn


1 分配服务Master角色
2 分配服务Worker与Client角色
3 服务配置
4 安装并启动服务
分配服务Master角色
* ResourceManager：
ResourceManager
ddp1 × ddp2 ×
* HistoryServer：
HistoryServer
ddp1

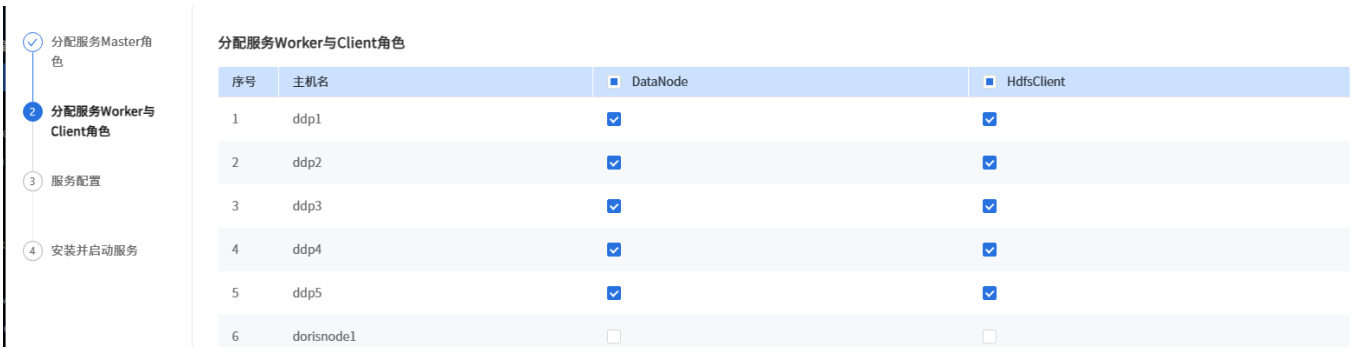

分配服务Master角色
2 分配服务Worker与Client角色
3 服务配置
4 安装并启动服务
分配服务Worker与Client角色
序号 主机名 NodeManager YarnClient
1 ddp1
2 ddp2
3 ddp3
4 ddp4
5 ddp5
6 dorisnode1

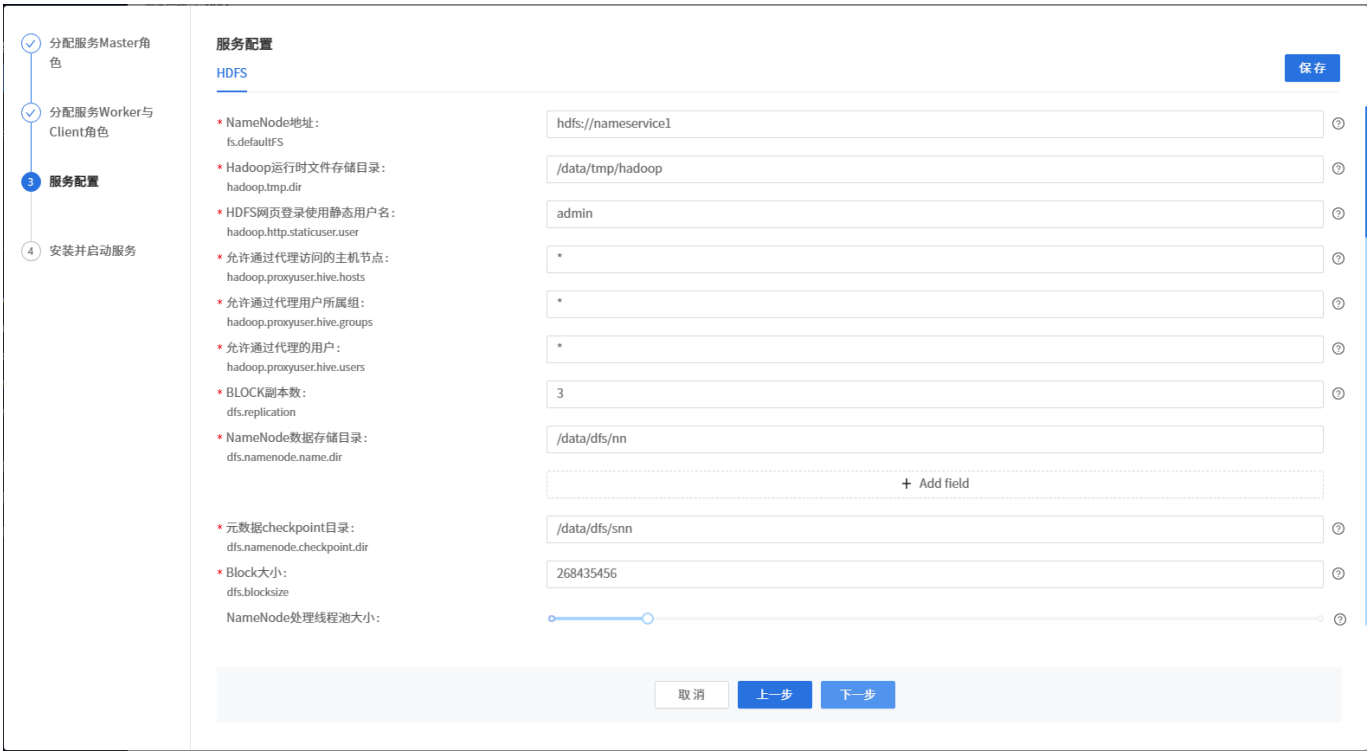

分配服务Master角色
分配服务Worker与Client角色
3 服务配置
4 安装并启动服务
服务配置
YARN
保存
启用webui2：
yarn.webapp.ui2.enable
* Node Label存储目录：
yarn.node-labels.fs-store.root-dir
hdfs://nameservice1/user/yarn/nodeLabels
* nodemanager虚拟核数：
yarn.nodemanager.resource.cpu-vcores
-1
是否将物理核数作为虚拟核数：
yarn.nodemanager.resource.count-logical-pr…
是否让yarn自己检测硬件进行配置：
yarn.nodemanager.resource.detect-hardwar…
* 虚拟核数与物理核数比例：
yarn.nodemanager.resource.pcores-vcores-…
0.75
* AM重试次数：
yarn.resourcemanager.am.max-attempts
4
* nodemanager内存：
yarn.nodemanager.resource.memory-mb
65536
* RM上每个容器请求的最小分配量（MB）：
yarn.scheduler.minimum-allocation-mb
1024
* RM上每个容器请求的最大分配量（MB）：
yarn.scheduler.maximum-allocation-mb
65536
* RM上每个容器请求的虚拟CPU核的最小分配数：
yarn.scheduler.minimum-allocation-vcores
1
* RM上每个容器请求的虚拟CPU核的最大分配数：
16
取消
上一步
下一步

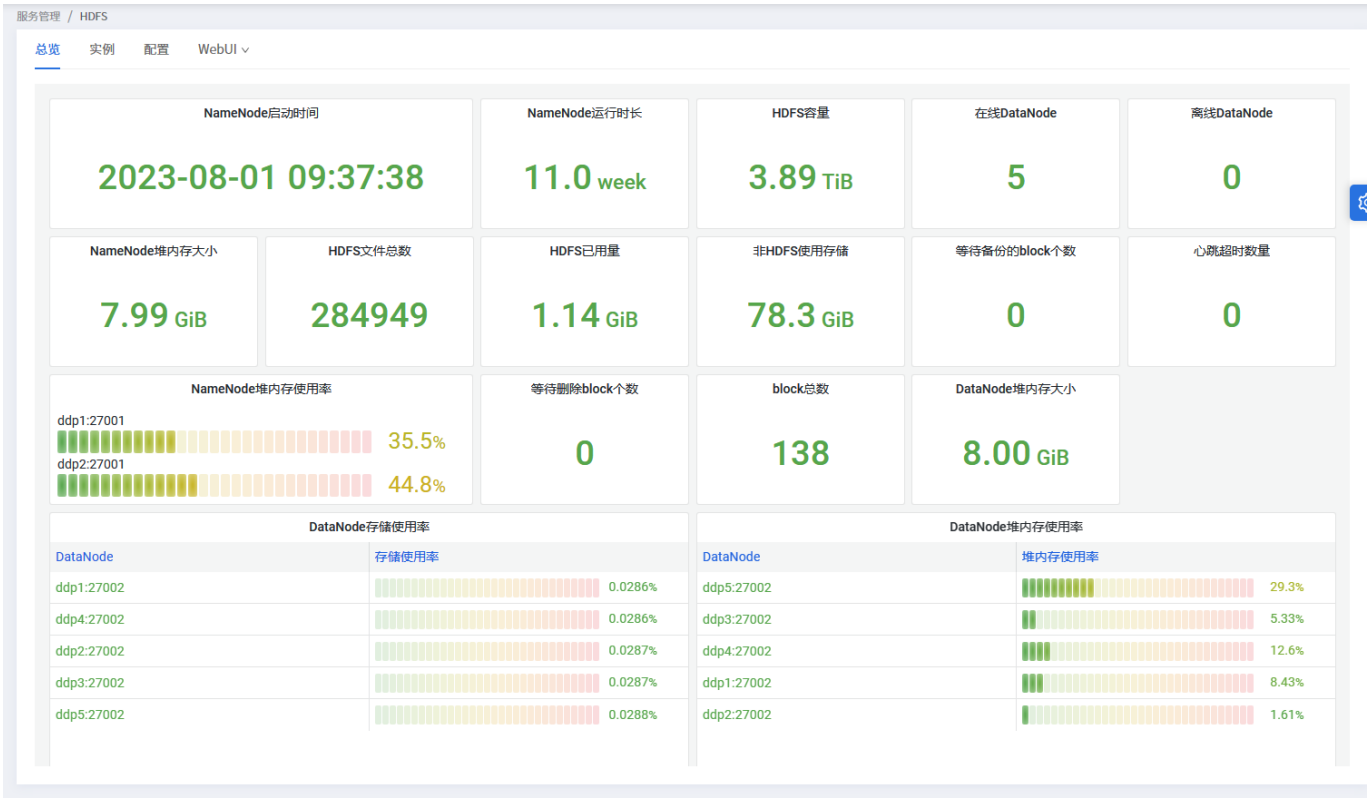

服务管理 / YARN
总览
实例
配置
资源配置
WebUI
ResourceManager启动时间
2023-08-01 09:38:20
Yarn虚拟内存
320 GiB
已用内存
0 GiB
总虚拟核数
60
已用核数
0
ResourceManager在线时长
11.0 week
NodeManager在线数
5
NodeManager离线数
0
Lost NodeManager数
0
application提交数
36
正在运行的application数量
0
Yarn内核使用率
0%
Yarn内存使用率
0%
ResourceManager堆内存大小
13.9 GiB
等待资源的application数量
0
等待处理的资源
0 MiB
已分配的container数
0
ResourceManager堆内存使用率
100%
80%
60%
40%
ddp1:9323
ddp2:9323
ResourceManager Gc时间
10.8 min
10 min
9.17 min
8.33 min
7.50 min
ddp1:9323
ddp2:9323

4.5 添加HDFS


1 分配服务Master角色
2 分配服务Worker与Client角色
3 服务配置
4 安装并启动服务
分配服务Master角色
* JournalNode:
JournalNode
ddp2 × ddp1 × ddp3 ×
* NameNode:
NameNode
ddp1 × ddp2 ×
* ZKFC:
ZKFC
ddp1 × ddp2 ×

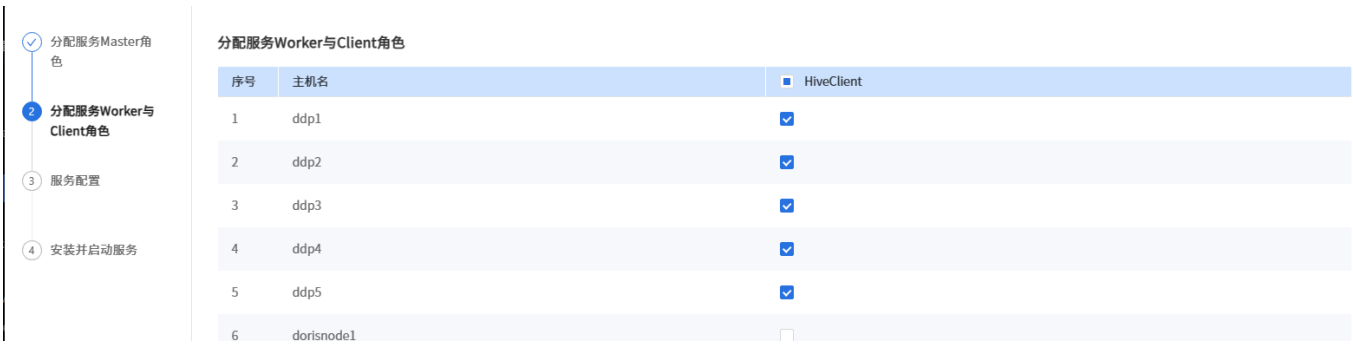

分配服务Master角色
2 分配服务Worker与Client角色
3 服务配置
4 安装并启动服务
分配服务Worker与Client角色
序号
主机名
DataNode
HdfsClient
1 ddp1
2 ddp2
3 ddp3
4 ddp4
5 ddp5
6 dorisnode1

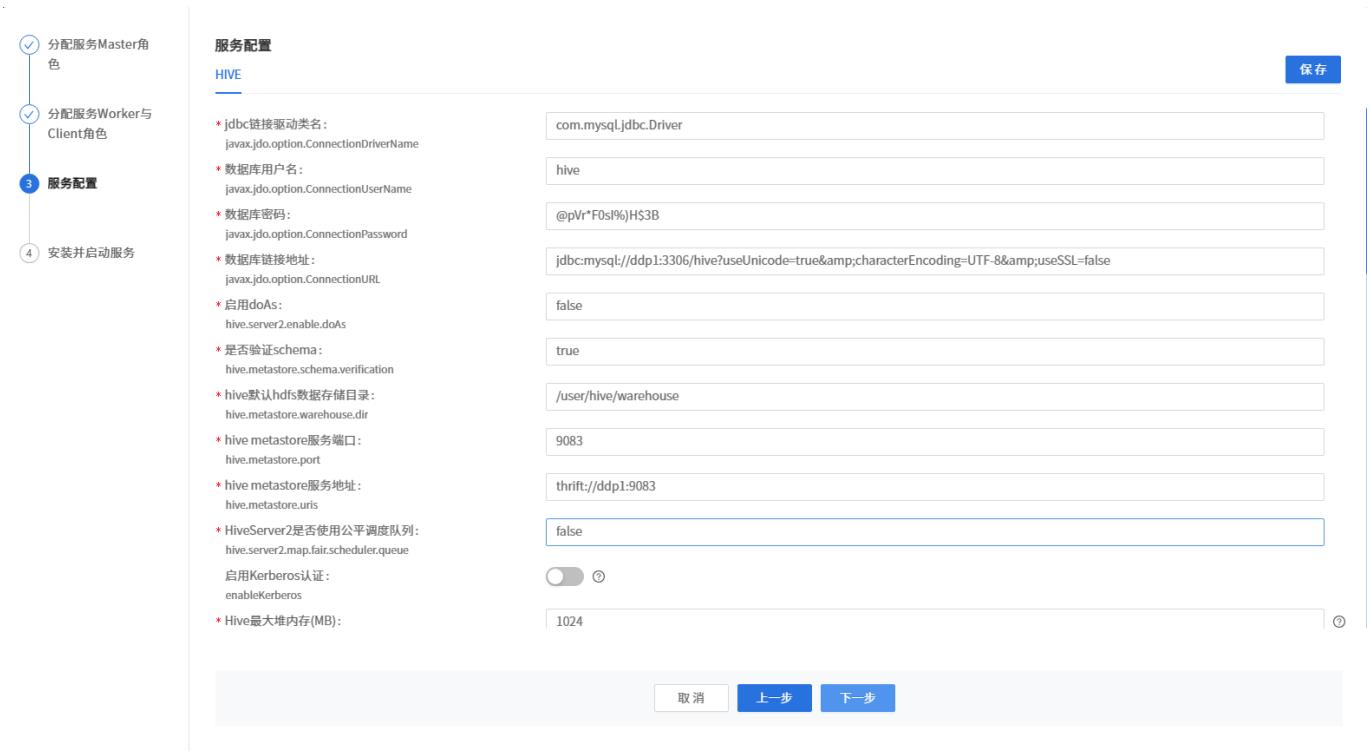

分配服务Master角色
分配服务Worker与Client角色
3 服务配置
4 安装并启动服务
服务配置
HDFS
保存
* NameNode地址：
fs.defaultFS
hdfs://nameservice1
* Hadoop运行时文件存储目录：
hadoop.tmp.dir
/data/tmp/hadoop
* HDFS网页登录使用静态用户名：
hadoop.http.staticuser.user
admin
* 允许通过代理访问的主机节点：
hadoop.proxyuser.hive.hosts
*
* 允许通过代理用户所属组：
hadoop.proxyuser.hive.groups
*
* 允许通过代理的用户：
hadoop.proxyuser.hive.users
*
* BLOCK副本数：
dfs.replication
3
* NameNode数据存储目录：
dfs.namenode.name.dir
/data/dfs/nn
+ Add field
* 元数据checkpoint目录：
dfs.namenode.checkpoint.dir
/data/dfs/snn
* Block大小：
dfs.blocksize
268435456
NameNode处理线程池大小：
取消
上一步
下一步

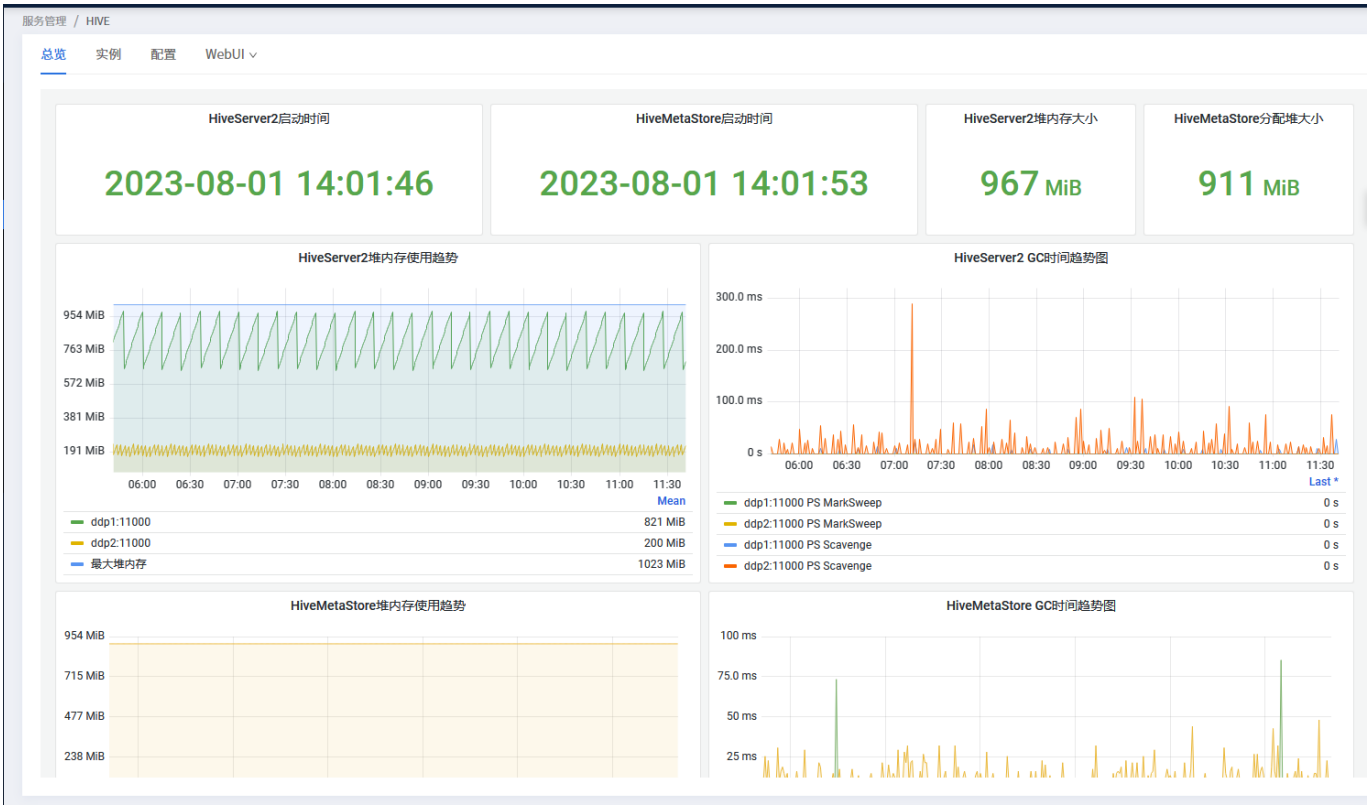

服务管理 / HDFS
总览
实例
配置
WebUI
NameNode启动时间
2023-08-01 09:37:38
NameNode运行时长
11.0 week
HDFS容量
3.89 TiB
在线DataNode
5
离线DataNode
0
NameNode堆内存大小
7.99 GiB
HDFS文件总数
284949
HDFS已用量
1.14 GiB
非HDFS使用存储
78.3 GiB
等待备份的block个数
0
心跳超时数量
0
NameNode堆内存使用率
ddp1:27001
35.5%
ddp2:27001
44.8%
等待删除block个数
0
block总数
138
DataNode堆内存大小
8.00 GiB
DataNode存储使用率
DataNode
存储使用率
ddp1:27002 0.0286%
ddp4:27002 0.0286%
ddp2:27002 0.0287%
ddp3:27002 0.0287%
ddp5:27002 0.0288%
DataNode堆内存使用率
DataNode
堆内存使用率
ddp5:27002 29.3%
ddp3:27002 5.33%
ddp4:27002 12.6%
ddp1:27002 8.43%
ddp2:27002 1.61%

4.6 添加Hive

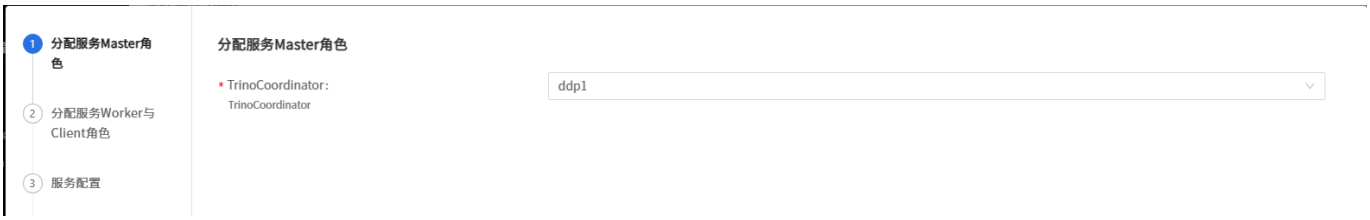

1 分配服务Master角色
2 分配服务Worker与Client角色
3 服务配置
分配服务Master角色
* HiveServer2:
HiveServer2
ddp1 × ddp2 ×
* HiveMetaStore:
HiveMetaStore
ddp1 ×

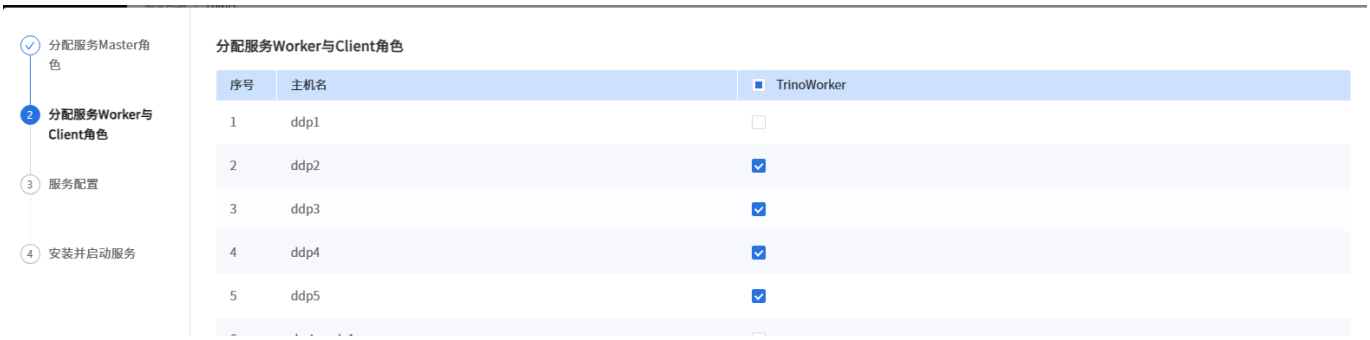

分配服务Master角色
2 分配服务Worker与Client角色
3 服务配置
4 安装并启动服务
分配服务Worker与Client角色
序号
主机名
HiveClient
1 ddp1
2 ddp2
3 ddp3
4 ddp4
5 ddp5
6 dorisnode1

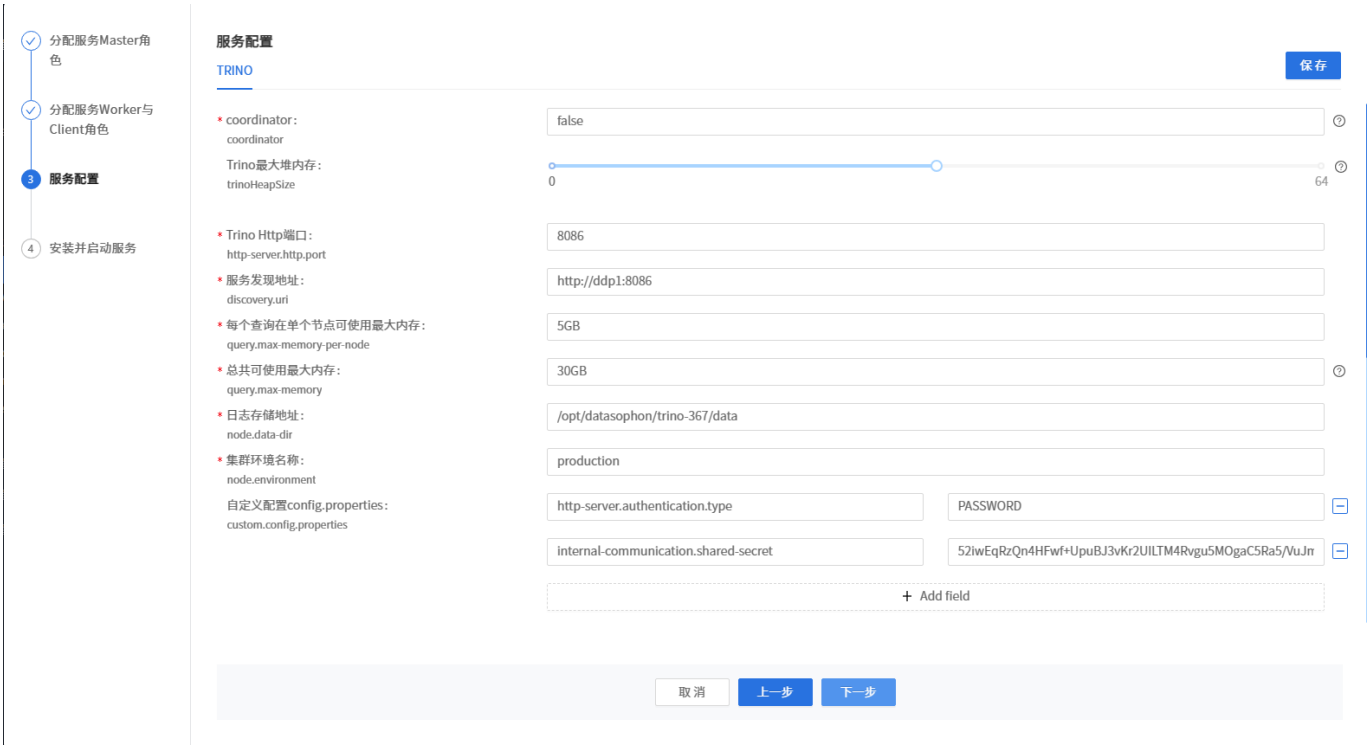

分配服务Master角色
分配服务Worker与Client角色
3 服务配置
4 安装并启动服务
服务配置
HIVE
保存
* jdbc链接驱动类名:
javax.jdo.option.ConnectionDriverName
com.mysql.jdbc.Driver
* 数据库用户名:
javax.jdo.option.ConnectionUserName
hive
* 数据库密码:
javax.jdo.option.ConnectionPassword
@pVr*F0sl%)H$3B
* 数据库链接地址:
javax.jdo.option.ConnectionURL
jdbc:mysql://ddp1:3306/hive?useUnicode=true&characterEncoding=UTF-8&useSSL=false
* 启用doAs:
hive.server2.enable.doAs
false
* 是否验证schema:
hive.metastore.schema.verification
true
* hive默认hdfs数据存储目录:
hive.metastore.warehouse.dir
/user/hive/warehouse
* hive metastore服务端口:
hive.metastore.port
9083
* hive metastore服务地址:
hive.metastore.uris
thrift://ddp1:9083
* HiveServer2是否使用公平调度队列:
hive.server2.map.fair.scheduler.queue
false
启用Kerberos认证:
enableKerberos
* Hive最大堆内存(MB):
1024
取消
上一步
下一步


服务管理 / HIVE
总览
实例
配置
WebUI
HiveServer2启动时间
2023-08-01 14:01:46
HiveMetaStore启动时间
2023-08-01 14:01:53
HiveServer2堆内存大小
967 MiB
HiveMetaStore分配堆大小
911 MiB
HiveServer2堆内存使用趋势
954 MiB
763 MiB
572 MiB
381 MiB
191 MiB
06:00 06:30 07:00 07:30 08:00 08:30 09:00 09:30 10:00 10:30 11:00 11:30
Mean
ddp1:11000 821 MiB
ddp2:11000 200 MiB
最大堆内存 1023 MiB
HiveServer2 GC时间趋势图
300.0 ms
200.0 ms
100.0 ms
0 s
06:00 06:30 07:00 07:30 08:00 08:30 09:00 09:30 10:00 10:30 11:00 11:30
Last *
ddp1:11000 PS MarkSweep 0 s
ddp2:11000 PS MarkSweep 0 s
ddp1:11000 PS Scavenge 0 s
ddp2:11000 PS Scavenge 0 s
HiveMetaStore堆内存使用趋势
954 MiB
715 MiB
477 MiB
238 MiB
HiveMetaStore GC时间趋势图
100 ms
75.0 ms
50 ms
25 ms

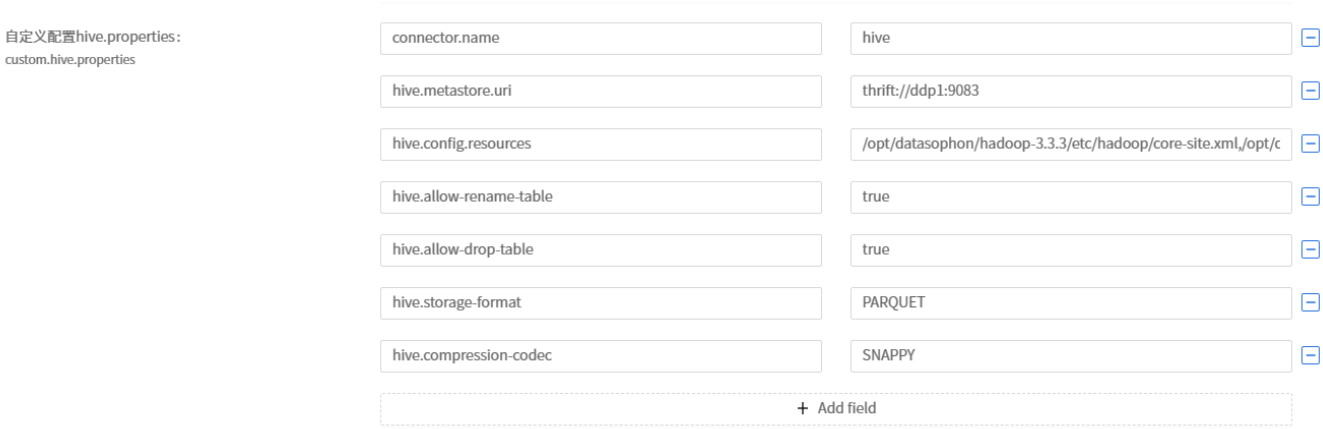
4.7 添加Trino

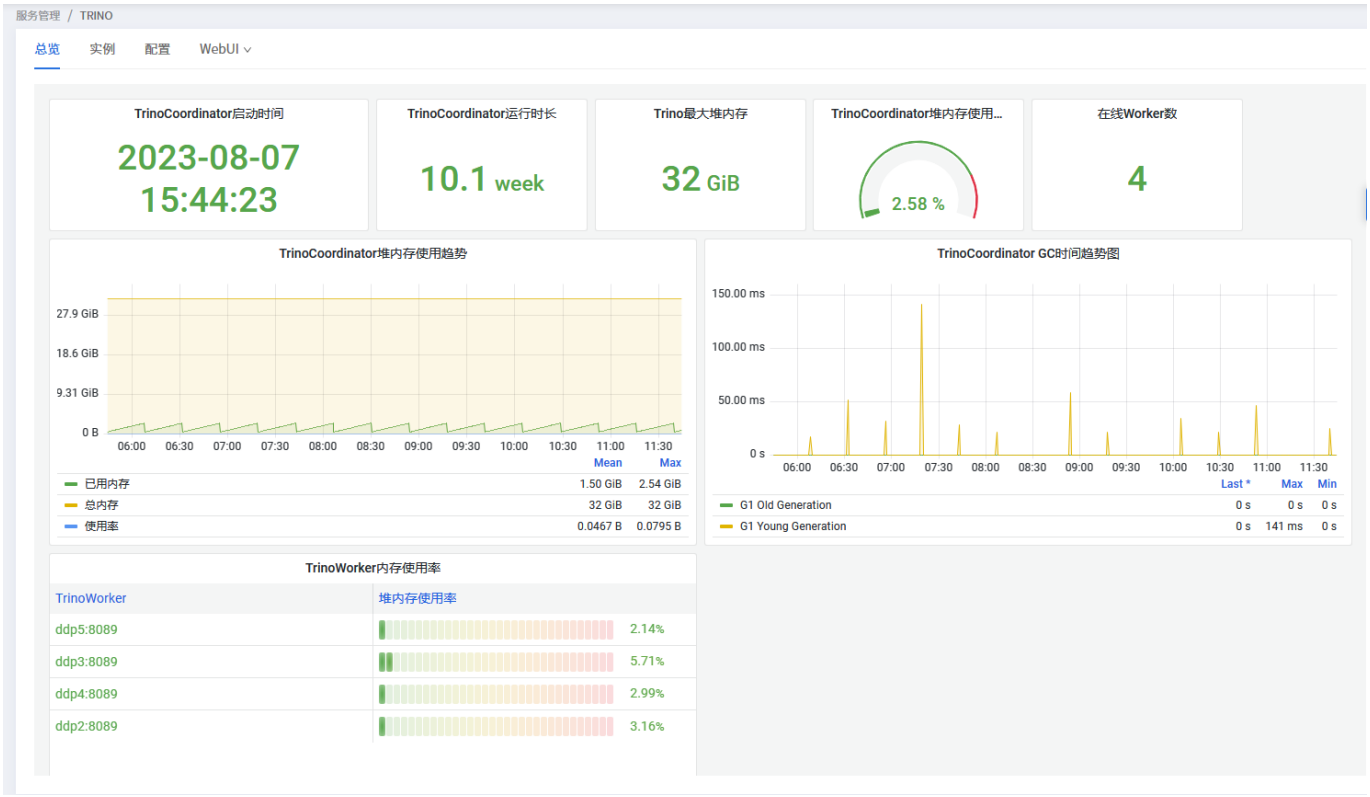

1 分配服务Master角色
2 分配服务Worker与Client角色
3 服务配置
分配服务Master角色
* TrinoCoordinator:
TrinoCoordinator
ddp1


分配服务Master角色
2 分配服务Worker与Client角色
3 服务配置
4 安装并启动服务
分配服务Worker与Client角色
序号
主机名
TrinoWorker
1 ddp1
2 ddp2
3 ddp3
4 ddp4
5 ddp5

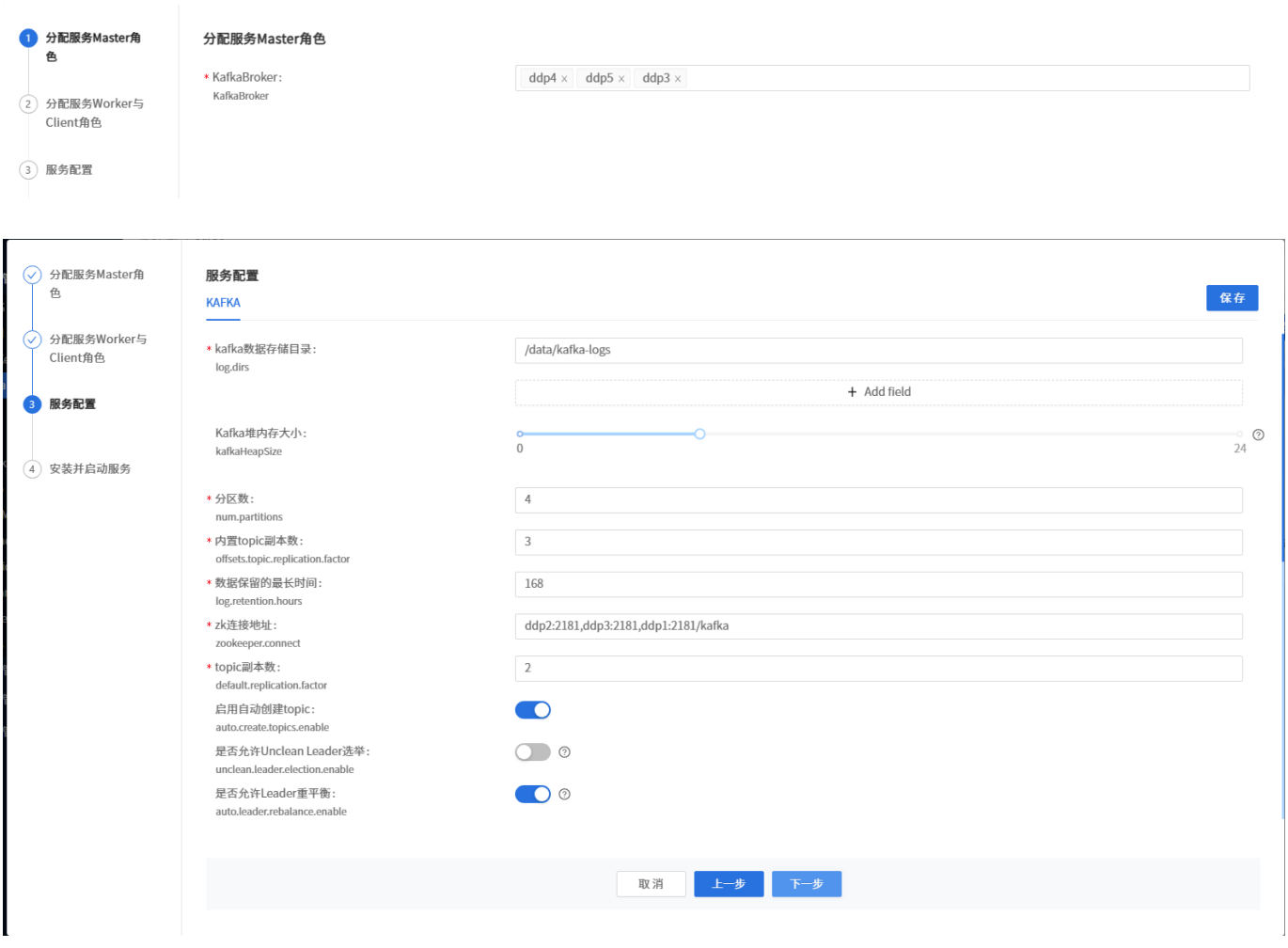


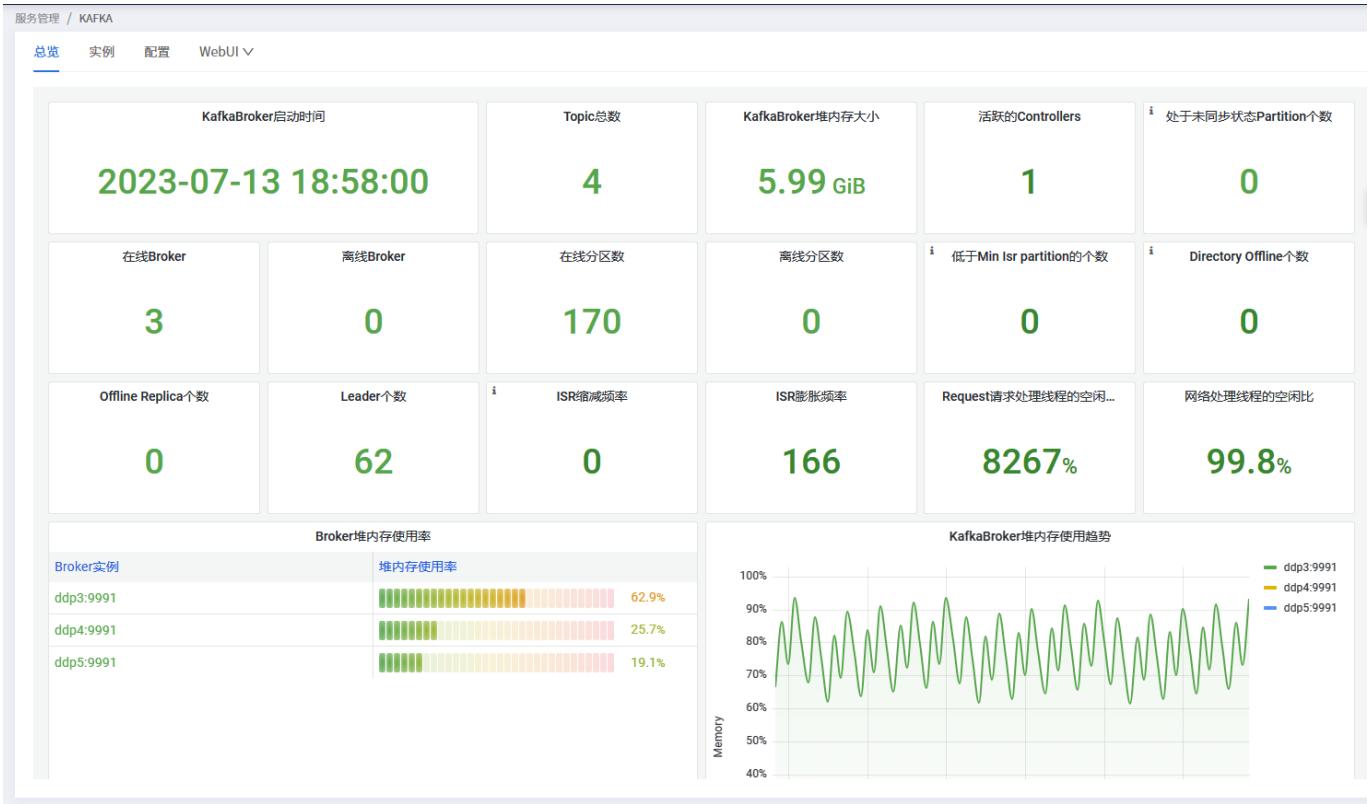
配置trino连接hive权限与格式等




服务管理 / TRINO
总览
实例
配置
WebUI
TrinoCoordinator启动时间
2023-08-07 15:44:23
TrinoCoordinator运行时长
10.1 week
Trino最大堆内存
32 GiB
TrinoCoordinator堆内存使用...
2.58 %
在线Worker数
4
TrinoCoordinator堆内存使用趋势
27.9 GiB
18.6 GiB
9.31 GiB
0 B
06:00 06:30 07:00 07:30 08:00 08:30 09:00 09:30 10:00 10:30 11:00 11:30
Mean Max
已用内存 1.50 GiB 2.54 GiB
总内存 32 GiB 32 GiB
使用率 0.0467 B 0.0795 B
TrinoCoordinator GC时间趋势图
150.00 ms
100.00 ms
50.00 ms
0 s
06:00 06:30 07:00 07:30 08:00 08:30 09:00 09:30 10:00 10:30 11:00 11:30
Last * Max Min
G1 Old Generation 0 s 0 s 0 s
G1 Young Generation 0 s 141 ms 0 s
TrinoWorker内存使用率
TrinoWorker
堆内存使用率
ddp5:8089 2.14%
ddp3:8089 5.71%
ddp4:8089 2.99%
ddp2:8089 3.16%

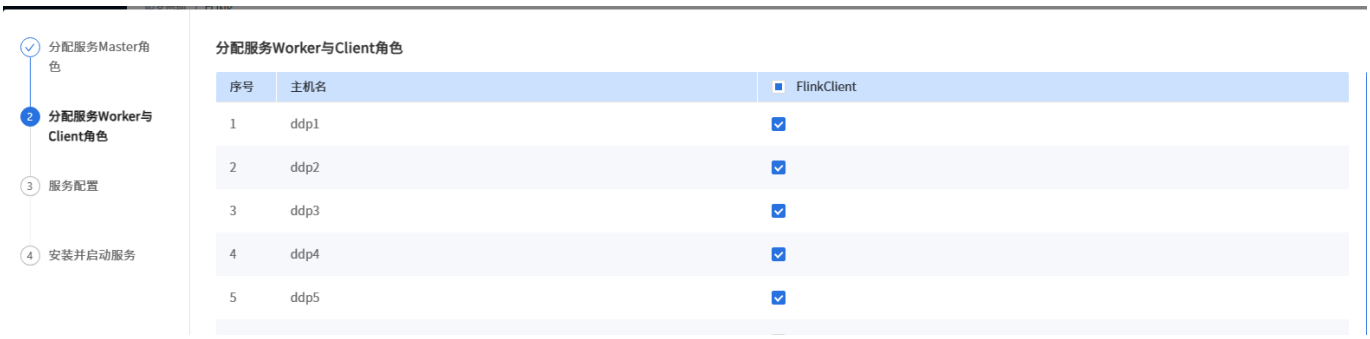
4.8 添加Kafka

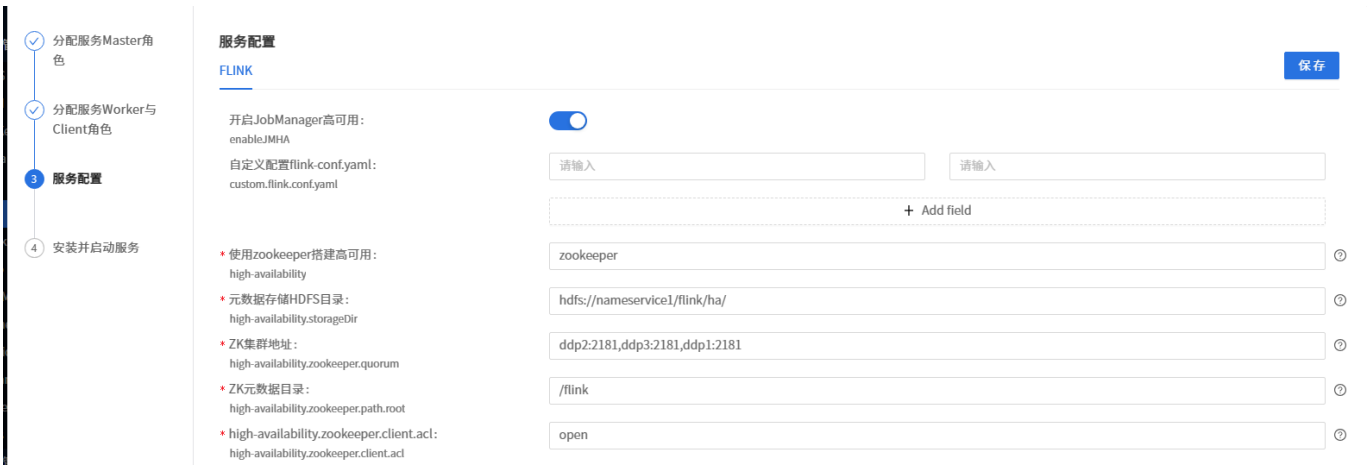

1 分配服务Master角色
2 分配服务Worker与Client角色
3 服务配置
分配服务Master角色
* KafkaBroker:
KafkaBroker
ddp4 x ddp5 x ddp3 x
分配服务Master角色
分配服务Worker与Client角色
3 服务配置
4 安装并启动服务
服务配置
KAFKA
保存
* kafka数据存储目录:
log.dirs
/data/kafka-logs
+ Add field
Kafka堆内存大小:
kafkaHeapSize
0
24
* 分区数:
num.partitions
4
* 内置topic副本数:
offsets.topic.replication.factor
3
* 数据保留的最长时间:
log.retention.hours
168
* zk连接地址:
zookeeper.connect
ddp2:2181,ddp3:2181,ddp1:2181/kafka
* topic副本数:
default.replication.factor
2
启用自动创建topic:
auto.create.topics.enable
是否允许Unclean Leader选举:
unclean.leader.election.enable
是否允许Leader重平衡:
auto.leader.rebalance.enable
取消
上一步
下一步



4.9 添加Flink

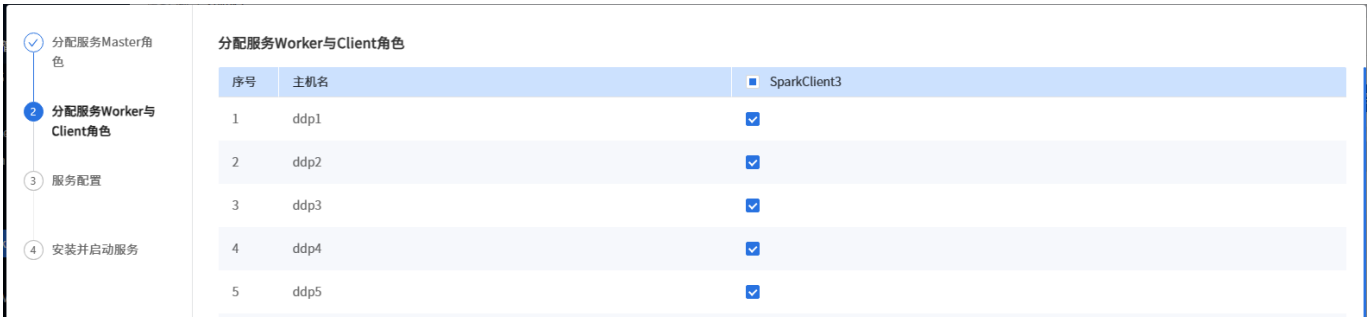
flink使用的是flinkClient部署方式，Application Mode方式，所以不提交任务没有状态

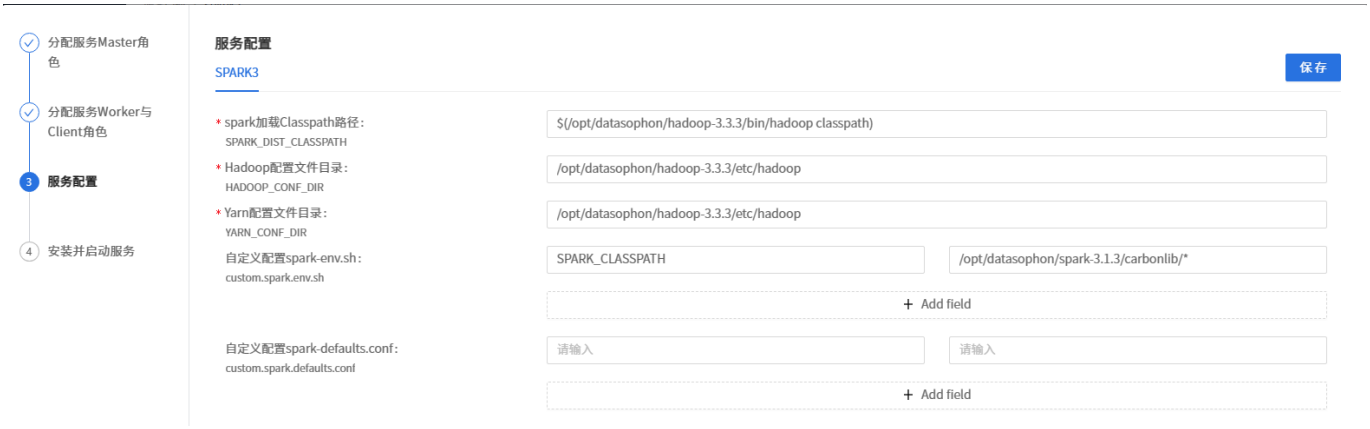




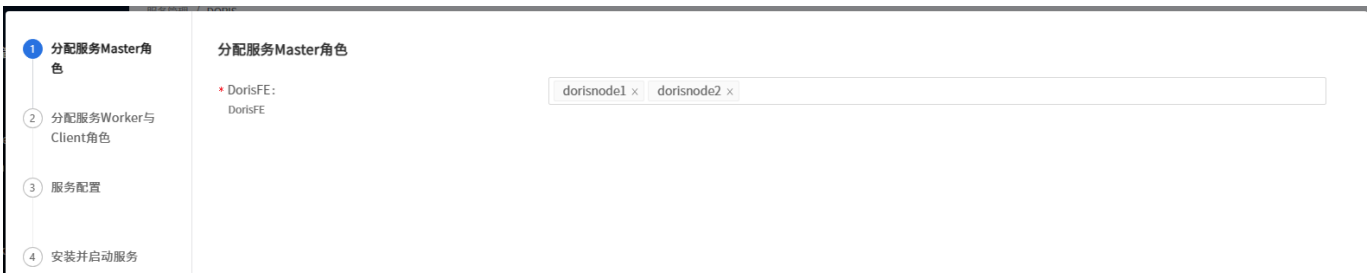
4.10 添加Spark3

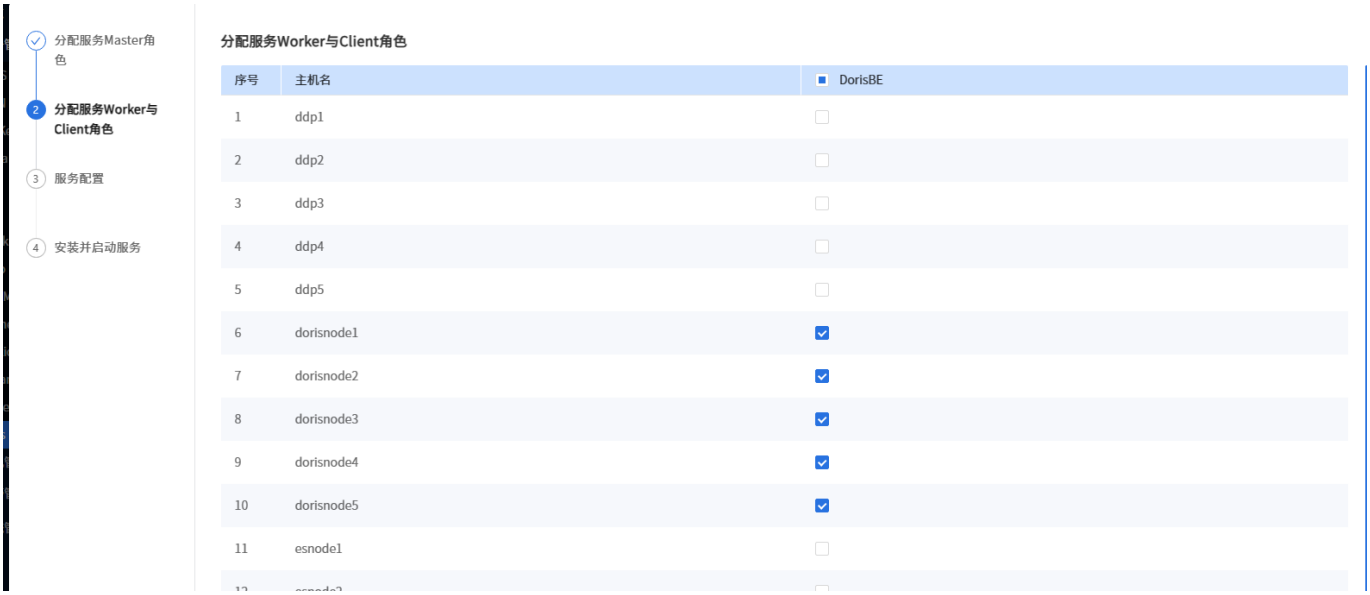
spark使用的是sparkClient部署方式，所以不提交任务没有状态

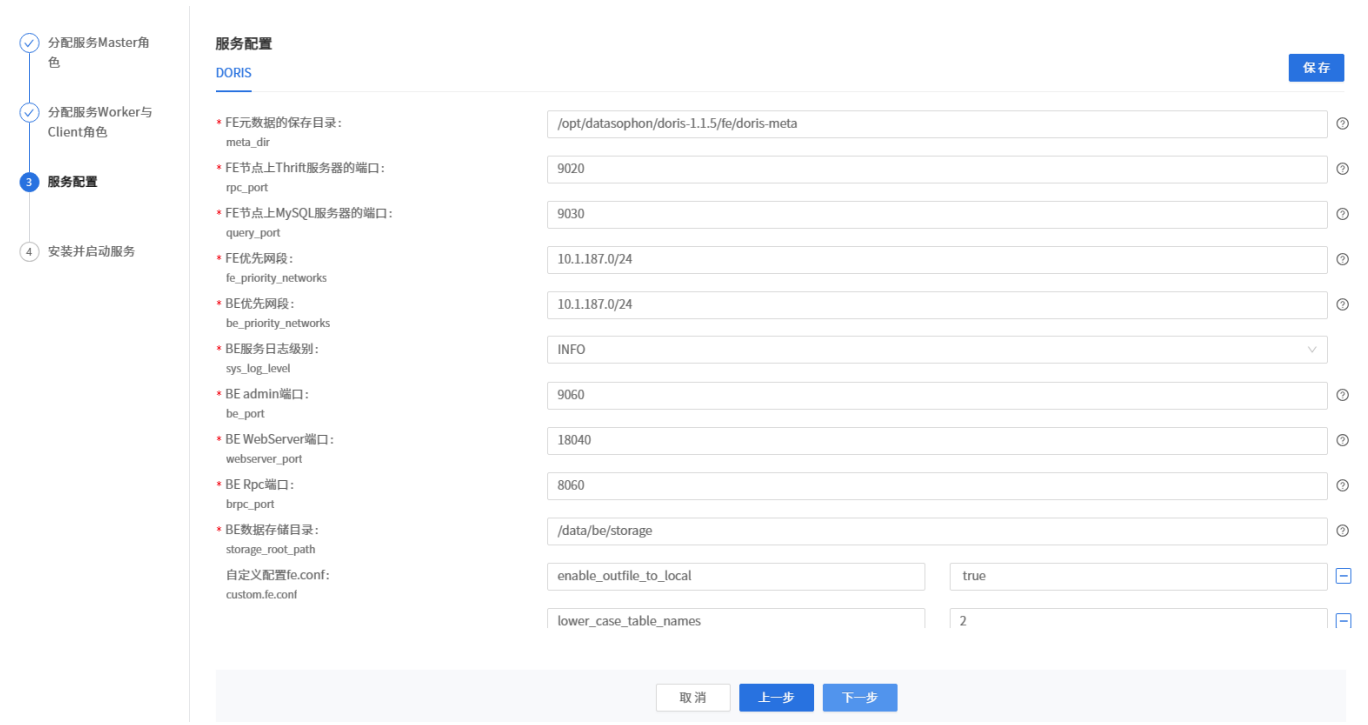


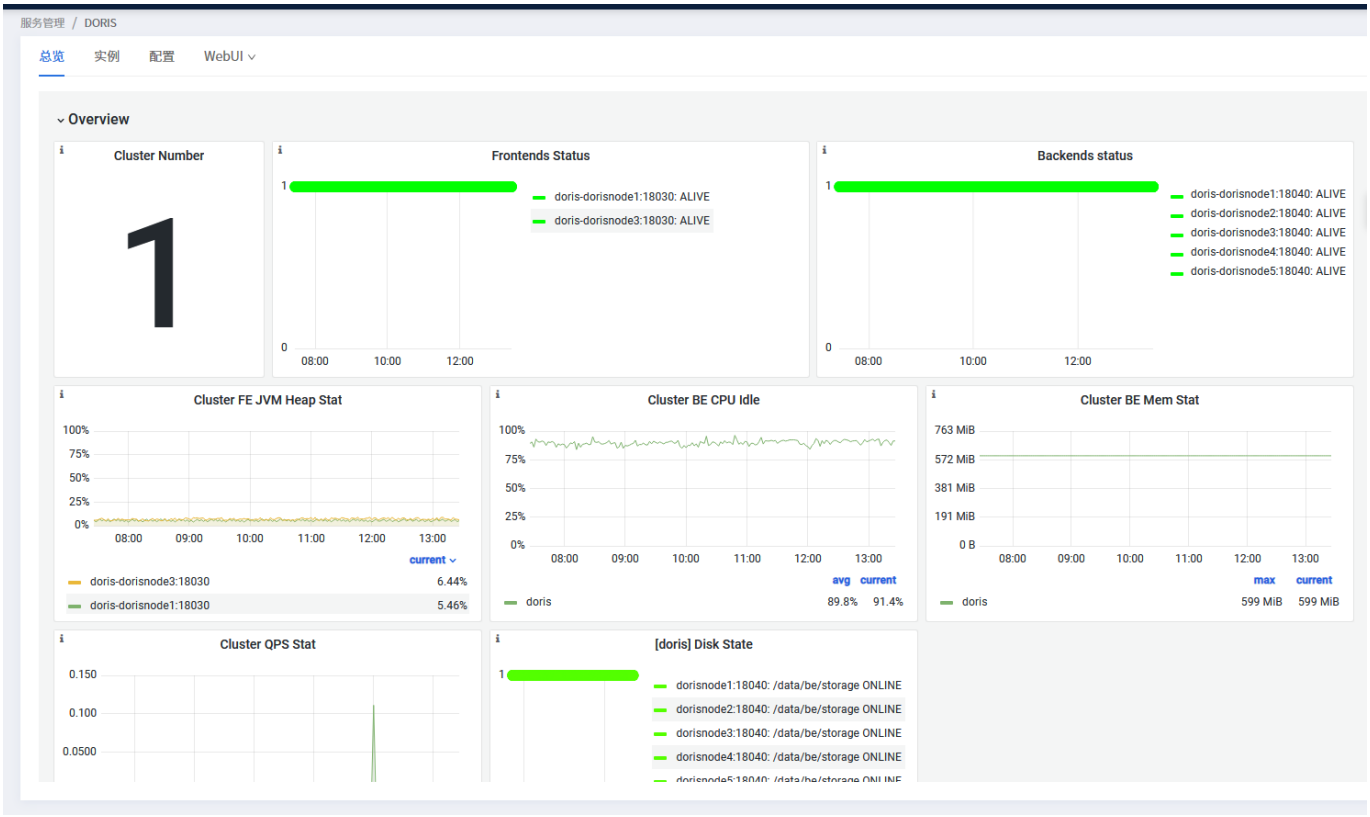


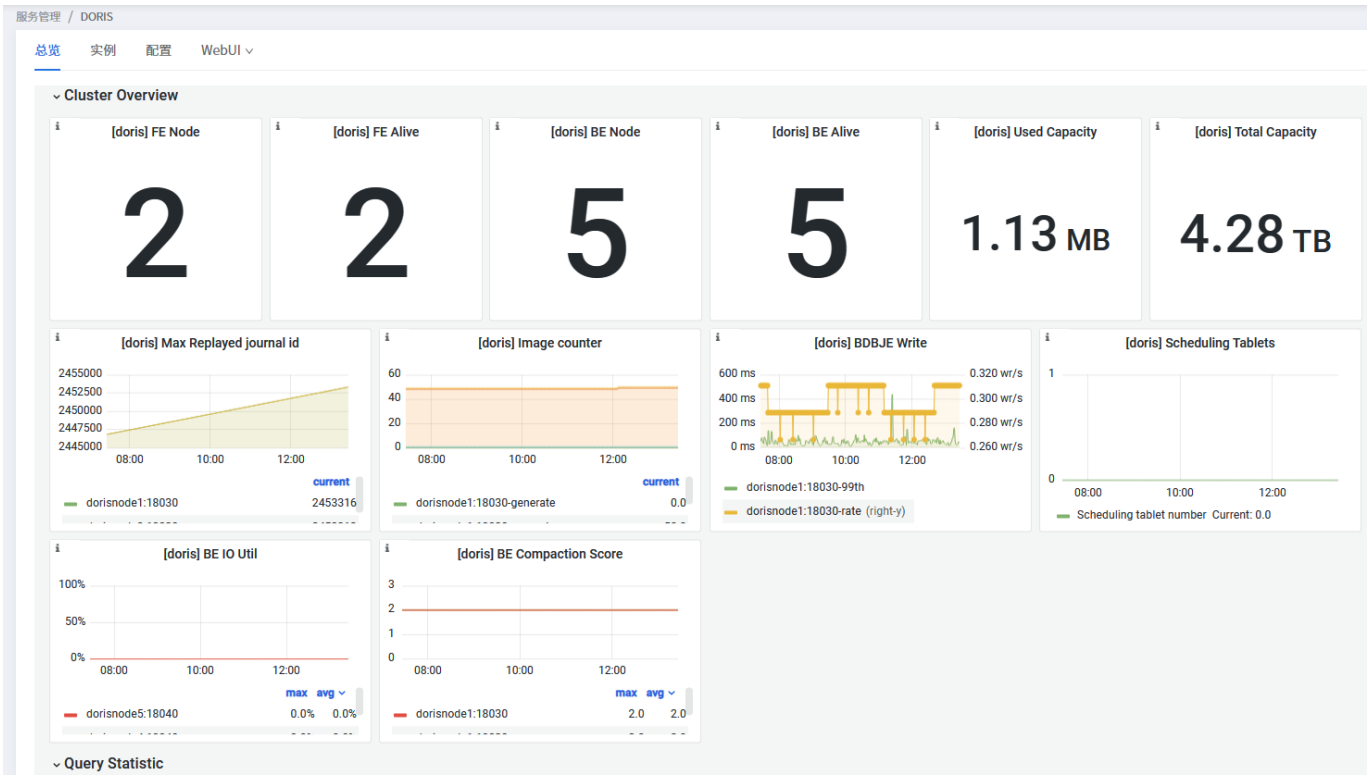
4.11 添加Doris



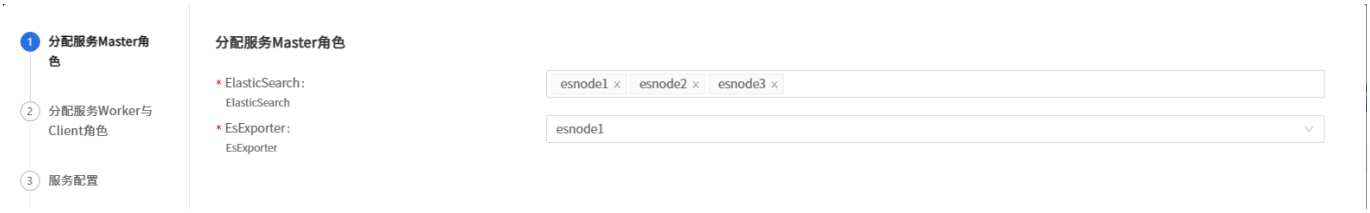

分配服务Master角色
2 分配服务Worker与Client角色
3 服务配置
4 安装并启动服务
分配服务Worker与Client角色
序号
主机名
DorisBE
1 ddp1
2 ddp2
3 ddp3
4 ddp4
5 ddp5
6 dorisnode1
7 dorisnode2
8 dorisnode3
9 dorisnode4
10 dorisnode5
11 esnode1

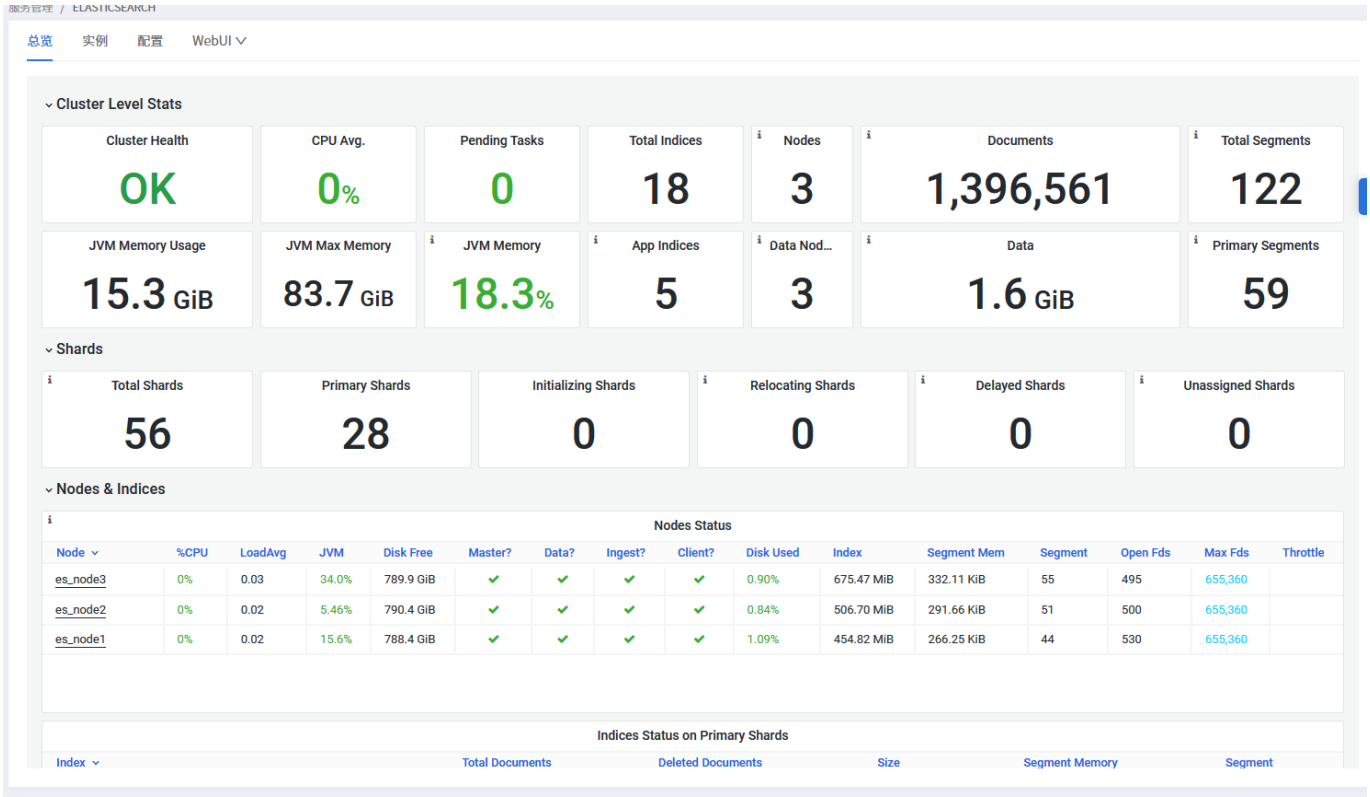

分配服务Master角色
分配服务Worker与Client角色
3 服务配置
4 安装并启动服务
服务配置
DORIS
保存
* FE元数据的保存目录: meta_dir
/opt/datasophon/doris-1.1.5/fe/doris-meta
* FE节点上Thrift服务器的端口: rpc_port
9020
* FE节点上MySQL服务器的端口: query_port
9030
* FE优先网段: fe_priority_networks
10.1.187.0/24
* BE优先网段: be_priority_networks
10.1.187.0/24
* BE服务日志级别: sys_log_level
INFO
* BE admin端口: be_port
9060
* BE WebServer端口: webserver_port
18040
* BE Rpc端口: brpc_port
8060
* BE数据存储目录: storage_root_path
/data/be/storage
自定义配置fe.conf: custom.fe.conf
enable_outfile_to_local
true
lower_case_table_names
2
取消
上一步
下一步





4.12 添加Elasticsearch





5.记录注意点

1. HADOOP的NameNode和ZKFC需要做HA

2. YARN的Resourcemanager也需要做HA

3. spark可以执行以下命令去启动thriftserver，使用spark的jar包可以直接使用jdbc连接到spark，实际上就是类似HiveServer2，需要注意的是，默认使用的是10000端口号，与hive相同，需要注意别端口冲突 因为权限问题，所以需要授权并使用hive用户启动

```
# 此处图省事，直接给了777权限，也可以给hive用户授权
chmod -R 777 /opt/datasophon/spark-3.1.3/pid/
chmod -R 777 /opt/datasophon/spark-3.1.3/logs/
su - hive -c /opt/datasophon/spark-3.1.3/sbin/start-thriftserver.sh
```

4. Hive需要修改JDBC数据库链接地址，并设置以下自定义配置hive-site.xml，默认的执行器是Tez而不是Yarn

```
#hive的执行引擎修改为yarn
mapreduce.framework.name    yarn
#如果hive的程序，只有maptask，将MapTask产生的所有小文件进行合并
hive.merge.mapfiles     true
#如果hive的程序，有Map和ReduceTask,将ReduceTask产生的所有小文件进行合并
hive.merge.mapredfiles      true
#每一个合并的文件的大小为256M
hive.merge.size.per.task    268435456
#平均每个文件的大小，如果小于这个值就会进行合并
hive.merge.smallfiles.avgsize   104857600
#hive的zookeeper的地址
hive.zookeeper.quorum   bigwork01,bigwork02,bigmaster
#hive的zookeeper的端口
hive.zookeeper.client.port  2181
#hive的zk的session超时时间
hive.zookeeper.session.timeout  600000
#hive的zk的节点名称
hive.server2.zookeeper.namespace  hiveserver2
#关闭在非local模式下,hive是否要在独立的jvm中执行，与jmx监控冲突
hive.exec.submit.local.task.via.child false
#关闭hive的自动收集统计信息，很重要！！！，不加不能对同一个表重复insert
hive.stats.autogather false
```

5. Grafana需要手动设置数据库，登录Grafana地址

```
http://ddp1:3000/login
账号密码admin，点击Configuration，设置mysql数据源
```

6. Trino部署需要python环境，默认是使用python命令，如果没有则可以创建软连接，且需要增加环境变量

```
ln -s /usr/bin/python3 /usr/bin/python
并且需要修改/opt/datasophon/trino-367/bin/launcher.py的print 'coordinator true'改为
print('coordinator true')
```

如果启动后出现GM+8时区问题，需要修改启动脚本，增加时区

```
vim /opt/datasophon/trino-367/bin/launcher
PATH前增加
export TZ='Asia/Shanghai'
```

trino默认没有删除hive表的权限，需要增加配置hive官网配置(https://trino.io/docs/current/connector/hive.html)

```
#允许删除表
hive.allow-rename-table=true
hive.allow-drop-table=true
#默认的存储格式
hive.storage-format=PARQUET
hive.compression-codec=SNAPPY
```

7. 开启Kerberos认证后，如果碰到hdfs启动成功，kinit也认证成功，但是执行hadoop命令还是出现认证错误可能是需要修改cache默认方式，默认是KCM:，修改为文件缓存，缓存在tmp文件夹中，注意长时间不用会被清掉

```
vim /etc/krb5.conf.d/kcm_default_ccache
#修改
default_ccache_name = FILE:/tmp/krb5cc_%{uid}
scp /etc/krb5.conf.d/kcm_default_ccache root@bigwork01:/etc/krb5.conf.d/
scp /etc/krb5.conf.d/kcm_default_ccache root@bigwork02:/etc/krb5.conf.d/
```

或者直接注释掉/etc/krb5.conf第一行includedir /etc/krb5.conf.d/

开启后会自动创建管理账号，admin/admin@HADOOP.COM,密码为admin

可以使用命令创建自己的用户

```
kinit admin/admin@HADOOP.COM
kadmin.local
addprinc hdfs/bigmaster@HADOOP.COM
ktadd -k  /etc/security/keytabs/hdfs.keytab -norandkey hdfs/bigmaster@HADOOP.COM
```

kafka需要修改下kafka-run-class.sh，里面关于Kerberos的配置未增加，需要手动添加(fix 1.1.2)

```
vim /opt/datasophon/kafka-2.4.1/bin
KAFKA_JVM_PERFORMANCE_OPTS="-server  -XX:MaxGCPauseMillis=20 -
XX:InitiatingHeapOccupancyPercent=35 -XX:+ExplicitGCInvokesConcurrent -
Djava.awt.headless=true"
改为
KAFKA_JVM_PERFORMANCE_OPTS="-server  -XX:MaxGCPauseMillis=20 -
XX:InitiatingHeapOccupancyPercent=35 -XX:+ExplicitGCInvokesConcurrent -
Djava.awt.headless=true -Djava.security.auth.login.config=/opt/datasophon/kafka-
2.4.1/config/kafka-jaas.conf -Djava.security.krb5.conf=/etc/krb5.conf"
```

PS：使用注意

1. 查看启动日志方式，点击命令，比如“ 安装 ZOOKEEPER”，点击主机即可查看日志。
2. 安装服务时可能失败或者部分失败，尤其是安装zookeeper，可能需要点击添加新实例，将失败的实例重新安装，但是会创建一个新的角色组，需要将zk统一为同一个角色组，如果zk启动有问题，可以删除有问题服务器的/data/log/version-2与/data/zookeeper/version-2中的文件，然后重新启动
3. 安装时会有各种的Client选项，安装的服务器可以直接使用命令
4. 配置会有版本号，如果想反悔直接修改为之前正确版本配置重启即可
5. 如果主机管理或添加服务时选不到服务器，可以到对应服务器重启下work