托管Hadoop

产品文档

文档目录

产品简介
- 产品概述
- 产品功能
- 产品优势
- 全面的服务可用性保障
- 弹性伸缩，灵活计费
- 多租户支持
- 支持更多的组件
- 应用场景
- 离线数据处理
- 数据查询和报表
- 流式数据处理
- NoSQL数据库
- 应用领域
  - 金融领域
  - 电商领域

产品架构

快速入门
- 步骤一：进入购买入口
- 步骤二：创建托管HADOOP集群
- 步骤三：登录集群主节点

最佳实践
- VPC网络配置
- 托管HADOOP网络架构
- 通过托管HADOOP集群访问非托管HADOOP集群云主机
- 通过非托管HADOOP集群云主机访问托管HADOOP集群
- 开通非master1节点的外网访问权限
- MapReduce
- 准备
- MapReduce作业
- 程序准备
- 作业输入输出
- 作业提交
- 准备
- Spark 离线任务处理

入门操作
场景一：数据储存在本地
SparkSQL 的使用及操作
数据基本操作
场景一：结构化数据处理
场景二：非结构化数据处理
场景三：多源数据整合处理
Spark SQL 命令行操作
Spark 在流式处理中的应用
Kafka 和 Spark 来实现流式处理
HUE使用实践指南
HUE 总体概览
HUE简介
相关连接
HUE功能
登录HUE
HDFS文件浏览
文件查看
新建文件夹
文件
SQL查询
Hive操作
新建数据库
新建表
查询
结果可视化
其他
修改Hue中“Query Editors”配置
Markdown功能使用说明
Markdown简介
Markdown语法简介
Notebook操作
MapReduce作业
新建Mapreduce Action
作业配置
DistCp作业
Kibana使用指南
Kibana简介

进入Kibana
添加索引
配置索引
Discover/数据查询
字段过滤
可视化/Visualize
仪表盘分析/Dashboard
ES基于时间索引
索引名中增加日期后缀
定时删除过期的文件
运维操作
告警处理
故障处理
常见问题
产品介绍问题
产品使用问题

产品简介

产品概述

最近更新时间: 2021-09-15 16:14:36

大数据是关于收集，储存，处理和展现大规模数据的技术，它可以帮助企业从这些数据中提取知识，并且做出更好的商业决策，所有的这些工作必须在有限的时间内尽快完成。大数据处理的一个最主要挑战就是数据分析平台的管理，包括了安装和操作管理，对于多种工作负载动态的分配数据处理能力，以及从多个来源收集数据进行整体分析。

托管Hadoop提供了强大的扩展能力和弹性伸缩能力，消除了Hadoop安装部署成本和管理复杂性，可以使您不必关注基础架构管理，而更加专注数据分析处理本身，任何的开发者或者公司只需要较低的成本就可以进行大规模的数据分析和处理工作。

产品功能

最近更新时间: 2021-09-15 16:14:36

托管Hadoop提供了丰富的管理功能和便捷的程序开发接口，使您可以高效自动化的进行数据处理和分析工作，节省管理成本和使用成本：

弹性扩展

托管Hadoop集群具备良好的横向扩展能力，您可以根据业务需求弹性的增加或者减少节点，适应多变的业务场景，节省集群使用成本。

集群主节点和元数据高可用

除了基础的服务可用性和数据可靠性保障外，托管Hadoop提供了主节点和元数据高可用功能来进一步保证集群持久对外提供服务。采用两个主节点作为集群管理节点，担当NameNode、ResourceManager、HbaesMaster等角色，当节点宕机时，监控系统会自动发现，由另一节点接管服务，并自动启动新的主节点使集群恢复到稳定状态。

Hadoop生态集成

托管Hadoop除集成了基础的Hadoop组件外，同时集成了Spark， Hbase， Flink， Kafka， Elasticsearch等生态组件，帮助您轻松构建复杂的大数据分析系统，满足批量计算、流式处理、消息队列、交互式查询、NoSQL等多种业务场景的需求。

计算作业管理

托管Hadoop可以支持MapReduce， Spark， Hive等多种计算作业，这些计算可通过可视化管理组件进行管理。

产品优势

最近更新时间: 2021-09-15 16:14:36

快速部署，完全托管

传统的Hadoop平台部署，通常需要经历业务评估、机器选型采购、硬件上架调试、操作系统和平台软件安装调试等一系列复杂的工作，这些工作需要花费1-3个月的时间。托管HADOOP高度集成，快速部署，通常情况下只需几分钟即可自动完成部署工作，您只需关心数据处理任务本身，无需关注硬件和底层系统的运维工作。

全面的服务可用性保障

最近更新时间: 2021-09-15 16:14:36

托管Hadoop在服务控制端，集群主节点，元数据管理，节点反亲近策略，硬件，监控告警，等多个层面提供了服务可用性和数据安全保障，大幅提升流式计算，Hbase数据读写，既时数据查询等场景的业务连续性。

弹性伸缩，灵活计费

最近更新时间: 2021-09-15 16:14:36

不同于传统业务系统，数据分析系统很难做到精确的容量规划，不同的任务使用的资源类型也不尽相同，托管Hadoop提供了多种节点配置，并且可以弹性的增加或者减少节点，能够减少您的使用成本，轻松应对多变的业务需求。

多租户支持

最近更新时间: 2021-09-15 16:14:32

相比于开源YARN、ambari等，托管Hadoop支持一个集群下多个租户的共享使用。可灵活配置每个租户可使用的计算和存储资源，通过用户的的角色，权限管理灵活控制用户对数据的访问权限。通过对用户的最大最小资源的使用限制，可实现弹性计算，提高资源的利用率。

支持更多的组件

最近更新时间: 2021-09-15 16:14:32

托管Hadoop支持ldap,kdc，flink等大数据平台基础组件的支持，用户可在托管Hadoop管理界面自由向集群中添加和删除这些组件。

开源组件增强和bug修复

修复hive默认时区配置默认为东八区

修改ranger用户同步时长，实现用户快速同步

HDFS等组件对对象存储的支持

Yarn On Docker的支持

应用场景

最近更新时间: 2019-11-26 15:30:11

离线数据处理

最近更新时间: 2021-09-15 16:14:32

离线数据处理是最常见的Hadoop应用场景，您可将原始数据上传到托管HADOOP或者集群HDFS文件系统中，通过控制台或者API来执行批量的离线处理作业。

数据查询和报表

最近更新时间: 2021-09-15 16:14:32

托管Hadoop提供了Hive和SparkSQL等类SQL查询方案，用户可使用简单直观的查询方法对海量的数据进行分析或者使用主流的BI工具生成报表。

流式数据处理

最近更新时间: 2021-09-15 16:14:32

流式数据处理逐渐成为大数据的热点，例如网站流量统计或游戏在线玩家数据，需要在不同粒度上对不同数据进行统计，既有实时性的需求，又需要涉及到聚合、去重、连接等较为复杂的统计需求，托管HADOOP提供了分布式消息队列Kafka，流式数据处理框架Flink以及Spark Streaming，帮助您轻松应对实时的数据处理需求。

NoSQL数据库

最近更新时间: 2021-09-15 16:14:32

互联网应用的典型特征是数据量大，高并发，业务增长快，托管HADOOP集成的Hbase是一种非常流行的分布式可扩展列存数据库，可以充分满足各种在线应用需求，同时又可以和其他大数据生态组件结合，形成端到端的方案。

应用领域

金融领域

最近更新时间: 2021-09-15 16:14:32

大数据时代数据价值不断被凸显，天然拥有大量数据的金融行业需要利用好自身资源，挖掘数据价值，优化客户体验，增强自身竞争力。同时，完善自身的数据管理能力，加强数据规范标准化建设，加强应用创新能力。在大数据时代实现以数字化为中心的转型升级。

托管HADOOP可以从多个角度提供支持：

构建PB级大数据平台，统一指标，查询方便，快速获得商业知识和决策支持

完善的数据管理能力，数据的全流程把控，减少数据重复性建设

租户管理，权限控制，保障数据的安全

电商领域

最近更新时间: 2021-09-15 16:14:32

电商行业正在面临新的挑战，网上购物，线下实体店铺和移动购物等等。用户选项不断增多，购物场景多元丰富。伴随着的是数据量的井喷式增长，如何利用大量数据为客户提供独特的购物体验，满足客户个性化的需求。是每一个零售行业参与者面对的共同问题。

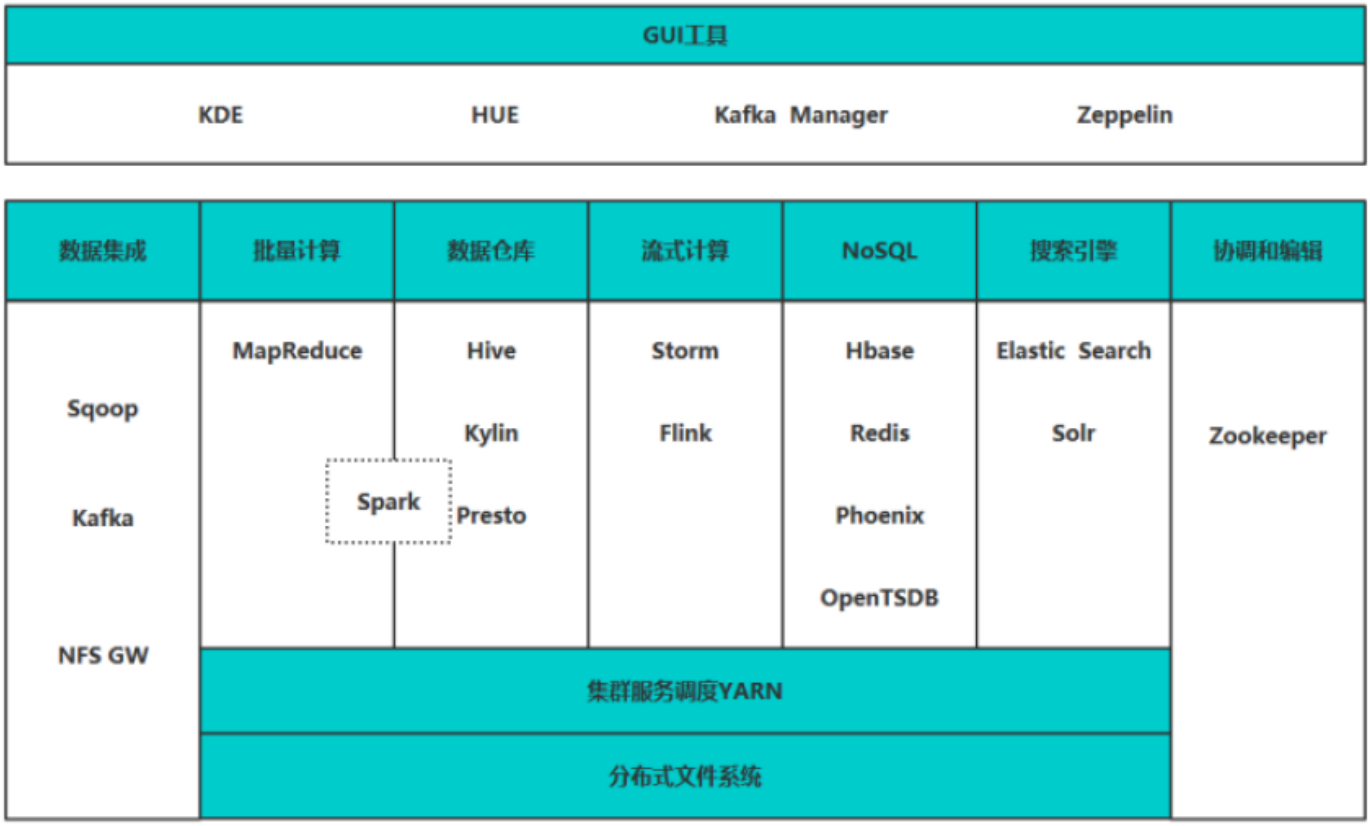
托管HADOOP可以从多个角度提供支持：

实现超大规模数据的储存，满足零售行业存量和增量数据的存储需求 历史明细数据查询，大批量数据分析，从数据中挖掘价值，优化购物流程

实时数据的查询处理，及时推送用户商品，提高用户购物体验

产品架构

最近更新时间: 2021-09-15 16:14:32



托管HADOOP对开源组件进行封装和增强，包含Manager和众多组件，分别提供功能如下：

HDFS

Hadoop分布式文件系统（Hadoop Distributed File System），提供高吞吐量的数据访问，适合大规模数据集方面的应用。

Mapreduce

提供快速并行处理大量数据的能力，是一种分布式数据处理模式和执行环境。

Yarn

资源管理系统，它是一个通用的资源模块，可以为各类应用程序进行资源管理和调度。

Hive

建立在Hadoop基础上的开源的数据仓库，提供类似SQL的Hive Query Language语言操作结构化数据存储服务和基本的数据分析服务。

HBase

提供海量数据存储功能，是一种构建在HDFS之上的分布式、面向列的存储系统。

Kafka

一个分布式的、分区的、多副本的实时消息发布和订阅系统。提供可扩展、高吞吐、低延迟、高可靠的消息分发服

务。

Spark

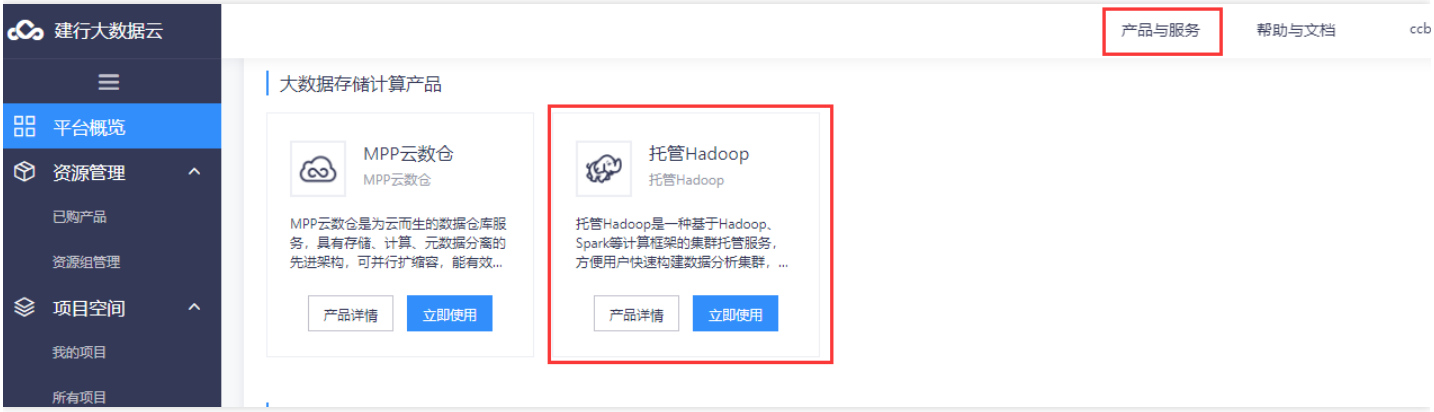
基于内存进行计算的分布式计算框架。

快速入门

步骤一：进入购买入口

最近更新时间: 2019-11-13 02:43:20

1. 注册账号，并开通托管Hadoop服务
2. 登录控制台，选择托管Hadoop

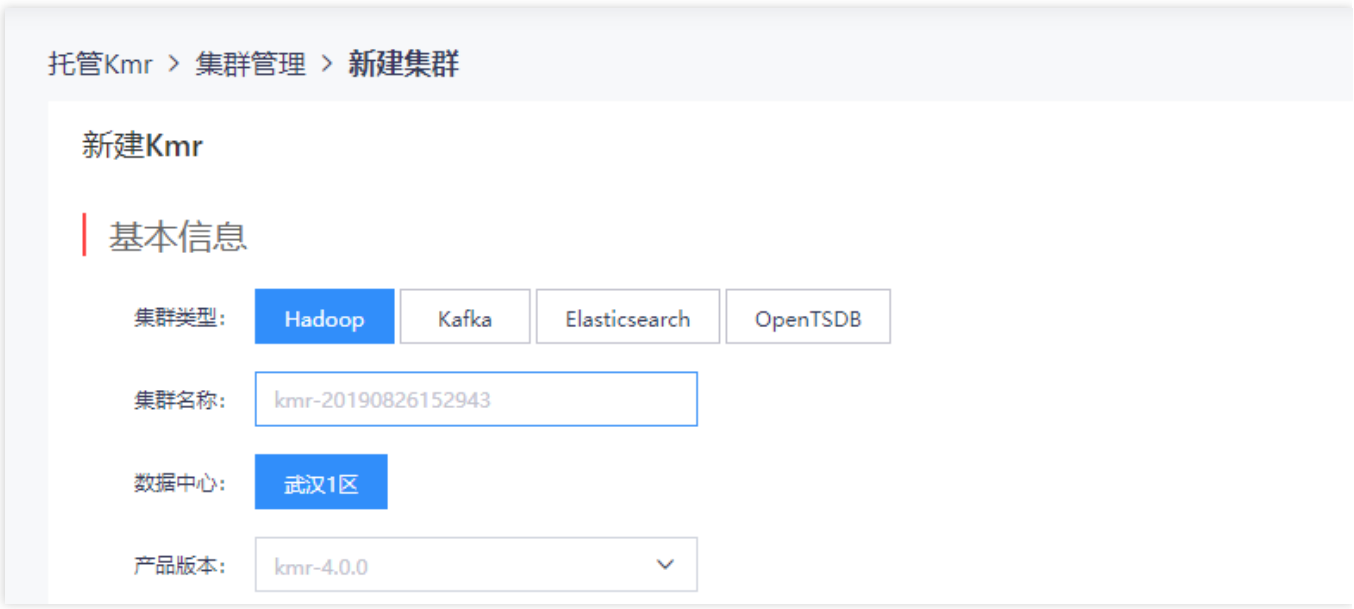


步骤二：创建托管HADOOP集群

最近更新时间: 2021-09-15 16:14:32

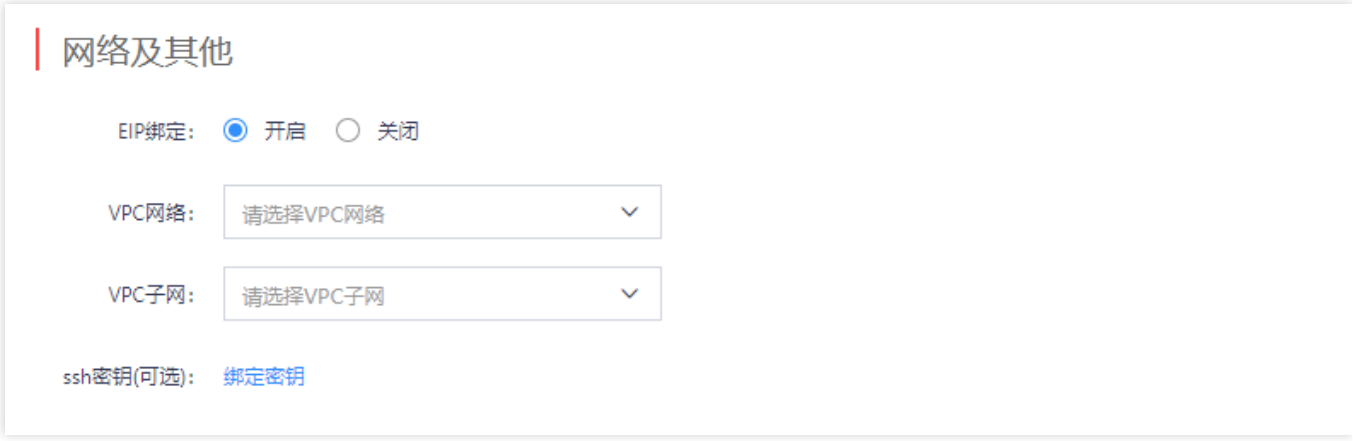
- 选择“集群管理”,点击“新建集群”按钮，进入集群创建向导

- “基本信息”模块选择，数据中心和创建版本类型。



- “节点组配置”模块选择节点的配置类型和相应的应用组件。

- “网络及其他”模块进行网络，IP相关设置。

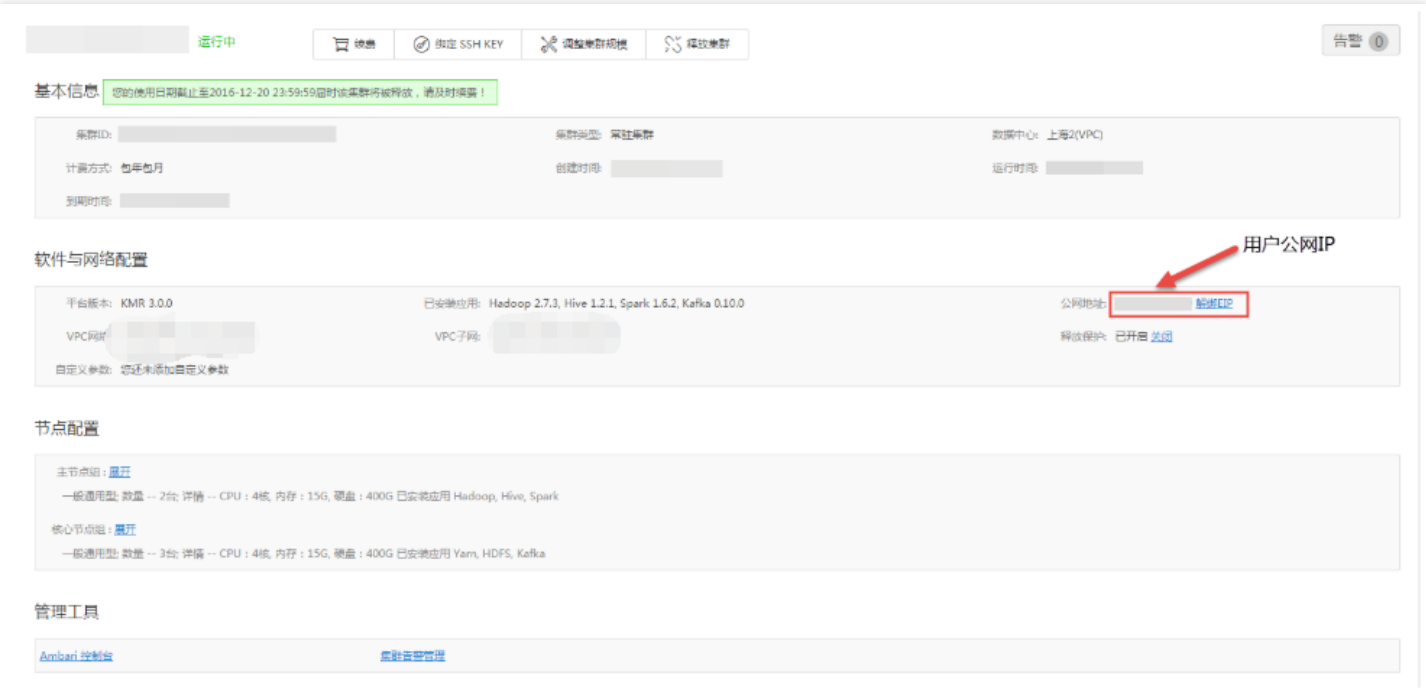


- 确认集群配置详情后，点击创建创建集群。

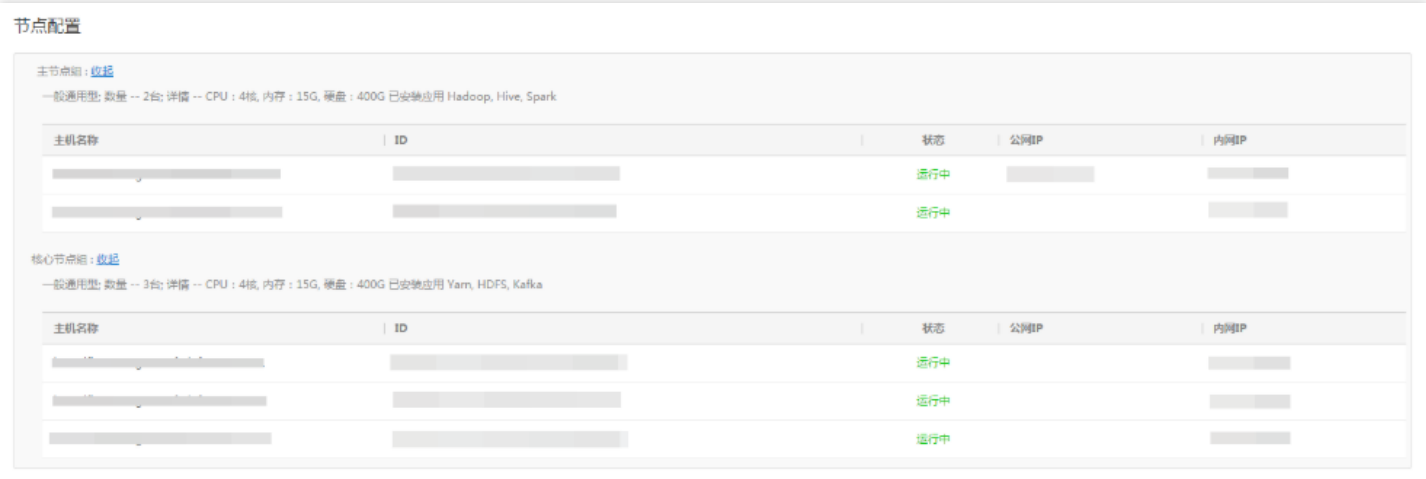
步骤三：登录集群主节点

最近更新时间: 2019-11-13 02:43:20

集群创建完成后点击集群名称进入集群详情页面



在节点配置项中点击“展开/收起”可以查看集群节点的详细信息



1. 点击”绑定SSH KEY”可以快速绑定公钥
2. SSH 登陆集群进行相关操作即可，登陆成功如下图所示

最佳实践

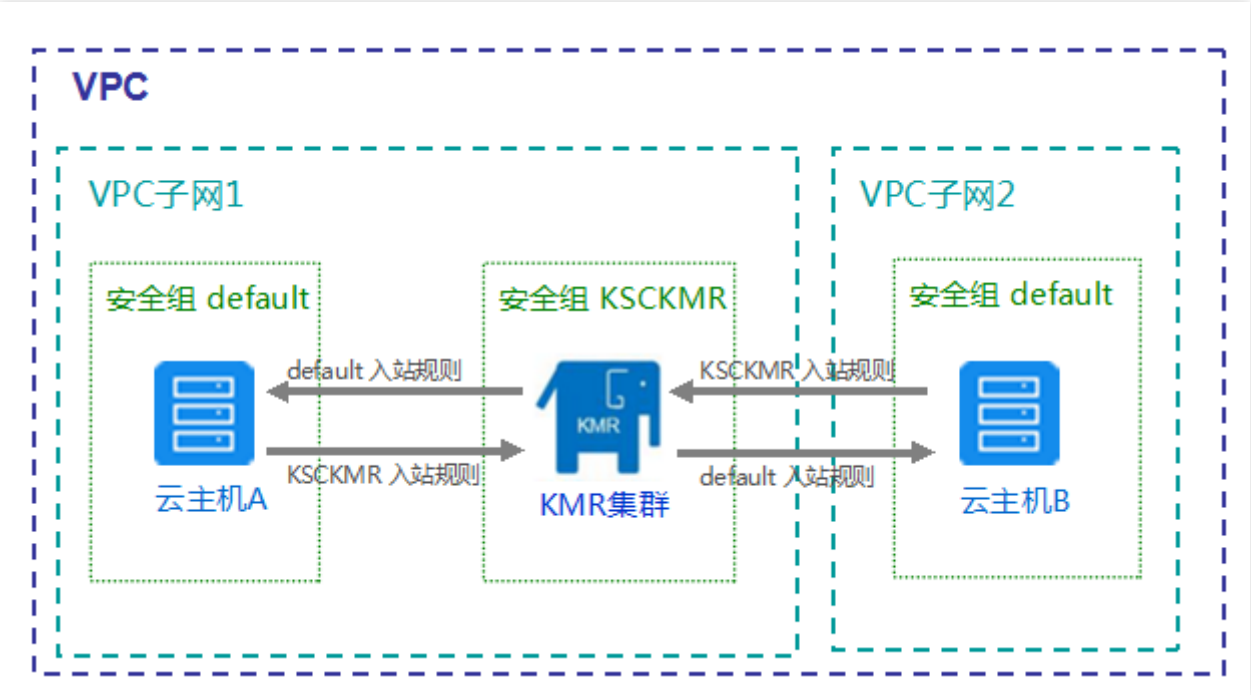
VPC网络配置

最近更新时间: 2019-10-31 02:46:18

托管HADOOP网络架构

最近更新时间: 2021-09-15 16:33:22

托管HADOOP采用云主机构建集群，二者服务都依托于VPC，创建于用户指定的VPC网络和VPC子网中，VPC具有安全组功能，将VPC中的产品划分不同的安全域，为每个安全域定义不同的访问控制规则。托管HADOOP集群默认安全组为“KSC托管HADOOP”，集群内部各节点在该安全组内可相互访问。每个云主机在创建时也会绑定唯一的安全组，默认为“default”，在同一VPC下，处于相同或不同子网的托管HADOOP集群和云主机可通过配置安全组规则



进行互通。

通过托管HADOOP集群访问非托管HADOOP集群云主机

最近更新时间: 2021-09-15 16:33:22

需要配置要访问的云主机对应VPC安全组的入站规则，允许托管HADOOP集群访问该云主机。

首先，从控制台进入“云服务器”，获取该云主机的VPC和安全组名称 从控制台进入“虚拟私有网络”，进入刚刚获取的VPC

选中该云主机对应的安全组，创建安全组规则

用户可根据需求创建不同方式的安全组规则，填写正确的托管HADOOP集群所在的网段即可。

通过非托管HADOOP集群云主机访问托管HADOOP集群

最近更新时间: 2021-09-15 16:33:22

需要配置托管HADOOP集群对应VPC安全组的入站规则，允许其他云主机访问托管HADOOP集群。
首先查看托管HADOOP集群的VPC网络，然后从控制台进入“虚拟私有网络”，进入该VPC，托管HADOOP集群默认的安全组为“KSC托管HADOOP”，进入该安全组并配置安全组入站规则，填写正确的非托管HADOOP集群云主机的网段即可配置互通。
注意：
以上配置托管HADOOP集群和其他云主机互通的前提是在同一VPC下，不同的VPC下主机互通需要先对VPC进行配置使VPC互通，再配置安全组规则。
VPC安全组规则默认出站规则全部放行，因此，可不配置出站规则，如有特殊需求，可自行更改。

开通非master1节点的外网访问权限

最近更新时间: 2021-09-15 16:33:22

默认情况下，托管HADOOP集群中只有Master1有外网访问权限，通过外网EIP联结外网。
如果其他节点也需要访问外网，则需要在对应VPC或子网中建议一个外网的NAT。
注意：一个VPC只能绑定一个NAT，因此，请删除，默认创建的NAT，再创建新的NAT。
NAT相关的问题可以联系负责网络的同事。

MapReduce

最近更新时间: 2019-10-31 02:40:51

准备

最近更新时间: 2021-09-15 16:33:22

1.您已经创建托管HADOOP集群以及相关其他服务，托管HADOOP集群默认安装Hadoop
2.您已经使用SSH连接到集群，以下操作均在集群主节点进行。

MapReduce作业

最近更新时间: 2021-09-15 16:33:22

以WordCount为例，介绍如何使用Hadoop实现单词统计功能

程序准备

最近更新时间: 2021-09-15 16:33:18

以下代码来源于Hadoop官网Wrodcount实例代码

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.Hadoop.conf.Configuration;
import org.apache.Hadoop.fs.Path;
import org.apache.Hadoop.io.IntWritable;
import org.apache.Hadoop.io.Text;
import org.apache.Hadoop.mapreduce.Job;
import org.apache.Hadoop.mapreduce.Mapper;
import org.apache.Hadoop.mapreduce.Reducer;
import org.apache.Hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.Hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

public static class TokenizerMapper
extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);
private Text word = new Text();

public void map(Object key, Text value, Context context
) throws IOException, InterruptedException {
StringTokenizer itr = new StringTokenizer(value.toString());
while (itr.hasMoreTokens()) {
word.set(itr.nextToken());
context.write(word, one);
}
}
}

public static class IntSumReducer
extends Reducer<Text,IntWritable,Text,IntWritable> {
private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,
Context context
) throws IOException, InterruptedException {
int sum = 0;
for (IntWritable val : values) {
```

```
sum += val.get();
}
result.set(sum);
context.write(key, result);
}
}

public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "word count");
job.setJarByClass(WordCount.class);
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

设置JDK和Hadoop环境变量，假设java安装路径为/usr/java/jdk1.7.0_51

export JAVA_HOME=/usr/java/jdk1.7.0_51

export PATH=${JAVA_HOME}/bin:${PATH}

export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar

编译程序

mkdir wordcount_classes

Hadoop com.sun.tools.javac.Main –d wordcount_classes/ WordCount.java

jar cvf wordcount.jar –C wordcount_classes .

这样就得到了wordcount.jar

作业输入输出

最近更新时间: 2019-10-31 02:40:20

Hadoop作业的输入和输出文件，可以放在HDFS上。 使用HDFS 将输入文件放到HDFS上，假设输入文件为TWILIGHT.txt Hadoop dfs –mkdir –p /user/Hadoop/examples/input

Hadoop dfs –put TWILIGHT.txt /user/Hadoop/examples/input

作业提交

最近更新时间: 2019-10-31 02:40:20

命令行提交 输入输出在HDFS上： Hadoop jar wordcount.jar WordCount /user/Hadoop/examples/input/ /user/Hadoop/examples/output

Hadoop Streaming允许用户使用可执行的命令或者脚本作为mapper和reducer。以下用几个示例说明Hadoop Streaming如何使用。详细可参考Hadoop官网Hadoop Streaming 使用shell命令作为mapper和reducer Hadoop jar /opt/Hadoop/share/Hadoop/tools/lib/Hadoop-streaming-*.jar -input /user/Hadoop/examples/input/ -output /user/Hadoop/examples/output1 -mapper /bin/cat -reducer /usr/bin/wc 使用python脚本作为mapper和reducer mapper.py

```
#!/usr/bin/env python
import sys
for line in sys.stdin:
line = line.strip()
words = line.split()
for word in words:
print "%s\t%s" % (word, 1)
reducer.py
#!/usr/bin/env python
from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
line = line.strip()
word, count = line.split('\t', 1)
try:
count = int(count)
except ValueError:
continue
if current_word == word:
current_count += count
else:
if current_word:
print "%s\t%s" % (current_word, current_count)
current_count = count
current_word = word
```

```
if word == current_word:
print "%s\t%s" % (current_word, current_count)
```

命令行执行streaming作业 chmod +x mapper.py

chmod +x reducer.py

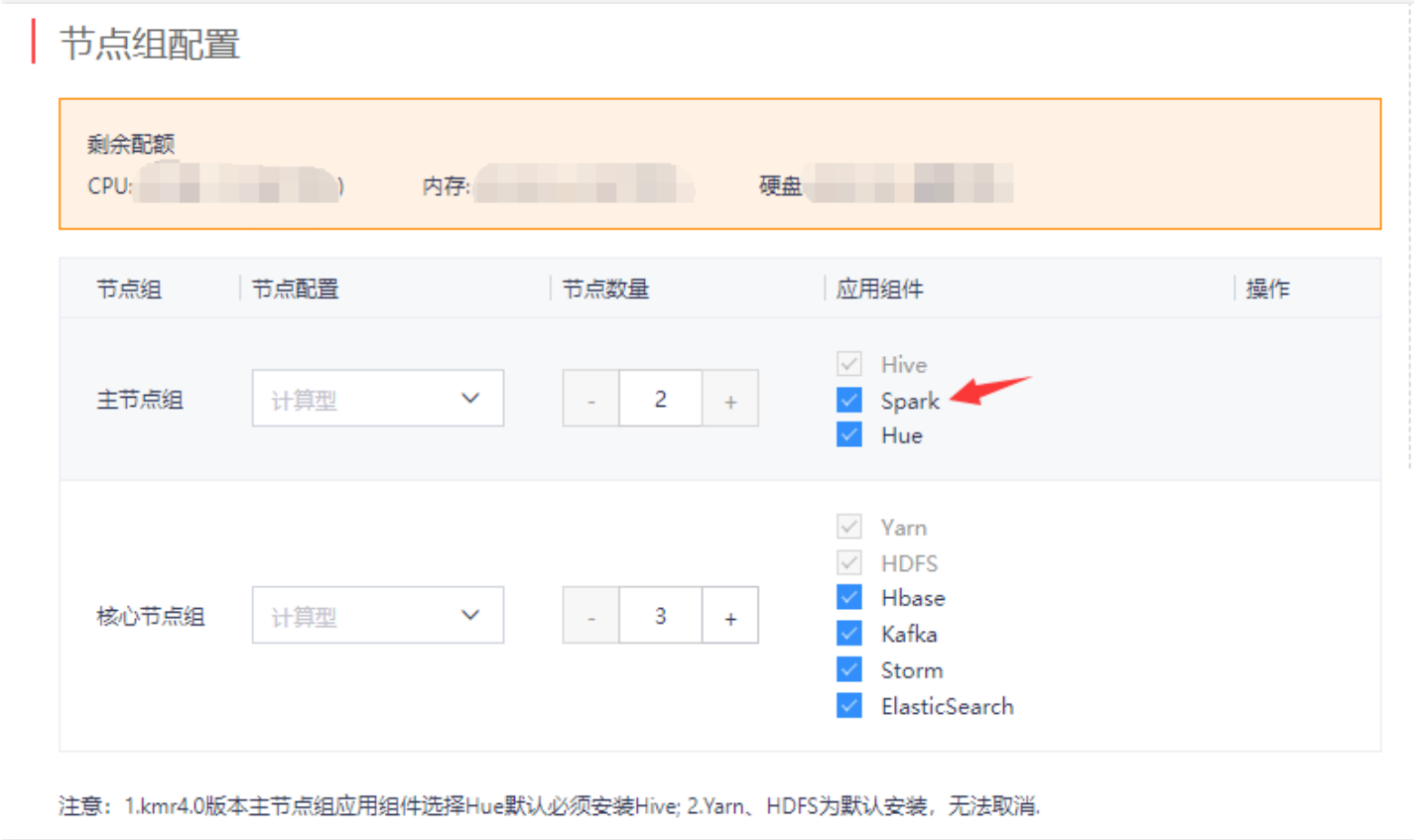
Hadoop jar /opt/Hadoop/share/Hadoop/tools/lib/Hadoop-streaming-*.jar -input /user/Hadoop/examples/input/ -output /user/Hadoop/examples/output2 -mapper mapper.py -reducer reducer.py -file mapper.py -file reducer.py

Spark实践指南 Apache Spark 是开源的集群框架和编程模型，与Hadoop类似，也是一款常用于大数据处理的分布式系统。Spark 与Hadoop的不同之处在于 Spark 拥有经过优化的有向无环图 (DAG) 执行引擎并会积极地在内存中缓存数据，这可提高性能，尤其是对于某些算法和交互式查询。 作为托管Hadoop的组件之一，我们对其进行了高可用处理，而且 Spark 可以与托管HADOOP的其他应用程序一同安装。除此之外托管HADOOP也将 Hive 与 Spark 做了集成，您可以通过 HiveContext 对象运行使用 Spark 的 Hive 相关的操作。 本文主要对一些 Spark 的基础操作进行了讲解，希望能够对您在托管HADOOP上使用 Spark 有所帮助。 全文会以场景的形式对 Spark 的一些功能和使用方式进行介绍，通过模拟各种场景，可以使您更快的熟悉托管HADOOP上的 Spark 组件的使用方法。 在正式了解相关内容前，您需要进行一些准备工作。

准备

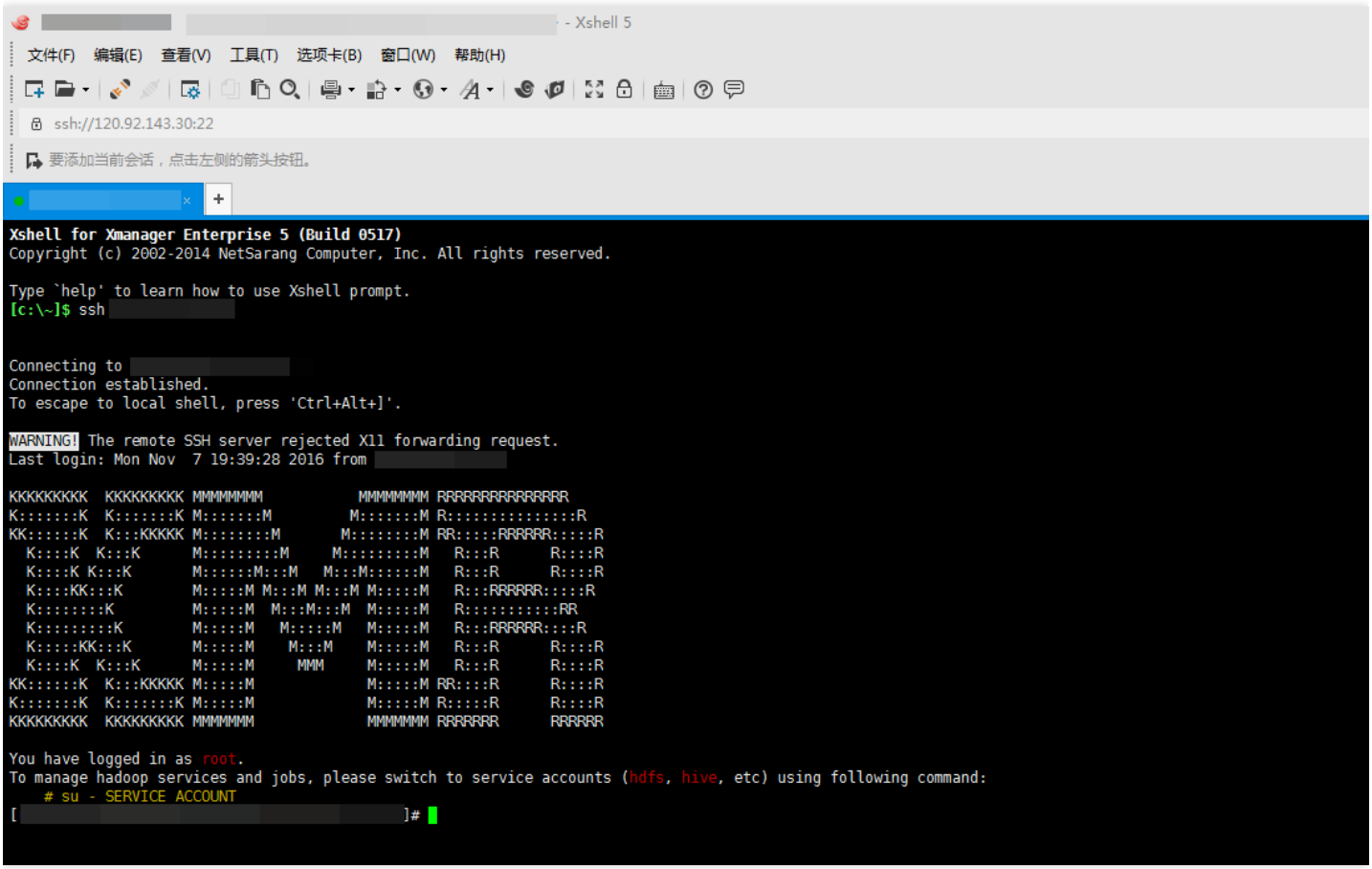
最近更新时间: 2019-11-13 02:43:20

创建托管HADOOP集群并勾选 Spark 服务



通过SSH连接到集群 获取公网的云主机 MASTER 节点的 IP，您可在托管HADOOP WEB 控制台数据分析>托管HADOOP>集群管理>集群详情页面查看，如下图所示

绑定 SSH 密钥（也可以使用密码登陆，但是 SSH 密钥更安全、方便） SSH登陆即可 ssh root@您要连接的远程主机IP ，登陆成功后如下图所示



到此已经完成所有准备工作

Spark 离线任务处理

最近更新时间: 2019-10-31 02:40:20

所谓离线任务，就是用户已经将待处理的数据存储在对象存储上或者是 HDFS 中，这些数据是“过去”产生的，并不是实时产生的。用户在提交任务（job）以后，由集群自动完成计算并得出结果。 一般情况下，在进行数据清洗时，大多采用离线处理的方式。具体到场景来说，比如日志分析、用户行为分析（日志分析的一种）等。这些数据大都具有一次写入，多次读取的特点。一般要处理具有这种特点的数据大都采用离线处理的方式。 以下会有几个例子，用来熟悉两个经常遇到的场景的处理方法。 以下内容不做特殊说明的情况下，凡是执行 SPARK 相关的命令时，请确保您的系统用户已经由 root用户 切换到 spark 用户； 切换当前工作目录到 /home/spark #切换系统用户为 spark #su spark #切换工作目录到当前用户的 home 目录 #cd ~

入门操作

最近更新时间: 2019-10-31 02:40:20

当前场景下，用户将待处理数据存储在本地计算机，需要将其上传至 HDFS 再进行计算处理。 程序准备 这里将会以计算 pi（圆周率）为例来演示离线任务处理。 用户可以使用spark-submit命令来提交 SPARK 任务。spark-submit 具体使用可以通过 spark-submit --help 查看。 注意：此包由官方提供，因此无需上传 jar 包 程序提交 在命令行中执行如下命令

```
yarn-client
#spark spark-submit --class org.apache.spark.examples.SparkPi --master yarn --deploy-mode client --driver-memory 4g --num-executors 2 --executor-memory 2g --executor-cores 2 /usr/hdp/2.4.0.0-169/spark/lib/spark-examples.jar 10 2>/dev/null
```

```
yarn-cluster
#spark-submit --class org.apache.spark.examples.SparkPi --master yarn --deploy-mode cluster --driver-memory 4g --num-executors 2 --executor-memory 2g --executor-cores 2 /usr/hdp/2.4.0.0-169/spark/lib/spark-examples.jar 10 2>/dev/null
```

注意：因为是 yarn cluster 模式，所以在终端看不到任何输出信息（这里仅指有效输出信息，debug 的标准输出已被屏蔽），需要在 yarn 的控制台 查看打印的日志 log，可通过: AMBARI > YARN > Quick Links > ResourceManager UI来查看 相关参数说明

参数	参考值	说明
class	org.apache.spark.examples.SparkP	作业的主类，程序的入口
master	yarn	托管HADOOP使用Yarn的模式（也支持Standalone模式，后接集群master的URL，如local或者spark:\/\/host:port，生产环境下推荐使用YARN）
yarn-client	简写，等效--masteryarn--deploy-modeclient，即--masteryarn-client	
yarn-cluster	简写，等效--masteryarn--deploy-modecluster，即--masteryarn-cluster	
deploy-mode	client	部署方式（client\/cluster），client模式表示作业的ApplicationMaster会放在Master节点上运行。注意，该参数需要和--masteryarn连用_
cluster	cluster模式表示作业的ApplicationMaster会随机的在core节点中	

	的任意一台上启动运行。注意，该参数需要和--masteryarn连用	
river-memory	4G	driver使用的内存
num-executors	2	创建executor的数量
executor-memory	2G	每个executor使用的最大内存，不能超过单机的最大可使用内存
executor-cores	2	各个executor使用的并发线程数目，也即每个executor最大可并发执行的Task数目

补充：上面仅是 spark-submit 一部分参数，详情请点击此处。 spark-submit 参数填写完毕，后跟 jar 包所在路径，最后 2>/dev/null 是为了屏蔽程序 debug 信息的标准输出。

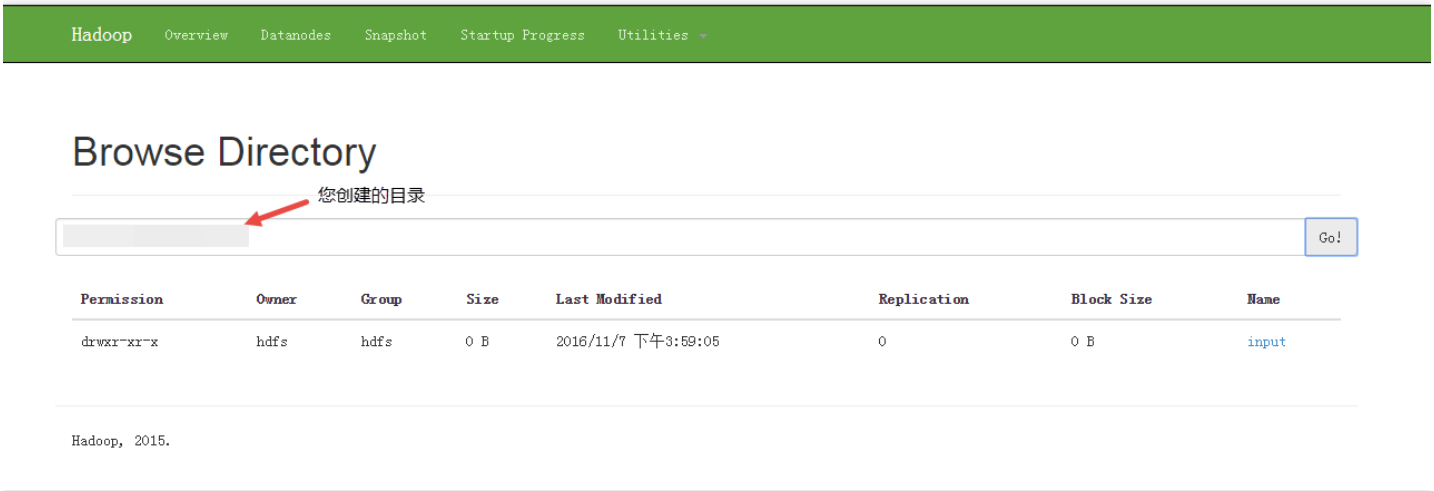
场景一：数据储存在本地

最近更新时间: 2019-11-13 02:43:20

程序准备 任务源码，点击这里下载 准备输入文件 格式要求 任意字符串，彼此用空格作为分隔符 若干行，也可以直接用 spark 包下面的README.md文件 上传所需 jar 包以及要处理的输入文件 将生成的 jar 包和待处理的文件 in.file ，通过xftp上传至 MASTER 主机，假设目录为/home/spark 将 in.file 文件上传至 HDFS ，命令如下:

```
# sudo -u hdfs hdfs dfs -mkdir -p /user/YOURNAME/testdata/input
# sudo -u hdfs hdfs dfs -put /home/spark/in.file /user/YOURNAME/testdata/input
```

注意：第一条命令为创建文件输入目录，请勿自行创建文件输出目录；另外，在执行第二条命令前请检查当前所在目录 通过 Ambari > HDFS > Quick Links >任意一activity集群> NameNode UI > Utilities > Browse the file system 可以访问 HDFS 的 WEB 界面查看文件是否上传成功以及输出的文件，如下图所示



也可通过以下命令查看

```
# sudo -u hdfs dfs -ls /user/YOURNAME/testdata/input
```

任务提交 执行下方指令即可

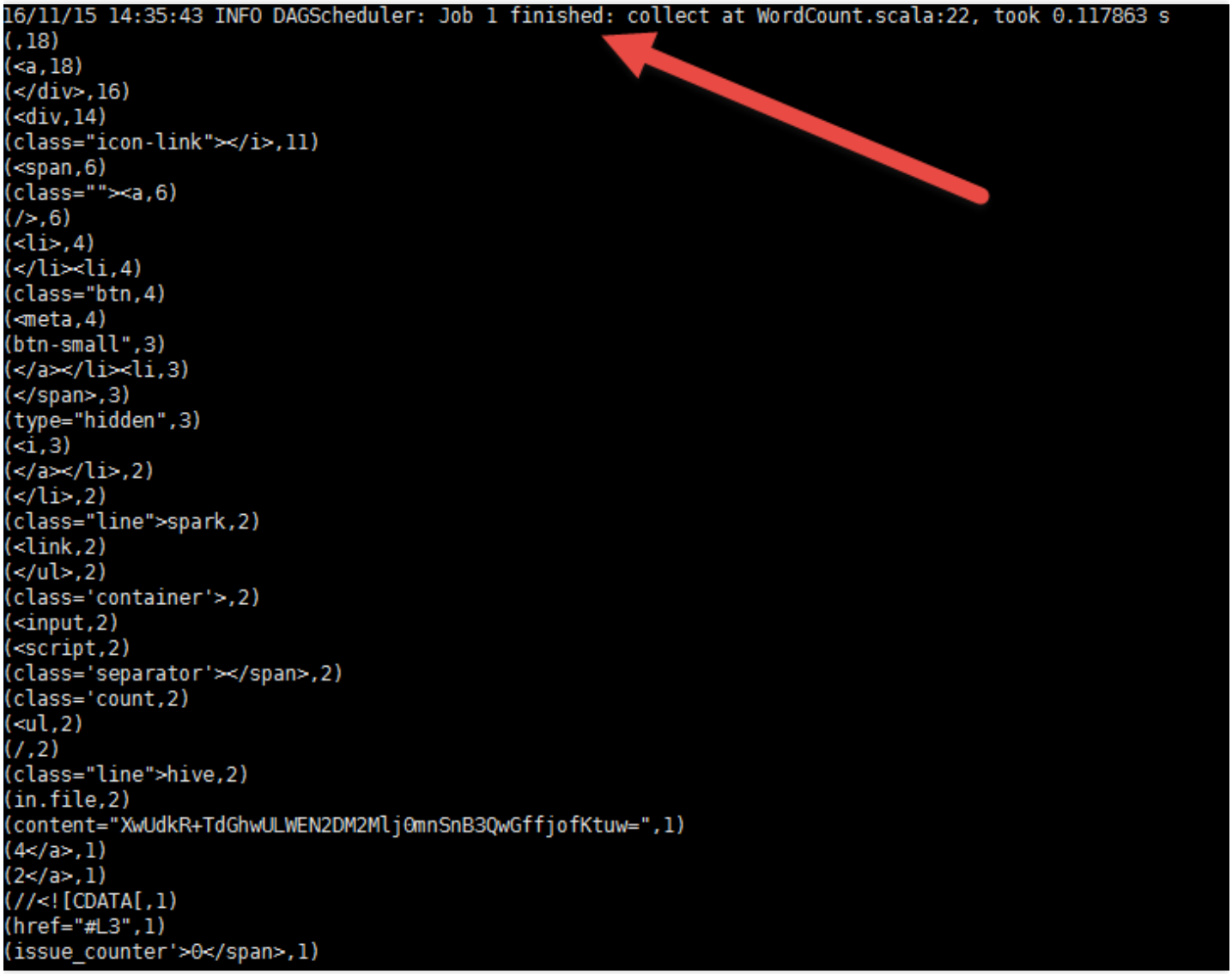
```
local(file on hdfs)
# sudo -u spark spark-submit --class com.托管Hadoop.demo.WordCount --master local[2] wordcount
-1.0-SNAPSHOT.jar "/user/YOURNAME/testdata/input" "/user/YOURNAME/testdata/output"
yarn-client(inputfile on local，outputfile on hdfs)

#sudo -u spark spark-submit --class com.托管Hadoop.demo.WordCount --master yarn-client /home/
spark/wordcount-1.0-SNAPSHOT.jar file:///usr/hdp/2.4.0.0-169/spark/README.md hdfs:///user/YOU
RNAME/output

yarn-cluster ( file on hdfs )
```

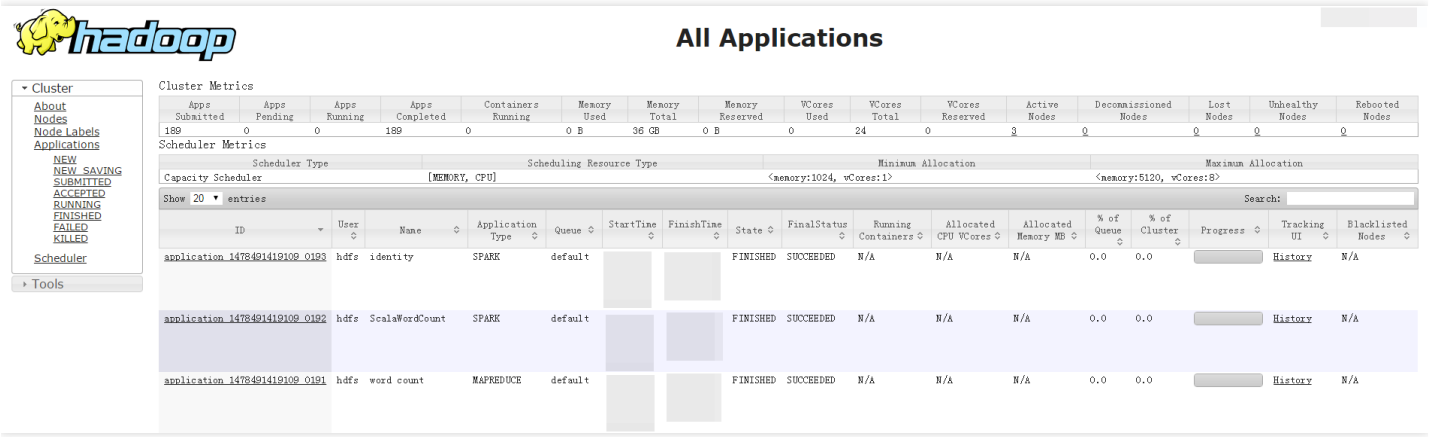
```
# sudo -u spark spark-submit --class com.托管Hadoop.demo.WordCount --master yarn --deploy-mo
de cluster /home/spark/wordcount-1.0-SNAPSHOT.jar "/user/YOURNAME/testdata/input" "/user/YO
URNAME/testdata/output"
```

查询结果 在命令行中查询 在执行完命令后，在屏幕的打印信息中可以找到相应内容，如下图所示



在Ambari控制台查询 任务完成后以在集群详情页 –> Ambari控制台 –> YARN –> Quicklinks –>集群ID –>

ResourceManager 查询，结果如下图所示



SparkSQL 的使用及操作

最近更新时间: 2019-10-31 02:40:20

在 Spark 中的一栈式解决方案中，最常用的组件之一就是 Spark SQL，它是 Spark 的一个结构化数据处理模块，其最大优势在于性能非常高，而且还使用了基于成本的优化器、列储存、代码生成等技术。此外 Spark SQL也可以扩展到上千个计算节点以及数小时的计算能力，并且支持自动容错恢复。 使用 Spark SQL 有两种方式：一种是作为分布式的 SQL 引擎，只需写 SQL 就可以进行计算，无需复杂编码；另一种是在 Spark 程序中，通过 API 的形式来操作数据。以下会有一些例子（场景）用于介绍如何在托管HADOOP中如何更高效的使用 Spark SQL。 以下例子中所有的源码可以点击这里下载

数据基本操作

最近更新时间: 2019-11-26 15:30:11

数据准备 数据输入源：spark 包自带 peopele.txt 和 people.json 数据输入源路径：/usr/hdp/2.4.0.0-169/spark/examples/src/main/resources

```
people.json
{"name":"Michael"}
{"name":"Andy", "age":30}
{"name":"Justin", "age":19}
people.txt
Michael, 29
Andy, 30
Justin, 19
```

通过 spark-shell 创建 DataFrame 操作

```
#切换 linux 系统用户为 spark (默认登录托管HADOOP集群的用户为 root )
#su spark
#进入 spark-shell 模式，资源管理使用时 yarn ，部署方式是 client
$ spark-shell --master yarn-client
#创建 DataFrame
scala> val sqlContext = new org.apache.spark.sql.SQLContext(sc)
scala> val df = sqlContext.read.json("file:///usr/hdp/2.4.0.0-169/spark/examples/src/main/resources/people.json")
scala> df.show()
+----+-------+
| age| name|
+----+-------+
|null|Michael|
| 30| Andy|
| 19| Justin|
+----+-------+
```

RDD 转成 DataFrame（使 people.txt 非结构化的数据也可以运行SQL操作） 方式一:反射机制推断 RDD 模式，得到 DataFrame(在源码中找到DDToDataFrame项目即可) 方式二:以编程方式定义 RDD 模式，得到 DataFrame(在源码中找到DDToDataFrame项目即可) 执行

```
#第一步：将代码打包
#第二步：上传代码到 MASTER 的主机 spark 目录下，即 /home/spark/
#第三步：提交 job ，启动 DRIVER 程序
#注意：最后一个参数为整数类型，其中 1 为方式一将普通 RDD 转成 DataFrame ；其他为方式二
#sudo -u spark spark-submit --class com.托管Hadoop.rddToDF.Demo --master yarn-client /home/sp
```

```
ark/RDDToDataFrame-1.0-SNAPSHOT.jar "file:///usr/hdp/2.4.0.0-169/spark/examples/src/main/resources/people.txt" "hdfs:///user/spark/out/person" 1
```

• 数据加载 DataFrame 提供统一接口加载以下数据load()方式， format() 指定具体数据源格式 结构化数据 Parquet JSON Hive 表 JDBC 连接外部数据 shell 下的代码如下：

```
#首先，切入 shell 模式，部署方式为 yarn-client
#cd /home/spark && sudo -u spark spark-shell --master yarn-client
#通过 spark-shell 为例， format() 指定加载格式，默认是 Parquet ，可以通过 spark.sql.sources.default 参数修改
scala> val df = sqlContext.read.format("json").load("file:///usr/hdp/2.4.0.0-169/spark/examples/src/main/resources/people.json")
#通过 save() 进行存盘
df.select("name","age").write.format("parquet").save("/usr/qjjia/namesAndAges.parquet")
*注意：执行#cd /home/spark && sudo -u spark spark-shell --master yarn-client这条命令前请检查当前所在目录*
```

场景一：结构化数据处理

最近更新时间: 2019-10-30 06:18:08

这里分别以 JSON 文件和 Hive 表为例 JSON文件操作（ people.json --> RDD --> DataFrame -->注册临时表 --> SQL查询） 项目名：JSONFile 项目核心代码（具体代码,您可以点这里，自行下载）：

```
import org.apache.spark.sql.SQLContext
import org.apache.spark.{SparkConf, SparkContext}

object Demo {
def main(args: Array[String]) {
val dirIn = args(0) //数据输入PATH

val conf = new SparkConf().setAppName("JSON")
// SparkContext 是程序和集群的唯一通道
val sc = new SparkContext(conf)
// 通过SparkContext 创建SQLContext
val sqlContext = new SQLContext(sc)

val people = sqlContext.read.json(dirIn)

people.printSchema()

//注册DataFrame作为一个临时表
people.registerTempTable("jsonTable")

//使用SQL 语句操作
val teenagers = sqlContext.sql("select name from jsonTable where age >=13 and age <= 19")
teenagers.map(t => "Name:" + t(0)).collect.foreach(println)

// val anotherRDD = sc.parallelize("""{"name":"spark","address":{"city":"USA","avenue":"SEVEN"}}""" ::
Nil)
// val anotherPeople = sqlContext.read.json(anotherRDD)
sc.stop()
}
}
```

提交 job，同上RDDToDataFrame submit 过程

sudo –u spark spark–submit ––class com.托管Hadoop.JSON.Demo ––master yarn–client /home/spark/JsonFile–1.0–SNAPSHOT.jar ”file:///usr/hdp/2.4.0.0–169/spark/examples/src/main/resources/people.json”

Hive 表操作 项目名：SparkSQL_Hive INPUT:/usr/hdp/2.4.0.0–169/spark/examples/src/main/resources/kv1.txt 格式为：string + 空格 + string （一行） 核心代码（具体代码,您可以点这里，自行下载）：

```
object Demo {
def main(args: Array[String]) {
val dirIn = args(0)
val conf = new SparkConf().setAppName("RDDToDF")
val sc = new SparkContext(conf)

//通过sc创建HiveContext的实例hiveContext
val hiveContext = new HiveContext(sc)
//通过HiveContext的sql命令创建表
hiveContext.sql("create table if not exists src (key int,value string)")
//加载数据
hiveContext.sql("load data local inpath '"+ dirIn +"' into table src")
//HiveQL 的查询表达
hiveContext.sql("from src select key,value").collect.foreach(println)
sc.stop()
}
}
```

提交 job ,命令如下：

sudo –u spark spark–submit ––class com.托管Hadoop.sparkHive.Demo ––master yarn–client /home/spark/sparkSQL_Hive–1.0–SNAPSHOT.jar

”file:///usr/hdp/2.4.0.0–169/spark/examples/src/main/resources/kv1.txt”

场景二：非结构化数据处理

最近更新时间: 2019-10-30 06:30:22

处理非结构化数据的逻辑是先将其转换为结构化数据再进行处理 处理流程为：people.txt --> RDD --> DataFrame -- saveAs Parquet;load Parquet -->重构 DataFrame -->注册临时表 --> SQL 查询 这里以上文中提到的people.txt为例 项目名：ParquetFile 项目核心代码（具体代码,您可以点这里，自行下载）：

```
val people = sqlContext.read.parquet(dirIn)
peopleDF.write.parquet(dirOut)
```

提交 job，同上RDDToDataFrame submit 过程

sudo –u spark spark–submit ––class com.托管Hadoop.parquet.Demo ––master yarn–client /home/spark/ParquetFile–1.0–SNAPSHOT.jar ”file:///usr/hdp/2.4.0.0–169/spark/examples/src/main/resources/people.txt” ”hdfs:///user/spark/out/person2.parquet”

场景三：多源数据整合处理

最近更新时间: 2019-10-30 06:30:22

有时我们就会遇到多种数据源同时存在的情况，此时，通过不同的数据源，构造基于相同的 Schema 的 DataFrame ，进行汇总操作，可用合并操作，这里有一个例子来帮助您理解。 项目名：comprehensive 核心代码（具体代码,您可以点这里，自行下载）：

```
// 使用case 定义类 Log
case class Log(id:String,info:String)

object Demo {
def main(args: Array[String]) {

val dirOut1 = args(1)
val dirOut2 = args(2)
val conf = new SparkConf().setAppName("RDDToDF")
val sc = new SparkContext(conf)
val sqlContext = new SQLContext(sc)
mix
import sqlContext.implicits._
val df1 = sc.parallelize(Array(("id1","info1"),("id2","info2"))).map(l => Log(l._1,l._2)).toDF

//查看 Schema 架构
df1.printSchema()

//df1 文件保存成 Parquet文件
df1.write.parquet(dirOut1)

val df2 = sc.parallelize(Array(("id3","info3"),("id4","info4"),("id5","info5"))).map(l =>Log(l._1,l._2)).toDF
df2.write.parquet(dirOut2)

//数据源1 进行加载
val data1 = sqlContext.read.parquet(dirOut1)
//数据源2 进行进行加载
val data2 = sqlContext.read.parquet(dirOut2)
//数据源进行整合
val data3 = data1.unionAll(data2)
//注册临时表
data3.registerTempTable("logs")
//查询执行
sqlContext.sql("select * from logs").collect.foreach(println)

sc.stop()
```

```
  }
  }
```

提交 job，命令如下：

```
sudo –u spark spark–submit ––class com.托管Hadoop.comprehensive.Demo ––master yarn–client /home/spark/comprehensive–1.0–SNAPSHOT.jar ”hdfs:///user/spark/out/log1.parqute” ”hdfs:///user/spark/out/log2.parqute”
```

Spark SQL 命令行操作

最近更新时间: 2019-10-30 06:12:11

- 本地模式运行 Hive Metastore 服务的一个有效工具，通过命令行接收查询输入
- 执行代码： #cd /home/spark && sudo -u spark spark-sql --master yarn-client

Spark 在流式处理中的应用

最近更新时间: 2019-11-26 15:30:11

- 日常处理数据的过程中，除了离线处理，也有数据实时产生实时处理的情况。为满足实时处理数据的需求，就需要整合数据源（数据生产者）、处理组件以及结果输出这三部分，以达到流式处理（实时处理）的目的。
- 以下会有一个例子用 Kafka 来实现流式处理，更多的使用方式请参考SparkStreaming 官方文档
- 在做流式处理以前，需要引入一个概念 Dstreams（Discretized Streams），这是 Spark 实现流式处理所必需的一种高度抽象的数据结构。详细信息请参考Apache Spark Streamming。

Kafka 和 Spark 来实现流式处理

最近更新时间: 2019-10-30 06:10:35

项目名：KafkaWordCount 输入源：伪造的数据（KafkaWordCountProducer） 模拟 Kafka 过程： Xshell 远程登录 MASTER 主机 用 ssh 切换到任意一台核心节的主机上（ CORE 节点），具体步骤如下 #查看 CORE 的主机名 #cat /etc/hosts 172.31.. 托管Hadoop-core-1-001.ksc.com 托管Hadoop-core-1-001 172.31.. 托管Hadoop-core-1-002.ksc.com 托管Hadoop-core-1-002 172.31.. 托管Hadoop-core-1-003.ksc.com 托管Hadoop-core-1-003 172.31.. 托管Hadoop-master-1-001.ksc.com 托管Hadoop-master-1-001 172.31.. 托管Hadoop-master-2-001.ksc.com 托管Hadoop-master-2-001

ssh 进入任意 CORE 节点

#ssh 托管Hadoop-core-1-001.ksc.com #查看 kafka 所在目录 #ps aux | grep kafka （producer）发送消息，具体执行过程如下： #切入 kafka 工作主目录

cd /usr/hdp/2.4.0.0-169/kafka/bin

#创建 topic #./kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test #检验 topic 创建是否成功（如果正常返回 test） #./kafka-topics.sh --list --zookeeper localhost:2181 #打开 producer，发送消息 #./kafka-console-producer.sh --broker-list 托管Hadoop-core-1-001.ksc.com:6667 --topic test #####启动成功后，输入以下内容测试 hdfs hdfs spark spark spark storm streaming DF DStream （consumer）接受消息，具体执行如下过程： #切入kafka工作主目录

cd /usr/hdp/2.4.0.0-169/kafka/bin

#保持 producer 端不动，另起，一个 shell 窗口登入当前 CORE 节点（托管Hadoop-core-1-001.ksc.com） #./kafka-console-consumer.sh --zookeeper localhost:2181 --topic test --from-beginning #####启动成功后，如果一切正常将会显示 producer 端输入的内容 源码路径： /usr/hdp/2.4.0.0-169/spark/examples/src/main/scala/org/apache/spark/examples/streaming/KafkaWordCount.scala 提交 job： 停止运行刚才的 kafka-console-producer 和 kafka-console-consumer 运行 KafkaWordCountProducer #切换到 MASTER 主机， spark 工作目录下 #cd /usr/hdp/2.4.0.0-169/spark/bin/

./run-example org.apache.spark.examples.streaming.KafkaWordCountProducer 托管Hadoop-core-1-001.ksc.com:6667 test 3 5

运行 KafkaWordCount #保持上一步骤的producer端口，另起，一个 shell 窗口连接 MASTER #切换到spark工作目录下 #cd /usr/hdp/2.4.0.0-169/spark/bin/

sudo -u spark ./run-example org.apache.spark.examples.streaming.KafkaWordCount 托管Hadoop-core-1-001.ksc.com:2181 test-consumer-group test 1

参数解释： KafkaWordCountProducer KafkaWordCountProducer 托管Hadoop-core-1-001.ksc.com:6667 表示 producer 的地址和端口 test 表示 topic 3 表示每秒发多少条消息 5 表示每条消息中有几个单词 KafkaWordCount 托管Hadoop-core-1-001.ksc.com:2181 表示 zookeeper 的监听地址 test-consumer-group 表示 consumer-group 的名称，必须和 $KAFKA_HOME/config/consumer.properties 中的 group.id 的配置内容一致 test 表示 topic 1 表示线程数。 到此您已经基本掌握了在托管HADOOP上使用 Spark 。

HUE使用实践指南

最近更新时间: 2019-11-26 15:29:57

HUE 总体概览

最近更新时间: 2019-11-26 15:29:57

HUE简介

最近更新时间: 2019-10-30 06:06:52

- Hue (Hadoop User Experience)是一个开源的Apache Hadoop UI系统，由Cloudera Desktop演化而来，最后Cloudera公司将其贡献给Apache基金会的Hadoop社区，其基于Python Web框架Django实现。Hue为托管HADOOP集群提供了图形化用户界面，便于用户配置、使用以及查看托管HADOOP集群。

相关连接

最近更新时间: 2019-11-26 15:29:57

- 官网: http://gethue.com/
- Github: https://github.com/cloudera/hue
- Reviews: https://review.cloudera.org

HUE功能

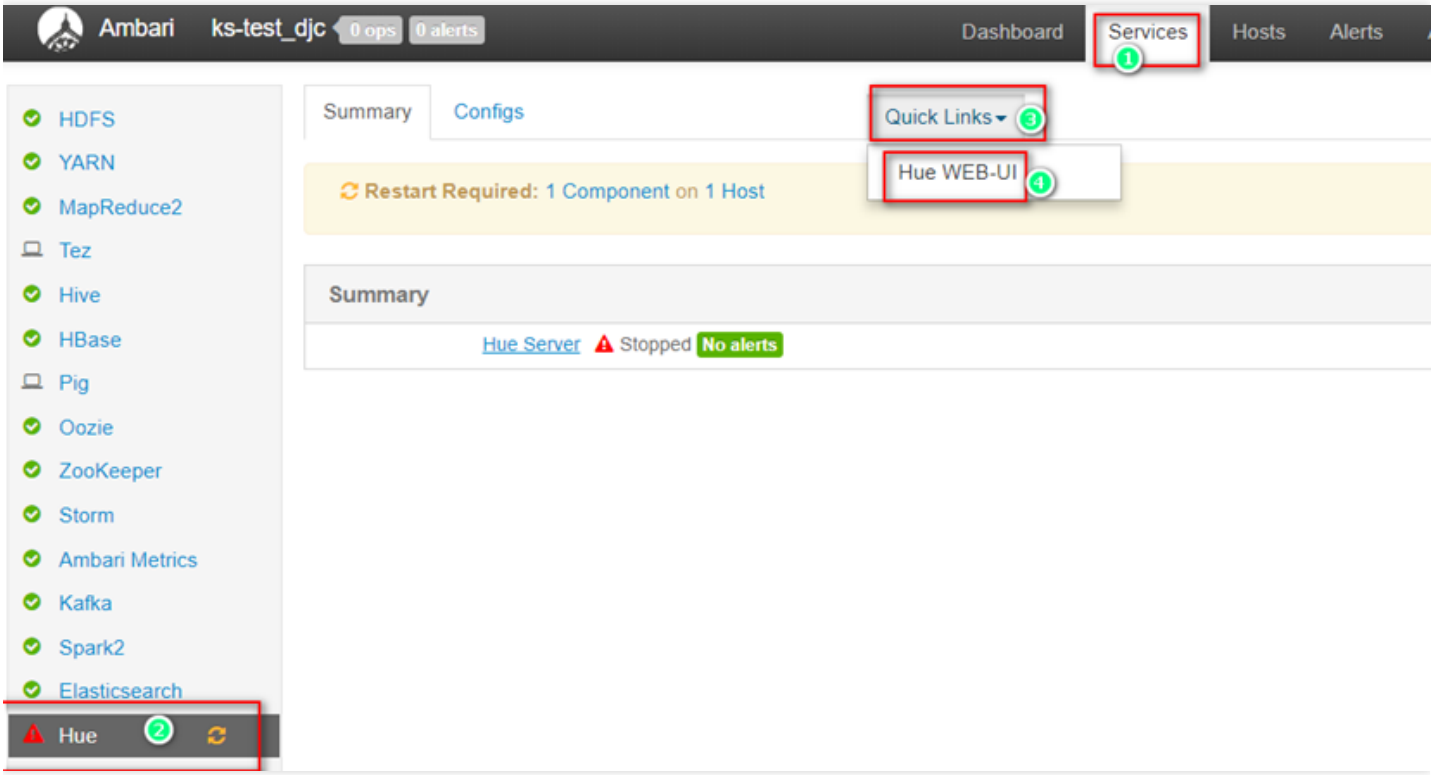
最近更新时间: 2019-11-26 15:29:57

- 访问HDFS和文件浏览
- 通过web调试和开发hive以及数据结果展示
- solr查询、结果展示、报表生成
- 通过web调试和开发impala交互式SQL Query
- spark调试和开发
- Pig开发和调试
- oozie任务的开发、监控和工作流协调调度
- Hbase数据查询和修改、数据展示
- Hive的元数据（metastore）查询
- MapReduce任务进度查看，日志追踪
- 创建和提交MapReduce，Streaming，Java job任务
- Sqoop2的开发和调试
- Zookeeper的浏览和编辑
- 数据库（MySQL、PostgreSQL、SQlite，Oracle）的查询和展示

登录HUE

最近更新时间: 2019-11-13 02:43:02

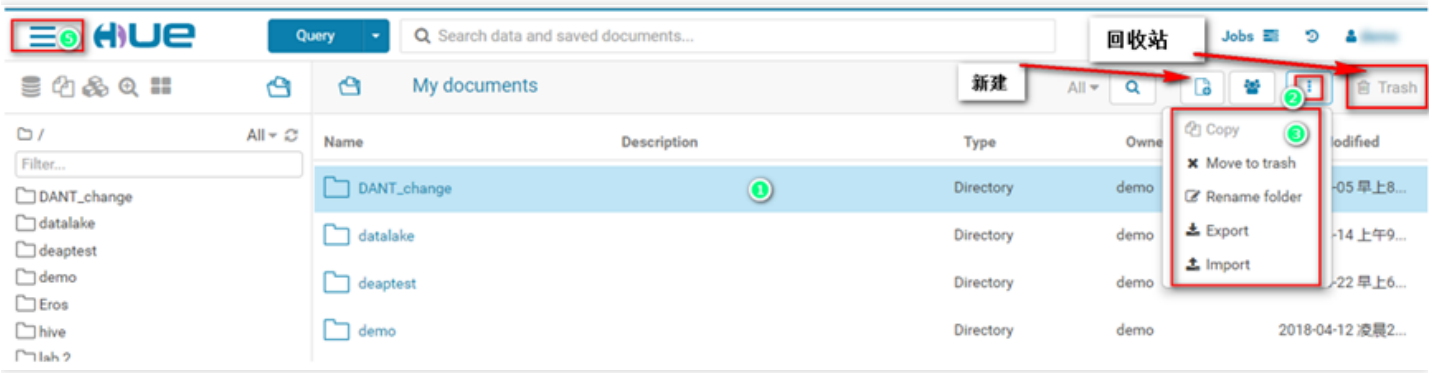
在托管HADOOP中的Ambari控制台Service界面中选择Hue，单击上边Quick Links下拉菜单，然后单击Hue WEB-UI即可进入Hue的Web页面。 首次登录会创建用户，请牢记首次登录的用户名和密码，如果忘记请联系售后重置密码。



HDFS文件浏览

最近更新时间: 2019-11-13 02:43:02

通过Hue的Web页面可方便查看HDFS中的文件及文件夹，以及对其进行创建、下载、上传、复制、修改及删除等操作。 单击文件/文件夹，单击②处，可对其进行③中所示的删除（Move to trash）、重命名（Rename folder）、导出（Export）、导入（Import）等工作。

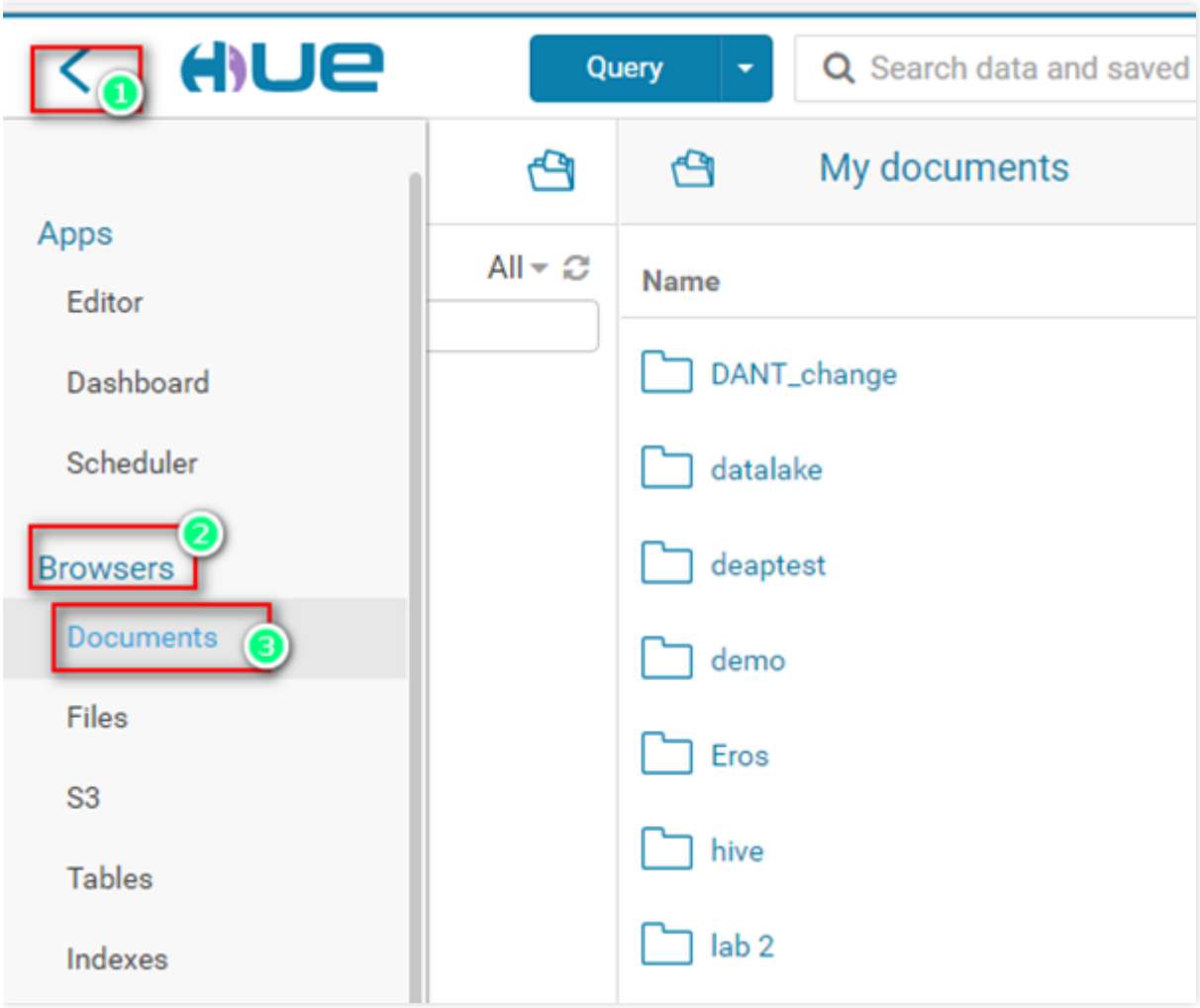


点击回收站处（Trash）可查看已删除文件/文件夹。

文件查看

最近更新时间: 2019-11-13 02:43:02

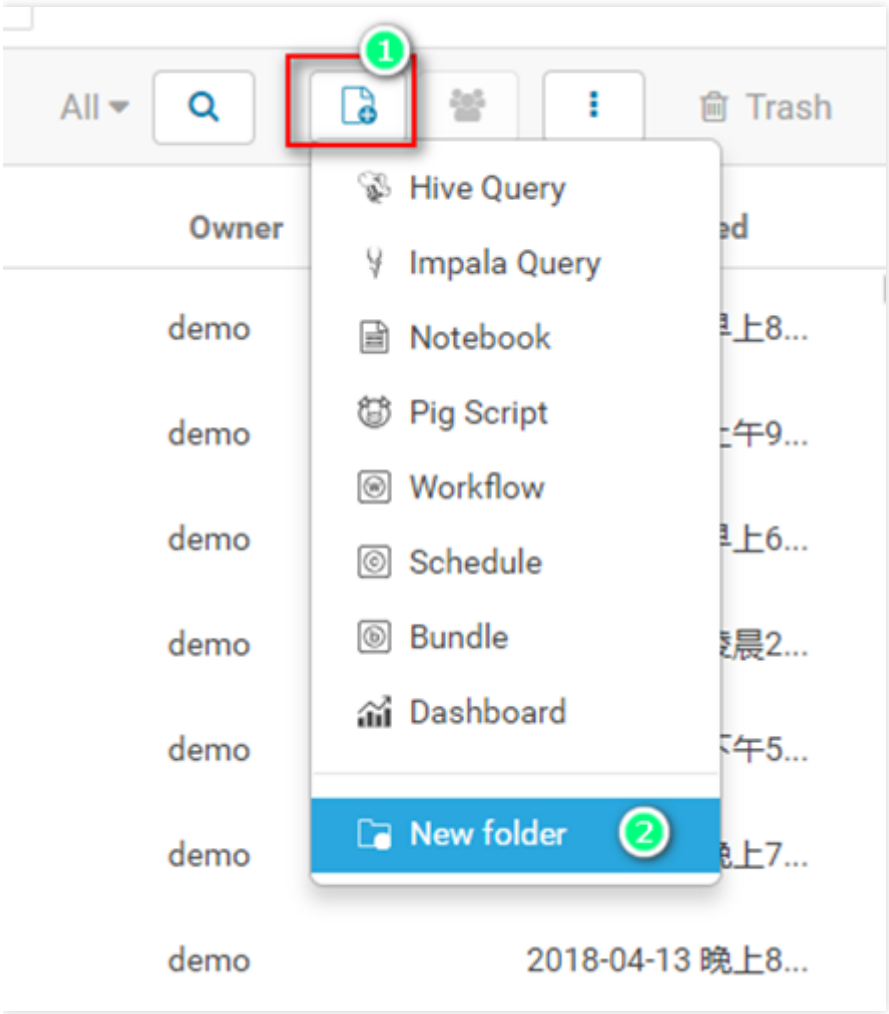
单击①处，本为上图中⑤处状态，鼠标放置其上时，变为下图①中状态，依次点击Browsers -- >Documents，可查看文件夹。



新建文件夹

最近更新时间: 2019-11-13 02:43:02

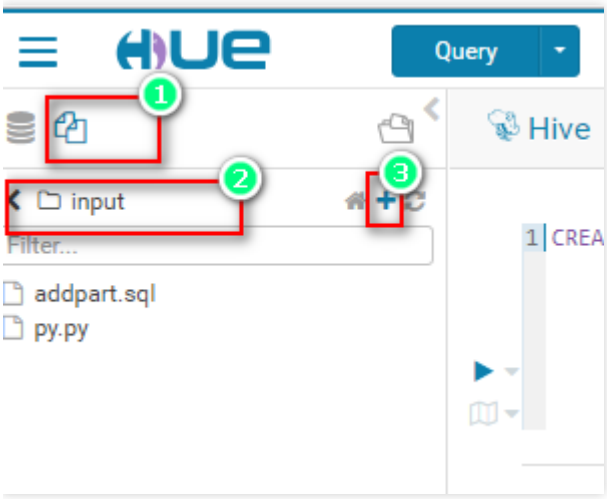
单击新建按钮（①处），单击最下方New Folder可新建文件夹。



文件

最近更新时间: 2019-11-13 02:43:02

在HUE Web页面可很方便的将文件上传至HDFS文件中 单击下图①处，选择文件要上传的文件夹，单击③处+号按钮选择本地文件进行上传



SQL查询

最近更新时间: 2019-11-26 15:29:57

Hive操作

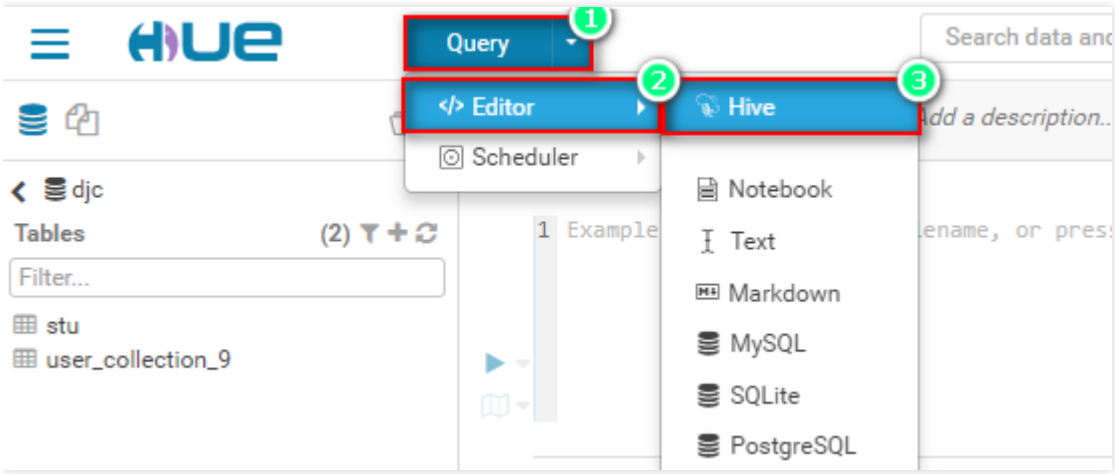
最近更新时间: 2019-10-30 05:59:36

HUE的beeswax app提供友好方便的Hive查询功能，能够选择不同的Hive数据库，编写HQL语句，提交查询任务，并且能够在界面下方看到查询作业运行的日志。在得到结果后，还提供进行简单的图表分析能力。

新建数据库

最近更新时间: 2019-11-13 02:43:02

页面新建 步骤： 点击Database后加号； 输入数据库名称； 输入数据库描述； 点击Submit； 输入框输入语句 页面单击Query→Editor→Hive



在语句输入框中输入要执行语句；然后单击动作执行按钮执行建表操作

注意： 在新建数据库时，可能会出现一下错误： Execution Error, return code 1 from org.apache.Hadoop.hive.ql.exec.DDLTask. MetaException(message:java.security.AccessControlException: Permission denied: user=hive, access=WRITE, inode="/":hdfs:hdfs:drwxr-xr-x 出现错误原因如下： HDFS权限不够 错误使用了 root 用户运行 hive/hbase 第一次以xxx用户运行 hive/hbase 使用 root 用户进行 HDFS 操作/跑 HIVE 脚本 ... 对于使用 root 用户进行 xxx 操作导致类似错误，我们不建议用户使用 root 用户进行类似操作，应切换到所使用服务对用的用户下进行操作，以减少不必要的问题。 对于第一次使用 Hive/HBase导致此错误，进行以下操作：

```
su -hdfs
#If error come with Hive
hdfs dfs -mkdir -p /apps/hive
hdfs dfs -chown hive:hdfs /apps/hive
```

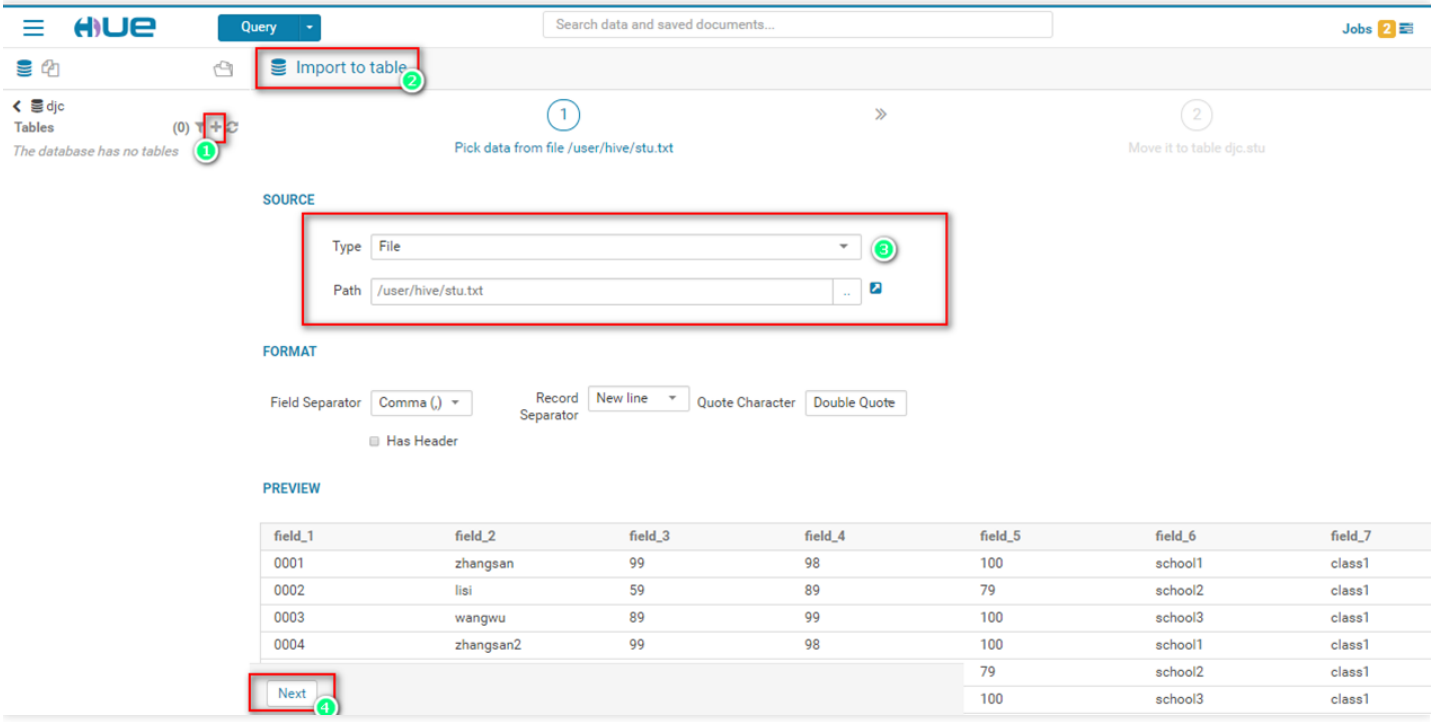
```
#If error come with HBase
hdfs dfs -mkdir -p /apps/hbase
hdfs dfs -chown hbase:hdfs /apps/hbase
```

新建表

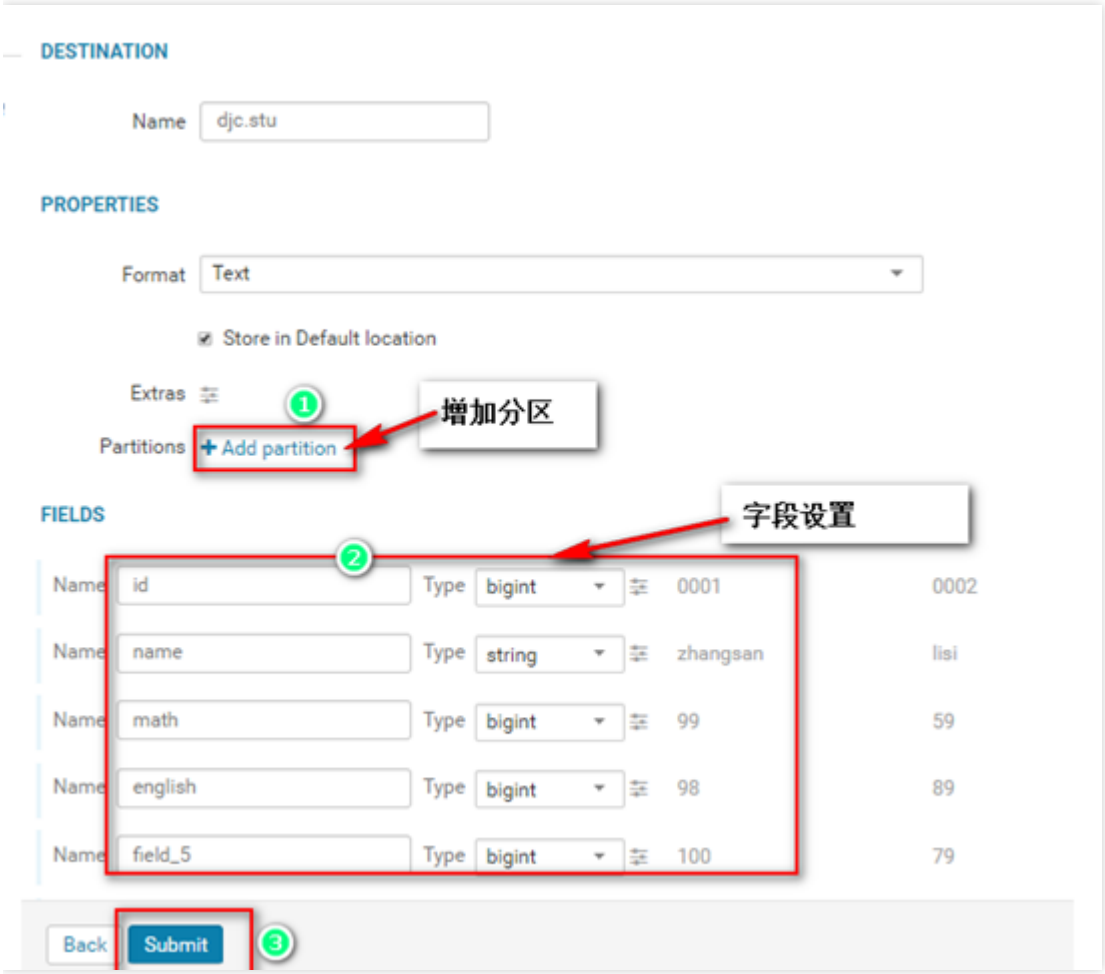
最近更新时间: 2019-11-13 02:42:51

点击Tables后加号标志，在SOURCE后导入文档（下列中导入的为txt文档），单击Next。您也可以直接输入SQL语句进行表创建，如

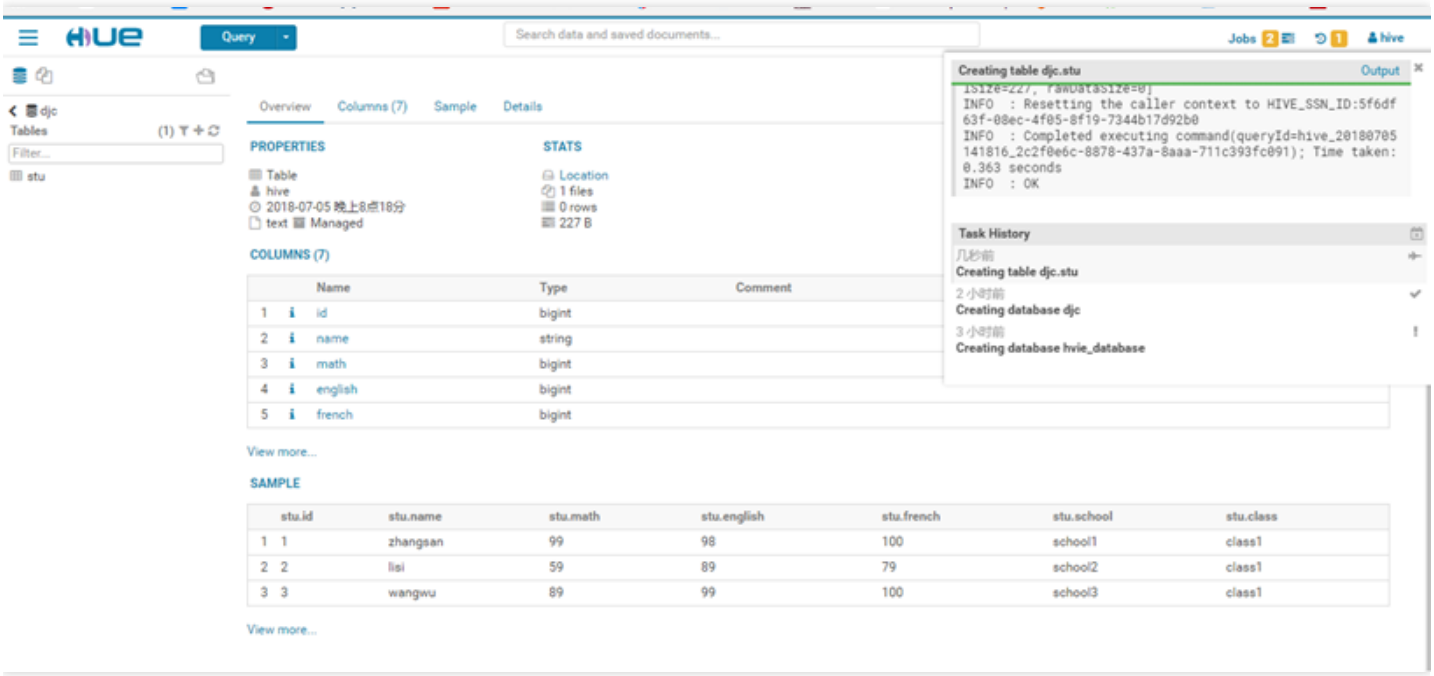
```
CREATE TABLE IF NOT EXISTS user_07 (
user_id string ,
seller_id string ,
product_id string ,
time string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' ;
```



导入文档内容如下 0001 zhangsan 99 98 100 school1 class1 0002 lisi 59 89 79 school1 class1 0003 wangwu 89 99 100 school3 class1 0004 zhangsan2 99 98 100 school1 class1 0005 lisi2 59 89 79 school2 class1 0006 wangwu2 89 99 100 school3 class1 对字段进行设置，可选择其类型（默认已配好对应类型），也可新增加分区，最后单击Submit。



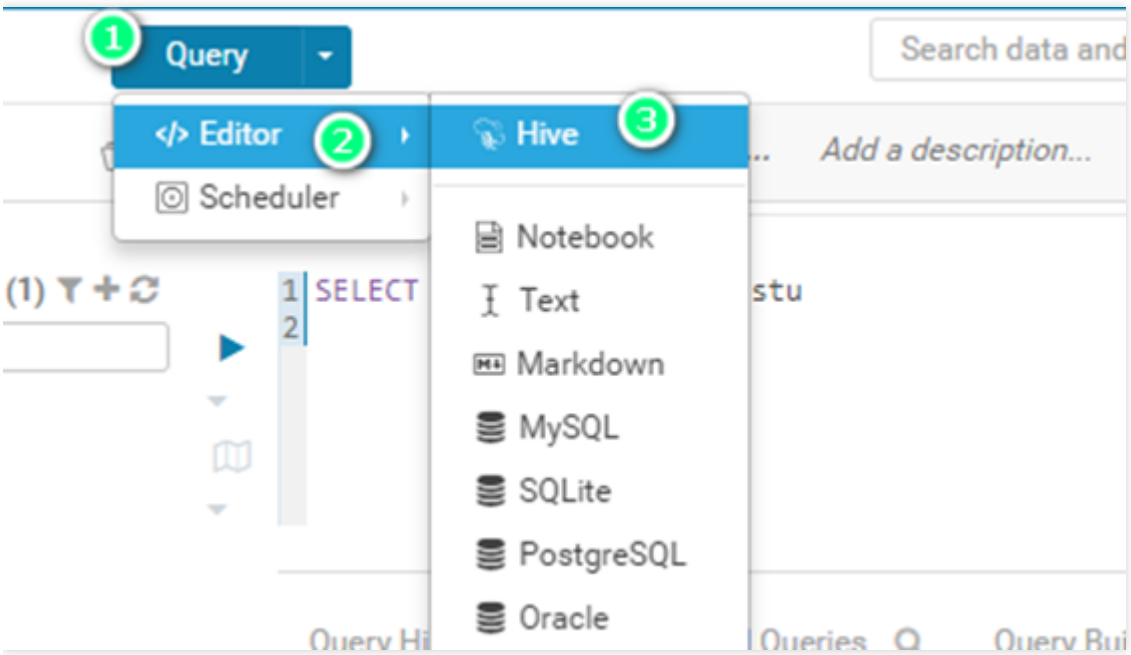
表创建完成后页面如下，可看到表结构，表状态等相应信息，在右上侧output中可以看到历史作业及其状态信息，日志信息也可在上方栏目中查看。


HUE
Query
Search data and saved documents...
Jobs 2 1 hive
djc
Tables (1)
Filter...
stu
Overview Columns (7) Sample Details
PROPERTIES
Table
hive
2018-07-05 晚上8点18分
text Managed
STATS
Location
1 files
0 rows
227 B
COLUMNS (7)
Name Type Comment
1 id bigint
2 name string
3 math bigint
4 english bigint
5 french bigint
View more...
SAMPLE
stu.id stu.name stu.math stu.english stu.french stu.school stu.class
1 1 zhangsan 99 98 100 school1 class1
2 2 lisi 59 89 79 school2 class1
3 3 wangwu 89 99 100 school3 class1
View more...
Creating table djc.stu
Output
INFO : Resetting the caller context to HIVE_SSN_ID:5f6df63f-08ec-4f05-8f19-7344b17d92b0
INFO : Completed executing command(queryId=hive_20180705141816_2c2f0e6c-8878-437a-8aaa-711c393fc091); Time taken: 0.363 seconds
INFO : OK
Task History
几秒前
Creating table djc.stu
2 小时前
Creating database djc
3 小时前
Creating database hvie_database

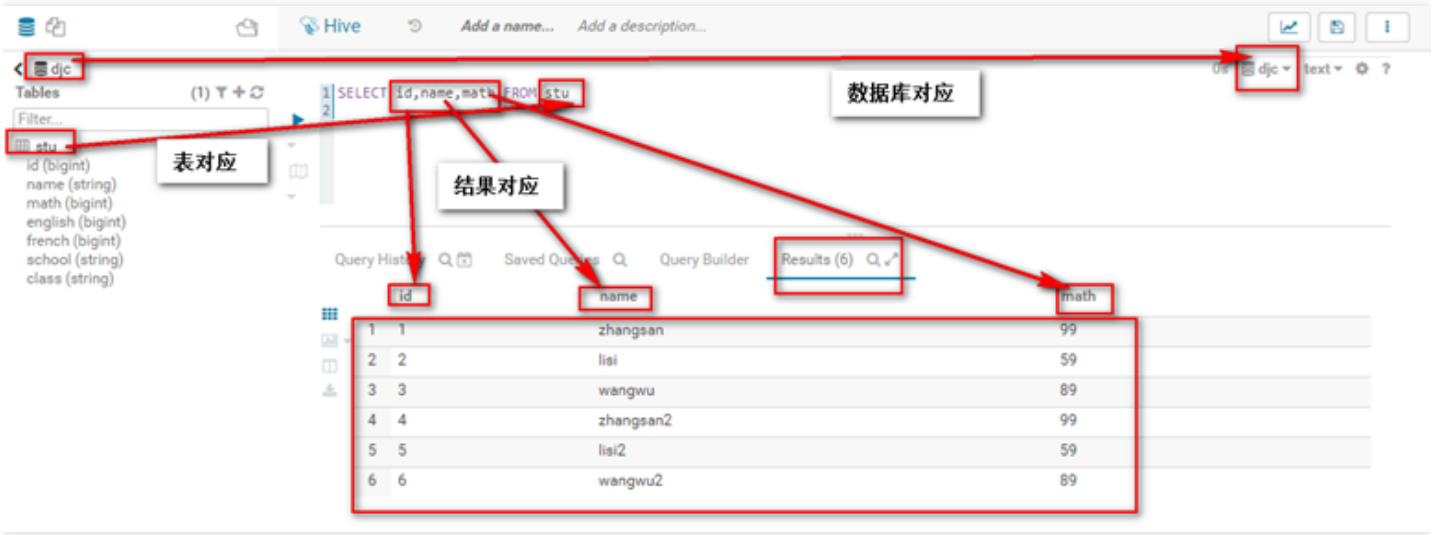
查询

最近更新时间: 2019-11-13 02:42:51

单击Query下拉菜单，依次选择Editor – – >Hive



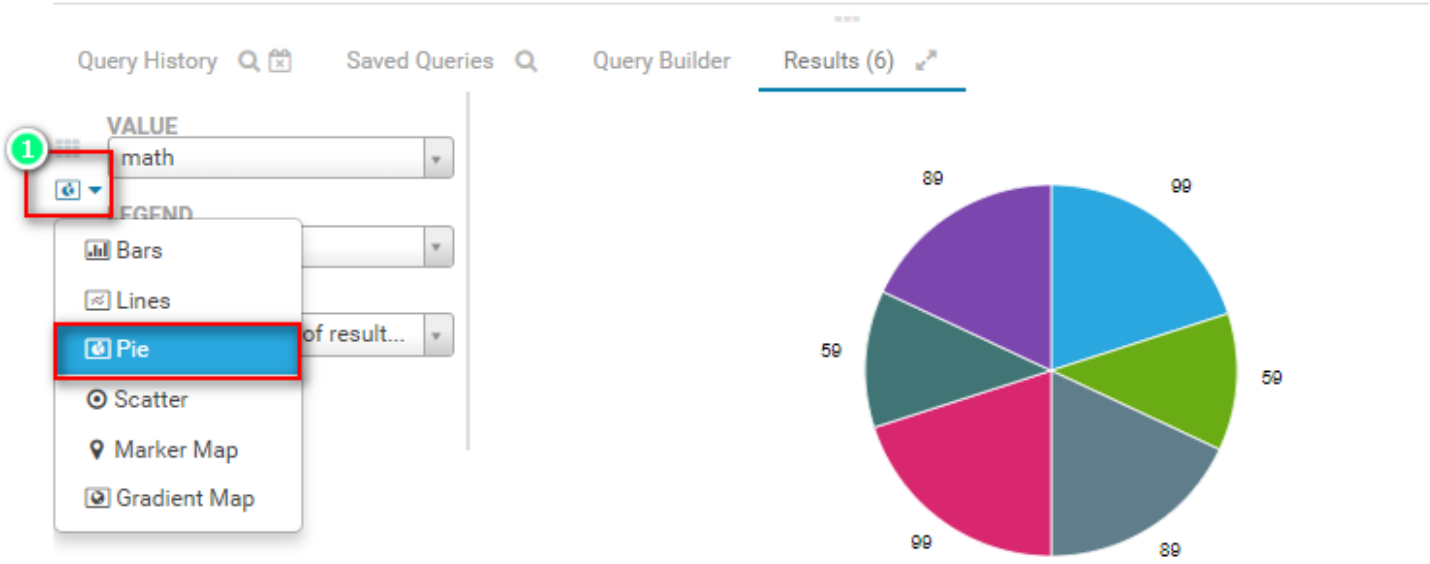
在输入框内输入相应SQL语句，进行数据库搜索，注意数据库（djc），表（stu）一一对应。在下方Results中可以看到搜索结果。



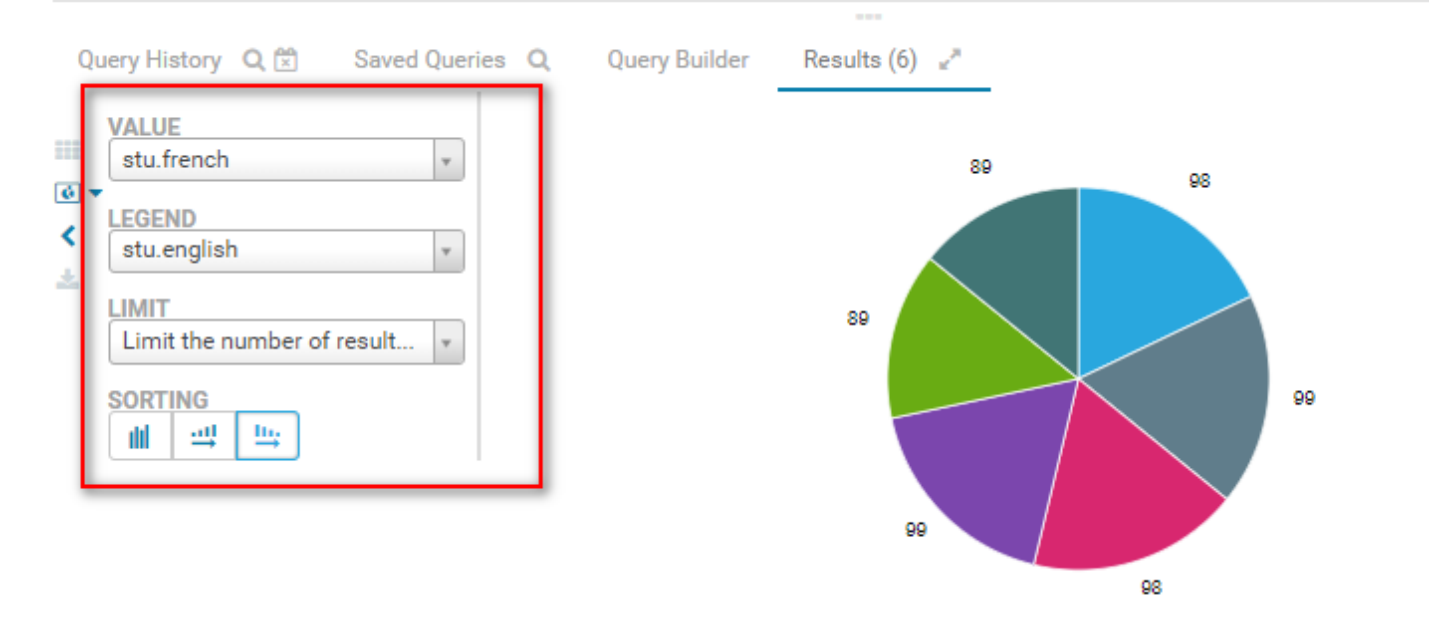
结果可视化

最近更新时间: 2019-11-13 02:55:28

在下图中单击①处，可选择展示图类型，此例中选择Pie饼状图，其结果在右方显示



也可以选择不同值对结果进行展示



您也可以在输入框中输入语句执行其他操作。

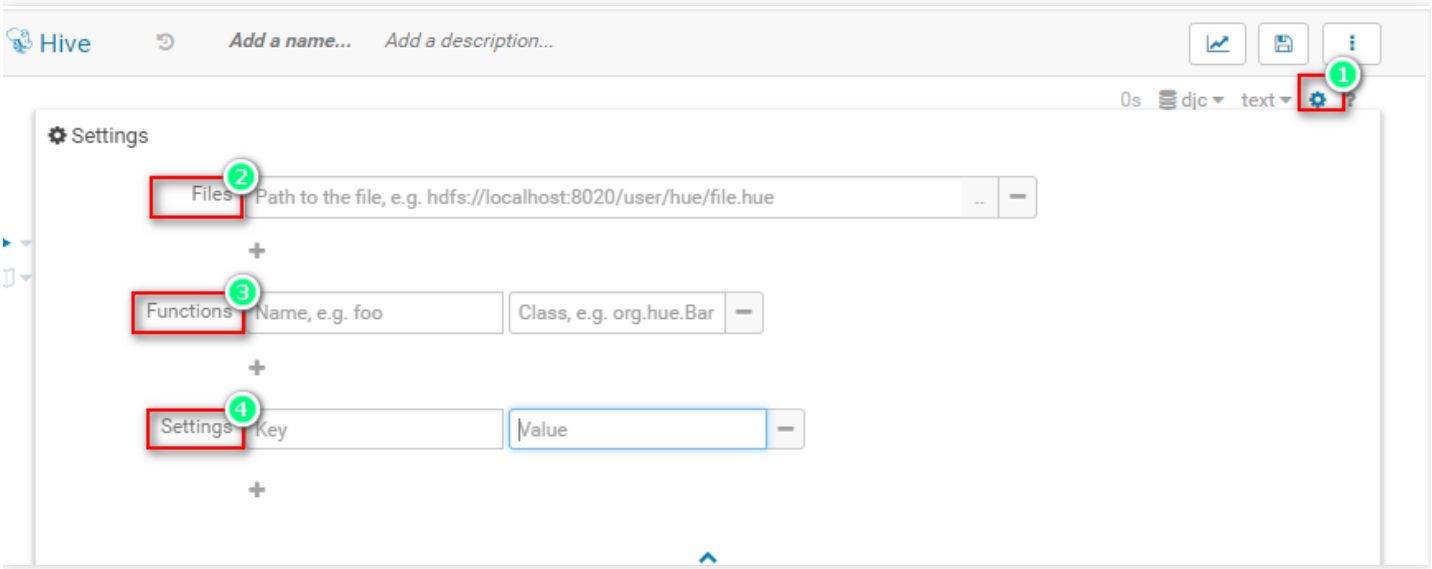
其他

最近更新时间: 2019-10-30 02:28:36

修改Hue中“Query Editors”配置

最近更新时间: 2019-11-13 03:00:47

单击①处 按钮； 在②处“Files”的右侧单击 指定该文件的存储目录。可以单击 新增加一个文件资源。 在③处“Functiions”的右侧可输入用户自定义的名称和函数的类名称。单击 新增加一个自定义函数。 在④处“Settings”的右侧中“Key”处输入Hive的参数名，在Value”输入对应的参数值，则当前Hive会话会以用户定义的配置连接Hive。可单击 新增加一个参数。



查看操作历史：在查询框下方单击“Query HIstory”可查看操作记录 **保存操作**：在输入命令后，点击上方save按钮，可对所执行操作进行保存 **查看所保存操作**：在查询框下方单击“Saved Queries”可查看已保存的历史操作

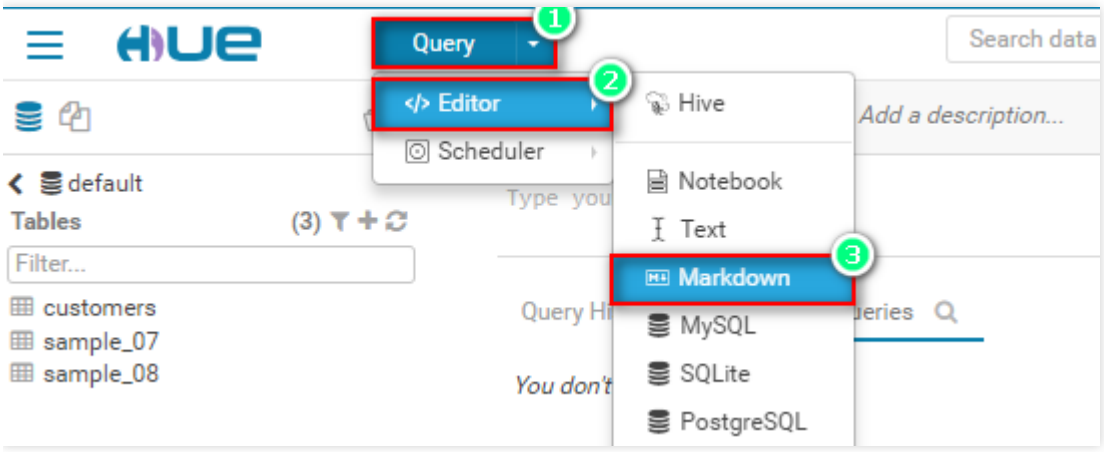
Markdown功能使用说明

最近更新时间: 2019-11-26 15:29:57

Markdown简介

最近更新时间: 2019-11-13 03:00:47

markdown是一种纯文本格式的标记语言。通过简单的标记语法，它可以使普通文本内容具有一定的格式。 依次单击Query-->Editor-->Markdown进入Markdown编辑页面；



输入内容及效果如下：

Markdown语法简介

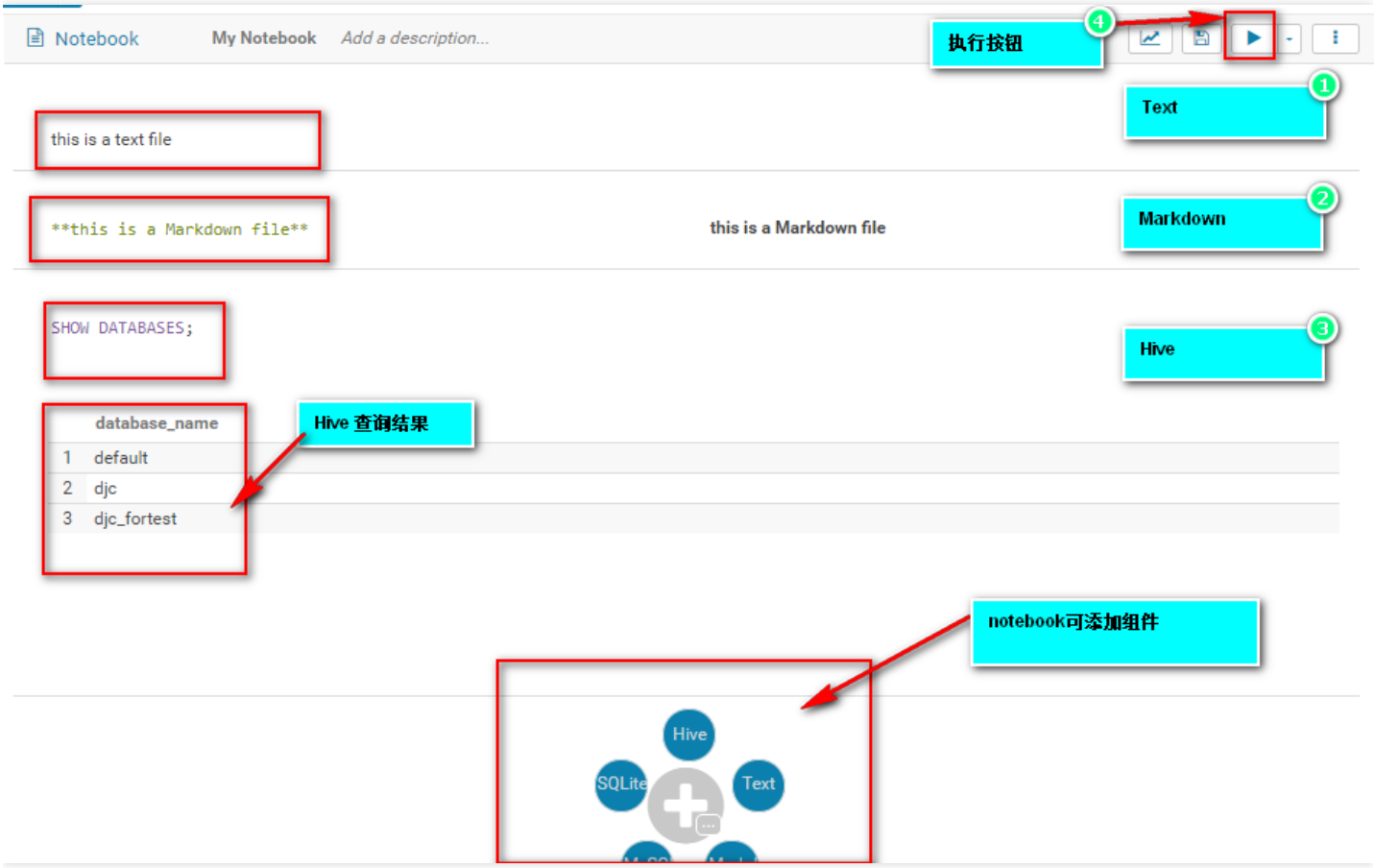
最近更新时间: 2019-10-30 02:23:33

- 标题 在想要设置为标题的文字前面加#来表示 一个#是一级标题，二个#是二级标题，以此类推。支持六级标题。
- 字体 加粗要加粗的文字左右分别用两个*号包起来 *斜体要倾斜的文字左右分别用一个*号包起来 斜体加粗要倾斜和加粗的文字左右分别用三个*号包起来 删除线要加删除线的文字左右分别用两个~~号包起来 超链接 超链接名 title可加可不加
- 表格 语法： 表头|表头|表头 ----|-----|----- 内容|内容|内容内容|内容|内容 第二行分割表头和内容。 - 有一个就行，为了对齐，多加了几个文字默认居左，-两边加：表示文字居中，右边加：表示文字居右 **注**：原生的语法两边都要用 | 包起来。

Notebook操作

最近更新时间: 2019-11-13 03:15:33

在notebook中可以添加Text、Markdown、Hive、MySQL、SQLite，其可以在同一页面中进行执行 点击加号，添加对应任务，输入命令后其可在同一页面中进行展示 下图分别为Text、Markdown及Hive操作结果



MapReduce作业

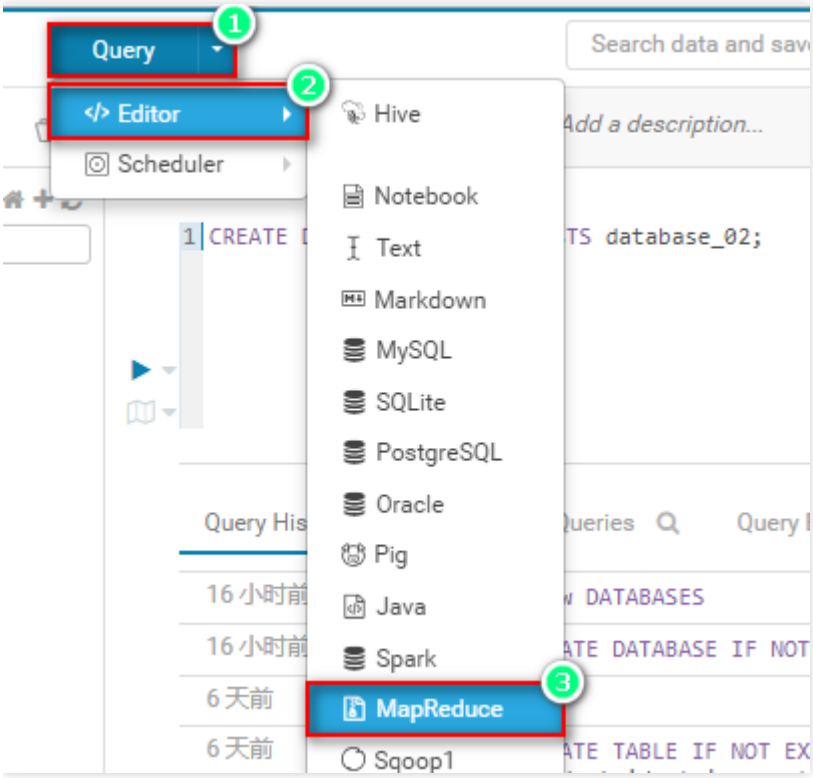
最近更新时间: 2019-10-30 02:15:43

在hue中提交mapreduce作业非常简单。

新建Mapreduce Action

最近更新时间: 2019-11-13 06:16:46

在页面依次选择Query-->Editor-->MapReduce

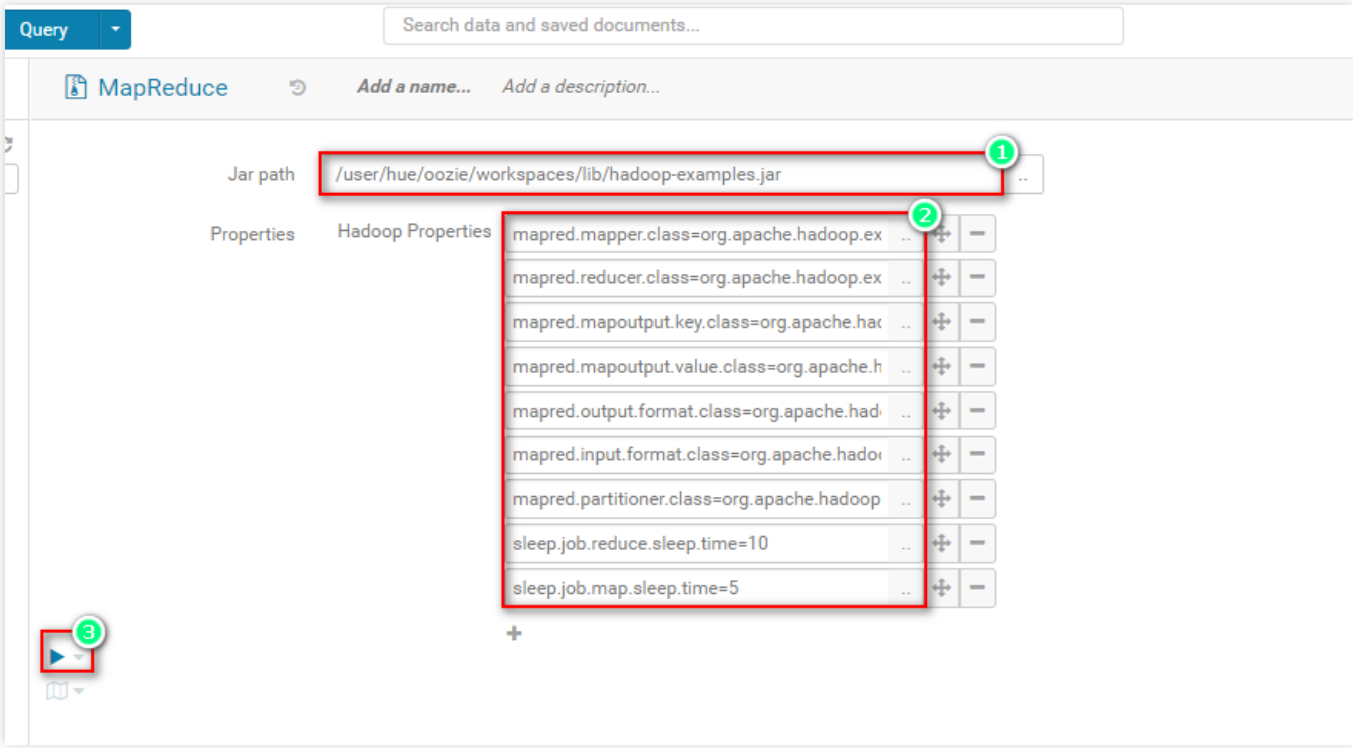


作业配置

最近更新时间: 2019-11-13 06:16:46

以执行以mapreduce sleep作业为例

1. 在①处填写jar包的HDFS路径，路径任意 2) 在②处对所有mapreduce作业的配置进行填写，此作业配置内容见下表 3) 点击提交按钮，提交作业



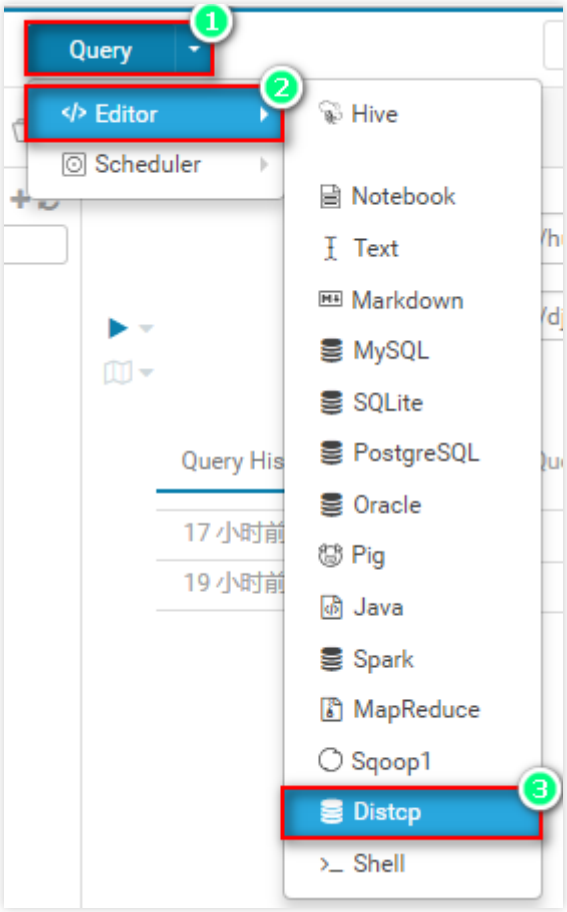
mapreduce作业配置内容：

```
mapred.mapper.class=org.apache.Hadoop.examples.SleepJob
mapred.reducer.class=org.apache.Hadoop.examples.SleepJob
mapred.mapoutput.key.class=org.apache.Hadoop.io.IntWritable
mapred.mapoutput.value.class=org.apache.Hadoop.io.NullWritable
mapred.output.format.class=org.apache.Hadoop.mapred.lib.NullOutputFormat
mapred.input.format.class=org.apache.Hadoop.examples.SleepJob$SleepInputFormat
mapred.partitioner.class=org.apache.Hadoop.examples.SleepJob
sleep.job.map.sleep.time=5
sleep.job.reduce.sleep.time=10
```

DistCp作业

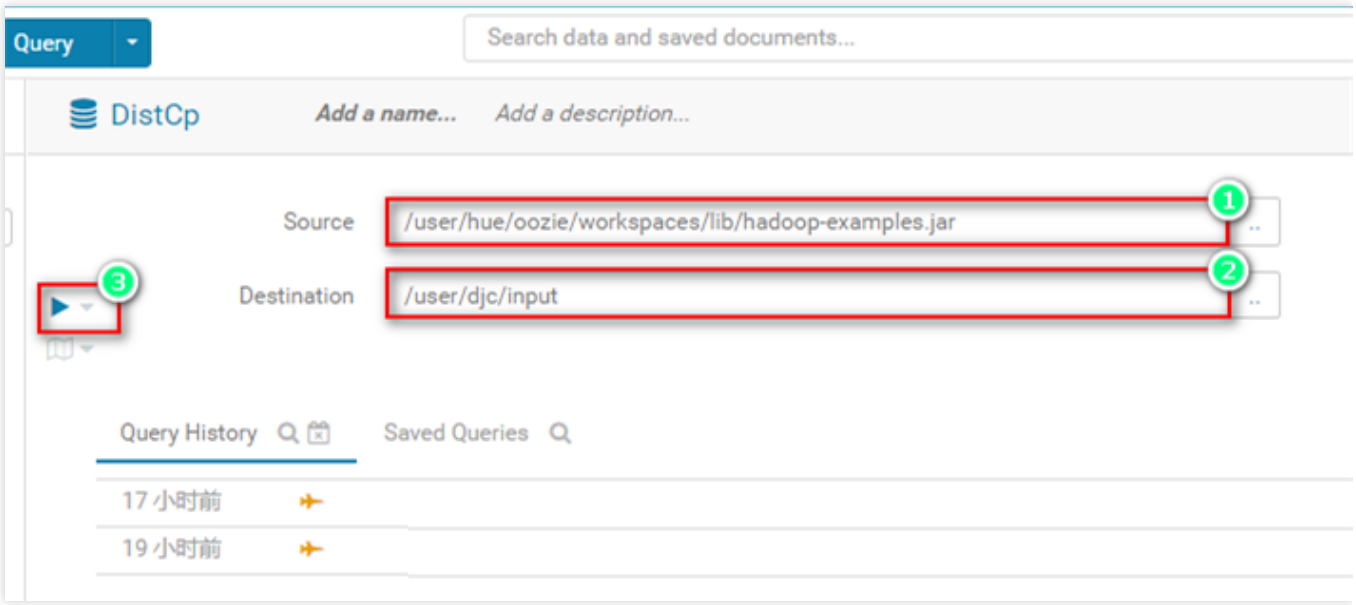
最近更新时间: 2019-11-13 06:16:46

使用Distcp可很方便的在HDFS中进行文件的传输



1. 依次单击Query-->Editor-->Distcp

2. 在下页面中分别配置号源文件（①处）和目标文件夹（②处），单击③处提交按钮进行作业提交



Kibana使用指南

最近更新时间: 2019-11-26 15:29:57

Kibana简介

最近更新时间: 2019-11-26 15:29:57

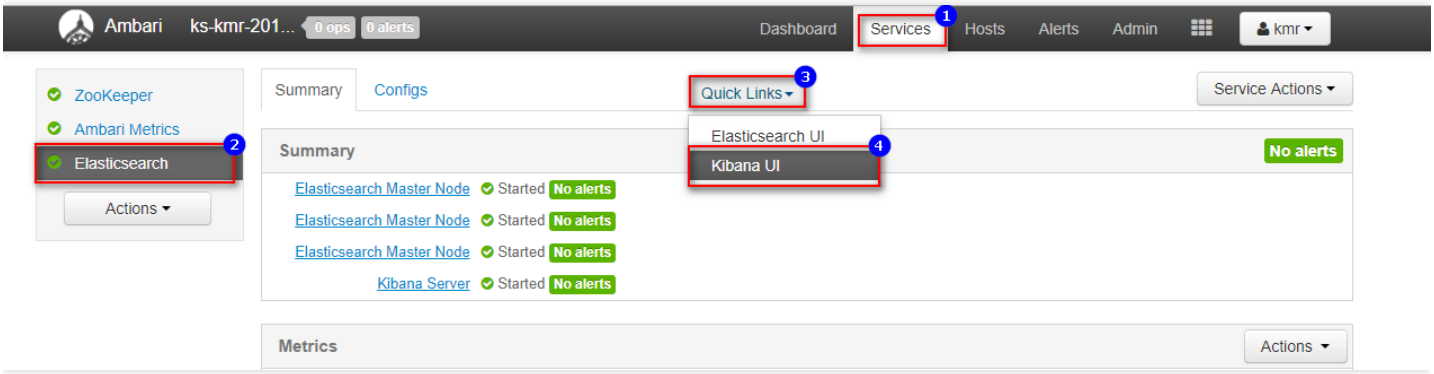
- Kibana是一个用于分析和可视化Elasticsearch数据开源平台，使用Kibana能够搜索、展示存储在Elasticsearch中的索引数据。使用它可以很方便用图表、表格、地图展示和分析数据。

- Kibana能够轻松处理大量数据，通过浏览器接口能够轻松的创建和分享仪表盘，通过改变Elasticsearch查询时间，可以完成动态仪表盘。

进入Kibana

最近更新时间: 2019-11-13 06:16:46

在托管ES集群的控制台中依次点击Service-- >Elasticsearch -- >Quick Links-- >Kibana UI 进入到Kibana Web页面



添加索引

最近更新时间: 2021-09-15 16:33:18

要使用kibana，须配置至少一个索引模式（index pattern）。索引模式用来识别Elasticsearch中的索引以便于查询分析，也可用来配置字段。

定义索引mapping

索引名称为my_books，索引类型为products。

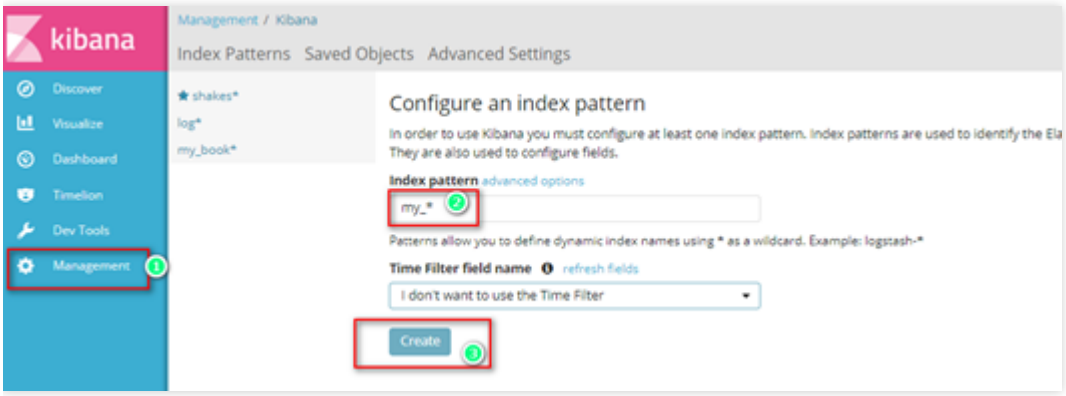
在Dev Tools中的Console进行定义，输入后单击PUT /my_books一行后的绿色三角形即可完成创建，定义及反馈如图

添加文档

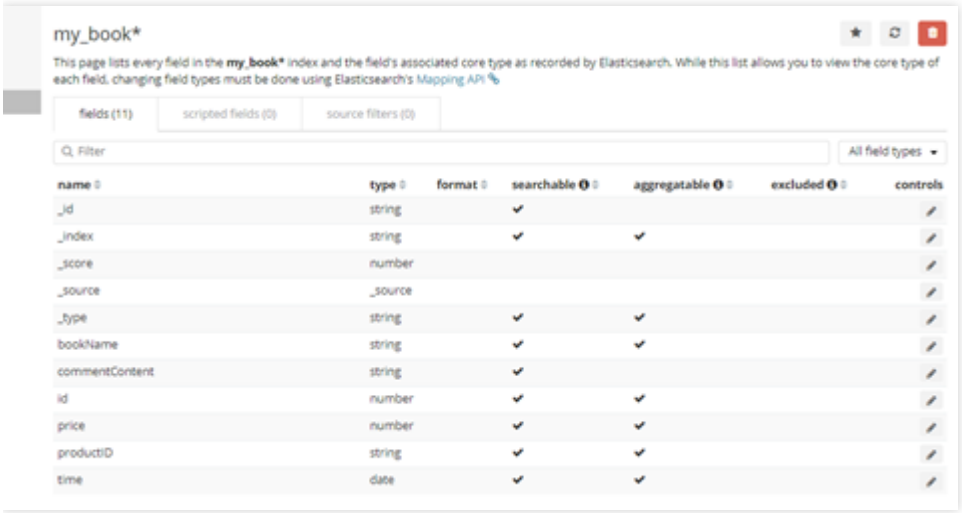
配置索引

最近更新时间: 2021-09-15 16:33:18

在kibana首页点击Management，然后点击Index Patterns对其进行创建，在Index Pattern中输入索引名称。 注：在定义索引模式时，在elasticsearch中必须存在相应mapping（即前所定义的my_store），并且在Elasticsearch中必须有数据(在my_store中插入数据)；可在Dev Tools的Console台中输入GET _ cat/indices进行索引查询。在Index Pattern名称中可输入索引部分字段，后以 * 作为通配符。



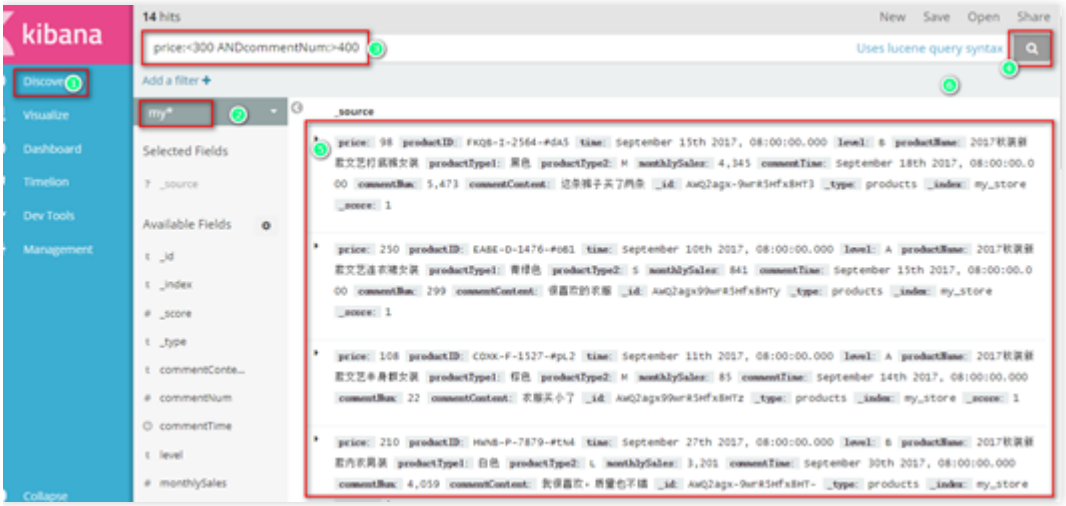
配置好索引后，可在management中Index Patterns对单击索引名称其进行查看。



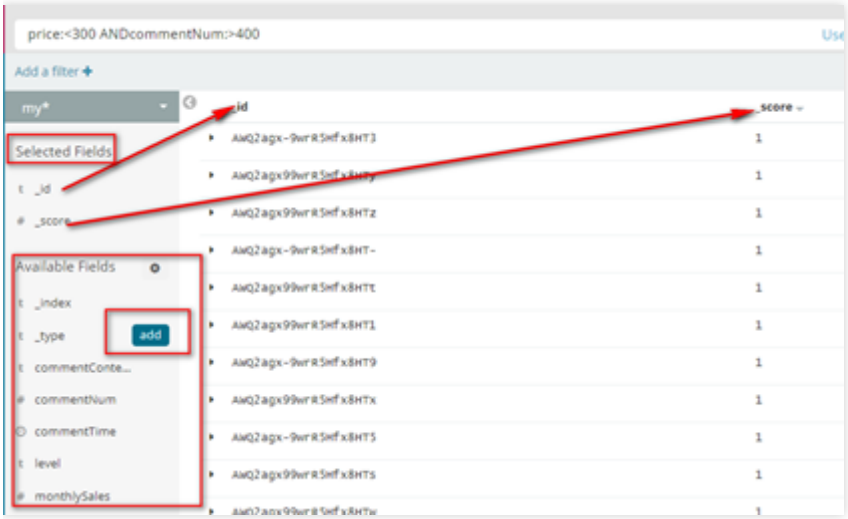
Discover/数据查询

最近更新时间: 2021-09-15 16:33:18

在discover应用中，您可以输入Elasticsearch查询来搜索data并对结果进行过滤。 单击Discover，在Add a filter+下方选择相应所配置好的索引模式，则在界面中回展示出相应数据。

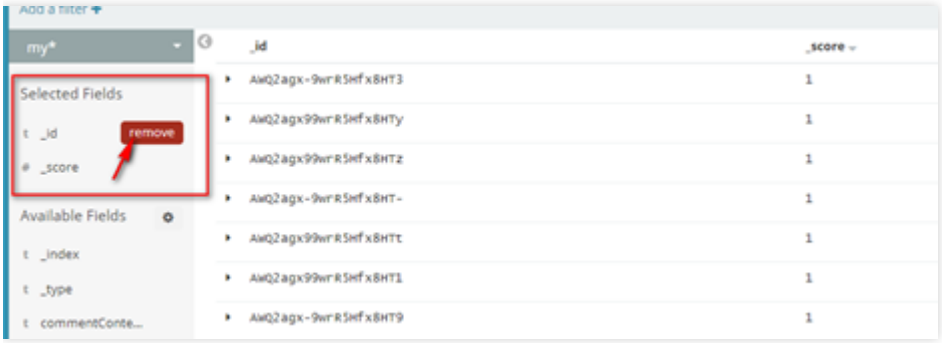


您也可以在Available Fields中选择自己需要展示出的数据；将鼠标停放在对应字段旁，其出现Add按钮，点击添加。



要移除字段，您可以将鼠标停放在Selected Fields下对应字段旁，其出现remove按钮，单击移除此字段在页面中的

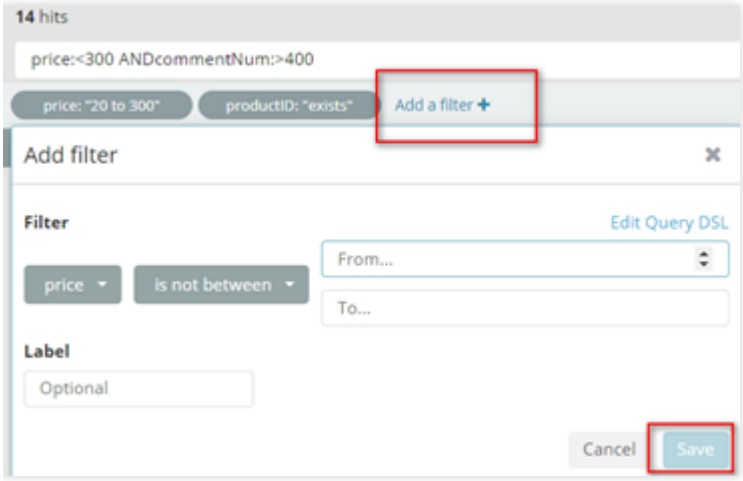
显示。


Add a filter
my*
Selected Fields
t _id
remove
_score
Available Fields
t _index
t _type
t commentConte...
_id
_score
AWQ2agx-9wrR5Hfx8HT3
AWQ2agx99wrR5Hfx8HTy
AWQ2agx99wrR5Hfx8HTz
AWQ2agx-9wrR5Hfx8HT-
AWQ2agx99wrR5Hfx8HTt
AWQ2agx99wrR5Hfx8HT1
AWQ2agx-9wrR5Hfx8HT9
1

字段过滤

最近更新时间: 2021-09-15 16:33:18

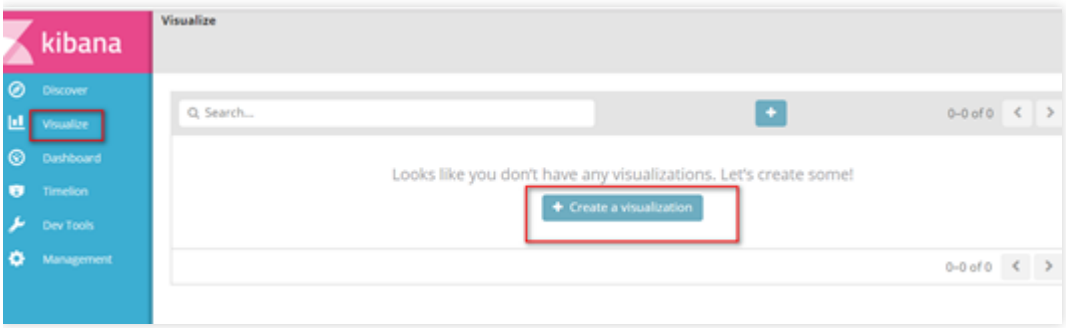
您也可以选择特定的字段进行查看，单击Add a filter+，选择过滤字段，设置相应条件，则可筛选出合适条件的数据。



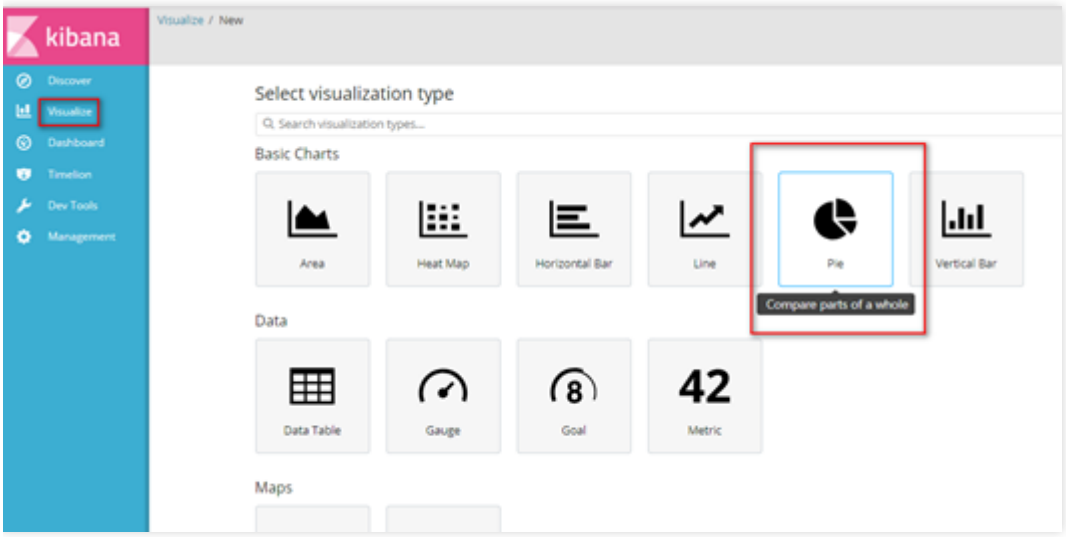
可视化/Visualize

最近更新时间: 2021-09-15 16:33:18

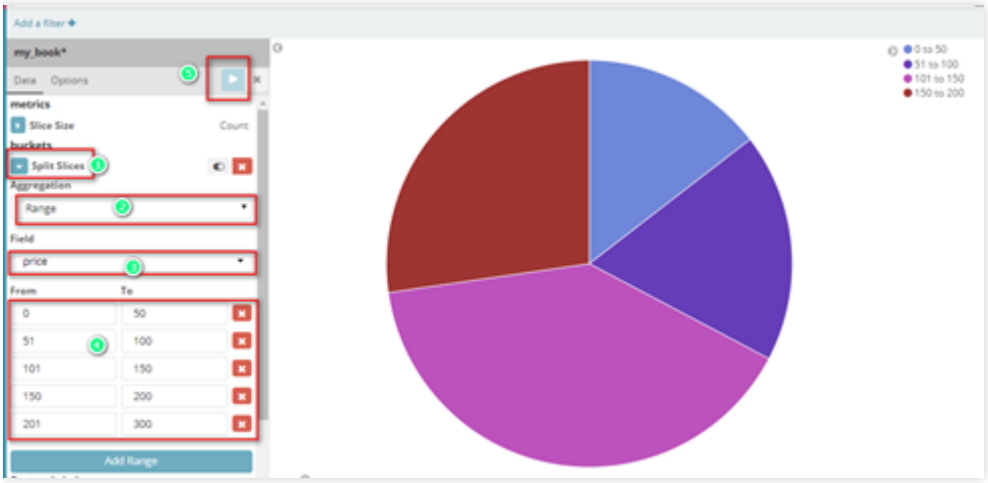
在可视化视图中，您可以通过图、表等形式对数据进行查看。单击Visualize，在右侧单击+create a visualization进行可视化格式的添加。



选择饼状图



选择相应索引，在Aggregation中选择Range，Field中选择price，在range中填写相应区间进行分析，最终结果如下:

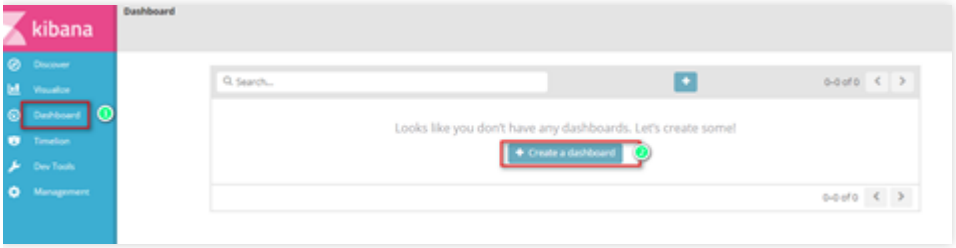


建立完成后，您可以将其保存（点击上方save按钮，输入名称后，再次点击save可对其进行保存），以便后续后序在Dashboard中进行操作。

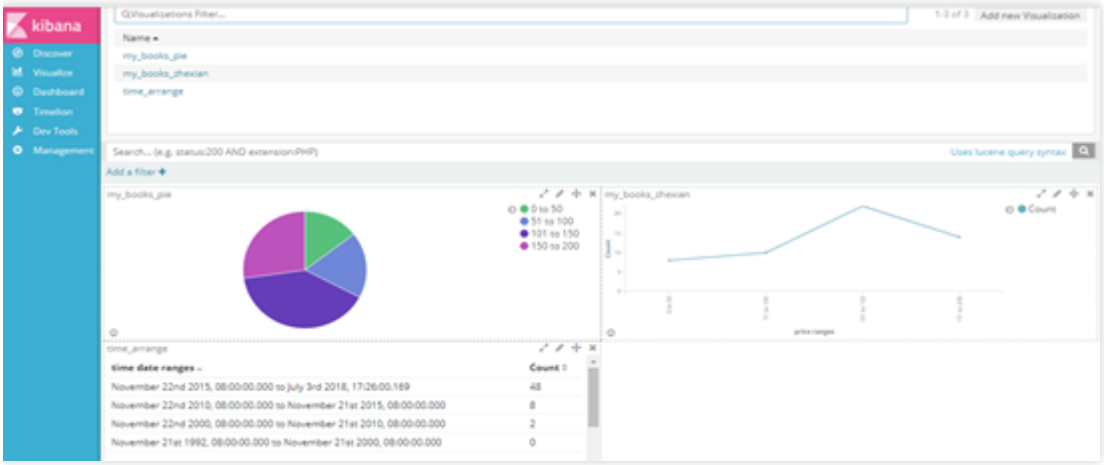
仪表盘分析/Dashboard

最近更新时间: 2021-09-15 16:33:18

在Visualize中我们建立好图表之后，可在Dashboard中进行对比分析。更多visualize图表可以参考Kibana官网：https://www.elastic.co/guide/en/kibana/current/visualize.html 单击左侧中Dashboard，在右侧点击Create a dashboard.



以此添加在visualize中建立好的图表，可在仪表盘中进行对比分析：



ES基于时间索引

最近更新时间: 2019-10-30 01:53:58

在日志分析的场景下，通常不需要保存所有的日志数据，而只需保留近期一段时间的日志数据，超过一定时间后对应的数据可以删除掉，这就涉及到基于时间索引的技巧。

索引名中增加日期后缀

最近更新时间: 2019-10-30 01:53:04

在logstash中可以在conf中设置为索引增加日期后缀

```
elasticsearch {
hosts => "host:9200"
index => "log-%{+YYYY-MM-dd}"
}
```

定时删除过期的文件

最近更新时间: 2019-10-30 01:51:29

使用ES的Delete by Query接口来删除过期的文件。 /usr/bin/curl -H'Content-Type:application/json' -d'{"query":{"range":{"@timestamp":{"lt":"now-${DELTA}d","format":"epoch_millis"}}}}' -XPOST "http://${SERVER_PORT}/${INDEX_PRFIX}*/_delete_by_query?pretty" ${DELTA}：为具体的天数，例如7，表示删除7天前的日志 ${SERVER_PORT}：为ES的服务地址+端口 ${INDEX_PRFIX}：为索引的前缀 将此命令添加到Crontab中即可按照创建日期定时删除过期的索引。

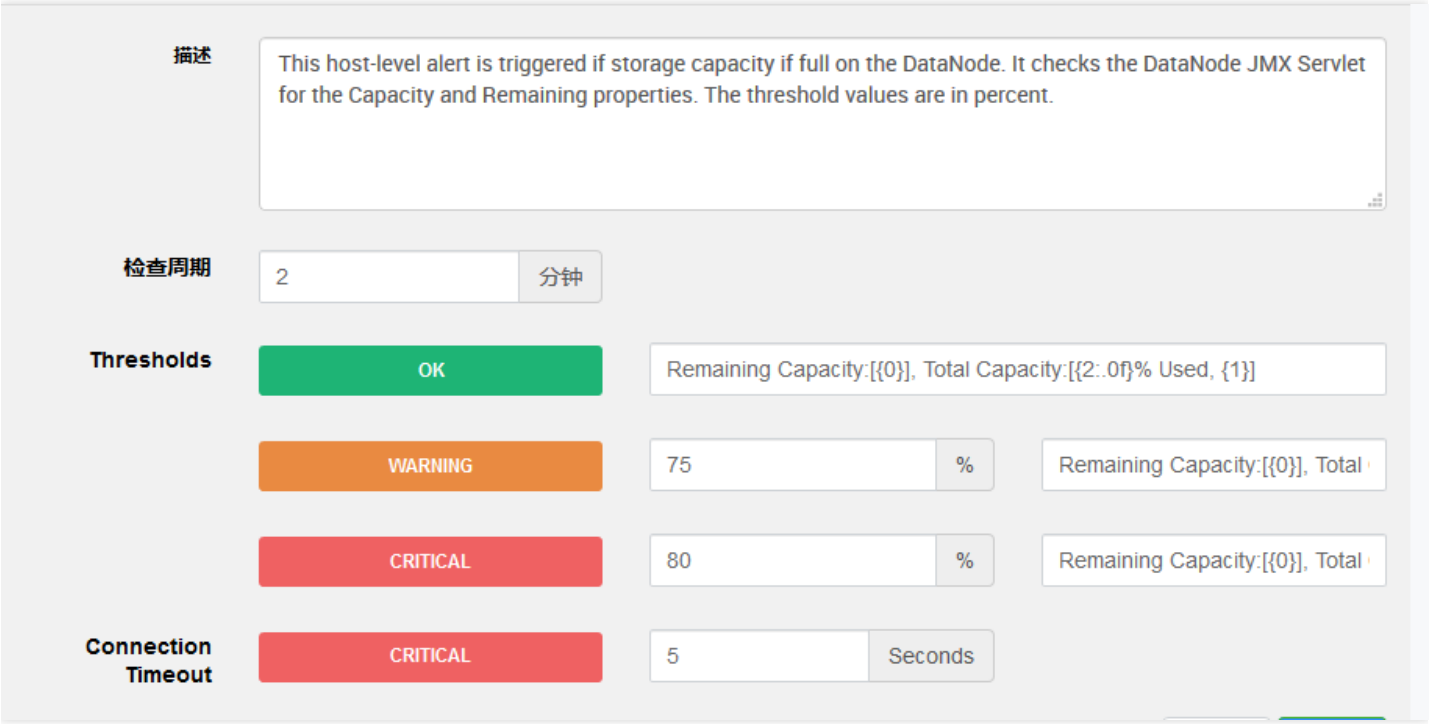
运维操作

最近更新时间: 2019-11-26 15:29:57

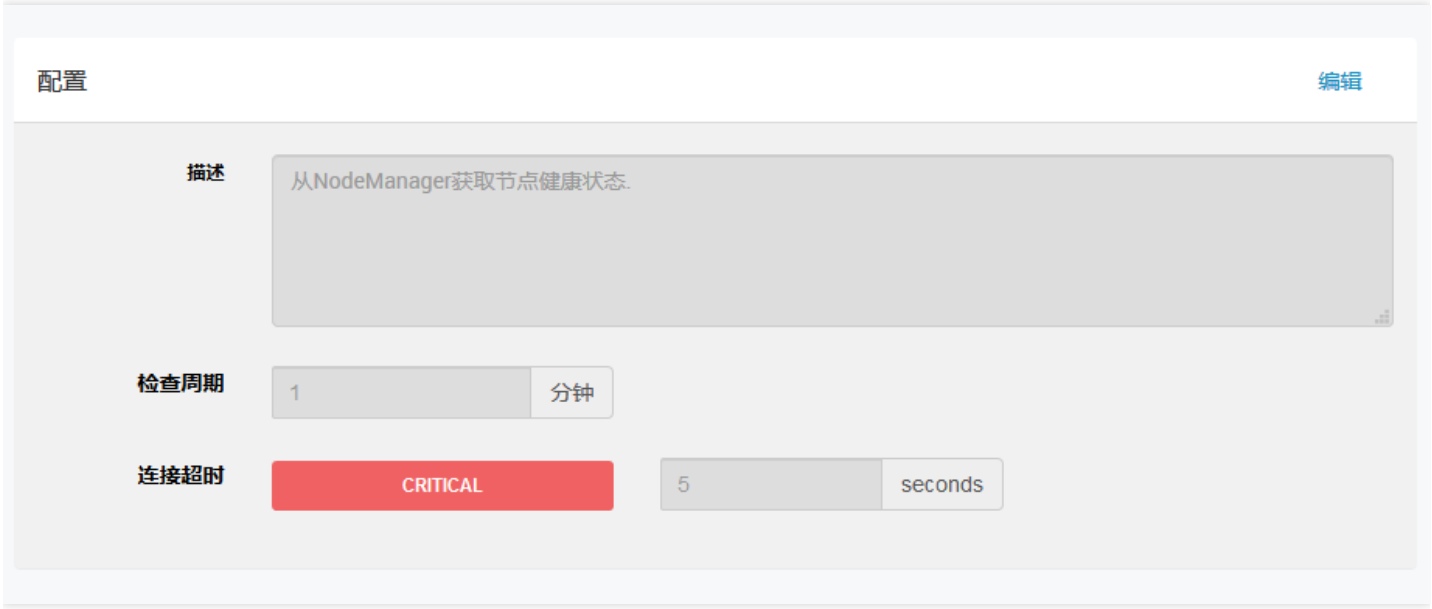
告警处理

最近更新时间: 2019-11-15 07:30:14

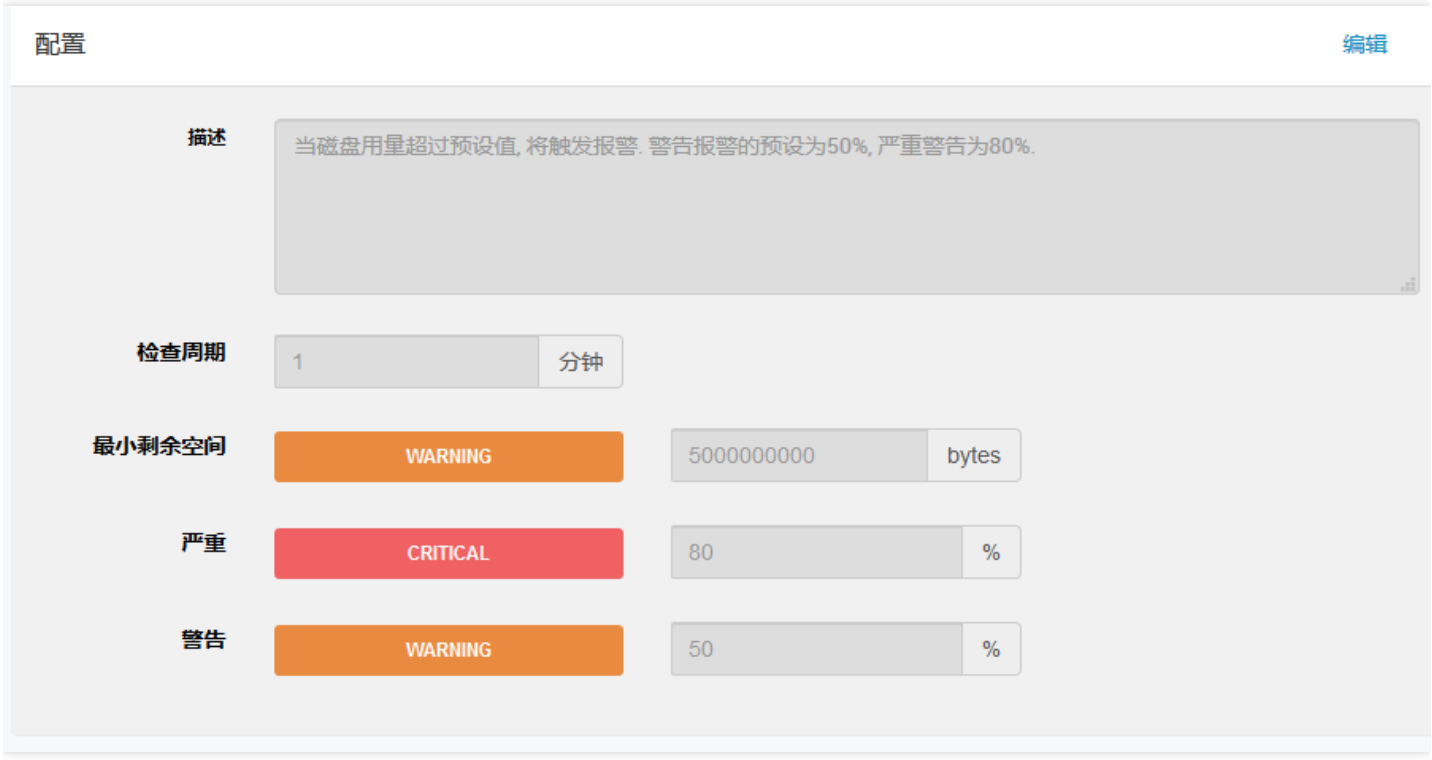
DataNode Storage存储 如果数据节点上的存储容量已满，则会触发此主机级警报。它检查datanode jmx servlet的容量和剩余属性。阈值以百分比为单位。 This host-level alert is triggered if storage capacity if full on the DataNode. It checks the DataNode JMX Servlet for the Capacity and Remaining properties. The threshold values are in percent.



当使用存储达到75%以上，会警告；使用存储达到80%以上严重警告。 从NodeManager获取节点健康状态. 从NodeManager获取节点健康状态.

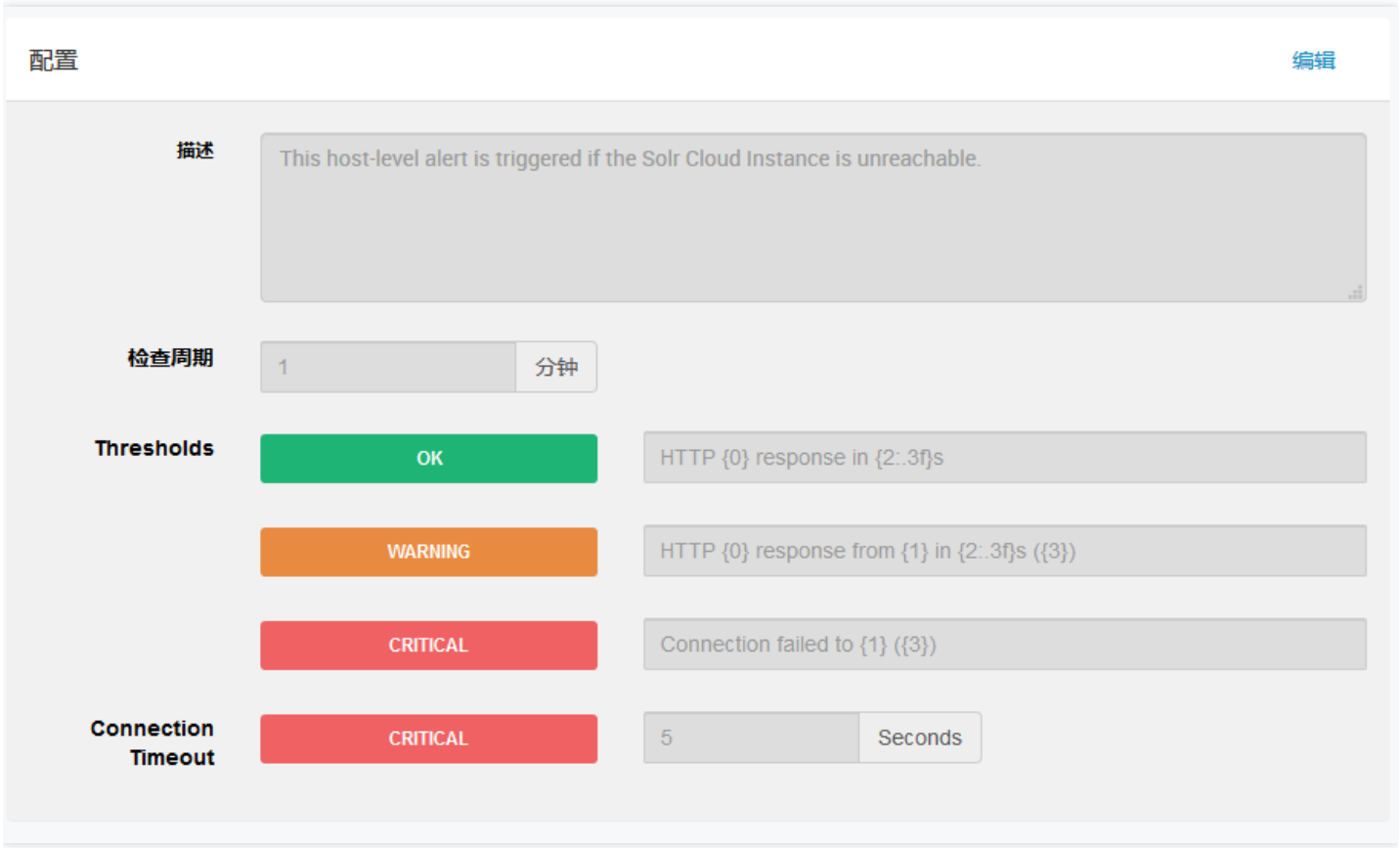


每1分钟检查一下，超过5分钟没有响应就是严重警告。 主机磁盘使用率 当磁盘用量超过预设值, 将触发报警. 警告报警的预设为50%, 严重警告为80%.



有可用空间的数据节点百分比 当停止的DataNode 可用存储空间百分比达到阈值时, 触发告警.

存储空间达到10%时，警告；达到30%时，严重警告 Infra Solr Web用户界面 This host-level alert is triggered if the Solr Cloud Instance is unreachable. 如果无法访问solr云实例，则会触发此主机级警报。



Kafka Broker Process 描述: ”当无法确定Kafka Broker 运行状态时, 将触发告警. 单位:秒.”。 Knox Gateway Process 告警名称: ”Knox Gateway Process”。描述: ”当无法确定Knox Gateway 运行状态时, 将触发告警. 单位:秒.”。 Kylin Master Web UI 告警名称: ”Kylin Master Web UI”。描述: ”如果无法访问Kylin Master Web UI，则会触发此主机级告警。 Kylin QUERY Web UI 告警名称: ”Kylin QUERY Web UI”。描述: ”如果无法访问Kylin QUERY Web UI，则会触发此主机级告警。”。 Percent RegionServers Available 告警名称: ”Percent RegionServers Available”。描述: ”当可用的RegionServer 服务百分比达到阈值时, 触发告警. 该结果聚合RegionServer 的所有检测结果.”。 HBase Master Process 告警名称: ”HBase Master Process”。描述: ”当无法确定HBase Master运行状态时, 将触发告警. 单位:秒”。 HBase Master CPU Utilization 告警名称: ”HBase Master CPU Utilization”。描述: ”当HBase Master CPU 使用率达到阈值时, 触发告警, 通过连接 HBase Master JMX Servlet获取SystemCPULoad 属性. 阈值设定为百分比.”。 HBase RegionServer Process 告警名称: ”HBase RegionServer Process”。描述: ”当无法确定HBase RegionServer 运行状态时, 将触发告警. 单位:秒”。 Percent Metrics Monitors Available 告警名称: ”Percent Metrics Monitors Available”。描述: ”当可用的Metrics Monitor 服务百分比达到阈值时, 触发告警”。 Metrics Collector – Auto-Restart Status 告警名称: ”Metrics Collector – Auto-Restart Status”。描述: ”Metrics Collector 在一小时内自动重启的频率过高, 将触发告警. 默认重启 2次(警告), 4次及以上(严重)”。 Metrics Collector Process 告警名称: ”Metrics Collector Process”。描述: ”当无法确定Metrics Collector运行状态时, 将触发告警. 单位:秒”。 Metrics Collector – HBase Master Process 告警名称: ”Metrics Collector – HBase Master Process”。描述: ”当无法确定HBase Master运行状态时, 将触发告警. 单位:秒”。 Metrics Collector – HBase Master CPU Utilization 告警名称: ”Metrics Collector – HBase Master CPU Utilization”。描述: ”当Metrics Collector HBase Master 服务的 CPU 使用率达到阈值时, 触发告警.

通过连接 HBase Master JMX Servlet获取SystemCPULoad 属性. 阈值设定为百分比”。 Metrics Monitor Status 告警名称: ”Metrics Monitor Status”。描述: ”monitor状态脚本检测的结果”。 Grafana Web UI 告警名称: ”Grafana Web UI”。描述: ”当无法确定Grafana Web UI运行状态时, 触发告警”。 Percent DataNodes Available 告警名称: ”Percent DataNodes Available”。描述: ”当停止的DataNode 百分比达到阈值时, 触发告警. 该结果聚合DataNode 的所有检测结果”。 Percent DataNodes With Available Space 告警名称: ”Percent DataNodes With Available Space”。描述: ”当停止的DataNode 可用存储空间百分比达到阈值时, 触发告警”。 Percent JournalNodes Available 告警名称: ”Percent JournalNodes Available”。描述: ”当停止的JournalNode 百分比达到阈值时, 触发告警. 该结果聚合JournalNode 的所有检测结果”。 NameNode Web UI 告警名称: ”NameNode Web UI”。描述: ”当无法确定NameNode Web UI运行状态时, 触发告警”。 HDFS Upgrade Finalized State 告警名称: ”HDFS Upgrade Finalized State”。描述: ”This service-level alert is triggered if HDFS is not in the finalized state”。 NameNode Host CPU Utilization 告警名称: ”NameNode Host CPU Utilization”。描述: ”NameNode CPU 使用率达到阈值时, 通过连接 NameNode JMX Servlet获取SystemCPULoad 属性, 触发告警, 阈值设定为百分比.”。 NameNode Blocks Health 告警名称: ”NameNode Blocks Health”。描述: ”当丢失的block数量达到阈值, 触发告警, 单位: block.”。 HDFS Pending Deletion Blocks 告警名称: ”HDFS Pending Deletion Blocks”。描述: ”等待删除的block数量达到阈值, 触发告警. 通过连接 NameNode JMX Servlet 的 PendingDeletionBlock 属性获取.”。 HDFS Capacity Utilization 告警名称: ”HDFS Capacity Utilization”。描述: ”当HDFS使用率达到阈值, 触发告警. 通过连接 NameNode JMX Servlet 的 CapacityUsed和CapacityRemaining 属性获取, 阈值设定为百分比.”。 NameNode RPC Latency 告警名称: ”NameNode RPC Latency”。描述: ”当 NameNode RPC 延迟达到阈值, 触发告警. 一般增加RPC队列长度时也要调整RPC 延迟时长. 单位: 毫秒(ms).”。 NameNode Directory Status 告警名称: ”NameNode Directory Status”。描述: ”当NameNode NameDirStatuses 指标 (name=NameNodeInfo/NameDirStatuses)有错误目录时, 触发告警. 阈值为错误目录的数量.”。 DataNode Health Summary 告警名称: ”DataNode Health Summary”。描述: ”当集群中有异常DataNode 时, 触发告警”。 NameNode Last Checkpoint 告警名称: ”NameNode Last Checkpoint”。描述: ”This service-level alert will trigger if the last time that the NameNode performed a checkpoint was too long ago. It will also trigger if the number of uncommitted transactions is beyond a certain threshold.”。 NameNode High Availability Health 告警名称: ”NameNode High Availability Health”。描述: ”This service-level alert is triggered if either the Active NameNode or Standby NameNode are not running.”。 NameNode Service RPC Queue Latency (Hourly) 告警名称: ”NameNode Service RPC Queue Latency (Hourly)”。描述: ”This service-level alert is triggered if the deviation of RPC queue latency on datanode port has grown beyond the specified threshold within an hour period.”7.1.36 NameNode Client RPC Queue Latency (Hourly)。 告警名称: ”NameNode Client RPC Queue Latency (Hourly)”。描述: ”This service-level alert is triggered if the deviation of RPC queue latency on client port has grown beyond the specified threshold within an hour period.”。 NameNode Service RPC Processing Latency (Hourly) 告警名称: ”NameNode Service RPC Processing Latency (Hourly)”。描述: ”This service-level alert is triggered if the deviation of RPC latency on datanode port has grown beyond the specified threshold within an hour period.”。 NameNode Client RPC Processing Latency (Hourly) 告警名称: ”NameNode Client RPC Processing Latency (Hourly)”。描述: ”This service-level alert is triggered if the deviation of RPC latency on client port has grown beyond the specified threshold within an hour period.”。 NameNode Heap Usage (Daily) 告警名称: ”NameNode Heap

Usage (Daily)”。描述: ”This service-level alert is triggered if the NameNode heap usage deviation has grown beyond the specified threshold within a day period.”。 NameNode Service RPC Processing Latency (Daily) 告警名称: ”NameNode Service RPC Processing Latency (Daily)”。描述: ”This service-level alert is triggered if the deviation of RPC latency on datanode port has grown beyond the specified threshold within a day period.”。 NameNode Client RPC Processing Latency (Daily) 告警名称: ”NameNode Client RPC Processing Latency (Daily)”。描述: ”This service-level alert is triggered if the deviation of RPC latency on client port has grown beyond the specified threshold within a day period.”。 NameNode Service RPC Queue Latency (Daily) 告警名称: ”NameNode Service RPC Queue Latency (Daily)”。描述: ”This service-level alert is triggered if the deviation of RPC latency on datanode port has grown beyond the specified threshold within a day period.”。 NameNode Client RPC Queue Latency (Daily) 告警名称: ”NameNode Client RPC Queue Latency (Daily)”。描述: ”This service-level alert is triggered if the deviation of RPC latency on client port has grown beyond the specified threshold within a day period.”。 HDFS Storage Capacity Usage (Daily) 告警名称: ”HDFS Storage Capacity Usage (Daily)”。描述: ”This service-level alert is triggered if the increase in storage capacity usage deviation has grown beyond the specified threshold within a day period.”。 NameNode Heap Usage (Weekly) 告警名称: ”NameNode Heap Usage (Weekly)”。描述: ”This service-level alert is triggered if the NameNode heap usage deviation has grown beyond the specified threshold within a week period.”。 HDFS Storage Capacity Usage (Weekly) 告警名称: ”HDFS Storage Capacity Usage (Weekly)”。描述: ”This service-level alert is triggered if the increase in storage capacity usage deviation has grown beyond the specified threshold within a week period.”。 Secondary NameNode Process 告警名称: ”Secondary NameNode Process”。描述: ”This host-level alert is triggered if the Secondary NameNode process cannot be confirmed to be up and listening on the network.”。 NFS Gateway Process 告警名称: ”NFS Gateway Process”。描述: ”This host-level alert is triggered if the NFS Gateway process cannot be confirmed to be up and listening on the network.”。 JournalNode Web UI 告警名称: ”JournalNode Web UI”。描述: ”当无法确定 JournalNode Web UI运行状态时, 触发告警.”。 DataNode Process 告警名称: ”DataNode Process”。描述: ”This host-level alert is triggered if the individual DataNode processes cannot be established to be up and listening on the network.”。 DataNode Web UI 告警名称: ”DataNode Web UI”。描述: ”当无法确定 DataNode Web UI运行状态时, 触发告警.”。 DataNode Storage 告警名称: ”DataNode Storage”。描述: ”This host-level alert is triggered if storage capacity if full on the DataNode. It checks the DataNode JMX Servlet for the Capacity and Remaining properties. The threshold values are in percent.”。 DataNode Unmounted Data Dir 告警名称: ”DataNode Unmounted Data Dir”。描述: ”This host-level alert is triggered if one of the data directories on a host was previously on a mount point and became unmounted. If the mount history file does not exist, then report an error if a host has one or more mounted data directories as well as one or more unmounted data directories on the root partition. This may indicate that a data directory is writing to the root partition, which is undesirable.”。 DataNode Heap Usage 告警名称: ”DataNode Heap Usage”。描述: ”This host-level alert is triggered if heap usage goes past thresholds on the DataNode. It checks the DataNode JMXServlet for the MemHeapUsedM and MemHeapMaxM properties. The threshold values are in percent.”。 ZooKeeper Failover Controller Process 告警名称: ”ZooKeeper Failover Controller Process”。描述: ”当无法确

定ZooKeeper Failover Controller运行状态时, 将触发告警. 单位:秒.”。 Hive Metastore Process 告警名称: ”Hive Metastore Process”。描述: ”This host-level alert is triggered if the Hive Metastore process cannot be determined to be up and listening on the network.”。 Sys DB status 告警名称: ”Sys DB status”。描述: ”This alert is triggered if the Sys Db is not created yet.”。 HiveServer2 Process 告警名称: ”HiveServer2 Process”。描述: ”This host-level alert is triggered if the HiveServer cannot be determined to be up and responding to client requests.”。 HiveServer2 Interactive Process 告警名称: ”HiveServer2 Interactive Process”。描述: ”This host-level alert is triggered if the HiveServerInteractive cannot be determined to be up and responding to client requests.”。 LLAP Application 告警名称: ”LLAP Application”。描述: ”This alert is triggered if the LLAP Application cannot be determined to be up and responding to requests.”。 Oozie Server Web UI 告警名称: ”Oozie Server Web UI”。描述: ”无法确定Oozie server Web UI运行状态时, 将触发告警.”。 Oozie Server Status 告警名称: ”Oozie Server Status”。描述: ”无法确定Oozie server 运行状态时, 将触发告警.”。 Ranger KMS Server Process 告警名称: ”Ranger KMS Server Process”。描述: ”当无法确定Ranger KMS Server运行状态时, 将触发告警. 单位:秒.”。 Spark2 History Server 告警名称: ”Spark2 History Server”。描述: ”当无法确定Spark2 History Server运行状态时, 将触发告警. 单位:秒.”。 Spark2 Livy Server 告警名称: ”Spark2 Livy Server”。描述: ”当无法确定Spark2 Livy Server运行状态时, 将触发告警.”。 Spark2 Thrift Server 告警名称: ”Spark2 Thrift Server”。描述: ”当无法确定Spark2 Thrift Server运行状态时, 将触发告警.”。 ./stacks/HDP/3.0/services/HUE/alerts.json Hue Web UI 告警名称: ”Hue Web UI”。描述: ”当无法确定 Hue Web UI运行状态时, 触发告警.”。 Solr Web UI 告警名称: ”Solr Web UI”。描述: ”无法确定Solr Cloud 运行状态时, 将触发告警.”。 Solr CPU Utilization 告警名称: ”Solr CPU Utilization”。描述: ”当CPU使用率达到设定的阈值时, 将触发告警. 通过Solr JMX Servlet 获取SystemCPULoad 属性. 阈值设定为百分比.”。 Solr Memory Utilization 告警名称: ”Solr Memory Utilization”。描述: ”当内存使用率达到设定的阈值时, 将触发告警. 通过Solr JMX Servlet 获取SystemCPULoad 属性. 阈值设定为百分比.”。 Druid Coordinator Web UI 告警名称: ”Druid Coordinator Web UI”。描述: ”当无法确定Druid Coordinator Web UI运行状态时, 将触发告警. 单位:秒.”。 Druid Overlord Web UI 告警名称: ”Druid Overlord Web UI”。描述: ”当无法确定Druid Overlord Web UI运行状态时, 将触发告警. 单位:秒.”。 Druid Historical Process 告警名称: ”Druid Historical Process”。描述: ”当无法确定Druid Historical Process运行状态时, 将触发告警. 单位:秒.”。 Druid Broker Process 告警名称: ”Druid Broker Process”。描述: ”当无法确定Druid Broker Process运行状态时, 将触发告警. 单位:秒.”。 Druid Middlemanager Process 告警名称: ”Druid Middlemanager Process”。描述: ”当无法确定Druid Middlemanager Process运行状态时, 将触发告警. 单位:秒.”。 Druid Router Process 告警名称: ”Druid Router Process”。描述: ”当无法确定Druid Router Process运行状态时, 将触发告警. 单位:秒.”。 Ranger Admin Process 告警名称: ”Ranger Admin Process”, 描述: ”无法确定Ranger Admin Web UI运行状态时, 将触发告警.”, Ranger Admin password check 告警名称: ”Ranger Admin password check”。描述: ”检测Ranger Admin 密码与托管HADOOP中是否一致.”。 Ranger Usersync Process 告警名称: ”Ranger Usersync Process”。描述: ”当无法确定Ranger Usersync运行状态时, 将触发告警. 单位:秒.”。 Zeppelin Server Status 告警名称: ”Zeppelin Server Status”。描述: ”This host-level alert is triggered if the Zeppelin server cannot be determined to be up and responding to client requests.”。 Percent ZooKeeper Servers Available 告警名称: ”Percent ZooKeeper Servers Available”。描述: ”This alert is triggered if the number of down ZooKeeper servers in the cluster is greater than the configured critical threshold. It aggregates the results of ZooKeeper process checks.”。 ZooKeeper Server

Process 告警名称: ”ZooKeeper Server Process”。描述: ”This host-level alert is triggered if the ZooKeeper server process cannot be determined to be up and listening on the network.”。 ElasticSearch Process Check 告警名称: ”ElasticSearch Process Check”。描述: ”当无法确定ES Master运行状态时, 将触发告警. 单位: 秒”。 History Server Web UI 告警名称: ”History Server Web UI”。描述: ”当无法确定History Server UI运行状态时, 触发告警.”。 History Server CPU 使用率 告警名称: ”History Server CPU 使用率”。描述: ”当History Server CPU 使用率达到阈值时, 触发告警, 阈值设定为百分比.”。 History Server RPC 延迟 告警名称: ”History Server RPC 延迟”。描述: ”当History Server RPC 延迟达到阈值, 触发告警. 一般增加RPC队列长度时也要调整RPC 延迟时长. 单位: 毫秒(ms).”。 Percent NodeManagers Available 告警名称: ”Percent NodeManagers Available”。描述: ”当停止的NodeManager 百分比达到阈值时, 触发告警. 该结果聚合NodeManager 的所有检测结果.”。 NodeManager Web UI 告警名称: ”NodeManager Web UI”。描述: ”当无法确定NodeManager Web UI运行状态时, 触发告警.”。 NodeManager Health 告警名称: ”NodeManager Health”。描述: ”从NodeManager获取节点健康状态.”。 ResourceManager Web UI 告警名称: ”ResourceManager Web UI”。描述: ”当无法确定ResourceManager Web UI运行状态时, 触发告警.”。 ResourceManager CPU 使用率 告警名称: ”ResourceManager CPU 使用率”。描述: ”ResourceManager CPU 使用率达到阈值时, 通过连接ResourceManager JMX Servlet获取SystemCPULoad 属性, 触发告警, 阈值设定为百分比.”。 ResourceManager RPC 延迟 告警名称: ”ResourceManager RPC 延迟”。描述: ”当ResourceManager RPC 延迟达到阈值, 触发告警. 一般增加RPC队列长度时也要调整RPC 延迟时长. 单位: 毫秒(ms).”。 NodeManager Health Summary 告警名称: ”NodeManager Health Summary”。描述: ”当集群中有不正常的 NodeManager时, 触发告警”。 App Timeline Web UI 告警名称: ”App Timeline Web UI”。描述: ”当无法确定App Timeline Web UI运行状态时, 触发告警.”。 Registry DNS 告警名称: ”Registry DNS”。描述: ”当无法确定Registry DNS运行状态时, 将触发告警. 单位:秒.”。 Metadata Server Web UI 告警名称: ”Metadata Server Web UI”。描述: ”当无法确定 Metadata Server Web UI运行状态时, 触发告警.”。 Superset Web UI 告警名称: ”Superset Web UI”。描述: ”当无法确定Superset Web UI运行状态时, 触发告警.”。 Accumulo Master Process 告警名称: ”Accumulo Master Process”。描述: ”当无法确定Accumulo Master运行状态时, 将触发告警. 单位:秒.”。 Accumulo TServer Process 告警名称: ”Accumulo TServer Process”。描述: ”当无法确定Accumulo TServer运行状态时, 将触发告警. 单位:秒.”。 Accumulo GC Process 告警名称: ”Accumulo GC Process”。描述: ”当无法确定Accumulo GC运行状态时, 将触发告警. 单位:秒.”。 Accumulo Monitor Process 告警名称: ”Accumulo Monitor Process”。描述: ”当无法确定Accumulo Monitor运行状态时, 将触发告警. 单位:秒.”。

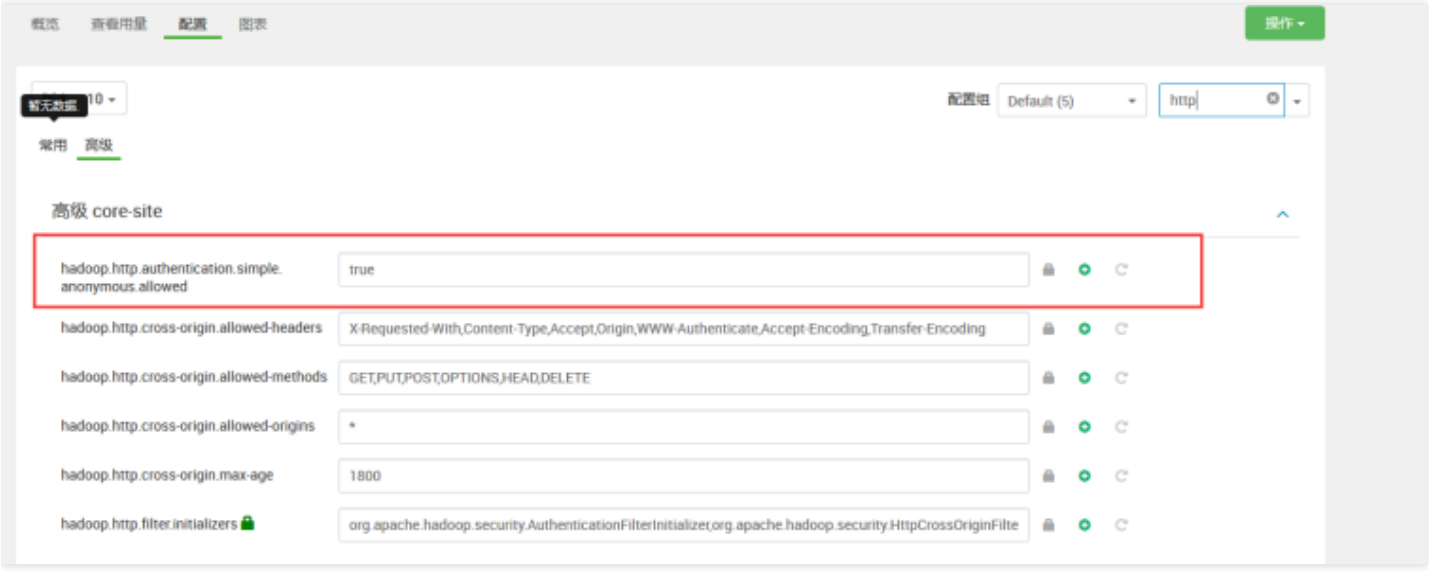
故障处理

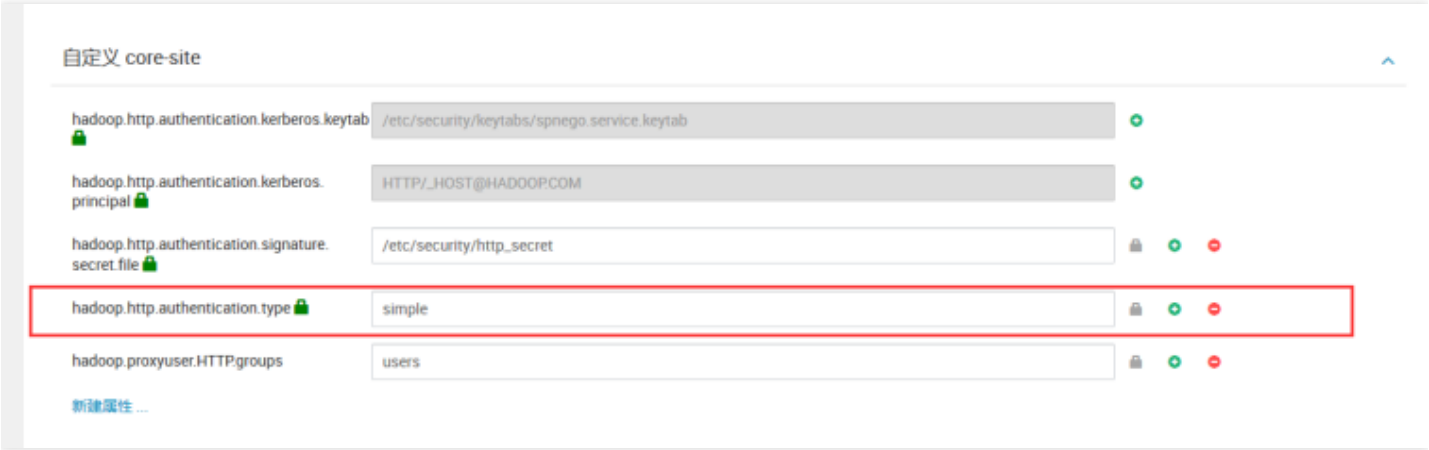
最近更新时间: 2021-09-15 16:33:18

HDFS Hdfs用户权限问题

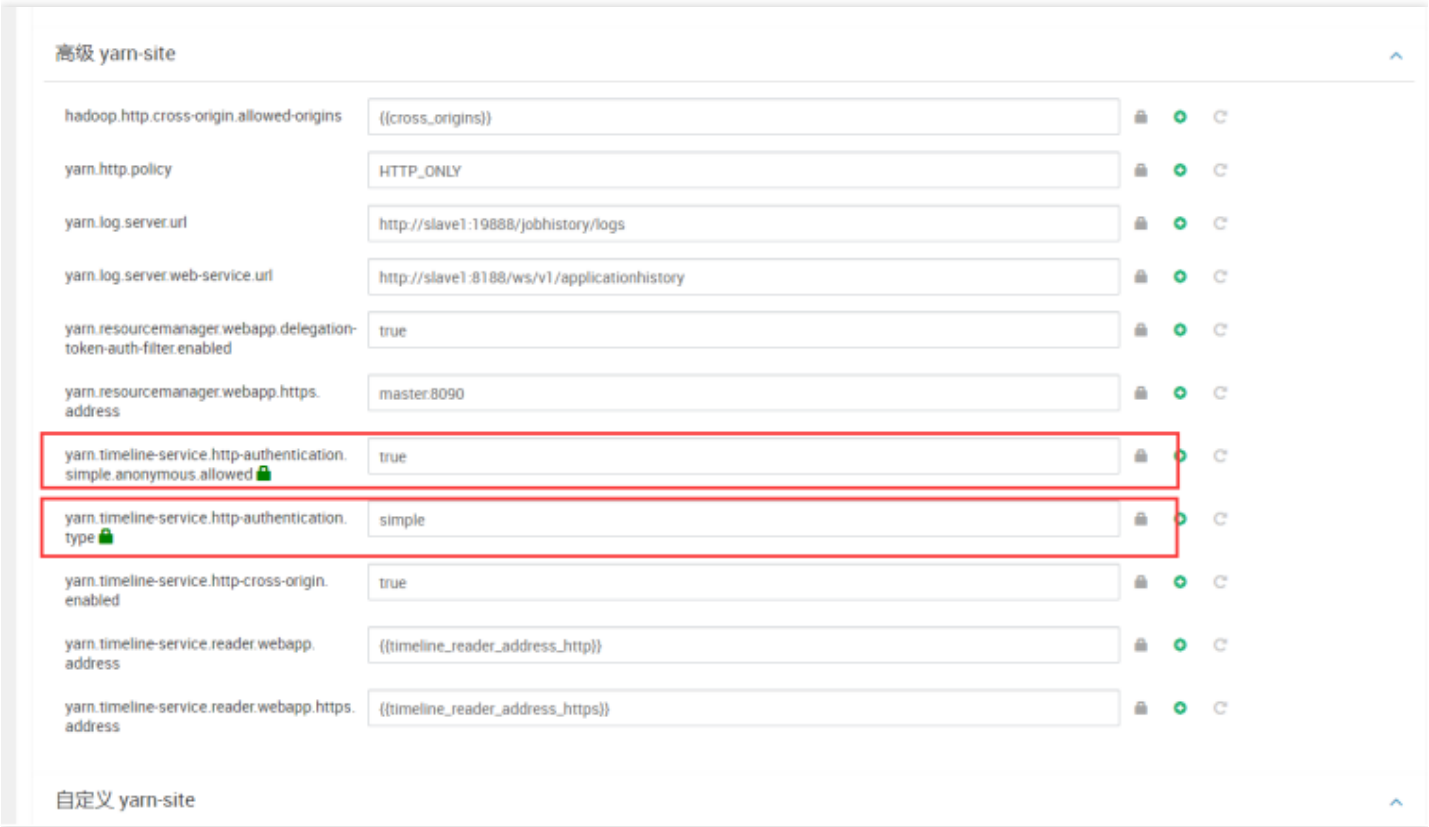
打开Hdfs和yarn的UI在Kerberos的环境下

在hdfs的配置中搜索http





在yarn的高级配置中搜索http



重启hdfs、yarn

namenode启动报错在日志文件中ERROR namenode.NameNode (NameNode.java:main(1712)) – Failed to start namenode

原因：日志中还有java.net.BindException: Port in use: gmaster:50070，Caused by: java.net.BindException: Address already in use

判断原因是50070上一次没有释放，端口占用

解决：

netstat下time_wait状态的tcp连接：

1.这是一种处于连接完全关闭状态前的状态；

2.通常要等上4分钟（windows server）的时间才能完全关闭；

3.这种状态下的tcp连接占用句柄与端口等资源，服务器也要为维护这些连接状态消耗资源；

4.解决这种time_wait的tcp连接只有让服务器能够快速回收和重用那些TIME_WAIT的资源：修改注册表[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters]添加dword值TcpTimedWaitDelay=30（30也为微软建议值；默认为2分钟）和MaxUserPort：65534(可选值5000 – 65534)；

5.具体tcpip连接参数配置还可参照这里： http://technet.microsoft.com/zh-tw/library/cc776295%28v=ws.10%29.aspx

6.linux下：

vi /etc/sysctl.conf

新增如下内容：

net.ipv4.tcp_tw_reuse = 1

net.ipv4.tcp_tw_recycle = 1

net.ipv4.tcp_syncookies=1

net.ipv4.tcp_fin_timeout=30

net.ipv4.tcp_keepalive_time=1800

net.ipv4.tcp_max_syn_backlog=8192

使内核参数生效：

[root@web02 ~]# sysctl –p

readme:

net.ipv4.tcp_syncookies=1 打开TIME-WAIT套接字重用功能，对于存在大量连接的Web服务器非常有效。

net.ipv4.tcp_tw_recyle=1

net.ipv4.tcp_tw_reuse=1 减少处于FIN-WAIT-2连接状态的时间，使系统可以处理更多的连接。

net.ipv4.tcp_fin_timeout=30 减少TCP KeepAlive连接侦测的时间，使系统可以处理更多的连接。

net.ipv4.tcp_keepalive_time=1800 增加TCP SYN队列长度，使系统可以处理更多的并发连接。

net.ipv4.tcp_max_syn_backlog=8192

Cannot set permission for /ats/done. Name node is in safe mode.

使用hdfs用户执行以下命令，让NameNode处于非安全模式即可。

$ sudo –u hdfs Hadoop dfsadmin –safemode leave HDFS块丢失异常

现象：

HDFS进入安全模式，服务不可用。

HDFS服务页面块丢失的值大于0。

HDFS服务启动失败，角色实例启动成功。

可能原因：

数据节点硬盘故障，或节点故障可能导致数据副本丢失。

HDFS在如下情况进入安全模式：

当NameNode启动且等待DataNode上报副本。

NameNode所在磁盘空间不足。

恢复NameNode数据后，元数据与业务数据无法匹配

HDFS对应文件的副本全部丢失。

解决：

打开托管HADOOP的管理界面，看是否有节点、硬盘故障告警。

以“hdfs”用户使用HDFS客户端工具执行fsck检查文件系统中文件是否完整。

若fsck校验仅显示副本丢失，而不是文件丢失（看已有副本数是否大于0），则执行*hdfs dfsadmin –safemode leave*退出HDFS安全模式，即可修复。

若文件丢失，检查是否执行了恢复NameNode数据操作。

是，表示找不到与元数据对应的业务数据块，执行*hdfs dfsadmin –safemode leave*退出HDFS安全模式

否，表示文件丢失

若有文件丢失，检查丢失文件的文件路径，并检查文件是否为重要文件。

MapReduce类文件是非重要的，对非重要文件，退出安全模式后，可执行删除操作。

hdfs fsck [path-to-file] -delete
恢复NameNode数据后可能导致元数据与业务数据无法匹配，需要执行删除操作。
hdfs fsck / -delete
检查NameNode的块丢失阈值是否配置过高。
重启HDFS服务。
HDFS客户端无法连接到HDFS集群
现象：
HDFS客户端无法连接到HDFS。
原因：
未通过Kerberos安全认证。
NameNode IP地址不正确。
HDFS服务不正常。
解决：
服务列表界面及告警界面检查HDFS服务是否正常
查看正确的ip地址。
如果集群为安全模式，执行命令kinit hdfs，以业务帐户登录HDFS客户端。
Hive
hive3.0 current_timestamp时间差问题
Hive3.0时间戳默认时区是UTC，currenttimestamp是从UTC取时间，不是从本地获取。所以要获取当前系统时间使用函数 fromutctimestamp(currenttimestamp,'Asia/Shanghai')。
hive中文注释乱码问题
原因：Hive字段中文乱码，如执行 show create table xxx 时，表级别注释、字段级别注释发现有乱码现象，一般都是由hive 元数据库的配置不当造成的。
解决：登录hive的元数据库mysql中：
设置hive 元数据库字符集
show create database hive;
查看为utf8，需变更为latin1
alter database hive character set latin1;
更改如下表字段为字符集编码为 utf8
hive权限管理、ranger权限管理、hdfs权限管理，hive权限
通过ranger管理，可跳过hdfs文件权限
在使用hive命令进入hive的时候报错误Permission denied: user=root, access=WRITE,
inode="/user/root":hdfs:hdfs:drwxr-xr-x hive
在有ranger的情况下使用ranger赋权
在没有ranger的情况下：
使用HDFS的命令行接口修改相应目录的权限，Hadoop fs -chmod 777 /user,后面的/user是要上传文件的路径，不同的情况可能不一样，比如要上传的文件路径为hdfs://namenode/user/xxx.doc，则这样的修改可以，如果要上传的文件路径为 hdfs://namenode/java/xxx.doc，则要修改的为Hadoop fs -chmod 777 /java或者Hadoop fs -

chmod 777 /，java的那个需要先在HDFS里面建立Java目录，后面的这个是为根目录调整权限。
脚本：
su – hdfs
Hadoop fs –chmod 777 /user
在/etc/profile文件中加上系统的环境变量或java JVM变量里面添加export HADOOP_USER_NAME=hdfs(ambari使用的Hadoop用户是hdfs)，这个值具体等于多少看自己的情况，以后会运行HADOOP上的Linux的用户名。
export HADOOP_USER_NAME=hdfs
Shell客户端提示“authentication failed”
现象：
安全版本的集群中，HiveServer服务正常的情况下，Shell客户端中执行beeline命令，登录失败，界面提示“authentication failed”关键字
原因：
客户端用户未进行Kerberos认证。 Kerberos认证超期 解决： 对客户端用户再次进行Kerberos认证 认证完成后，再次登录。 恢复该故障正常设置耗时1分钟以内。 操作表时Shell客户端提示“Unable to determine if xxx encryptede” 现象： 在HiveServer服务正常的情况下，用户poctest去访问表checkpoint，命令行终端显示类似如下错误：

```
0: jdbc:hive2://ha-cluster/default> select * from checkpoint;
Error: Error while compiling statement: FAILED: SemanticException Unable to determine if hdfs://hacl
uster/user/hive/warehouse/checkpointis encrypted: org.apache.Hadoop.security.AccessControlExcept
ion: Permission denied: user=poctest, access=READ, inode="/user/hive/warehouse/checkpoint":poco
wn:hive:drwx------
```

原因: 如果没有给poctest用户授予checkpoint表的访问权限，用户无法访问该表。 解决： 查看poctest用户权限是否有checkpoint表的访问权限。 使用Ranger界面授权授予poctest用户checkpoint表的访问权限。 正常授权耗时1分钟以内。 Hive服务器端日志显示“File does not exist”异常 现象： 在HiveServer服务正常的情况下，开启Hive本地模式，对HiveoverHBase表（数据存于HBase的表）进行操作（如查询），Hive客户端显示执行错误，同时Hive服务器端的子进程日志显示“File does not exist”异常。 Hive客户端执行错误，具体信息如下：

```
0: jdbc:hive2://10.64.35.144:21066/default> select count(*) from default__tb_1_index_3__;
Error: Error while processing statement: FAILED: Execution Error, return code 1 from org.apache.Hado
op.hive.ql.exec.mr.MapRedTask (state=08S01,code=1)
查询“/var/log/Bigdata/hive/hiveserver/hive.log”路径下的Hive服务器端的子进程日志显示“File does n
ot exist”异常，具体信息如下：
2015-03-30 14:17:34,586 ERROR [main]: mr.ExecDriver (SessionState.java:printError(548)) - Job Submi
ssion failed with exception 'java.io.FileNotFoundException(File does not exist: hdfs://hacluster/opt/hu
awei/Bigdata/FusionInsight_HD_V100R002CXX/install/FusionInsight-Hive-1.3.0/hive-1.3.0/lib/hive-hb
ase-handler-1.3.0.jar)'
java.io.FileNotFoundException: File does not exist: hdfs://hacluster/opt/huawei/Bigdata/FusionInsight
_HD_V100R002CXX/install/FusionInsight-Hive-1.3.0/hive-1.3.0/lib/hive-hbase-handler-1.3.0.jar
```

原因： 在集群模式下，运行MapReduce作业时，第三方jar包放置在分布式缓存中，作业运行完后jar包释放。在本地模式（Local Mode）下，需要从本机中获取第三方jar包，但现有的MapReduce机制不支持从本机获取第三方jar包。在Hive提交作业后，MapReduce会把第三方jar包的路径强制转换为本地模式下不存在的HDFS路径，导致系统运行作业时取不到jar包而出错。 例如，第三方jar包路径为“file:/opt/huawei/BigData/hive/lib/hbase-exec.jar”。运行MapReduce作业前，系统会强制将该路径转化为不存在的路径“hdfs:/opt/huawei/BigData/hive/lib/hbase-exec.jar”。 解决： 在进行HiveoverHBase表操作之前，上传第三方jar包到HDFS的指定路径，可解决该问题。 通过Hue界面执行SQL失败时页面没有提示信息 现象： 通过Hue界面的“Query Editor > Hive”页面，执行SQL语句。因为语句原因或者权限问题导致语句执行失败，但是页面没有错误信息提示。 如下图所示：

正常情况下应该有如下提示信息：

原因：
可能是浏览器缓存导致信息无法正常显示。
极少数情况会出现错误信息无法提示的情况。
解决：
清除浏览器缓存即可解除此问题。
执行动态插入分区时，在MapReduce日志中报“java.lang.OutOfMemoryError: GC overhead limit exceeded”错

误 现象： 在HiveServer服务正常的情况下，执行动态插入分区时，在MapReduce日志报“java.lang.OutOfMemoryError: GC overhead limit exceeded”错误。 原因： 产生OOM的原因是单个任务处理的分区数过多，需要针对具体场景，减少单个task处理的分区数。
解决：
参照如下样例进行操作。
样例建表语句如下：
create table test(id int)partitioned by (dt int); create table test1(id int, dt int); 正常的动态插入分区语句为：insert overwrite table test partition (dt) select id, dt from test; HBase HMaster Web UI显示空region，重新创建相同的table时失败 现象： HMaster Web UI可能显示空region的table。

当客户端重试创建相同的table时会上报下面的错误。
ERROR: java.io.IOException: the table directory:hdfs://10.18.106.212:8020/hbase/data/default/t1 exists,this may be created by other process,please delete the directory first. at org.apache.Hadoop.hbase.master.handler.CreateTableHandler.prepare(CreateTableHandler.java:131) at org.apache.Hadoop.hbase.master.HMaster.createTable(HMaster.java:1559) at org.apache.Hadoop.hbase.master.MasterRpcServices.createTable(MasterRpcServices.java:468) at org.apache.Hadoop.hbase.protobuf.generated.MasterProtos$MasterService$2.callBlockingMethod(MasterProtos.java:59779) at org.apache.Hadoop.hbase.ipc.RpcServer.call(RpcServer.java:2171) at org.apache.Hadoop.hbase.ipc.CallRunner.run(CallRunner.java:112) at org.apache.Hadoop.hbase.ipc.RpcExecutor.consumerLoop(RpcExecutor.java:133) at org.apache.Hadoop.hbase.ipc.RpcExecutor$1.run(RpcExecutor.java:108) at java.lang.Thread.run(Thread.java:745) 原因： 在故障情况下，当HMaster在文件系统中已经创建了table，但是由于一些原因在meta table中不能更新region信息。所以table在文件系统中存在但是在meta table中不存在。 在HMaster主备倒换/重启时，HMaster Web UI将会显示如上图所示的空region。这里的table是基于文件系统而不是meta table列出的，所以hbase中不存在表的region信息，meta table显示为空。 解决： 扫描“hbase:meta”table，并检查是否存在table region信息。 Could not create the Java Virtual Machine. 现象：执行$ hbase hbck 命令时，出现以下提示： Invalid maximum heap size: -Xmx4096m The specified size exceeds the maximum representable size. Error: Could not create the Java Virtual Machine. Error: A fatal exception has occurred. Program will exit. 原因： jvm设置的内存过大。 解决： 减小配置文件hbase-env.sh内的设置即可。 export HBASE_HEAPSIZE=1024 无法启动hbase，regionserver log里会有这样的错误，zookeeper也有初始化问题的错误 现象： FATAL org.apache.Hadoop.hbase.regionserver.HRegionServer:

ABORTING region server 10.210.70.57,60020,1340088145399: Initialization of RS failed. Hence aborting RS. 解决： 因为之前安装配置的时候是好好的，中间经历过强行kill daemon的过程，又是报错初始化问题，所以估计是有缓存影响了，所以清理了tmp里的数据，然后发现HRegionServer依然无法启动，不过还好的是zookeeper启动了，一怒之下把hdfs里的hbase数据也都清理了，同时再清理tmp，检查各个节点是否有残留hbase进程，kill掉，重启hbase，然后这个世界都正常了。不知道具体哪里影响了，不推荐这种暴力解决办法，如果有谁知道原因请告之 无法启动reginserver daemon 现象： Exception in thread ”main” java.lang.RuntimeException: Failed construction of Regionserver: class org.apache.Hadoop.hbase.regionserver.HRegionServer ... Caused by: java.net.BindException: Problem binding to /10.210.70.57:60020 : Cannot assign requested address 解决：根据错误提示，检查ip对应的机器是否正确，如果出错机器的ip正确，检查60020端口是否被占用。 regionserver 无法启动 现象： 如果启动hbase集群出现regionserver无法启动，日志报告如下类似错误时，说明是集群的时间不同步，只需要同步即可解决。 FATAL org.apache.Hadoop.hbase.regionserver.HRegionServer: ABORTING region server 10.210.78.22,60020,1344329095415: Unhandled exceptio n: org.apache.Hadoop.hbase.ClockOutOfSyncException: Server 10.210.78.22,60020,1344329095415 has been rejected; Reported time is too far out of sync with mast er. Time difference of 90358ms > max allowed of 30000ms org.apache.Hadoop.hbase.ClockOutOfSyncException: org.apache.Hadoop.hbase.ClockOutOfSyncException: Server 10.210.78.22,60020,1344329095415 has been rejected; Reported time is too far out of sync with master. Time difference of 90358ms > max allowed of 30000ms ..... Caused by: org.apache.Hadoop.ipc.RemoteException: org.apache.Hadoop.hbase.ClockOutOfSyncException: Server 10.210.78.22,60020,1344329095415 has been rejected; Reported time is too far out of sync with master. Time difference of 90358ms > max allowed of 30000ms 解决： 只需要执行一下这条命令即可同步国际时间： /usr/sbin/ntpdate tick.ucla.edu tock.gpsclock.com ntp.nasa.gov timekeeper.isi.edu usno.pa-x.dec.com;/sbin/hwclock --systohc > /dev/null Failed all from refion=r 现象：

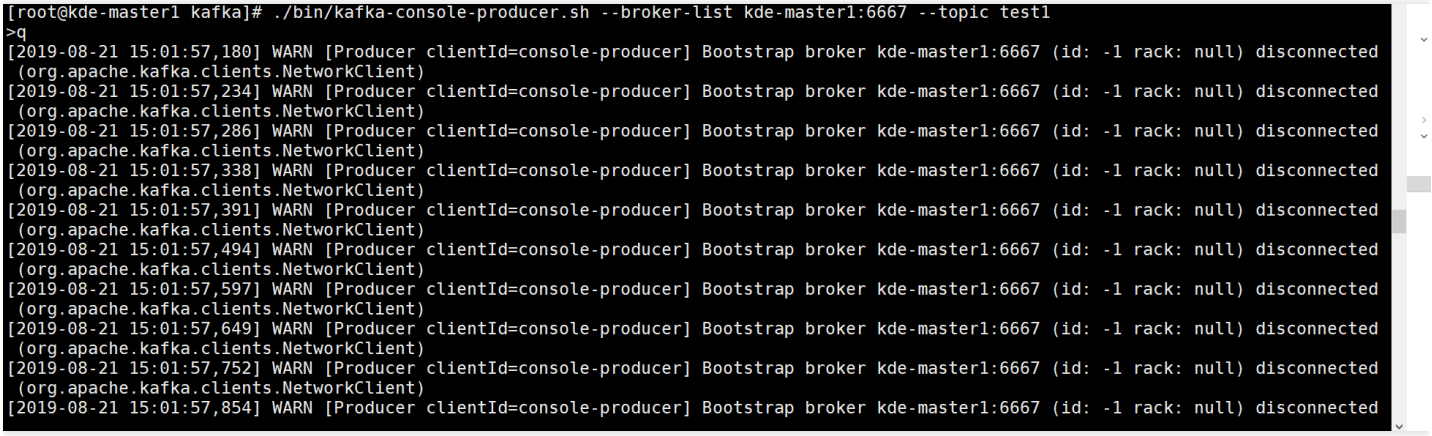
```
14/04/11 13:36:00 INFO mapred.LocalJobRunner: reduce > reduce
14/04/11                    13:36:56                    WARN
client.HConnectionManager$HConnectionImplementation:   Failed   all
from region=r
oomRecord,,1397222782606.ee7bebac03a04c8457dbf166d8494fbd.,
hostname=slave1, port=60020
java.util.concurrent.ExecutionException:
java.net.SocketTimeoutException: Call to slave1/192.168.239
.102:60020     failed     on     socket     timeout     exception:
java.net.SocketTimeoutException: 60000 millis timeout
 while   waiting   for   channel   to   be   ready   for   read.   ch   :
java.nio.channels.SocketChannel[connected loca
l=/192.168.239.101:5968 remote=slave1/192.168.239.102:60020]
    at
java.util.concurrent.FutureTask$Sync.innerGet(FutureTask.java:222
)
    at java.util.concurrent.FutureTask.get(FutureTask.java:83)
    at
org.apache.hadoop.hbase.client.HConnectionManager$HConnectionImpl
ementation.processBatchCallback
(HConnectionManager.java:1598)
    at
org.apache.hadoop.hbase.client.HConnectionManager$HConnectionImpl
ementation.processBatch(HConnec
tionManager.java:1450)
    at
org.apache.hadoop.hbase.client.HTable.flushCommits(HTable.java:91
6)
    at org.apache.hadoop.hbase.client.HTable.doPut(HTable.java:772)
    at org.apache.hadoop.hbase.client.HTable.put(HTable.java:748)
    at
org.apache.hadoop.hbase.mapreduce.TableOutputFormat$TableRecordWr
iter.write(TableOutputFormat.ja
va:123)
```

原因： 因为服务器处理时间过长，客户端自动断开连接，当服务器处理完成返回数据时，发现连接断开，故抛出异常。 解决：hbase.rpc.timeout默认值为60000ms，可以适当调大这个值，可以从配置文件里调整，也可以通过conf.set(”hbase.rpc.timeout”, ”6000000”)进行调整。 将zookeeper的时间调大：调整zookeeper.session.timeout在hbase-site.xml中调大如下两个值（根据实际情况调大）

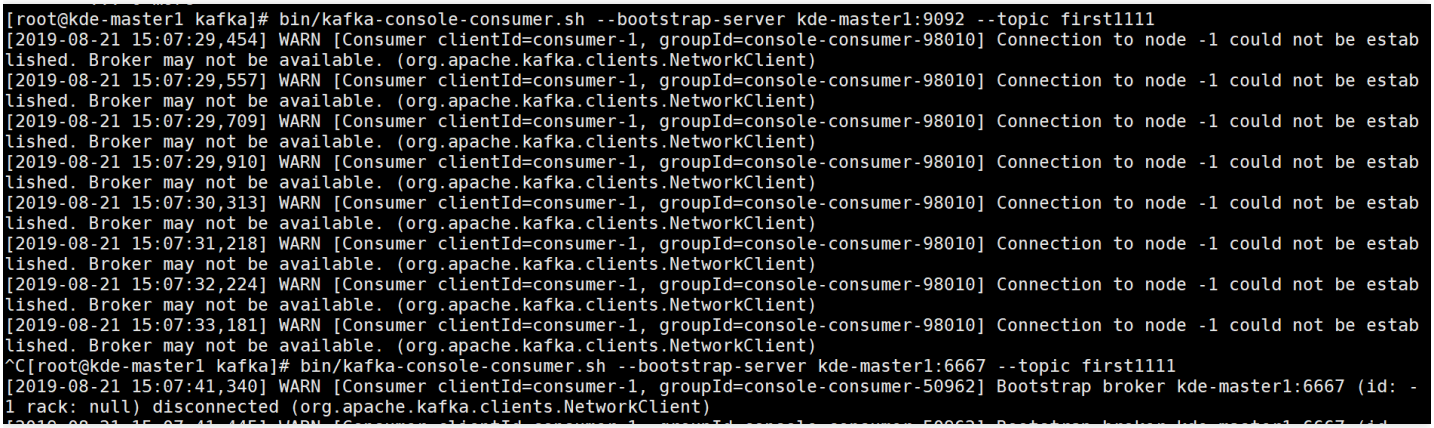
```
<property>
<name>zookeeper.session.timeout</name>
<value>60000</value>
</property>
<property>
<name>hbase.zookeeper.property.tickTime</name>
```

```
<value>2000</value>
</property>
```

加大region数据，让region均匀分配：调节hbase-site.xml中的hbase.hregion.max.filesize值，默认为256M，可以调整到1G，有人甚至调到4G（更大的Region可以使你集群上的Region的总数量较少。一般来言，更少的Region可以使你的集群运行更加流畅。）。 Kafka kafka的消费和生产在添加kerberos都出现问题 现象： WARN [Producer clientId=console-producer] Bootstrap broker 托管Hadoop-master1:6667 (id: -1 rack: null) disconnected (org.apache.kafka.clients.NetworkClient) [root@托管Hadoop-master1 kafka]# ./bin/kafka-console-producer.sh --broker-list 托管Hadoop-master1:6667 --topic test1 [root@托管Hadoop-master1 kafka]# bin/kafka-console-consumer.sh --bootstrap-server 托管Hadoop-master1:6667 --topic first1111



```
[root@kde-master1 kafka]# ./bin/kafka-console-producer.sh --broker-list kde-master1:6667 --topic test1
>q
[2019-08-21 15:01:57,180] WARN [Producer clientId=console-producer] Bootstrap broker kde-master1:6667 (id: -1 rack: null) disconnected
 (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:01:57,234] WARN [Producer clientId=console-producer] Bootstrap broker kde-master1:6667 (id: -1 rack: null) disconnected
 (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:01:57,286] WARN [Producer clientId=console-producer] Bootstrap broker kde-master1:6667 (id: -1 rack: null) disconnected
 (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:01:57,338] WARN [Producer clientId=console-producer] Bootstrap broker kde-master1:6667 (id: -1 rack: null) disconnected
 (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:01:57,391] WARN [Producer clientId=console-producer] Bootstrap broker kde-master1:6667 (id: -1 rack: null) disconnected
 (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:01:57,494] WARN [Producer clientId=console-producer] Bootstrap broker kde-master1:6667 (id: -1 rack: null) disconnected
 (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:01:57,597] WARN [Producer clientId=console-producer] Bootstrap broker kde-master1:6667 (id: -1 rack: null) disconnected
 (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:01:57,649] WARN [Producer clientId=console-producer] Bootstrap broker kde-master1:6667 (id: -1 rack: null) disconnected
 (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:01:57,752] WARN [Producer clientId=console-producer] Bootstrap broker kde-master1:6667 (id: -1 rack: null) disconnected
 (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:01:57,854] WARN [Producer clientId=console-producer] Bootstrap broker kde-master1:6667 (id: -1 rack: null) disconnected
```

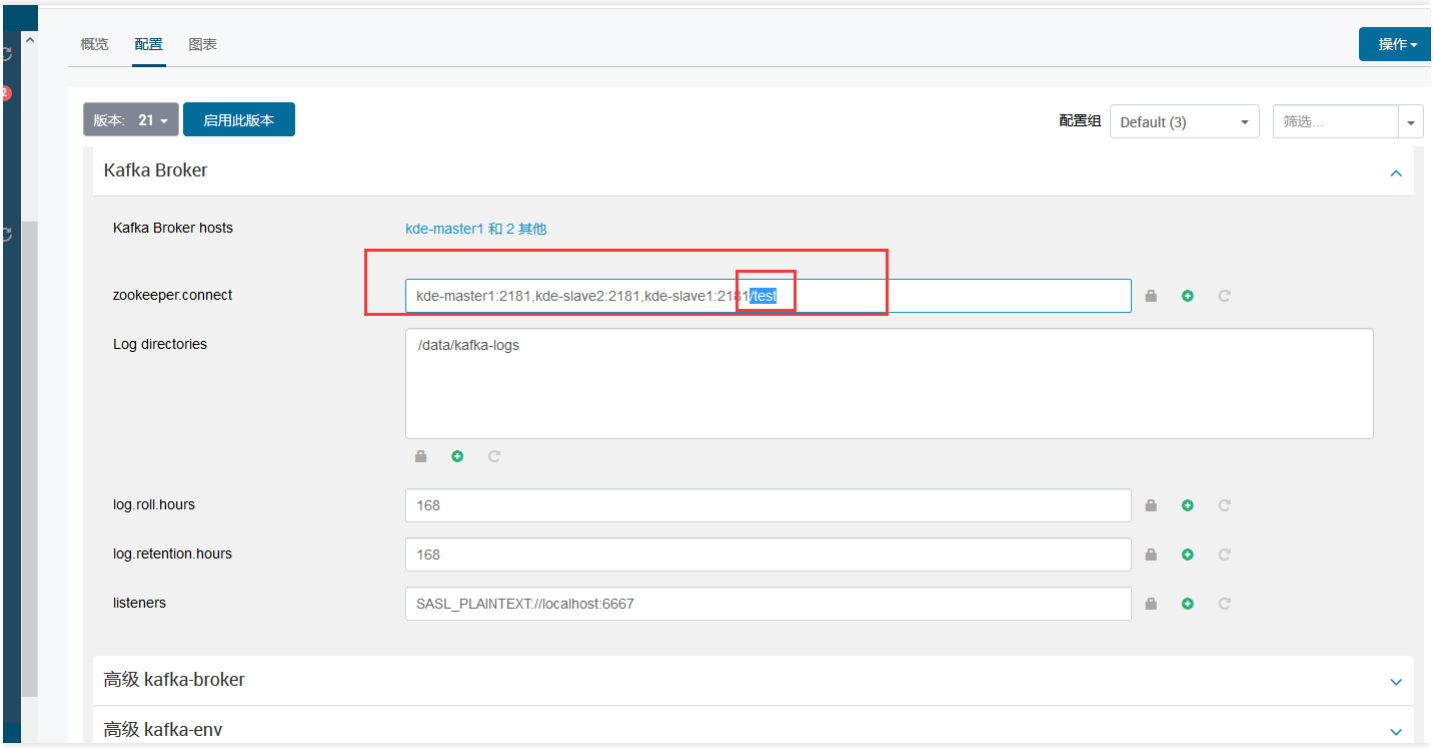


```
[root@kde-master1 kafka]# bin/kafka-console-consumer.sh --bootstrap-server kde-master1:9092 --topic first1111
[2019-08-21 15:07:29,454] WARN [Consumer clientId=consumer-1, groupId=console-consumer-98010] Connection to node -1 could not be estab
lished. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:07:29,557] WARN [Consumer clientId=consumer-1, groupId=console-consumer-98010] Connection to node -1 could not be estab
lished. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:07:29,709] WARN [Consumer clientId=consumer-1, groupId=console-consumer-98010] Connection to node -1 could not be estab
lished. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:07:29,910] WARN [Consumer clientId=consumer-1, groupId=console-consumer-98010] Connection to node -1 could not be estab
lished. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:07:30,313] WARN [Consumer clientId=consumer-1, groupId=console-consumer-98010] Connection to node -1 could not be estab
lished. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:07:31,218] WARN [Consumer clientId=consumer-1, groupId=console-consumer-98010] Connection to node -1 could not be estab
lished. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:07:32,224] WARN [Consumer clientId=consumer-1, groupId=console-consumer-98010] Connection to node -1 could not be estab
lished. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:07:33,181] WARN [Consumer clientId=consumer-1, groupId=console-consumer-98010] Connection to node -1 could not be estab
lished. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
^C[root@kde-master1 kafka]# bin/kafka-console-consumer.sh --bootstrap-server kde-master1:6667 --topic first1111
[2019-08-21 15:07:41,340] WARN [Consumer clientId=consumer-1, groupId=console-consumer-50962] Bootstrap broker kde-master1:6667 (id: -
1 rack: null) disconnected (org.apache.kafka.clients.NetworkClient)
```

解决： 在命令的最后添加--security-protocol PLNTEXTSASL [root@托管Hadoop-master1 kafka]# ./bin/kafka-console-producer.sh --broker-list 托管Hadoop-master1:6667 --topic first1111 --security-protocol PLAINTEXTSASL [root@托管Hadoop-master1 kafka]# bin/kafka-console-consumer.sh --bootstrap-server 托管Hadoop-master1:6667 --topic first1111 --security-protocol PLNTEXTSASL 找不到zk的元数据 现象： WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 1 : {first111=LEADERNOTAVAILABLE} (org.apache.kafka.clients.NetworkClient)

```
[root@kde-master1 kafka]# ./bin/kafka-console-producer.sh --broker-list kde-master1:6667 --topic first111 --security-protocol PLAINTEX
TSASL
>qq
[2019-08-21 15:00:51,585] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 1 : {first111=LE
ADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:00:51,685] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 2 : {first111=LE
ADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:00:51,788] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 3 : {first111=LE
ADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:00:51,891] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 4 : {first111=LE
ADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:00:51,994] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 5 : {first111=LE
ADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:00:52,097] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 6 : {first111=LE
ADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:00:52,198] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 7 : {first111=LE
ADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:00:52,300] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 8 : {first111=LE
ADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:00:52,403] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 9 : {first111=LE
ADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:00:52,506] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 10 : {first111=L
EADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:00:52,607] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 11 : {first111=L
EADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-08-21 15:00:52,709] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 12 : {first111=L
```

原因： zk的地址出现了问题，可能是原数据里面有脏数据或者数据损坏，在zk的元数据连接后面重新添加一个目录
解决：



Kafka节点磁盘占用率达到100%后导致Broker实例状态为“恢复中” 现象： 当Kafka配置的数据目录所在的磁盘占用率达到100%后，该节点的Broker实例出现“恢复中”状态，导致当前节点无法提供Kafka服务。 原因： Topic的Partition划分不合理，导致个别磁盘占用率达到100%。 本地磁盘上还存在除Kafka外的其他数据，导致磁盘占用率达到100%。 解决： 定位磁盘占用率达到100%的根因。 删除或移动数据到其他空闲磁盘，重启Broker。 解决磁盘容量不足问题 Yarn NodeManager节点故障 现象： 一个或者多个NodeManager处于“恢复中”状态。 上报NodeManager不健康告警或者NodeManager心跳丢失告警。 当MapReduce任务进行过程中，出现NodeManager节点故障，可能会导致任务失败或者运行速度变慢。 原因： 该节点网络异常。 ZooKeeper服务异常。 该节点

NodeManager进程异常。 解决： 查看ZooKeeper服务是否正常。 查看ResourceManager页面中提供的异常信息。 NodeManager出现内存超限错误导致执行Spark任务失败 现象： 当开启Spark动态调度时，运行大数据量（百万task）的Spark任务，在执行到reduce阶段时出现重试，多次重试后任务执行失败。运行过程中，NodeManager进程出现“java.lang.OutOfMemoryError: Direct buffer memory”的错误，日志如下：

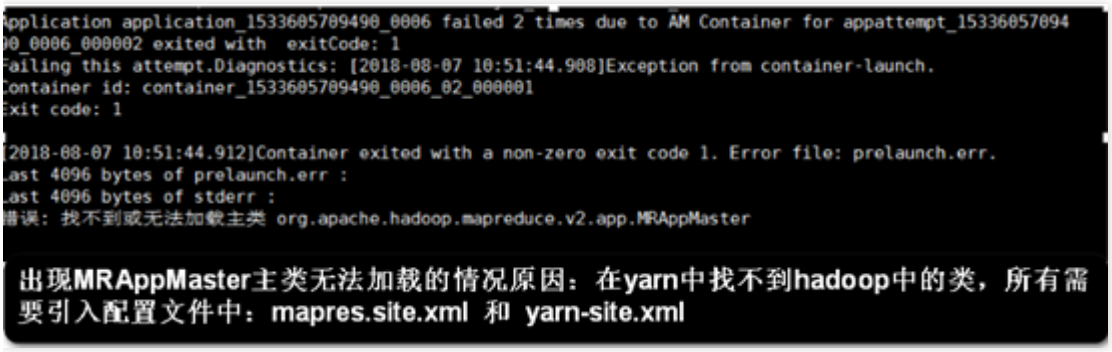
```
2019-07-06 19:05:15,794 | WARN | shuffle-server-37 | Exception in connection from /172.16.0.130:496
54 | TransportChannelHandler.java:79
io.netty.handler.codec.DecoderException: java.lang.OutOfMemoryError: Direct buffer memory
at io.netty.handler.codec.ByteToMessageDecoder.channelRead(ByteToMessageDecoder.java:153)
at io.netty.channel.AbstractChannelHandlerContext.invokeChannelRead(AbstractChannelHandlerCont
ext.java:333)
at io.netty.channel.AbstractChannelHandlerContext.fireChannelRead(AbstractChannelHandlerContext.
java:319)
at io.netty.channel.DefaultChannelPipeline.fireChannelRead(DefaultChannelPipeline.java:787)
at io.netty.channel.nio.AbstractNioByteChannel$NioByteUnsafe.read(AbstractNioByteChannel.java:13
0)
at io.netty.channel.nio.NioEventLoop.processSelectedKey(NioEventLoop.java:511)
at io.netty.channel.nio.NioEventLoop.processSelectedKeysOptimized(NioEventLoop.java:468)
at io.netty.channel.nio.NioEventLoop.processSelectedKeys(NioEventLoop.java:382)
at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:354)
at io.netty.util.concurrent.SingleThreadEventExecutor$2.run(SingleThreadEventExecutor.java:116)
at java.lang.Thread.run(Thread.java:745)
Caused by: java.lang.OutOfMemoryError: Direct buffer memory
at java.nio.Bits.reserveMemory(Bits.java:693)
at java.nio.DirectByteBuffer.<init>(DirectByteBuffer.java:123)
at java.nio.ByteBuffer.allocateDirect(ByteBuffer.java:311)
at io.netty.buffer.PoolArena$DirectArena.newChunk(PoolArena.java:434)
at io.netty.buffer.PoolArena.allocateNormal(PoolArena.java:179)
at io.netty.buffer.PoolArena.allocate(PoolArena.java:168)
at io.netty.buffer.PoolArena.reallocate(PoolArena.java:277)
at io.netty.buffer.PooledByteBuf.capacity(PooledByteBuf.java:108)
at io.netty.buffer.AbstractByteBuf.ensureWritable(AbstractByteBuf.java:251)
at io.netty.buffer.AbstractByteBuf.writeBytes(AbstractByteBuf.java:849)
at io.netty.buffer.AbstractByteBuf.writeBytes(AbstractByteBuf.java:841)
at io.netty.buffer.AbstractByteBuf.writeBytes(AbstractByteBuf.java:831)
at io.netty.handler.codec.ByteToMessageDecoder.channelRead(ByteToMessageDecoder.java:146)
```

原因： 当开启Spark动态调度特性时，NodeManager处理spark shuffle数据需要大量的内存，如果NodeManager相关的内存参数配置的不够大，便会出现“java.lang.OutOfMemoryError: Direct buffer memory”的错误。 解决：查看NodeManager的日志，看是否存在错误日志“OutOfMemoryError”。 不能提交任务 现象： Hadoop jar /usr/hdp/current/Hadoop-mapreduce-client/Hadoop-mapreduce-examples.jar pi -Dmapred.job.queue.name=default 10 100

```
9/08/23 16:24:56 INFO input.FileInputFormat: Total input files to process : 10
9/08/23 16:24:56 INFO mapreduce.JobSubmitter: number of splits:10
9/08/23 16:24:56 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1566544480145_0007
9/08/23 16:24:56 INFO mapreduce.JobSubmitter: Executing with tokens: [Kind: HDFS_DELEGATION_TOKEN, Service: 10.77.0.27:8020, Ident: (
oken for yarn: HDFS_DELEGATION_TOKEN owner=yarn/master1@HADOOP.COM, renewer=yarn, realUser=, issueDate=1566548696458, maxDate=1567153
96458, sequenceNumber=18, masterKeyId=9)]
9/08/23 16:24:56 INFO conf.Configuration: found resource resource-types.xml at file:/etc/hadoop/3.0.1.0-187/0/resource-types.xml
9/08/23 16:24:57 INFO impl.TimelineClientImpl: Timeline service address: slave1:8188
9/08/23 16:24:57 INFO impl.YarnClientImpl: Submitted application application_1566544480145_0007
9/08/23 16:24:57 INFO mapreduce.Job: The url to track the job: http://master1:8088/proxy/application_1566544480145_0007/
9/08/23 16:24:57 INFO mapreduce.Job: Running job: job_1566544480145_0007
9/08/23 16:24:58 INFO mapreduce.Job: Job job_1566544480145_0007 running in uber mode : false
9/08/23 16:24:58 INFO mapreduce.Job:  map 0% reduce 0%
9/08/23 16:24:58 INFO mapreduce.Job: Job job_1566544480145_0007 failed with state FAILED due to: Application application_156654448014
 0007 failed 2 times due to AM Container for appattempt_1566544480145_0007_000002 exited with  exitCode: -1000
ailing this attempt.Diagnostics: [2019-08-23 16:24:57.653]Application application_1566544480145_0007 initialization failed (exitCode=
5) with output: main : command provided 0
ain : run as user is yarn
ain : requested yarn user is yarn
equested user yarn is banned

r more detailed output, check the application tracking page: http://master1:8088/cluster/app/application_1566544480145_0007 Then cli
k on links to logs of each attempt.
 Failing the application.
9/08/23 16:24:58 INFO mapreduce.Job: Counters: 0
b job_1566544480145_0007 failed!
```

解决： 不用yarn用户提交任务 找不到或无法加载主类 现象： 找不到或无法加载主类 org.apache.Hadoop.mapreduce.v2.app.MRAppMaster



解决： 在mapred-site.xml 和 yarn-site.xml添加如下

```
<property>
<name>yarn.application.classpath</name>
<value>
/opt/Hadoop-2.6.0/etc/Hadoop,
/opt/Hadoop-2.6.0/share/Hadoop/common/*,
/opt/Hadoop-2.6.0/share/Hadoop/common/lib/*,
/opt/Hadoop-2.6.0/share/Hadoop/hdfs/*,
/opt/Hadoop-2.6.0/share/Hadoop/hdfs/lib/*,
/opt/Hadoop-2.6.0/share/Hadoop/mapreduce/*,
/opt/Hadoop-2.6.0/share/Hadoop/mapreduce/lib/*,
/opt/Hadoop-2.6.0/share/Hadoop/yarn/*,
/opt/Hadoop-2.6.0/share/Hadoop/yarn/lib/*
</value>
</property>
```

ZooKeeper session失效 现象： session失效，导致注册的watcher全部丢失。 原因： 如果zookeeper client与server在协商的超时时间内仍没有建立连接，当client与server再次建立连接时，由于session失效了，所有watcher已经被服务器端删除，从而导致所有的watcher需要重新注册。 session 失效，zookeeper client与server重连后所有watcher都会收到两次触发，第一次 wathetr state = 1，type = –1（state = 1表示正在连接中，type = –1 表示session事件）；第二次 watcher state = –112，type = –1（state = –112表示session失效）。 解决： 可以通过以下两种方法解决session失效问题 获取触发session失效watcher后，业务重新注册所有的watcher。 不能根本解决，但是可以减小session失效的概率。通过zookeeper client 与server设置更长的session超时时间。 zookeeperinit设置recvtimeout较长却没有效果 现象： zookeeperinit设置recvtimeout 100000ms，但客户端与服务端断开连接30s就session失效了。 原因： 关于session超时时间的确定：zookeeperinit中设置的超时时间并非真正的session超时时间，session超时时间需要 server与client协商，业务通过zoorecvtimeout(zhandlet* zh)获取server与client协商后的超时时间。服务端: minSessionTimeout (默认值为：tickTime * 2)，maxSessionTimeout(默认值为:tickTime * 20), ticktime的默认值为2000ms。所以session范围为4s ~ 40s 。客户端： sessionTimeout，无默认值，创建实例时设置recv_timeout 值。经常会认为创建zookeeper客户端时设置了sessionTimeout为100s，而没有改变server端的配置，默认值是不会生效的。原因：客户端的zookeeper实例在创建连接时，将sessionTimeout参数发送给了服务端，服务端会根据对应的 minSession/maxSession Timeout的设置，强制修改sessionTimeout参数，也就是修改为4s~40s 返回的参数。所以服务端不一定会以客户端的sessionTImeout做为session expire管理的时间。 解决： 增加zookeeperinit recvtimeout大小的同时，需要配置tickTime的值。 tickTime设置是在 conf/zoo.cfg 文件中

The number of milliseconds of each

ticktickTime=2000 （默认） 注： tickTime 心跳基本时间单位毫秒，ZK基本上所有的时间都是这个时间的整数倍。 zookeeper连接数问题 现象： zookeeper服务器都运行正常，而客户端连接异常。 原因： 这是由于zookeeper client连接数已经超过了zookeeper server获取的配置最大连接数。所以导致zookeeper client连接失败。 解决： 修改zookeeper安装目录下 conf/zoo.cfg文件。将maxClientCnxns参数改成更大的值。 zookeeper_init函数的使用 现象： 开发人员在调用zookeeperinit函数时，若返回一个非空句柄zhandlet *zh，则认为初始化成功，这样可能会导致后续操作失败。 原因： zhandlet *zookeeperinit(const char *host, watcherfn fn, int recvtimeout,const clientidt *clientid, void *context, int flags) 函数返回一个zookeeper客户端与服务器通信的句柄，通常我们仅仅根据返回句柄情况来判断zookeeper 客户端与zookeeper服务器是否建立连接。如果句柄为空则认为是失败，非空则成功。其实不然，zookeeperinit创建与ZooKeeper服务端通信的句柄以及对应于此句柄的会话，而会话的创建是一个异步的过程，仅当会话建立成功，zookeeper_init才返回一个可用句柄。 解决： 如何正确判断zookeepr_init初始化成功，可通过以下三种方式 判断句柄的state是否为ZOOCONNECTEDSTATE状态，通过zoostate(zh)判断状态值是否为ZOOCONNECTED_STATE。

```
void ensureConnected()
{
pthread_mutex_lock(&lock);
```

```
while (zoo_state(zh)!=ZOO_CONNECTED_STATE)
{
pthread_cond_wait(&cond, &lock);
}
pthread_mutex_unlock(&lock);
}
```

在zookeeperinit中设置watcher，当zookeeper client与server会话建立后，触发watcher，当 watcher 的state = 3（ZOOCONNECTEDSTATE）， type = –1（ZOOSESSION_EVENT）时，确认会话成功建立，此时zookeeper client 初始化成功，可进行后续操作。 业务上可以做保证，调用zookeeperinit返回句柄zh，通过该句柄尝试做zooexists()或zoogetdata（）等操作，根据操作结果来判断是否初始化成功。 ZooKeeper服务不断重启 现象：ZooKeeper界面中ZooKeeper角色实例一直处于“未知”状态 原因： ZooKeeper所在磁盘空间不足。 ZooKeeper服务数据文件被损坏。 解决： 检查ZooKeeper磁盘空间是否足够。 删除损坏的ZooKeeper数据文件。 ZooKeeper无法对外提供服务 现象： ZooKeeper服务异常，HDFS无法启动。 原因： ZooKeeper无法选主。 ZooKeeper集群中有机器IP冲突，或主机名冲突。 解决： 检查ZooKeeper安装并运行的实例是否为奇数个，如3个、5个。 恢复故障的ZooKeeper服务。 误删ZooKeeper文件后再恢复时失败 现象： 误删了ZooKeeper文件夹，然后通过FusionInsight Manager卸载并重装ZooKeeper实例来恢复。恢复结束后，界面显示进程的健康状态为“恢复中”。 原因： 用户误删除了ZooKeeper的pid文件，例如“Hadoop–omm–quorumpeer.pid”。但是ZooKeeper的进程还在后台运行。导致恢复此实例后，该实例无法启动，状态显示为“恢复中”。 例如，删除此pid文件前，ZooKeeper实例的进程号为“6419”，删除后去恢复ZooKeeper实例，会新建一个pid文件，且起一个新的进程，进程号为“6405”。此时由于“6419”进程未清除，导致“6405”进程无法启动。 解决： 使用root用户登录ZooKeeper实例所在节点，使用*kill –9*old_pid命令删除进程。 old_pid表示出现错误前的进程号，您可以通过*jps*命令查看。 Spark内存不足，无法退出应用程序 现象： 运行开发的应用程序时，当Driver内存不足时，Spark任务会挂起，不能退出应用程序。查看Container信息，出现以下错误：

```
Exception in thread "handle-read-write-executor-1" java.lang.OutOfMemoryError: Java heap space
at java.nio.HeapByteBuffer.<init>(HeapByteBuffer.java:57)
at java.nio.ByteBuffer.allocate(ByteBuffer.java:331)
at org.apache.spark.network.nio.Message$.create(Message.scala:90)
at org.apache.spark.network.nio.ReceivingConnection$Inbox.org$apache$spark$network$nio$Receivi
ngConnection$Inbox$$createNewMessage$1(Connection.scala:454)
at org.apache.spark.network.nio.ReceivingConnection$Inbox$$anonfun$1.apply(Connection.scala:46
4)
at org.apache.spark.network.nio.ReceivingConnection$Inbox$$anonfun$1.apply(Connection.scala:46
4)
at scala.collection.mutable.MapLike$class.getOrElseUpdate(MapLike.scala:189)
at scala.collection.mutable.AbstractMap.getOrElseUpdate(Map.scala:91)
at org.apache.spark.network.nio.ReceivingConnection$Inbox.getChunk(Connection.scala:464)
at org.apache.spark.network.nio.ReceivingConnection.read(Connection.scala:541)
at org.apache.spark.network.nio.ConnectionManager$$anon$6.run(ConnectionManager.scala:198)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
at java.lang.Thread.run(Thread.java:745)
```

原因： 因为提供给应用程序的Driver内存不足，导致内存空间出现full gc（全量垃圾回收）。但Garbage Collectors未能及时清理内存空间，使得full gc状态一直存在，导致了应用程序不能退出。 解决： 执行命令强制将任务退出，然后通过修改内存参数的方式解决内存不足的问题，使任务执行成功。 针对此类数据量大的任务，希望任务不再挂起，遇到内存不足时，直接提示任务运行失败。 org.apache.spark.shuffle.FetchFailedException 现象： 这种问题一般发生在有大量shuffle操作的时候,task不断的failed,然后又重执行，一直循环下去，非常的耗时。 missing output location org.apache.spark.shuffle.MetadataFetchFailedException: Missing an output location for shuffle 0 shuffle fetch faild org.apache.spark.shuffle.FetchFailedException: Failed to connect to spark047215/192.168.47.215:50268 当前的配置为每个executor使用1cpu,5GRAM,启动了20个executor 解决： 一般遇到这种问题提高executor内存即可,同时增加每个executor的cpu,这样不会减少task并行度。 spark.executor.memory 15G spark.executor.cores 3 spark.cores.max 21 启动的execuote数量为:7个 execuoteNum = spark.cores.max/spark.executor.cores 每个executor的配置： 3core,15G RAM 消耗的内存资源为:105G RAM 15G*7=105G 可以发现使用的资源并没有提升，但是同样的任务原来的配置跑几个小时还在卡着，改了配置后几分钟就结束了。 spark.executor.memoryOverhead 现象： 堆外内存（默认是executor内存的10%），当数据量比较大的时候，如果按默认的就会有下面的异常，导致程序崩溃 Container killed by YARN for exceeding memory limits. 1.8 GB of 1.8 GB physical memory used. Consider boosting spark.yarn.executor.memoryOverhead. 解决： 具体值根据实际情况配置 新版： --conf spark.executor.memoryOverhead=2048 旧版： --conf spark.yarn.executor.memoryOverhead=2048 新版如果用旧版，会: WARN SparkConf: The configuration key 'spark.yarn.executor.memoryOverhead' has been deprecated as of Spark 2.3 and may be removed in the future. Please use the new key 'spark.executor.memoryOverhead' instead. No more replicas available for rdd 现象：

```
19/01/08 12:36:46 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_3250_73 !
19/01/08 12:36:46 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_12_38 !
19/01/08 12:36:46 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_3250_38 !
19/01/08 12:36:46 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_3250_148 !
19/01/08 12:36:46 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_3250_6 !
19/01/08 12:36:46 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_3250_112 !
19/01/08 12:36:46 WARN BlockManagerMasterEndpoint: No more replicas available for rdd_12_100 !
```

解决： 增大executor的内存 --executor-memory 4G Executor&Task Lost 现象： 因为网络或者gc的原因,worker或executor没有接收到executor或task的心跳反馈 executor lost WARN TaskSetManager: Lost task 1.0 in stage 0.0 (TID 1, aa.local): ExecutorLostFailure (executor lost) task lost WARN TaskSetManager: Lost task 69.2 in stage 7.0 (TID 1145, 192.168.47.217): java.io.IOException: Connection from /192.168.47.217:55483 closed 各种timeout java.util.concurrent.TimeoutException: Futures timed out after [120 second ERROR TransportChannelHandler: Connection to /192.168.47.212:35409 has been quiet for 120000 ms while there are outstanding requests. Assuming connection is dead; please adjust spark.network.timeout if this is wrong 解决： 提高 spark.network.timeout 的值，根据情况改成300(5min)或更高。 默认为 120(120s),配置所有网络传输的延时，如果没有主动设置以下参数，默认覆盖其属性 spark.core.connection.ack.wait.timeout spark.akka.timeout spark.storage.blockManagerSlaveTimeoutMs spark.shuffle.io.connectionTimeout spark.rpc.askTimeout or spark.rpc.lookupTimeout 倾斜 现象： 大多数任

务都完成了，还有那么一两个任务怎么都跑不完或者跑的很慢。 分为数据倾斜和task倾斜两种。 解决： 数据倾斜数据倾斜大多数情况是由于大量null值或者””引起，在计算前过滤掉这些数据既可。 sqlContext.sql(”...where col is not null and col != ””) 任务倾斜 task倾斜原因比较多，网络io,cpu,mem都有可能造成这个节点上的任务执行缓慢，可以去看该节点的性能监控来分析原因。以前遇到过同事在spark的一台worker上跑R的任务导致该节点spark task运行缓慢。 或者可以开启spark的推测机制，开启推测机制后如果某一台机器的几个task特别慢，推测机制会将任务分配到其他机器执行，***Spark会选取最快的作为最终结果。 spark.speculation true spark.speculation.interval 100 – 检测周期，单位毫秒; spark.speculation.quantile 0.75 – 完成task的百分比时启动推测 spark.speculation.multiplier 1.5 – 比其他的慢多少倍时启动推测。 大数据计算时出现“Channel空闲超时”在10节点集群，30T数据量下，执行tpcds测试时，出现如下错误。 Connection to 10.10.10.1 has been quiet for 123450 ms while there are still 5 outstanding requests. Assuming connection is dead; please adjust spark.network.timeout if this is wrong. 原因： 当Map Server繁忙时，Reduce Client发出请求，得不到响应。当等待时间超过一个阈值时，出现错误。默认的时间为120秒。 解决： 上述问题是在request个数很大时发生的，属于正常现象。解决措施有两种： 将spark.shuffle.io.connectionTimeout参数调大。10节点、30T数据的TPCDS测试中设置为2000s，运行正常。此参数与spark.network.timeout配合使用，优先使用spark.shuffle.io.connectionTimeout参数设置的值。如果spark.shuffle.io.connectionTimeout未设置，则使用spark.network.timeout的参数值。 调大spark.shuffle.io.serverThreads来解决，将此参数的值设置为core个数的两倍。 Spark任务运行失败，ApplicationMaster出现物理内存溢出异常 现象： 在YARN上运行Spark任务失败，ApplicationMaster出现物理内存溢出异常。报错内容如下： 原因： 日志中显示“Killing container”，直接原因是物理内存使用超过了限定值，YARN的NodeManager监控到内存使用超过阈值，强制终止该container进程。 解决： 在Spark客户端“spark-defaults.conf”配置文件中增加如下参数，或者在提交命令时添加--conf指定如下参数，来增大memoryOverhead。 spark.yarn.driver.memoryOverhead：设置堆外内存大小（cluster模式使用）。 spark.yarn.am.memoryOverhead：设置堆外内存大小（client模式使用）。 执行Spark SQL语句时，出现joinedRow.isNullAt的空指针异常 现象： 在执行Spark SQL语句时，出现“joinedRow.isNullAt”的空指针异常，异常信息如下所示。

```
6/09/08 11:04:11 WARN TaskSetManager: Lost task 1.0 in stage 7.0 (TID 10, vm1, 1): java.lang.NullPointerException
at org.apache.spark.sql.catalyst.expressions.JoinedRow.isNullAt(JoinedRow.scala:70)
at org.apache.spark.sql.catalyst.expressions.GeneratedClass$SpecificMutableProjection.apply(Unknown Source)
at org.apache.spark.sql.execution.aggregate.TungstenAggregationIterator$$anonfun$generateProcessRow$1.apply(TungstenAggregationIterator.scala:194)
at org.apache.spark.sql.execution.aggregate.TungstenAggregationIterator$$anonfun$generateProcessRow$1.apply(TungstenAggregationIterator.scala:192)
at org.apache.spark.sql.execution.aggregate.TungstenAggregationIterator.processInputs(TungstenAggregationIterator.scala:372)
at org.apache.spark.sql.execution.aggregate.TungstenAggregationIterator.start(TungstenAggregationIterator.scala:626)
at org.apache.spark.sql.execution.aggregate.TungstenAggregate$$anonfun$doExecute$1.org$apache$spark$sql$execution$aggregate$TungstenAggregate$$anonfun$$executePartition$1(TungstenAggregate.scala:135)
```

```
at org.apache.spark.sql.execution.aggregate.TungstenAggregate$$anonfun$doExecute$1$$anonfun
$3.apply(TungstenAggregate.scala:144)
at org.apache.spark.sql.execution.aggregate.TungstenAggregate$$anonfun$doExecute$1$$anonfun
$3.apply(TungstenAggregate.scala:144)
at org.apache.spark.rdd.MapPartitionsWithPreparationRDD.compute(MapPartitionsWithPreparationR
DD.scala:64)
at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:334)
at org.apache.spark.rdd.RDD.iterator(RDD.scala:267)
at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38)
at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:334)
at org.apache.spark.rdd.RDD.iterator(RDD.scala:267)
at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:75)
at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:42)
at org.apache.spark.scheduler.Task.run(Task.scala:90)
at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:253)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
```

原因： 由如下日志信息可知，该错误是由于内存不足，导致buffer在申请内存时申请失败返回为null，对null进行操作就返回了空指针异常。 当集群中内存相关的关键配置项的值设置的比较小时，例如设置为如下所示的值：

```
spark.executor.cores = 8
spark.executor.memory = 512M
spark.buffer.pageSize = 16M
此时，执行任务会出现内存申请失败返回null的异常，关键日志如下：
6/09/08 11:04:11 WARN TaskSetManager: Lost task 1.0 in stage 7.0 (TID 10, vm1, 1): java.lang.NullPoi
nterException
at org.apache.spark.sql.catalyst.expressions.JoinedRow.isNullAt(JoinedRow.scala:70)
```

解决： 根据executor日志提示信息，您可以通过调整如下两个参数解决此问题。在客户端的“spark-defaults.conf”配置文件中调整如下参数。 spark.executor.memory：增加executor的内存，即根据实际业务量，适当增大“spark.executor.memory”的参数值。需满足公式：spark.executor.memory > spark.buffer.pageSize * (num * spark.executor.cores) / spark.shuffle.memoryFraction / spark.shuffle.safetyFraction
spark.executor.cores：减小executor的核数，即减小executor-cores的参数值。需满足公式：spark.executor.cores < spark.executor.memory / spark.buffer.pageSize / num * spark.shuffle.memoryFraction * spark.shuffle.memoryFraction。 在调整这两个参数时，需满足spark.executor.memory * spark.shuffle.memoryFraction *spark.shuffle.safetyFraction / (num * spark.executor.cores) > spark.buffer.pageSize公式，在内存充足的情况下，建议直接将常数num设置为16，可解决所有场景遇到的内存问题。 出现java.io.FileNotFoundException异常 现象： 任务开始执行时，Driver端出现以下错误。

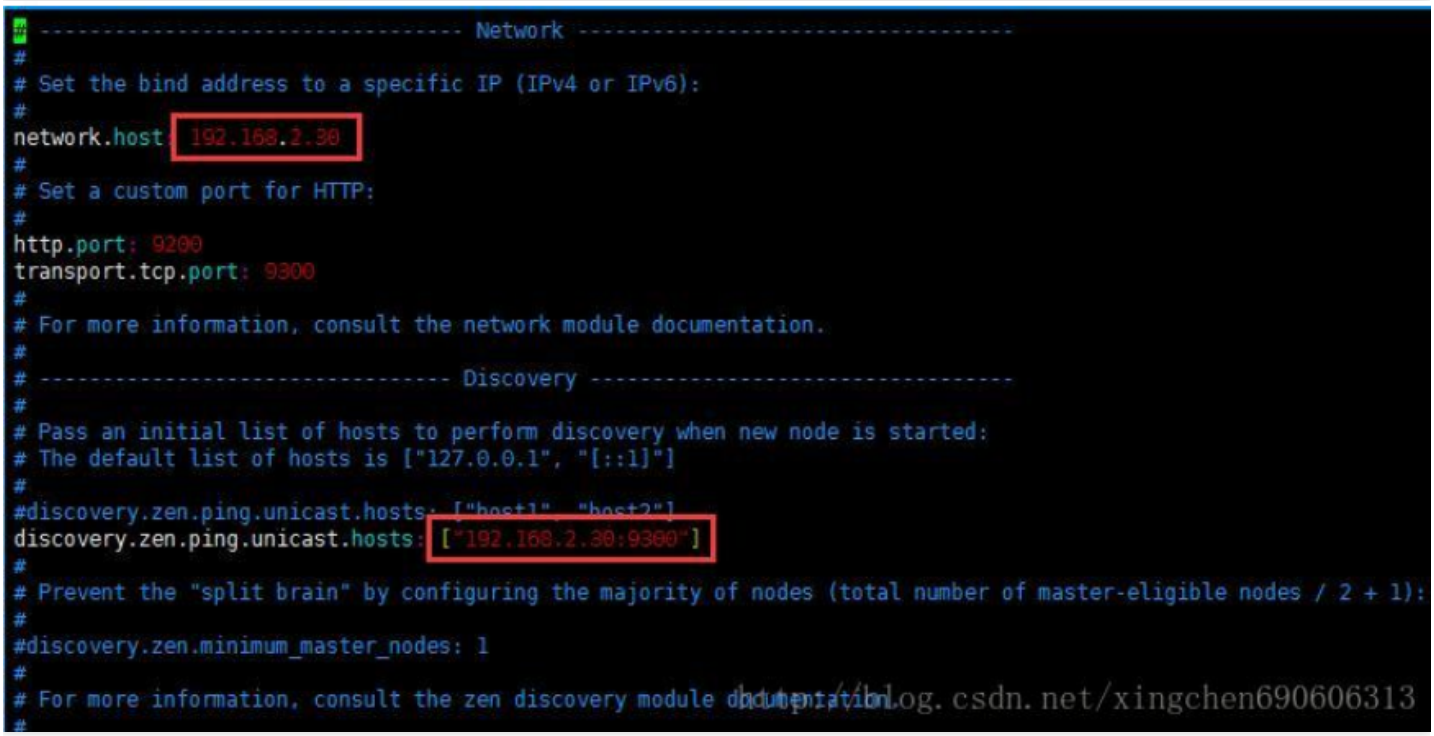
```
java.io.FileNotFoundException: /opt/huawei/Bigdata/FusionInsight/FusionInsight-Spark-1.5.0/spark/c
onf/log4j.properties (No such file or directory)
```

```
at java.io.FileInputStream.open(Native Method)
at java.io.FileInputStream.<init>(FileInputStream.java:146)
at java.io.FileInputStream.<init>(FileInputStream.java:101)
at sun.net.www.protocol.file.FileURLConnection.connect(FileURLConnection.java:90)
```

原因：“spark.driver.extraJavaOption”配置值仅适用于Driver端的JVM参数配置。在使用yarn_client的模式时，需要将此配置值替换为客户端的实际路径。 当出现此错误时，不影响任务的执行。 解决： 在客户端的安装目录下，修改“spark-defaults.conf”文件，将“spark.driver.extraJavaOption”客户端机器的log4j的配置文件路径和名称。 cd $/Spark/spark/conf vi spark-defaults.conf 修改配置参数（如果该参数的值为空，执行命令时，会使用本地默认的log4j的配置路径）： spark.driver.extraJavaOptions = -Dlog4j.configuration=file:/home/client/Spark/spark/conf/log4j.properties -Dlog4j.configuration.watch=false ElasticSearch 无法指定被请求的地址 现象：

```
org.elasticsearch.bootstrap.StartupException:BindTransportException[Failed to bind to [9300]]; nested: BindException[无法指定被请求的地址];
        atorg.elasticsearch.bootstrap.Elasticsearch.init(Elasticsearch.java:127)~[elasticsearch-5.4.1.jar:5.4.1]
        atorg.elasticsearch.bootstrap.Elasticsearch.execute(Elasticsearch.java:114)~[elasticsearch-5.4.1.jar:5.4.1]
        atorg.elasticsearch.cli.EnvironmentAwareCommand.execute(EnvironmentAwareCommand.java:67)~[elasticsearch-5.4.1.jar:5.4.1]
        atorg.elasticsearch.cli.Command.mainWithoutErrorHandling(Command.java:122)~[elasticsearch-5.4.1.jar:5.4.1]
        atorg.elasticsearch.cli.Command.main(Command.java:88)~[elasticsearch-5.4.1.jar:5.4.1]
        atorg.elasticsearch.bootstrap.Elasticsearch.main(Elasticsearch.java:91)~[elasticsearch-5.4.1.jar:5.4.1]
        atorg.elasticsearch.bootstrap.Elasticsearch.main(Elasticsearch.java:84)~[elasticsearch-5.4.1.jar:5.4.1]
Caused by:org.elasticsearch.transport.BindTransportException: Failed to bind to [9300]
        atorg.elasticsearch.transport.TcpTransport.bindToPort(TcpTransport.java:769)~[elasticsearch-5.4.1.jar:5.4.1]
        atorg.elasticsearch.transport.TcpTransport.bindServer(TcpTransport.java:734)~[elasticsearch-5.4.1.jar:5.4.1]
        atorg.elasticsearch.transport.netty4.Netty4Transport.doStart(Netty4Transport.java:173)~[?:?]
```

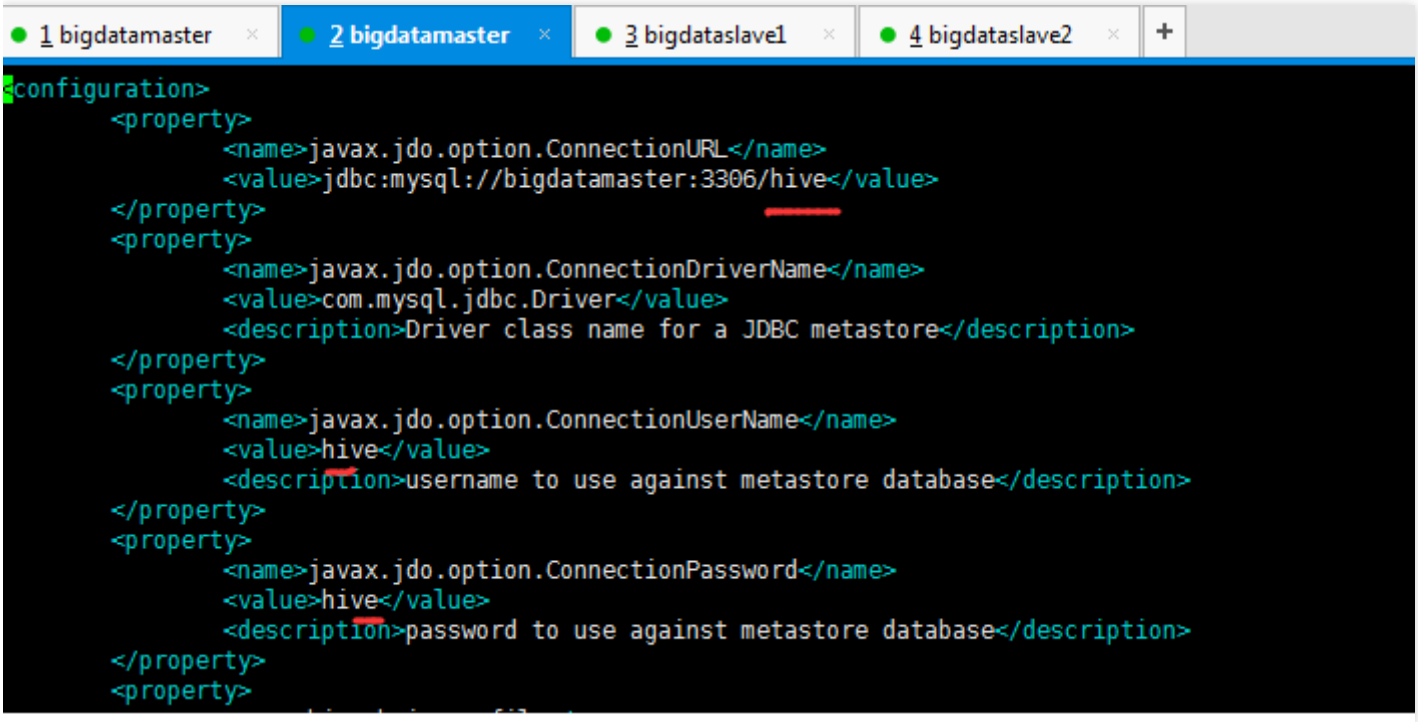
原因：elasticsearch.yml文件的参数配置不正确 解决： 编辑node节点对应的配置文件，例如： （1）在命令行输入：vim /usr/elk/elasticsearch/elasticsearch-master/config/elasticsearch.yml （2）打开文件后，把文件中的这两个地方的IP地址改成ES所在服务器的IP地址即可。



org.elasticsearch.bootstrap.StartupException: java.lang.RuntimeException: can not run elasticsearch as root 原因： 因为安全问题elasticsearch不让用root用户直接运行，所以要创建新用户 解决： 创建一个单独的用户用来运行ElasticSearch max file descriptors [4096] for elasticsearch process is too low, increase to at least [65536] 原因： 普通用户执行问题 解决： 切换到root用户，编辑/etc/security/limits.conf 添加 elk hard nofile 65536 elk soft nofile 65536 (elk是用户名) org.elasticsearch.transport.RemoteTransportException: Failed to deserialize exception response from stream 原因： es节点之间的JDK版本不一样 解决： 统一JDK版本和环境 由gc引起节点脱离集群 原因： 因为gc时会使jvm停止工作，如果某个节点gc时间过长，master ping3次(zen discovery默认ping失败重试3次)不通后就会把该节点剔除出集群，从而导致索引进行重新分配。 解决： 优化gc，减少gc时间。 调大zen discovery的重试次数(es参数：pingretries)和超时时间(es参数：pingtimeout)。后来发现根本原因是有个节点的系统所在硬盘满了。导致系统性能下降。 ES索引设置不当 集群名称配置 ES启动的默认群集名称称为elasticsearch。如果群集中有许多节点，最好保持命名标志尽可能一致，例如：cluster.name:app_es_production node.name:app_es_node_001 集群恢复设置 节点的恢复设置也很重要。假设群集中的某些节点由于故障而重新启动，并且某些节点在其他节点之后重启。为了使所有这些节点之间的数据保持一致，我们必须运行一致性程序，以使所有集群保持一致状态。 举例1：只要10个数据或主节点已加入群集，即可恢复。 gateway.recover_after_nodes：10 举例2：集群中期待启动节点达到20个以及时间超过7分钟后，集群重启或恢复。 gateway.expected_nodes：20 gateway.recover_after_time：7m 使用正确的配置，可能需要数小时的恢复缩减到只需要分钟级，极大提高工作效率。 防脑裂配置 minimum_master_nodes对于群集稳定性非常重要。它们有助于防止脑裂。此设置的建议值为（N / 2）+ 1 ，其中N是候选主节点的节点数。有了这个，如果你有10个可以保存数据并成为主数据的候选主节点，那么该值将是6。如果您有三个专用主节点和1,000个数据节点，则该值为两个（仅计算候选主节点）： discovery.zen.minimummasternodes：2 Solr Solr客户端无法实时索引 原因： Solr客户端写索引失败，异常信息如下： Error from server at

https://192.168.152.14:21104/solr/col_test_shard1_replica2: Exception writing document id 6907a8e1154a4bdb9a4238cd7bb874c4 to the index; possible analysis error., retry? 0 原因： 索引文件损坏。 索引文件所在磁盘或文件系统故障。 解决： 查看“/var/log/Bigdata/solr”下SolrServer1的日志“solr-主机名.log”。在日志中搜索关键词“IndexWriter AlreadyCloseException”。 Hue Hue的页面打不开 现象： Hue服务组件的Hue WebUI页面打不开。 原因： 端口未对外开放。 浮动IP配置错误。 解决： 在浏览器所在的本地机器，通过命令行执行telnet指令，查看端口是否开放。 在Hue所在节点执行ifconfig，查看配置的浮动IP是否生效。 恢复该故障大概需要10分钟。 HUE中Hive 查询有问题 现象： Could not connect to localhost:10000 或者 Could not connect to bigdatamaster:10000

解决： hive-site.xml里的配置信息



在hue里面查看HDFS文件浏览器报错 现象： 当前用户没有权限查看，cause:org.apache.Hadoop.ipc.StandbyException: Operation category READ is not supported in state standby 解决： Web页面查看两个NameNode状态，是不是之前的namenode是standby状态了. 我现有的集群就是这种情况, 由于之前的服务是master1起的, 挂了之后自动切换到master2, 但是hue中webhdfs还是配置的master1,导致在hue中没有访问权限. hue使用mysql作为元数据库 现象： hue默认使用sqlite作为元数据库，不推荐

在生产环境中使用。会经常出现database is lock的问题。 解决： 先在mysql里面创建数据库hue，然后修改hue.ini

```
[[database]]

engine=mysql
host=slave1
port=3306
user=Hadoop
password=Hadoop
name=hue
```

完成以上的这个配置，启动Hue,通过浏览器访问，会发生错误，原因是mysql数据库没有被初始化 DatabaseError: (1146, ”Table ’hue.desktop_settings’ doesn’t exist”) 初始化数据库 cd hue-release-3.11.0/build/env/ bin/hue syncdb bin/hue migrate 执行完以后，可以在mysql中看到，hue相应的表已经生成。 启动hue, 能够正常访问了 开启hbase查询模块 解决： 需要在hbase集群已经启动的基础上，再启动thrift，默认端口为9090 hbase-daemon.sh start thrift 修改配置hue.ini的配置文件 [hbase] hbase_clusters=(Cluster|master1:9090) hbase_conf_dir=/usr/hbase-1.1.6/conf Cluster 为在Hue展现中的名字，可配置多个hbase集群 master1:9090 hbase启动的thrift主机及端口 hive查询报错TTransportException 现象： 在hue里面查询hive数据，之前一直使用正常，突然就查询不了，一直转不出结果最后页面可能还会报504 Gateway Time-out。查看日志定位error: Could not connect to localhost:9090 (code THRIFTTRANSPORT): TTransportException(’Could not connect to localhost:9090’,) 解决： 看来可能是thrift服务器通信的原因，重启了hive的hiveserver2进程，问题解决。 nohup bin/hiveserver2 & Filesystem root ‘/’ should be owned by ‘hdfs’ 原因： hue 文件系统根目录“/”应归属于“hdfs” 解决： 修改文件desktop/libs/Hadoop/src/Hadoop/fs/webhdfs.py 中的 DEFAULTHDFSSUPERUSER = ‘hdfs’ 更改为你的Hadoop用户 安装启动异常 scp传秘钥的时候报错Permission denied (publickey).

```
[root@vm172-31-100-17 .ssh]# ssh-copy-id -i /root/.ssh/id_rsa.pub root@slave1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
Permission denied (publickey).
```

解决：将秘钥先复制到跳板机上，然后再从跳板机复制到你所要到的目的机 [root@vm172-31-100-17 .ssh]# scp -p ./id_rsa.pub root@10.69.64.93:/root/

```
[root@vm172-31-100-17 .ssh]# scp -p ./id_rsa.pub root@10.69.64.93:/root/
The authenticity of host '10.69.64.93 (10.69.64.93)' can't be established.
ECDSA key fingerprint is SHA256:Pd7HPeOyy7z4DutvDiQjiS5hJZc9HcERbdyrq5mEMJE.
ECDSA key fingerprint is MD5:5d:2a:41:de:d7:3d:4b:b8:0d:09:1b:f8:49:c7:6d:77.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.69.64.93' (ECDSA) to the list of known hosts.
root@10.69.64.93's password:
id_rsa.pub
You have new mail in /var/spool/mail/root
```

[root@jumpserver ~]# scp –i /root/.ssh/id_rsa_ccb /root/id_rsa.pub root@172.31.100.6:/root/.ssh/

```
root@jumpserver ~]# scp -i /root/.ssh/id_rsa_ccb /root/id_rsa.pub root@172.31.100.6:/root/.ssh/
d_rsa.pub
root@jumpserver ~]# cd /root/.ssh/
root@jumpserver .ssh]# ll
otal 84
rwx------ 1 root root  2829 Jul 11 13:16 authorized_keys
rw------- 1 root root   887 Jun 17 18:14 id_ras_mpp
rwx------ 1 root root  1679 Mar  5 19:59 id_rsa
rwx------ 1 root root   887 Apr 18 10:23 id_rsa_ai
rwx------ 1 root root   887 Apr  1 12:52 id_rsa_ccb
rwx------ 1 root root  1675 Apr  3 19:16 id_rsa_ccb_test
rw------- 1 root root   887 Apr 13 19:03 id_rsa_jzh
rwx------ 1 root root  1679 Oct 29  2018 id_rsa_mhc
rw------- 1 root root   887 Jul  8 13:58 id_rsa_mpp
rwx------ 1 root root   408 Mar  5 19:59 id_rsa.pub
rwx------ 1 root root 43004 Jul 22 10:54 known_hosts
```

[root@vm172–31–100–6 .ssh]# cat ./id_rsa.pub >> authorized_keys

```
[root@slave3 /]# cd /root/.ssh/
[root@slave3 .ssh]# cat ./id_rsa.pub >> authorized_keys
[root@slave3 .ssh]# ll
total 16
-rw------- 1 root root 2288 Jul 22 11:19 authorized_keys
-rw------- 1 root root 1675 Jul  1 16:12 id_rsa
-rw-r--r-- 1 root root  394 Jul 22 11:17 id_rsa.pub
-rw-r--r-- 1 root root  706 Jul  3 18:16 known_hosts
[root@slave3 .ssh]# vim authorized_keys
[root@slave3 .ssh]#
```

```
[root@master-node1 ~]# ssh slave1-node2
Last login: Thu Jul 25 17:13:24 2019 from 172.31.0.22
[root@slave1-node2 ~]# exit
logout
```

ambari启动：ERROR: Exiting with exit code –1.

```
[root@master-node1 /]# ambari-server start
Using python  /usr/bin/python
Starting kde-server
KDE Server running with administrator privileges.
Organizing resource files at /var/lib/ambari-server/resources...
Ambari database consistency check started...
Server PID at: /var/run/ambari-server/ambari-server.pid
Server out at: /var/log/ambari-server/ambari-server.out
Server log at: /var/log/ambari-server/ambari-server.log
Waiting for server start.....................................ERROR: Exiting with exit code -1.
REASON: Ambari Server java process has stopped. Please check the logs for more information.
```

原因：mariadb数据库没有设置开机自启动 解决： [root@master–node1 /]# systemctl start mariadb
[root@master–node1 /]# systemctl enable mariadb

常见问题

产品介绍问题

最近更新时间: 2021-09-17 16:47:13

Q：什么是托管Hadoop?

A：托管Hadoop是一个可伸缩的通过用数据计算和分析平台，它以Apache Hadoop和Apache Spark两大数据计算框架为基础，通过自动调度弹性计算服务，帮助您快速构建分布式数据分析系统。托管HADOOP提供了强大的扩展能力和弹性伸缩能力，消除了Hadoop安装部署成本和管理复杂性，可以使您不必关注基础架构管理，而更加专注数据分析处理本身，任何的开发者或者公司只需要较低的成本就可以进行大规模的数据分析和处理工作。

Q：目前托管Hadoop支持哪几种组件?

A：托管HADOOP除集成了基础的Hadoop组件外，同时集成了Spark， Hbase，Flink， Kafka， Elasticsearch等生态组件，帮助您轻松构建复杂的大数据分析系统，满足批量计算、流式处理、消息队列、交互式查询、NoSQL等多种业务场景的需求。

产品使用问题

最近更新时间: 2021-09-17 16:47:13

Q: 客户端工具如何部署?

A：在创建集群的时候可选择同时创建客户端节点，客户端节点默认与Hadoop集群节点免密登陆。客户端节点上以默认安装beeline, Hbase, Shell，且已配置环境变量，可直接使用该工具以命令行方式访问集群。

Q：如何管理运维Hadoop集群?

A：每个托管Hadoop集群会默认部署一套托管HADOOP管理系统，该管理系统可通过访问集群主Master节点的IP地址加端口8080访问，用户名密码都默认为kmr，该管理系统可查看集群监控状态信息、性能指标项监控信息以及告警等信息。

Q：托管Hadoop集群规模是否有限制?

A：由于托管Hadoop的管理控制节点仅能部署在两台云主机上，且不支持横向扩展，因此建议单个托管Hadoop集群规模不超过100台。

Q：能否通过SparkSQL直接访问Hive表数据?

A：不能，由于Hive和Spark原数据相互独立，分别存储于不同数据库之中，因此默认无法通过SparkSQL直接访问Hive表数据。