阿里云
E-MapReduce

用户指南

文档版本：20181219

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止：** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告：** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明：** 您也可以通过按**Ctrl** + **A**选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

法律声明.....I
通用约定.....I
1 角色授权..... 1
2 集群规划..... 5
2.1 实例类型..... 5
2.2 Gateway实例..... 6
2.3 ECS实例说明..... 6
2.4 存储说明..... 7
2.5 本地盘机型支持.....9
2.6 经典网络与VPC互访..... 10
3 集群.....11
3.1 创建集群..... 11
3.2 扩容集群..... 15
3.3 集群列表..... 17
3.4 释放集群..... 18
3.5 集群详情..... 19
3.6 用户管理..... 22
3.7 服务列表..... 23
3.8 集群脚本..... 24
3.9 集群续费..... 25
3.10 集群自动续费管理.....27
3.11 安全组.....28
3.12 节点变配..... 29
3.13 磁盘扩容..... 30
3.14 切换支付类型..... 33
3.15 移除异常节点..... 33
4 作业.....34
4.1 Hadoop MapReduce 作业配置.....34
4.2 Hive 作业配置.....36
4.3 Pig 作业配置.....37
4.4 Spark 作业配置.....39
4.5 Spark SQL 作业配置.....41
4.6 Shell 作业配置.....42
4.7 Sqoop 作业配置.....43
4.8 作业操作..... 44
4.9 作业日期设置..... 44

5 执行计划 47
5.1 创建执行计划 47
5.2 管理执行计划 49
5.3 执行计划列表 50
5.4 作业结果和日志查看 51
5.5 多执行计划并行执行 52
6 报警管理 54
6.1 集群报警管理 54
7 软件配置 56
8 引导操作 61
9 专有网络 63
10 Python 使用说明 65
11 开源组件介绍 66
11.1 Hue 使用说明 66
11.2 Oozie 使用说明 67
11.3 Presto 使用说明 70
11.4 Zeppelin 使用说明 71
11.5 ZooKeeper 使用说明 71
11.6 Kafka使用说明 72
11.6.1 Kafka 快速入门 72
11.6.2 Kafka 跨集群访问 74
11.6.3 Kafka Ranger使用说明 76
11.6.4 Kafka SSL使用说明 80
11.6.5 Kafka Manager 使用说明 82
11.6.6 Kafka 常见问题 83
11.7 Druid使用说明 84
11.7.1 Druid简介 84
11.7.2 快速入门 87
11.7.3 数据格式描述文件 96
11.7.4 Tranquility 99
11.7.5 Kafka Indexing Service 102
11.7.6 Superset 105
11.7.7 常见问题 112
11.8 Presto 115
11.8.1 连接器 115
11.8.2 产品简介 122
11.8.3 快速入门 124
11.8.4 数据类型 131

11.8.5 常用函数和操作符.....134
11.8.6 SQL语句.....158
11.8.7 技术支持.....191
11.9 TensorFlow使用说明.....191
11.10 Knox 使用说明.....193
11.11 Flume使用说明.....196
12 OSS 数据权限隔离.....205
13 SSH 登录集群.....207
14 创建Gateway.....212
15 MetaService.....218
16 交互式工作台.....220
16.1 交互式工作台简介.....220
16.2 交互式工作台操作说明.....225
16.3 交互式工作台示例.....231
16.3.1 银行员工信息查询示例.....231
16.3.2 视频播放数据示例.....231
17 表管理.....234
18 Kerberos认证.....241
18.1 Kerberos简介.....241
18.2 兼容MIT Kerberos认证.....246
18.3 RAM认证.....249
18.4 LDAP认证.....252
18.5 执行计划认证.....253
18.6 跨域互信.....254
19 组件授权.....257
19.1 HDFS授权.....257
19.2 YARN授权.....259
19.3 Hive授权.....264
19.4 HBase授权.....268
19.5 Kafka授权.....271
19.6 RANGER.....276
19.6.1 Ranger简介.....276
19.6.2 HDFS配置.....278
19.6.3 Hive配置.....283
19.6.4 HBase配置.....287
19.6.5 Kafka配置.....292
20 集群容灾能力说明.....297
20.1 EMR集群容灾能力.....297

21 资源池使用说明.....298
22 弹性伸缩..... 301
22.1 弹性伸缩概述..... 301
22.2 按时间伸缩规则配置..... 301
22.3 弹性伸缩抢占式实例..... 303
22.4 弹性伸缩记录..... 305
23 数据开发..... 306
23.1 项目管理..... 306
23.2 作业编辑..... 307
23.3 临时查询..... 309
23.4 工作流编辑.....311

1 角色授权

在用户开通 E-MapReduce 服务时，需要授予一个名称为“AliyunEMRDefaultRole”的系统默认角色给 E-MapReduce 的服务账号，当且仅当该角色被正确授予后，E-MapReduce 才能正常地调用相关服务（ECS，OSS 等），创建集群以及保存日志等。

角色授权流程

1. 当用户创建集群或创建按需执行计划时，如果没有正确地给 E-MapReduce 的服务账号授予默认角色，则会看到如下提示，此时需单击前往 **RAM** 进行授权，进行角色授权。

2. 单击同意授权，将默认角色 AliyunE-MapReduceDefaultRole 授予给 E-MapReduce 的服务账号。

3. 当完成以上授权步骤后，用户需刷新 E-MapReduce 的控制台，然后就可以进行操作了。如果您想查看 AliyunE-MapReduceDefaultRole 相关的详细策略信息，可以登录 RAM 的控制台查看，也可以单击查看链接。

默认角色包含的权限内容

默认角色 AliyunEMRDefaultRole 包含的权限信息如下：

- ECS 相关权限：

权限名称（**Action**）	权限说明
ecs:CreateInstance	创建 ECS 实例
ecs:RenewInstance	ECS 实例续费
ecs:DescribeRegions	查询 ECS 地域信息
ecs:DescribeZones	查询 Zone 信息
ecs:DescribeImages	查询镜像信息
ecs:CreateSecurityGroup	创建安全组
ecs:AllocatePublicIpAddress	分配公网 IP
ecs:DeleteInstance	删除机器实例
ecs:StartInstance	启动机器实例
ecs:StopInstance	停止机器实例
ecs:DescribeInstances	查询机器实例
ecs:DescribeDisks	查询机器相关磁盘信息
ecs:AuthorizeSecurityGroup	设置安全组入规则
ecs:AuthorizeSecurityGroupEgress	设置安全组出规则
ecs:DescribeSecurityGroupAttribute	查询安全组详情
ecs:DescribeSecurityGroups	查询安全组列表信息

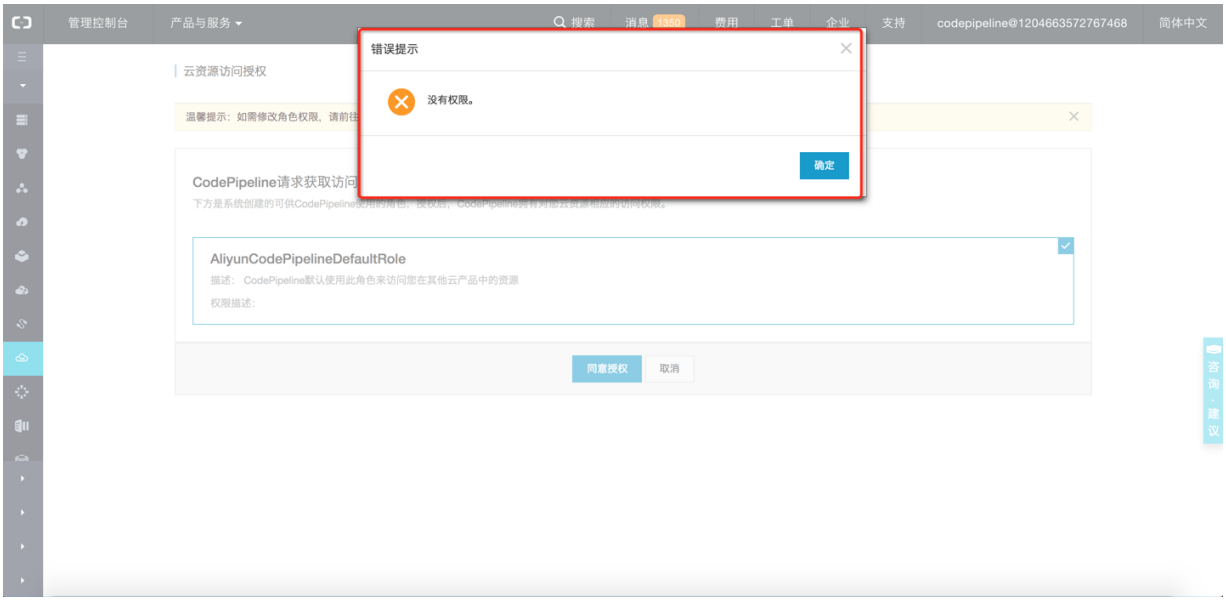
- OSS 相关权限：

权限名称（**Action**）	权限说明
oss:PutObject	上传文件或文件夹对象
oss:GetObject	获取文件或文件夹对象
oss:ListObjects	查询文件列表信息

子账户**passRole**权限

为确保子账号登录控制台后能正常使用CodePipeline控制台的功能，除授予CodePipeline相应的访问权限外还需为子账户授权passRole权限。

如果主账户没有为子账户授权passRole权限， 子账户登录CodePipeline控制台后会提示一下信息：



新建 passRole授权策略：示例如下：

```
{
      "Version": "1",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": "ram:PassRole",
          "Resource": "*",
          "Condition": {
            "StringEquals": {
              "acs:Service": "pipeline.aliyuncs.com"
            }
          }
        }
      ]
```

```
    }
```

授权给子账户passRole权限后，子账户就可以正确访问CodePipeline控制台。

2 集群规划

2.1 实例类型

EMR集群中由多个不同的节点实例类型组成，他们分别是主实例节点（Master），核心实例节点（Core）和计算实例节点（Task），每一种不同的实例在部署的时候会部署完全不同的服务进程，以完成完全不同的任务。举例来说，我们会在主实例节点（Master）上部署 Hadoop HDFS 的 Namenode 服务，Hadoop YARN 的 ResourceManager 服务，而在核心实例节点（Core）上部署 Datanode 服务，Hadoop YARN 的 NodeManager 服务，在计算实例节点（Task）顾名思义，只进行计算，部署 Hadoop YARN 的 NodeManager 服务，不部署任何 HDFS 相关的服务。

在创建集群的时候需要确定对应的三种实例类型的 ECS 规格，相同实例类型的 ECS 在同一个实例组内。并且可以在后期通过扩容来扩容对应实例组内的机器数量（主实例组除外）。

说明：

计算实例节点（Task）从3.2.0及以后版本开始支持。

Master 主实例

主实例是集群服务的管控等组件的部署的节点，举例来说，Hadoop YARN 的 ResourceManager 就部署在主实例节点上。用户可以通过 SSH 的方式连接到主实例上，通过软件的 Web UI 来查看集群上的服务的运行情况。同时当需要进行快速的测试或者运行作业的时候，也可以登录到主实例上，通过命令行来直接提交作业。当集群开启了高可用的时候会有2个主实例节点，默认只有1个。

Core 核心实例

核心实例是被主实例管理的实例节点。上面会运行 Hadoop HDFS 的 Datanode 服务，并保存所有的数据。同时也会部署计算服务，比如 Hadoop YARN 的 NodeManager 服务，来执行计算任务。为满足存储数据量或者是计算量上升的需要，核心实例可以随时的扩容，不影响当前集群的正常运行。核心使用可以使用多种不同的存储介质来保存数据。参考磁盘介绍。

Task 计算实例

计算实例是专门负责计算的实例节点，是一个可选的实例类型。如果核心实例的计算能力足够的情况下，可以不使用计算实例。计算实例可以在任何时候快速的为集群增加额外的计算能力，如 Hadoop 的 MapReduce tasks， Spark executors 等。在计算实例上不会保存 HDFS 的数据，因此在计算实例上不运行 Hadoop HDFS 的 Datanode服务。计算实例可以随时的新增和减少，都不会

影响到现有集群的运行。计算实例节点的减少可能会引起 MapReduce 和 Spark 的作业的失败，能否成功取决于该计算服务的重试容错能力。

2.2 Gateway实例

Gateway一般为独立的一个集群，由多台相同配置的节点组成。

在创建Gateway集群时，可以关联到一个已经存在的Hadoop集群上，该集群上会部署Hadoop（HDFS+YARN）、Hive、Spark、Sqoop、Pig等客户端，方便对集群进行操作。这样做的好处是：它可以作为一个独立的提交点，不会占用集群的资源，尤其是在Master提交的方式，可以提高Master节点的稳定性。如果作业太多，可以动态的增加节点。

您可以创建多个不同的Gateway集群，来给不同的用户使用，让他们可以使用各自独有的环境配置来满足不同的业务需求。

2.3 ECS实例说明

ECS实例说明

EMR目前支持的ECS实例类型

- 通用型

  VCPU与Memory比为1：4，如32核128G，使用云盘作为存储。

- 计算型

  VCPU与Memory比为1：2，如32核64G，使用云盘作为存储，提供了更多的计算资源。

- 内存型

  VCPU与Memory比为1：8，如32核256G，使用云盘作为存储，提供了更多的内存资源。

- 大数据型

  使用本地SATA盘作为数据存储，拥有很高的存储性价比，是大数据量（T级别的数据量）场景下的推荐机型。

- 本地SSD型

  使用本地SSD盘，拥有极高的本地iops和吞吐能力。总体的存储

- 共享型（入门级）

  共享CPU的机型，在大计算量的场景下，稳定性不够。入门级学习使用，不推荐企业客户使用。

- GPU

使用GPU的异构机型，可以用来运行机器学习等场景。

实例类型适用场景

- Master 主实例

  适合通用型或内存型实例，数据直接使用阿里云的云盘来保存，有三个备份的保证，数据高可靠。

- Core 核心实例

  小数据量（T以下）或者是使用OSS作为主要的数据存储时，可以使用通用型，计算型或内存型。当数据量较大的情况下，如10T或以上，推荐使用大数据机型，会获得极高的性价比。使用本地盘会有一个数据可靠性的挑战，会由 EMR 平台来进行维护和保证。

- Task 计算实例

  作为集群的计算能力的补充，可以使用除大数据型以外的所有的机型。目前本地SSD型尚未支持，后续会加入到 Task 中。

2.4 存储说明

在节点上存在2种角色的磁盘，一类是系统盘，用来安装操作系统。一类是数据盘，用来保存数据。系统盘默认都是一块，必须使用云盘。而数据盘可以有很多块，目前上限可以一个节点挂16块。每一块都可以有不同的配置，类型和容量都可以不同。EMR默认使用SSD云盘作为集群的系统盘。EMR 默认挂载4块云盘，目前的内网带宽的情况下4块云盘是比较合理的配置。

云盘与本地盘

有2种类型的磁盘可以用作数据的存储：

- 云盘

  包括，SSD 云盘，高效云盘，普通云盘。

  特点是，磁盘并不直接挂载在本地的计算节点上，通过网络访问远端的一个存储节点。每一份数据在后端都有2个实时备份，一共三份数据。所以当一份数据损坏的时候（磁盘损坏，不是用户自己的业务上的破坏），会自动的使用备份数据恢复。

- 本地盘

  包括，大数据型的 SATA 本地盘，和本地 SSD 的本地 SSD 盘。

直接挂载在计算节点上的磁盘，拥有超过云盘的性能表现。使用本地盘的时候不能选择数量，只能使用默认配置好的数量，和线下物理机一样，数据没有后端的备份机制，需要上层的软件来保证数据可靠性。

适用的场景

在EMR中，所有云盘和本地盘都会在节点释放的时候清除数据，磁盘无法独立的保存下来，并再次使用。Hadoop HDFS 会使用所有的数据盘作为数据存储。 Hadoop YARN 也会使用所有的数据作为计算的临时存储。

当业务数据量并不太大（T级别以下）的时候，可以使用云盘，IOPS和吞吐相比本地盘都会小些。数据量大的时候，推荐都使用本地盘，EMR 会来维护本地盘的数据可靠性。如果发现在使用中明显的吞吐量不够用，可以切换到本地盘的存储上。

OSS

在 EMR 中可以将 OSS 作为 HDFS 使用。 用户可以非常方便的读写OSS，所有使用 HDFS 的代码也可以简单的修改就能访问 OSS 上的数据了。

比如：

Spark中读取数据

```
sc.Textfile("hdfs://user/path")
```

替换存储类型 hdfs -> oss

```
sc.Textfile("oss://user/path")
```

对于MR或者Hive作业也是一样

HDFS命令直接操作OSS数据

```
hadoop fs -ls oss://bucket/path
hadoop fs -cp hdfs://user/path  oss://bucket/path
```

这个过程，不需要输入AK和endpoint，EMR都会自动替用户使用当前集群所有者的信息补全。

但 OSS 的 iops 不高，在一些需要高 IOPS的场景，不适合使用，比如流式计算 Spark Streaming 或 HBase。

2.5 本地盘机型支持

阿里云为了满足大数据场景下的存储的需求，在云上推出了本地盘的机型：D1系列。这个系列提供了本地盘而非云盘作为存储。解决了之前使用云盘的多份冗余数据导致的成本高问题，同时数据的传输不再需要全部通过网络，从而提高了磁盘的吞吐能力。同时还能发挥 Hadoop 的就近计算的优势。

相比于使用云盘的方式，极大的提高了存储的性能，并降低了存储的单价，达到和线下物理机几乎相同的成本。

本地盘机型在提供了大量的优势的情况下，也带来了一个问题：数据可靠性。对于云盘来说，由于有阿里云默认的磁盘多备份策略，所以用户可以说完全感知不到磁盘的损坏，由云盘自动保证数据可靠，当使用了本地盘以后这个就需要由上层的软件来保证。同时如果有磁盘与节点的故障情况，也需要进行人工的运维处理。

EMR + D1 方案

EMR 产品针对本地盘机型，如 D1，推出了一整套的自动化运维方案，帮助阿里云用户方便可靠的使用本地盘机型，不需要关心整个运维的过程的同时，做到数据高可靠，服务高可用。

主要的一些点如：

- 强制节点的高可靠分布
- 本地盘与节点的故障监控
- 数据迁移时机自动决策
- 自动的故障节点迁移与数据平衡
- 自动的HDFS数据检测
- 网络拓扑调优

EMR通过整个后台的管控系统的自动化运维，协助用户更好的使用本地盘机型，实现高性价比的大数据系统。

说明：

如需使用 D1 机型搭建 Hadoop 集群，请工单联系我们协助操作。

2.6 经典网络与VPC互访

目前阿里云存在两种网络类型，一个是经典网络，一个是VPC。很多用户的业务系统因为历史的原因还会在经典网络中，而EMR集群是在VPC中，本节将会介绍如何使经典网络的ECS可以和VPC网络下的EMR集群网络互访。

ClassicLink

为了解决这个问题，阿里云推出了*ClassLink*方案。

大致步骤如下，详细的请参考上面的ClassLink文档：

1. 首先按照上面文档中指定网段创建vswitch。
2. 在创建集群的时候，请使用该网段的vswitch来部署集群。
3. 在ECS控制台将对应的经典网络节点连接到这个VPC。
4. 设置安全组访问规则。

然后就可以让经典网络的ECS和VPC的集群互访了。

3 集群

3.1 创建集群

本文将介绍创建集群的操作步骤和相关配置。

进入创建集群页面

1. 登录阿里云 *E-MapReduce* 控制台。
2. 完成 RAM 授权，操作步骤请参见角色授权。
3. 在上方选择所在的地域（Region），所创建集群将会在对应的地域内，一旦创建后不能修改。
4. 单击创建集群，进行创建。

创建集群流程

注意：

集群除了名字以外，一旦创建完成就无法被修改。所以在创建时请仔细确认需要的配置。

要创建集群，您需要继续完成以下 3 个步骤：

1. 软件配置

   配置项说明：

   - 产品版本：E-MapReduce 产品的主要版本，代表了一整套的开源软件环境，它会定时的根据内部组成软件的升级进行升级。一般如果 Hadoop 相关的软件有进行升级，E-MapReduce 也会升级，这个时候就会升级这个主版本号。低版本的集群无法自动的升级到一个高版本上。
   - 集群类型：目前的EMR提供了:
     - Hadoop集群，提供半托管的Hadoop、Hive、Spark离线大规模分布式数据存储和计算，SparkStreaming、Flink、Storm流式数据计算，Presto、Impala交互式查询，Oozie、Pig等Hadoop生态圈的组件，具体的组件信息可以在选择界面的列表中查看。
     - Kafka集群，是半托管分布式的、高吞吐量、高可扩展性的消息系统。提供一套完整的服务监控体系，保障集群稳定运行，用户无需部署运维，更专业、更可靠、更安全。广泛用于日志收集、监控数据聚合等场景，支持离线或流式数据处理、实时数据分析等。
     - Druid集群，提供半托管式实时交互式分析服务，大数据查询毫秒级延迟，支持多种数据摄入方式。可与 EMR Hadoop、EMR Spark、OSS、RDS 等服务搭配组合使用，构建灵活稳健的实时查询解决方案。

- Data Science集群，主要面向大数据+AI场景，提供了Hive、Spark离线大数据ETL，TensorFlow模型训练，用户可以选择CPU+GPU的异构计算框架，利用英伟达GPU对部分深度学习算法进行高性能计算。

- 必选服务：展示选择的集群类型下的所有的软件组件列表，包括名称和版本号。
- 可选服务：根据需求，您可选择不同的组件，被选中的组件会默认启动相关的服务进程。

> **说明：**
> 您选择的组件越多，对您机器的配置要求就越高，否则很可能无法有足够的资源来运行这些服务。

- 安全模式：是否开启集群的 Kerberos 认证功能。
- 软件配置（可选）：可以对集群中的基础软件例如 Hadoop、Spark、Hive 等进行配置，详细使用方法请参见软件配置。

2. 硬件配置

配置项说明：

- 付费配置
  - 付费类型：包年包月是一次性支付一个长期的费用，价格相对来说会比较便宜，特别是包三年的时候折扣会很大。按量付费是根据实际使用的小时数来支付费用，每个小时计一次费用。适合与短期的测试或者是灵活的动态任务，价格相对来说会贵一些。
  - 购买时长：您可选择购买 1 个月、2 个月、3 个月、6 个月、9 个月、1 年、2 年、3 年。
- 集群网络配置
  - 集群可用区：选择集群所在的可用区（Zone），不同的可用区会有不同的机型和磁盘。在每个 Region 内存在多个可用区。可用区在物理上属于不同的区域，一般来说如果需要较好的网络，推荐您选择相同的可用区，但是这样也会使创建集群失败的风险增大，因为单个可用区的存库不一定那么充足。如果需要大量的机器可以工单咨询我们。
  - 网络类型：默认使用专有网络（VPC），专有网络需要额外提供所属 VPC 以及子网（交换机），若还未创建，可前往*VPC*控制台进行创建。E-MapReduce 专有网络详细使用说明查看专有网络。
  - 可用区：可用区为在同一地域下的不同物理区域，可用区之间内网互通。
  - **VPC**：选择在该地域的VPC。如没有，单击创建 **VPC** / 子网(交换机）前往新建。

- **交换机**：选择在对应的VPC下的在对应可用区的交换机，如果在这个可用区没有可用的交换机，那么就需要前往去创建一个新的使用。
- **安全组**：集群所属的安全组。这里只展示用户在 E-MapReduce 产品中创建的安全组，目前尚不支持选择在 E-MapReduce 外创建的安全组。如果需要新建安全组，可以输入新的安全组的名字完成新建。长度限制为 2-64 个字符，以大小写字母或中文开头，可使用中文、字母、数字、“-”和“_”。

- 集群节点配置
  - 高可用：打开后，Hadoop 集群会有 2 个 master 来支持 Resource Manager 和 Name Node 的高可用。HBase 集群原来就支持高可用，只是另一个节点用其中一个 core 节点来充当，如果打开高可用，会独立使用一个 master 节点来支持高可用，更加的安全可靠。默认为非高可用模式，master节点数量为1。
  - 节点类型：
    - Master主实例节点，主要负责Resource Manager，Name node等控制进程的部署。
    - Core核心实例节点，主要负责集群所有数据的存储，可以按照需要进行扩容。
    - Task纯计算节点，不保存数据。调整集群的计算力使用。
  - 节点配置：不同规格的机型的选择。各个机型有各自比较适用的场景，可以根据需要选择。
  - 数据盘类型：集群的节点使用的数据盘类型，数据盘有 3 种类型，普通云盘、高效云盘和 SSD 云盘，根据不同机型和不同的 Region，会有不同。当用户选择不同的区的时候，该区支持什么盘，下拉框就会展示什么类型的盘。数据盘默认设置为随着集群的释放而释放。本地盘的计算节点，磁盘是默认选定的，无法修改。
  - 数据盘容量：目前推荐的集群容量最小是 40G 单机，最大可以到32T单节点。本地盘的容量是默认的，无法调整。
  - 实例数量：需要的总的节点的台数。一个集群至少需要 3 台实例（高可用集群需增加 1 个 Master 节点，至少 4 台）。按量集群目前最大台数是 50 台，如果需要超过 50 台，请工单联系我们。包月集群最大100台，超过50台请工单联系我们。

3. 基础配置

   配置项说明：

   - **基础信息**

     集群名称：集群的名字，长度限制为 1-64 个字符，仅可使用中文、字母、数字、“-”和“_”。

- 运行日志
  - 运行日志：是否保存作业的日志，日志保存默认是打开的。开启后会需要您选择用来保存日志的 OSS 目录位置，会将您的作业的日志保存到该 OSS 存储目录上。当然，您要使用这个功能必须先开通 OSS，同时上传的文件会按照使用的量来计算用户的费用。强烈建议您打开 OSS 日志保存功能，这会对您的作业调试和错误排查有极大的帮助。
  - 日志路径：保存日志的 OSS 路径。
  - 统一**Meta**数据库：将你所有的 Hive 的元数据都保存到集群外部的数据库上，由EMR产品提供。推荐当集群使用 OSS 作为主要的存储的时候，使用这个功能。
- 权限设置
  - 服务角色：这个是用户将权限授予EMR服务，允许 EMR 代表用户调用其他阿里云的服务，例如 ECS 和 OSS
  - **ECS**应用角色：这个是当用户的程序在 EMR 计算节点上运行的时候，可以不填写阿里云的 AK 来访问相关的云服务，比如OSS。EMR 会自动的申请一个临时 AK 来授权这次访问。而这个 AK 的权限控制将由这个角色来控制。
- 登录设置

  远程登录：是否打开安全组22端口，默认开启。

  登录密码：设置 master 节点的登录密码。8 - 30 个字符，且必须同时包含大写字母、小写字母、数字和特殊字符!@#$%^&*。
- 引导操作（可选）： 您可以在集群启动 Hadoop 前执行您自定义的脚本，详细使用说明请参见引导操作。

配置清单和集群费用

页面右边会显示您所创建集群的配置清单以及集群费用。根据付费类型的不同，会展示不同的价格信息。按量付费集群显示每小时费用，包年包月显示总费用。

确认创建

当所有的信息都有效填写以后，创建按钮会亮起，确认无误后单击创建将会创建集群。

说明：

- 若是按量付费集群，集群会立刻开始创建。页面会返回概览页，就能看到在集群概览中有一个初始化中的集群。请耐心等待，集群创建会需要几分钟时间。完成之后集群的状态会切换为空闲。
- 若是包年包月集群，则会先生成订单，在支付完成订单以后集群才会开始创建。

登录Core节点

登录Core节点，请按照以下步骤操作：

1. 在Master节点上切换到hadoop账号。

```
su hadoop
```

2. 免密码SSH登录到对应的Core节点。

```
ssh emr-worker-1
```

3. 通过sudo命令可以获得root权限。

```
sudo vi /etc/hosts
```

创建失败

如果创建失败，在集群列表页上会显示集群创建失败，将鼠标移动到红色的感叹号上会看到失败原因。

创建失败的集群可以不用处理，对应的计算资源并没有真正的创建出来。这个集群会在停留3天以后自动隐藏。

3.2 扩容集群

当您的集群资源（计算资源、存储资源）不足的时候，您可以将您的集群进行水平扩展。目前支持扩展您的 Core 节点和Task节点，且使用的配置默认与您之前购买的 ECS 配置一致。

扩容入口

在集群管理页上，找到需要扩展的集群条目，单击更多，在下拉框中选择扩容就会进入集群扩容页面。也可以单击详情，然后在详情页的 右上角单击扩容。

扩容界面

如下图所示：

说明：

目前，只支持扩容，不支持缩容。

- 配置：当前实例组的配置。
- 付费类型：当前集群的付费类型。
- 当前 Core 台数：默认显示的是您当前的所有 Core 节点数量。
- 增加 Core 台数: 输入您实际需要增加的量，右侧会显示扩容后的集群总费用，然后单击确定就会进行扩容。安全起见，强烈推荐您在扩容的时候小批量的进行，比如 1 到 2 台，不要一次扩容太大的数量。
- 交换机：当前集群的交换机。

扩容状态

如下图所示：

您可以在集群基础信息页的核心实例组(CORE)的信息上看到集群的扩容情况，正在扩容的节点，其状态会显示为扩容中。当这台 ECS 的状态转为正常后，该 ECS 即已经加入该集群，并可正常提供服务了。

修改密码

扩容成功后用户可以SSH登录到扩容节点上修改root密码，具体步骤如下：

1. 使用如下命令 SSH 登录到 master 主机。请在集群详情页的主机信息栏中获取集群 master 机器的公网 IP。

```
ssh root@ip.of.master
```

2. 切换到hadoop用户：

```
su hadoop
```

3. 登录到扩容节点，扩容节点的内网IP请在集群详情页的主机信息栏中获取：

```
ssh ip.of.worker
```

4. 使用如下命令修改root用户密码：

```
sudo passwd root
```

3.3 集群列表

集群列表用来展示您所拥有的所有集群的基本信息。

集群列表展示的集群信息如下图所示：

列表栏各项说明如下：

- 集群 ID/名称：集群的 ID 以及名称。在这里将鼠标移动到名称上，可以对集群的名称进行修改。
- 集群类型：目前有 Hadoop 、Druid、Data Science，Kafka四种类型。
- 状态：集群的状态，请参见集群状态。当集群出现异常的时候，比如创建失败，在右侧会有提示信息，鼠标悬停能够查看到详细的错误信息。您也可以单击状态（默认）对状态进行筛选。
- 创建时间：集群创建的时间。
- 运行时间：从创建开始到目前的运行时间。集群一旦被释放，计时终止。

- 付费类型：集群的付费类型。
- 操作：当前集群可以被施加的操作，包含以下操作。
  - 监控数据：监控E-MapReduce集群的CPU空闲率、内存容量、磁盘容量等多个监控项，帮助用户监测集群的运行状态。
  - 管理：进入集群与服务管理页面。
  - 详情：进入集群的详情页，查看集群建立以后的详细信息。
  - 更多
    - 扩容：集群扩容功能入口。
    - 释放：释放一个集群，请参见释放集群。
    - 重启：重启一个集群。

3.4 释放集群

在集群列表页面，您可以单击集群条目右侧的释放按钮对集群进行释放操作。

只有以下状态的按量付费的集群才可以被释放。

- 创建中
- 运行中
- 空闲中

普通释放

释放前会提示您再次确认，一旦确认释放，会发生以下的操作：

- 所有在集群上的作业都会被强制终止。
- 如果您选择了保存日志到 OSS，那么当前作业的日志会被保存到 OSS，所需时间取决于日志大小。日志的上传和作业的运行是并行的，作业生成日志的同时，就会进行日志的上传。所以最终作业停止时，需要上传的日志一般不会特别多，正常在几分钟内都会完成。
- 终止并释放所有的ECS。这个过程取决于集群的大小，越小的集群会越快，正常都在几秒内完成，至多不会超过5分钟。最迟释放的ECS在等待释放时仍然计费。

> **警告：**
> 如果您想要省钱而控制在整点前释放，请务必留出一定的释放时间保证确实在整点前释放。

强制释放

如果您不需要任何日志，只是想快速结束集群的运行，那么可以开启强制释放。释放过程就会跳过日志收集（如果有打开日志收集的话），直接进入ECS释放阶段。

释放失败的集群

由于系统错误等原因，集群有可能会在确认释放后，释放失败。您单击释放是一个异步的过程，可能等一会儿会发生集群释放失败的情况，但是不用担心，E-MapReduce 会启动后台保护，自动重试释放该集群，直到集群被成功释放为止。

3.5 集群详情

集群详情用来展示用户集群的详细信息。

用户集群的详细信息包含以下四大部分：

集群信息

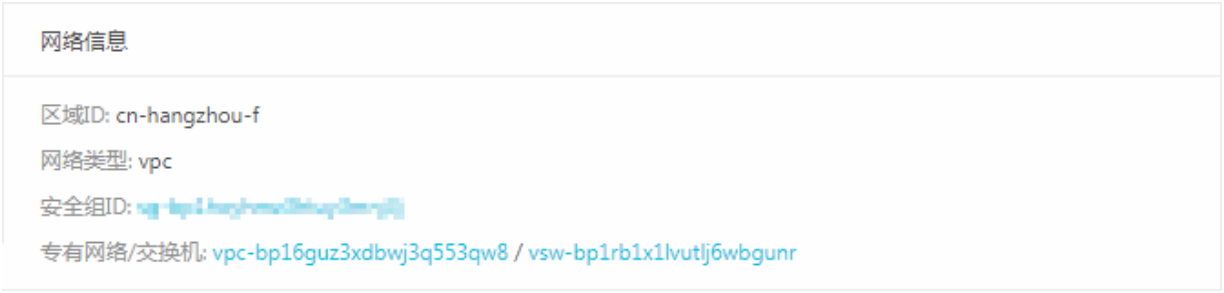
- **名称**：集群的名称
- **ID**：集群的实例 ID 。
- **地域**：集群所在的 Region。
- **开始时间**：集群的创建时间。
- **软件配置**：软件的配置信息。
- **IO优化**：是否开启了IO优化。
- **高可用集群**：是否开启了高可用集群。
- **安全模式**：集群中的软件以Kerberos安全模式启动。请参见*Kerberos*简介。
- **付费类型**：集群的付费类型。
- **当前状态**：请参见集群状态。
- **运行时间**：集群的运行时间。
- **引导操作**：这里列出了所有配置的引导操作的名称、路径以及参数。

- **ECS应用角色**: 这个是当用户的程序在 EMR 计算节点上运行的时候，可以不填写阿里云的 AK 来访问相关的云服务，比如OSS。EMR 会自动的申请一个临时 AccessKey 来授权这次访问。而这个 AK 的权限控制将由这个角色来控制。

软件信息

- **EMR版本**：使用的 E-MapReduce 的主版本。
- **集群类型**：选择的集群类型。
- **软件信息**：列出了用户安装的所有的应用程序及其版本，例如，HDFS2.7.2，Hive 2.3.3，spark 2.3.1。

网络信息



- **区域ID**：集群所在的可用区，例如 cn-hangzhou-b，与 ECS 的一致。
- **网络类型**：集群所在的网络。
- **安全组ID**：集群加入的安全组ID
- **专有网络/交换机**：用户集群所在的 VPC 与子网交换机的 ID。

主机信息

- 主实例组（Master）：所有 Master 节点对应的配置，包含以下内容。
  - 主机数量：当前的节点数量和实际申请的节点数量。理论上这 2 个值一定是一样的，但是在创建过程中，当前节点会小于申请节点，直到创建完成。
  - **CPU**：单个节点的 CPU 的核数。
  - 内存：单个节点的内存的容量。
  - 数据盘配置：数据盘类型和单个节点的数据盘容量。
  - **ECS ID**：所购买的 ECS 的ID
  - 状态：包含创建中、正常、扩容中和已释放。
  - 公网 ：Master 的公网 IP。
  - 内网 ：机器的内网 IP，可以被集群中的所有节点访问到。
  - 创建时间：所购买的 ECS 的创建时间。
- 核心实例组（Core） ：所有 Core 节点对应的配置，包含以下内容。
  - 主机数：当前的节点数量和实际申请的节点数量。
  - **CPU**：单个节点的 CPU 的核数。
  - 内存：单个节点的内存的容量。
  - 数据盘配置：数据盘类型和单个节点的数据盘容量。

— **ECS ID**：所购买的 ECS 的ID

— **状态**：包含创建中，正常和扩容中。

— **内网** ：机器的内网 IP，可以被集群中的所有节点访问到。

— **创建时间**：所购买的 ECS 的创建时间。

3.6 用户管理

用户管理是在指定集群上创建相关服务所需的账号，目前支持创建Knox账号和Kerberos账号。

创建RAM账号

1. 登录阿里云 *E-MapReduce* 控制台，进入**集群列表**页面。
2. 单击集群ID右侧的**管理**。
3. 在左侧导航栏中单击**用户管理**。
4. 单击右侧页面**创建RAM子账号**。

添加Knox账号

1. 进入**用户管理**页面，在页面上选择需要添加到集群的账号，单击**操作**列的**设置Knox密码**。
2. 在**添加Knox用户**的对话框中设置密码，然后单击**确认**。
3. 刷新**用户管理**页面，当**Knox账号**列的状态显示**已同步**时，表示Knox账号添加成功。

   添加成功后，您即可使用**账号名称**及第2步中设置的密码登录Knox。

删除Knox账号

1. 进入**用户管理**页面，在页面上选择需要添加到集群的账号，单击**操作**列的**删除Knox密码**。
2. 刷新**用户管理**页面，当Knox账号列的状态显示**未同步**时，表示Knox账号删除成功。

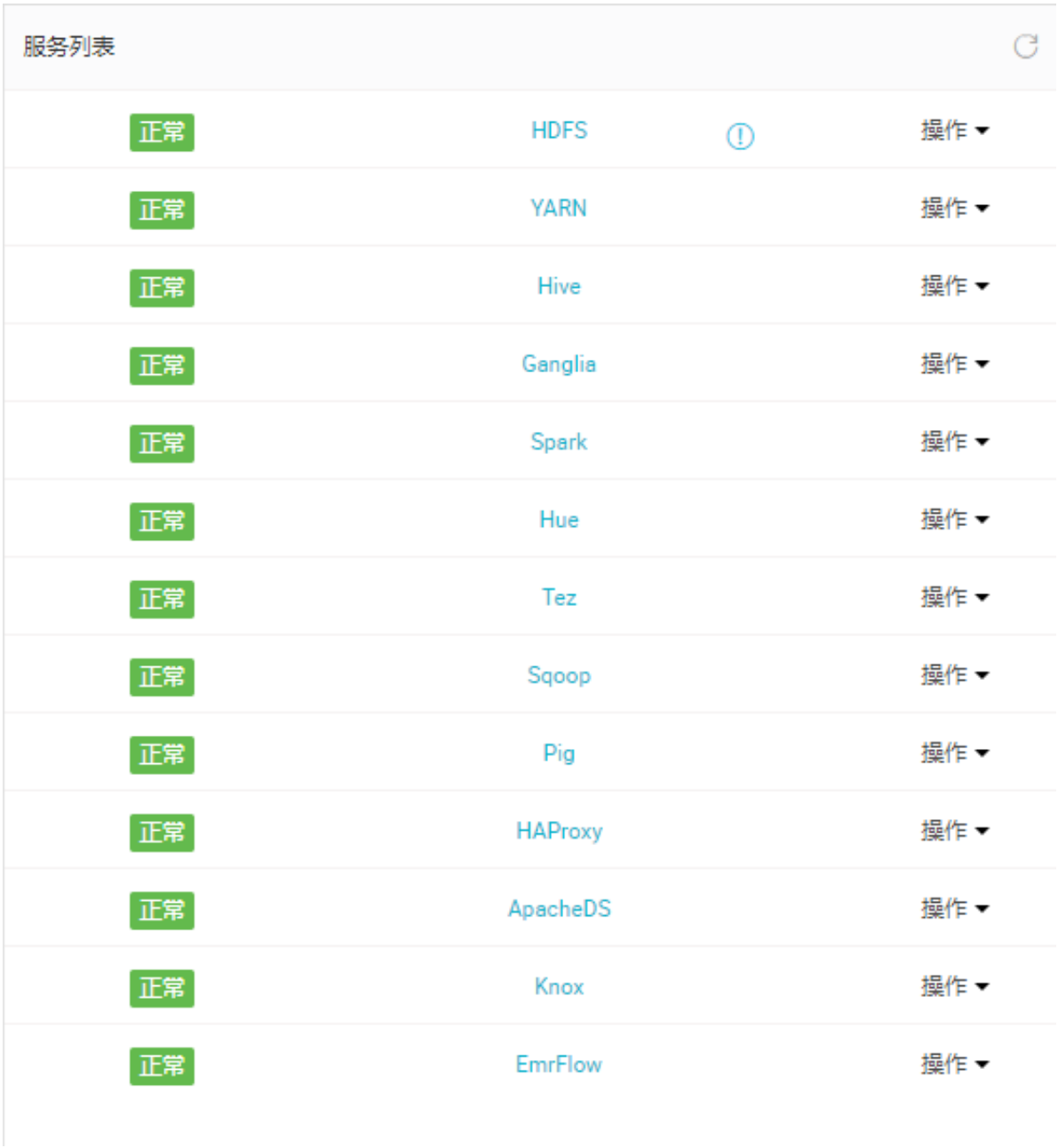
常见问题

- 不同集群不能共享Knox账号，即：在cluster-1上添加Knox账号A之后，并不会共享给cluster-2，若想在cluster-2上使用Knox账号A，需要在cluster-2上重新添加账号A。原因是Knox账号是创建在集群中的，每个集群的Knox账号不互通。
- 添加Knox账号过程中，若显示**同步失败**，请单击**重试**重新添加。
- 添加Knox账号过程中，多次**重试**仍然失败，单击左侧**集群与服务管理**，检查ApacheDS是否已停止，若已停止则启动ApacheDS后，再回到**用户管理**进行重试。

3.7 服务列表

集群与服务管理页展示了HDFS，YARN等服务在集群节点上的运行状态。

服务列表展示的信息如下图所示。



创建集群时未勾选的服务，例如Storm，将不会显示在服务列表中。

单击服务列表中的服务可查看对应服务的状态、部署拓扑、配置和配置修改历史。服务列表中服务的状态有正常和错误两种。如果某个节点上的服务状态是错误，您可以通过master节点跳转，登录到对应节点上查看服务进程情况。

3.8 集群脚本

集群，特别是包年包月集群，在使用过程中，可能会有新的第三方软件安装，修改集群运行环境的需求。集群脚本功能可以在集群创建好后批量选择节点，运行您指定的脚本，以实现个性化的需求。

集群脚本的作用

集群脚本类似引导操作，您可以在集群创建好后安装很多目前集群尚未支持的软件到您的集群上，例如：

- 使用 yum 安装已经提供的软件。
- 直接下载公网上的一些公开的软件。
- 读取 OSS 中您的自有数据。
- 安装并运行一个服务，例如 Flink 或者 Impala，但需要编写的脚本会复杂些。

强烈建议您先用单个节点进行集群脚本的测试，测试都正确以后再在整个集群上操作。

如何使用

1. 集群状态是空闲或者运行中的集群可以运行集群脚本，集群列表页面点击对应集群的**查看详情**按钮
2. 左侧菜单单击**集群脚本**，即会进入该集群的集群脚本执行界面，右侧是已经执行过的集群脚本列表。
3. 单击右上角**创建并执行**，进入创建界面。
4. 填写创建界面上的配置项，选择执行的节点，点击执行，完成添加并执行操作。
5. 集群脚本列表可以看到新创建的集群脚本，点击刷新可以更新集群脚本的状态。
6. 单击**查看详情**可以看到脚本在各个节点上的运行情况，单击**刷新**可以更新脚本在各个节点上的运行状态。

只有空闲或者运行中的可用集群才能使用集群脚本功能。集群脚本适用于长期存在的集群，对按需创建的临时集群，应使用引导操作来完成集群初始化工作。

集群脚本会在您指定的节点上下载oss上的脚本并运行，根据返回值是否为0判断执行成功还是失败。如果运行状态是失败，您可以登录到各个节点上查看运行日志，运行日志记录在每个节点的*/var/log/cluster-scripts/clusterScriptId*目录下。如果集群配置了oss日志目录，运行日志也会上传到*osslogpath/clusterId/ip/cluster-scripts/clusterScriptId*目录下方便查看。

默认会使用 root 账户执行您指定的脚本，您可以在脚本中使用 **su hadoop** 切换到 Hadoop 账户。

集群脚本可能在部分节点上运行成功，部分节点上运行失败，例如节点重启导致的脚本运行失败。您可以在解决异常问题后，单独指定失败的节点再次运行。当集群扩容后，您也可以指定扩容的节点单独运行集群脚本。

1个集群同一时间只能运行一个集群脚本，如果有正在运行的集群脚本，无法提交执行新的集群脚本。每个集群最多保留10个集群脚本记录，超过10个需要将之前的记录删除才能创建新的集群脚本。

脚本的例子

类似引导操作的脚本，您可以在脚本中指定从 OSS 下载需要的文件，下面的例子会将 *oss://yourbucket/myfile.tar.gz* 这个文件下载到本地，并解压到 */yourdir*目录下：

```
#!/bin/bash
osscmd --id=<yourid> --key=<yourkey> --host=oss-cn-hangzhou-internal.
aliyuncs.com get oss://<yourbucket>/<myfile>.tar.gz ./<myfile>.tar.gz
mkdir -p /<yourdir>
tar -zxvf <myfile>.tar.gz -C /<yourdir>
```

osscmd 已预安装在节点上，可以直接调用来下载文件。

说明：

OSS 地址 host 有内网地址、外网地址和 VPC 网络地址之分。如果用经典网络，需要指定内网地址，杭州是 oss-cn-hangzhou-internal.aliyuncs.com。如果用 VPC 网络，要指定 VPC 内网可访问的域名，杭州是 vpc100-oss-cn-hangzhou.aliyuncs.com。

脚本也可以通过 yum 安装额外的系统软件包，下面的例子会安装 ld-linux.so.2：

```
#!/bin/bash
yum install -y ld-linux.so.2
```

3.9 集群续费

当您的包年包月集群服务即将到期的时候，您需要执行集群的续费操作，以继续您的 E-MapReduce 集群服务。集群续费包括 E-MapReduce 服务费的续费以及集群中 ECS 的续费。

续费入口

1. 登录阿里云 *E-MapReduce* 控制台。
2. 单击上方的**集群管理**页签，进入集群列表页面
3. 找到需要续费的集群条目。

4. 单击相应集群条目操作栏下的续费，进入集群续费页面。

续费界面

如下图所示：

- **ECS 当前到期时间**：这台 ECS 到期的时间。
- **EMR到期时间**：E-MapReduce 服务到期的时间。
- **数量**：实例组的机器数量。
- **ECS列表**：集群中机器的 ECS 实例 ID。
- **ECS 续费时长**：需要对这个节点续费的时长，目前支持 1-9 个月、1 年、2 年和 3 年的续费时长。
- **EMR 续费时长**：对应的这个节点的 E-MapReduce 服务费的时长，推荐和 ECS 的设置一致。
- **续费价格**：E-MapReduce 服务以及ECS 节点对应的续费价格。

支付订单

> 注意：
> 集群续费的费用为 ECS 续费价格与 E-MapReduce 服务产品价格的总和。当集群列表中存在未支付的订单时，您将无法执行集群的扩容和续费操作。

1. 单击**确定**后，会弹出下单成功提示的提示框（该提示信息可能会有较长的时间延迟，请耐心等待）。
2. 单击**前往支付**，跳转到订单支付页面。支付页面会显示您应付的总金额以及各订单的详情，其中一个为 E-MapReduce 产品费用订单，其它的为集群续费的 ECS 订单。
3. 单击**确认支付**，完成付款。
4. 完成支付之后，单击**支付完成**，返回集群列表页面。

此时，在集群列表页上续费成功集群的到期时间会更新为续费之后的到期时间，对应的 ECS 的到期时间更新目前存在一定的延迟，一般约 3-5 分钟之后会更新成续费之后的到期时间。

若您只是确认了续费订单但并未进行支付，您可在集群列表页面中找到该集群条目，在其右侧的操作栏中会出现**前往支付**和**取消订单**的按钮。您可单击**前往支付**，完成对应的订单支付和集群扩容流程；或单击**取消订单**，以取消本次续费操作。

3.10 集群自动续费管理

如果您购买的是包年包月集群服务，且要长期使用该项服务，您可以开通自动续费功能。开通自动续费功能后，系统将在资源到期前执行自动续费操作，无需您手动操作续费。以此保证不会因为资源未及时续费，而导致资源、数据被删除。

续费入口

1. 登录阿里云 *E-MapReduce* 控制台。
2. 单击上方的**集群管理**页签，进入集群列表页面
3. 在需要续费的集群条目后单击**详情**进入集群详情页。
4. 单击右上角的**费用管理**，在下拉框中选择**自动续费管理**。

- **ECS ID**： 对应的节点的ecsid
- **ECS到期时间**： ECS节点的到期时间
- **EMR到期时间**：ECS节点对应的EMR服务的到期时间
- **ECS当前自动续费状态**： 开通或未开通，开通表示自动续费的。未开通，表示没有进行自动续费
- **EMR当前自动续费状态**： 开通或未开通，开通表示自动续费的。未开通，表示没有进行自动续费
- **ECS续费操作**：勾选上以后，并进行了保存。对应就会进入到自动续费的开通状态。按照后面设置的续费时长来进行续费，比如一个月那么就是每一个月进行一次续费； 若不勾选，并进行保存。则取消自动续费的动态。
- **E-MapReduce续费操作**： 勾选上以后，并进行了保存。对应就会进入到自动续费的开通状态。按照后面设置的续费时长来进行续费，比如一个月那么就是每一个月进行一次续费；若不勾选，并进行保存。则取消自动续费的动态。

说明：

- 如果您的实例将于下一天到期，需手动续费，无法设置自动续费。
- 自动续费支持使用优惠券。

3.11 安全组

目前 E-MapReduce 创建集群的时候需要使用在 E-MapReduce 中创建的安全组。

主要是因为 E-MapReduce 创建的集群只开放了 22 端口，推荐您将 ECS 实例也按照功能划分，放于不同的用户安全组中。例如，E-MapReduce 的安全组为 **E-MapReduce** 安全组，而您已有的安全组为用户-安全组，每个安全组按照不同的需要开启不同的访问控制。

如果需要和已有集群进行联动，请参考如下做法。

将 E-MapReduce 集群加入现有安全组

1. 登录阿里云 *E-MapReduce* 控制台。
2. 单击上方的集群管理页签，进入集群列表页面
3. 找到需加入安全组的集群条目，单击其操作中的详情，即会进入集群详情页。
4. 在集群详情页上，找到网络信息下的所属安全组ID，单击安全组**ID**。
5. 单击左侧的安全组内实例列表查看集群所有 ECS 实例所在的安全组名称。
6. 前往阿里云*ECS*的管理控制台，单击页面左侧的安全组，在列表中找到步骤 3 中查看到的安全组条目。
7. 单击该安全组操作中的管理实例，您会看到很多名字以 emr-xxx 开头的 ECS 实例，这些就是对应的 E-MapReduce 集群中的 ECS。
8. 全选这些实例并单击移入安全组，选择想要加入的新的安全组即可。

将现有集群加入 E-MapReduce 安全组

和上面的操作一样，先找到现有集群所在安全组，重复如上的操作，移入 E-MapReduce 的安全组即可。如果是一些零散的机器，也可以直接在 ECS 的控制台界面上选择机器，然后通过下方的批量操作将集群移入 E-MapReduce 的安全组。

安全组的规则

一个 ECS 实例在多个不同的安全组的时候，安全组的规则是"或"的关系。举例来说就是，E-MapReduce 的安全组只开放了 22 端口，而用户-安全组开放了所有的端口。当 E-MapReduce 的集群加入用户-安全组以后，E-MapReduce 中的机器也会开放所有端口，所以在使用上请特别注意。

3.12 节点变配

在实际使用中，我们可能会发现我们集群中的节点，尤其是 Master 节点的 CPU 或者内存不够了。目前 EMR 还不能支持直接的升级配置。我们推荐您可以通过如下的方式来完成节点的配置升级。

说明：

只有包年包月集群才支持升级配置。

节点配置升级引导

1. 在集群管理页上，找到对应的集群条目，单击**详情**，进入集群详情页面。
2. 单击右上角的**配置升级**。
3. 修改需要升级的节点配置。
4. 单击**确认**，等待一段时间生成订单。
5. 支付订单。

   返回到集群管理页面，刷新页面确认节点信息已经变为要升配的目标规格，例如，CPU：4核 内存：16G。
6. 在*ECS控制台*，找到已升配的实例，逐台重启。
7. 修改集群配置，使得Yarn可以使用新增的资源。

   a. 修改Yarn服务的*yarn-site.xml*文件。

   b. 修改 **yarn.nodemanager.resource.memory-mb** 的值为 机器内存 * 0.8，单位为MB。修改 **yarn.scheduler.maximum-allocation-mb** 的值为 机器内存 * 0.8，单位为MB。举例来说，我现在新的配置下，内存是32G那么这里就配置为：

   ```
   yarn.nodemanager.resource.memory-mb=26214
   yarn.scheduler.maximum-allocation-mb=26214
   ```

   如果您的集群还没有支持页面修改的方式，那么需要登录到节点上，修改每一个节点的*/etc/ecm/hadoop-conf/yarn-site.xml*文件中对应的配置值。

   c. 重启Yarn服务，一般只需要重启worker节点，但是重启后nodemanger的端口变了，在yarn控制台看到很多节点lost，所以最好resourcemanager也重启下。
8. 工单联系 EMR 团队，提供升级以后的新的机型配置信息。EMR 团队会进行配置同步，保证后续的集群续费和操作正常。

3.13 磁盘扩容

如果业务发展中发现数据存储空间不够了，EMR 3.11.0及以上版本的集群可以在集群详情页对磁盘直接进行扩容来实现。

扩容入口

在集群管理页上，找到需要扩展的集群条目，单击详情，然后在详情页的 右上角单击磁盘扩容。

> 注意：
> 确保账号余额充足，扩容数据盘自动扣款，如果余额不够时，扩容数据盘流程会中断。

扩容界面

如下图所示：

- 配置：当前实例组数据盘的配置。
- 扩容至：扩容后数据盘大小。

> 说明：
> - 目前，只支持扩容，不支持缩容。
> - 磁盘扩容目前只支持数据盘扩容。

重启集群

1. 扩容结束后，在对应的机器组（比如：Core）会提示“扩容磁盘已完成，机器组待重启”。如下图所示：

2. 单击扩容磁盘已完成，机器组待重启。如下图所示：

注意：

重启集群会重启集群的ECS实例，所以重启中的ECS实例上的大数据服务不可用，请务必确保不影响业务的情况下操作

- 滚动重启
  - 勾选表示一个ECS实例重启完成且该实例上的大数据服务全部恢复后再启动下一个ECS实例。每个节点重启耗时约5分钟
  - 不勾选表示同时重启所有ECS实例
- 只重启变配节点
  - 变配节点，指已经完成过扩容磁盘或者升级配置操作的节点（如：CORE，MASTER等）
  - 勾选表示只重启变配节点，未变配的节点不会被重启。例如：只是对CORE组的节点做了磁盘扩容，但未对MASTER扩容磁盘和升级配 置操作，那么只会重启CORE组下的ECS实例，不会重启MASTER组下的ECS实例
  - 不勾选表示所有节点都将重启。即：集群下的所有机器都会被重启

3. 重启过程中，在对应的机器组（比如：CORE）会提示“机器组重启中”，如下图所示：

4. 待步骤3的提示消失后，扩容数据盘全部完成并生效。可以登录集群查验。

节点数据盘扩容

EMR 3.11.0以下版本的集群可以通过在ECS控制台上对磁盘进行扩容。

1. 登录EMR控制台，在集群列表中找到待扩容的集群，单击详情进入集群详情页面。

2. 查看集群中的待扩容的 Core 节点的 **ECS Id**，类似 i-bp1bsithym5hh9h93xxx。默认的情况下，在扩容的时候，请将集群中所有的节点磁盘都进行统一的扩容，保证集群内部的节点的磁盘容量都是一致的。
3. 复制 **ECS Id**，并前往 ECS 控制台。 选择左侧的实例标签，然后在搜索中选择实例 Id 并输入 ECS Id。 注意要选择相同的地域。
4. 找到对应的 ECS 节点后，单击管理，进入实例详情页面，然后点击左侧的本实例磁盘标签。
5. 扩容数据磁盘。 由于目前无法指定多个磁盘批量扩容，所以需要对每一个磁盘重复如下的扩容操作。
6. 首先在 ECS 控制台，扩容所有磁盘，并重启节点。
7. 参考 *ECS* 磁盘扩容说明 进行磁盘扩容。

   说明：

   这里umount操作失败的时候，需要先在集群上关闭 YARN 和 HDFS 服务。另外在 Disk1 操作的时候可能会碰到 ilogtail 写日志而无法 umount 的情况。需要通过 **sudo pgrep ilogtail | sudo xargs kill** 将 ilogtail 暂时杀掉。后续可以通过重启节点来恢复 ilogtail 服务。
8. 完成以后在节点通过 **df -h** 命令，能够看到所有的磁盘都完成了扩容。
9. 为了保证在后续的 EMR 扩容过程中使磁盘能和扩容后的一致，请工单联系我们扩容事项，我们进行集群数据更新。

节点系统盘扩容

系统盘扩容是一个比较复杂的操作，如无必要尽量避免操作。

1. 在EMR控制台上点击进入待扩容的集群详情页面。
2. 查看集群中的待扩容的 Master 节点的 **ECS Id**，类似 i-bp1bsithym5hh9h93xxx。默认的情况下，在扩容的时候，请将集群中所有的节点磁盘都进行统一的扩容，保证集群内部的节点的磁盘容量都是一致的。
3. 复制 **ECS Id**，并前往 ECS 控制台。 选择左侧的实例标签，然后在搜索中选择实例 Id 并输入 ECS Id。 注意要选择相同的地域。
4. 找到对应的 ECS 节点后，单击管理，进入实例详情页面，然后点击左侧的本实例磁盘标签。
5. 找到系统磁盘。系统磁盘只会有一块。
6. 参考 *ECS* 系统盘扩容说明 进行系统盘扩容。

   注意：

- 对于非HA集群，扩容系统盘会导致集群在扩容期间不可用。
- 扩容完成以后，因为ECS会做一些磁盘的处理，可能会导致节点的 `/etc/hosts` 文件变化，需要在扩容完成以后修复。另外ssh的免登也会被破坏，但这个不会影响服务本身，可以手动修复。

3.14 切换支付类型

在EMR中有2种支付类型，按量（后付费）和包月（预付费），一般在刚开始进行产品尝试的时候，我们都希望是按量的，少量的费用的尝试。一旦确定以后我们再转换为包月的方式。

按量转包月

现在EMR支持先使用按量集群测试，然后在任何时间，在集群详情页面中，单击**按量转包月**按钮，将集群从按量付费模式切换为包月模式。整个集群的付费模式都将切换。

包月转按量

目前不支持从包月切换到按量模式。

3.15 移除异常节点

当组成EMR集群的ECS节点异常时，如果用户不需要该节点并移除该异常节点，可以使用移除异常节点功能。

移除异常节点的操作步骤为：

1. 登录阿里云 *E-MapReduce* 控制台。
2. 单击**集群管理**。
3. 单击需要移出异常节点的集群ID
4. 在左侧导航栏中单击**主机列表**。
5. 查询找到需要移出的实例ID，操作栏单击**移除**，只有ECS状态为停止，或已经释放的ECS才会被移除。
6. 确认要移除实例，单击**确定**。

4 作业

4.1 Hadoop MapReduce 作业配置

本文将介绍Hadoop MapReduce 作业配置的操作步骤。

操作步骤

1. 通过主账号登录阿里云 *E-MapReduce* 控制台。
2. 单击上方的**数据开发**页签，进入**项目列表**页面。
3. 单击对应项目右侧的**工作流设计**，进入**作业编辑**页面。
4. 在页面左侧，在需要操作的文件夹上单击右键，选择**新建作业**。
5. 填写作业名称，作业描述。
6. 选择 Hadoop 作业类型。表示创建的作业是一个 Hadoop Mapreduce 作业。这种类型的作业，其后台实际上是通过以下的方式提交的 Hadoop 作业。

```
hadoop jar xxx.jar [MainClass] -Dxxx ....
```

7. 单击**确定**。

> **说明：**
> 您还可以通过在文件夹上单击右键，进行创建子文件夹、重命名文件夹和删除文件夹操作。

8. 在**作业内容**输入框中填写提交该 job 需要提供的命令行参数。这里需要说明的是，这个输入框中需要填写的内容从 hadoop jar 后面的第一个参数开始填写。也就是说，输入框中第一个要填写的是运行该作业需要提供的 jar 包所在地址，然后后面紧跟 [MainClass] 以及其他用户可以自行提供的命令行参数。

   举个例子，假设用户想要提交一个 Hadoop 的 sleep job，该 job不读写任何数据，只是提交一些 mapper 和 reducer task 到集群中，每个 task sleep 一段时间，然后 job 成功。在 Hadoop

中（hadoop-2.6.0 为例）以，该 job 被打包在 Hadoop 发行版的 hadoop-mapreduce-client-jobclient-2.6.0-tests.jar 中。那么，若是在命令行中提交该 job，则命令如下：

```
hadoop jar /path/to/hadoop-mapreduce-client-jobclient-2.6.0-tests.
jar sleep -m 3 -r 3 -mt 100 -rt 100
```

要在 E-MapReduce 中配置这个作业，那么作业配置页面的**作业内容**输入框中，需要填写的内容即为：

```
/path/to/hadoop-mapreduce-client-jobclient-2.6.0-tests.jar sleep -m
3 -r 3 -mt 100 -rt 100
```

说明：

这里用的 jar 包路径是 E-MapReduce 宿主机上的一个绝对路径，这种方式有一个问题，就是用户可能会将这些 jar 包放置在任何位置，而且随着集群的创建和释放，这些 jar 包也会跟着释放而变得不可用。所以，请使用以下方法上传 jar 包：

1. 用户将自己的 jar 包上传到 OSS 的 bucket 中进行存储，当配置 Hadoop 的参数时，单击**选择 OSS 路径**，从 OSS 目录中进行选择要执行的 jar 包。系统会为用户自动补齐 jar 包所在的 OSS 地址。请务必将代码的 jar 的前缀切换为 ossref（单击**切换资源类型**），以保证这个 jar 包会被 E-MapReduce 正确下载。
2. 单击**确定**，该 jar 包所在的 OSS 路径地址就会自动填充到**应用参数**选项框中。作业提交的时候，系统能够根据这个路径地址自动从 OSS 找到相应的 jar 包。
3. 在该 OSS 的 jar 包路径后面，即可进一步填写作业运行的其他命令行参数。

9. 单击**保存**，作业配置即定义完成。

上面的例子中，sleep job 并没有数据的输入输出，如果作业要读取数据，并输出处理结果（比如 wordcount），则需要指定数据的 input 路径和 output 路径。用户可以读写 E-MapReduce 集群 HDFS 上的数据，同样也可以读写 OSS 上的数据。如果需要读写 OSS 上的数据，只需要在填写 input 路径和 output 路径时，数据路径写成 OSS 上的路径地址即可，例如：

```
jar ossref://emr/checklist/jars/chengtao/hadoop/hadoop-mapreduce-
examples-2.6.0.jar randomtextwriter -D mapreduce.randomtextwriter.
```

```
totalbytes=320000 oss://emr/checklist/data/chengtao/hadoop/Wordcount/
Input
```

4.2 Hive 作业配置

E-MapReduce 中，用户申请集群的时候，默认为用户提供了 Hive 环境，用户可以直接使用 Hive 来创建和操作自己的表和数据。

操作步骤

1. 用户需要提前准备好 Hive SQL 的脚本，例如：

```
USE DEFAULT;
 DROP TABLE uservisits;
 CREATE EXTERNAL TABLE IF NOT EXISTS uservisits  (sourceIP STRING,
destURL STRING,visitDate STRING,adRevenue DOUBLE,user
 Agent STRING,countryCode STRING,languageCode STRING,searchWord
STRING,duration INT ) ROW FORMAT DELIMITED FIELDS TERMI
 NATED BY ',' STORED AS SEQUENCEFILE LOCATION '/HiBench/Aggregation/
Input/uservisits';
 DROP TABLE uservisits_aggre;
 CREATE EXTERNAL TABLE IF NOT EXISTS uservisits_aggre ( sourceIP
STRING, sumAdRevenue DOUBLE) STORED AS SEQUENCEFILE LO
 CATION '/HiBench/Aggregation/Output/uservisits_aggre';
 INSERT OVERWRITE TABLE uservisits_aggre SELECT sourceIP, SUM(
adRevenue) FROM uservisits GROUP BY sourceIP;
```

2. 将该脚本保存到一个脚本文件中，例如叫 *uservisits_aggre_hdfs.hive*，然后将该脚本上传到 OSS 的某个目录中（例如：*oss://path/to/uservisits_aggre_hdfs.hive*）。
3. 通过主账号登录阿里云 *E-MapReduce* 控制台。
4. 单击上方的**数据开发**页签，进入**项目列表**页面。
5. 单击对应项目右侧的**工作流设计**，进入**作业编辑**页面。
6. 在页面左侧，在需要操作的文件夹上单击右键，选择**新建作业**。
7. 填写作业名称，作业描述。
8. 选择 Hive 作业类型，表示创建的作业是一个 Hive 作业。这种类型的作业，其后台实际上是通过以下的方式提交。

```
hive [user provided parameters]
```

9. 单击**确定**。

说明：

您还可以通过在文件夹上单击右键，进行创建子文件夹、重命名文件夹和删除文件夹操作。

10. 在作业内容输入框中填入 Hive 命令后续的参数。例如，如果需要使用刚刚上传到 OSS 的 Hive 脚本，则填写的内容如下：

```
-f ossref://path/to/uservisits_aggre_hdfs.hive
```

您也可以单击选择 **OSS** 路径，从 OSS 中进行浏览和选择，系统会自动补齐 OSS 上 Hive 脚本的绝对路径。请务必将 Hive 脚本的前缀修改为 ossref（单击切换资源类型），以保证 E-MapReduce 可以正确下载该文件。

11. 单击保存，Shell 作业即定义完成。

4.3 Pig 作业配置

E-MapReduce 中，用户申请集群的时候，默认为用户提供了 Pig 环境，用户可以直接使用 Pig 来创建和操作自己的表和数据。

操作步骤

1. 用户需要提前准备好 Pig 的脚本，例如：

```
```shell
 /*
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions
and
 * limitations under the License.
 */
 -- Query Phrase Popularity (Hadoop cluster)
 -- This script processes a search query log file from the Excite
search engine and finds search phrases that occur with particular
high frequency during certain times of the day.
 -- Register the tutorial JAR file so that the included UDFs can be
called in the script.
 REGISTER oss://emr/checklist/jars/chengtao/pig/tutorial.jar;
 -- Use the  PigStorage function to load the excite log file into
the “raw” bag as an array of records.
 -- Input: (user,time,query)
 raw = LOAD 'oss://emr/checklist/data/chengtao/pig/excite.log.bz2'
USING PigStorage('\t') AS (user, time, query);
```
```

```
 -- Call the NonURLDetector UDF to remove records if the query field
 is empty or a URL.
 clean1 = FILTER raw BY org.apache.pig.tutorial.NonURLDetector(query
);
 -- Call the ToLower UDF to change the query field to lowercase.
 clean2 = FOREACH clean1 GENERATE user, time, org.apache.pig.
tutorial.ToLower(query) as query;
 -- Because the log file only contains queries for a single day, we
are only interested in the hour.
 -- The excite query log timestamp format is YYMMDDHHMMSS.
 -- Call the ExtractHour UDF to extract the hour (HH) from the time
field.
 houred = FOREACH clean2 GENERATE user, org.apache.pig.tutorial.
ExtractHour(time) as hour, query;
 -- Call the NGramGenerator UDF to compose the n-grams of the query.
 ngramed1 = FOREACH houred GENERATE user, hour, flatten(org.apache.
pig.tutorial.NGramGenerator(query)) as ngram;
 -- Use the  DISTINCT command to get the unique n-grams for all
records.
 ngramed2 = DISTINCT ngramed1;
 -- Use the  GROUP command to group records by n-gram and hour.
 hour_frequency1 = GROUP ngramed2 BY (ngram, hour);
 -- Use the  COUNT function to get the count (occurrences) of each n
-gram.
 hour_frequency2 = FOREACH hour_frequency1 GENERATE flatten($0),
COUNT($1) as count;
 -- Use the  GROUP command to group records by n-gram only.
 -- Each group now corresponds to a distinct n-gram and has the
count for each hour.
 uniq_frequency1 = GROUP hour_frequency2 BY group::ngram;
 -- For each group, identify the hour in which this n-gram is used
with a particularly high frequency.
 -- Call the ScoreGenerator UDF to calculate a "popularity" score
for the n-gram.
 uniq_frequency2 = FOREACH uniq_frequency1 GENERATE flatten($0),
flatten(org.apache.pig.tutorial.ScoreGenerator($1));
 -- Use the  FOREACH-GENERATE command to assign names to the fields
.
 uniq_frequency3 = FOREACH uniq_frequency2 GENERATE $1 as hour, $0
as ngram, $2 as score, $3 as count, $4 as mean;
 -- Use the  FILTER command to move all records with a score less
than or equal to 2.0.
 filtered_uniq_frequency = FILTER uniq_frequency3 BY score > 2.0;
 -- Use the  ORDER command to sort the remaining records by hour and
 score.
 ordered_uniq_frequency = ORDER filtered_uniq_frequency BY hour,
score;
 -- Use the  PigStorage function to store the results.
 -- Output: (hour, n-gram, score, count, average_counts_among
_all_hours)
 STORE ordered_uniq_frequency INTO 'oss://emr/checklist/data/
chengtao/pig/script1-hadoop-results' USING PigStorage();
 ```
```

2. 将该脚本保存到一个脚本文件中，例如叫 *script1-hadoop-oss.pig*，然后将该脚本上传到 OSS 的某个目录中（例如：*oss://path/to/script1-hadoop-oss.pig*）。
3. 通过主账号登录阿里云 *E-MapReduce* 控制台。
4. 单击上方的**数据开发**页签，进入**项目列表**页面。
```

5. 单击对应项目右侧的工作流设计，进入作业编辑页面。
6. 在页面左侧，在需要操作的文件夹上单击右键，选择新建作业。
7. 填写作业名称，作业描述。
8. 选择 Pig 作业类型，表示创建的作业是一个 Pig 作业。这种类型的作业，其后台实际上是通过以下的方式提交。

```
pig [user provided parameters]
```

9. 单击确定。

> 说明：
> 您还可以通过在文件夹上单击右键，进行创建子文件夹、重命名文件夹和删除文件夹操作。

10. 在作业内容输入框中填入 Pig 命令后续的参数。例如，如果需要使用刚刚上传到 OSS 的 Pig 脚本，则填写如下：

```
-x mapreduce ossref://emr/checklist/jars/chengtao/pig/script1-hadoop
-oss.pig
```

    您也可以单击选择 **OSS** 路径，从 OSS 中进行浏览和选择，系统会自动补齐 OSS 上 Pig 脚本的绝对路径。请务必将 Pig 脚本的前缀修改为 ossref（单击切换资源类型），以保证 E-MapReduce 可以正确下载该文件。

11. 单击保存，Shell 作业即定义完成。

4.4 Spark 作业配置

本文将介绍Spark 作业配置的操作步骤。

操作步骤

1. 通过主账号登录阿里云 *E-MapReduce* 控制台，进入集群列表页面。
2. 单击上方的数据开发页签，进入项目列表页面。
3. 单击对应项目右侧的工作流设计，进入作业编辑页面。
4. 在页面左侧，在需要操作的文件夹上单击右键，选择新建作业。
5. 填写作业名称，作业描述。

6. 选择 Spark 作业类型，表示创建的作业是一个 Spark 作业。Spark 作业在 E-MapReduce 后台使用以下的方式提交：

```
spark-submit [options] --class [MainClass] xxx.jar args
```

7. 单击确定。

说明：

您还可以通过在文件夹上单击右键，进行创建子文件夹、重命名文件夹和删除文件夹操作。

8. 在作业内容输入框中填写提交该 Spark 作业需要的命令行参数。请注意，应用参数框中只需要填写**spark-submit**之后的参数即可。以下分别示例如何填写创建 Spark 作业和 pyspark 作业的参数。

- 创建 Spark 作业

  新建一个 Spark WordCount 作业。

  - 作业名称： Wordcount
  - 类型：选择 Spark
  - 应用参数：
    - 在命令行下完整的提交命令是：

```
spark-submit --master yarn-client --driver-memory 7G --
executor-memory 5G --executor-cores 1 --num-executors 32 --
class com.aliyun.emr.checklist.benchmark.SparkWordCount emr
-checklist_2.10-0.1.0.jar oss://emr/checklist/data/wc oss://
emr/checklist/data/wc-counts 32
```

    - 在 E-MapReduce 作业的作业内容输入框中只需要填写：

```
--master yarn-client --driver-memory 7G --executor-memory 5G
 --executor-cores 1 --num-executors 32 --class com.aliyun.emr
.checklist.benchmark.SparkWordCount ossref://emr/checklist/
jars/emr-checklist_2.10-0.1.0.jar oss://emr/checklist/data/wc
 oss://emr/checklist/data/wc-counts 32
```

注意：

作业 Jar 包保存在 OSS 中，引用这个 Jar 包的方式是 *ossref://emr/checklist/jars/emr-checklist_2.10-0.1.0.jar*。您可以单击选择 **OSS** 路径，从 OSS 中进行浏览和选择，系统会自动补齐 OSS 上 Spark 脚本的绝对路径。请务必将默认的 oss 协议切换成 ossref 协议。

- 创建 pyspark 作业

E-MapReduce 除了支持 Scala 或者 Java 类型作业外，还支持 python 类型 Spark 作业。以下新建一个 python 脚本的 Spark Kmeans 作业。

- 作业名称：Python-Kmeans
- 类型：Spark
- 应用参数：

```
--master yarn-client --driver-memory 7g --num-executors 10 --
executor-memory 5g --executor-cores 1  ossref://emr/checklist/
python/kmeans.py oss://emr/checklist/data/kddb 5 32
```

- 支持 Python 脚本资源的引用，同样使用 ossref 协议。
- pyspark 目前不支持在线安装 Python 工具包。

9. 单击保存，Spark 作业即定义完成。

4.5 Spark SQL 作业配置

本文将介绍Spark SQL作业配置的操作步骤。

说明：

Spark SQL 提交作业的模式默认是 yarn-client 模式。

操作步骤

1. 通过主账号登录阿里云 *E-MapReduce* 控制台，进入集群列表页面。
2. 单击上方的**数据开发**页签，进入**项目列表**页面。
3. 单击对应项目右侧的**工作流设计**，进入**作业编辑**页面。
4. 在页面左侧，在需要操作的文件夹上单击右键，选择**新建作业**。
5. 填写作业名称，作业描述。
6. 选择 Spark SQL 作业类型，表示创建的作业是一个 Spark SQL 作业。Spark SQL 作业在 E-MapReduce 后台使用以下的方式提交：

```
spark-sql [options] [cli option]
```

7. 单击**确定**。

说明：

您还可以通过在文件夹上单击右键，进行创建子文件夹、重命名文件夹和删除文件夹操作。

8. 在作业内容输入框中填入 Spark SQL 命令后续的参数。

- -e 选项

  -e 选项可以直接写运行的 SQL，在作业作业内容输入框中直接输入，如下所示：

  ```
  -e "show databases;"
  ```

- -f 选项

  -f 选项可以指定 Spark SQL 的脚本文件。通过将编写好的 Spark SQL 脚本文件放在 OSS 上，可以更灵活，建议您使用这种运行方式。如下所示：

  ```
  -f ossref://your-bucket/your-spark-sql-script.sql
  ```

9. 单击保存，Spark SQL 作业即定义完成。

4.6 Shell 作业配置

本文将介绍Shell作业配置的操作步骤。

> 注意：
> 目前 Shell 脚本默认是使用 Hadoop 用户执行的，如果需要使用 root 用户，可以使用 sudo。请谨慎使用 Shell 脚本作业。

操作步骤

1. 通过主账号登录阿里云 *E-MapReduce* 控制台，进入集群列表页面。
2. 单击上方的数据开发页签，进入项目列表页面。
3. 单击对应项目右侧的工作流设计，进入作业编辑页面。
4. 在页面左侧，在需要操作的文件夹上单击右键，选择新建作业。
5. 填写作业名称，作业描述。
6. 选择 Shell 作业类型，表示创建的作业是一个 Bash Shell 作业。
7. 单击确定。

   > 说明：
   > 您还可以通过在文件夹上单击右键，进行创建子文件夹、重命名文件夹和删除文件夹操作。

8. 在作业内容输入框中填入 Shell 命令后续的参数。

   - -c 选项

-c 选项可以直接设置要运行的 Shell 脚本，在作业作业内容输入框中直接输入，如下所示：

```
-c "echo 1; sleep 2; echo 2; sleep 4; echo 3; sleep 8; echo 4;
sleep 16; echo 5; sleep 32; echo 6; sleep 64; echo 8; sleep 128;
echo finished"
```

- -f 选项

  -f 选项可以直接运行 Shell 脚本文件。通过将 Shell 脚本文件上传到 OSS 上，在 job 参数里面可以直接制定 OSS 上的 Shell 脚本，比使用 -c 选项更加灵活，如下所示：

  ```
  -f ossref://mxbucket/sample/sample-shell-job.sh
  ```

9. 单击保存，Shell 作业即定义完成。

4.7 Sqoop 作业配置

本文将介绍Sqoop作业配置的操作步骤。

> 说明：
> 只有 E-MapReduce 产品版本 V1.3.0（包括）以上支持 Sqoop 作业类型。在低版本集群上运行 Sqoop 作业会失败，errlog 会报不支持的错误。参数细节请参见数据传输 *Sqoop*。

操作步骤

1. 通过主账号登录阿里云 *E-MapReduce* 控制台，进入集群列表页面。
2. 单击上方的数据开发页签，进入项目列表页面。
3. 单击对应项目右侧的工作流设计，进入作业编辑页面。
4. 在页面左侧，在需要操作的文件夹上单击右键，选择新建作业。
5. 填写作业名称，作业描述。
6. 选择 Sqoop 作业类型，表示创建的作业是一个 Sqoop 作业。Sqoop 作业在 E-MapReduce 后台使用以下的方式提交：

   ```
   sqoop [args]
   ```

7. 在作业内容输入框中填入 Sqoop 命令后续的参数。
8. 单击确定。

   > 说明：
   > 您还可以通过在文件夹上单击右键，进行创建子文件夹、重命名文件夹和删除文件夹操作。

9. 单击保存，Sqoop 作业即定义完成。

4.8 作业操作

您可以对作业进行创建、克隆、修改，删除操作。

作业的创建

一个新作业可以在任何时候被创建。被创建的作业目前只可以在所创建的 Region 内被使用。

作业的克隆

完全的克隆一个已经存在作业的配置。同样也只限定在同一个 Region 内。

作业的修改

如果要将作业加入到一个执行计划中，需要保证该执行计划当前没有在运行中，同时也需要保证执行计划的周期调度没有在调度中，这个时候才可以修改该作业。

如果要将这个作业加入到多个执行计划中，需停止要加入的所有执行计划的运行和周期调度后才可以修改。因为修改作业会导致所有使用该作业的执行计划也发生变化，可能会导致正在执行的或者周期调度的执行计划的错误。

如果想要进行调试，推荐使用克隆功能，完成调试后，替换执行计划中的原作业。

作业的删除

和修改一样，只有在作业加入的执行计划当前没有在运行中，同时周期调度也没有在调度中的情况下，才能被删除。

4.9 作业日期设置

在创建作业过程中，支持在作业参数中设置时间变量通配符。

变量通配符格式

E-MapReduce 所支持的变量通配符的格式为**`${dateexpr-1d}`** 或者 **`${dateexpr-1h}`** 的格式。例如，假设当前时间为 `20160427 12:08:01`：

- 如果在作业参数中写成 **`${yyyyMMdd HH:mm:ss-1d}`**，那么这个参数通配符在真正执行的时候会被替换成 `20160426 12:08:01`，即在当前日期上减了一天并精确到了秒。
- 如果写成 **`${yyyyMMdd-1d}`**，则执行时会替换成 `20160426`， 表示当前日期的前一天。
- 如果写成 **`${yyyyMMdd}`**，则会被替换成 `20160427`，直接表示当前的日期。

dateexpr 表示标准的时间格式表达式，对应的时间会按照该表达式指定的格式进行格式化，后面可以再跟上对应加减的时间。支持表达式后面的加减 1d（加减1天），也可以写成加减 N 天或者加减

N 小时，例如 **${yyyyMMdd-5d}**、**${yyyyMMdd+5d}**、**${yyyyMMdd+5h}**、**${yyyyMMdd-5h}**都可以支持，对应的替换方式和前面描述的一致。

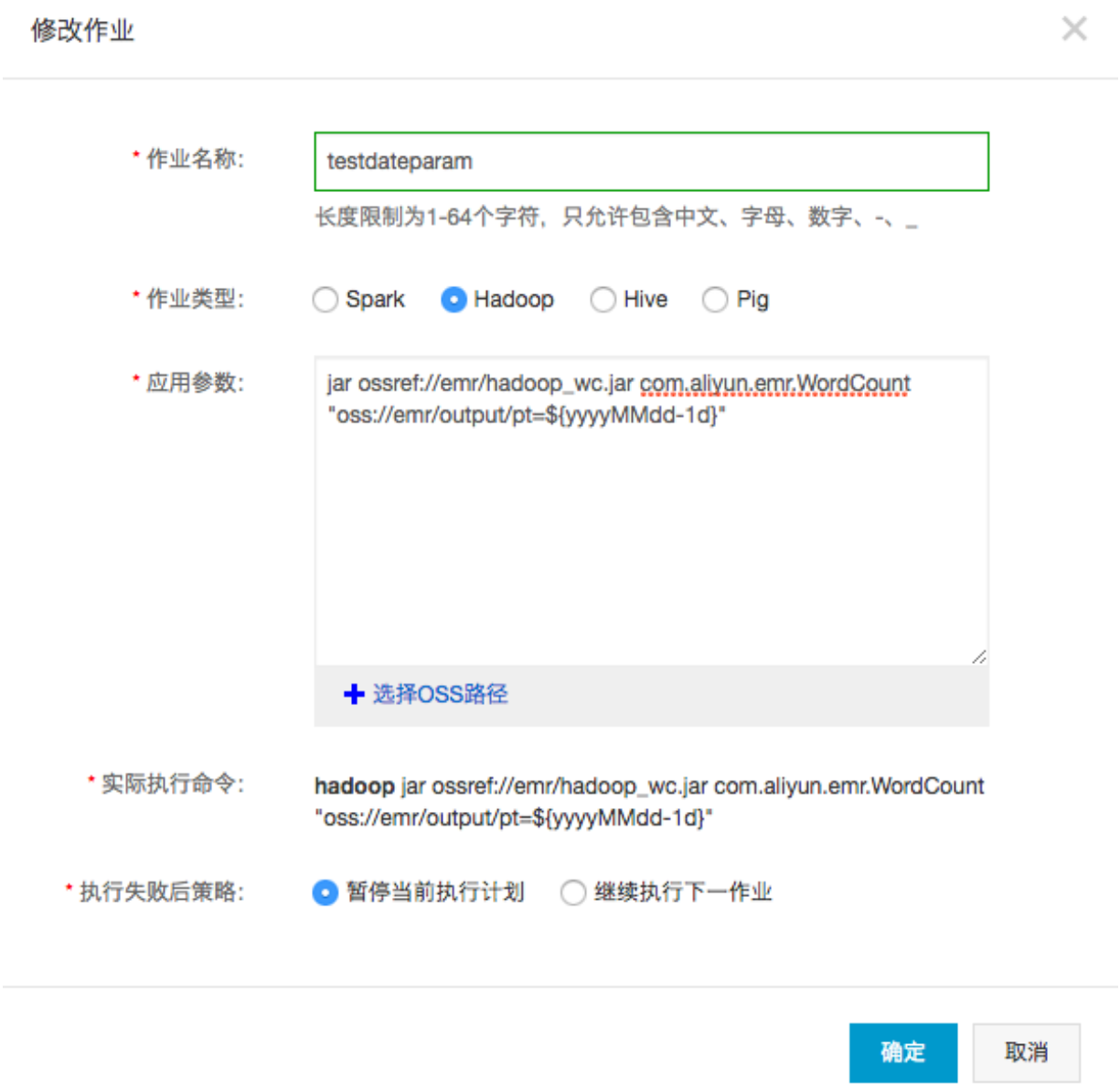
说明：

目前 E-MapReduce 仅支持小时和天维度的加减，即只支持在 dateexpr 后面 +Nd、-Nd、+Nh、-Nh 的形式（dateexpr 为时间格式表达式，N 为整数）。

示例

作业中的应用参数在实际执行时会被替换成：

```
jar ossref://emr/jar/hadoop/hadoop_wc.jar com.aliyun.emr.WordCount oss
://emr/output/pt=20160426
```



5 执行计划

5.1 创建执行计划

执行计划是一组作业的集合，他们通过调度上的配置，可以被一次性或者周期性的执行。他可以在一个现有的 E-MapReduce 集群上运行，也可以动态的按需创建出一个临时集群来运行作业。它最大的优势就是跑多少就用多少资源，最大化的节省资源的浪费。

操作步骤

1. 登录阿里云 *E-MapReduce* 控制台。
2. 选择地域（Region）。
3. 单击上方的**老版作业调度**页签，进入作业列表页面
4. 单击左侧的**执行计划**页签，进入执行计划页面
5. 单击右上角的**创建执行计划**，进入创建执行计划页面。
6. 在选择集群方式页面上，有两个选项，分别是**按需创建**和**已有集群**。
   a. **按需创建**：创建一个全新的集群，用来运行作业。
      - 一次性调度的执行计划，会在开始执行的时候创建对应配置的集群，并在运行完成以后释放该集群。具体创建参数说明参考创建集群。
      - 周期调度的执行计划，会在每一个调度周期开始时，按照用户的设置创建出一个新的集群运行作业，并在运行结束后，释放集群。
   b. **已有集群**：使用一个已有的集群，并且该集群要符合以下要求：
      - 目前只有运行中和空闲这 2 个状态的集群可以被提交执行计划。

   如果选择**已有集群**，则进入选择集群页面。用户可选择要将该执行计划关联到的集群。
7. 单击下一步，进入配置作业页面。左边表中会列出用户所有的作业，可以单击选中需要执行的作业，然后单击中央的右向按钮将作业加入已选作业队列。已选作业队列中的作业会被按排列顺序提交到集群中执行。同一个作业可以被添加多次，就会多次执行。如果您还没有创建任何作业，请您先参见创建作业的操作说明创建作业。
8. 单击下一步，进入配置调度方式页面。配置项说明如下：
   a. **名称**：长度限制为 1-64 个字符，只允许包含中文、字母、数字、’-‘、’_’。
   b. **调度策略**

- **手动执行**：创建完执行计划以后，并不会自动执行。需要用户手动执行。一旦已经在运行中了，不可以被再次执行。
- **周期调度**：创建完执行计划以后，周期调度功能会立刻启动。并在用户设置的调度时间点上开始执行。可以在列表页面关闭周期调度。当调度执行开始的时候，上一周期的执行还未结束，本次调度就会被忽略。

c. **设置调度周期**：可以有天或小时两种调度的周期。天默认是一天，且无法更改。若选择小时，则可设置具体间隔时间，范围从 1-23。

d. **首次执行时间**：调度有效的开始时间。从这个时间开始，按照调度周期进行周期调度。第一次调度按照实际的时间满足要求的最近一个时间点开始调度。

9. 单击**确认**，完成执行计划的创建。

其他

- 周期调度示例

这个设置表示，从 2015 年 10 月 31 日 10 点 0 分开始第一次调度，以后每隔一天调度一次。第二次调度是 2015 年 11 月 1 日 10 点 0 分。

- 作业的执行顺序

  执行计划中的作业，按照用户选择的作业在作业列表中的顺序，从第一个开始一直执行到最后一个。

- 多个执行计划的执行顺序

  每一个执行计划都可以看做是一个整体。当多个执行计划被提交到同一个集群上后，每一个执行计划都会按照自身内部的作业顺序提交作业，和单个执行计划的顺序是一致。而多个执行计划之间的作业是并行的。

- 实践示例 —— 前期作业调试

在作业的调试阶段，如果经常用按需自动创建集群的方式会比较慢，每次都需要启动集群会花费不少的时间。推荐的方式是：先手动创建一个集群，然后在执行计划中，选择关联该集群来运行作业，并设置调度方式为立即执行。调试的时候，每次都通过单击执行计划列表页上的“立即运行”来多次运行，查看结果。一旦作业调试完成，修改执行计划。将关联现有集群的方式，修改为按需创建新集群。并将调度方式修改为周期调度（视实际情况而定）。后续就可以按需自动跑任务了。

5.2 管理执行计划

您可以通过以下步骤管理，查看和修改执行计划。

1. 登录阿里云 *E-MapReduce* 控制台。
2. 选择地域（Region）。
3. 单击上方的**老版作业调度**页签，进入作业列表页面
4. 单击左侧的**执行计划**页签，进入执行计划页面
5. 找到相应的执行计划条目，单击其操作栏中的**管理**，进入执行计划管理页面。在这里您可以：

   - 查看执行计划详情

     您可以查看到该执行计划的名称、关联集群、作业配置等基本信息，还有其调度策略、调度状态、报警信息等。

   - 修改执行计划

     > 注意：
     > 一个执行计划当前并没有在运行中且它没有在被周期调度中，才能够被修改。如果是一个立即执行的执行计划，只要它当前没有在运行中就可以被修改。如果是一个周期调度的执行计划，首先要等待它当前的运行结束，然后确认它是否正在被周期调度中，如果是请单击**停止调度**，然后才可修改执行计划。

     独立修改

     每一个单独的模块，都可以被独立的修改。单击条目右侧的**修改**图标即可进行修改。

   - 配置报警通知

     共有三类报警通知：

     — 启动超时通知：周期调度任务，在指定时间点，没有正常调度，并在 10 分钟的超时时间内，仍然没有调度执行，发送报警通知。

— 执行失败通知：执行计划内有任何一个作业失败，发送报警通知。

— 执行成功通知：执行计划内的所有作业执行成功，发送通知。

- 运行与查看结果

  在**基本信息**中的**调度状态**右侧，当执行计划可以被运行的时候，会有**立即运行**的按钮，单击后即会产生一次调度执行。

  在页面的最下方，是运行记录的部分，会显示执行计划每次被执行的实例，可以方便用户查看对应的作业列表和日志。

5.3 执行计划列表

执行计划列表用来展示您所有的执行计划的基本信息。

- **ID/名称**：执行计划的 ID 和对应的名称。
- **最近执行集群**：最近一次执行该执行计划的集群，是一个按需创建的集群或是一个关联的已有集群。如果是按需的，那么在集群的名字下面会显示（自动创建），表示这个集群是有 E-MapReduce 按需自动创建出来的，运行完成以后会自动释放。
- **最近运行**：最近一次执行计划的运行状态。

  — **开始时间**：最近一次执行计划开始的时间。

  — **运行时间**：最近一次执行计划的运行时长。

  — **运行状态**：最近一次执行计划的运行状态。
- **调度状态**：是否在调度中还是调度已经停止。只有周期作业才会有调度状态。
- **操作**

  — **管理**：查看执行计划详情，修改执行计划。

  — **立即运行**：非调度中且非运行中，才能手动运行。单击后，会立刻运行一次执行计划。

  — **更多**

    ■ **启动调度/停止调度**：当调度停止状态时出现启动调度，单击即会开始调度。当调度运行中显示停止调度，单击即会停止调度。只有周期执行计划才有此按钮。

■ 运行记录：单击会进入执行计划中的作业日志查看页面。

■ 删除：删除执行计划。调度中或者运行中的执行计划不能被删除。

5.4 作业结果和日志查看

本文将介绍如何查看作业结果和对应作业的运行日志。

执行记录查看

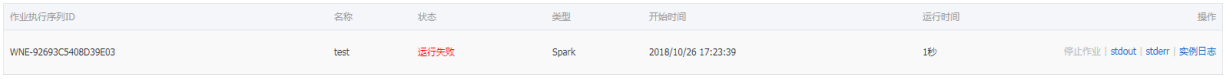
1. 登录阿里云 E-MapReduce 控制台。
2. 选择地域（Region）。
3. 单击上方的老版作业调度页签，进入作业列表页面
4. 单击左侧的执行计划页签，进入执行计划页面
5. 单击相应执行计划条目右侧操作中的更多 > 运行记录，即可进入执行记录页面。

- 执行序列 ID： 本次执行记录的执行次数，表明了它在整个执行队列中的顺序位置。比如第一次执行就是1，第n次就是n。
- 运行状态： 每一次执行记录的运行状态。
- 开始时间： 执行计划开始运行的时间。
- 运行时间： 到查看页面当时为止，一共运行的时间。
- 执行集群： 执行计划运行的集群，可以是按需也可以是一个关联的已有集群。点击可以前往集群的详情页查看。
- 操作

  查看作业列表：单击该按钮，即可进入单次执行计划的作业列表查看每个作业的执行情况。

作业记录查看

在作业列表中可以查看单次执行计划的执行记录中的作业列表，以及每一个作业的具体信息。



- 作业执行序列**ID**：作业每一次执行都会产生一个对应的 ID，它和作业本身的 ID 是不同的。这个 ID 可以想象成作业每运行一次的一个记录的唯一标示，您可用其在 OSS 上进行日志查询。
- 名称：作业的名称。
- 状态：作业的运行状态。
- 类型：作业的类型。
- 开始时间：这个作业开始运行的时间，都已经转换为本地时间。
- 运行时间：这个作业一共运行了多久，以秒为单位。
- 操作
  - 停止作业：无论作业在提交中还是在运行中，都可以被停止。如果是提交中，那么停止作业会让这个作业不执行。如果是在运行中，那么这个作业会被 kill 掉。
  - **stdout**：记录 master 进程的标准输出（即通道 1）的所有输出内容。如果运行作业的集群没有打开日志保存，不会有此查看功能。
  - **stderr**：记录 master 进程的诊断输出（即通道 2）的所有输出内容。如果运行作业的集群没有打开日志保存，不会有此查看功能。
  - 实例日志：查看作业的所有 worker 的节点的日志。如果运行作业的集群没有打开日志保存，不会有此查看功能。

作业**worker**日志查看

- 云服务器实例 **ID/IP**：运行作业的 ECS 实例 ID，以及对应的内网 IP。
- 容器 **ID**：Yarn 运行的容器 ID。
- 类型：日志的不同类型。stdout 与 stderr，来自不同的输出。
- 操作

  查看日志：单击对应的类型，查看对应的日志。

5.5 多执行计划并行执行

为了最大化利用集群的可用计算资源，目前可以将多个执行计划挂载到同一个集群来达到多个执行计划并行执行的效果。

总结为如下几点：

- 同一个执行计划内的作业是串行执行的，默认认为前序作业执行完毕，后序作业才能被提交执行。

- 在集群资源足够的情况下，如果想要让多个作业达到并行执行的效果，需要创建多个不同的执行计划，同时关联到同一个集群提交运行即可（默认一个集群最多支持 20 个执行计划同时执行）。
- 目前管控系统支持将关联到同一个集群的执行计划并行提交到 Yarn，但如果集群本身资源不足，还是可能阻塞在 Yarn 队列中等待调度。

创建执行计划并关联到集群的流程参见：创建执行计划。

6 报警管理

6.1 集群报警管理

您可以在云监控产品中设置E-MapReduce的集群报警规则。云监控通过监控E-MapReduce集群的CPU空闲率、内存容量、磁盘容量等多个监控项，帮助用户监测集群的运行状态。如果集群在运行过程中触发了告警规则，可以及时地通知通知组中的联系人，以便及时处理问题。

设置报警规则

设置E-MapReduce的集群报警规则，请按照以下步骤操作：

1. 进入云监控产品的管理控制台。
2. 在左侧的导航栏中单击云服务监控 > **E-MapReduce**，进入E-MapReduce监控列表页面。
3. 单击报警规则页签。
4. 在页面的右上角单击创建报警规则，进入报警规则配置页面。
5. 在关联资源部分，配置产品和资源范围。
   - 产品：从下拉菜单中选择E-MapReduce。
   - 资源范围：报警规则的作用范围，包括全部资源、应用分组、集群三种。选择全部资源时，报警的资源最多1000个，超过1000个可能会出现达到阈值不报警的问题，建议使用应用分组按业务划分资源后再设置报警。
     - 全部资源：表示该规则作用在账户下对应产品的所有实例上。比如设置了全部资源粒度的实例CPU使用率大于80%报警，则只要账户下有实例的CPU使用率大于80%，就会命中这条规则。
     - 应用分组：表示该规则作用在某个应用分组下的所有实例上。比如设置了应用分组粒度的主机CPU使用率大于80%报警，则只要这个分组下有主机CPU使用率大于80%，就会命中这条规则。
     - 集群：表示该规则只作用在某个集群的实例上。比如设置了实例粒度的主机 CPU 使用率大于80%报警，则只要这个集群中有实例的CPU使用率大于80% ，就会命中这条规则。
6. 在设置报警规则部分进行配置。
   - 规则名称：设置报警规则的名称。

- **规则描述**：报警规则的主体，定义在监控项数据满足何种条件时，触发报警规则。例如规则描述为CPU使用率1分钟平均值>=90%，则报警服务会1分钟检查一次1分钟内的数据是否满足平均值>=90%。关于E-MapReduce集群的监控项，请参见*E-MapReduce*监控。
- **角色**：默认情况下，任意角色都适用。

  单击**添加报警规则**，您可以设置多条报警规则（计费时按多条计算），只要其中一条规则被触发，系统就会给通知组发送通知。
- **通道沉默时间**：指报警发生后如果未恢复正常，间隔多久重复发送一次报警通知。
- **连续几次超过阈值后报警**：连续几次报警的探测结果符合您设置的规则，才会触发报警。例如规则设置为：系统态CPU使用率1分钟内平均值>80%，连续3次超过阈值后报警，则连续出现3次系统态CPU使用率1分钟内平均值大于80%的情况，才会触发报警。
- **生效时间**：报警规则每天的生效时间段。系统仅在生效时间内才会检查监控数据是否需要报警。

7. 在**通知方式**部分进行配置。
   - **通知对象**：在搜索框中输入通知组名称的关键字，快速定位到您想关联的通知组，然后单击右箭头图标，通知组即被加入到右侧的通知列表。如果您还没有创建合适的通知组，单击**快速创建联系人组**进行创建。在右侧通知列表中选定通知组后，单击左箭头图标，即可从通知列表中删除该通知组。
   - **报警级别**：报警信息包括三种严重级别，分别是Critical、Warning和Info。不同的报警级别对应着不同的通知方式。当需要配置为Critical级别时，请购买电话报警资源包。
   - **邮件备注（可选）**：自定义报警邮件补充信息。填写邮件备注后，发送报警的邮件通知中会附带您的备注。
   - **报警回调（可选）**：该功能可以让您将云监控发送的报警通知集成到已有运维体系或消息通知体系中。云监控通过HTTP协议的POST请求推送报警通知到您指定的公网URL，您在接收到报警通知后，可以根据通知内容做进一步处理。详情请参见使用报警回调。
8. 单击**确认**，完成报警规则配置。

7 软件配置

Hadoop、Hive、Pig 等软件含有大量的配置，当需要对其软件配置进行修改时，就可以使用软件配置功能来实现。例如，HDFS 服务器的服务线程数目 dfs.namenode.handler.count 默认是 10，假设要加大到 50；HDFS 的文件块的大小 dfs.blocksize 默认是 128MB，假设系统都是小文件，想要改小到 64MB。

软件配置的作用

目前软件配置操作只能在集群启动的时候执行一次。

操作步骤

1. 登录阿里云 *E-MapReduce* 控制台。
2. 在上方选择所在的地域（Region），所创建集群将会在对应的Region内。
3. 单击**创建集群**，即会进入创建集群的操作界面。
4. 在创建集群的软件配置这一步中可以看到所有包含的软件以及对应的版本。若想修改集群的配置，可以通过**软件自定义配置**（可选）框选择相应的 json 格式配置文件，对集群的默认参数进行覆盖或添加。json 文件的样例内容如下：

```
{
 "configurations": [
     {
         "classification": "core-site",
         "properties": {
             "fs.trash.interval": "61"
         }
     },
     {
         "classification": "hadoop-log4j",
         "properties": {
             "hadoop.log.file": "hadoop1.log",
             "hadoop.root.logger": "INFO",
             "a.b.c": "ABC"
         }
     },
     {
         "classification": "hdfs-site",
         "properties": {
             "dfs.namenode.handler.count": "12"
         }
     },
     {
         "classification": "mapred-site",
         "properties": {
             "mapreduce.task.io.sort.mb": "201"
         }
     },
     {
         "classification": "yarn-site",
```

```
        "properties": {
            "hadoop.security.groups.cache.secs": "251",
            "yarn.nodemanager.remote-app-log-dir": "/tmp/logs1"
        }
    },
    {
        "classification": "httpsfs-site",
        "properties": {
            "a.b.c.d": "200"
        }
    },
    {
        "classification": "capacity-scheduler",
        "properties": {
            "yarn.scheduler.capacity.maximum-am-resource-percent":
 "0.2"
        }
    },
    {
        "classification": "hadoop-env",
        "properties": {
            "BC":"CD"
        },
        "configurations":[
            {
                "classification":"export",
                "properties": {
                    "AB":"${BC}",
                    "HADOOP_CLIENT_OPTS":"\"-Xmx512m -Xms512m $
HADOOP_CLIENT_OPTS\""
                }
            }
        ]
    },
    {
        "classification": "httpfs-env",
        "properties": {
        },
        "configurations":[
            {
                "classification":"export",
                "properties": {
                    "HTTPFS_SSL_KEYSTORE_PASS":"passwd"
                }
            }
        ]
    },
    {
        "classification": "mapred-env",
        "properties": {
        },
        "configurations":[
            {
                "classification":"export",
                "properties": {
                    "HADOOP_JOB_HISTORYSERVER_HEAPSIZE":"1001"
                }
            }
        ]
    },
    {
        "classification": "yarn-env",
```

```
          "properties": {
          },
          "configurations":[
              {
                  "classification":"export",
                  "properties": {
                      "HADOOP_YARN_USER":"${HADOOP_YARN_USER:-yarn1}"
                  }
              }
          ]
      },
      {
          "classification": "pig",
          "properties": {
              "pig.tez.auto.parallelism": "false"
          }
      },
      {
          "classification": "pig-log4j",
          "properties": {
              "log4j.logger.org.apache.pig": "error, A"
          }
      },
      {
          "classification": "hive-env",
          "properties": {
              "BC":"CD"
          },
          "configurations":[
              {
                  "classification":"export",
                  "properties": {
                      "AB":"${BC}",
                      "HADOOP_CLIENT_OPTS1":"\"-Xmx512m -Xms512m $
HADOOP_CLIENT_OPTS1\""
                  }
              }
          ]
      },
      {
          "classification": "hive-site",
          "properties": {
              "hive.tez.java.opts": "-Xmx3900m"
          }
      },
      {
          "classification": "hive-exec-log4j",
          "properties": {
              "log4j.logger.org.apache.zookeeper.ClientCnxnSocketNIO
": "INFO,FA"
          }
      },
      {
          "classification": "hive-log4j",
          "properties": {
              "log4j.logger.org.apache.zookeeper.server.NIOServerCnxn
": "INFO,DRFA"
          }
      }
 ]
```

```
}
```

classification 参数指定要修改的配置文件，**properties** 参数放置要修改的 key value 键值对，默认配置文件有对应的 key 有则只覆盖 value，没有则添加对应的 key value 键值对。

配置文件与 classification 的对应关系如下列表格所示：

- Hadoop

Filename	classification
core-site.xml	core-site
log4j.properties	hadoop-log4j
hdfs-site.xml	hdfs-site
mapred-site.xml	mapred-site
yarn-site.xml	yarn-site
httpsfs-site.xml	httpsfs-site
capacity-scheduler.xml	capacity-scheduler
hadoop-env.sh	hadoop-env
httpfs-env.sh	httpfs-env
mapred-env.sh	mapred-env
yarn-env.sh	yarn-env

- Pig

Filename	classification
pig.properties	pig
log4j.properties	pig-log4j

- Hive

Filename	classification
hive-env.sh	hive-env
hive-site.xml	hive-site
hive-exec-log4j.properties	hive-exec-log4j
hive-log4j.properties	hive-log4j

core-site 这类扁平的 xml 文件只有一层，配置都放在 **properties** 里。而 hadoop-en v这类 sh 文件可能有两层结构，可以通过嵌套 configurations 的方式来设置，请参见示例里 hadoop-env 的部分，为 export 的 **HADOOP_CLIENT_OPTS** 属性添加了 `-Xmx512m -Xms512m` 的设置。

设置好后，确认后单击下一步。

8 引导操作

引导操作的作用是在集群启动 Hadoop 前执行您自定义的脚本，以便安装您需要的第三方软件，或者修改集群运行环境。

引导操作的作用

通过引导操作，您可以安装很多目前集群尚未支持的东西到您的集群上，例如：

- 使用 yum 安装已经提供的软件。
- 直接下载公网上的一些公开的软件。
- 读取 OSS 中您的自有数据。
- 安装并运行一个服务，例如 Flink 或者 Impala，但需要编写的脚本会复杂些。

强烈建议您先用按量付费的集群来进行引导操作的测试，测试都正确以后再创建包年包月的集群。

如何使用

1. 登录阿里云 *E-MapReduce* 控制台。
2. 在上方选择所在的地域（Region），所创建集群将会在对应的 Region 内。
3. 单击**创建集群**，即会进入创建集群的操作界面。
4. 在创建集群的基础配置页面的引导操作部分，单击**添加**，进入添加引导操作界面。
5. 填写添加引导操作界面上的配置项，完成添加。

您最多可以添加 16 个引导操作，它们会按照您指定的顺序在集群初始化时执行。默认会使用 root 账户执行您指定的脚本，您可以在脚本中使用 **su hadoop** 切换到 Hadoop 账户。

引导操作可能会执行失败。为方便您的使用，引导操作失败并不会影响集群的创建。集群创建成功后，您可以在集群详情页集群信息栏内的引导/软件配置查看是否有异常发生。如果有异常，您可以登录到各个节点上查看运行日志，运行日志在 `/var/log/bootstrap-actions` 目录下。

引导操作类型

引导操作有两种，一种是自定义引导操作，另一种是运行条件引导操作。两者的主要区别是运行条件引导操作只会在满足条件的节点上运行您的指定操作。

自定义引导操作

自定义引导操作需要指定引导操作名称和执行脚本在 OSS 中的位置，根据需要指定可选参数。集群初始化时各个节点会下载您指定的 OSS 脚本，直接执行或者附加上可选参数执行。

您可以在脚本中指定从 OSS 下载需要的文件，下面的例子会将 *oss://yourbucket/myfile.tar.gz* 这个文件下载到本地，并解压到 */yourdir* 目录下：

```
#!/bin/bash
osscmd --id=<yourid> --key=<yourkey> --host=oss-cn-hangzhou-internal.aliyuncs.com get oss://<yourbucket>/<myfile>.tar.gz ./<myfile>.tar.gz
mkdir -p /<yourdir>
tar -zxvf <myfile>.tar.gz -C /<yourdir>
```

osscmd 已预安装在节点上，可以直接调用来下载文件。

 注意：

OSS 地址 host 有内网地址、外网地址和 VPC 网络地址之分。如果用经典网络，需要指定内网地址，杭州是 oss-cn-hangzhou-internal.aliyuncs.com。如果用 VPC 网络，要指定 VPC 内网可访问的域名，杭州是 vpc100-oss-cn-hangzhou.aliyuncs.com。

引导操作也可以通过 yum 安装额外的系统软件包，下面的例子会安装 ld-linux.so.2：

```
#!/bin/bash
yum install -y ld-linux.so.2
```

运行条件引导操作

运行条件引导操作的执行脚本是预定义的不需要您额外指定，您只需要指定名称和可选参数。运行条件引导操作必须提供可选参数，可选参数需要包括运行条件和操作命令，以空格间隔。运行条件支持 instance.isMaster=true/false，指定只在 master 或者在非 master 节点上运行。以下示例为运行条件引导操作下面的可选参数指定只在 master 节点上创建目录：

```
instance.isMaster=true mkdir -p /tmp/abc
```

如果需要指定多个操作命令，您可以用分号“；”分割多个语句，例如：`instance.isMaster=true mkdir -p /tmp/abc;mkdir -p /tmp/def`。

9 专有网络

专有网络（Virtual Private Cloud，VPC）为用户创建一个隔离的网络环境，用户可以选择自有的 IP 地址范围、划分网络、配置路由表、网关等。

详见专有网络产品简介。另外通过高速通道 可以实现跨地域或跨用户的 VPC 内网互通、VPC 与物理 IDC 机房互通。

创建专有网络集群

E-MapReduce 在创建集群的时候可以选择网络类型，即经典网络/专有网络，若选择专有网络，需要如下额外操作：

- VPC：选择当前创建的 E-MapReduce 集群归属的VPC，如果还没创建可以进入*VPC* 控制台进行创建，一般一个账号最多创建 2 个 VPC 网络，超过 2 个需要提工单。
- 交换机：E-MapReduce 集群内的 ECS 实例通过交换机进行通信，如果还没创建可以进入*VPC* 控制台，单击**交换机**页签进入交换机页面进行创建，因为交换机有可用区的属性，所以在 E-MapReduce 创建集群时选定了可用区后，创建的交换机也必须属于该可用区。
- 安全组名称：集群所属的安全组，经典网络的安全组不能在 VPC 中使用，VPC 的安全组只能在当前 VPC 中使用。这里只展示用户在 E-MapReduce 产品中创建的安全组。因为一些安全的原因目前尚不支持选择在 E-MapReduce 外创建的安全组。如果需要新建安全组，可以输入安全组的名字完成新建。

示例

不同 VPC 中的 EMR 集群通信（Hive 访问 HBase）

1. 创建集群。

   在 E-MapReduce 上面创建两个集群，Hive 集群 C1 处于 VPC1 中，HBase 集群 C2 处于 VPC2 中，两个集群都在杭州区域。

2. 配置高速通道。

   详细配置请参考同账号*VPC*互连，地域选择同地域即可。

3. ssh 登录 HBase 集群，通过 HBase Shell 创建表。

   ```
   hbase(main):001:0> create 'testfromHbase','cf'
   ```

4. ssh 登录 Hive集群。

a. 修改 hosts，增加如下一行：

```
$zk_ip emr-cluster //$zk_ip为Hbase集群的zk节点IP
```

b. 通过 Hive Shell 访问 HBase。

```
hive> set hbase.zookeeper.quorum=172.16.126.111,172.16.126.112,172
.16.126.113;
hive> CREATE EXTERNAL TABLE IF NOT EXISTS testfromHive (rowkey
STRING, pageviews Int, bytes STRING) STORED BY 'org.apache.hadoop
.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ('hbase.
columns.mapping' = ':key,cf:c1,cf:c2') TBLPROPERTIES ('hbase.table
.name' = 'testfromHbase');
```

此时命令会卡住，然后会报 java.net.SocketTimeoutException 的异常，原因是 HBase 集群的 ECS 所在的安全组限制了相关端口的访问（E-MapReduce 创建的安全组默认只开放 22 端口）E-MapReduce，所以需要给 HBase 集群的安全组增加安全组规则开放端口给 Hive 集群，如下图所示：

10 Python 使用说明

Python 使用说明

Python 2.7支持

从 E-MapReduce 的 2.0.0 版本开始，支持Python 2.7版本。

Python文件安装位置为：usr/local/Python-2.7.11/ 已包含Numpy

Python 3.6支持

EMR 2.10.0、3.10.0及以上版本开始支持python 3.6.4 ，安装目录为 /usr/bin/python3；EMR 2.10.0、3.10.0以下版本默认不支持Python3版本，用户需自行下载安装：

- 下载Python3软件包：*wget*链接地址
- 解压下载文件并安装

```
tar zxvf Python-3.6.4.tgz
cd Python-3.6.4 ./configure --prefix=/usr/local/Python-3.6.4
make && make install
ln -s /usr/local/Python-3.6.4/bin/python3.6 /bin/python3
ln -s /usr/local/Python-3.6.4/bin/pip3 /bin/pip3
```

- 验证Python3的环境

```
[root@emr-header-1 bin]# python3
```

```
Python 3.6.4 (default, Mar 12 2018, 14:03:26)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linuxType “help”, “
copyright”, “credits” or “license” for more information.
```

```
[root@emr-header-1 bin]# pip3 -V
```

```
pip 9.0.1 from /usr/local/Python-3.6.4/lib/python3.6/site-packages (
python 3.6)
```

输出以上信息说明安装成功。

11 开源组件介绍

11.1 Hue 使用说明

目前 E-MapReduce 中支持了 *Hue*，可以通过Apache Knox访问Hue。

准备工作

在集群安全组中设置安全组规则，打开8888端口。

> 注意：
> 设置安全组规则时要针对有限的IP范围。禁止在配置的时候对0.0.0.0/0开放规则。

访问Hue

EMR的Web界面中提供了快速访问Hue的链接，访问链接查看方式如下：

1. 在集群列表页面，单击集群ID右侧的管理。
2. 在页面左侧导航栏中单击**访问链接与端口**。

访问密码

由于Hue默认第一次运行时，如果还没有设置管理员，第一个登录的用户就默认设置为管理员。为了安全起见，EMR会默认生成一个管理员账号与密码，管理员账号是admin，密码通过如下途径查看：

1. 在集群列表页面，单击集群ID右侧的管理。
2. 在服务列表中，选择**Hue**。
3. 单击**配置**页签，其中有一个**admin_pwd**的参数，这个就是随机密码。

忘记密码

如果用户忘记了自己的Hue账号所对应的密码，可以通过以下方式重新创建一个账号：

1. 在集群列表页面，单击集群ID右侧的管理。
2. 在页面左侧导航栏中单击**集群基础信息**。
3. 在主实例组部分获取master节点的公网IP。
4. 通过SSH方式登录master节点。

5. 执行以下命令，创建新账号。

```
/opt/apps/hue/build/env/bin/hue createsuperuser
```

6. 输入新用户名、电子邮件，然后输入密码，再次输入密码后按回车键。

   如果提示 **Superuser created successfully**，则说明新账号创建成功，稍后用新账号登录Hue即可。

添加/修改配置

1. 在EMR管理控制台，单击集群ID右侧的管理。
2. 在服务列表中，选择**Hue**，然后单击**配置**页签。
3. 点击右上角的**自定义配置**按钮，添加需要添加/修改的key/value值，其中key需要遵循下面规范：

```
$section_path.$real_key
```

> 说明：
>
> - **$real_key** 即为需要添加的实际的key，如hive_server_host。
> - **$real_key** 前面的 **$section_path** 可以通过 *hue.ini* 文件进行查看，例如：
>
>   hive_server_host，通过*hue.ini*文件可以看出它属于**[beeswax]**这个section下，则 **$section_path** 为 beeswax。
> - 综上，添加的key为 beeswax.hive_server_host。
> - 同理，如需修改*hue.ini*文件中的多级 section **[desktop] -> [[ldap]] -> [[[ldap_servers]]] -> [[[[users]]]] ->user_name_attr**的值，则需要配置的key为desktop.ldap.ldap_servers.users.user_name_attr。

11.2 Oozie 使用说明

Oozie 使用说明

> 说明：
>
> 阿里云 E-MapReduce 在 2.0.0 及之后的版本中提供了对 Oozie 的支持，如果需要在集群中使用 Oozie，请确认集群的版本不低于 2.0.0。

准备工作

在集群建立出来之后，需要打通 ssh 隧道，详细步骤请参考*SSH* 登录集群。

这里以 MAC 环境为例，使用 Chrome 浏览器实现端口转发（假设集群 master 节点公网 IP 为 **xx.xx.xx.xx**）：

1. 登录到 master 节点。

```
ssh root@xx.xx.xx.xx
```

2. 输入密码。
3. 查看本机的 **id_rsa.pub** 内容（注意在本机执行，不要在远程的 master 节点上执行）。

```
cat ~/.ssh/id_rsa.pub
```

4. 将本机的 **id_rsa.pub** 内容写入到远程 master 节点的 *~/.ssh/authorized_keys* 中（在远端 master 节点上执行）。

```
mkdir ~/.ssh/
vim ~/.ssh/authorized_keys
```

5. 然后将步骤 2 中看到的内容粘贴进来，现在应该可以直接使用 `ssh root@xx.xx.xx.xx` 免密登录 master 节点了。
6. 在本机执行以下命令进行端口转发。

```
ssh -i ~/.ssh/id_rsa -ND 8157 root@xx.xx.xx.xx
```

7. 启动 Chrome（在本机新开 terminal 执行）。

```
/Applications/Google\ Chrome.app/Contents/MacOS/Google\ Chrome --
proxy-server="socks5://localhost:8157" --host-resolver-rules="MAP *
0.0.0.0 , EXCLUDE localhost" --user-data-dir=/tmp
```

访问 Oozie UI 页面

在进行端口转发的 Chrome 浏览器中访问：xx.xx.xx.xx:11000/oozie，localhost:11000/oozie 或者内网 ip:11000/oozie。

提交workflow作业

运行 Oozie 需要先安装 Oozie 的sharelib：*https://oozie.apache.org/docs/4.2.0/WorkflowFunctionalSpec.html#ShareLib*

在 E-MapReduce 集群中，默认给 Oozie 用户安装了 sharelib，即如果使用 Oozie 用户来提交 workflow 作业，则不需要再进行 sharelib 的安装。

由于开启 HA的集群和没有开启 HA 的集群，访问 NameNode 和 ResourceManager 的方式不同，在提交 oozie workflow job 的时候，job.properties 文件中需要指定不同的 NameNode 和 JobTracker（ResourceManager）。具体如下：

- 非 HA 集群

```
nameNode=hdfs://emr-header-1:9000
jobTracker=emr-header-1:8032
```

- HA 集群

```
nameNode=hdfs://emr-cluster
jobTracker=rm1,rm2
```

下面操作示例中，已经针对是否是 HA 集群配置好了，即样例代码不需要任何修改即可以直接运行。关于 workflow 文件的具体格式，请参考 Oozie 官方文档：*https://oozie.apache.org/docs/4.2.0/*.

- 在非 **HA** 集群上提交 **workflow** 作业

1. 登录集群的主 master 节点。

```
ssh root@master公网Ip
```

2. 下载示例代码。

```
[root@emr-header-1 ~]# su oozie
[oozie@emr-header-1 root]$ cd /tmp
[oozie@emr-header-1 tmp]$ wget http://emr-sample-projects.oss-cn-
hangzhou.aliyuncs.com/oozie-examples/oozie-examples.zip
[oozie@emr-header-1 tmp]$ unzip oozie-examples.zip
```

3. 将 Oozie workflow 代码同步到 hdfs 上。

```
[oozie@emr-header-1 tmp]$ hadoop fs -copyFromLocal examples/ /user
/oozie/examples
```

4. 提交 Oozie workflow 样例作业。

```
[oozie@emr-header-1 tmp]$ $OOZIE_HOME/bin/oozie job -config
examples/apps/map-reduce/job.properties -run
```

执行成功之后，会返回一个 jobId，类似：

```
job: 0000000-160627195651086-oozie-oozi-W
```

5. 访问 Oozie UI 页面，可以看到刚刚提交的 Oozie workflow job。

- 在 **HA** 集群上提交 **workflow** 作业

1. 登录集群的主 master 节点。

```
ssh root@主master公网Ip
```

可以通过是否能访问 Oozie UI 来判断哪个 master 节点是当前的主 master 节点， Oozie server 服务默认是启动在主 master 节点 `xx.xx.xx.xx:11000/oozie`。

2. 下载 HA 集群的示例代码。

```
[root@emr-header-1 ~]# su oozie
[oozie@emr-header-1 root]$ cd /tmp
[oozie@emr-header-1 tmp]$ wget http://emr-sample-projects.oss-cn-
hangzhou.aliyuncs.com/oozie-examples/oozie-examples-ha.zip
[oozie@emr-header-1 tmp]$ unzip oozie-examples-ha.zip
```

3. 将 Oozie workflow 代码同步到 hdfs 上。

```
[oozie@emr-header-1 tmp]$ hadoop fs -copyFromLocal examples/ /user
/oozie/examples
```

4. 提交 Oozie workflow 样例作业。

```
[oozie@emr-header-1 tmp]$ $OOZIE_HOME/bin/oozie job -config
examples/apps/map-reduce/job.properties -run
```

执行成功之后，会返回一个 jobId，类似：

```
job: 0000000-160627195651086-oozie-oozi-W
```

5. 访问 Oozie UI 页面，可以看到刚刚提交的 Oozie workflow job。

11.3 Presto 使用说明

2.0.0 以上版本支持 *presto*，选择支持 presto 的镜像并勾选 presto 软件即可在 E-MapReduce 中使用 presto。

集群创建后，登录 master，presto 软件被安装在 `/usr/lib/presto-current` 目录，可以 jps 看到 PrestoServer 进程。

presto 服务进程分为 coordinator 和 worker，master 上 (HA 集群为 hostname 以emr-header-1 开头的 master 节点) 启动 coordinator，core 节点启动 worker 进程。服务进程的配置在 `/usr/lib/presto-current/etc` 目录下，其中 coordinator 使用 coordinator-config.properties，worker 使用 worker-config.preperties,其他配置文件公用。web 端口设置为 9090。

presto 服务默认设置了 Hive 的支持，连接集群 hive 的 metastore，可以读取 Hive 的表信息，进行查询。集群预装了 presto cli，可以直接执行以下命令：

```
presto --server localhost: 9090 --catalog hive --schema default --user
 hadoop --execute "show tables"
```

查看 Hive 的表。需要注意同步 Hive 表会有几秒的延时。

11.4 Zeppelin 使用说明

阿里云EMR可以通过Apache Knox访问Zeppelin。

准备工作

1. 在集群安全组中设置安全组规则，打开8080端口。
2. 在Knox中，添加访问用户名和密码，详细参见*Knox* 使用说明，设置Knox用户。用户名和密码仅用于登录Knox的各项服务，与阿里云RAM的用户名无关。

> 注意：
> 设置安全组规则时要针对有限的IP范围。禁止在配置的时候对0.0.0.0/0开放规则。

访问Zeppelin

访问链接查看方式如下：

1. 在EMR管理控制单击集群ID右侧的管理。
2. 页面左侧单击**访问链接与端口**。

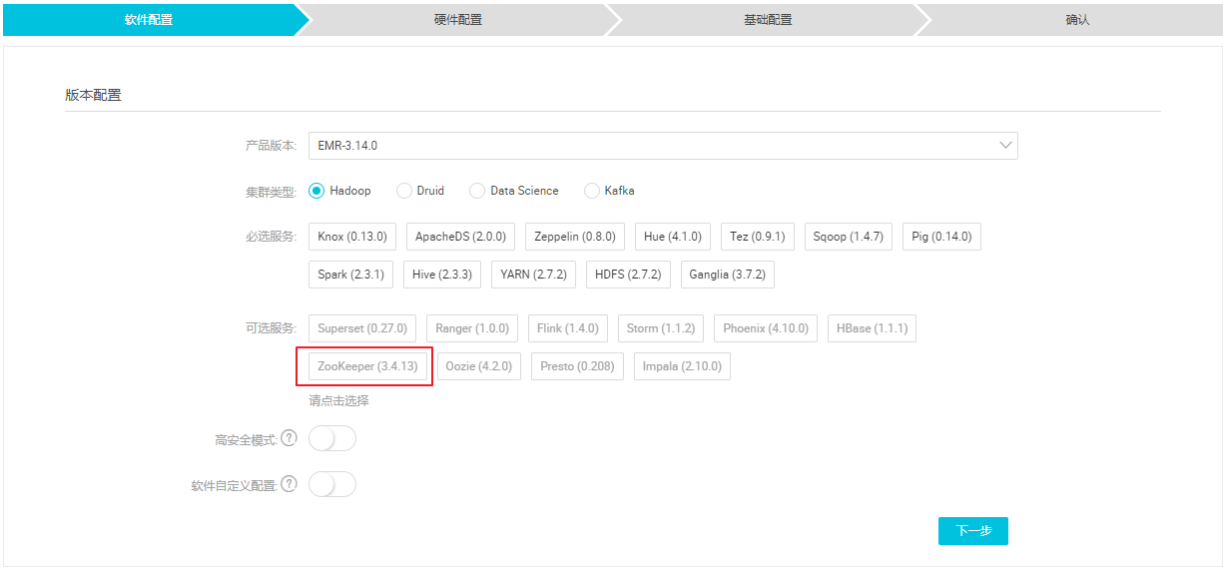
11.5 ZooKeeper 使用说明

目前 E-MapReduce 集群中默认启动了 *ZooKeeper* 服务。

> 说明：
> 目前无论集群内有多少台机器，ZooKeeper 只会有 3 个节点。目前还不支持更多的节点。

创建集群

E-MapReduce 创建集群的软件配置页面，可选择ZooKeeper服务 。



节点信息

集群创建成功，状态空闲后，在集群与服务管理页面选择ZooKeeper，单击部署拓扑可以查看到 ZooKeeper 的节点信息，E-MapReduce 会启动 3 个 ZooKeeper 节点。IP一栏标有 ZooKeeper 节点对应的内网 IP (端口默认为 2181)，即可访问 ZooKeeper 服务。

11.6 Kafka使用说明

11.6.1 Kafka 快速入门

从EMR-3.4.0版本开始将支持Kafka服务。

创建Kafka集群

在E-MapReduce控制台创建集群时，选择集群类型为Kafka，则会创建一个默认只包含Kafka组件的集群，除了基础组件外包括Zookeeper，Kafka和KafkaManager三个组件。每个节点将只部署一个Kafka broker。我们建议您的Kafka集群是一个专用集群，不要和Hadoop相关服务混部在一起。

本地盘Kafka集群

为了更好地降低单位成本以及应对更大的存储需求，E-MapReduce将在EMR-3.5.1版本开始支持基于本地盘（D1类簇机型，详情参考*ECS机型*介绍文档）的Kafka集群。相比较云盘，本地盘Kafka集群有如下特点：

- 实例配备大容量、高吞吐SATA HDD本地盘，单磁盘 190 MB/s 顺序读写性能，单实例最大 5 GB/s 存储吞吐能力。
- 本地存储价格比 SSD 云盘降低 97%。

- 更高网络性能，最大17 Gbit/s实例间网络带宽，满足业务高峰期实例间数据交互需求。

但本地盘机型也有特殊之处：

操作	本地磁盘数据状态	说明
操作系统重启/控制台重启/强制重启	保留	本地磁盘存储卷保留，数据保留
操作系统关机/控制台停止/强制停止	保留	本地磁盘存储卷保留，数据保留
控制台上释放（实例）	擦除	本地磁盘存储卷擦除，数据不保留

注意：

- 当宿主机宕机或者磁盘损坏时，磁盘中的数据将会丢失。
- 请勿在本地磁盘上存储需要长期保存的业务数据，并及时做好数据备份和采用高可用架构。如需长期保存，建议将数据存储在云盘上。

为了能够在本地盘上部署Kafka服务，E-MapReduce默认要求：

1. **`default.replication.factor`** = 3，即要求topic的分区副本数至少为3。如果设置更小副本数，则会增加数据丢失风险。
2. **`min.insync.replicas`** = 2，即要求当producer设定acks为all(-1)时，每次至少写入两个副本才算写入成功。

当出现本地盘损坏时，E-MapReduce会进行：

1. 将坏盘从Broker的配置中剔除，重启Broker，在其他正常可用的本地盘上恢复坏盘丢失的数据。根据坏盘上已经写入的数据量不等，恢复的总时间也不等。
2. 当机器磁盘损坏数目过多（超过20%）是，E-MapReduce将主动进行机器迁移，恢复异常的磁盘。
3. 如果当前机器上可用剩余磁盘空间不足以恢复坏盘丢失数据时，Broker将异常Down掉。这种情况，您可以选择清理一些数据，腾出磁盘空间并重启Broker服务；也可以联系E-MapReduce进行机器迁移，恢复异常的磁盘。

参数说明

您可以在E-MapReduce的集群配置管理中查看Kafka的软件配置，当前主要有：

配置项	说明
zookeeper.connect	Kafka集群的Zookeeper连接地址
kafka.heap.opts	Kafka broker的堆内存大小
num.io.threads	Kafka broker的IO线程数，默认为机器CPU核数目的2倍
num.network.threads	Kafka broker的网络线程数，默认为机器的CPU核数目

11.6.2 Kafka 跨集群访问

通常，我们会单独部署一个Kafka集群来提供服务，所以经常需要跨集群访问Kafka服务。

Kafka跨集群访问说明

跨集群访问Kafka场景分为两种：

- 阿里云内网环境中访问E-MapReduce Kafka集群。
- 公网环境访问E-MapReduce Kafka集群。

针对不同EMR主版本，我们提供了不同的解决方案。

EMR-3.11.x及之后版本

- 阿里云内网中访问Kafka

  直接使用Kafka集群节点的内网IP访问即可，内网访问Kafka请使用9092端口。

  访问Kafka前请保证网络是互通的：

  - 经典网络访问VPC，请参考经典网络与VPC互访。
  - VPC访问VPC，请参考配置VPC到VPC连接。
- 公网环境访问Kafka

  Kafka集群的Core节点默认无法访问公网，所以如果您需要公网环境访问Kafka集群，可以参考以下步骤完成：

  1. 使Kafka集群和公网主机网络互通。
     - Kafka集群部署在VPC网络环境，有两种方式：
       - 部署高速通道打通内网和公网网络，参考高速通道文档。
       - 集群Core节点挂载弹性公网IP。以下步骤使用挂载EIP方式。
     - Kafka集群部署在经典网络，有两种方式：

- 如果您创建的是按量集群，目前只能通过ECS API来实现，请参考*API*文档。
- 如果您创建的是包年包月集群，您可以直接在ECS控制台对相应机器变配开启分配公网IP。

2. 在*VPC*控制台申请EIP，根据您Kafka集群Core节点个数购买相应的EIP。
3. 配置kafka集群安全组规则来限制公网可访问Kafka集群的IP等，目的是提高Kafka集群暴露在公网中的安全性。您可以在EMR控制台查看到集群所属的安全组，根据安全组ID去查找并配置安全组规则。文档地址
4. 在EMR控制台的集群管理页面，单击指定集群后的管理，在页面左侧选择集群基础信息页签， 然后单击页面右上角的同步主机信息按钮。
5. 重启集群Kafka服务。
6. 公网环境使用Kafka集群节点的EIP访问即可，内网访问Kafka请使用9093端口。

EMR-3.11.x以前版本

- 阿里云内网中访问Kafka

  我们需要在主机上配置Kafka集群节点的host信息。注意，这里我们需要在client端的主机上配置Kafka集群节点的长域名，否则会出现访问不到Kafka服务的问题。示例如下：

```
/etc/hosts
# kafka cluster
10.0.1.23 emr-header-1.cluster-48742
10.0.1.24 emr-worker-1.cluster-48742
10.0.1.25 emr-worker-2.cluster-48742
10.0.1.26 emr-worker-3.cluster-48742
```

- 公网环境访问Kafka

  Kafka集群的Core节点默认无法访问公网，所以如果您需要在公网环境访问Kafka集群，可以参考以下步骤完成：

  1. 使Kafka集群和公网主机网络互通。
     - Kafka集群部署在VPC网络环境，有两种方式：
       - 部署高速通道打通内网和公网网络，参考高速通道文档。
       - 集群Core节点挂载弹性公网IP。以下步骤使用挂载EIP方式。
     - Kafka集群部署在经典网络，有两种方式：
       - 如果您创建的是按量集群，目前只能通过ECS API来实现，请参考*API*文档。

- 如果您创建的是包年包月集群，您可以直接在ECS控制台对相应机器变配开启分配公网IP。

2. 在VPC控制台申请EIP，根据您Kafka集群Core节点个数购买相应的EIP。
3. 配置kafka集群安全组规则来限制公网可访问Kafka集群的IP等，目的是提高Kafka集群暴露在公网中的安全性。您可以在EMR控制台查看到集群所属的安全组，根据安全组ID去查找并配置安全组规则。文档地址
4. 修改Kafka集群的软件配置 **`listeners.address.principal`** 为 `HOST`，并重启Kafka集群。
5. 参考步骤5，配置本地客户端主机的 *`hosts`* 文件。

11.6.3 Kafka Ranger使用说明

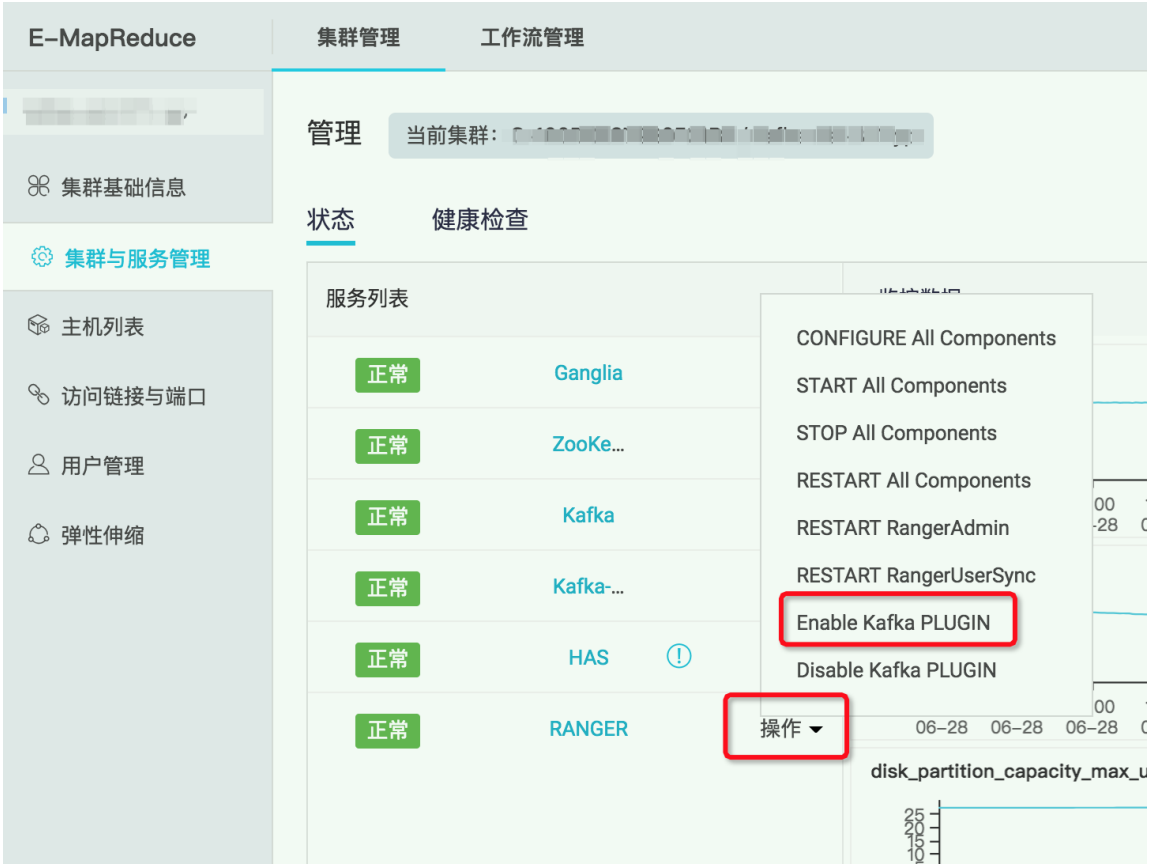
从 EMR-3.12.0 版本开始，E-MapReduce Kafka 支持用Ranger进行权限配置。

Kafka集成Ranger

前面简介中介绍了E-MapReduce中创建启动Ranger服务的集群，以及一些准备工作，本节介绍Kafka集成Ranger的一些步骤流程。

- Enable Kakfa Plugin

1. 在集群与服务管理页面的 **Ranger** 服务下，单击右侧的操作下拉菜单，选择 **Enable Kafka PLUGIN**。



2. 单击右上角的**查看操作历史**查看任务进度，等待任务完成100%。

- 重启Kafka broker

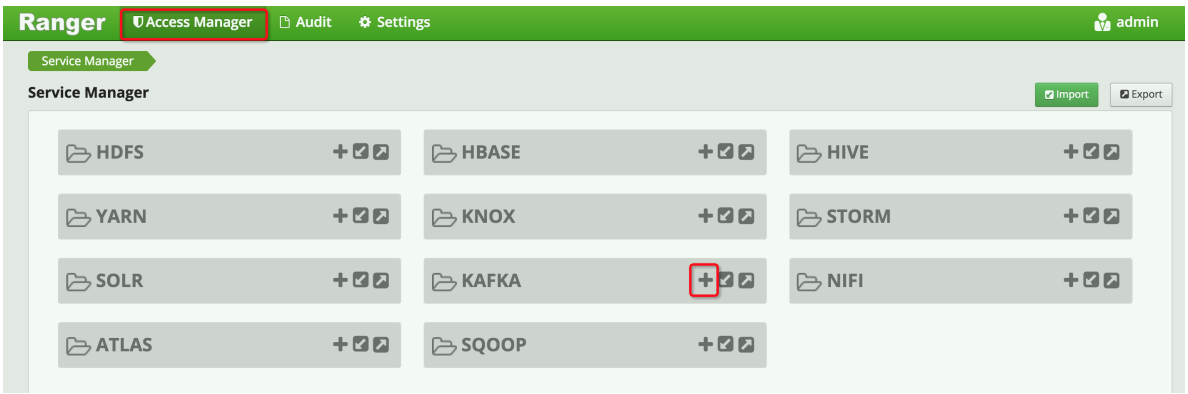
  上述任务完成后，需要重启broker才能生效。

  1. 在**集群与服务管理**页面的服务列表中单击**Kafka**进入Kafka服务配置页面。
  2. 点击右上角 **操作** 中的 **RESTART Kafka Broker**。
  3. 点击右上角**查看操作历史**查看任务进度，等待重启任务完成。

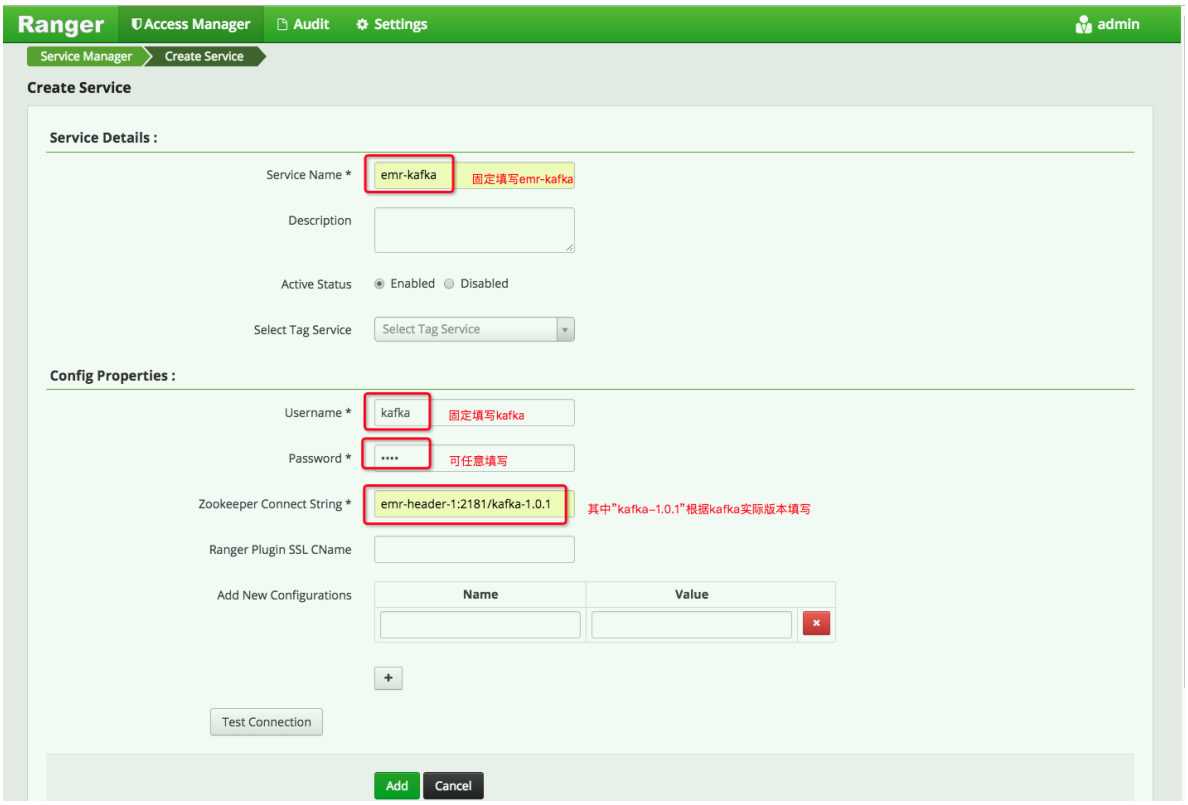
- Ranger UI页面添加Kafka Service

  参见*Ranger简介*的介绍进入Ranger UI页面。

  在 **Ranger** 的UI页面添加Kafka Service：



  配置Kafka Service：

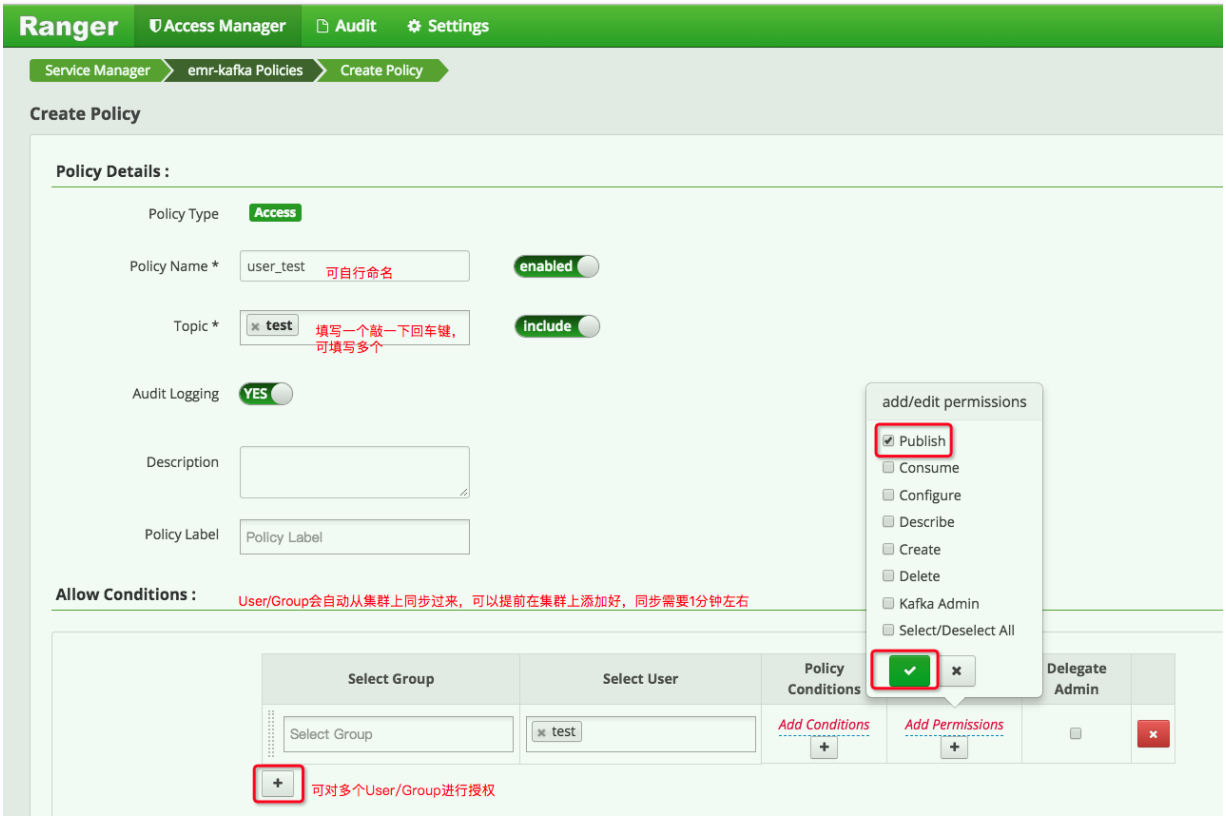


权限配置示例

上面一节中已经将Ranger集成到Kafka，现在可以进行相关的权限设置。

> 注意：
> 标准集群中，在添加了 Kafka Service 后，ranger 会默认生成规则 all - topic，不作任何权限限制（即允许所有用户进行所有操作），此时ranger无法通过用户进行权限识别。

以test用户为例，添加Publish权限：



按照上述步骤设置添加一个Policy后，就实现了对 **test** 的授权，然后用户 **test** 就可以对 **test** 的topic进行写入操作。

> 说明：
> Policy添加后需要1分钟左右才会生效。

11.6.4 Kafka SSL使用说明

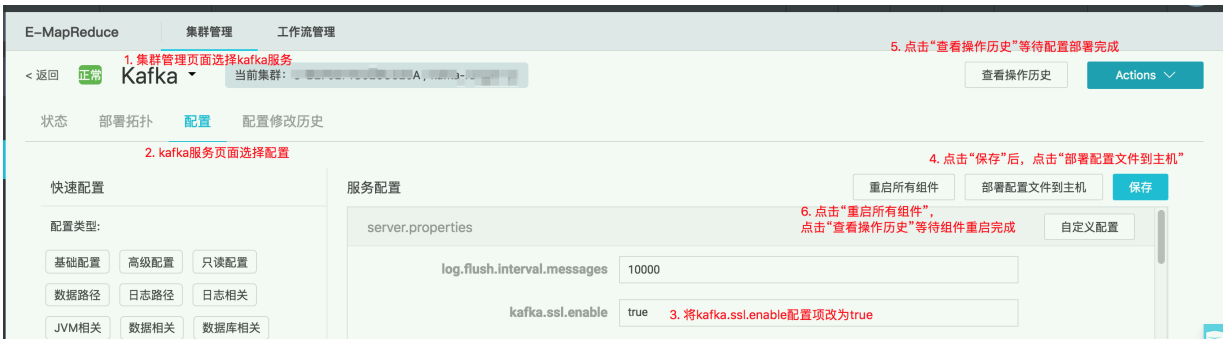
从 EMR-3.12.0 版本开始，E-MapReduce Kafka支持SSL功能。

创建集群

具体的创建集群操作，请参见创建集群。

开启SSL服务

默认Kafka集群没有开启SSL功能，您可以在Kafka服务的配置页面开启SSL。



如图，修改配置项**kafka.ssl.enable**为true，部署配置并重启组件。

客户端访问Kafka

客户端通过SSL访问Kafka时需要设置 **security.protocol**、**truststore** 和 **keystore** 的相关配置。以非安全集群为例，如果是在kafka集群运行作业，可以配置如下：

```
security.protocol=SSL
ssl.truststore.location=/etc/ecm/kafka-conf/truststore
ssl.truststore.password=${password}
ssl.keystore.location=/etc/ecm/kafka-conf/keystore
ssl.keystore.password=${password}
```

如果是在Kafka集群以外的环境运行作业，可将 Kafka 集群中的 truststore 和 keystore 文件（位于集群任意一个节点的*/etc/ecm/kafka-conf/*目录中）拷贝至运行环境作相应配置。

以Kafka自带的producer和consumer程序，在 Kafka 集群运行为例：

1. 创建配置文件 *ssl.properties*， 添加配置项。

   ```
   security.protocol=SSL
    ssl.truststore.location=/etc/ecm/kafka-conf/truststore
    ssl.truststore.password=${password}
    ssl.keystore.location=/etc/ecm/kafka-conf/keystore
    ssl.keystore.password=${password}
   ```

2. 创建topic。

   ```
   kafka-topics.sh --zookeeper emr-header-1:2181/kafka-1.0.1 --
   replication-factor 2 --
   ```

```
partitions 100 --topic test --create
```

3. 使用SSL配置文件产生数据。

```
kafka-producer-perf-test.sh --topic test --num-records 123456 --
throughput 10000 --record-size 1024 --producer-props bootstrap.
servers=emr-worker-1:9092 --producer.config ssl.properties
```

4. 使用SSL配置文件消费数据。

```
kafka-consumer-perf-test.sh --broker-list emr-worker-1:9092 --
messages 100000000 --topic test --consumer.config ssl.properties
```

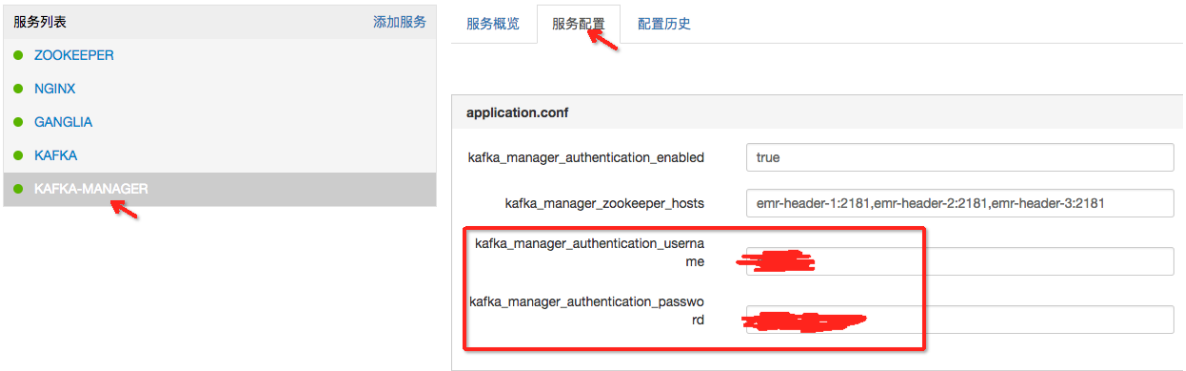
11.6.5 Kafka Manager 使用说明

从EMR-3.4.0版本开始将支持Kafka Manager服务进行Kafka集群管理。

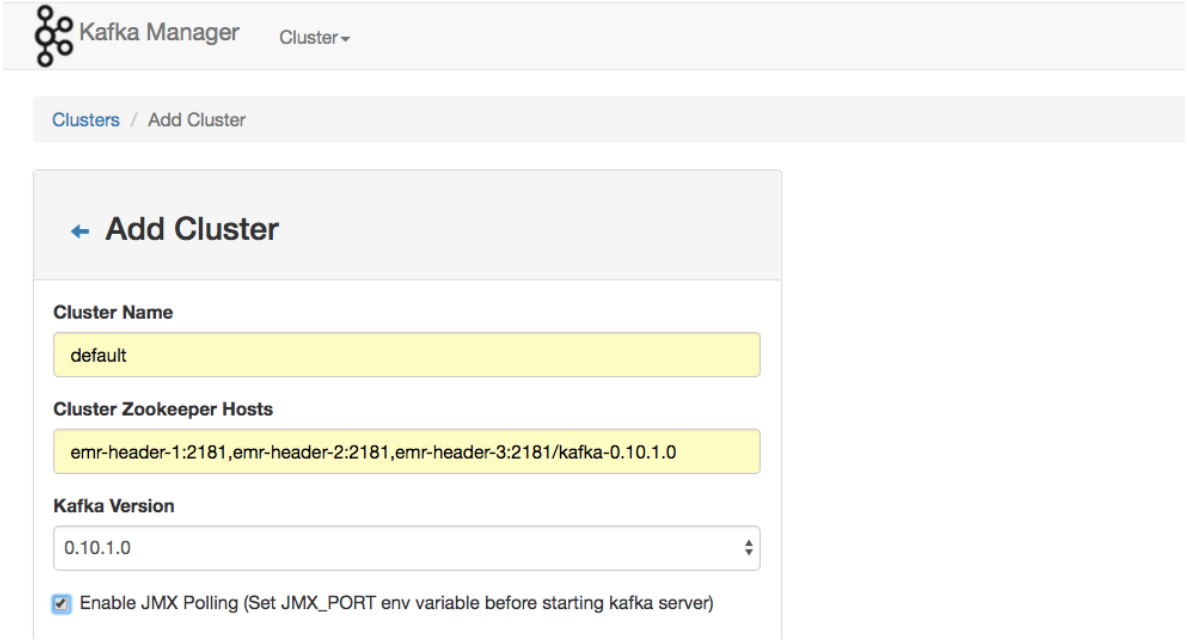
操作步骤

> 注意：
> 创建Kafka集群时将默认安装Kafka Manager软件服务，并开启Kafka Manager的认证功能。我们强烈建议您首次使用Kafka Manager时修改默认密码，且使用SSH隧道方式来访问。不建议您公网暴露8085端口，否则需要做好IP白名单保护，避免数据泄漏。

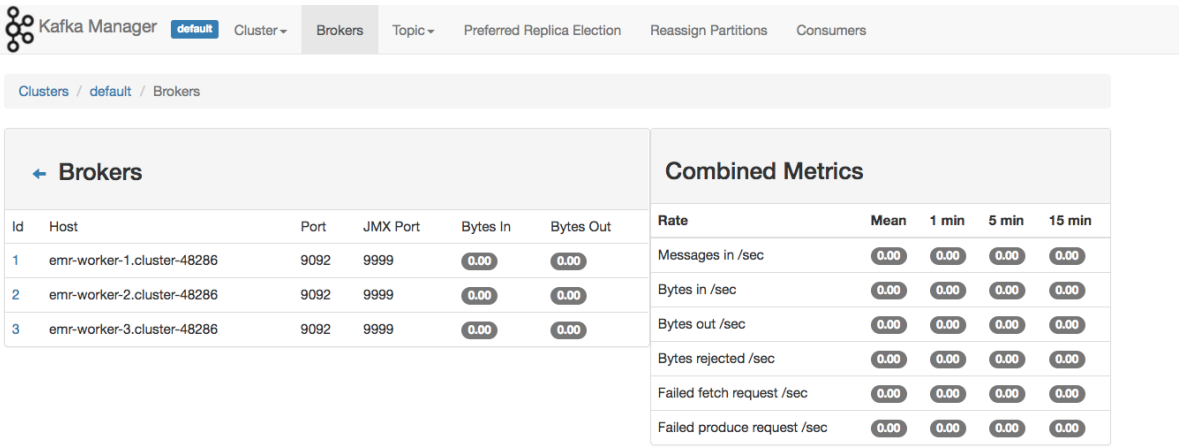
- 建议使用SSH隧道方式访问Web界面，参考*SSH* 登录集群。
- 访问 《*http://localhost:8085*》。
- 输入用户名密码，请参考Kafka Manager的配置信息。



- 添加一个创建好的Kafka集群，需要注意配置正确Kafka集群的Zookeeper地址，不清楚的可以看Kakfa的配置信息。选择对应的Kafka版本，另外建议打开JMX功能。



- 创建好之后即可使用一些常见的Kakfa功能。



11.6.6 Kafka 常见问题

本文介绍Kafka 常见问题的一些问题以及解决方法。

- `Error while executing topic command : Replication factor: 1 larger than available brokers: 0.`

  常见原因：

  — Kafka服务异常，集群Broker进程都退出了，需要结合日志排查问题。

  — Kafka服务的Zookeeper地址使用错误，请注意查看并使用集群配置管理中Kafka组件的Zookeeper连接地址。

- `java.net.BindException: Address already in use (Bind failed)`

当您使用kafka命令行工具时，有时会碰到 `java.net.BindException: Address already in use (Bind failed)` 异常，这一般由JMX端口被占用导致，您可以在命令行前手动指定一个JMX端口即可。例如：

```
JMX_PORT=10101 kafka-topics --zookeeper emr-header-1:2181/kafka-1.0.0 --list
```

11.7 Druid使用说明

11.7.1 Druid简介

Druid是Metamarkets公司（一家为在线媒体或广告公司提供数据分析服务的公司）推出的一个分布式内存实时分析系统，用于解决如何在大规模数据集下进行快速的、交互式的查询和分析。

基本特点

Druid具有以下特点：

- 亚秒级OLAP查询，包括多维过滤、ad-hoc的属性分组、快速聚合数据等等。
- 实时的数据消费，真正做到数据摄入实时、查询结果实时。
- 高效的多租户能力，最高可以做到几千用户同时在线查询。
- 扩展性强，支持PB级数据、千亿级事件快速处理，支持每秒数千查询并发。
- 极高的高可用保障，支持滚动升级。

应用场景

实时数据分析是Druid最典型的使用场景。该场景涵盖的面很广，例如：

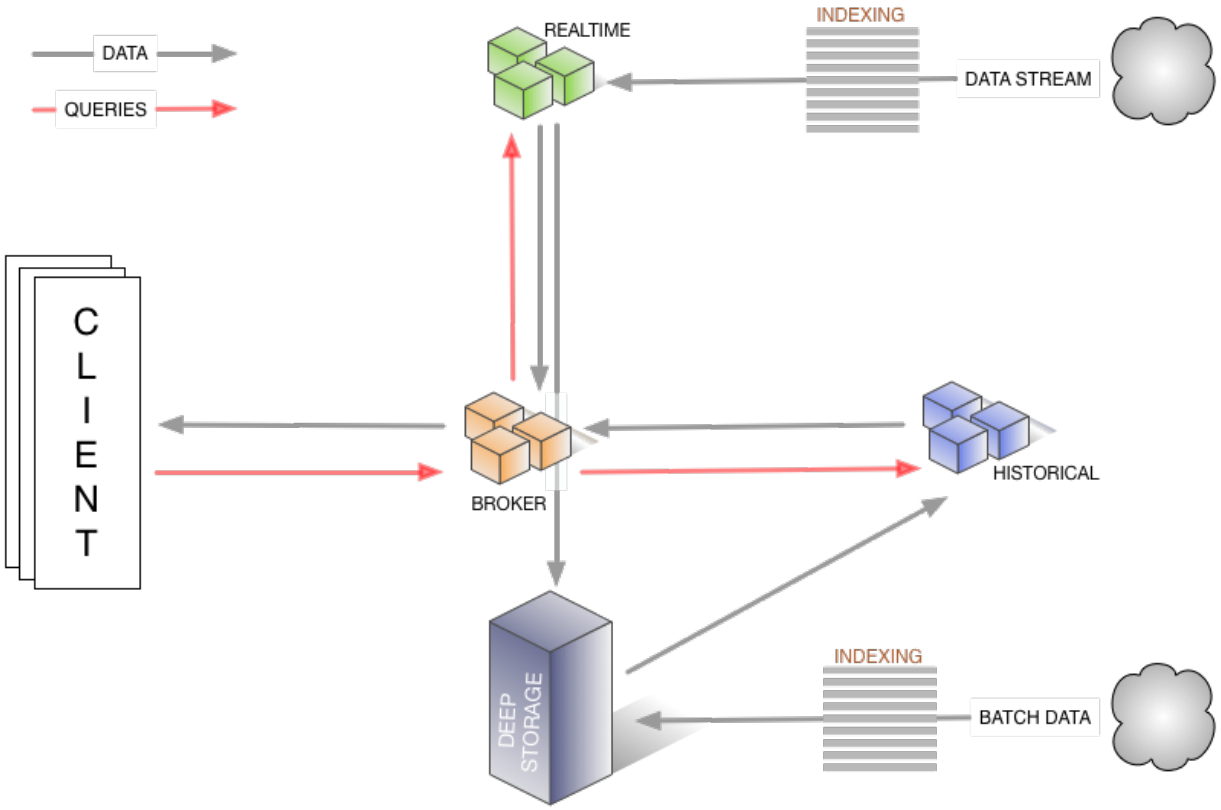
- 实时指标监控
- 推荐模型
- 广告平台
- 搜索模型

这些场景的特点都是拥有大量的数据，且对数据查询的时延要求非常高。在实时指标监控中，系统问题需要在出现的一刻被检测到并被及时给出报警。在推荐模型中，用户行为数据需要实时采集，并及时反馈到推荐系统中。用户几次点击之后系统就能够识别其搜索意图，并在之后的搜索中推荐更合理的结果。

Druid架构

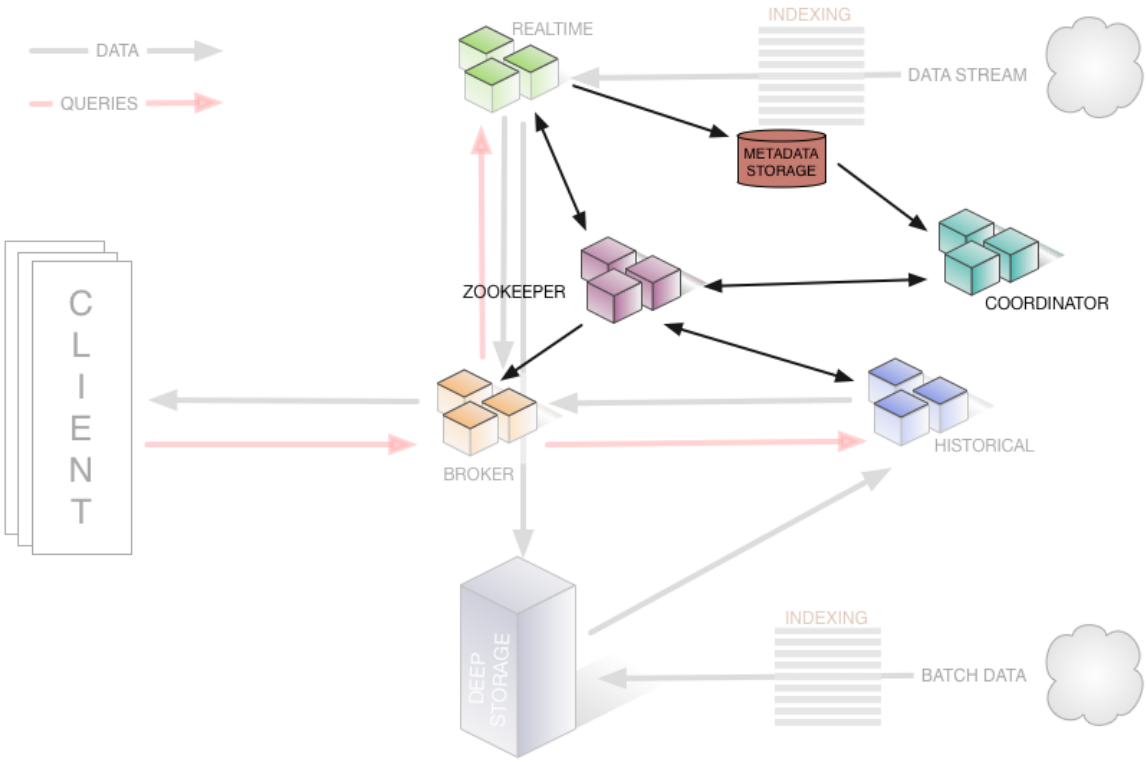
Druid拥有优秀的架构设计，多个组件协同工作，共同完成数据从摄取到索引、存储、查询等一系列流程。

下图是Druid工作层（数据索引以及查询）包含的组件。



- Realtime 组件负责数据的实时摄入。
- Broker 阶段负责查询任务的分发以及查询结果的汇总，并将结果返回给用户。
- Historical节点负责索引后的历史数据的存储，数据存储在deep storage。deep storage可以是本地，也可以是HDFS等分布式文件系统。
- Indexing service 包含两个组件（图中未画出）。
  - Overlord 组件负责索引任务的管理、分发。
  - MiddleManager 负责索引任务的具体执行。

下图是 Druid segments（Druid 索引文件）管理层所涉及的组件。



- Zookeeper 负责存储集群的状态以及作为服务发现组件，例如集群的拓扑信息，overlord leader 的选举，indexing task 的管理等等。
- Coordinator 负责 segments 的管理，如 segments 下载、删除以及如何在 historical 之间做均衡等等。
- Metadata storage 负责存储 segments 的元信息，以及管理集群各种各样的持久化或临时性数据，比如配置信息、审计信息等等。

产品优势

E-MapReduce Druid 基于开源 Druid 做了大量的改进，包括与E-MapReduce、阿里云周边生态的集成、方便的监控与运维支持、易用的产品接口等等，真正做到了即买即用和7*24免运维。

EMR Druid 目前支持的特性如下所示：

- 支持以OSS作为deep storage
- 支持将OSS文件作为批量索引的数据来源
- 支持将元数据存储到 RDS
- 集成了Superset工具
- 方便地扩容、缩容（缩容针对 task 节点）

- 丰富的监控指标和告警规则
- 坏节点迁移
- 支持高安全
- 支持HA

11.7.2 快速入门

E-MapReduce从EMR-3.11.0版本开始支持Druid作为E-MapReduce的一个集群类型。

将Druid作为一种单独的集群类型（而不是在Hadoop集群中增加Druid组件）主要基于几个方面的考虑：

- Druid可以完全脱离Hadoop使用。
- 大数据量下Druid对内存要求比较高，尤其是Broker节点和Historical节点。Druid本身不受YARN管控，在多服务运行时容易发生资源争抢。
- Hadoop作为基础设施，其规模可以比较大，而Druid集群可以比较小，两者配合起来工作灵活性更高。

创建Druid集群

在创建集群时选择Druid集群类型即可。您在创建Druid集群时可以勾选HDFS和YARN，Druid集群自带的HDFS和YARN仅供测试使用，原因如本文档开头所述。对于生产环境，我们强烈建议您采用专门的Hadoop集群。

配置集群

- 配置使用HDFS作为Druid的deep storage

  对于独立的Druid集群，您可能需要将您的索引数据存放在另外一个Hadoop集群的HDFS之上。为此，您需要首先设置一下两个集群的连通性（见下文*与Hadoop集群交互*一节），再在Druid的配置页面配置如下两个选项并重启服务即可（配置项位于配置页面的 common.runtime）。

  - druid.storage.type: hdfs
  - druid.storage.storageDirectory:（请注意这里HDFS 目录最好写完整目录，比如 hdfs://emr-header-1.cluster-xxxxxxxx:9000/druid/segments）

  > 说明：
  > 如果Hadoop集群为HA集群，emr-header-1.cluster-xxxxx:9000 需要改成 emr-cluster，或者把端口9000改成8020，下同。

- 配置使用OSS作为Druid的deep storage

  EMR Druid支持以OSS作为deep storage，借助于EMR的免AccessKey能力，Druid不用做AccessKey配置即可访问OSS。由于OSS的访问能力是借助于HDFS的OSS功能实现的，因此在配置时，druid.storage.type 需要仍然配置为HDFS。

  - druid.storage.type: hdfs
  - druid.storage.storageDirectory: （如 oss://emr-druid-cn-hangzhou/segments）

  由于OSS访问借助了HDFS，因此您需要选择以下两种方案之一：

  - 建集群的时候选择安装HDFS，系统自动配好（安装好HDFS您可以不使用它，关闭它，或者仅作为测试用途）。
  - 在Druid的配置目录*/etc/ecm/druid-conf/druid/_common/*下新建*hdfs-site.xml*，内容如下，然后将该文件拷贝至所有节点的相同目录下。

    ```
    <?xml version="1.0"?>
      <configuration>
        <property>
          <name>fs.oss.impl</name>
          <value>com.aliyun.fs.oss.nat.NativeOssFileSystem</value>
        </property>
        <property>
          <name>fs.oss.buffer.dirs</name>
          <value>file:///mnt/disk1/data,...</value>
        </property>
        <property>
          <name>fs.oss.impl.disable.cache</name>
          <value>true</value>
        </property>
      </configuration>
    ```

    其中**fs.oss.buffer.dirs**可以设置多个路径。

- 配置使用RDS作为Druid的元数据存储

  默认情况下Druid利用header-1节点上的本地MySQL数据库作为元数据存储。您也可以配置使用阿里云RDS作为元数据存储。

  下面以RDS MySQL版作为例演示配置。在具体配置之前，请先确保：

  - RDS MySQL实例已经被创建。
  - 为Druid访问RDS MySQL创建了单独的账户（不推荐使用root），假设账户名为 druid，密码为druidpw。
  - 为Druid元数据创建单独的MySQL数据库，假设数据库名为druiddb。
  - 确保账户druid有权限访问druiddb。

在EMR管理控制台，进入Druid集群，单击Druid组件，选择**配置**选项卡，找到 *common.runtime* 配置文件。单击**自定义配置**，添加如下三个配置项：

- druid.metadata.storage.connector.connectURI，值为 jdbc:mysql://rm-xxxxx.mysql.rds.aliyuncs.com:3306/druiddb。
- druid.metadata.storage.connector.user，值为druid。
- druid.metadata.storage.connector.password，值为druidpw。

依次点击右上角的**保存**、**部署配置文件到主机**、**重启所有组件**，配置即可生效。

登录RDS控制台查看druiddb创建表的情况，如果正常，您将会看到一些druid自动创建的表。

- 组件内存配置

Druid组件内存设置主要包括两方面：堆内存（通过jvm.config 配置）和direct内存（通过jvm.config 和runtime.properteis配置）。在创建集群时，EMR会自动生成一套配置，不过在某些情况下您仍然可能需要自己调整内存配置。

要调整组件内存配置，您可以通过EMR控制台进入到集群组件，在页面上进行操作。

说明：

对于direct内存，调整时请确保：

```
-XX:MaxDirectMemorySize >= druid.processing.buffer.sizeBytes * (
druid.processing.numMergeBuffers + druid.processing.numThreads + 1)
```

批量索引

- 与 Hadoop 集群交互

您在创建Druid集群时如果勾选了HDFS和YARN（自带 Hadoop 集群），那么系统将会自动为您配置好与HDFS和YARN的交互，您无需做额外操作。下面的介绍是配置独立Druid集群与独立Hadoop集群之间交互，这里假设Druid集群cluster id为1234，Hadoop 集群cluster id 为5678。另外请仔细阅读并严格按照指导进行操作，如果操作不当，集群可能就不会按照预期工作。

对于与非安全独立Hadoop集群交互，请按照如下操作进行：

1. 确保集群间能够通信（两个集群在一个安全组下，或两个集群在不同安全组，但两个安全组之间配置了访问规则）。
2. 将Hadoop集群*/etc/ecm/hadoop-conf*的 core-site.xml、hdfs-site.xml、yarn-site.xml、mapred-site.xml 放在Druid集群每个节点的 */etc/ecm/duird-conf/druid/_common* 目

录下（如果创建集群时选了自带Hadoop，在该目录下会有几个软链接指向自带 Hadoop 的配置，请先移除这些软链接）。

3. 将Hadoop集群的hosts写入到Druid集群的hosts列表中，注意Hadoop集群的 hostname 应采用长名形式，如 emr-header-1.cluster-xxxxxxxx，且最好将Hadoop的hosts放在本集群hosts之后，例如：

```
...
10.157.201.36   emr-as.cn-hangzhou.aliyuncs.com
10.157.64.5     eas.cn-hangzhou.emr.aliyuncs.com
192.168.142.255 emr-worker-1.cluster-1234 emr-worker-1 emr-header-
2.cluster-1234 emr-header-2 iZbp1h9g7boqo9x23qbifiZ
192.168.143.0   emr-worker-2.cluster-1234 emr-worker-2 emr-header-
3.cluster-1234 emr-header-3 iZbp1eaa5819tkjx55yr9xZ
192.168.142.254 emr-header-1.cluster-1234 emr-header-1 iZbp1e3zwu
vnmakmsjer2uZ
--以下为hadoop集群的hosts信息
192.168.143.6   emr-worker-1.cluster-5678 emr-worker-1 emr-header-
2.cluster-5678 emr-header-2 iZbp195rj7zvx8qar4f6b0Z
192.168.143.7   emr-worker-2.cluster-5678 emr-worker-2 emr-header-
3.cluster-5678 emr-header-3 iZbp15vy2rsxoegki4qhdpZ
192.168.143.5   emr-header-1.cluster-5678 emr-header-1 iZbp10tx4e
gw3wfnh5oii1Z
```

对于安全Hadoop集群，请按如下操作进行：

1. 确保集群间能够通信（两个集群在一个安全组下，或两个集群在不同安全组，但两个安全组之间配置了访问规则）。
2. 将Hadoop集群 */etc/ecm/hadoop-conf* 的 core-site.xml、hdfs-site.xml、yarn-site.xml、mapred-site.xml放在Druid集群每个节点的 */etc/ecm/duird-conf/druid/_common* 目录下（如果创建集群时选了自带Hadoop，在该目录下会有几个软链接指向自带Hadoop的配置，请先移除这些软链接），并修改 core-site.xml 中 **`hadoop.security.authentication.use.has`** 为 `false`（这是一条客户端配置，目的是让用户能够使用AccessKey认证，如果用Kerberos认证方式，需disable该配置）。
3. 将Hadoop集群的hosts写入到Druid集群每个节点的hosts列表中，注意Hadoop集群的hostname应采用长名形式，如 emr-header-1.cluster-xxxxxxxx，且最好将hadoop的 hosts放在本集群hosts之后。
4. 设置两个集群间的Kerberos跨域互信（详情参考跨域互信）。
5. 在Hadoop集群的所有节点下都创建一个本地Druid账户（useradd -m -g hadoop druid），或者设置 druid.auth.authenticator.kerberos.authToLocal（具体预发规则参考这里）创建Kerberos账户到本地账户的映射规则。推荐第一种做法，操作简便不易出错。

说明：

默认在安全Hadoop集群中，所有Hadoop命令必须运行在一个本地的账户中，该本地账户需要与principal的name部分同名。YARN也支持将一个principal映射至本地一个账户，即上文第二种做法。

6. 重启Druid服务。

- **使用Hadoop对批量数据创建索引**

Druid自带了一个名为wikiticker的例子，位于*${DRUID_HOME}/quickstart*下面（*${DRUID_HOME}*默认为 /usr/lib/druid-current）。wikiticker文件（wikiticker-2015-09-12-sampled.json.gz）的每一行是一条记录，每条记录是个json对象。其格式如下所示：

```json
{
    "time": "2015-09-12T00:46:58.771Z",
    "channel": "#en.wikipedia",
    "cityName": null,
    "comment": "added project",
    "countryIsoCode": null,
    "countryName": null,
    "isAnonymous": false,
    "isMinor": false,
    "isNew": false,
    "isRobot": false,
    "isUnpatrolled": false,
    "metroCode": null,
    "namespace": "Talk",
    "page": "Talk:Oswald Tilghman",
    "regionIsoCode": null,
    "regionName": null,
    "user": "GELongstreet",
    "delta": 36,
    "added": 36,
    "deleted": 0
}
```

使用Hadoop对批量数据创建索引，请按照如下步骤进行操作：

1. 将该压缩文件解压，并放置于HDFS的一个目录下（如 *hdfs://emr-header-1.cluster-5678:9000/druid*）。 在Hadoop集群上执行如下命令。

```
### 如果是在独立Hadoop集群上进行操作，需要做好两个集群互信之后需要拷贝一个druid.keytab到Hadoop集群再kinit。
 kinit -kt /etc/ecm/druid-conf/druid.keytab druid
 ###
 hdfs dfs -mkdir hdfs://emr-header-1.cluster-5678:9000/druid
```

```
 hdfs dfs -put ${DRUID_HOME}/quickstart/wikiticker-2015-09-16-
sampled.json hdfs://emr-header-1.cluster-5678:9000/druid
```

说明：

- 对于安全集群执行HDFS命令前先修改 */etc/ecm/hadoop-conf/core-site.xml* 中 **hadoop.security.authentication.use.has** 为 false。
- 请确保已经在Hadoop集群每个节点上创建名为druid的Linux账户。

2. 修改Druid集群 *${DRUID_HOME}/quickstart/wikiticker-index.json*，如下所示：

```
{
     "type" : "index_hadoop",
     "spec" : {
         "ioConfig" : {
             "type" : "hadoop",
             "inputSpec" : {
                 "type" : "static",
                 "paths" : "hdfs://emr-header-1.cluster-5678:9000/
druid/wikiticker-2015-09-16-sampled.json"
             }
         },
         "dataSchema" : {
             "dataSource" : "wikiticker",
             "granularitySpec" : {
                 "type" : "uniform",
                 "segmentGranularity" : "day",
                 "queryGranularity" : "none",
                 "intervals" : ["2015-09-12/2015-09-13"]
             },
             "parser" : {
                 "type" : "hadoopyString",
                 "parseSpec" : {
                     "format" : "json",
                     "dimensionsSpec" : {
                         "dimensions" : [
                             "channel",
                             "cityName",
                             "comment",
                             "countryIsoCode",
                             "countryName",
                             "isAnonymous",
                             "isMinor",
                             "isNew",
                             "isRobot",
                             "isUnpatrolled",
                             "metroCode",
                             "namespace",
                             "page",
                             "regionIsoCode",
                             "regionName",
                             "user"
                         ]
                     },
                     "timestampSpec" : {
                         "format" : "auto",
                         "column" : "time"
                     }
```

```
                }
            },
            "metricsSpec" : [
                {
                    "name" : "count",
                    "type" : "count"
                },
                {
                    "name" : "added",
                    "type" : "longSum",
                    "fieldName" : "added"
                },
                {
                    "name" : "deleted",
                    "type" : "longSum",
                    "fieldName" : "deleted"
                },
                {
                    "name" : "delta",
                    "type" : "longSum",
                    "fieldName" : "delta"
                },
                {
                    "name" : "user_unique",
                    "type" : "hyperUnique",
                    "fieldName" : "user"
                }
            ]
        },
        "tuningConfig" : {
            "type" : "hadoop",
            "partitionsSpec" : {
                "type" : "hashed",
                "targetPartitionSize" : 5000000
            },
            "jobProperties" : {
                "mapreduce.job.classloader": "true"
            }
        }
    },
    "hadoopDependencyCoordinates": ["org.apache.hadoop:hadoop-
client:2.7.2"]
 }
```

说明：

- **spec.ioConfig.type** 设置为 hadoop。
- **spec.ioConfig.inputSpec.paths** 为输入文件路径。
- **tuningConfig.type** 为 hadoop。
- **tuningConfig.jobProperties** 设置了 mapreduce job 的 classloader。
- **hadoopDependencyCoordinates** 制定了 hadoop client 的版本。

3. 在Druid集群上运行批量索引命令。

```
cd ${DRUID_HOME}
```

```
 curl --negotiate -u:druid -b ~/cookies -c ~/cookies -XPOST -H '
Content-Type:application/json' -d @quickstart/wikiticker-index.
json http://emr-header-1.cluster-1234:18090/druid/indexer/v1/task
```

其中 `- -negotiate`、`-u`、`-b`、`-c` 等选项是针对安全Druid集群的。Overlord的端口默认为18090。

4. 查看作业运行情况。

   在浏览器访问 http://emr-header-1.cluster-1234:18090/console.html 查看作业运行情况。为了能正常访问该页面，您可能需要事先开启一个SSH隧道（详见*SSH* 登录集群“查看Hadoop、Spark、Ganglia 等系统的 WebUI”小节），并启动一个代理Chrome。如果Druid集群开启了高安全，您还得配置您的浏览器支持Kerberos的认证流程，可参考这里。

5. 根据Druid语法查询数据。

   Druid有自己的查询语法。您需要准备一个描述您如何查询的json格式的查询文件，如下所示为对wikiticker数据的一个top N查询（*`${DRUID_HOME}/quickstart/wikiticker-top-pages.json`*）：

```
{
     "queryType" : "topN",
     "dataSource" : "wikiticker",
     "intervals" : ["2015-09-12/2015-09-13"],
     "granularity" : "all",
     "dimension" : "page",
     "metric" : "edits",
     "threshold" : 25,
     "aggregations" : [
         {
             "type" : "longSum",
             "name" : "edits",
             "fieldName" : "count"
         }
     ]
 }
```

   在命令行界面运行下面的命令即可看到查询结果：

```
cd ${DRUID_HOME}
 curl --negotiate -u:druid -b ~/cookies -c ~/cookies -XPOST -H '
Content-Type:application/json' -d @quickstart/wikiticker-top-pages
.json 'http://emr-header-1.cluster-1234:18082/druid/v2/?pretty'
```

   其中 `- -negotiate`、`-u`、`-b`、`-c` 等选项是针对安全Druid集群的。如果一切正常，您将能看到具体地查询结果。

- 实时索引

我们推荐使用*Tranquility 客户端*向Druid发送实时数据。Tranquility支持Kafka、Flink、Storm、Spark Streaming等多种方式向Druid发送数据。对于Kafka方式，可参考*Tranquility*中Druid使用Tranquility Kafka一节。更多的使用方式以及SDK的使用方法，可参考*Tranquility帮助文档*。

对于Kafka方式，您还可以使用kafka-indexing-service扩展，详见*Kafka Indexing Service*中“使用Druid Kafka Indexing Service实时消费Kafka数据”一节。

- 索引失败问题分析思路

  当发现索引失败时，一般遵循如下排错思路：

  - 对于批量索引

    1. 如果curl直接返回错误，或者不返回，检查一下输入文件格式。或者curl加上 **-v** 参数，观察REST API的返回情况。
    2. 在Overlord页面观察作业执行情况，如果失败，查看页面上的logs。
    3. 在很多情况下并没有生成logs，如果是Hadoop作业，打开YARN页面查看是否有索引作业生成，并查看作业执行log。
    4. 如果上述情况都没有定位到错误，需要登录到Druid集群，查看overlord的执行日志（位于*/mnt/disk1/log/druid/overlord—emr-header-1.cluster-xxxx.log*），如果是HA 集群，查看您提交作业的那个Overlord。
    5. 如果作业已经被提交到Middlemanager，但是从Middlemanager返回了失败，则需要从Overlord中查看作业提交到了那个worker，并登录到相应的worker，查看Middlemanager的日志（位于*/mnt/disk1/log/druid/middleManager-emr-header-1.cluster-xxxx.log*）。

  - 对于Tranquility实时索引

    查看Tranquility log，查看消息是否被接收到了或者是否被丢弃（drop）掉了。

    其余的排查步骤同批量索引的步骤2~步骤5。

    错误多数情况为集群配置问题和作业问题。集群配置问题包括：内存参数是否合理、跨集群联通性是否正确、安全集群访问是否通过、principal是否正确等等，作业问题包括作业描述文件格式正确，输入数据是否能够正常被解析，以及一些其他的作业相关的配置（如ioConfig等）。

11.7.3 数据格式描述文件

本小节简要介绍一下索引数据的描述文件——Ingestion Spec文件，更为详细的信息请参考Druid官方文档。

Ingestion Spec（数据格式描述）是Druid对要索引数据的格式以及如何索引该数据格式的一个统一描述，它是一个JSON文件，一般由三部分组成：

```
{
    "dataSchema" : {...},
    "ioConfig" : {...},
    "tuningConfig" : {...}
}
```

键	格式	描述	是否必须
dataSchema	JSON 对象	描述所要消费数据的schema信息。dataSchema是固定的，不随数据消费方式改变	是
ioConfig	JSON 对象	描述所要消费数据的来源和消费去向。数据消费方式不同，ioConfig也不相同	是
tuningConfig	JSON 对象	调节数据消费时的一些参数。数据消费方式不同，可调节的参数也不相同	否

DataSchema

第一部分的dataSchema描述了数据的格式，如何解析该数据，典型结构如下：

```
{
    "dataSrouce": <name_of_dataSource>,
    "parser": {
        "type": <>,
        "parseSpec": {
            "format": <>,
            "timestampSpec": {},
            "dimensionsSpec": {}
        }
    },
    "metricsSpec": {},
    "granularitySpec": {}
}
```

键	格式	描述	是否必须
dataSource	字符串	数据源的名称	是
parser	JSON 对象	数据的解析方式	是

键	格式	描述	是否必须
metricsSpec	JSON 对象数组	聚合器（aggregator）列表	是
granularitySpec	JSON 对象	数据聚合设置，如创建segments，聚合粒度等等	是

- **parser**

  parser部分决定了您的数据如何被正确地解析，metricsSpec定义了数据如何被聚集计算，granularitySpec定义了数据分片的粒度、查询的粒度。

  对于parser，type有两个选项：string和hadoopString，后者用于Hadoop索引的job。parseSpec是数据格式解析的具体定义。

键	格式	描述	是否必须
type	字符串	数据格式，可以是 “json”, “jsonLowercase”, “csv”, “tsv” 几种格式	是
timestampSpec	JSON 对象	时间戳和时间戳类型	是
dimensionsSpec	JSON 对象	数据的维度（包含哪些列）	是

  对于不同的数据格式，可能还有额外的 parseSpec 选项。下面的表是 timestampSpec 和 dimensionsSpec 的描述：

键	格式	描述	是否必须
column	字符串	时间戳对应的列	是
format	字符串	时间戳类型，可选”iso”, “millis”, “posix”, “auto” 和 *joda time* 支持的类型	是

键	格式	描述	是否必须
dimensions	JSON 数组	描述数据包含哪些维度。每个维度可以只是个字符串，或者可以额外指明维度的属性，比如 “dimensions”: [“dimenssion1”, “dimenssion2”, “{“ type”: “long”, “name”: “dimenssion3”}]，默认是 string类型。	是

键	格式	描述	是否必须
dimensionExclusions	JSON 字符串数组	数据消费时要剔除的维度	否
spatialDimensions	JSON 对象数组	空间维度	否

- **metricsSpec**

  metricsSpec是一个JSON对象数组，定义了一些聚合器（aggregators）。聚合器通常有如下的结构：

  ```json
  {
      "type": <type>,
      "name": <output_name>,
      "fieldName": <metric_name>
  }
  ```

  官方提供了以下常用的聚合器：

类型	**type** 可选
count	count
sum	longSum, doubleSum, floatSum
min/max	longMin/longMax, doubleMin/doubleMax, floatMin/floatMax
first/last	longFirst/longLast, doubleFirst/doubleLast, floatFirst/floatLast
javascript	javascript
cardinality	cardinality
hyperUnique	hyperUnique

   说明：

  后三个属于高级聚合器，您需要参考 *Druid* 官方文档学习如何使用它们。

- **granularitySpec**

  聚合支持两种聚合方式：uniform和 arbitrary，前者以一个固定的时间间隔聚合数据，后者尽量保证每个segments大小一致，时间间隔是不固定的。目前uniform是默认选项。

键	格式	描述	是否必须
segmentGranularity	字符串	segments 粒度。uniform方式使用。	否，默认是”DAY”
queryGranularity	字符串	可供查询的最小数据聚合粒度	否
rollup	bool值	是否聚合	否，默认值为”true”
intervals	字符串	数据消费时间间隔	对于batch是，对于realtime否

ioConfig

第二部分ioConfig描述了数据来源。以下是一个Hadoop索引的例子：

```
{
    "type": "hadoop",
    "inputSpec": {
        "type": "static",
        "paths": "hdfs://emr-header-1.cluster-6789:9000/druid/
quickstart/wikiticker-2015-09-16-sampled.json"
    }
}
```

对于通过Tranquility处理的流式数据，这部分是不需要的。

Tunning Config

Tuning Config是指一些额外的设置。比如Hadoop对批量数据创建索引，可以在这里指定一些MapReduce参数。Tunning Config的内容依赖于您的数据来源可能有不同的内容。详细的例子可参考本服务自带的示例文件或官方文档。

11.7.4 Tranquility

Tranquility是一个以push方式向Druid实时发送数据的应用。它替用户解决了分区、多副本、服务发现、防止数据丢失等多个问题，简化了用户使用Druid的难度。它支持多种数据来源，包括Samza、Spark、Storm、Kafka、Flink等等。本文以Kafka为例，介绍在EMR中如何使用Tranquility从Kafka集群采集数据，并实时推送至Druid集群。

与Kafka集群交互

首先是Druid集群与Kafka集群的交互。两个集群交互的配置方式大体和Hadoop集群类似，均需要设置连通性、hosts等。对于非安全Kafka集群，请按照以下步骤操作：

1. 确保集群间能够通信（两个集群在一个安全组下，或两个集群在不同安全组，但两个安全组之间配置了访问规则）。
2. 将 Kafka 集群的 hosts 写入到 Druid 集群每一个节点的 hosts 列表中，注意 Kafka 集群的 hostname 应采用长名形式，如 emr-header-1.cluster-xxxxxxxx。

对于安全的 Kafka 集群，您需要执行下列操作（前两步与非安全 Kafka 集群相同）：

1. 确保集群间能够通信（两个集群在一个安全组下，或两个集群在不同安全组，但两个安全组之间配置了访问规则）。
2. 将 Kafka 集群的 hosts 写入到 Druid 集群每一个节点的 hosts 列表中，注意 Kafka 集群的 hostname 应采用长名形式，如 emr-header-1.cluster-xxxxxxxx。
3. 设置两个集群间的 Kerberos 跨域互信（详情参考跨域互信），且最好做双向互信。
4. 准备一个客户端安全配置文件：

```
KafkaClient {
      com.sun.security.auth.module.Krb5LoginModule required
      useKeyTab=true
      storeKey=true
      keyTab="/etc/ecm/druid-conf/druid.keytab"
      principal="druid@EMR.1234.COM";
  };
```

   之后将该配置文件同步到 Druid 集群的所有节点上，放置于某一个目录下面（例如*/tmp/kafka/kafka_client_jaas.conf*）。

5. 在 Druid 配置页面的 overlord.jvm 里新增如下选项：

```
Djava.security.auth.login.config=/tmp/kafka/kafka_client_jaas.conf
```

   。

6. 在 Druid 配置页面的 middleManager.runtime 里配置`druid.indexer.runner.javaOpts=-Djava.security.auth.login.confi=/tmp/kafka/kafka_client_jaas.conf`和其他JVM启动参数。
7. 重启Druid服务。

Druid使用Tranquility Kafka

由于Tranquility是一个服务，它对于Kafka来说是消费者，对于Druid来说是客户端。您可以使用中立的机器来运行Tranquility，只要这台机器能够同时连通 Kafka 集群和 Druid 集群即可。

1. Kafka端创建一个名为 pageViews 的 topic。

```
-- 如果开启了kafka 高安全：
```

```
 export KAFKA_OPTS="-Djava.security.auth.login.config=/etc/ecm/kafka
-conf/kafka_client_jaas.conf"
 --
 ./bin/kafka-topics.sh --create --zookeeper emr-header-1:2181,
emr-header-2:2181,emr-header-3:2181/kafka-1.0.1 --partitions 1 --
replication-factor 1 --topic pageViews
```

2. 下载 Tranquility 安装包，并解压至某一路径下。
3. 配置 datasource。

   这里假设您的 topic name 为 pageViews，并且每条 topic 都是如下形式的 json 文件：

```
{"time": "2018-05-23T11:59:43Z", "url": "/foo/bar", "user": "alice",
 "latencyMs": 32}
 {"time": "2018-05-23T11:59:44Z", "url": "/", "user": "bob", "
latencyMs": 11}
 {"time": "2018-05-23T11:59:45Z", "url": "/foo/bar", "user": "bob",
 "latencyMs": 45}
```

   对应的 dataSrouce 的配置如下：

```
{
   "dataSources" : {
     "pageViews-kafka" : {
       "spec" : {
         "dataSchema" : {
           "dataSource" : "pageViews-kafka",
           "parser" : {
             "type" : "string",
             "parseSpec" : {
               "timestampSpec" : {
                 "column" : "time",
                 "format" : "auto"
               },
               "dimensionsSpec" : {
                 "dimensions" : ["url", "user"],
                 "dimensionExclusions" : [
                   "timestamp",
                   "value"
                 ]
               },
               "format" : "json"
             }
           },
           "granularitySpec" : {
             "type" : "uniform",
             "segmentGranularity" : "hour",
             "queryGranularity" : "none"
           },
           "metricsSpec" : [
             {"name": "views", "type": "count"},
             {"name": "latencyMs", "type": "doubleSum", "fieldName":
 "latencyMs"}
           ]
         },
         "ioConfig" : {
           "type" : "realtime"
         },
         "tuningConfig" : {
```

```
            "type" : "realtime",
            "maxRowsInMemory" : "100000",
            "intermediatePersistPeriod" : "PT10M",
            "windowPeriod" : "PT10M"
          }
        },
        "properties" : {
          "task.partitions" : "1",
          "task.replicants" : "1",
          "topicPattern" : "pageViews"
        }
      }
    },
    "properties" : {
      "zookeeper.connect" : "localhost",
      "druid.discovery.curator.path" : "/druid/discovery",
      "druid.selectors.indexing.serviceName" : "druid/overlord",
      "commit.periodMillis" : "15000",
      "consumer.numThreads" : "2",
      "kafka.zookeeper.connect" : "emr-header-1.cluster-500148518:
2181,emr-header-2.cluster-500148518:2181,   emr-header-3.cluster-
500148518:2181/kafka-1.0.1",
      "kafka.group.id" : "tranquility-kafka",
    }
 }
```

4. 运行如下命令启动 Tranquility。

```
./bin/tranquility kafka -configFile
```

5. 在 Kafka 端启动 producer 并发送一些数据。

```
./bin/kafka-console-producer.sh --broker-list emr-worker-1:9092,emr-
worker-2:9092,emr-worker-3:9092 --topic pageViews
```

输入：

```
{"time": "2018-05-24T09:26:12Z", "url": "/foo/bar", "user": "alice",
 "latencyMs": 32}
 {"time": "2018-05-24T09:26:13Z", "url": "/", "user": "bob", "
latencyMs": 11}
 {"time": "2018-05-24T09:26:14Z", "url": "/foo/bar", "user": "bob",
 "latencyMs": 45}
```

在Tranquility日志中查看相应的消息，在Druid端则可以看到启动了相应的实时索引 task。

11.7.5 Kafka Indexing Service

Kafka Indexing Service是Druid推出的利用Druid的Indexing Service服务实时消费Kafka数据的插件。该插件会在Overlord中启动一个supervisor，supervisor启动之后会在 Middlemanager中启动一些indexing tasks，这些tasks会连接到Kafka集群消费topic数据，并完成索引创建。您需要做的，就是准备一个数据消费格式文件，之后通过REST API手动启动supervisor。

与Kafka集群交互

请参考*Druid 使用 Tranquility Kafka*一节的介绍。

使用Druid Kafka Indexing Service实时消费Kafka数据

1. 在Kafka集群上（或者gateway上）执行下述命令创建一个名为“metrics” 的topic。

```
-- 如果开启了Kafka 高安全：
 export KAFKA_OPTS="-Djava.security.auth.login.config=/etc/ecm/kafka
-conf/kafka_client_jaas.conf"
 --
 kafka-topics.sh --create --zookeeper emr-header-1:2181,emr-header-
2,emr-header-3/kafka-1.0.0 --partitions 1 --replication-factor 1 --
topic metrics
```

其中各个参数可根据需要进行调整。—**zookeeper** 参数中 */kafka-1.0.0* 部分为path，其具体值您可以登录EMR控制台，在Kafka集群的Kafka服务配置页面查看 zookeeper.connect配置项的值。如果是您自己搭建的Kafka集群，—**zookeeper** 参数可根据您的实际配置进行改变。

2. 定义数据源的数据格式描述文件，我们将其命名为 metrics-kafka.json，并置于当前目录下面（或者放置于其他您指定的目录）。

```
{
     "type": "kafka",
     "dataSchema": {
         "dataSource": "metrics-kafka",
         "parser": {
             "type": "string",
             "parseSpec": {
                 "timestampSpec": {
                     "column": "time",
                     "format": "auto"
                 },
                 "dimensionsSpec": {
                     "dimensions": ["url", "user"]
                 },
                 "format": "json"
             }
         },
         "granularitySpec": {
             "type": "uniform",
             "segmentGranularity": "hour",
             "queryGranularity": "none"
         },
         "metricsSpec": [{
                 "type": "count",
                 "name": "views"
             },
             {
                 "name": "latencyMs",
                 "type": "doubleSum",
                 "fieldName": "latencyMs"
             }
         ]
     },
```

```
    "ioConfig": {
        "topic": "metrics",
        "consumerProperties": {
            "bootstrap.servers": "emr-worker-1.cluster-xxxxxxxx:
9092(您 Kafka 集群的 bootstrap.servers)",
            "group.id": "kafka-indexing-service",
            "security.protocol": "SASL_PLAINTEXT",
            "sasl.mechanism": "GSSAPI"
        },
        "taskCount": 1,
        "replicas": 1,
        "taskDuration": "PT1H"
    },
    "tuningConfig": {
        "type": "Kafka",
        "maxRowsInMemory": "100000"
    }
}
```

说明：

ioConfig.consumerProperties.security.protocol 和 **ioConfig.consumerProperties.sasl.mechanism** 为安全相关选项（非安全 Kafka 集群不需要）。

3. 执行下述命令添加Kafka supervisor。

```
curl --negotiate -u:druid -b ~/cookies -c ~/cookies -XPOST -H '
Content-Type: application/json' -d @metrics-kafka.json http://emr-
header-1.cluster-1234:18090/druid/indexer/v1/supervisor
```

其中—negotiate、-u、-b、-c 等选项是针对安全Druid集群。

4. 在Kafka集群上开启一个console producer。

```
-- 如果开启了Kafka高安全：
 export KAFKA_OPTS="-Djava.security.auth.login.config=/etc/ecm/kafka
-conf/kafka_client_jaas.conf"
 echo -e "security.protocol=SASL_PLAINTEXT\nsasl.mechanism=GSSAPI"
 > /tmp/Kafka/producer.conf
 --
 Kafka-console-producer.sh --producer.config /tmp/kafka/producer.
conf --broker-list emr-worker-1:9092，emr-worker-2:9092，emr-worker-3
:9092 --topic metrics
 >
```

其中 *—producer.config /tmp/Kafka/producer.conf* 是针对安全 Kafka 集群的选项。

5. 在 kafka_console_producer 的命令提示符下输入一些数据。

```
{"time": "2018-03-06T09:57:58Z", "url": "/foo/bar", "user": "alice",
 "latencyMs": 32}
 {"time": "2018-03-06T09:57:59Z", "url": "/", "user": "bob", "
latencyMs": 11}
```

```
 {"time": "2018-03-06T09:58:00Z", "url": "/foo/bar", "user": "bob",
 "latencyMs": 45}
```

其中时间戳可用如下 python 命令生成：

```
python -c 'import datetime; print(datetime.datetime.utcnow().
strftime("%Y-%m-%dT%H:%M:%SZ"))'
```

6. 准备一个查询文件，命名为 metrics-search.json。

```
{
     "queryType" : "search",
     "dataSource" : "metrics-kafka",
     "intervals" : ["2018-03-02T00:00:00.000/2018-03-08T00:00:00.000
"],
     "granularity" : "all",
     "searchDimensions": [
         "url",
         "user"
     ],
     "query": {
         "type": "insensitive_contains",
         "value": "bob"
     }
 }
```

7. 在 Druid 集群 master 上执行查询。

```
curl --negotiate -u:Druid -b ~/cookies -c ~/cookies -XPOST -H '
Content-Type: application/json' -d @metrics-search.json http://emr-
header-1.cluster-1234:8082/druid/v2/?pretty
```

其中—negotiate、-u、-b、-c 等选项是针对安全 druid 集群。

8. 如果一切正常，您将看到类似如下的查询结果。

```
[ {
   "timestamp" : "2018-03-06T09:00:00.000Z",
   "result" : [ {
     "dimension" : "user",
     "value" : "bob",
     "count" : 2
   } ]
 } ]
```

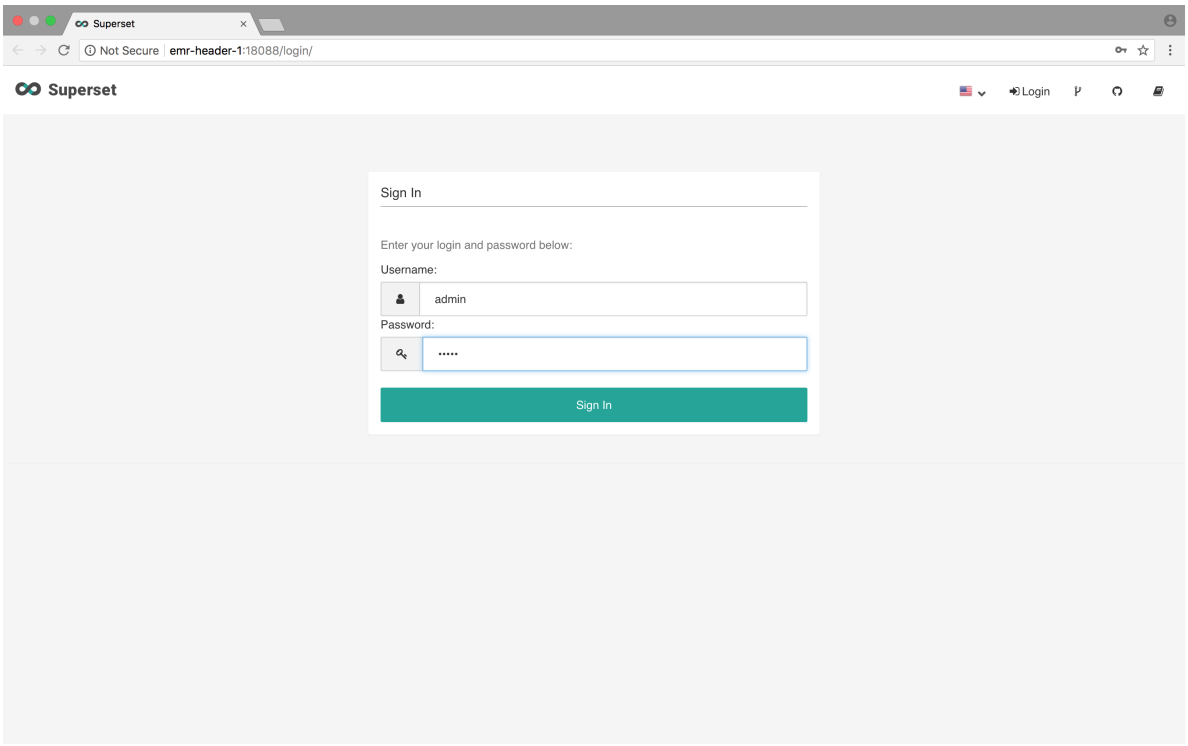
11.7.6 Superset

Druid集群集成了Superset工具。Superset对Druid做了深度集成，同时也支持多种关系型数据库。由于Druid也支持SQL，所以可以通过Superset以两种方式访问Druid，即Druid原生查询语言或者SQL。

Superset默认安装在emr-header-1节点，目前还不支持HA。在使用该工具前，确保您的主机能够正常访问emr-header-1。您可以通过打 *SSH* 隧道 的方式连接到主机。

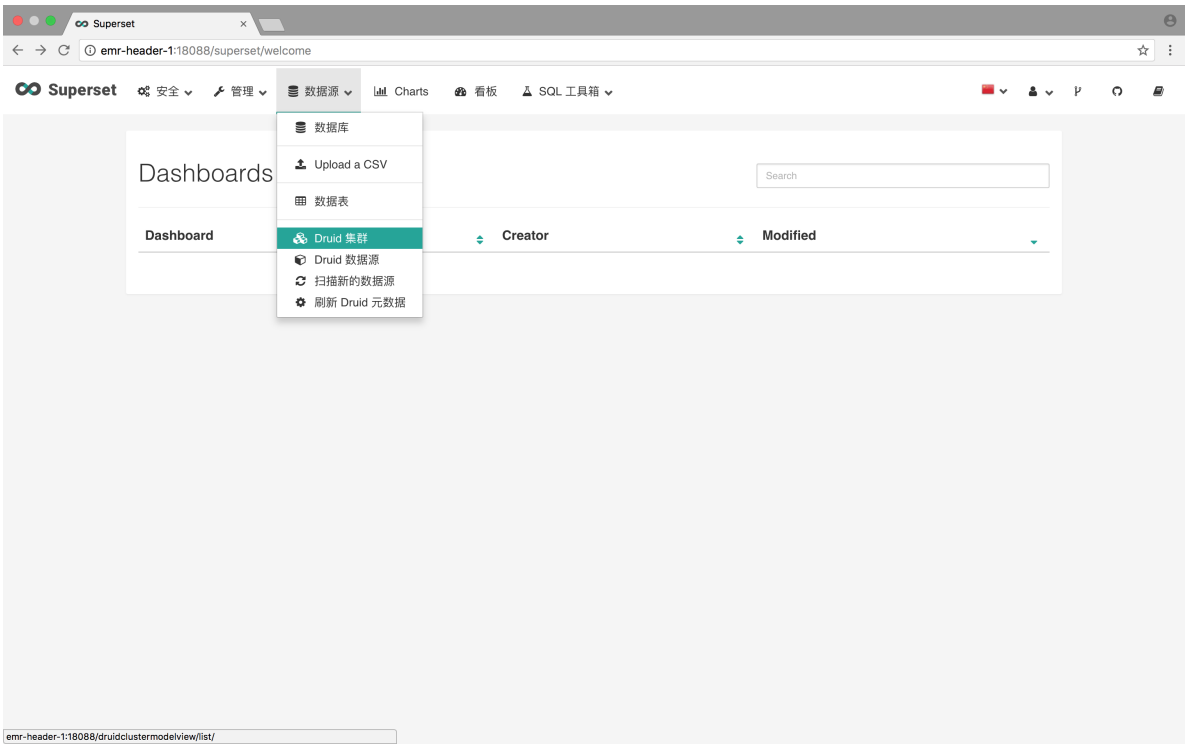
1. 登录Superset。

   在浏览器地址栏中输入 http://emr-header-1:18088 后回车，打开Superset登录界面，默认用户名/密码为 admin/admin，请您登录后及时修改密码。

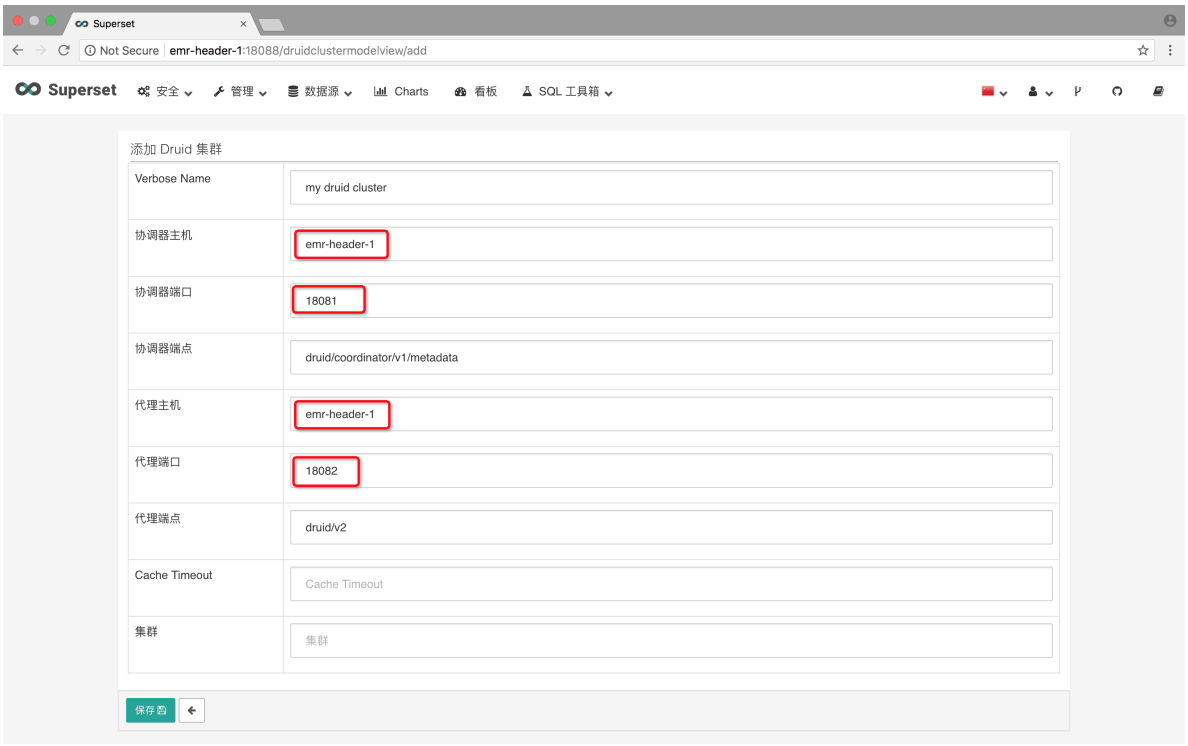


2. 添加 Druid 集群。

   登录后默认为英文界面，可点击右上角的国旗图标选择合适的语言。接下来在上方菜单栏中依次选择**数据源 > Druid 集群**来添加一个 Druid 集群。



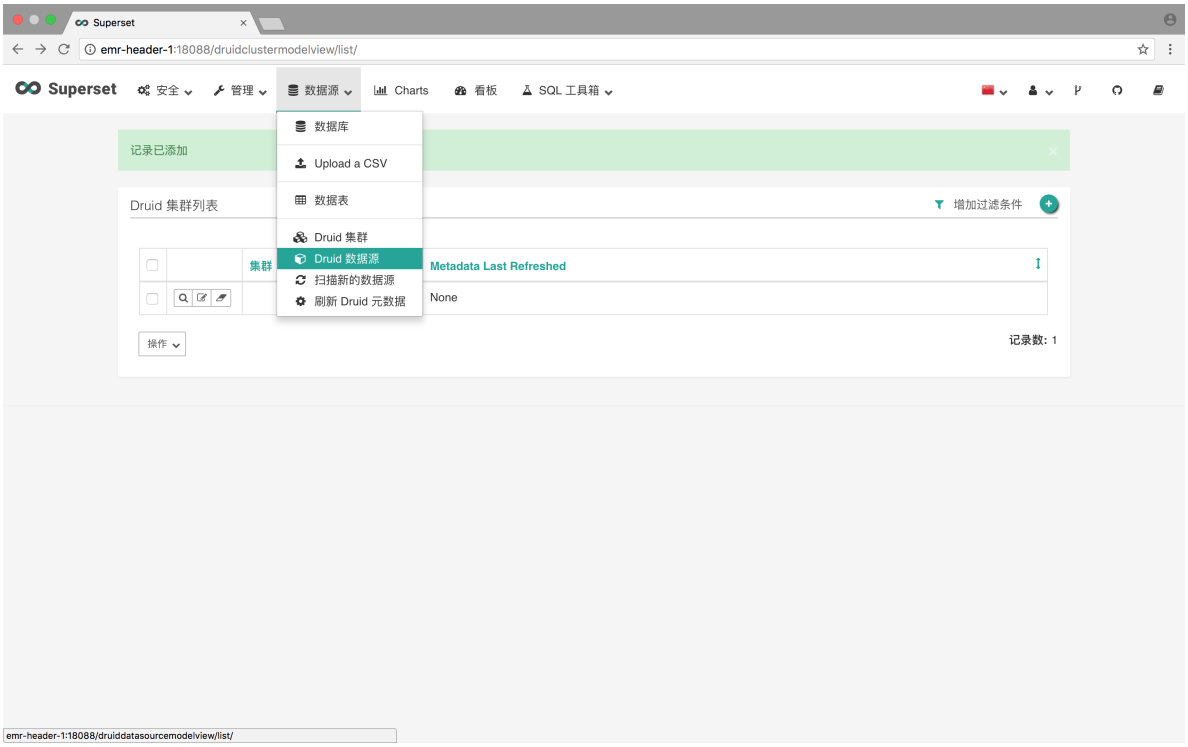
配置好协调机（Coordinator）和代理机（Broker）的地址，注意 E-MapReduce 中默认端口均为相应的开源端口前加数字1，例如开源 Broker 端口为 8082，E-MapReduce 中为 18082。



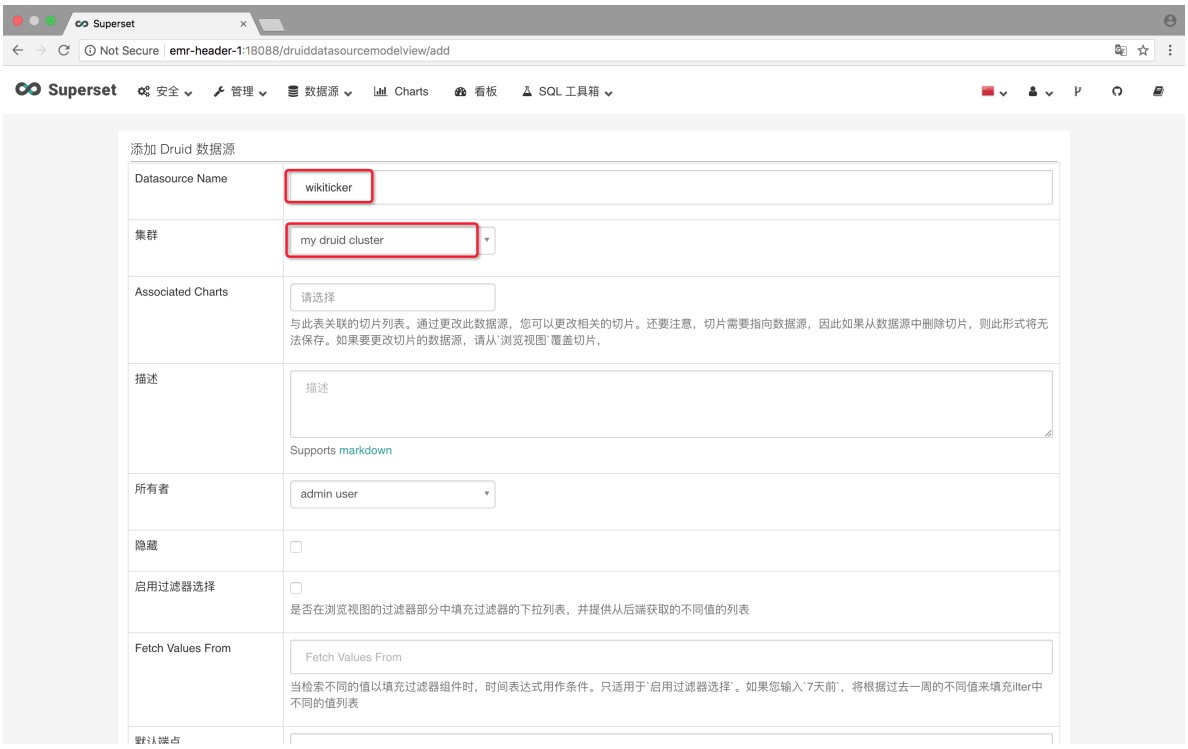
3. 刷新或者添加新数据源。

添加好 Druid 集群之后，您可以单击**数据源 > 扫描新的数据源**，这时 Druid 集群上的数据源（datasource）就可以自动被加载进来。

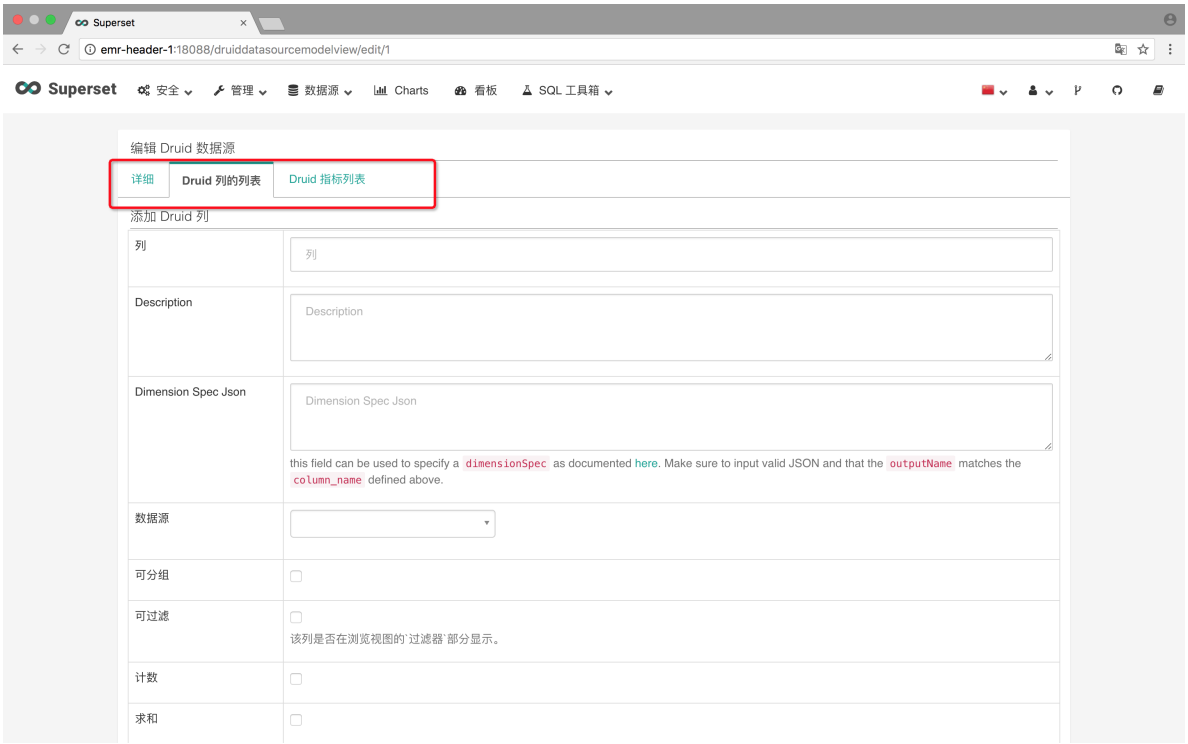
您也可以在界面上单击**数据源 > Druid 数据源**自定义新的数据源（其操作等同于写一个 data source ingestion 的 json 文件），步骤如下：



自定义数据源时需要填写必要的信息，然后保存。

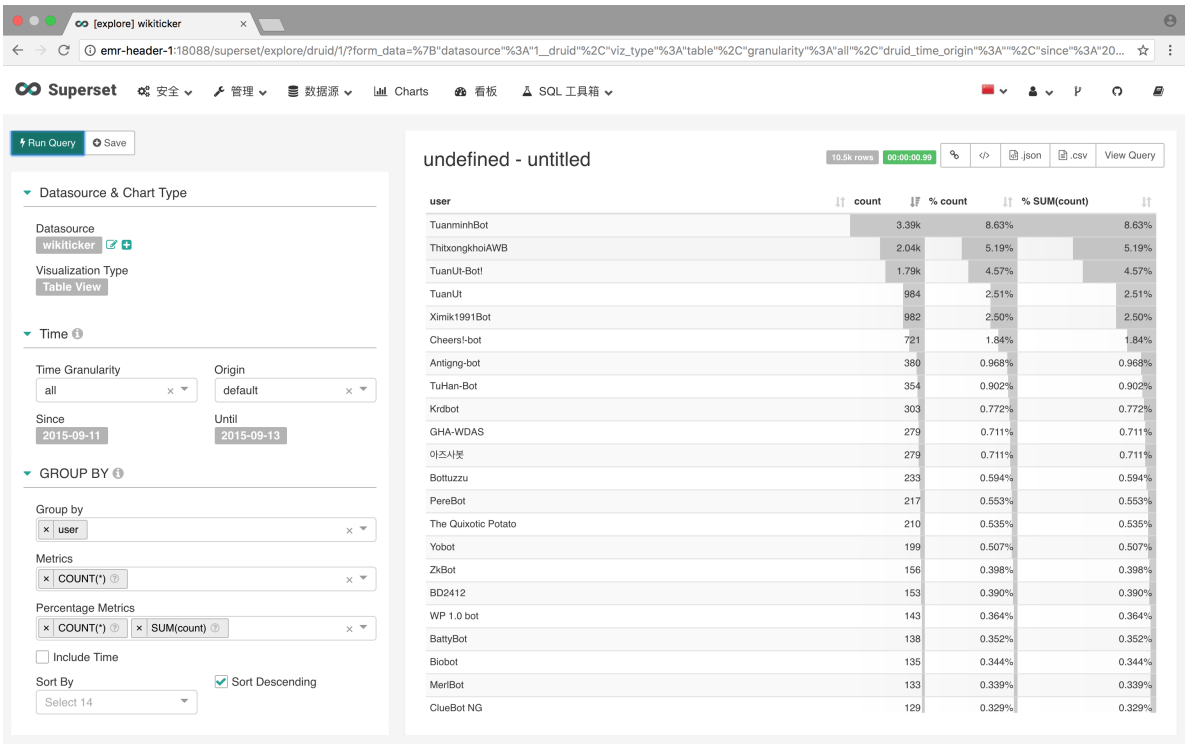


保存之后点击左侧三个小图标中的第二个，编辑该数据源，填写相应的维度列与指标列等信息。



4. 查询 Druid。

数据源添加成功后，单击数据源名称，进入查询页面进行查询。

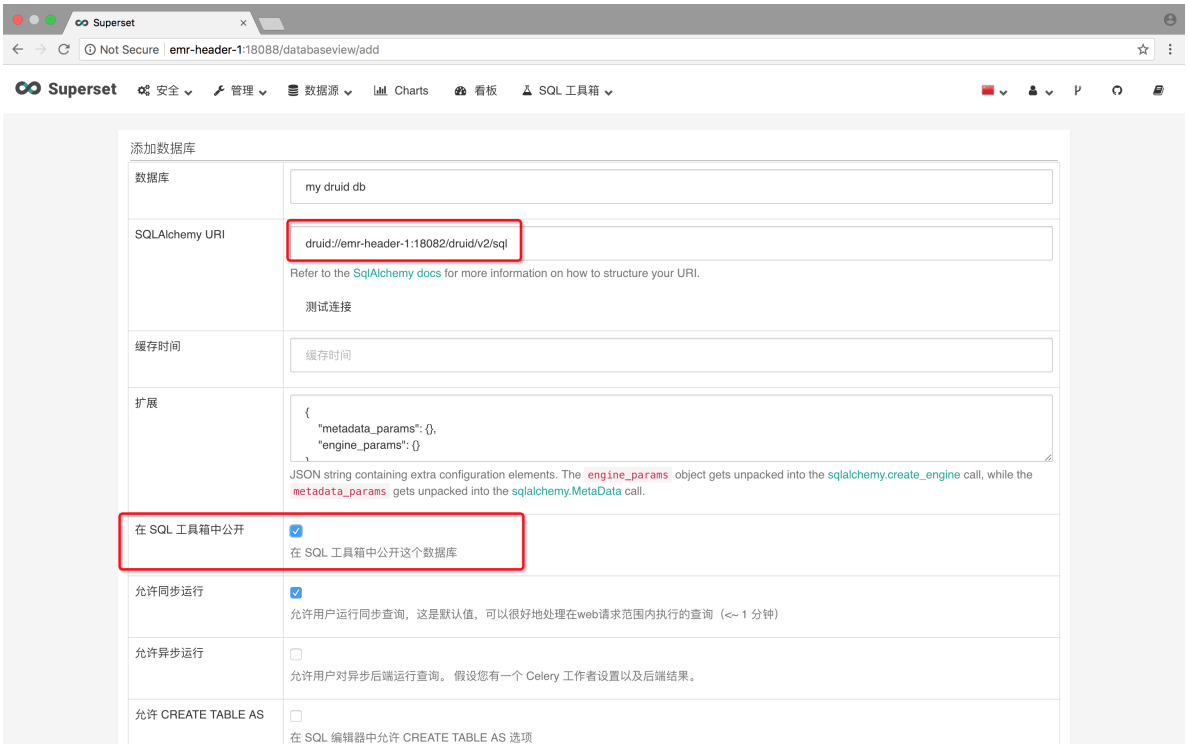


5. （可选）将 Druid 作为 数据库使用。

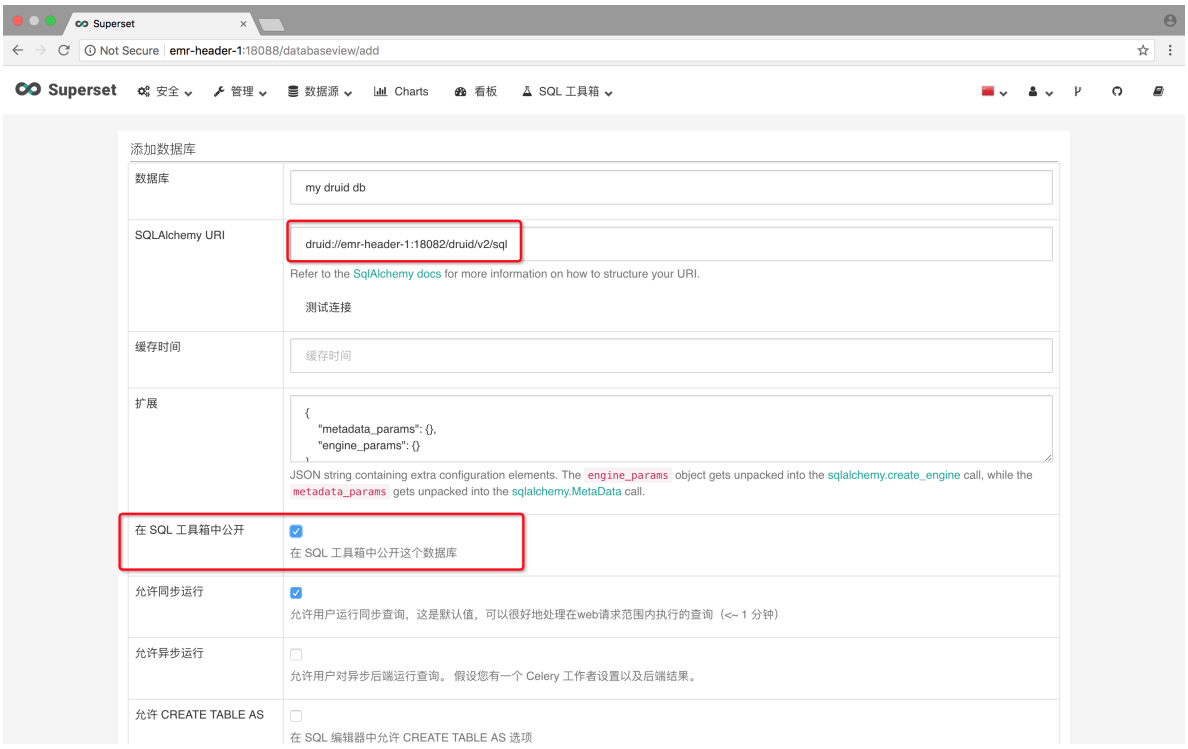
Superset 提供了 SQLAlchemy 以多种方言支持各种各样的数据库，其支持的数据库类型如下表所示。

database	pypi package	SQLAlchemy URI prefix
MySQL	`pip install mysqlclient`	`mysql://`
Postgres	`pip install psycopg2`	`postgresql+psycopg2://`
Presto	`pip install pyhive`	`presto://`
Oracle	`pip install cx_Oracle`	`oracle://`
sqlite		`sqlite://`
Redshift	`pip install sqlalchemy-redshift`	`redshift+psycopg2://`
MSSQL	`pip install pymssql`	`mssql://`
Impala	`pip install impyla`	`impala://`
SparkSQL	`pip install pyhive`	`jdbc+hive://`
Greenplum	`pip install psycopg2`	`postgresql+psycopg2://`
Athena	`pip install "PyAthenaJDBC>1.0.9"`	`awsathena+jdbc://`
Vertica	`pip install sqlalchemy-vertica-python`	`vertica+vertica_python://`
ClickHouse	`pip install sqlalchemy-clickhouse`	`clickhouse://`
Kylin	`pip install kylinpy`	`kylin://`

Superset 亦支持该方式访问 Druid，Druid 对应的 SQLAlchemy URI 为“druid://emr-header-1:18082/druid/v2/sql”，如下图所示，将 Druid 作为一个数据库添加：



接下来就可以在 SQL 工具箱里用 SQL 进行查询了：



11.7.7 常见问题

Druid使用过程中遇到的一些常见问题以及解决方法。

索引失败问题分析思路

当发现索引失败时，一般遵循如下排错思路：

- 对于批量索引

1. 如果 curl 直接返回错误，或者不返回，检查一下输入文件格式。或者 curl 加上 **-v** 参数，观察REST API的返回情况。
2. 在 Overlord 页面观察作业执行情况，如果失败，查看页面上的 logs。
3. 在很多情况下并没有生成 logs。如果是 Hadoop 作业，打开 YARN 页面查看是否有索引作业生成，并查看作业执行 log。
4. 如果上述情况都没有定位到错误，需要登录到 Druid 集群，查看 overlord 的执行日志（位于 *`/mnt/disk1/log/druid/overlord—emr-header-1.cluster-xxxx.log`* ），如果是 HA 集群，查看您提交作业的那个 Overlord。
5. 如果作业已经被提交到 Middlemanager，但是从 Middlemanager 返回了失败，则需要从 Overlord 中查看作业提交到了那个 worker，并登录到相应的 worker，查看 Middlemanager

的日志（位于*/mnt/disk1/log/druid/middleManager-emr-header-1.cluster-xxxx.log*）。

- 对于 Tranquility 实时索引

  查看 Tranquility log，查看消息是否被接收到了或者是否被丢弃（drop）掉了。

  其余的排查步骤同批量索引的 2～5。

  错误多数情况为集群配置问题和作业问题。集群配置问题包括：内存参数是否合理、跨集群联通性是否正确、安全集群访问是否通过、principal 是否正确等等，作业问题包括作业描述文件格式正确、输入数据是否能够正常被解析，以及一些其他的作业相关的配置（如ioConfig等）。

常见问题列表

- 组件启动失败

  此类问题多数是由于组件 JVM 运行参数配置问题，例如机器可能没有很大的内存，而配置了较大的 JVM 内存或者较多的线程数量。

  解决方法：查看组件日志并调整相关参数即可解决。JVM 内存涉及堆内存和直接内存。具体可参考*http://druid.io/docs/latest/operations/performance-faq.html*。

- 索引时 YARN task 执行失败，显示诸如`Error: class com.fasterxml.jackson.datatype.guava.deser.HostAndPortDeserializer overrides final method deserialize.(Lcom/fasterxml/jackson/core/JsonParser;Lcom/fasterxml/jackson/databind/DeserializationContext;)Ljava/lang/Object;` 之类的 jar 包冲突错误。

  解决方法：在 indexing 的作业配置文件中加入如下配置：

  ```
  "tuningConfig" : {
       ...
       "jobProperties" : {
           "mapreduce.job.classloader": "true"
           或者
           "mapreduce.job.user.classpath.first": "true"
       }
       ...
   }
  ```

  其中参数 **`mapreduce.job.classloader`** 让MR job用独立的 classloader，**`mapreduce.job.user.classpath.first`** 是让MapReduce优先使用用户的jar包，两个配置项配置一个即可。 可参考：*http://druid.io/docs/0.9.2-rc1/operations/other-hadoop.html*。

- indexing作业的日志中报reduce无法创建segments目录

解决方法：

- 注意检查 deep storage 的设置，包括 type 和 directory。当 type 为 local 时，注意 directory 的权限设置。当 type 为 HDFS 时，directory 尽量用完整的 HDFS 路径写法，如 hdfs://:9000/。hdfs_master 最好用 IP，如果用域名，要用完整域名，如 emr-header-1.cluster-xxxxxxxx，而不是 emr-header-1。
- 用 Hadoop 批量索引时，要将 segments 的 deep storage 设置为 “hdfs”，“local” 的方式会导致 MR 作业处于 UNDEFINED 状态，这是因为远程的 YARN 集群无法在 reduce task 下创建 local 的 segments 目录。（此针对独立 Druid 集群）。

- Failed to create directory within 10000 attempts…

  此问题一般为 JVM 配置文件中 java.io.tmp 设置的路径不存在。设置该路径并确保 Druid 账户有权限访问即可。

- com.twitter.finagle.NoBrokersAvailableException: No hosts are available for disco!firehose:druid:overlord

  此问题一般是 ZooKeeper 的连接问题。确保 Druid 与 Tranquility 对于 ZooKeeper 有相同的连接字符串。注意：Druid 默认的ZooKeeper路径为 /druid，因此确保 Tranquility 设置中 zookeeper.connect 包含路径 /druid。（另注意 Tranquility Kafka 设置中有两个 ZooKeeper 的设置，一个为 zookeeper.connect，连接的 Druid 集群的 ZooKeeper，一个为 kafka.zookeeper.connect，连接的 Kafka 集群的 ZooKeeper。这两个 ZooKeeper 可能不是一个 ZooKeeper 集群）。

- 索引时 MiddleManager 报找不到类 com.hadoop.compression.lzo.LzoCodec

  这是因为 EMR 的 Hadoop 集群配置了 lzo 压缩。

  解决方法：拷贝 EMR HADOOP_HOME/lib 下的 jar 包和 native 文件夹到 Druid 的 druid.extensions.hadoopDependenciesDir（默认为 DRUID_HOME/hadoop-dependencies）

- 索引时报如下错误：

```
2018-02-01T09:00:32,647 ERROR [task-runner-0-priority-0] com.hadoop.
compression.lzo.GPLNativeCodeLoader - could not unpack the binaries
  java.io.IOException: No such file or directory
          at java.io.UnixFileSystem.createFileExclusively(Native
Method) ~[?:1.8.0_151]
          at java.io.File.createTempFile(File.java:2024) ~[?:1.8.
0_151]
          at java.io.File.createTempFile(File.java:2070) ~[?:1.8.
0_151]
```

```
            at com.hadoop.compression.lzo.GPLNativeCodeLoader.
unpackBinaries(GPLNativeCodeLoader.java:115) [hadoop-lzo-0.4.21-
SNAPSHOT.jar:?]
```

这个问题还是因为 java.io.tmp 路径不存在。设置该路径并确保 Druid 账户有权限访问。

11.8 Presto

11.8.1 连接器

系统连接器

- 概述

  通过本连接器可以使用SQL查询Presto集群的基本信息和度量。

- 配置

  无需配置，所有信息都可以通过名为 system 的catalog获取。

  示例如下：

```
--- 列出所有支持的数据项目
SHOW SCHEMAS FROM system;
--- 列出运行时项目中的所有数据项
SHOW TABLES FROM system.runtime;
--- 获取节点状态
SELECT * FROM system.runtime.nodes;
     node_id |         http_uri         | node_version | coordinator
 | state
-------------+--------------------------+--------------
+-------------+--------
 3d7e8095-...| http://192.168.1.100:9090| 0.188        | false
 | active
 7868d742-...| http://192.168.1.101:9090| 0.188        | false
 | active
 7c51b0c1-...| http://192.168.1.102:9090| 0.188        | true
 | active
--- 强制取消一个查询
CALL system.runtime.kill_query('20151207_215727_00146_tx3nr');
```

- 数据表

  本连接器提供了如下几个数据表：

TABLE	SCHEMA	说明
catalogs	metadata	该表包含了本连接器支持的所有catalog列表
schema_properties	metadata	本表包含创建新Schema时可以设置的可用属性列表。
table_properties	metadata	本表包含创建新表时可以设置的可用属性列表。

TABLE	SCHEMA	说明
nodes	runtime	本表包含了Presto集群中所有可见节点及其状态列表
queries	runtime	查询表包含了Presto群集上当前和最近运行的查询的信息，包括原始查询文本（SQL），运行查询的用户的身份以及有关查询的性能信息，如查询排队和分析的时间等。
tasks	runtime	任务表包含了关于Presto查询中涉及的任务的信息，包括它们的执行位置以及每个任务处理的行数和字节数。
transactions	runtime	本表包含当前打开的事务和相关元数据的列表。这包括创建时间，空闲时间，初始化参数和访问目录等信息。

- 存储过程

  本连接器支持如下存储过程：

  `runtime.kill_query(id)` 取消给定id的查询

JMX连接器

- 概述

  可以通过JMX连接器查询Presto集群中所有节点的JMX信息。本连接器通常用于系统监控和调试。通过修改本连接器的配置，可以实现JMX信息定期转储的功能。

- 配置

  创建 *`etc/catalog/jmx.properties`* 文件，添加如下内容，一边启用JMX连接器。

  ```
  connector.name=jmx
  ```

  如果希望定期转储JMX数据，可以在配置文件中添加如下内容：

  ```
  connector.name=jmx
  jmx.dump-tables=java.lang:type=Runtime,com.facebook.presto.execution
  .scheduler:name=NodeScheduler
  jmx.dump-period=10s
  jmx.max-entries=86400
  ```

  其中：

  — **`dump-tables`** 是用逗号隔开的 MBean（Managed Beans）列表。该配置项指定了每个采样周期哪些MBean指标会被采样并存储到内存中。

  — **`dump-period`** 用于设置采样周期，默认为10s。

  — **`max-entries`** 用于设置历史记录的最大长度，默认为86400条。

如果指标项的名称中包含逗号，则需要使用 `\,` 进行转义，如下所示：

```
connector.name=jmx
jmx.dump-tables=com.facebook.presto.memory:type=memorypool\\,name=
general,\
   com.facebook.presto.memory:type=memorypool\\,name=system,\
   com.facebook.presto.memory:type=memorypool\\,name=reserved
```

- 数据表

  JMX连接器提供了两个 schemas，分别为 **current** 和 **history**。其中：

  current 中包含了Presto集群中每个节点的当前的MBean，BMean的名称即为 **current** 中的表名（如果bean的名称中包含非标准字符，则需在查询时用双引号将表名扩起来），可以通过如下语句获取：

  ```
  SHOW TABLES FROM jmx.current;
  ```

  示例如下：

  ```
  --- 获取每个节点的jvm信息
  SELECT node, vmname, vmversion
  FROM jmx.current."java.lang:type=runtime";
        node    |              vmname               | vmversion
  --------------+-----------------------------------+-----------
   ddc4df17-xxx | Java HotSpot(TM) 64-Bit Server VM | 24.60-b09
  (1 row)
  ```

  ```
  --- 获取每个节点最大和最小的文件描述符个数指标
  SELECT openfiledescriptorcount, maxfiledescriptorcount
  FROM jmx.current."java.lang:type=operatingsystem";
   openfiledescriptorcount | maxfiledescriptorcount
  -------------------------+------------------------
                       329 |                  10240
  (1 row)
  ```

  history 中包含了配置文件中配置的需要转储的指标对应的数据表。可以通过如下语句进行查询：

  ```
  SELECT "timestamp", "uptime" FROM jmx.history."java.lang:type=
  runtime";
          timestamp        | uptime
  -------------------------+--------
   2016-01-28 10:18:50.000 |  11420
   2016-01-28 10:19:00.000 |  21422
   2016-01-28 10:19:10.000 |  31412
  (3 rows)
  ```

Kafka连接器

- 概述

本连接器将Kafka上的topic映射为Presto中的表。kafka中的每条记录都映射为Presto表中的消息。

说明：

由于Kafka中，数据是动态变化的，因此，在使用presto进行多次查询时，有时候会出现一些比较奇怪的现象。目前，Presto还没有处理这些情况的能力。

- 配置

创建 *etc/catalog/kafka.properties* 文件，添加如下内容，一边启用JMX连接器。

```
connector.name=kafka
kafka.table-names=table1,table2
kafka.nodes=host1:port,host2:port
```

说明：

Presto可以同时连接多个Kafka集群，只需要在配置目录中添加新的properties文件即可，文件名将会被映射为Presto的catalog。如新增一个 *orders.properties* 配置文件，Presto会创建一个新的名为 orders 的catalog。

```
## orders.properties
connector.name=kafka  # 这个表示连接器类型，不能变
kafka.table-names=tableA,tableB
kafka.nodes=host1:port,host2:port
```

Kafka连接器提供了如下几个属性：

- kafka.table-names

  说明： 必选，定义本连接器支持的表格列表。

  详情： 这里的文件名可以使用Schema名称进行修饰，形式如*{schema_name}.{table_name}*。也可以不使用Schema名称修饰，此时，表格将被映射到 *kafka.default-schema* 中定义的schema中。

- kafka.default-schema

  说明： 可选，默认的Schema名称, 默认值为 `default`。

- kafka.nodes

  说明： 必选，Kafka集群中的节点列表。

详情： 配置格式形如 `hostname:port[,hostname:port…]`。此处的可以只配置部分kafka节点，但是，Presto必须能够连接到kafka集群中的所有节点。否则，可能拿不到部分数据。

- kafka.connect-timeout

  说明： 可选，连接器与Kafka集群的超时时间，默认为10s。

  详情： 如果Kafka集群压力比较大，创建连接可能需要相当长的时间，从而导致Presto在执行查询时出现超时的情况。此时，增加当前的配置值是一个不错的选择。

- kafka.buffer-size

  说明： 可选，读缓冲区大小，默认为64kb。

  详情： 用于设置从Kafka读取数据的内部数据缓冲区的大小。 数据缓冲区必须至少大于一条消息的大小。 每个Worker和数据节点都会分配一个数据缓冲区。

- kafka.table-description-dir

  说明： 可选，topic（表）描述文件目录，默认为 *`etc/kafka`*。

  详情： 该目录下保存着JSON格式的数据表定义文件（必须使用 .json 作为后缀）。

- kafka.hide-internal-columns

  说明：可选，需要隐藏的预置列清单，默认为`true`。

  详情： 除了在表格描述文件中定义的数据列之外，连接器还为每个表格维护了许多额外的列。本属性用于控制这些列在是否会在 `DESCRIBE` 和 `SELECT *` 语句的执行结果中显示。无论本配置设置是什么，这些列都可以参与查询过程中。

Kafka连接器提供的内部列如下表所示：

列名	类型	说明
_partition_id	BIGINT	本行记录所在的Kafka分区ID。
_partition_offset	BIGINT	本行记录在Kafka分区中的偏移量。
_segment_start	BIGINT	包含此行的数据段的最低偏移量。此偏移量是针对每个分区的。
_segment_end	BIGINT	包含此行的数据段的最大偏移量（为下一个段的起始偏移量）。此偏移量是针对每个分区的。

列名	类型	说明
_segment_count	BIGINT	当前行在数据段中的序号。对于没有压缩的topic，**_segment_start** + **_segment_count** = **_partition_offset**。
_message_corrupt	BOOLEAN	如果解码器无法解码本行记录，本字段将会被设为TRUE。
_message	VARCHAR	将消息字节作为UTF-8编码的字符串。当topic的消息为文本类型是，这个字段比较有用。
_message_length	BIGINT	本行消息的字节长度。
_key_corrupt	BOOLEAN	如果键解码器无法解码本行记录，该字段将会被设为TRUE。
_key	VARCHAR	将键字节作为UTF-8编码的字符串。当topic的消息为文本类型是，这个字段比较有用。
_key_length	BIGINT	消息中键的字节长度。

 说明：

对于那些没有定义文件的表，**_key_corrupt** 和 **_message_corrupt** 永远为 FALSE。

- 表格定义文件

Kafka本身是Schema-Less的消息系统，消息的格式需要生产者和消费者自己来定义。而Presto要求数据必须可以被映射成表的形式。因此，通常需要用户根据消息的实际合适，提供对应的表格定义文件。对于JSON格式的消息，如果没有提供定义文件，也可以在查询时使用Presto的JSON函数进行解析。这种处理方式虽然比较灵活，但是对会增加SQL语句的编写难度。

一个表格描述文件使用JSON来定义一个表，文件名可以任意，但是必须以 .json 作为扩展名。

```
{
    "tableName": ...,
    "schemaName": ...,
    "topicName": ...,
    "key": {
        "dataFormat": ...,
        "fields": [
            ...
        ]
    },
    "message": {
        "dataFormat": ...,
        "fields": [
            ...
       ]
```

```
    }
}
```

字段	可选性	类型	说明
tableName	必选	string	Presto表名
schemaName	可选	string	表所在的Schema名称
topicName	必选	string	Kafka topic名称
key	可选	JSON object	消息键到列的映射规则
message	可选	JSON object	消息到列的映射规则

其中，键和消息的映射规则使用如下几个字段来描述。

字段	可选性	类型	说明
dataFormat	必选	string	设置一组列的解码器
fields	必选	JSON数组	列定义列表

这里的 **fields** 是一个JSON数组，每一个元素为如下的JOSN对象：

```
{
    "name": ...,
    "type": ...,
    "dataFormat": ...,
    "mapping": ...,
    "formatHint": ...,
    "hidden": ...,
    "comment": ...
}
```

字段	可选性	类型	说明
name	必选	string	列名
type	必选	string	列的数据类型
dataFormat	可选	string	列数据解码器
mapping	可选	string	解码器参数
formatHint	可选	string	设置在该列上的提示，可以被解码器使用
hiddent	可选	boolean	是否隐藏
comment	可选	string	列的描述

- 解码器

解码器的功能就是Kafka的消息（key+message）映射到数据表到列中。如果没有表的定义文件，Presto将使用**dummy**解码器。

Kafka连接器提供了如下3中解码器：

- raw：不做转换，直接使用原始的bytes
- csv： 将消息作为CSV格式的字符串进行处理
- json：将消息作为JSON进行处理

11.8.2 产品简介

Presto是一款由FaceBook开源的一个分布式SQL-on—Hadoop分析引擎。Presto目前由开源社区和FaceBook内部工程师共同维护，并衍生出多个商业版本。

基本特性

Presto使用Java语言进行开发，具备易用、高性能、强扩展能力等特点，具体的：

- 完全支持ANSI SQL。
- 支持丰富的数据源 Presto可接入丰富的数据来源，如下所示：
  - 与Hive数仓互操作
  - Cassandra
  - Kafka
  - MongoDB
  - MySQL
  - PostgreSQL
  - SQL Server
  - Redis
  - Redshift
  - 本地文件
- 支持高级数据结构：
  - 支持数组和Map数据。
  - 支持JSON数据。
  - 支持GIS数据。
  - 支持颜色数据。

- 功能扩展能力强 Presto提供了多种扩展机制，包括：
  - 扩展数据连接器。
  - 自定义数据类型。
  - 自定义SQL函数。

  用户可以根据自身业务特点扩展相应的模块，实现高效的业务处理。
- 基于Pipeline处理模型 数据在处理过程中实时返回给用户。
- 监控接口完善：
  - 提供友好的WebUI，可视化的呈现查询任务执行过程。
  - 支持JMX协议。

应用场景

Presto是定位在数据仓库和数据分析业务的分布式SQL引擎，比较适合如下几个应用场景：

- ETL
- Ad-Hoc查询
- 海量结构化数据/半结构化数据分析
- 海量多维数据聚合/报表

特别需要注意的是，Presto是一个数仓类产品，其设计目标并不是为了替代MySQL、PostgreSQL等传统的RDBMS数据库，对事务对支持有限，不适合在线业务场景。

产品优势

EMR Presto产品除了开源Presto本身具有的优点外，还具备如下优势：

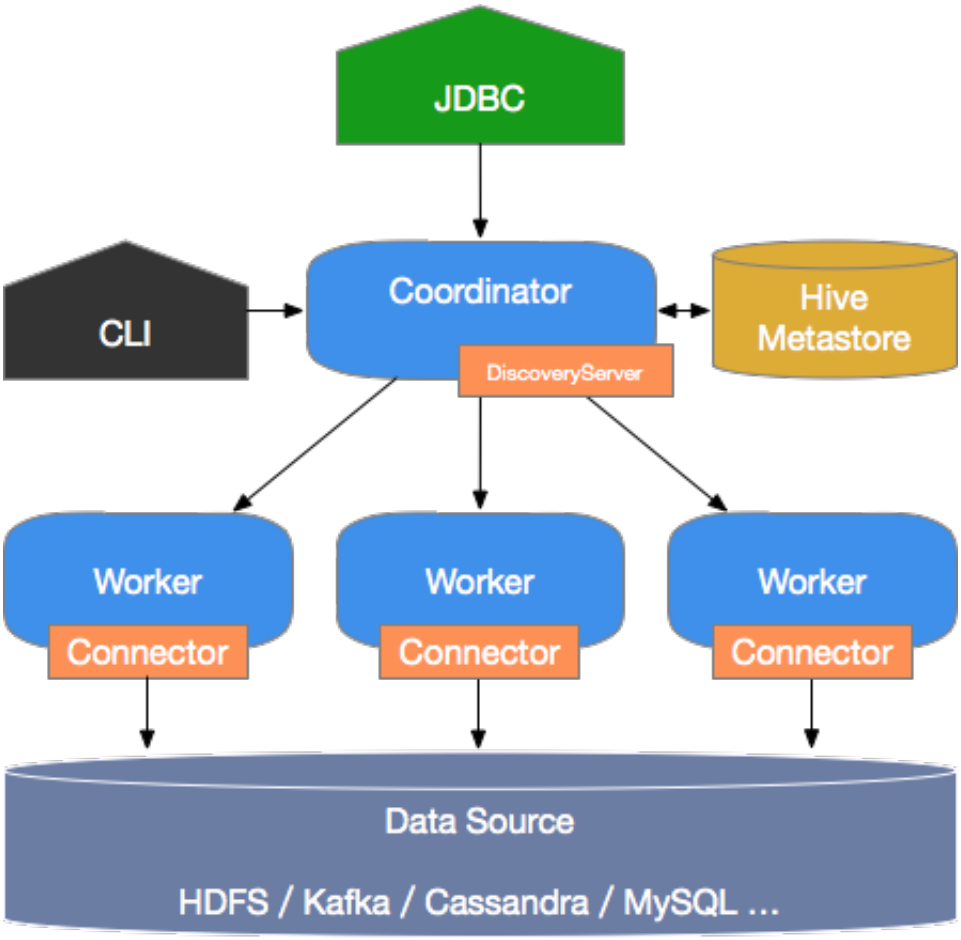
- 即买即用 分分钟完成上百节点的Presto集群搭建。
- 弹性扩容 简单操作即可完成集群的扩容和缩容。
- 与EMR软件栈完美结合，支持处理存储在OSS的数据。
- 免运维 7*24一站式服务。

11.8.3 快速入门

本章将介绍Presto数据库的基本使用方法和应用开发方法，使开发者能够快速的使用Presto数据库进行应用开发。

系统组成

Presto的系统组成如下图所示：



Presto是典型的M/S架构的系统，由一个Coordinator节点和多个Worker节点组成。Coordinator负责如下工作：

- 接收用户查询请求，解析并生成执行计划，下发Worker节点执行。
- 监控Worker节点运行状态，各个Worker节点与Coordinator节点保持心跳连接，汇报节点状态。
- 维护MetaStore数据。

Worker节点负责执行下发到任务，通过连接器读取外部存储系统到数据，进行处理，并将处理结果发送给Coordinator节点。

基本概念

本节介绍Presto中到基本概念，以便更好到理解Presto到工作机制。

- 数据模型

  数据模型即数据的组织形式。Presto使用 Catalog、Schema 和 Table 这3层结构来管理数据。

  - Catalog

    一个 Catalog 可以包含多个 Schema，物理上指向一个外部数据源，可以通过 Connector 访问改数据源。一次查询可以访问一个或多个 Catalog。

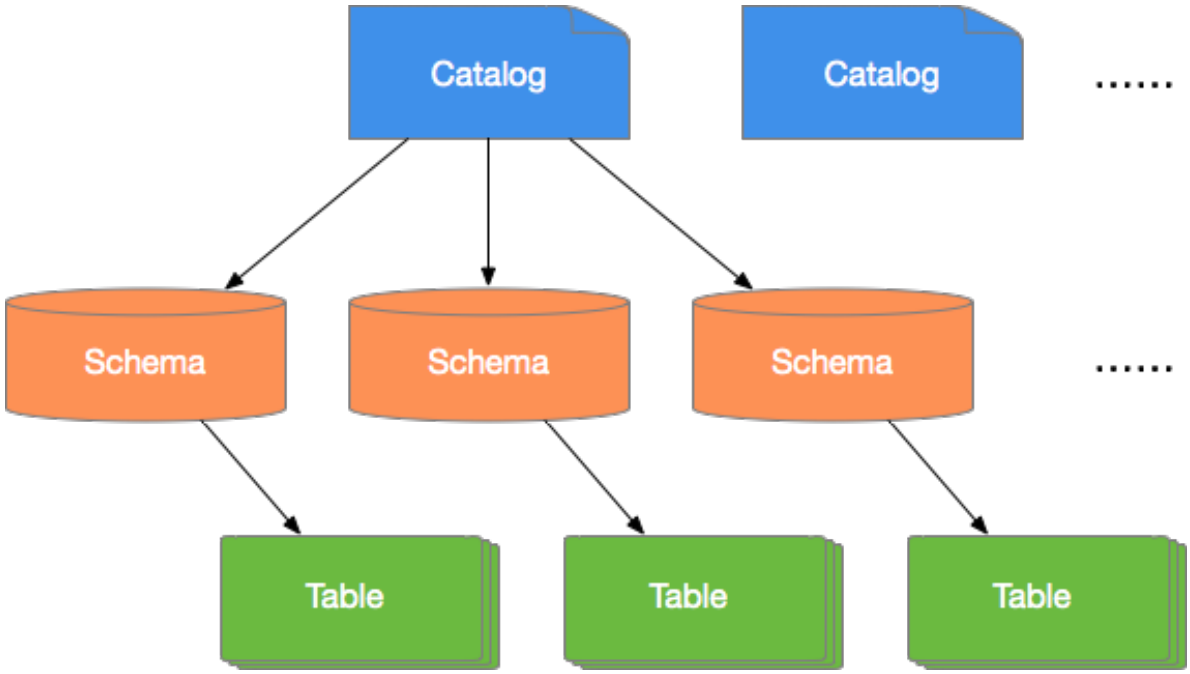
  - Schema

    相当于一个数据库示例，一个 Schema 包含多张数据表。

  - Table

    数据表，与一般意义上的数据库表相同。

  Catalog、Schema 和 Table 之间的关系如下图所示：



- Connector

  Presto通过各种 Connector 来接入多种外部数据源。Presto提供了一套标准的*SPI*接口，用户可以使用这套接口，开发自己的 Connector，以便访问自定义的数据源。

  一个 Catalog 一般会绑定一种类型的 Connector（在 Catalog 的 Properties 文件中设置）。Presto内置了多种 Connector 实现。

- 查询相关概念

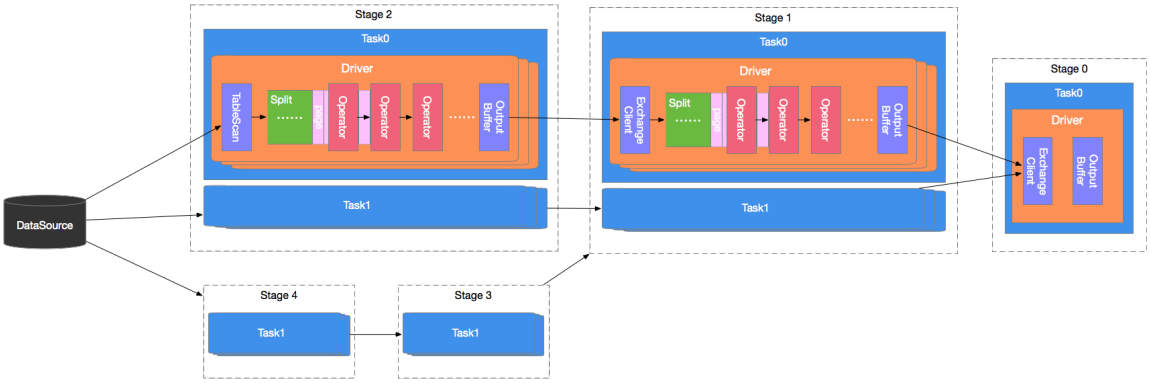
本节主要介绍Presto查询过程中的相关概念，以便用户能够更好的理解Presto查询语句执行过程和性能调优方法。

Statement

即查询语句，表示用户通过JDBC或CLI输入对SQL语句。

Query

表示一次查询的执行过程，Presto接收到一个SQL Statement 后，在Coordinator中进行解析，生成执行计划，下发到Worker中执行。一个 Query 逻辑上由 Stage，Task，Driver，Split，Operator 和 DataSource 几个组件组成，如下所示：



Stage

Presto中的一个 Query 由多个 Stage 组成，Stage 是一个逻辑概念，表示查询过程的一个阶段，包含了一个或多个执行任务（Task）。Presto采用树状结构组织 Stage，该树的Root节点为 Single Stage，该Stage会汇聚其上游Stage输出的数据，进行聚合运算，将结果直接发送给 Coordinator。该树的叶子节点为 Source Stage，该Stage从 Connector 获取数据进行处理。

Task

Task 表示一个具体的执行任务，是Presto任务调度的最小单元，执行过程中，Presto任务调度器会将这些Task调度到各个Worker上执行。一个Stage中的任务可以并行执行。两个Stage之间的任务通过 Exchange 模块传递数据。Task也是一个逻辑概念，包含了任务的参数和内容，实际的任务执行工作由 Driver 完成。

Driver

Driver 负责执行具体的任务工作。一个 Task 可以包含多个 Driver 实例，从而实现Task内部的并行处理。一个Driver 处理一个数据分片（Split）。一个 Driver 内部一组 Operator 组成，负责具体的数据操作，如转换、过滤等。

- Operator

  Operator 是最小的执行单元，负责对数据分片（Split）中的每一个 Page 进行处理，如加权、转换等，概念上与算子类似。Page 是 Operator 处理的最小数据单元，是一个列式的数据结构。一个 Page 对象由多个 Block 组成，每个 Block 代表一个字段的多行数据。一个Page最大为1MB，最多包含16*1024行数据。

- Exchange

  两个 Stage 之间通过 Exchange 模块进行数据交换。实际的数据传输过程出现在两个 Task 之间。通常，下游的 Task 会通过其上的 Exchange Client 模块从其上游 Task 的 Output Buffer 中拉去数据。拉取的数据以 Split 为单位传递给 Driver 进行处理。

命令行工具

通过SSH登入*EMR*集群，执行下面对命令进入Presto控制台：

```
$ presto --server emr-header-1:9090 --catalog hive --schema default --
user hadoop
```

高安全集群使用如下命令形式：

```
$ presto  --server https://emr-header-1:7778  \
          --enable-authentication \
          --krb5-config-path /etc/krb5.conf \
          --krb5-keytab-path  /etc/ecm/presto-conf/presto.keytab \
          --krb5-remote-service-name presto \
          --keystore-path /etc/ecm/presto-conf/keystore \
          --keystore-password 81ba14ce6084 \
          --catalog hive --schema default \
          --krb5-principal  presto/emr-header-1.cluster-XXXX@EMR.XXXX.
COM
```

- *XXXX* 为集群的 ecm id，为一串数字，可以通过 *cat /etc/hosts* 获取。
- *81ba14ce6084* 为 */etc/ecm/presto-conf/keystore* 的默认密码，建议部署后替换为自己的keystore.

接下来就可已在该控制台下执行以下命令：

```
presto:default> show schemas;
    Schema
--------------------
default
hive
```

```
information_schema
tpch_100gb_orc
tpch_10gb_orc
tpch_10tb_orc
tpch_1tb_orc
(7 rows)
```

执行 **presto --help** 命令可以获取控制台的帮助，各个参数对解释如下所示：

```
--server <server>                          # 指定Coordinator的URI
--user <user>                              # 设置用户名
--catalog <catalog>                        # 指定默认的Catalog
--schema <schema>                          # 指定默认的Schema
--execute <execute>                        # 执行一条语句，然后退出
-f <file>, --file <file>                   # 执行一个SQL文件，然后退出
--debug                                    # 显示调试信息
--client-request-timeout <timeout>         # 指定客户端超时时间，默认为2m
--enable-authentication                    # 使能客户端认证
--keystore-password <keystore password> # KeyStore密码
--keystore-path <keystore path>            # KeyStore路径
--krb5-config-path <krb5 config path>   # Kerberos配置文件路径(默认为/
etc/krb5.conf)
--krb5-credential-cache-path <path>        # Kerberos凭据缓存路径
--krb5-keytab-path <krb5 keytab path>   # Kerberos Key table路径
--krb5-principal <krb5 principal>          # 要使用的Kerberos principal
--krb5-remote-service-name <name>          # 远程Kerberos节点名称
--log-levels-file <log levels>             # 调试日志配置文件路径
--output-format <output-format>            # 批量导出的数据格式，默认为 CSV
--session <session>                        # 指定回话属性，格式如下 key=value
--socks-proxy <socks-proxy>                # 设置代理服务器
--source <source>                          # 设置查询的Source
--version                                  # 显示版本信息
-h, --help                                 # 显示帮组信息
```

使用JDBC

Java应用可以使用Presto提供的JDBC driver连接数据库进，使用方式与一般RDBMS数据库差别不大。

- 在Maven中引入

  可以在pom文件中加入如下配置引入Presto JDBC driver：

  ```
  <dependency>
      <groupId>com.facebook.presto</groupId>
      <artifactId>presto-jdbc</artifactId>
      <version>0.187</version>
  </dependency>
  ```

- Driver类名

  Presto JDBC driver类为`com.facebook.presto.jdbc.PrestoDriver`。

- 连接字串

  可以使用如下格式的连接字串：

  ```
  jdbc:presto://<COORDINATOR>:<PORT>/[CATALOG]/[SCHEMA]
  ```

  例如：

  ```
  jdbc:presto://emr-header-1:9090                # 连接数据库，使用默认的Catalog和Schema
  jdbc:presto://emr-header-1:9090/hive           # 连接数据库，使用Catalog(hive)和默认的Schema
  jdbc:presto://emr-header-1:9090/hive/default   # 连接数据库，使用Catalog(hive)和Schema(default)
  ```

- 连接参数

  Presto JDBC driver支持很多参数，这些参数既可以通过 **Properties** 对象传入，也可以通过URL参数传入，这两种方式是等价的。

  通过 **Properties** 对象传入示例：

  ```
  String url = "jdbc:presto://emr-header-1:9090/hive/default";
  Properties properties = new Properties();
  properties.setProperty("user", "hadoop");
  Connection connection = DriverManager.getConnection(url, properties);
  ......
  ```

  通过URL参数传入示例：

  ```
  String url = "jdbc:presto://emr-header-1:9090/hive/default?user=hadoop";
  Connection connection = DriverManager.getConnection(url);
  ......
  ```

  下面对各个参数进行说明：

参数名称	格式	参数说明
user	STRING	用户名
password	STRING	密码
socksProxy	\:\	SOCKS代理服务器地址，如localhost:1080
httpProxy	\:\	HTTP代理服务器地址，如localhost:8888
SSL	true\	是否使用HTTPS连接，默认为 false
SSLTrustStorePath	STRING	Java TrustStore文件路径

参数名称	格式	参数说明
SSLTrustStorePassword	STRING	Java TrustStore密码
KerberosRemoteServiceName	STRING	Kerberos服务名称
KerberosPrincipal	STRING	Kerberos principal
KerberosUseCanonicalHostname	true\	是否使用规范化主机名，默认为 false
KerberosConfigPath	STRING	Kerberos配置文件路径
KerberosKeytabPath	STRING	Kerberos KeyTab文件路径
KerberosCredentialCachePath	STRING	Kerberos credential缓存路径

- Java示例

  下面给出一个Java使用Presto JDBC driver的例子。

```
.....
// 加载JDBC Driver类
try {
    Class.forName("com.facebook.presto.jdbc.PrestoDriver");
} catch(ClassNotFoundException e) {
    LOG.ERROR("Failed to load presto jdbc driver.", e);
    System.exit(-1);
}
Connection connection = null;
Statement statement = null;
try {
    String url = "jdbc:presto://emr-header-1:9090/hive/default";
    Properties properties = new Properties();
    properties.setProperty("user", "hadoop");
    // 创建连接对象
    connection = DriverManager.getConnection(url, properties);
    // 创建Statement对象
    statement = connection.createStatement();
    // 执行查询
    ResultSet rs = statement.executeQuery("select * from t1");
    // 获取结果
    int columnNum = rs.getMetaData().getColumnCount();
    int rowIndex = 0;
    while (rs.next()) {
        rowIndex++;
        for(int i = 1; i <= columnNum; i++) {
            System.out.println("Row " + rowIndex + ", Column " + i +
 ": " + rs.getInt(i));
        }
    }
} catch(SQLException e) {
    LOG.ERROR("Exception thrown.", e);
} finally {
  // 销毁Statement对象
```

```
  if (statement != null) {
      try {
        statement.close();
    } catch(Throwable t) {
        // No-ops
    }
  }
  // 关闭连接
  if (connection != null) {
      try {
        connection.close();
    } catch(Throwable t) {
        // No-ops
    }
  }
}
```

11.8.4 数据类型

Presto默认支持多种常见的数据类型，包括布尔类型、整型、浮点型、字符串、日期等。同时，用户可以通过插件等方式增加自定义的数据类型。并且，自定义的Presto连接器不需要支持所有数据类型。

数值类型

Presto内置支持如下几种数值类型：

- BOOLEAN

  表示一个二值选项，值为 TRUE 或 FALSE。
- TINYINT

  表示一个8位有符号整型，二进制补码形式存储。
- SMALLINT

  表示一个16位有符号整型,二进制补码形式存储。
- INTEGER

  表示一个32位有符号整型,二进制补码形式存储。
- BIGINT

  表示一个64位有符号整型,二进制补码形式存储。
- REAL

  一个32位多精度的*[IEEE-754]*二进制浮点数值实现。
- DOUBLE

  一个64位多精度的*[IEEE-754]*二进制浮点数值实现。

- DECIMAL 一个固定精度的数值类型，最大可支持38位有效数字，有效数字在17位以下性能最好。定义DECIMAL 类型字段时需要确定两个字面参数：
  - 精度（precision）数值总的位数，不包括符号位。
  - 范围（scale）小数位数，可选参数，默认为0。

    示例：`DECIMAL '-10.7'` 该值可用 `DECIMAL(3,1)` 类型表示。

  下表说明整型数值类型的位宽和取值范围：

数值类型	位宽	最小值	最大值
TINYINT	8 bit	-2^7	2^7 - 1
SMALLINT	16 bit	2^15	2^15 - 1
INTEGER	32 bit	-2^31	-2^31 - 1
BIGINT	64 bit	-2^63	-2^63 - 1

字符类型

Presto内置支持如下几种字符类型：

- VARCHAR

  表示一个可变长度的字符串类型，可以设置最大长度。

  示例：`VARCHAR`，`VARCHAR(10)`
- CHAR

  表示一个固定长度的字符串类型，使用时可以指定字符串的长度，不指定则默认为 1。

  示例：`CHAR`，`CHAR(10)`

  > **说明：**
  > 指定了长度的字符串总是包含与该长度相同的字符，如果字符串字面长度小于指定长度，则不足部分会用不可见字符填充，填充部分也会参与字符比较，因此，两个字面量一样的字段，如果它们的长度定义不一样，就永远不可能相等。
- VARBINARY

  表示一块可变长度的二进制数据。

日期和时间

Presto内置支持如下几种时间和日期类型：

- DATE

  表示一个日期类型的字段，日期包括年、月、日，但是不包括时间。

  示例：`DATE '1988-01-30'`

- TIME

  表示一个时间类型，包括时、分、秒、毫秒。时间类型可以加时区进行修饰。

  示例：

  — `TIME '18:01:02.345'`，无时区定义，使用系统时区进行解析。

  — `TIME '18:01:02.345 Asia/Shanghai'`，有时区定义，使用定义的时区进行解析。

- TIMESTAMP

  表示一个时间戳类型的字段，时间戳包含了日期和时间两个部分的信息，取值范围为`'1970-01-01 00:00:01' UTC`到`'2038-01-19 03:14:07' UTC`，支持使用时区进行修饰。

  示例：`TIMESTAMP '1988-01-30 01:02:03.321'`，`TIMESTAMP '1988-01-30 01:02:03.321 Asia/Shanghai'`

- INTERVAL

  主要用于时间计算表达式中，表示一个间隔，单位可以是如下几个：

  — YEAR - 年

  — QUARTER - 季度

  — MONTH - 月

  — DAY - 天

  — HOUR - 小时

  — MINUTE - 分钟

  — SECOND - 秒

  — MILLISECOND - 毫秒

  示例：`DATE '2012-08-08' + INTERVAL '2' DAY`

复杂类型

Presto内置支持多种复杂的数据类型，以便支持更加复杂的业务场景，这些类型包括：

- JSON

表示字段为一个JSON字符串，包括JSON对象、JSON数组、JSON单值（数值或字符串），还包括布尔类型的true、false 以及表示空的 null。

示例：

- `JSON '[1, null, 1988]'`
- `JSON '{"k1":1, "k2": "abc"}'`

- ARRAY

  表示一个数组，数组中各个元素的类型必须一致。

  示例：`ARRAY[1, 2, 3]`

- MAP

  表示一个映射关系，由键数组和值数组组成。

  示例：`MAP(ARRAY['foo', 'bar'], ARRAY[1, 2])`

- ROW

  表示一行数据，行中每个列都有列名对应，可以使用`.`运算符+列名的方式来访问数据列。

  示例：`CAST(ROW(1988, 1.0, 30) AS ROW(y BIGINT, m DOUBLE, d TINYINT ))`

- IPADDRESS

  表示一个IPv4或IPv6地址。内部实现上，将IPv4处理成IPv6地址使用（*IPv4到IPv6映射表*）。

  示例：`IPADDRESS '0.0.0.0'`，`IPADDRESS '2001:db8::1'`

11.8.5 常用函数和操作符

本节介绍Presto中常用的函数和操作符。

逻辑运算符

Presto支持与、或、非3中逻辑操作符，并支持 NULL参与逻辑运算，如下所示：

```
SELECT CAST(null as boolean) AND true; --- null
SELECT CAST(null AS boolean) AND false; -- false
SELECT CAST(null AS boolean) AND CAST(null AS boolean); -- null
SELECT NOT CAST(null AS boolean); -- null
```

完整的运算真值表如下所示：

a	b	a AND b	a OR b
TRUE	TRUE	TRUE	TRUE

a	b	a AND b	a OR b
TRUE	FALSE	FALSE	TRUE
TRUE	NULL	NULL	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE
FALSE	NULL	FALSE	NULL
NULL	TRUE	NULL	TRUE
NULL	FALSE	FALSE	NULL
NULL	FALSE	NULL	NULL

另外，NOT TRUE 的结果为 FALSE，NOT FALSE 的结果为 TRUE， NOT NULL 的结果为 NULL。有关NOT运算符的详情，请参见*NOT*运算符。

比较函数和运算符

- 比较函数和运算符

  Presto支持的比较操作如下所示：

操作符	说明
<	小于
>	大于
<=	小于等于
>=	大于等于
=	等于
<>/!=	不等于
[NOT] BETWEEN	值X[不]介于min和max之间
IS [NOT] NULL	判断是否为NULL
IS [NOT] DISTINCT FROM	用于比较两个值是否一致。一般情况下，NULL 作为未定义数据存在，任何有 NULL 参与的比较都会返回 NULL。但是IS [NOT] DISTINCT FROM 将 NULL 作为值处理，返回 TRUE 或 FALSE。

- 比较函数

  Presto提供了如下几个比较操作相关的函数：

- GREATEST

  返回最大值。

  示例：`GREATEST(1, 2)`

- LEAST

  返回最小值。

  示例：`LEAST(1, 2)`

- 比较相关的修饰谓语

  Presto还提供了几个比较相关的修饰谓语，可以增强比较语义的表达能力。使用方式如下：

  ```
  <EXPRESSION> <OPERATOR> <QUANTIFIER> (<SUBQUERY>)
  ```

  例如：

  ```
  SELECT 21 < ALL (VALUES 19, 20, 21); -- false
  SELECT 42 >= SOME (SELECT 41 UNION ALL SELECT 42 UNION ALL SELECT 43
  ); -- true
  ```

  其中，ANY、ALL、SOME 是比较修饰谓语。

  - `A = ALL (…)`：A和所有值相等，则返回 TRUE。例如 `SELECT 21 = ALL (VALUES 20, 20, 20);` 返回TRUE。
  - `A <> ALL (…)`：A和所有值不相等，则返回 TRUE。例如 `SELECT 21 <> ALL (VALUES 19, 20, 22);` 返回TRUE。
  - `A < ALL (…)`：A小于所有值，则返回 TRUE。例如 `SELECT 18 < ALL (VALUES 19, 20, 22);` 返回TRUE。
  - `A = ANY (…)`：A只要等于其中一个值，则返回 TRUE，等价与 `A IN (…)`。例如 `SELECT 'hello' = ANY (VALUES 'hello', 'world');` 返回true。
  - `A <> ANY (…)`：A只要和其中一个值不相等，则返回 TRUE，等价与 `A IN (…)`。例如 `SELECT 21 <> ALL (VALUES 19, 20, 21);` 返回 TRUE。
  - `A < ANY (…)`：A只要小于其中一个值，则返回 TRUE。例如 `SELECT 21 < ALL (VALUES 19, 20, 22);` 返回 TRUE。

  ANY 和 SOME 含义相同，使用时可以互换。

条件表达式

条件表达是主要用于表达分支逻辑。Presto支持如下几种条件表达式形式：

- CASE 表达式

  在标准SQL中，CASE 表达式有两种不同的形式，如下所示：

  ```
  CASE expression
      WHEN <value|condition> THEN result
      [ WHEN ... ]
      [ ELSE result]
  END
  ```

  CASE 语句将比较 *expression* 和 *value|condition* 中的值 / 条件，如果符合（值相等或条件匹配）则返回结果。

  示例：

  ```
  --- 比较值
  SELECT a,
         CASE a
             WHEN 1 THEN 'one'
             WHEN 2 THEN 'two'
             ELSE 'many'
         END
  ```

  ```
  --- 比较条件表达式
  SELECT a, b,
         CASE
             WHEN a = 1 THEN 'aaa'
             WHEN b = 2 THEN 'bbb'
             ELSE 'ccc'
         END
  ```

- IF 函数

  IF 函数是一个简单的比较函数，用于简化二值比较逻辑的写法。其表达形式如下：

  ```
  IF(condition, true_value, [false_value])
  ```

  如果 *condition* 返回 TRUE，则函数返回 true_value，否则返回 false_value。其中，false_value 可选，不设置则返回NULL。

- COALESCE

  COALESCE 函数返回参数列表中第一个不是 NULL 的参数。其表达形式如下：

  ```
  COALESCE(value1, value2[, ...])
  ```

- NULLIF

NULLIF 函数在 value1 与 value2 相等时，返回 NULL，否则返回 value1。函数使用方法如下：

```
NULLIF(value1, value2)
```

- TRY

  TRY 函数会捕获 expression 计算过程中抛出的异常，并返回 NULL。TRY 捕获的异常包括如下几类：

  - 除零异常，如 x/0
  - 类型转换错误
  - 数值越界

  通常和 COALESCE 一起使用，以便在出错时，返回默认值。使用方法如下：

  ```
  TRY(expression)
  ```

  示例：

  ```
  --- COALESCE和TRY搭配使用，在packages=0，抛出除零异常时，返回默认值（0）。
  SELECT COALESCE(TRY(total_cost / packages), 0) AS per_package FROM
  shipping;
  per_package
  -------------
             4
            14
             0
            19
  (4 rows)
  ```

转换函数

Presto提供给了如下几个显式类型转换函数：

- CAST

  显式的进行类型转换，只是在出现转换错误的时候抛出异常。使用方法如下：

  ```
  CAST(value AS type) -> value1:type
  ```

- TRY_CAST

  与CAST功能相同，只是在出现转换错误的时候，返回NULL。使用方法如下：

  ```
  TRY_CAST(value AS TYPE) -> value1:TYPE | NULL
  ```

- TYPEOF

获取参数或表达式值的类型字符串，使用方法如下：

```
TYPEOF(expression) -> type:VARCHAR
```

示例：

```
SELECT TYPEOF(123); -- integer
SELECT TYPEOF('cat'); -- varchar(3)
SELECT TYPEOF(cos(2) + 1.5); -- double
```

数学函数与运算符

- 数学操作符

操作符	说明
+	加
-	减
*	乘
/	除（整数相除会截断）
%	取余

- 数学函数

Presto提供了十分丰富的数学函数，如下表所示：

函数	语法	说明
abs	abs(x) →	返回绝对值 x
cbrt	cbrt(x) → double	返回立方根
ceil	ceil(x)	返回比x大的最小整数，是**ceiling**的别名
ceiling	ceiling(x)	返回比x大的最小整数
cosine_similarity	cosine_similarity(x, y) → double	返回两个稀疏向量的余弦相似度
degrees	degress(x) -> double	弧度换算成角度
e	e()->double	获取欧拉常数
exp	exp(x)->double	指数函数
floor	floor(x)	获取小于x的最大整数

函数	语法	说明
from_base	from_base(string, radix) → bigint	获取字面量的值，该值的基数为radix
inverse_normal_cdf	inverse_normal_cdf(mean,sd,p)->double	计算给定正态分布（均值、标准差和累计概率）的累计分布函数的倒数
ln	ln(x)->double	返回自然对数值
log2	log2(x)->double	返回以2为底的对数
log10	log10(x)->double	返回以10为底的对数
log	log(x,b) -> double	返回以b为底数的对数
mod	mod(n,m)	返回n/m的余数
pi	pi()->double	返回常数Pi
pow	pow(x,p)->double	计算x^p的值，**power**的别名
power	power(x,p)->double	计算x^p的值
radians	radians(x)->double	将角度换算成弧度
rand	rand()->double	获取一个为随机数，返回值范围为[0.0,1.0)。**random**别名
random	random()->double	获取一个为随机数，返回值范围为[0.0,1.0)。
random	random(n)	获取一个为随机数，返回值范围为[0.0,n)。
round	round(x)	返回与x最接近的整数
round	round(x, d)	返回与x最接近的数，精确到小数后n位
sign	sign(x)	符号函数，x如果是整数，则当x=0,返回0; x>0,返回1; x<0,返回-1.x如果是浮点数，则当x为NaN时，返回Nan；当x为+∞时，返回1；当x为-∞时，返回-1.
sqrt	sqrt(x)->double	平方根函数
to_base	to_base(x, radix)->varchar	返回x以radix为基数的字面量
truncate	truncate(x) → double	截取整数部分
width_bucket	width_bucket(x, bound1, bound2, n) → bigint	获取x在[bound1, bound2]范围内n等的分直方图中的bin数
width_bucket	width_bucket(x, bins)	获取x在给定分布的直方图中的bin数

函数	语法	说明
acos	acos(x)->double	获取x的反余弦值，x为弧度
asin	asin(x)->double	获取x的反正弦值，x为弧度
atan	atan(x)->double	获取x的反正切值，x为弧度
atan2	atan2(y,x)->double	获取y/x的反正切值，x为弧度
cos	cos(x)->double	获取x的余弦值，x为弧度
cosh	cosh(x)->double	获取x的双曲余弦值，x为弧度
sin	sin(x)->double	获取x的正弦值，x为弧度
tan	tan(x)->double	获取x的正切值，x为弧度
tanh	tanh(x)->double	获取x的双曲正切值，x为弧度
infinity	infinity() → double	获取正无穷常数
is_finite	is_finite(x) → boolean	判断x是否为有限数值
is_infinite	is_infinite(x) → boolean	判断x是否为无穷数值
is_nan	is_nan(x) → boolean	判断x是否不是一个数值类型的变量
nan	nan()	获取一个表示NAN（not-a-number）的常数

位运算函数

Presto提供了如下几种位运算函数：

函数	语法	说明
bit_count	bit_count(x, bits) → bigint	返回x的补码中置1的位数
bitwise_and	bitwise_and(x, y) → bigint	位与函数
bitwise_not	bitwise_not(x) → bigint	取非操作
bitwise_or	bitwise_or(x, y) → bigint	位或函数
bitwise_xor	bitwise_xor(x, y) → bigint	抑或函数
bitwise_and_agg	bitwise_and_agg(x) → bigint	返回x中所有值的与操作结果，x为数组
bitwise_or_agg	bitwise_or_agg(x) → bigint	返回x中所有值的或操作结果，x位数组

示例：

```
SELECT bit_count(9, 64); -- 2
SELECT bit_count(9, 8); -- 2
SELECT bit_count(-7, 64); -- 62
SELECT bit_count(-7, 8); -- 6
```

Decimal函数和运算

- 字面量

  使用如下形式来表示一个类型为 DECIMAL 的值的字面量：

  ```
  DECIMAL 'xxxx.yyyyy'
  ```

  DECIMAL字面量的 *precision* 和其字面数字个数相同（包括前导的0），而 *scale* 和其小数位相同（包括后置的0），示例如下：

字面量	数据类型
DECIMAL ‘0’	DECIMAL(1)
DECIMAL ‘12345’	DECIMAL(5)
DECIMAL ‘0000012345.1234500000’	DECIMAL(20, 10)

- 运算符

  算术运算符

  假设有如下两个 DECIMAL 类型的变量 x，y：

  - x ： DECIMAL(xp,xs)
  - y ： DECIMAL(yp,ps)

  两个变量在参与算术运算时，遵循如下规则：

  - x + y 或 x - y
    - precision = min(38, 1 + min(xs, ys) + min(xp-xs, yp-ys))
    - scale = max(xs, ys)
  - x * y
    - precision = min(38, xp + yp)
    - scale = xs + ys
  - x / y

- precision = min(38, xp + ys + max(0, ys-xs))
- scale = max(xs, ys)

- x % y
  - precision = min(xp - xs, yp - ys) + max(xs, bs)
  - scale = max(xs, ys)
- 比较运算符

  DECIMAL 可以使用标准比较运算符和 BETWEEN 进行比较运算。
- 一元运算符

  DECIMAL 可以使用一元运算符 - 取负数。

字符函数和运算

- 拼接运算符

  使用 `||` 运算符实现字符串的拼接。
- 字符函数

  下表列出了Presto支持的字符函数：

函数名	语法	说明
chr	chr(n) → varchar	返回UCP(Unicode code point) **n** 作为一个字符返回。
codepoint	codepoint(string) → integer	返回字符 **`string`** 的UCP值
concat	concat(string1, ⋯, stringN) → varchar	连接字符串，功能同运算符
hamming_distance	hamming_distance(string1, string2) → bigint	返回两个字符串之间的汉明距离.注意，两个字符串的长度应该相等。
length	length(string) → bigint	返回字符串的长度
levenshtein_distance	levenshtein_distance(string1, string2) → bigint	返回两个字符串之间的*Levenshtein*编辑距离
lower	lower(string) → varchar	转成小写
upper	upper(string) → varchar	转成大写
replace	replace(string, search) → varchar	用空字符来替换字符串 **`string`** 中和 **`search`** 相同的子串

函数名	语法	说明
replace	replace(string, search, replace) → varchar	用 **replace** 来替换字符串 **string** 中和 **search** 相同的子串
reverse	reverse(string) → varchar	反转字符串
lpad	lpad(string, size, padstring) → varchar	用 **padstring** 从左边开始填充字符串 **string**，填充成长度为 **size** 的字符串。**padstring** 不能为空，**size** 不能为0。
rpad	rpad(string, size, padstring) → varchar	用 **padstring** 从右边开始填充字符串 **string**，填充成长度为 **size** 的字符串。**padstring** 不能为空，**size** 不能为0。
ltrim	ltrim(string) → varchar	删除前置空白符
rtrim	rtrim(string) → varchar	删除后置空白符
split	split(string, delimiter) → array	拆分字符串
split	split(string, delimiter, limit) → array	拆分字符串，对数组大小有限制
split_part	split_part(string, delimiter, index) → varchar	从 **index** 处开始拆分字符串，index从1开始。
split_to_map	split_to_map(string, entryDelimiter, keyValueDelimiter) → map	将字符串拆成一个map
strpos	strpos(string, substring) → bigint	返回第一个符合子串的位置序号，序号从1开始。没找到则返回0。
position	position(substring IN string) → bigint	返回子串在给定字符串中的起始位置
substr	substr(string, start, [length]) → varchar	截取从位置 **start** 位置开始的子串。位置序号从1开始。长度参数可选。

- Unicode相关的函数
  - normalize(string) → varchar

    转换成*NFC*标准形式的字符串。
  - normalize(string, form) → varchar

    按给定的格式归一化字符串，**form** 可选如下：
    - NFD Canonical Decomposition

- NFC Canonical Decomposition, followed by Canonical Composition
- NFKD Compatibility Decomposition
- NFKC Compatibility Decomposition, followed by Canonical Composition

— to_utf8(string) → varbinary

转换成UTF8格式的字符串。

— from_utf8(binary, [replace]) → varchar

将二进制数据解析成UTF-8字符串。非法序列会用 **replace** 替代，该项参数可选，默认为Unicode字符 **U+FFFD**。需要注意的是，**replace** 必须是单个字符，可以为空字符。

正则表达式函数

Presto使用*Java Pattern*的正则表达式语法。但也有几个例外，如下所示：

- 多行模式
  - 使用?m开启多行模式
  - 行终止符为\n
  - 不支持?d选项
- 大小写敏感模式
  - 使用?i开启
  - 不支持?u选项
  - 不支持上下文敏感匹配
  - 不支持局部敏感匹配
- 不支持Surrogate pairs

  如Unicode U+10000必须使用\x{10000}来表示，而不能用\uD800\uDC00来表示。
- 如果模式字符串中不包含基本字符，并且没有间隔，那么，使用边界字符 \b会出错。
- 不支持在字符类（如[A-Z123]）中使用\Q和\E。
- 在Unicode字符类（\p{prop}）的支持上存在如下不同：
  - 不支持下划线，如使用OldItalic代替Old_Italic。
  - 不支持使用Is，script=，sc= 来指定脚本，取而代之的是直接使用脚本名。如\p{Hiragana}，而不是\p{script=Hiragana}。
  - 不支持使用block=，blk=来表示区块，只能使用 In。如\p{InMongolia}。

- 不支持使用Is，general_category=，gc=来指定分类，直接使用类别名称，如\p{L}。
- 直接使用二进制属性，如使用\p{NoncharacterCodePoint}而不是\p{IsNoncharacterCodePoint}。

下面介绍Presto提供的正则表达式函数：

- regexp_extract_all(string, pattern, [group]) → array

  提取字符串**string**中所有与模式**pattern**匹配的子串。**pattern**中如果使用了分组的功能，则可以通过设置**group**参数，用于说明匹配哪个捕获组。

  示例

```
SELECT regexp_extract_all('1a 2b 14m', '\d+'); -- [1, 2, 14]
SELECT regexp_extract_all('1a 2b 14m', '(\d+)([a-z]+)', 2); -- ['a',
 'b', 'm']
```

- regexp_extract(string, pattern, [group]) → varchar

  功能和用法与regexp_extract_all类似，只是本函数只提取第一个匹配的结果。

  示例

```
SELECT regexp_extract_all('1a 2b 14m', '\d+'); -- [1, 2, 14]
SELECT regexp_extract_all('1a 2b 14m', '(\d+)([a-z]+)', 2); -- ['a',
 'b', 'm']
```

- regexp_extract_all(string, pattern, [group]) → array

  提取字符串**string**中所有与模式**pattern**匹配的子串。**pattern**中如果使用了分组的功能，则可以通过设置**group**参数，用于说明匹配哪个捕获组。

  示例

```
SELECT regexp_extract('1a 2b 14m', '\d+'); -- 1
SELECT regexp_extract('1a 2b 14m', '(\d+)([a-z]+)', 2); -- 'a'
```

- regexp_like(string, pattern) → boolean

  判断字符串**string**中是否包含符合**pattern**模式的子串，包含返回TRUE，否则返回FALSE。本函数的功能与SQL中的 LIKE 语句功能相似，不同的是LIKE需要匹配整个模式字串，而本函数只需要字串中包含与模式字串相匹配子串即返回 TRUE。

  示例

```
SELECT regexp_like('1a 2b 14m', '\d+b'); -- true
```

- regexp_replace(string, pattern, [replacement]) → varchar

用**replacement**替换字符串**string**中所有符合模式**pattern**的子串。**replacement**可选，不设置将使用空字符‘’替换（即删除匹配的子串）。

可以通过在**replacement**字串中使用*$g*（g为捕获组的序号，从1开始）或*${*组名称*}*来设置捕获组。美元符号$在**replacement**字串中需要使用\$进行转义。

示例

```
SELECT regexp_replace('1a 2b 14m', '\d+[ab] '); -- '14m'
SELECT regexp_replace('1a 2b 14m', '(\d+)([ab]) ', '3c$2 '); -- '3ca
 3cb 14m'
```

- regexp_split(string, pattern) → array

  使用模式字串**pattern**拆分字符串，保留尾部的空字符串。

  示例

```
SELECT regexp_split('1a 2b 14m', '\s*[a-z]+\s*'); -- ['1', '2', '14
', ''] 4个元素
                                                    -- 最后一个为空字符
```

二进制函数

- 拼接运算符

  使用||运算符实现二进制串的拼接。

- 二进制函数

函数	语法	说明
length	length(binary) → bigint	返回二进制块的字节长度
concat	concat(binary1, …, binaryN) → varbinary	将多个二进制块拼接在一起
to_base64	to_base64(binary) → varchar	获取二进制块的base64编码
from_base64	from_base64(string) → varbinary	base64解码
to_base64url	to_base64url(binary) → varchar	使用URL安全字符进行base64编码
from_base64url	from_base64url(string) → varbinary	使用URL安全字符进行base64解码
to_hex	to_hex(binary) → varchar	将二进制块编码为16进制字符串

函数	语法	说明
from_hex	from_hex(string) → varbinary	将16进制编码的字符串解码成二进制块
to_big_endian_64	to_big_endian_64(bigint) → varbinary	将bigint编码为64位大端补码格式
from_big_endian_64	from_big_endian_64(binary) → bigint	64位大端补码格式的二进制解码位bigint类型的数字
to_ieee754_32	to_ieee754_32(real) → varbinary	根据*IEEE 754*算法，将单精度浮点数编码为一个32位大端字节序的二进制块
to_ieee754_64	to_ieee754_64(double) → varbinary	根据*IEEE 754*算法，将双精度浮点数编码为一个64位大端字节序的二进制块
crc32	crc32(binary) → bigint	计算二进制块的CRC32值
md5	md5(binary) → varbinary	计算二进制块的MD5哈希值
sha1	sha1(binary) → varbinary	计算二进制块的SHA1哈希值
sha256	sha256(binary) → varbinary	计算二进制块的SHA256哈希值
sha512	sha512(binary) → varbinary	计算二进制块的SHA512哈希值
xxhash64	xxhash64(binary) → varbinary	计算二进制块的XXHASH64值

日期时间函数

- 日期和时间操作符

  Presto支持两种日期时间操作符`+`和`-`。

  示例

```
--- +
 date '2012-08-08' + interval '2' day                --- 2012-08-10
 time '01:00' + interval '3' hour                    --- 04:00:00.
000
 timestamp '2012-08-08 01:00' + interval '29' hour    --- 2012-08-09
 06:00:00.000
 timestamp '2012-10-31 01:00' + interval '1' month    --- 2012-11-30
 01:00:00.000
 interval '2' day + interval '3' hour                --- 2 03:00:00.
000
 interval '3' year + interval '5' month                --- 3-5
 --- -
 date '2012-08-08' - interval '2' day                --- 2012-08-06
 time '01:00' - interval '3' hour                    --- 22:00:00.
000
 timestamp '2012-08-08 01:00' - interval '29' hour    --- 2012-08-06
 20:00:00.000
 timestamp '2012-10-31 01:00' - interval '1' month    --- 2012-09-30
 01:00:00.000
```

```
 interval '2' day - interval '3' hour                 --- 1 21:00:00.
000
 interval '3' year - interval '5'                    --- month    2-7
```

- 时区转换

  使用AT TIME ZONE操作符，可以实现时区转换。

  示例

```
SELECT timestamp '2012-10-31 01:00 UTC';
 --- 2012-10-31 01:00:00.000 UTC
 SELECT timestamp '2012-10-31 01:00 UTC' AT TIME ZONE 'America/
Los_Angeles';
 --- 2012-10-30 18:00:00.000 America/Los_Angeles
```

- 时间和日期函数

  ▬ 基本函数

函数	语法	说明
current_date	current_date -> date	返回当前(查询启动时)日期
current_time	current_time -> time with time zone	返回当前时间
current_timestamp	current_timestamp -> timestamp with time zone	返回当前时戳
current_timezone	current_timezone() → varchar	返回当前时区
date	date(x) → date	将日期字面量转换成日期类型的变量
from_iso8601_timestamp	from_iso8601_timestamp(string) → timestamp with time zone	将ISO8601格式的时戳字面量转换成带时区的时戳变量
from_iso8601_date	from_iso8601_date(string) → date	将ISO8601格式的日期字面量转换成日期类型的变量
from_unixtime	from_unixtime(unixtime, [timezone_str]) → timestamp	将UNIX时戳转换成时戳变量，可以带时区选项。
from_unixtime	from_unixtime(unixtime, hours, minutes) → timestamp with time zone	将UNIX时戳转换成带时区的时戳变量。**hours**和**minutes**表示时区偏移量。
localtime	localtime -> time	获取当前时间
localtimestamp	localtimestamp -> timestamp	获取当前时戳

函数	语法	说明
now	now() → timestamp with time zone	获取当前时间，**current_time**的别名
to_iso8601	to_iso8601(x) → varchar	将**x**转换成ISO8601格式的字符串。这里**x**可以是DATE、TIMESTAMP [with time zone]这几个类型。
to_milliseconds	to_milliseconds(interval) → bigint	获取当前距当天零时已经过去的毫秒数
to_unixtime	to_unixtime(timestamp) → double	将时间戳转换成UNIX时间

说明：

使用下列SQL标准函数时，不用使用圆括号。

- current_data
- current_time
- current_timestamp
- localtime
- localtimestamp

- 截断函数

截断函数将时间日期变量按给定的单位进行截取，返回该单位的时间日期值。使用方法如下：

```
date_trunc(unit, x) -> [与x相同类型的变量]
```

其中，**unit** 可以选如下几个值：

- second：截取到秒
- minute： 截取到分钟
- hour： 截取到小时
- day： 截取到天
- week： 截取到星期
- month： 截取到月份
- quarter： 截取到季度

- `year`： 截取到年

— 时间间隔函数

Presto提供两个函数用于时间间隔计算，分别是：

- date_add(unit, value, timestamp) → [same as input]

  计算给定增量的时间戳，增量可以为负，带单位。

- date_diff(unit, timestamp1, timestamp2) → bigint

  计算两个时间戳的时间间隔，带单位。

  上面两个函数中，**`unit`** 可以选如下几个值：

  - `ns`：纳秒
  - `us`：微秒
  - `ms`：毫秒
  - `s`：秒
  - `m`：分
  - `h`：小时
  - `d`：天

— 日期时间域提取函数

Presto提供了从一个日期变量中提取指定域的函数 `extract`，具体如下：

extract(field FROM x) → bigint

其中，**`x`**为要提取的日期时间变量，**`field`**为要提取的域，可取如下列表中的值：

- `YEAR`： 年
- `QUARTER`： 季度
- `MONTH`： 季度
- `WEEK`： 星期
- `DAY`： 天
- `DAY_OF_MONTH`： 一个月中的第几天
- `DAY_OF_WEEK`： 一星期中的第几天
- `DOW`： 同DAY_OF_WEEK
- `DAY_OF_YEAR`： 一年中的第几天

- DOY： 同DAY_OF_YEAR
- YEAR_OF_WEEK： *ISO Week*中的年份
- YOW： 同 YEAR_OF_WEEK
- HOUR： 小时
- MINUTE： 分钟
- SECOND： 秒
- TIMEZONE_HOUR： 小时，带时区
- TIMEZONE_MINUTE： 分钟，带时区

为了方便使用，Presto提供了如下辅助函数：

函数	语法	说明
day	day(x) → bigint	返回当前日期是一个月中的第几天
day_of_month	day_of_month(x) → bigint	同`day`
day_of_week	day_of_week(x) → bigint	返回当前日期是一天中的第几天
day_of_year	day_of_year(x) → bigint	返回当前时间是一年中的第几天
dow	dow(x) → bigint	同`day_of_week`
doy	doy(x) → bigint	同`day_of_year`
hour	hour(x) → bigint	返回给定时间的小时数，取值范围为[0, 23]
minute	minute(x) → bigint	返回给定时间的分钟数，取之范围为[0, 59]
month	month(x) → bigint	返回给定时间的月份，取值范围为[1, 12]
quarter	quarter(x) → bigint	返回给定时间所属的季度
second	second(x) → bigint	返回给定时间的秒数，取之范围为[0, 59]
timezone_hour	timezone_hour(timestamp) → bigint	返回时区偏移量，单位小时
timezone_minute	timezone_minute(timestamp) → bigint	返回时区偏移量，单位分钟
week	week(x) → bigint	返回一年中的第几个信息，取值范围[1, 53]
week_of_year	week_of_year(x) → bigint	同`week`
year	year(x) → bigint	返回给定时间的年份值

函数	语法	说明
year_of_week	year_of_week(x) → bigint	返回 **x** (*ISO Week*格式)中的年份
yow	yow(x) → bigint	同 `year_of_week`

- MySQL日期函数

Presto提供了两个日期解析相关的函数，用于兼容MySQL的日期函数`date_parse`和`str_to_date`，它们分别是：

- date_format(timestamp, format) → varchar

  使用**`format`**格式化**`timestamp`**。

- date_parse(string, format) → timestamp

  按**`format`**格式解析日期字面量。

Presto支持的MySQL格式符号如下表所示：

符号	说明
%a	星期简写 (Sun .. Sat)
%b	月份简写 (Jan .. Dec)
%c	月份，数字(1 .. 12)，不可以为0
%d	月中天数，数字(01 .. 31)，不可以为0
%e	月中天数，数字 (1 .. 31)，不可以为0
%f	秒数 (6 digits for printing: 000000 .. 999000; 1 - 9 digits for parsing: 0 .. 999999999)
%H	小时 (00 .. 23)
%h	小时 (01 .. 12)
%I	小时 (01 .. 12)
%i	分钟 (00 .. 59)
%j	一年中的第几天 (001 .. 366)
%k	小时 (0 .. 23)
%l	小时 (1 .. 12)
%M	月份名称 (January .. December)

符号	说明
%m	月份，数字 (01 .. 12) [4]
%p	AM / PM
%r	时间，12小时 (hh:mm:ss AM/PM)
%S	秒 (00 .. 59)
%s	秒 (00 .. 59)
%T	时间，24hour (hh:mm:ss)
%v	星期 (01 .. 53), 第一条为星期一，与%X配合使用
%W	星期名称 (Sunday .. Saturday)
%x	年份，数字，4位，第一天为星期一
%Y	年份，数字，4位
%y	年份，数字，2位, 表示年份范围为[1970, 2069]
%%	表示字符‘%’

说明：

Presto目前不支持的符号有： %D、%U、%u、%V、%w、%X

- Java日期函数

  下列函数用于兼容Java的日期格式（*JodaTime Pattern*）：

  - format_datetime(timestamp, format) → varchar： 格式化时间戳
  - parse_datetime(string, format) → timestamp with time zone： 解析时间戳字符串

聚合函数

聚合函数具有如下特点：

- 输入一个数据集
- 输出一个单一的计算结果

绝大部分聚合函数都会在计算时忽略null值，并且在输入为空或均为null时，返回null。但也有例外，如下几个聚合函数：

- count
- count_if

- max_by
- min_by
- approx_distinct
- 基本聚合函数

函数	语法	说明
arbitrary	arbitrary(x) → [same as input]	随机返回x中的一个非null值
array_agg	array_agg(x) → array<[same as input]>	从输入的元素中创建数组
avg	avg(x) → double	求算术平均值
avg	avg(time interval type) → time interval type	计算输入时间序列的平均时间间隔
bool_and	bool_and(boolean) → boolean	如果所有输入的值都为TRUE，则返回TRUE，否则返回FALSE
bool_or	bool_or(boolean) → boolean	如果输入的序列中有一个为TRUE，则返回TRUE，否则返回FALSE
checksum	checksum(x) → varbinary	返回x的校验和（顺序不敏感）
count	count(*) → bigint	返回行数
count	count(x) → bigint	返回非null元素的个数
count_if	count_if(x) → bigint	返回x中元素为true的个数，等同于`count(CASE WHEN x THEN 1 END)`
every	every(boolean) → boolean	同**bool_and**
geometric_mean	geometric_mean(x) → double	返回x的几何平均值
max_by	max_by(x, y) → [same as x]	返回与y的最大值相关的x值
max_by	max_by(x, y, n) → array<[same as x]>	返回与y的前n个最大值相关的x值的数组
min_by	min_by(x, y) → [same as x]	返回与y的最小值相关的x值
min_by	min_by(x, y, n) → array<[same as x]>	返回与y的前n个最小值相关的x值的数组
max	max(x) → [same as input]	返回最大值

函数	语法	说明
max	max(x, n) → array<[same as x]>	返回前n个最大值列表
min	min(x) → [same as input]	返回最小值
min	min(x, n) → array<[same as x]>	返回前n个最小值列表
sum	sum(x) → [same as input]	求和

- 位聚合函数

  位聚合函数参见《位运算函数》中介绍的`bitwise_and_agg`和`bitwise_or_agg`函数。

- Map聚合函数

函数	语法	说明
histogram	histogram(x) → map	统计直方图。
map_agg	map_agg(key, value) → map	创建一个 MAP 类型的变量。
map_union	map_union(x) → map	返回输入map列表的Union结果，如果有多个map对象包含相同的key，最终的结果中，对于key的value随机的从输入的map中选取。
multimap_agg	multimap_agg(key, value) → map>	创建一个多重映射的 MAP 变量。

- 近似聚合函数

函数	语法	说明
approx_distinct	approx_distinct(x, [e]) → bigint	返回输入列表中不重复的值的个数。本函数返回的是`count(DISTINCT x)`的近似值，如果所有值都是null，则返回0。 **e**为期望标准差的上界，可选，默认为2.3%，当前的实现方式对**e**的取值范围有约束，要求在[0.01150, 0.26000]之间。对于特定对输入，不保证误差上界。
approx_percentile	approx_percentile(x, percentage) → [same as x]	估计序列x中位于第百分之percentage位的数值。
approx_percentile	approx_percentile(x, percentages) → array<[same as x]>	类似上面，percentages为数组，返回值与之一一对应。

函数	语法	说明
approx_percentile	approx_percentile(x, w, percentage) → [same as x]	类似上面，w为x的权值。
approx_percentile	approx_percentile(x, w, percentage, accuracy) → [same as x]	类似上面，**accuracy**为预估精度的上线，取值范围为[0, 1]。
approx_percentile	approx_percentile(x, w, percentages) → array<[same as x]>	类似上面，percentages为数组，返回值与之一一对应。
numeric_histogram	numeric_histogram(buckets, value, [weight]) → map	按给定的桶数计算数值直方图。**buckets**必须是 BIGINT 类型， **value**和**weight**必须是数值类型。权重列表**weight**可选，默认为1。

- 统计聚合函数

函数	语法	说明
corr	corr(y, x) → double	计算相关系数
covar_pop	covar_pop(y, x) → double	计算总体协方差
covar_samp	covar_samp(y, x) → double	计算样本协方差
kurtosis	kurtosis(x) → double	计算超值峰度。 使用下列表达式进行无偏估计：kurtosis(x) = n(n+1)/((n-1)(n-2)(n-3))sum[(x_i-mean)^4]/sttdev(x)^4-3(n-1)^2/((n-2)(n-3))
regr_intercept	regr_intercept(y, x) → double	计算线性回归截距。 *y*为相关变量，*x*为独立变量。
regr_slope	regr_slope(y, x) → double	计算线性回归斜率。*y*为相关变量， *x*为独立变量。
skewness	skewness(x) → double	计算偏度。
sttdev_pop	sttdev_pop(x) → double	计算总体标准差。
sttdev_samp	sttdev_samp(x) → double	计算样本标准差。
sttdev	sttdev(x) → double	计算标准差，同`sttdev_samp`。
var_pop	var_pop(x) → double	计算总体方差。
var_samp	var_samp(x) → double	计算样本方差。
variance	variance(x) → double	同`var_samp`。

11.8.6 SQL语句

本文介绍Presto使用中一些常见的SQL语句以及相关示例。

ALTER SCHEMA

- 概要

```
ALTER SCHEMA name RENAME TO new_name
```

- 描述

  重命名SCHEMA。

- 示例

```
ALTER SCHEMA web RENAME TO traffic -- 将Schema 'web'重命名为'traffic'
```

ALTER TABLE

- 概要

```
ALTER TABLE name RENAME TO new_name
ALTER TABLE name ADD COLUMN column_name data_type
ALTER TABLE name DROP COLUMN column_name
ALTER TABLE name RENAME COLUMN column_name TO new_column_name
```

- 描述

  更新表格定义。

- 示例

```
ALTER TABLE users RENAME TO people; --- 重命名
ALTER TABLE users ADD COLUMN zip varchar; --- 添加列
ALTER TABLE users DROP COLUMN zip; --- 删除列
ALTER TABLE users RENAME COLUMN id TO user_id; --- 重命名列
```

CALL

- 概要

```
CALL procedure_name ( [ name => ] expression [, ...] )
```

- 描述

  调用存储过程。存储过程可以由连接器提供，用于数据操作或管理任务。如果底层存储系统具有自己的存储过程框架，如PostgreSQL，则需要通过连接器提供的存储过程来访问这些存储系统自己的存储过程,而不能使用**CALL**直接访问。

- 示例

```
CALL test(123, 'apple'); --- 调用存储过程并传参（参数位置）
CALL test(name => 'apple', id => 123); --- 调用存储过程并传参（命名参数）
CALL catalog.schema.test(); --- 调用权限定名称的存储过程
```

COMMIT

- 概要

```
COMMIT [WORK]
```

- 描述

  提交当前事务。

- 示例

```
COMMIT;
COMMIT WORK;
```

CREATE SCHEMA

- 概要

```
CREATE SCHEMA [ IF NOT EXISTS ] schema_name
[ WITH ( property_name = expression [, ...] ) ]
```

- 描述

  创建新的SCHEMA。Schema是管理数据库表、视图和其他数据库对象的容器。

  - 如果SCHEMA已经存在，使用 `IF NOT EXISTS` 子句可以避免抛错。
  - 使用 `WITH` 子句可以在创建时为SCHEMA设置属性。可以通过下列查询语句获取所有支持的属性列表：

    ```
    SELECT * FROM system.metadata.schema_properties;
    ```

- 示例

```
CREATE SCHEMA web;
CREATE SCHEMA hive.sales;
CREATE SCHEMA IF NOT EXISTS traffic;
```

CREATE TABLE

- 概要

```
CREATE TABLE [ IF NOT EXISTS ]
table_name (
  { column_name data_type [ COMMENT comment ]
```

```
  | LIKE existing_table_name [ { INCLUDING | EXCLUDING } PROPERTIES
 ] }
  [, ...]
)
[ COMMENT table_comment ]
[ WITH ( property_name = expression [, ...] ) ]
```

- 描述

  创建空表。可以使用CREATE TABLE AS从已有数据集中创建表。

  - 使用IF NOT EXISTS子句可以避免在表存在时抛出异常。
  - 使用WITH子句可以在创建表格时，为表格设置属性。支持的属性列表可以通过下列语句获取：

    ```
    SELECT * FROM system.metadata.table_properties;
    ```

  - 使用LIKE子句可以引用已经存在的表的列定义。支持使用多个LIKE子句。
  - 使用INCLUDING PROPERTIES可以在创建表时，引用已有表的属性。如果同时还使用了WITH子句，WITH子句中指定的属性会覆盖INCLUDING PROPERTIES引入的同名属性。默认为EXCLUDING PROPERTIES。

- 示例

```
--- 创建表orders
CREATE TABLE orders (
  orderkey bigint,
  orderstatus varchar,
  totalprice double,
  orderdate date
)
WITH (format = 'ORC')
--- 创建表orders，带备注信息
CREATE TABLE IF NOT EXISTS orders (
  orderkey bigint,
  orderstatus varchar,
  totalprice double COMMENT 'Price in cents.',
  orderdate date
)
COMMENT 'A table to keep track of orders.'
--- 创建表bigger_orders， 其中部分列的定义引用自表orders
CREATE TABLE bigger_orders (
  another_orderkey bigint,
  LIKE orders,
  another_orderdate date
)
```

CREATE TABLE AS

- 概要

```
CREATE TABLE [ IF NOT EXISTS ] table_name [ ( column_alias, ... ) ]
[ COMMENT table_comment ]
```

```
[ WITH ( property_name = expression [, ...] ) ]
AS query
[ WITH [ NO ] DATA ]
```

- 描述

  创建新表，表中的数据来自 `SELECT` 子句。

  - 使用 `IF NOT EXISTS` 子句可以避免在表存在时抛出异常；
  - 使用 `WITH` 子句可以在创建表格时，为表格设置属性。支持的属性列表可以通过下列语句获取：

    ```
    SELECT * FROM system.metadata.table_properties;
    ```

- 示例

```
--- 从表orders中选择两个列来创建新的表
CREATE TABLE orders_column_aliased (order_date, total_price)
AS
SELECT orderdate, totalprice
FROM orders
--- 创建新表，使用聚合函数
CREATE TABLE orders_by_date
COMMENT 'Summary of orders by date'
WITH (format = 'ORC')
AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate
--- 创建新表，使用**IF NOT EXISTS**子句
CREATE TABLE IF NOT EXISTS orders_by_date AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate
--- 创建新表，schema和表nation一样，但是没有数据
CREATE TABLE empty_nation AS
SELECT *
FROM nation
WITH NO DATA
```

CREATE VIEW

- 概要

```
CREATE [ OR REPLACE ] VIEW view_name AS query
```

- 描述

  创建视图。视图是一个逻辑表，不包含实际数据，可以在查询中被引用。每次查询引用视图时，定义视图的查询语句都会被执行一次。

  创建视图时使用 `OR REPLACE` 可以避免在视图存在时抛出异常。

- 示例

```
--- 创建一个简单的视图
CREATE VIEW test AS
SELECT orderkey, orderstatus, totalprice / 2 AS half
FROM orders
--- 创建视图，使用聚合函数
CREATE VIEW orders_by_date AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate
--- 创建视图，如果视图已经存在，就替换它
CREATE OR REPLACE VIEW test AS
SELECT orderkey, orderstatus, totalprice / 4 AS quarter
FROM orders
```

DEALLOCATE PREPARE

- 概要

```
DEALLOCATE PREPARE statement_name
```

- 描述

  从会话中删除一个命名Statement。

- 示例

```
--- 释放名为my_query的查询
DEALLOCATE PREPARE my_query;
```

DELETE

- 概要

```
DELETE FROM table_name [ WHERE condition ]
```

- 描述

  删除表中与 `WHERE` 子句匹配的行，如果没有指定 `WHERE` 子句，将删除所有行。

- 示例

```
--- 删除匹配行
DELETE FROM lineitem WHERE shipmode = 'AIR';
--- 删除匹配行
DELETE FROM lineitem
WHERE orderkey IN (SELECT orderkey FROM orders WHERE priority = 'LOW
');
--- 清空表
DELETE FROM orders;
```

- 局限

部分连接器没有可能不支持 `DELETE`。

DESCRIBE

- 概要

```
DESCRIBE table_name
```

- 描述

获取表格定义信息，等同与*SHOW COLUMNS*。

- 示例

```
DESCRIBE orders;
```

DESCRIBE INPUT

- 概要

```
DESCRIBE INPUT statement_name
```

- 描述

列出一个预编译查询中各个参数的位置及其类型。

- 示例

```
--- 创建一个预编译查询'my_select1'
PREPARE my_select1 FROM
SELECT ? FROM nation WHERE regionkey = ? AND name < ?;
--- 获取该预编译查询的描述信息
DESCRIBE INPUT my_select1;
```

查询结果：

```
Position | Type
--------------------
        0 | unknown
        1 | bigint
        2 | varchar
(3 rows)
```

DESCRIBE OUTPUT

- 概要

```
DESCRIBE OUTPUT statement_name
```

- 描述

列出输出结果的所有列信息，包括列名（或别名）、Catalog、Schema、表名、类型、类型大小（单位字节）和一个标识位，说明该列是否为别名。

- 示例

  - 示例1

    准备一个预编译查询：

    ```
    PREPARE my_select1 FROM
    SELECT * FROM nation;
    ```

    执行 `DESCRIBE OUTPUT`，输出：

    ```
    DESCRIBE OUTPUT my_select1;
     Column Name | Catalog | Schema | Table  |  Type   | Type Size |
    Aliased
    -------------+---------+--------+--------+---------+-----------
    +---------
     nationkey   | tpch    | sf1    | nation | bigint  |         8 |
    false
     name        | tpch    | sf1    | nation | varchar |         0 |
    false
     regionkey   | tpch    | sf1    | nation | bigint  |         8 |
    false
     comment     | tpch    | sf1    | nation | varchar |         0 |
    false
    (4 rows)
    ```

  - 示例2

    ```
    PREPARE my_select2 FROM
    SELECT count(*) as my_count, 1+2 FROM nation
    ```

    执行 `DESCRIBE OUTPUT`，输出：

    ```
    DESCRIBE OUTPUT my_select2;
     Column Name | Catalog | Schema | Table |  Type  | Type Size |
    Aliased
    -------------+---------+--------+-------+--------+-----------
    +---------
     my_count    |         |        |       | bigint |         8 |
    true
     _col1       |         |        |       | bigint |         8 |
    false
    (2 rows)
    ```

  - 示例3

    ```
    PREPARE my_create FROM
    ```

```
CREATE TABLE foo AS SELECT * FROM nation;
```

执行 `DESCRIBE OUTPUT`，输出：

```
DESCRIBE OUTPUT my_create;
 Column Name | Catalog | Schema | Table |  Type  | Type Size |
Aliased
-------------+---------+--------+-------+--------+-----------
+---------
 rows        |         |        |       | bigint |         8 |
false
(1 row)
```

DROP SCHEMA

- 概要

  ```
  DROP SCHEMA [ IF EXISTS ] schema_name
  ```

- 描述

  删除Schema。

  - Schema必须为空。
  - 使用 `IF EXISTS` 来避免删除不存在的Schema时报错。

- 示例

  ```
  DROP SCHEMA web;
  DROP TABLE IF EXISTS sales;
  ```

DROP TABLE

- 概要

  ```
  DROP TABLE [ IF EXISTS ] table_name
  ```

- 描述

  删除数据表。使用**IF EXISTS**来避免删除不存在的表时报错。

- 示例

  ```
  DROP TABLE orders_by_date;
  ```

```
DROP TABLE IF EXISTS orders_by_date;
```

DROP VIEW

- 概要

```
DROP VIEW [ IF EXISTS ] view_name
```

- 描述

  删除视图。使用**IF EXISTS**来避免删除不存在的视图时报错。

- 示例

```
DROP VIEW orders_by_date;
DROP VIEW IF EXISTS orders_by_date;
```

EXECUTE

- 概要

```
EXECUTE statement_name [ USING parameter1 [ , parameter2, ... ] ]
```

- 描述

  执行预编译查询， 使用**USING**子句按位置传参。

- 示例

  — 示例1

```
PREPARE my_select1 FROM
SELECT name FROM nation;
--- 执行预编译查询
EXECUTE my_select1;
```

  — 示例2

```
PREPARE my_select2 FROM
SELECT name FROM nation WHERE regionkey = ? and nationkey < ?;
--- 执行预编译查询
EXECUTE my_select2 USING 1, 3;
--- 上述语句等价与执行如下查询：
SELECT name FROM nation WHERE regionkey = 1 AND nationkey < 3;
```

EXPLAIN

- 概要

```
EXPLAIN [ ( option [, ...] ) ] statement
这里Option可是时如下几个:
    FORMAT { TEXT | GRAPHVIZ }
```

```
    TYPE { LOGICAL | DISTRIBUTED | VALIDATE }
```

- 描述

  根据选项不同，可以实现如下几个功能：

  - 显示查询语句的逻辑计划。
  - 显示查询语句的分布式执行计划。
  - 验证一个查询语句的合法性。

  使用 `TYPE DISTRIBUTED` 选项来显示计划分片。每个计划分片都在单个或多个Presto节点上执行。分片之间的间隔表示Presto节点间的数据交换。分片类型说明分片是如何被Presto节点执行的，以及数据是如何在分片间分布的。 分片类型如下：

  - `SINGLE`：分片在单个节点上执行。
  - `HASH`：分片在固定个数的节点上执行，输入数据在这些节点上通过hash函数分布。
  - `ROUND_ROBIN`：分片在固定个数的节点上执行，输入数据在这些节点上通过`ROUND_ROBIN`的模式分布。
  - `BROADCAST`：分片在固定个数的节点上执行，处理数据通过广播的方式分发到所有节点。
  - `SOURCE`：分片在存储数据的节点上执行。

- 示例

  ▬ 示例1

  逻辑计划：

```
presto:tiny> EXPLAIN SELECT regionkey, count(*) FROM nation GROUP
BY 1;
                                                  Query Plan
-----------------------------------------------------------------------------
 - Output[regionkey, _col1] => [regionkey:bigint, count:bigint]
         _col1 := count
     - RemoteExchange[GATHER] => regionkey:bigint, count:bigint
         - Aggregate(FINAL)[regionkey] => [regionkey:bigint, count
:bigint]
                count := "count"("count_8")
             - LocalExchange[HASH][$hashvalue] ("regionkey") =>
regionkey:bigint, count_8:bigint, $hashvalue:bigint
                 - RemoteExchange[REPARTITION][$hashvalue_9] =>
regionkey:bigint, count_8:bigint, $hashvalue_9:bigint
                     - Project[] => [regionkey:bigint, count_8:
bigint, $hashvalue_10:bigint]
                             $hashvalue_10 := "combine_hash"(
BIGINT '0', COALESCE("$operator$hash_code"("regionkey"), 0))
                         - Aggregate(PARTIAL)[regionkey] => [
regionkey:bigint, count_8:bigint]
                                 count_8 := "count"(*)
                             - TableScan[tpch:tpch:nation:sf0.1,
originalConstraint = true] => [regionkey:bigint]
```

```
                                                regionkey := tpch:regionkey
```

- 示例2

  分布式计划：

```
presto:tiny> EXPLAIN (TYPE DISTRIBUTED) SELECT regionkey, count
(*) FROM nation GROUP BY 1;
                                           Query Plan
-----------------------------------------------------------------------------
 Fragment 0 [SINGLE]
     Output layout: [regionkey, count]
     Output partitioning: SINGLE []
     - Output[regionkey, _col1] => [regionkey:bigint, count:bigint
]
             _col1 := count
         - RemoteSource[1] => [regionkey:bigint, count:bigint]
 Fragment 1 [HASH]
     Output layout: [regionkey, count]
     Output partitioning: SINGLE []
     - Aggregate(FINAL)[regionkey] => [regionkey:bigint, count:
bigint]
             count := "count"("count_8")
         - LocalExchange[HASH][$hashvalue] ("regionkey") =>
regionkey:bigint, count_8:bigint, $hashvalue:bigint
             - RemoteSource[2] => [regionkey:bigint, count_8:
bigint, $hashvalue_9:bigint]
 Fragment 2 [SOURCE]
     Output layout: [regionkey, count_8, $hashvalue_10]
     Output partitioning: HASH [regionkey][$hashvalue_10]
     - Project[] => [regionkey:bigint, count_8:bigint, $hashvalue_
10:bigint]
             $hashvalue_10 := "combine_hash"(BIGINT '0', COALESCE
("$operator$hash_code"("regionkey"), 0))
         - Aggregate(PARTIAL)[regionkey] => [regionkey:bigint,
count_8:bigint]
                 count_8 := "count"(*)
             - TableScan[tpch:tpch:nation:sf0.1, originalCo
nstraint = true] => [regionkey:bigint]
                     regionkey := tpch:regionkey
```

- 示例3

  验证：

```
presto:tiny> EXPLAIN (TYPE VALIDATE) SELECT regionkey, count(*)
FROM nation GROUP BY 1;
 Valid
-------
 true
```

EXPLAIN ANALYZE

- 概要

```
EXPLAIN ANALYZE [VERBOSE] statement
```

- 描述

执行Statement，并且显示该次查询实际的分布式执行计划以及每个执行过程的成本。启用 VERBOSE 选项可以获取更多详细信息和底层的统计量。

- 示例

在下面的示例中，你可以看到每个Stage消耗的CPU时间和该Stage中每个计划节点的相对成本。需要注意的是，相对成本使用实际时间来计量，因此，无法显示其与CPU时间的相关性。对于每个计划节点，还会显示一些附加的统计信息，这些统计信息有助于发现一次查询中的数据异常情况（如倾斜，hash冲突等）。

```
presto:sf1> EXPLAIN ANALYZE SELECT count(*), clerk FROM orders WHERE
 orderdate > date '1995-01-01' GROUP BY clerk;
                                                Query Plan
-----------------------------------------------------------------------------
Fragment 1 [HASH]
    Cost: CPU 88.57ms, Input: 4000 rows (148.44kB), Output: 1000
rows (28.32kB)
    Output layout: [count, clerk]
    Output partitioning: SINGLE []
    - Project[] => [count:bigint, clerk:varchar(15)]
            Cost: 26.24%, Input: 1000 rows (37.11kB), Output: 1000
rows (28.32kB), Filtered: 0.00%
            Input avg.: 62.50 lines, Input std.dev.: 14.77%
        - Aggregate(FINAL)[clerk][$hashvalue] => [clerk:varchar(15),
 $hashvalue:bigint, count:bigint]
                Cost: 16.83%, Output: 1000 rows (37.11kB)
                Input avg.: 250.00 lines, Input std.dev.: 14.77%
                count := "count"("count_8")
            - LocalExchange[HASH][$hashvalue] ("clerk") => clerk:
varchar(15), count_8:bigint, $hashvalue:bigint
                    Cost: 47.28%, Output: 4000 rows (148.44kB)
                    Input avg.: 4000.00 lines, Input std.dev.: 0.00%
                - RemoteSource[2] => [clerk:varchar(15), count_8:
bigint, $hashvalue_9:bigint]
                        Cost: 9.65%, Output: 4000 rows (148.44kB)
                        Input avg.: 4000.00 lines, Input std.dev.: 0
.00%
Fragment 2 [tpch:orders:1500000]
    Cost: CPU 14.00s, Input: 818058 rows (22.62MB), Output: 4000
rows (148.44kB)
    Output layout: [clerk, count_8, $hashvalue_10]
    Output partitioning: HASH [clerk][$hashvalue_10]
    - Aggregate(PARTIAL)[clerk][$hashvalue_10] => [clerk:varchar(15
), $hashvalue_10:bigint, count_8:bigint]
            Cost: 4.47%, Output: 4000 rows (148.44kB)
            Input avg.: 204514.50 lines, Input std.dev.: 0.05%
            Collisions avg.: 5701.28 (17569.93% est.), Collisions
std.dev.: 1.12%
            count_8 := "count"(*)
        - ScanFilterProject[table = tpch:tpch:orders:sf1.0,
originalConstraint = ("orderdate" > "$literal$date"(BIGINT '9131
')), filterPredicate = ("orderdate" > "$literal$date"(BIGINT '9131
'))] => [cler
                Cost: 95.53%, Input: 1500000 rows (0B), Output:
818058 rows (22.62MB), Filtered: 45.46%
                Input avg.: 375000.00 lines, Input std.dev.: 0.00%
```

```
                $hashvalue_10 := "combine_hash"(BIGINT '0', COALESCE
("$operator$hash_code"("clerk"), 0))
                orderdate := tpch:orderdate
                clerk := tpch:clerk
```

如果开启了 VERBOSE 选项，运行操作符会返回更多信息：

```
EXPLAIN ANALYZE VERBOSE SELECT count(clerk) OVER() FROM orders WHERE
 orderdate > date '1995-01-01';
                                        Query Plan
------------------------------------------------------------------------------
  ...
         - Window[] => [clerk:varchar(15), count:bigint]
                Cost: {rows: ?, bytes: ?}
                CPU fraction: 75.93%, Output: 8130 rows (230.24kB)
                Input avg.: 8130.00 lines, Input std.dev.: 0.00%
                Active Drivers: [ 1 / 1 ]
                Index size: std.dev.: 0.00 bytes , 0.00 rows
                Index count per driver: std.dev.: 0.00
                Rows per driver: std.dev.: 0.00
                Size of partition: std.dev.: 0.00
                count := count("clerk")
 ...
```

GRANT

- 概要

```
GRANT ( privilege [, ...] | ( ALL PRIVILEGES ) )
ON [ TABLE ] table_name TO ( grantee | PUBLIC )
[ WITH GRANT OPTION ]
```

- 描述

  授予指定的权限指定的受让对象。

  - ALL PRIVILEGES 包括 DELETE, INSERT 和 SELECT 权限。
  - PUBLIC表示所有用户。
  - 使用WITH GRANT OPTION允许权限受让对象给其他用户赋予相同的权限。

- 示例

```
GRANT INSERT, SELECT ON orders TO alice; --- 给用户alice赋权
GRANT SELECT ON nation TO alice WITH GRANT OPTION; --- 给用户alice赋
权，同时，alice还具有赋予其他用户**SELECT**权限的权限，
GRANT SELECT ON orders TO PUBLIC; --- 开放表order的**SELECT**权限给所有
人
```

- 局限

  某些连接器可能不支持 GRANT。

INSERT

- 概要

```
INSERT INTO table_name [ ( column [, ... ] ) ] query
```

- 描述

  插入数据。如果指定了列名列表,该列表必须精确匹配 `query` 的结果集。没有出现在列名列表中的列将使用`null`进行填充。

- 示例

```
INSERT INTO orders SELECT * FROM new_orders; --- 将select结果插入表orders中
INSERT INTO cities VALUES (1, 'San Francisco'); --- 插入一行数据
INSERT INTO cities VALUES (2, 'San Jose'), (3, 'Oakland'); --- 插入多行数据
INSERT INTO nation (nationkey, name, regionkey, comment) VALUES (26, 'POLAND', 3, 'no comment'); --- 插入一行数据
INSERT INTO nation (nationkey, name, regionkey) VALUES (26, 'POLAND', 3); --- 插入一行数据(只包括部分列)
```

PREPARE

- 概要

```
PREPARE statement_name FROM statement
```

- 描述

  创建一个预处理语句，以备后续使用。预处理语句就是一组保存在会话中的命名查询语句集合，可以在这些语句中定义参数，在实际执行时填充实际的值。定义的参数使用?占位。

- 示例

```
--- 不带参数的预处理查询语句
PREPARE my_select1 FROM
SELECT * FROM nation;
--- 带参数的预处理查询语句
PREPARE my_select2 FROM
SELECT name FROM nation WHERE regionkey = ? AND nationkey < ?;
--- 不带参数的预处理插入语句
PREPARE my_insert FROM
INSERT INTO cities VALUES (1, 'San Francisco');
```

RESET SESSION

- 概要

```
RESET SESSION name
```

```
RESET SESSION catalog.name
```

- 描述

  重置会话，使用默认属性。

- 示例

```
RESET SESSION optimize_hash_generation;
RESET SESSION hive.optimized_reader_enabled;
```

REVOKE

- 概要

```
REVOKE [ GRANT OPTION FOR ]
( privilege [, ...] | ALL PRIVILEGES )
ON [ TABLE ] table_name FROM ( grantee | PUBLIC )
```

- 描述

  撤回指定用户的指定权限。

  - 使用ALL PRIVILEGE可以撤销SELECT，INSERT和DELETE权限。
  - 使用PUBLIC表示撤销对象为PUBLIC角色，用户通过其他角色获得的权限不会被收回。
  - 使用GRANT OPTION FOR将进一步收回用户使用GRANT进行赋权的权限。
  - 语句中grantee既可以是单个用户也可以是角色。

- 示例

```
--- 撤回用户alice对表orders进行INSET和SELECT的权限
REVOKE INSERT, SELECT ON orders FROM alice;
--- 撤回所有人对表nation进行SELECT的权限，
--- 同时，撤回所有人赋予其他人对该表进行SELECT操作权限的权限
REVOKE GRANT OPTION FOR SELECT ON nation FROM PUBLIC;
--- 撤回用户alice在表test上的所有权限
REVOKE ALL PRIVILEGES ON test FROM alice;
```

- 局限

  有些连接器不支持REVOKE语句。

ROLLBACK

- 概要

```
ROLLBACK [ WORK ]
```

- 描述

  回滚当前事物。

- 示例

```
ROLLBACK;
ROLLBACK WORK;
```

SELECT

- 概要

```
[ WITH with_query [, ...] ]
SELECT [ ALL | DISTINCT ] select_expr [, ...]
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ GROUP BY [ ALL | DISTINCT ] grouping_element [, ...] ]
[ HAVING condition]
[ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] select ]
[ ORDER BY expression [ ASC | DESC ] [, ...] ]
[ LIMIT [ count | ALL ] ]
```

其中，`from_item`有如下两种形式：

```
table_name [ [ AS ] alias [ ( column_alias [, ...] ) ] ]
```

```
from_item join_type from_item [ ON join_condition | USING (
join_column [, ...] ) ]
```

这里的 `join_type` 可以是如下之一：

- [INNER] JOIN
- LEFT [OUTER] JOIN
- RIGHT [OUTER] JOIN
- FULL [OUTER] JOIN
- CROSS JOIN

而 `grouping_element` 可以是如下之一：

- ()
- expression
- GROUPING SETS ((column [, ⋯]) [, ⋯])
- CUBE (column [, ⋯])
- ROLLUP (column [, ⋯])

- 描述

检索0到多张数据表，获取结果集。

- WITH子句

- 基本功能

  WITH子句可以定义一组命名关系，这些关系可以在查询中被使用，从而扁平化嵌套查询或简化子查询。如下两个查询语句是等价的：

```
--- 不使用WITH子句
SELECT a, b
FROM (
  SELECT a, MAX(b) AS b FROM t GROUP BY a
) AS x;
--- 使用WITH子句，查询语句看起来更加明了
WITH x AS (SELECT a, MAX(b) AS b FROM t GROUP BY a)
SELECT a, b FROM x;
```

- 定义多个子查询

  WITH 子句中可以定义多个子查询：

```
WITH
  t1 AS (SELECT a, MAX(b) AS b FROM x GROUP BY a),
  t2 AS (SELECT a, AVG(d) AS d FROM y GROUP BY a)
SELECT t1.*, t2.*
FROM t1
JOIN t2 ON t1.a = t2.a;
```

- 组成链式结构

  WITH 子句中的关系还可以组成链式结构：

```
WITH
  x AS (SELECT a FROM t),
  y AS (SELECT a AS b FROM x),
  z AS (SELECT b AS c FROM y)
SELECT c FROM z;
```

- GROUP BY子句

  - 基本功能

    使用 `GROUP BY` 子句可以对 `SELECT` 结果进行分组。`GROUP BY`子句支持任意的表达式，既可以使用列名，也可以使用序号（从1开始）。

    下列示例中的查询语句是等价的（列 `nationkey` 的位置为2）。

```
--- 使用列序号
SELECT count(*), nationkey FROM customer GROUP BY 2;
--- 使用列名
SELECT count(*), nationkey FROM customer GROUP BY nationkey;
```

    没有在输出列表中指定的列也可以用于 `GROUP BY` 子句，如下所示：

```
--- 列mktsegment没有在SELECT列表中指定，
```

```
--- 结果集中不包括mktsegment列的内容。
SELECT count(*) FROM customer GROUP BY mktsegment;
 _col0
-------
 29968
 30142
 30189
 29949
 29752
(5 rows)
```

说明：

在SELECT语句中使用GROUP BY子句时，其输出表达式只能是聚合函数或者是GROUP BY子句中使用的列。

复杂分组操作

Presto支持如下3中复杂的聚合语法，可以在一个查询中实现多个列集合的聚合分析：

- GROUPING SETS（分组集）

  GROUPING SETS可以在一条查询语句中完成多个列的分组聚合。没有在分组列表中的列使用NULL进行填充。

  表shipping是一个包含5个列的数据表，如下所示：

```
SELECT * FROM shipping;
 origin_state | origin_zip | destination_state | destinatio
n_zip | package_weight
--------------+------------+-------------------
+-----------------+----------------
 California   |      94131 | New Jersey        |
8648 |              13
 California   |      94131 | New Jersey        |
8540 |              42
 New Jersey   |       7081 | Connecticut       |
6708 |             225
 California   |      90210 | Connecticut       |
6927 |            1337
 California   |      94131 | Colorado          |
80302 |               5
 New York     |      10002 | New Jersey        |
8540 |               3
(6 rows)
```

  现在希望在一个查询中获取如下几个分组结果：

  - 按origin_state进行分组，获取package_weight的总和。
  - 按origin_state和origin_zip分组，获取package_weight的总和。
  - 按destination_state分组，获取package_weight的总和。

使用GROUPING SETS可以在一条语句中获取上述3个分组的结果集，如下所示：

```
SELECT origin_state, origin_zip, destination_state, sum(
package_weight)
FROM shipping
GROUP BY GROUPING SETS (
    (origin_state),
    (origin_state, origin_zip),
    (destination_state));
 origin_state | origin_zip | destination_state | _col0
--------------+------------+-------------------+-------
 New Jersey   | NULL       | NULL              |   225
 California   | NULL       | NULL              |  1397
 New York     | NULL       | NULL              |     3
 California   |      90210 | NULL              |  1337
 California   |      94131 | NULL              |    60
 New Jersey   |       7081 | NULL              |   225
 New York     |      10002 | NULL              |     3
 NULL         | NULL       | Colorado          |     5
 NULL         | NULL       | New Jersey        |    58
 NULL         | NULL       | Connecticut       |  1562
(10 rows)
```

上述查询逻辑也可以通过UNION ALL多个GROUP BY查询实现：

```
SELECT origin_state, NULL, NULL, sum(package_weight)
FROM shipping GROUP BY origin_state
UNION ALL
SELECT origin_state, origin_zip, NULL, sum(package_weight)
FROM shipping GROUP BY origin_state, origin_zip
UNION ALL
SELECT NULL, NULL, destination_state, sum(package_weight)
FROM shipping GROUP BY destination_state;
```

但是，使用GROUPING SETS语法往往能获得更好的性能，因为，该语法在执行时，只会读取一次基表数据，而使用上述UNION ALL方式，会读取3次，因此，如果在查询期间基表数据有变化，使用UNION ALL的方式容易出现不一致的结果。

- CUBE（多维立方）

使用CUBE可以获得给定列列表所有可能的分组结果。如下所示：

```
SELECT origin_state, destination_state, sum(package_weight)
FROM shipping
GROUP BY CUBE (origin_state, destination_state);
 origin_state | destination_state | _col0
--------------+-------------------+-------
 California   | New Jersey        |    55
 California   | Colorado          |     5
 New York     | New Jersey        |     3
 New Jersey   | Connecticut       |   225
 California   | Connecticut       |  1337
 California   | NULL              |  1397
 New York     | NULL              |     3
 New Jersey   | NULL              |   225
 NULL         | New Jersey        |    58
```

```
 NULL          | Connecticut        |  1562
 NULL          | Colorado           |     5
 NULL          | NULL               |  1625
(12 rows)
```

该查询等价于如下语句：

```
SELECT origin_state, destination_state, sum(package_weight)
FROM shipping
GROUP BY GROUPING SETS (
    (origin_state, destination_state),
    (origin_state),
    (destination_state),
    ());
```

- ROLLUP (汇总)

使用ROLLUP可以获得给定列集和的小记结果。如下所示：

```
SELECT origin_state, origin_zip, sum(package_weight)
FROM shipping
GROUP BY ROLLUP (origin_state, origin_zip);
 origin_state | origin_zip | _col2
--------------+------------+-------
 California   |      94131 |    60
 California   |      90210 |  1337
 New Jersey   |       7081 |   225
 New York     |      10002 |     3
 California   | NULL       |  1397
 New York     | NULL       |     3
 New Jersey   | NULL       |   225
 NULL         | NULL       |  1625
(8 rows)
```

上述示例等价与如下语句：

```
SELECT origin_state, origin_zip, sum(package_weight)
FROM shipping
GROUP BY GROUPING SETS ((origin_state, origin_zip), (origin_sta
te), ());
```

- 综合使用

下列3个语句是等价的：

```
SELECT origin_state, destination_state, origin_zip, sum(
package_weight)
FROM shipping
GROUP BY
    GROUPING SETS ((origin_state, destination_state)),
    ROLLUP (origin_zip);
```

```
SELECT origin_state, destination_state, origin_zip, sum(
package_weight)
FROM shipping
GROUP BY
    GROUPING SETS ((origin_state, destination_state)),
```

```
    GROUPING SETS ((origin_zip), ());
```

```
SELECT origin_state, destination_state, origin_zip, sum(
package_weight)
FROM shipping
GROUP BY GROUPING SETS (
    (origin_state, destination_state, origin_zip),
    (origin_state, destination_state));
```

输出结果如下：

```
origin_state | destination_state | origin_zip | _col3
--------------+-------------------+------------+-------
 New York     | New Jersey        |      10002 |     3
 California   | New Jersey        |      94131 |    55
 New Jersey   | Connecticut       |       7081 |   225
 California   | Connecticut       |      90210 |  1337
 California   | Colorado          |      94131 |     5
 New York     | New Jersey        | NULL       |     3
 New Jersey   | Connecticut       | NULL       |   225
 California   | Colorado          | NULL       |     5
 California   | Connecticut       | NULL       |  1337
 California   | New Jersey        | NULL       |    55
(10 rows)
```

在GROUP BY子句中，可以使用ALL和DISTINCT修饰符，用来说明是否可以生成重复的统计维度。如下所示：

```
SELECT origin_state, destination_state, origin_zip, sum(
package_weight)
FROM shipping
GROUP BY ALL
    CUBE (origin_state, destination_state),
    ROLLUP (origin_state, origin_zip);
```

等价于：

```
SELECT origin_state, destination_state, origin_zip, sum(
package_weight)
FROM shipping
GROUP BY GROUPING SETS (
    (origin_state, destination_state, origin_zip),
    (origin_state, origin_zip),
    (origin_state, destination_state, origin_zip),
    (origin_state, origin_zip),
    (origin_state, destination_state),
    (origin_state),
    (origin_state, destination_state),
    (origin_state),
    (origin_state, destination_state),
    (origin_state),
    (destination_state),
```

```
    ());
```

其中有多个维度是重复的。如果使用DISTINCT，则只会输出不重复的维度。

```
SELECT origin_state, destination_state, origin_zip, sum(
package_weight)
FROM shipping
GROUP BY DISTINCT
    CUBE (origin_state, destination_state),
    ROLLUP (origin_state, origin_zip);
```

等价于：

```
SELECT origin_state, destination_state, origin_zip, sum(
package_weight)
FROM shipping
GROUP BY GROUPING SETS (
    (origin_state, destination_state, origin_zip),
    (origin_state, origin_zip),
    (origin_state, destination_state),
    (origin_state),
    (destination_state),
    ());
```

说明：

GROUP BY默认使用的修饰符为ALL。

- GROUPING函数

Presto提供了一个grouping函数，用于生成一个标记数，该数中的每一个比特位表示对应的列是否出现在该分组条件中。语法如下：

```
grouping(col1, ..., colN) -> bigint
```

grouping通常与GROUPING SETS，ROLLUP，CUBE或GROUP BY一起使用。grouping中的列必须与GROUPING SETS，ROLLUP，CUBE或GROUP BY中指定的列一一对应。

```
SELECT origin_state, origin_zip, destination_state, sum(package_we
ight),
       grouping(origin_state, origin_zip, destination_state)
FROM shipping
GROUP BY GROUPING SETS (
        (origin_state),
        (origin_state, origin_zip),
        (destination_state));
origin_state | origin_zip | destination_state | _col3 | _col4
-------------+------------+-------------------+-------+-------
California   | NULL       | NULL              |  1397 |     3
 --- 011
New Jersey   | NULL       | NULL              |   225 |     3
 --- 011
New York     | NULL       | NULL              |     3 |     3
 --- 011
```

```
California   |      94131 | NULL               |    60 |     1
 --- 001
New Jersey   |       7081 | NULL               |   225 |     1
 --- 001
California   |      90210 | NULL               |  1337 |     1
 --- 001
New York     |      10002 | NULL               |     3 |     1
 --- 001
NULL         | NULL       | New Jersey         |    58 |     6
 --- 100
NULL         | NULL       | Connecticut        |  1562 |     6
 --- 100
NULL         | NULL       | Colorado           |     5 |     6
 --- 100
(10 rows)
```

如上所示，grouping函数返回的是右对齐的比特标记数，0 表示列存在，1 表示列不存在。

- HAVING子句

  HAVING子句用来控制查询中分组的选择，与聚合函数和GROUP BY子句一起使用。HAVING子句的功能会在分组和聚合计算完成后进行，过滤掉不满足条件的分组。

  下面的示例将账户结余大于5700000的用户选出来。

```
SELECT count(*), mktsegment, nationkey,
       CAST(sum(acctbal) AS bigint) AS totalbal
FROM customer
GROUP BY mktsegment, nationkey
HAVING sum(acctbal) > 5700000
ORDER BY totalbal DESC;
```

  输出如下：

```
_col0 | mktsegment | nationkey | totalbal
-------+------------+-----------+----------
  1272 | AUTOMOBILE |        19 |  5856939
  1253 | FURNITURE  |        14 |  5794887
  1248 | FURNITURE  |         9 |  5784628
  1243 | FURNITURE  |        12 |  5757371
  1231 | HOUSEHOLD  |         3 |  5753216
  1251 | MACHINERY  |         2 |  5719140
  1247 | FURNITURE  |         8 |  5701952
(7 rows)
```

- 集合运算

  Presto支持UNION、INTERSECT和EXCEPT3种集合运算。这些子句用来将多个查询语句的结果合并成一个总的结果集。使用方法如下所示：

```
query UNION [ALL | DISTINCT] query
query INTERSECT [DISTINCT] query
query EXCEPT [DISTINCT] query
```

  参数**ALL**和**DISTINCT**来控制最终哪些行会出现在结果集中，默认设为**DISTINCT**。

— **ALL**，有可能返回重复的行。

— **DISTINCT**，对结果集进行去重。

INTERSECT和EXCEPT不支持**ALL**选项。

上述3个集合运算可以组合使用，运算从左到右进行，INTERSECT的优先级最高，因此，组合运算A UNION B INTERSECT C EXCEPT D实际的顺序是A UNION (B INTERSECT C) EXCEPT D。

- UNION

  UNION对两个查询的结果集做并集运算，通过**ALL**和**DISTINCT**来控制是否剔除重复项。

  - 示例1

    ```
    SELECT 13
    UNION
    SELECT 42;
     _col0
    -------
        13
        42
    (2 rows)
    ```

  - 示例2

    ```
    SELECT 13
    UNION
    SELECT * FROM (VALUES 42, 13);
     _col0
    -------
        13
        42
    (2 rows)
    ```

  - 示例3

    ```
    SELECT 13
    UNION ALL
    SELECT * FROM (VALUES 42, 13);
     _col0
    -------
        13
        42
        13
    (3 rows)
    ```

- **INTERSECT**

  INTERSECT对两个查询的结果集做交集运算。

示例：

```
SELECT * FROM (VALUES 13, 42)
INTERSECT
SELECT 13;
 _col0
-------
   13
(1 rows)
```

- EXCEPT

  EXCEPT求两个查询结果集的补集。

```
SELECT * FROM (VALUES 13, 42)
EXCEPT
SELECT 13;
 _col0
-------
   42
(1 rows)
```

- ORDER BY子句

  在查询中可以使用ORDER BY子句对查询结果进行排序。语法如下：

```
ORDER BY expression [ ASC | DESC ] [ NULLS { FIRST | LAST } ]
 [, ...]
```

  其中：

  - **expression**由列名或列位置序号（从1开始）组成。
  - ORDER BY在GROUP BY和HAVING之后执行。
  - **NULLS { FIRST | LAST }**可以控制**NULL**值的排序方式（与ASC/DESC无关），默认为LAST。

- LIMIT子句

  LIMIT子句用来控制结果集的规模（即行数）。使用LIMIT ALL等价于不使用LIMIT子句。

  示例：

```
--- 本例中，因为没有使用ORDER BY子句，返回结果是随机的
SELECT orderdate FROM orders LIMIT 5;
 orderdate
-------------
 1996-04-14
 1992-01-15
 1995-02-01
 1995-11-12
 1992-04-26
```

```
(5 rows)
```

- TABLESAMPLE

  Presto提供了两种采样方式，BERNOULLI和SYSTEM，但是无论哪种采样方式， 都无法确定的给出采样结果集的行数。

  - BERNOULLI：

    本方法基于样本百分比概率进行采样。当使用Bernoulli方法对表格进行采样时，执行器会扫描表格的所有物理块，并且基于样本百分比与运行时计算的随机值之间的比较结果跳过某些行。

    每一行被选中对概率是独立对，与其他行无关，但是，这不会减少从磁盘读取采样表所需的时间。如果还需对采样结果做进一步处理，可能会对总查询时间产生影响。

  - SYSTEM：

    本方法将表格分成数据的逻辑段，并以此粒度对表格进行采样。这种抽样方法要么从一个特定的数据段中选择所有的行，要么跳过它（根据样本百分比和在运行时计算的一个随机值之间的比较）。

    采样结果与使用哪种连接器有关。例如，与Hive一起使用时，取决于数据在HDFS上的分布。这种方法不保证独立的抽样概率。

  示例：

```
--- 使用BERNOULLI采样
SELECT *
FROM users TABLESAMPLE BERNOULLI (50);
--- 使用SYSTEM采样
SELECT *
FROM users TABLESAMPLE SYSTEM (75);
Using sampling with joins:
--- 采样过程中使用JOIN
SELECT o.*, i.*
FROM orders o TABLESAMPLE SYSTEM (10)
JOIN lineitem i TABLESAMPLE BERNOULLI (40)
  ON o.orderkey = i.orderkey;
```

- UNNEST

  UNNEST可以将 ARRAY 和 MAP 类型的变量展开成表。其中 ARRAY 展开为单列的表， MAP 展开为双列的表（键，值）。UNNEST可以一次展开多个 ARRAY 和 MAP 类型的变量，在这种情况下，它们被扩展为多个列，行数等于输入参数列表中最大展开行数（其他列用空值填充）。UNNEST可以有一个WITH ORDINALITY子句，此时，查询结果中会附加一个序数列。UNNEST通常与JOIN一起使用，并可以引用join左侧的关系的列。

- 示例1

```
--- 使用一个列
SELECT student, score
FROM tests
CROSS JOIN UNNEST(scores) AS t (score);
```

- 示例2

```
--- 使用多个列
SELECT numbers, animals, n, a
FROM (
  VALUES
    (ARRAY[2, 5], ARRAY['dog', 'cat', 'bird']),
    (ARRAY[7, 8, 9], ARRAY['cow', 'pig'])
) AS x (numbers, animals)
CROSS JOIN UNNEST(numbers, animals) AS t (n, a);
  numbers  |     animals      |  n   |  a
-----------+------------------+------+------
 [2, 5]    | [dog, cat, bird] |    2 | dog
 [2, 5]    | [dog, cat, bird] |    5 | cat
 [2, 5]    | [dog, cat, bird] | NULL | bird
 [7, 8, 9] | [cow, pig]       |    7 | cow
 [7, 8, 9] | [cow, pig]       |    8 | pig
 [7, 8, 9] | [cow, pig]       |    9 | NULL
(6 rows)
```

- 示例3

```
--- 使用 WITH ORDINALITY 子句
SELECT numbers, n, a
FROM (
  VALUES
    (ARRAY[2, 5]),
    (ARRAY[7, 8, 9])
) AS x (numbers)
CROSS JOIN UNNEST(numbers) WITH ORDINALITY AS t (n, a);
  numbers  | n | a
-----------+---+---
 [2, 5]    | 2 | 1
 [2, 5]    | 5 | 2
 [7, 8, 9] | 7 | 1
 [7, 8, 9] | 8 | 2
 [7, 8, 9] | 9 | 3
(5 rows)
```

- **Joins**

Joins可以将多个关系的数据合并到一起。`CROSS JOIN` 返回两个关系到笛卡尔积（所有排列组合的集合）。`CROSS JOIN`可以通过如下两种方式来指定：

- 显式的使用`CROSS JOIN`语法。
- 在`FROM`子句中指定多个关系。

下列两个SQL语句是等价的：

```
--- 显式的使用**CROSS JOIN**语法
SELECT *
FROM nation
CROSS JOIN region;
--- 在**FROM**子句中指定多个关系
SELECT *
FROM nation, region;
```

示例：表nation包括25行记录，表region包括5行，对这两个表执行cross join操作，其结果集为125行记录。

```
SELECT n.name AS nation, r.name AS region
FROM nation AS n
CROSS JOIN region AS r
ORDER BY 1, 2;
     nation     |   region
----------------+-------------
 ALGERIA        | AFRICA
 ALGERIA        | AMERICA
 ALGERIA        | ASIA
 ALGERIA        | EUROPE
 ALGERIA        | MIDDLE EAST
 ARGENTINA      | AFRICA
 ARGENTINA      | AMERICA
...
(125 rows)
```

如果参与join的表中包含相同名称的列，则需要用表名（或别名）加以修饰，示例如下：

```
--- 正确
SELECT nation.name, region.name
FROM nation
CROSS JOIN region;
--- 正确
SELECT n.name, r.name
FROM nation AS n
CROSS JOIN region AS r;
--- 正确
SELECT n.name, r.name
FROM nation n
CROSS JOIN region r;
--- 错误，会抛出"Column 'name' is ambiguous"
SELECT name
FROM nation
CROSS JOIN region;
```

- 子查询

子查询是由一个查询组成的表达式。当子查询引用了子查询之外的列时，子查询与外部查询是相关的。Presto对相关子查询的支持比较有限。

- EXISTS

谓语EXISTS用于确定子查询是否返回所有行，如果子查询有返回值，WHERE表达式为TRUE，否则为FALSE。

示例：

```
SELECT name
FROM nation
WHERE EXISTS (SELECT * FROM region WHERE region.regionkey =
nation.regionkey);
```

- IN

如果WHERE指定的列在子查询结果集中存在，则返回结果，否则不返回。子查询只能返回一个列。

示例：

```
SELECT name
FROM nation
WHERE regionkey IN (SELECT regionkey FROM region);
```

- 标量子查询

标量子查询是一个非相关子查询，返回0～1行结果。该类子查询最多只能返回一行数据，如果子查询结果集为空，则返回 null。

示例：

```
SELECT name
FROM nation
WH ERregionkey = (SELECT max(regionkey) FROM regio;);
```

SET SESSION

- 概要

```
SET SESSION name = expression
SET SESSION catalog.name = expression
```

- 描述

设置会话属性。

- 示例

```
SET SESSION optimize_hash_generation = true;
```

```
SET SESSION hive.optimized_reader_enabled = true;
```

SHOW CATALOGS

- 概要

```
SHOW CATALOGS [ LIKE pattern ]
```

- 描述

  获取可用的Catalog清单。可以使用LIKE子句过滤catalog名称。

- 示例

```
SHOW CATALOGS;
```

SHOW COLUMNS

- 概要

```
SHOW COLUMNS FROM table
```

- 描述

  获取给定表格所有的列及其属性。

- 示例

```
SHOW COLUMNS FROM orders;
```

SHOW CREATE TABLE

- 概要

```
SHOW CREATE TABLE table_name
```

- 描述

  显示定义给定表格的SQL语句。

- 示例

```
SHOW CREATE TABLE sf1.orders;
----------------------------------------
 CREATE TABLE tpch.sf1.orders (
    orderkey bigint,
    orderstatus varchar,
    totalprice double,
    orderdate varchar
 )
 WITH (
    format = 'ORC',
    partitioned_by = ARRAY['orderdate']
 )
```

```
(1 row)
```

SHOW CREATE VIEW

- 概要

```
SHOW CREATE VIEW view_name
```

- 描述

  显示定义给定视图的SQL语句。

- 示例

```
SHOW CREATE VIEW view1;
```

SHOW FUNCTIONS

- 概要

```
SHOW FUNCTIONS
```

- 描述

  列出所有可用于用于查询的函数。

- 示例

```
SHOW FUNCTIONS
```

SHOW GRANTS

- 概要

```
SHOW GRANTS [ ON [ TABLE ] table_name ]
```

- 描述

  显示权限列表。

- 示例

```
--- 获取当前用户在表orders上的权限
SHOW GRANTS ON TABLE orders;
--- 获取当前用户在当前catalog中的权限
SHOW GRANTS;
```

- 局限

  有些连接器不支持`SHOW GRANTS`操作。

SHOW SCHEMAS

- 概要

```
SHOW SCHEMAS [ FROM catalog ] [ LIKE pattern ]
```

- 描述

  列出给定catalog下的所有schema，catalog不指定则表示当前catalog。使用`LIKE`子句可以过滤schema名称。

- 示例

```
SHOW SCHEMAS;
```

SHOW SESSION

- 概要

```
SHOW SESSION
```

- 描述

  显示当前回话的属性列表。

- 示例

```
SHOW SESSION
```

SHOW TABLES

- 概要

```
SHOW TABLES [ FROM schema ] [ LIKE pattern ]
```

- 描述

  列出给定Schema下的所有表格，schema不指定则表示当前schema。使用`LIKE`子句可以过滤表名。

- 示例

```
SHOW TABLES;
```

START TRANSACTION

- 概要

```
START TRANSACTION [ mode [, ...] ]
其中**mode**可以从如下几个选项中选择：
```

```
ISOLATION LEVEL { READ UNCOMMITTED | READ COMMITTED | REPEATABLE
READ | SERIALIZABLE }
READ { ONLY | WRITE }
```

- 描述

  在当前会话中启动一个新的事务。

- 示例

```
START TRANSACTION;
START TRANSACTION ISOLATION LEVEL REPEATABLE READ;
START TRANSACTION READ WRITE;
START TRANSACTION ISOLATION LEVEL READ COMMITTED, READ ONLY;
START TRANSACTION READ WRITE, ISOLATION LEVEL SERIALIZABLE;
```

USE

- 概要

```
USE catalog.schema
USE schema
```

- 描述

  更新当前回话，使用指定的Catalog和Schema。如果Catalog没有指定，则使用当前Catalog下的Schema。

- 示例

```
USE hive.finance;
USE information_schema;
```

VALUES

- 概要

```
VALUES row [, ...]
其中， **row**是一个表达式，或者如下形式的表达式列表：
( column_expression [, ...] )
```

- 描述

  定义一张字面内联表。

  - 任何可以使用查询语句的地方都可以使用VALUE， 如放在SELECT语句的FROM后面，放在INSERT里，甚至放在最顶层。
  - VALUE默认创建一张匿名表，并且没有列名。表名和列名可以通过AS进行命名。

- 示例

```
--- 返回一个表对象，包含1列，3行数据
```

```
VALUES 1, 2, 3
--- 返回一个表对象，包含2列，3行数据
VALUES
    (1, 'a'),
    (2, 'b'),
    (3, 'c')
--- 在查询语句中使用
SELECT * FROM (
    VALUES
        (1, 'a'),
        (2, 'b'),
        (3, 'c')
) AS t (id, name)
--- 创建一个表
CREATE TABLE example AS
SELECT * FROM (
    VALUES
        (1, 'a'),
        (2, 'b'),
        (3, 'c')
) AS t (id, name)
```

11.8.7 技术支持

遇到问题可通过以下几种方式获得技术支持。

有问题，找支持：

- 提工单

11.9 TensorFlow使用说明

E-MapReduce 3.13.0以后版本支持*TensorFlow* 。用户在软件配置可选服务中添加TensorFlow组件。用户使用E-MapReduce TensorFlow运行高性能计算时，可通过YARN对集群中的CPU、GPU进行调度。

准备工作

- 软件层面，E-MapReduce集群安装TensorFlow和TensorFlow on YARN组件。
- 硬件层面，E-MapReduce支持使用CPU和GPU两种资源进行计算。如您需使用GPU进行计算，可在core或task节点中选择ECS异构计算GPU计算型，如ecs.gn5，ecs.gn6机型。选定GPU机型后，选择您所需的CUDA和CuDNN版本。

提交TensorFlow作业

用户可以登录emr主节点已命令行的方式提交tensorflow作业，如：

```
el_submit [-h] [-t APP_TYPE] [-a APP_NAME] [-m MODE] [-m_arg MODE_ARG]
[-interact INTERACT] [-x EXIT]
```

```
[-enable_tensorboard ENABLE_TENSORBOARD]
[-log_tensorboard LOG_TENSORBOARD] [-conf CONF] [-f FILES]
[-pn PS_NUM] [-pc PS_CPU] [-pm PS_MEMORY] [-wn WORKER_NUM]
[-wc WORKER_CPU] [-wg WORKER_GPU] [-wm WORKER_MEMORY]
[-wnpg WNPG] [-ppn PPN] [-c COMMAND [COMMAND ...]]
```

基础参数说明：

- -t APP_TYPE 提交的任务类型，支持三种类型的任务类型tensorflow-ps, tensorflow-mpi, standalone，三种类型要配合后面运行模式使用。
  - tensorflow-ps采用Parameter Server方式通信，该方式为原生TensorFlow PS模式。
  - tensorflow-mpi采用的是UBER开源的MPI架构horovod进行通信。
  - standalone模式是用户将任务调度到YARN集群中启动单机任务，类似于单机运行。
- －a APP_NAME是指提交TensorFlow提交的作业名称，用户可以根据需要自行命名。
- －m MODE，是指TensorFlow作业提交的运行时环境，E-MapReduce支持三种类型运行环境local, virtual-env, docker。
  - local 模式，使用的是emr-worker上面的python运行环境，所以如果要使用一些第三方python包需要手动在所有机器上进行安装。
  - docker模式， 使用的是emr-worker上面的docker运行时，tensorflow运行在docker container内。
  - virtual-env 模式，用户上传的python环境，可以在python环境中安装一些不同于worker节点的python库。
- -m_arg MODE_ARG，提交运行模式的补充参数，和MODE配合使用，如果运行时是docker，则设置为docker镜像名称，如果是virtual-env，则指定python环境文件名称，tar.gz打包。
- －x Exit分布式TensorFlow有些API需要用户手动退出PS，在这种情况下用户可以指定-x选项，当所有worker完成训练并成 功后，PS节点自动退出。
- -enable_tensorboard是否在启动训练任务的同时启动TensorBoard。
- -log_tensorboard如果训练同时启动TensorBoard，需要指定TensorBoard日志位置，需要时HDFS上的位置。
- -conf CONF Hadoop conf位置，默认可以不设，使用EMR默认配置。

- －f FILES运行TensorFlow所有依赖的文件和文件夹，包含执行脚本，如果设置virtual-env执行的virtual-env文件，用户可 以将所有依赖放置到一个文件夹中，脚本会自动将文件夹按照文件夹层次关系上传到HDFS中。
- -pn TensorFlow启动的参数服务器个数。
- -pc每个参数服务器申请的CPU核数。
- -pm每个参数服务器申请的内存大小。
- -wn TensorFlow启动的Worker节点个数。
- -wc每个Worker申请的CPU核数。
- -wg每个Worker申请的GPU核数。
- -wm每个Worker申请的内存个数。
- -c COMMAND执行的命令，如pythoncensus.py。

进阶选项，用户需要谨慎使用进阶选项，可能造成任务失败。

- -wnpg每个GPU核上同时跑的worker数量(针对tensorflow-ps)。
- -ppn 每个GPU核上同时跑的worker数量(针对horovod) 以上两个选项指的是单卡多进程操作，由于共用一张显卡，需要在程序上进行一定限制，否则会造成显卡OOM。

11.10 Knox 使用说明

目前E-MapReduce中支持了*Apache Knox*，选择支持Knox的镜像创建集群，完成以下准备工作后，即可在公网直接访问Yarn，HDFS，SparkHistory等服务的Web UI。

准备工作

- 开启Knox公网IP访问
  1. 在E-MapReduce上Knox的服务端口是8443，在集群详情中找到集群所在的ECS安全组。
  2. 在ECS控制台修改对应的安全组，在入方向添加一条规则，打开8443端口。

  > 注意：
  >
  > - 为了安全原因，这里设置的授权对象必须是您的一个有限的ip段范围，禁止使用 0.0.0.0/0。
  > - 打开安全组的8443端口之后，该安全组内的所有机器均会打开公网入方向的 8443 端口，包括非E-MapReduce的ecs机器。
- 设置Knox用户

访问Knox时需要对身份进行验证，会要求您输入用户名和密码。Knox的用户身份验证基于LDAP，您可以使用自有LDAP服务，也可以使用集群中的Apache Directory Server的LDAP服务。

- 使用集群中的LDAP服务

  方式一（推荐）

  在用户管理中直接添加Knox访问账号。

  方式二

  1. ssh登录到集群上，详细步骤请参考*SSH*登录集群。
  2. 准备您的用户数据，如：Tom。将文件中所有的**emr-guest**替换为 Tom，将 **cn:EMR GUEST** 替换为 cn:Tom，设置 **userPassword** 的值为您自己的密码。

     ```
     su knox
     cd /usr/lib/knox-current/templates
     vi users.ldif
     ```

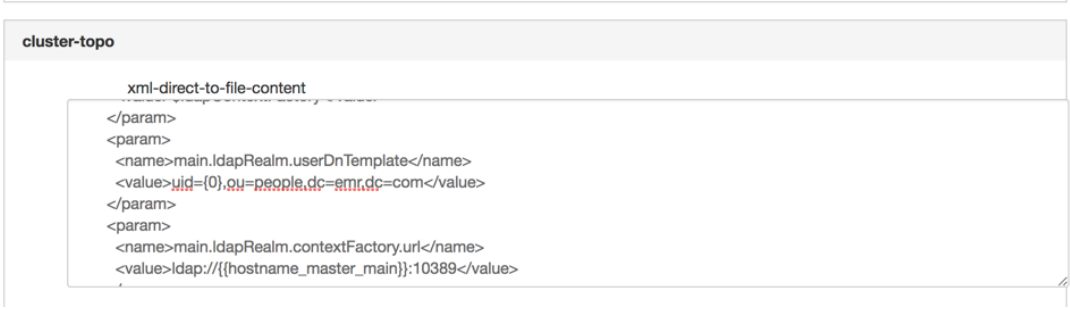
     
     注意：

     出于安全原因，导入前务必修改users.ldif的用户密码，即：设置**userPassword**的值为您自己的用户密码。
  3. 导入到LDAP。

     ```
     su knox
     cd /usr/lib/knox-current/templates
     sh ldap-sample-users.sh
     ```

- 使用自有LDAP服务的情况

  1. 在集群配置管理中找到KNOX的配置管理，在cluster-topo配置中设置两个属性：**main.ldapRealm.userDnTemplate**与**main.ldapRealm.contextFactory.url**。main.ldapRealm.userDnTemplate设置为自己的用户DN模板，**main.ldapRealm.contextFactory.url**设置为自己的LDAP服务器域名和端口。设置完成后保存并重启Knox。



2. 一般自己的LDAP服务不在集群上运行，所以需要开启Knox访问公网LDAP服务的端口，如：10389。参考8443端口的开启步骤，选择出方向。

> 注意：
> 为了安全原因，这里设置的授权对象必须是您的Knox所在集群的公网ip，禁止使用0.0.0.0/0。

开始访问Knox

- 使用E-MapReduce快捷链接访问
  1. 登录阿里云 *E-MapReduce* 控制台。
  2. 单击对应的集群ID
  3. 在左侧导航栏中单击**集群与服务管理**。
  4. 单击相应的服务，如：HDFS，Yarn等。
  5. 单击右上角的**快捷链接**。
- 使用集群公网ip地址访问
  1. 通过集群详情查看公网ip。
  2. 在浏览器中访问相应服务的URL。
     - HDFS UI：*https://{集群公网ip}:8443/gateway/cluster-topo/hdfs/*
     - Yarn UI: *https://{集群公网ip}:8443/gateway/cluster-topo/yarn/*
     - SparkHistory UI : *https://{集群公网ip}:8443/gateway/cluster-topo/sparkhistory/*
     - Ganglia UI: *https://{集群公网ip}:8443/gateway/cluster-topo/ganglia/*
     - Storm UI: *https://{集群公网ip}:8443/gateway/cluster-topo/storm/*
     - Oozie UI: *https://{集群公网ip}:8443/gateway/cluster-topo/oozie/*
  3. 浏览器显示**您的链接不是私密链接**，是因为Knox服务使用了自签名证书，请再次确认访问的是自己集群的ip，且端口为8443。单击**高级 > 继续前往**。

4. 在登录框中输入您在LDAP中设置的用户名和密码

用户权限管理（**ACLs**）

Knox提供服务级别的权限管理，可以限制特定的用户，特定的用户组和特定的ip地址访问特定的服务，可以参考*Apache Knox* 授权。

- 示例：
  - 场景：Yarn UI只允许用户Tom访问。
  - 步骤：在集群配置管理中找到KNOX的配置管理，找到cluster-topo配置，在cluter-topo配置的`<gateway>...</gateway>`标签之间添加ACLs代码：

```
<provider>
      <role>authorization</role>
      <name>AclsAuthz</name>
      <enabled>true</enabled>
      <param>
          <name>YARNUI.acl</name>
          <value>Tom;*;*</value>
      </param>
</provider>
```

- 注意事项：

  Knox会开放相应服务的REST API，用户可以通过各服务的REST API操作服务，如：HDFS的文件添加，删除等。出于安全原因，请务必确保在ecs控制台开启安全组Knox端口8443时，授权对象必须是您的一个有限ip地址段，禁止使用0.0.0.0/0；请勿使用Knox安装目录下的LDAP用户名和密码作为Knox的访问用户。

11.11 Flume使用说明

E-MapReduce从3.16.0版本开始支持Apache Flume。本文介绍使用Flume将数据从EMR Kafka集群同步至EMR Hadoop集群的HDFS、Hive和HBase，以及阿里云的OSS。

准备工作

- 创建Hadoop集群时，在可选服务中选择Flume。
- 创建Kafka集群，并创建名称为flume-test的topic，用于生成数据。

说明：

- 如果创建的是Hadoop高安全集群，消费标准Kafka集群的数据，参照兼容*MIT Kerberos*认证在Hadoop集群配置Kerberos认证；

- 如果创建的是Kafka高安全集群，通过Flume将数据写入标准Hadoop集群，参见*Kerberos Kafka Source*部分；
- 如果创建的Hadoop集群和Kafka集群都是高安全集群，参照跨域互信进行配置，参见跨域互信使用*Flume*；

Kafka->HDFS

- 配置Flume

  创建配置文件*flume.properties*，添加如下配置。其中，配置项**a1.sources.source1.kafka.bootstrap.servers**为Kafka集群broker的host和端口号，**a1.sources.source1.kafka.topics**为Flume消费Kafka数据的topic，**a1.sinks.k1.hdfs.path**为Flume向HDFS写入数据的路径：

```
a1.sources = source1
a1.sinks = k1
a1.channels = c1

a1.sources.source1.type = org.apache.flume.source.kafka.KafkaSource
a1.sources.source1.channels = c1
a1.sources.source1.kafka.bootstrap.servers = kafka-host1:port1,kafka
-host2:port2...
a1.sources.source1.kafka.topics = flume-test
a1.sources.source1.kafka.consumer.group.id = flume-test-group

# Describe the sink
a1.sinks.k1.type = hdfs
a1.sinks.k1.hdfs.path = /tmp/flume/test-data
a1.sinks.k1.hdfs.fileType=DataStream

# Use a channel which buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 100
a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel
a1.sources.source1.channels = c1
a1.sinks.k1.channel = c1
```

- 启动服务

  Flume的默认配置文件放在*/etc/ecm/flume-conf*下，使用该配置启动Flume Agent：

```
flume-ng agent --name a1 --conf /etc/ecm/flume-conf  --conf-file
flume.properties
```

  启动Agent后，因为使用了*/etc/ecm/flume-conf*下的*log4j.properties*，会在当前路径下生成日志logs/flume.log，可根据实际使用对*log4j.properties*进行配置。

- 测试

在Kafka集群使用kafka-console-producer.sh输入测试数据abc

```
[root@emr-header-1 ~]# kafka-console-producer.sh --topic flume-test --broker-list emr-header-1:9092
>abc
>
```

Flume会在HDFS中以当前时间的(毫秒)时间戳生成文件FlumeData.xxxx，查看文件内容，会看到在Kafka中输入的数据

```
[root@emr-header-1 ~]# hdfs dfs -cat /tmp/flume/test-data/FlumeData.1543386053173
abc
[root@emr-header-1 ~]#
```

Kafka->Hive

- 创建Hive表

  Flume使用事务操作将数据写入Hive，需要在创建Hive表时设置**transactional**属性，如创建flume_test表：

  ```
  create table flume_test (id int, content string)
  clustered by (id) into 2 buckets
  stored as orc  TBLPROPERTIES('transactional'='true');
  ```

- 配置Flume

  创建配置文件*flume.properties*，添加如下配置。其中，配置项**a1.sources.source1.kafka.bootstrap.servers**填写Kafka集群broker的host和端口号，**a1.sinks.k1.hive.metastore**为Hive metastore URI，配置为*hive-site.xml*中配置项**hive.metastore.uris**的值:

  ```
  a1.sources = source1
  a1.sinks = k1
  a1.channels = c1

  a1.sources.source1.type = org.apache.flume.source.kafka.KafkaSource
  a1.sources.source1.channels = c1
  a1.sources.source1.kafka.bootstrap.servers = kafka-host1:port1,kafka
  -host2:port2...
  a1.sources.source1.kafka.topics = flume-test
  a1.sources.source1.kafka.consumer.group.id = flume-test-group

  # Describe the sink
  a1.sinks.k1.type = hive
  a1.sinks.k1.hive.metastore = thrift://xxxx:9083
  a1.sinks.k1.hive.database = default
  a1.sinks.k1.hive.table = flume_test
  a1.sinks.k1.serializer = DELIMITED
  a1.sinks.k1.serializer.delimiter = ","
  a1.sinks.k1.serializer.serdeSeparator = ','
  a1.sinks.k1.serializer.fieldnames =id,content
  ```

```
a1.channels.c1.type = memory
a1.channels.c1.capacity = 100
a1.channels.c1.transactionCapacity = 100

a1.sources.source1.channels = c1
a1.sinks.k1.channel = c1
```

- 启动Flume

```
flume-ng agent --name a1 --conf /etc/ecm/flume-conf  --conf-file
flume.properties
```

- 生成数据

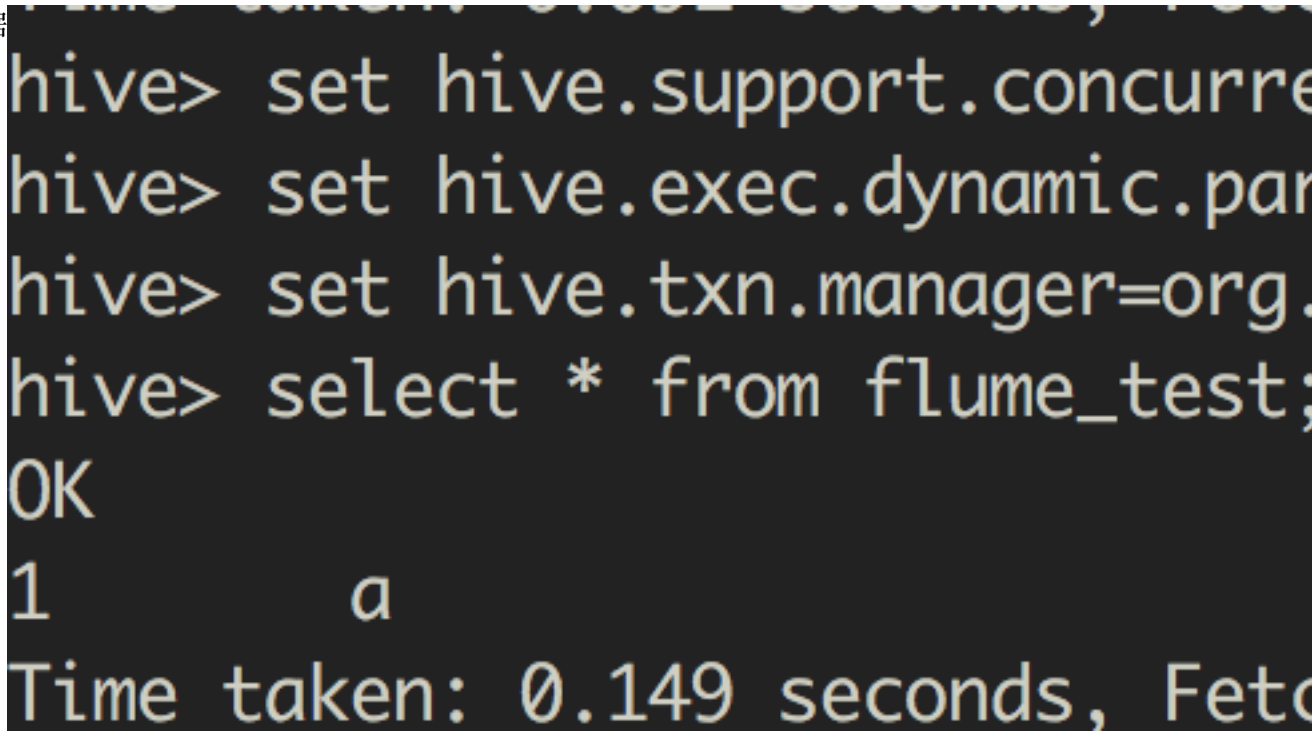
  在Kafka集群中使用*kafka-console-producer.sh*，以逗号为分隔符输入测试数据 1,a

- 检测数据写入

  查询Hive事务表需要在客户端进行配置：

```
hive.support.concurrency – true
hive.exec.dynamic.partition.mode – nonstrict
hive.txn.manager – org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
```

  配置好后查询flume_test表中的数据



Kafka->HBase

- 创建HBase表

  创建HBase表flume_test及列簇column

```
hbase(main):001:0> create 'flume_test', 'column'
0 row(s) in 1.3940 seconds

=> Hbase::Table - flume_test
```

- 配置Flume

  创建配置文件*flume.properties*，添加如下配置。其中，配置项**a1.sources.source1.kafka.bootstrap.servers**为Kafka集群broker的host和端口号，**a1.sinks.k1.table**为HBase表名，**a1.sinks.k1.columnFamily**为列簇名：

```
a1.sources = source1
a1.sinks = k1
a1.channels = c1

a1.sources.source1.type = org.apache.flume.source.kafka.KafkaSource
a1.sources.source1.channels = c1
a1.sources.source1.kafka.bootstrap.servers = kafka-host1:port1,kafka
-host2:port2...
a1.sources.source1.kafka.topics = flume-test
a1.sources.source1.kafka.consumer.group.id = flume-test-group

a1.sinks.k1.type = hbase
a1.sinks.k1.table = flume_test
a1.sinks.k1.columnFamily = column


# Use a channel which buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel
a1.sources.source1.channels = c1
a1.sinks.k1.channel = c1
```

- 启动服务

```
flume-ng agent --name a1 --conf /etc/ecm/flume-conf  --conf-file
flume.properties
```

- 测试

  在Kafka集群使用kafka-console-producer.sh生成数据后，就可以在HBase查到数据

```
=> ["flume_test"]
hbase(main):003:0> scan 'flume_test'
ROW                   COLUMN+CELL
 defaultf2add0ee-5040-4 column=column:pCol, timestamp=1543493834351, value=data
 7dc-b002-f269b679977b
 incRow                column=column:iCol, timestamp=1543493834373, value=\x00\x00\x00\
                       x00\x00\x00\x00\x01
2 row(s) in 0.0310 seconds
```

Kafka->OSS

- 创建OSS路径

  创建OSS Bucket及目录，如*oss://flume-test/result*

- 配置Flume

  Flume向OSS写入数据时，需要占用较大的JVM内存，可以改小OSS缓存或者增大Flume Agent的Xmx：

  - 修改OSS缓存大小

    将hdfs-site.xml配置文件从*/etc/ecm/hadoop-conf*拷贝至*/etc/ecm/flume-conf*，改小配置项**smartdata.cache.buffer.size**的值，如修改为1048576。

  - 修改Xmx

    在Flume的配置路径*/etc/ecm/flume-conf*下，复制配置文件*flume-env.sh.template*并重命名为*flume-env.sh*，设置Xmx，如设置为1g ：

    ```
    export JAVA_OPTS="-Xmx1g"
    ```

  创建配置文件*flume.properties*，添加如下配置。其中，配置项*a1.sources.source1.kafka.bootstrap.servers*填写kafka集群broker的host和端口号，*a1.sinks.k1.hdfs.path*为OSS路径：

  ```
  a1.sources = source1
  a1.sinks = k1
  a1.channels = c1

  a1.sources.source1.type = org.apache.flume.source.kafka.KafkaSource
  a1.sources.source1.channels = c1
  a1.sources.source1.kafka.bootstrap.servers = kafka-host1:port1,kafka
  -host2:port2...
  a1.sources.source1.kafka.topics = flume-test
  a1.sources.source1.kafka.consumer.group.id = flume-test-group

  a1.sinks.k1.type = hdfs
  a1.sinks.k1.hdfs.path = oss://flume-test/result
  a1.sinks.k1.hdfs.fileType=DataStream

  # Use a channel which buffers events in memory
  a1.channels.c1.type = memory
  a1.channels.c1.capacity = 100
  a1.channels.c1.transactionCapacity = 100

  # Bind the source and sink to the channel
  a1.sources.source1.channels = c1
  a1.sinks.k1.channel = c1
  ```

- 启动Flume

如果配置Flume时修改了OSS缓存大小，使用classpath参数传入OSS相关依赖和配置：

```
flume-ng agent --name a1 --conf /etc/ecm/flume-conf  --conf-file
flume.properties  --classpath "/opt/apps/extra-jars/*:/etc/ecm/flume
-conf/hdfs-site.xml"
```

如果修改了Flume Agent的Xmx，只需传入OSS相关依赖：

```
flume-ng agent --name a1 --conf /etc/ecm/flume-conf  --conf-file
flume.properties  --classpath "/opt/apps/extra-jars/*"
```

- 测试

  在Kafka集群使用*kafka-console-producer.sh*生成数据后，在OSS的*oss://flume-test/result*路径下会以当前时间的(毫秒)时间戳为后缀生成文件*FlumeData.xxxx*

Kerberos Kafka source

消费高安全Kafka集群的数据时，需要做额外的配置：

- 参照兼容*MIT Kerberos*认证在Kafka集群配置Kerberos认证，将生成的keytab文件*test.keytab*拷贝至Hadoop集群的*/etc/ecm/flume-conf*路径下；将Kafka集群的*/etc/ecm/has-conf/krb5.conf*文件拷贝至Hadoop集群的/etc/ecm/flume-conf路径下。
- 配置*flume.properties*

  在*flume.properties*中添加如下配置：

  ```
  a1.sources.source1.kafka.consumer.security.protocol = SASL_PLAINTEXT
  a1.sources.source1.kafka.consumer.sasl.mechanism = GSSAPI
  a1.sources.source1.kafka.consumer.sasl.kerberos.service.name = kafka
  ```

- 配置Kafka client

  - 在*/etc/ecm/flume-conf*下创建文件*flume_jaas.conf*，内容如下：

    ```
    KafkaClient {
      com.sun.security.auth.module.Krb5LoginModule required
      useKeyTab=true
      storeKey=true
      keyTab="/etc/ecm/flume-conf/test.keytab"
      serviceName="kafka"
      principal="test@EMR.${realm}.COM";
    };
    ```

    其中，**${realm}**替换为Kafka集群的Kerberos realm。获取方式为，在Kafka集群执行命令**hostname**，得到形式为*emr-header-1.cluster-xxx*的主机名ame，如e*mr-header-1.cluster-123456*，最后的数字串123456即为realm

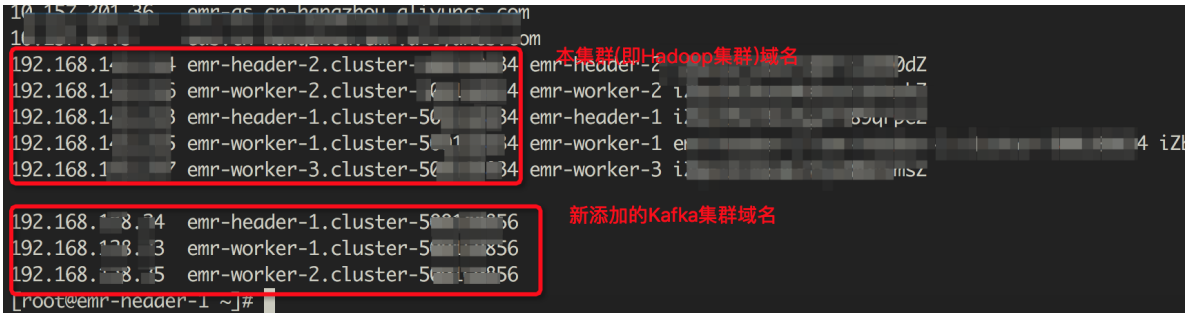
  - 修改*/etc/ecm/flume-conf/flume-env.sh*

初始情况下，*/etc/ecm/flume-conf/*下没有*flume-env.sh*文件，需要拷贝*flume-env.sh.template*并重命名为*flume-env.sh*。添加如下内容：

```
export JAVA_OPTS="$JAVA_OPTS -Djava.security.krb5.conf=/etc/ecm/
flume-conf/krb5.conf"
export JAVA_OPTS="$JAVA_OPTS -Djava.security.auth.login.config=/
etc/ecm/flume-conf/flume_jaas.conf"
```

- 设置域名

  将Kafka集群各节点的长域名和IP的绑定信息添加到Hadoop集群的*/etc/hosts*。长域名的形式例如*emr-header-1.cluster-123456*



跨域互信使用Flume

在配置了跨域互信后，其他配置如下：

- 参照兼容*MIT Kerberos*认证在Kafka集群配置Kerberos认证，将生成的keytab文件*test.keytab*拷贝至Hadoop集群的*/etc/ecm/flume-conf*路径下。
- 配置*flume.properties*

  在*flume.properties*中添加如下配置：

```
a1.sources.source1.kafka.consumer.security.protocol = SASL_PLAINTEXT
a1.sources.source1.kafka.consumer.sasl.mechanism = GSSAPI
a1.sources.source1.kafka.consumer.sasl.kerberos.service.name = kafka
```

- 配置Kafka client

  - 在*/etc/ecm/flume-conf*下创建文件*flume_jaas.conf*，内容如下：

```
KafkaClient {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  storeKey=true
  keyTab="/etc/ecm/flume-conf/test.keytab"
  serviceName="kafka"
  principal="test@EMR.${realm}.COM";
```

```
};
```

其中，**${realm}**替换为Kafka集群的Kerberos realm。获取方式为，在Kafka集群执行命令**hostname**，得到形式为*emr-header-1.cluster-xxx*的主机名，如e*mr-header-1.cluster-123456*，最后的数字串123456即为realm

- 修改*/etc/ecm/flume-conf/flume-env.sh*

  初始情况下，*/etc/ecm/flume-conf/*下没有*flume-env.sh*文件，需要拷贝*flume-env.sh.template*并重命名为*flume-env.sh*。添加如下内容：

```
export JAVA_OPTS="$JAVA_OPTS -Djava.security.auth.login.config=/
etc/ecm/flume-conf/flume_jaas.conf""
```

12 OSS 数据权限隔离

OSS 数据权限隔离

操作步骤

E-MapReduce 支持使用 RAM 来隔离不同子账号的数据。操作步骤如下所示：

1. 登录阿里云 *RAM* 的管理控制台。
2. 在RAM中创建子账号，具体流程请参见如何在 *RAM* 中创建子账号。
3. 单击页面左侧的策略管理，进入授权策略管理界面。
4. 单击自定义授权策略页签。
5. 单击页面右上方的新建授权策略按钮，即进入创建授权策略界面，然后按照提示步骤进行创建。

   您需要多少套不同的权限控制，就创建多少个策略。

   假设您需要以下 2 套数据控制策略：

   - 测试环境， bucketname：test-bucket。其所对应的完整策略如下：

```
{
"Version": "1",
"Statement": [
{
"Effect": "Allow",
"Action": [
  "oss:ListBuckets"
],
"Resource": [
  "acs:oss:*:*:*"
]
},
{
"Effect": "Allow",
"Action": [
  "oss:Listobjects",
  "oss:GetObject",
  "oss:PutObject",
  "oss:DeleteObject"
],
"Resource": [
  "acs:oss:*:*:test-bucket",
  "acs:oss:*:*:test-bucket/*"
]
}
]
}
```

   - 生产环境， bucketname：prod-bucket。其所对应的完整策略如下：

```
{
"Version": "1",
```

```
"Statement": [
{
"Effect": "Allow",
"Action": [
  "oss:ListBuckets"
],
"Resource": [
  "acs:oss:*:*:*"
]
},
{
"Effect": "Allow",
"Action": [
  "oss:Listobjects",
  "oss:GetObject",
  "oss:PutObject"
],
"Resource": [
  "acs:oss:*:*:prod-bucket",
  "acs:oss:*:*:prod-bucket/*"
]
}
]
}
```

6. 单击页面左侧的**用户管理**。
7. 找到需要将策略赋给的子账号条目，单击其右侧的**管理**按钮，进入用户管理页面。
8. 单击页面左侧的**用户授权策略**。
9. 单击右上角的**编辑授权策略**按钮，进入策略授权页面。
10. 选择并添加授权策略。
11. 单击**确定**，完成对子账号的策略授权。
12. 单击用户管理页面左侧的**用户详情**，进入子账号的用户详情页面。
13. 在 Web 控制台登录管理栏中，单击**启用控制台登录**，以打开子账号的登录控制台的权限。

完成并使用

完成以上所有步骤以后，使用对应的子账号登录 E-MapReduce，会有以下限制：

- 在创建集群、创建作业和创建执行计划的 OSS 选择界面，可以看到所有的 bucket，但是只能进入被授权的 bucket。
- 只能看到被授权的 bucket 下的内容，无法看到其他 bucket 内的内容。
- 作业中只能读写被授权的 bucket，读写未被授权的 bucket 会报错。

13 SSH 登录集群

若您觉得在网页上的作业和执行计划无法满足您更加复杂的应用需求，您可以登录到 E-MapReduce 集群的主机上，找到集群的详情页，其中就有集群 master 机器的公网 IP 地址，您可以直接 SSH 登录到这台机器上，查看各种设置与状态。

登录集群内部的用途

机器上已为您设置好相关的环境变量，其中包括以下常用的环境变量：

- JAVA_HOME
- HADOOP_HOME
- HADOOP_CONF_DIR
- HADOOP_LOG_DIR
- YARN_LOG_DIR
- HIVE_HOME
- HIVE_CONF_DIR
- PIG_HOME
- PIG_CONF_DIR

您可以在脚本中直接引用这些变量，但请不要去修改这些变量的值，以免造成 E-MapReduce 的意外错误。

登录 master 主机步骤

1. 使用如下命令 SSH 登录到 master 主机。请在集群详情页的主机信息栏中获取集群 master 机器的公网 IP。

   ```
   ssh root@ip.of.master
   ```

2. 输入创建时设定的密码。

打通本地机器与集群 master 机器的 SSH 无密码登录

通常，您需要登录到集群上进行一些管理和操作。为便于登录集群 master 机器，您可打通与 master 机器的 SSH 无密码登录（集群 master 机器默认开通了公网 IP）。操作步骤如下：

1. 通过上面提到的 root + 密码的方式登录到 master 主机。
2. 切换到到Hadoop 用户或者 hdfs 用户。

Linux 的 SSH 方式

1. 复制私钥到本地。

```
sz ~/.ssh/id_rsa
```

2. 回到您的本地机器，尝试重新登录 master 机器。

```
ssh -i 私钥存放路径/id_rsa hadoop@120.26.221.130
```

当然如果你只有这一个私钥，也可以直接放到你的 *~/.ssh/* 下，默认使用这个私钥，就不需要 `-i` 指定了。

Windows 的 SSH 方式

在 Windows 下你可以有多种方式来使用 SSH 免密码登录 master 机器。

- 方式一：使用 PuTTY

1. 下载 *PuTTY*。
2. 在同样的位置下载 PuTTYgen。
3. 打开 PuTTYgen，并 Load 您的私钥。

> 注意：
> 请妥善保管这个私钥，保证该私钥的安全。若私钥不幸泄漏了，请立刻重新生成一个新的取代。

4. 使用默认的配置，并 Save private key。会保存出一个后缀为 ppk 的 PuTTY 使用的密钥文件。
5. 运行 PuTTY，并在配置页面选择 Session。
6. 输入您要连接的目标机器公网 IP 地址，要加上登录使用的用户名，类似 hadoop@MasterNodeIP。
7. 在配置页面，选择**Connetion** > **SSH** > **Auth**。
8. 选择之前生成好的 ppk 文件。
9. 最后单击 **Open**，就会自动登录到 master 节点了。

- 方式二：使用 Cygwin | MinGW

这是在 Windows 上模拟 Linux 的非常方便的工具，使用起来也非常简单。

如果采用这种方式，连接过程就可以参考上面的 Linux 的 SSH 方式了。

推荐采用 MinGW 的方式，这个是最小巧的一种方式。如果官网打不开，可以下载 git 的客户端，默认带的 Git Bash 就可以满足。

查看 Hadoop、Spark、Ganglia 等系统的 webui

> 说明：
> 在进行本步骤前，请确认您已经完成了上面的 *SSH* 无密码登录流程。

由于安全的缘故，E-MapReduce 集群的 Hadoop、Spark 和 Ganglia 等系统的 webui 监控系统的端口都没有对外开放。如果用户想要访问这些 webui，需要建立一个 SSH 隧道，通过端口转发的方式来达到目的。有如下两种方式：

> 注意：
> 下面的操作是在您本地机器上完成的，不是集群内部机器。

- 方式一：端口动态转发

  创建一个 SSH 隧道，该隧道可打通您本地机器跟 E-MapReduce 集群的 master 机器的某个动态端口的连接。

  ```
  ssh -i /path/id_xxx -ND 8157 hadoop@masterNodeIP
  ```

  8157 是您本地机器没有被使用过的任何一个端口，用户可以自定定义。

  完成动态转发以后，您可以选择如下两种方式来查看。

  - 推荐方式

    推荐使用 Chrome 浏览器，可以使用如下的方式来访问 Web UI：

    ```
    chrome --proxy-server="socks5://localhost:8157" --host-resolver-
    rules="MAP * 0.0.0.0 , EXCLUDE localhost" --user-data-dir=/tmp/
    ```

    若是 Windows 系统，这里的 *tmppath* 可以写成类似 *d:/tmppath*，若是 Linux 或者 OSX，可以直接写成 */tmp/*。

    在不同的操作系统中，Chrome 的位置不同，请参见下表：

操作系统	**Chrome** 位置
Mac OS X	/Applications/Google Chrome.app/Contents/MacOS/Google Chrome
Linux	/usr/bin/google-chrome

操作系统	Chrome 位置
Windows	C:\Program Files (x86)\Google\Chrome\Application\chrome.exe

- 插件方式

  此时，您本地机器跟 E-MapReduce 集群的 master 主机的 SSH 通道已经打通，要在浏览器中查看 Hadoop、Spark、Ganglia 的 webui，您还需要配置一个本地代理。操作步骤如下：

  1. 假设您使用的是 Chrome 或者 Firefox 浏览器，请点击下载 *FoxyProxy Standard* 代理软件。
  2. 安装完成并重启浏览器后，打开一个文本编辑器，编辑如下内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<foxyproxy>
<proxies>
<proxy name="aliyun-emr-socks-proxy" id="2322596116" notes
="" fromSubscription="false" enabled="true" mode="manual"
selectedTabIndex="2" lastresort="false" animatedIcons="true
" includeInCycle="true" color="#0055E5" proxyDNS="true"
noInternalIPs="false" autoconfMode="pac" clearCacheBeforeUse
="false" disableCache="false" clearCookiesBeforeUse="false"
rejectCookies="false">
<matches>
<match enabled="true" name="120.*" pattern="http://120.*"
isRegEx="false" isBlackList="false" isMultiLine="false"
caseSensitive="false" fromSubscription="false" ></match>
</matches>
<manualconf host="localhost" port="8157" socksversion="5"
isSocks="true" username="" password="" domain="" ></manualconf>
</proxy>
</proxies>
</foxyproxy>
```

     其中：

     - `Port 8157` 是您本地用来建立与集群 master 机器 SSH 连接的端口，这个需要跟您之前执行的在终端中执行的 SSH 命令中使用的端口匹配。
     - `120.*` 这个匹配是用来匹配 master 主机的 IP 地址，请根据 master 的 IP 地址的情况来定。
  3. 在浏览器中单击**Foxyproxy**按钮，选择 **Options**。
  4. 选择 **Import/Export**。
  5. 选择刚才您编辑的 xml 文件，单击 **Open**。
  6. 在 **Import FoxyProxy Setting** 对话框中，单击 **Add**。

7. 点击浏览器中的 **Foxyproxy** 按钮，选择 **Use Proxy aliyun-emr-socks-proxy for all URLs**。
8. 在浏览器中输入 localhost:8088，就可以打开远端的 Hadoop 界面了。

- 方式二：本地端口转发

> 注意：
> 这个方式的缺陷是只能看到最外层的界面，一旦要看详细的作业信息，就会出错。

```
ssh -i /path/id_rsa -N -L 8157:masterNodeIP:8088 hadoop@masterNodeIP
```

参数说明：

- **path**：私钥存放路径。
- **masterNodeIP**：要连接的 master节点 IP。
- **8088**：是 master 节点上 ResourceManager 的访问端口号。

14 创建Gateway

Gateway是与EMR集群处于同一个内网中的ECS服务器，用户可以使用Gateway实现负载均衡和安全隔离，也可以通过Gateway向E-MapReduce集群提交作业。

您可以通过以下两种方式创建Gateway：

- （推荐）通过阿里云 *E-MapReduce* 控制台创建。
- 手动搭建。

通过E-MapReduce控制台创建Gateway

当前E-MapReduce Gateway仅支持E-MapReduce Hadoop类型的集群。在创建Gateway前，请确保您已经创建了E-MapReduce Hadoop类型集群。创建Gateway，请按照如下步骤进行操作：

1. 登录阿里云 *E-MapReduce* 控制台。
2. 单击创建**Gateway**按钮。
3. 在创建**Gateway**页面中进行配置。
   - 付费类型
     - **包年包月**：一次性支付一段时间的费用，价格相对来说会比较便宜，特别是包三年的时候折扣会比较大。
     - **按量付费**：根据实际使用的小时数来支付费用，每个小时计一次费用。
   - **我的集群**：为该集群创建Gateway，即创建的Gateway可以向哪个集群提交作业。Gateway将会自动配置与该集群一致的Hadoop环境。
   - **配置**：该区域内可选择的ECS实例规格。
   - **系统盘配置**：Gateway节点使用的系统盘类型。系统盘有2种类型：SSD云盘和高效云盘，根据不同机型和不同的Region，系统盘显示类型会有不同。系统盘默认随着集群的释放而释放。
   - **系统盘大小**：最小为40GB，最大为500GB。默认值为300GB。
   - **数据盘配置**：Gateway节点使用的数据盘类型。数据盘有2种类型：SSD云盘和高效云盘，根据不同机型和不同的Region，数据盘显示类型会有不同。数据盘默认随着集群的释放而释放。
   - **数据盘大小**：最小为200GB，最大为4000GB。默认值为300GB。
   - **数量**：数据盘的数量，最小设置为1台，最大设置为10台。

- **集群名称**：创建的Gateway的名称，长度限制为1~64个字符，只允许包含中文、字母、数字、连接号（-）、下划线（_）。
- **密码/密钥对**：
  - **密码**：在文本框中输入登录Gateway的密码。
  - **密钥对**：在下拉菜单中选择登录Gateway的密钥对名称。如果还未创建过密钥对，单击右侧的**创建密钥对**链接，进入ECS控制台创建。请妥善保管好密钥对所对应的私钥文件（.pem文件）。在Gateway创建成功后，该密钥对的公钥部分会自动绑定到Gateway所在的云服务器ECS上。在您通过SSH登录Gateway时，请输入该私钥文件中的私钥。

4. 单击**创建**完成Gateway的创建。

   新创建的Gateway会显示在集群列表中，创建完成后，**状态**列会变为**空闲**状态。

手动搭建Gateway

- 网络环境

  首先要保证Gateway节点在EMR对应集群的安全组中，Gateway节点可以顺利的访问EMR集群。设置节点的安全组请参考创建安全组。

- 软件环境
  - 系统环境：推荐使用CentOS 7.2及以上版本。
  - Java环境：安装JDK 1.7及以上版本，推荐使用OpenJDK version 1.8.0版本。
- 搭建步骤
  - EMR 2.7及以上版本，3.2及以上版本

    这些版本推荐直接使用EMR控制台来创建Gateway。

    如果您选择手动搭建，请先创建一个脚本，脚本内容如下所示，然后在Gataway节点上执行。执行命令为：**`sh deploy.sh <masteri_ip> master_password_file`**。

    - *`deploy.sh`*：脚本名称，内容见下面代码。
    - *`masteri_ip`*：集群的master节点的IP，请确保可以访问。
    - *`master_password_file`*：保存master节点的密码文件，将master节点的密码直接写在文件内即可。

```
#!/usr/bin/bash
if [ $# != 2 ]
then
   echo "Usage: $0 master_ip master_password_file"
```

```
   exit 1;
fi
masterip=$1
masterpwdfile=$2
if ! type sshpass >/dev/null 2>&1; then
   yum install -y sshpass
fi
if ! type java >/dev/null 2>&1; then
   yum install -y java-1.8.0-openjdk
fi
mkdir -p /opt/apps
mkdir -p /etc/ecm
echo "Start to copy package from $masterip to local gateway(/opt/
apps)"
echo " -copying hadoop-2.7.2"
sshpass -f $masterpwdfile scp -r -o 'StrictHostKeyChecking no'
root@$masterip:/usr/lib/hadoop-current /opt/apps/
echo " -copying hive-2.0.1"
sshpass -f $masterpwdfile scp -r root@$masterip:/usr/lib/hive-
current /opt/apps/
echo " -copying spark-2.1.1"
sshpass -f $masterpwdfile scp -r root@$masterip:/usr/lib/spark-
current /opt/apps/
echo "Start to link /usr/lib/\${app}-current to /opt/apps/\${app}"
if [ -L /usr/lib/hadoop-current ]
then
   unlink /usr/lib/hadoop-current
fi
ln -s /opt/apps/hadoop-current  /usr/lib/hadoop-current
if [ -L /usr/lib/hive-current ]
then
   unlink /usr/lib/hive-current
fi
ln -s /opt/apps/hive-current  /usr/lib/hive-current
if [ -L /usr/lib/spark-current ]
then
   unlink /usr/lib/spark-current
fi
ln -s /opt/apps/spark-current /usr/lib/spark-current
echo "Start to copy conf from $masterip to local gateway(/etc/ecm
)"
sshpass -f $masterpwdfile scp -r root@$masterip:/etc/ecm/hadoop-
conf  /etc/ecm/hadoop-conf
sshpass -f $masterpwdfile scp -r root@$masterip:/etc/ecm/hive-conf
 /etc/ecm/hive-conf
sshpass -f $masterpwdfile scp -r root@$masterip:/etc/ecm/spark-
conf /etc/ecm/spark-conf
echo "Start to copy environment from $masterip to local gateway(/
etc/profile.d)"
sshpass -f $masterpwdfile scp root@$masterip:/etc/profile.d/hdfs.
sh /etc/profile.d/
sshpass -f $masterpwdfile scp root@$masterip:/etc/profile.d/yarn.
sh /etc/profile.d/
sshpass -f $masterpwdfile scp root@$masterip:/etc/profile.d/hive.
sh /etc/profile.d/
sshpass -f $masterpwdfile scp root@$masterip:/etc/profile.d/spark.
sh /etc/profile.d/
if [ -L /usr/lib/jvm/java ]
then
   unlink /usr/lib/jvm/java
fi
echo "" >>/etc/profile.d/hdfs.sh
```

```
echo export JAVA_HOME=/usr/lib/jvm/jre-1.8.0 >>/etc/profile.d/hdfs
.sh
echo "Start to copy host info from $masterip to local gateway(/etc
/hosts)"
sshpass -f $masterpwdfile scp root@$masterip:/etc/hosts /etc/
hosts_bak
cat /etc/hosts_bak | grep emr | grep cluster >>/etc/hosts
if ! id hadoop >& /dev/null
then
   useradd hadoop
fi
```

- EMR 2.7以下版本，3.2以下版本

创建一个脚本，脚本内容如下所示，然后在Gataway节点上执行。执行命令为：**sh deploy.sh <masteri_ip> master_password_file**。

- *deploy.sh*：脚本名称，内容见下面代码。
- *masteri_ip*：集群的master节点的IP，请确保可以访问。
- *master_password_file*：保存master节点的密码文件，将master节点的密码直接写在文件内即可。

```
!/usr/bin/bash
if [ $# != 2 ]
then
   echo "Usage: $0 master_ip master_password_file"
   exit 1;
fi
masterip=$1
masterpwdfile=$2
if ! type sshpass >/dev/null 2>&1; then
   yum install -y sshpass
fi
if ! type java >/dev/null 2>&1; then
   yum install -y java-1.8.0-openjdk
fi
mkdir -p /opt/apps
mkdir -p /etc/emr
echo "Start to copy package from $masterip to local gateway(/opt/
apps)"
echo " -copying hadoop-2.7.2"
sshpass -f $masterpwdfile scp -r -o 'StrictHostKeyChecking no'
root@$masterip:/usr/lib/hadoop-current /opt/apps/
echo " -copying hive-2.0.1"
sshpass -f $masterpwdfile scp -r root@$masterip:/usr/lib/hive-
current /opt/apps/
echo " -copying spark-2.1.1"
sshpass -f $masterpwdfile scp -r root@$masterip:/usr/lib/spark-
current /opt/apps/
echo "Start to link /usr/lib/\${app}-current to /opt/apps/\${app}"
if [ -L /usr/lib/hadoop-current ]
then
   unlink /usr/lib/hadoop-current
fi
ln -s /opt/apps/hadoop-current  /usr/lib/hadoop-current
if [ -L /usr/lib/hive-current ]
then
```

```
   unlink /usr/lib/hive-current
fi
ln -s /opt/apps/hive-current  /usr/lib/hive-current
if [ -L /usr/lib/spark-current ]
then
   unlink /usr/lib/spark-current
fi
ln -s /opt/apps/spark-current /usr/lib/spark-current
echo "Start to copy conf from $masterip to local gateway(/etc/emr
)"
sshpass -f $masterpwdfile scp -r root@$masterip:/etc/emr/hadoop-
conf  /etc/emr/hadoop-conf
sshpass -f $masterpwdfile scp -r root@$masterip:/etc/emr/hive-conf
 /etc/emr/hive-conf
sshpass -f $masterpwdfile scp -r root@$masterip:/etc/emr/spark-
conf /etc/emr/spark-conf
echo "Start to copy environment from $masterip to local gateway(/
etc/profile.d)"
sshpass -f $masterpwdfile scp root@$masterip:/etc/profile.d/hadoop
.sh /etc/profile.d/
if [ -L /usr/lib/jvm/java ]
then
   unlink /usr/lib/jvm/java
fi
ln -s /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-3.b12.el7_3.x86_64
/jre /usr/lib/jvm/java
echo "Start to copy host info from $masterip to local gateway(/etc
/hosts)"
sshpass -f $masterpwdfile scp root@$masterip:/etc/hosts /etc/
hosts_bak
cat /etc/hosts_bak | grep emr | grep cluster >>/etc/hosts
if ! id hadoop >& /dev/null
then
   useradd hadoop
fi
```

- 测试

  - Hive

    ```
    [hadoop@iZ23bc05hrvZ ~]$ hive
    hive> show databases;
    OK
    default
    Time taken: 1.124 seconds, Fetched: 1 row(s)
    hive> create database school;
    OK
    Time taken: 0.362 seconds
    hive>
    ```

  - 运行Hadoop作业

    ```
    [hadoop@iZ23bc05hrvZ ~]$ hadoop  jar /usr/lib/hadoop-current/share
    /hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar pi 10 10
    Number of Maps  = 10
    Samples per Map = 10
    Wrote input for Map #0
    Wrote input for Map #1
    Wrote input for Map #2
    Wrote input for Map #3
    Wrote input for Map #4
    ```

```
Wrote input for Map #5
Wrote input for Map #6
Wrote input for Map #7
Wrote input for Map #8
Wrote input for Map #9
  File Input Format Counters
      Bytes Read=1180
  File Output Format Counters
      Bytes Written=97
Job Finished in 29.798 seconds
Estimated value of Pi is 3.20000000000000000000
```

15 MetaService

E-MapReduce环境下提供MetaService服务。基于此服务，您可以在E-MapReduce集群中以免AK的方式访问阿里云资源。

默认应用角色

默认地，您在创建集群时将需要向E-MapReduce服务授权一个应用角色（AliyunEmrEcsDefaultRole）。授权之后，您在E-MapReduce上的作业将可以无需显式输入AK来访问阿里云资源。AliyunEmrEcsDefaultRole默认授予以下权限策略：

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "oss:GetObject",
        "oss:ListObjects",
        "oss:PutObject",
        "oss:DeleteObject",
        "oss:ListBuckets",
        "oss:AbortMultipartUpload"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

所以默认情况下，基于MetaService的作业将只能访问OSS数据。如果您想基于MetaService访问其他阿里云资源，例如LogService等等，则需要给AliyunEmrEcsDefaultRole补充授予相应的权限。以上操作需要登录*RAM*控制台完成。

注意：

当前metaservice服务只支持OSS，LogService和MNS数据的免AK操作。请谨慎编辑，删除默认角色，否则会造成集群创建失败或者作业运行失败。

自定义应用角色

大多数情况下，您只需要使用默认应用角色或者修改默认应用角色即可。E-MapReduce同时支持您使用自定义的应用角色。在创建集群时，您既可以使用默认应用角色，也可以选择自定义应用角色。如何创建角色并授权给服务，请参考*RAM*的相关文档。

访问MetaService

MetaService是一个HTTP服务，您可以直接访问这个HTTP服务来获取相关Meta信息：例如 **`curl http://localhost:10011/cluster-region`**可以获得当前集群所在Region。

当前MetaService支持以下几类信息：

- Region： /cluster-region
- 角色名： /cluster-role-name
- AccessKeyId：/role-access-key-id
- AccessKeySecret：/role-access-key-secret
- SecurityToken：/role-security-token
- 网络类型：/cluster-network-type

使用MetaService

基于MetaSerivce服务，我们可以在作业中免AK地访问阿里云资源，这样可以带来两个优势：

- 降低AK泄漏的风险。基于RAM的使用方式，可以将安全风险降到最低。需要什么权限就给角色授予什么权限，做到权限最小化。
- 提高用户体验。尤其在交互式访问OSS资源时，可以避免写一长串的OSS路径。

下面示例几种使用方式：

```
I. Hadoop命令行查看OSS数据
    旧方式： hadoop fs -ls oss://ZaH******As1s:Ba23N**************
sdaBj2@bucket.oss-cn-hangzhou-internal.aliyuncs.com/a/b/c
    新方式： hadoop fs -ls oss://bucket/a/b/c
II. Hive建表
    旧方式：
        CREATE EXTERNAL TABLE test_table(id INT, name string)
        ROW FORMAT DELIMITED
        FIELDS TERMINATED BY '/t'
        LOCATION 'oss://ZaH******As1s:Ba23N**************sdaBj2@bucket
.oss-cn-hangzhou-internal.aliyuncs.com/a/b/c';
    新方式：
        CREATE EXTERNAL TABLE test_table(id INT, name string)
        ROW FORMAT DELIMITED
        FIELDS TERMINATED BY '/t'
        LOCATION 'oss://bucket/a/b/c';
III. Spark
    旧方式： val data = sc.textFile("oss://ZaH******As1s:Ba23N
**************sdaBj2@bucket.oss-cn-hangzhou-internal.aliyuncs.com/a/b/
c")
    新方式： val data = sc.textFile("oss://bucket/a/b/c")
```

16 交互式工作台

16.1 交互式工作台简介

交互式工作台提供在E-MapReduce管理控制台直接编写并运行spark，sparksql，hivesql任务的能力，您可以在工作台直接看到运行结果。交互式工作台适合处理运行时间较短、想要直接看到数据结果、调试性质的任务，对于运行时间很长，需要定期执行的任务应使用作业和执行计划功能。 本节会介绍如何新建演示任务并运行，其他示例和操作说明请参考后面的章节。

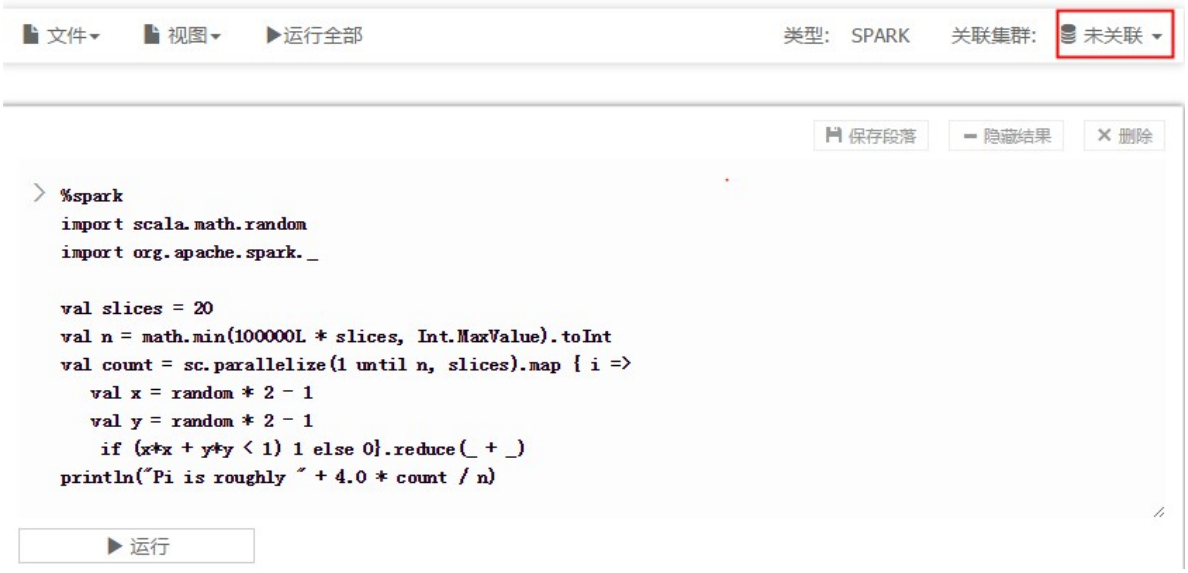
创建演示任务

1. 登录阿里云 *E-MapReduce* 控制台。
2. 单击上方的**老版作业调度**。
3. 在左侧导航栏中单击**交互式工作台**。
4. 单击**新建演示任务**。

5. 弹出确认框，提示运行需要的集群环境，单击**确认创建演示任务**。会新建三个示例的交互式任务。

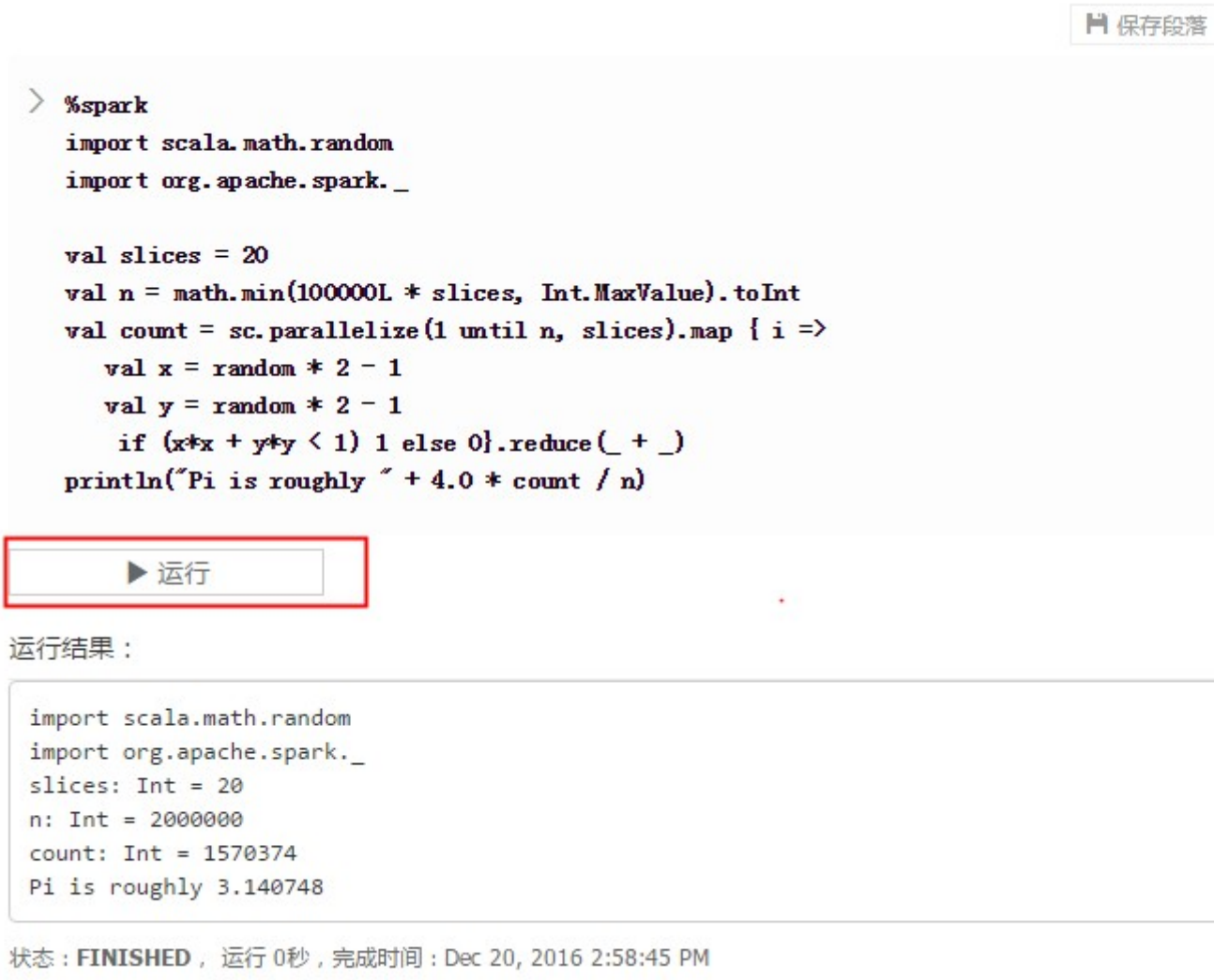
运行Spark演示任务

1. 单击**EMR-Spark-Demo**，显示Spark的交互式示例。运行之前首先要关联一个已经创建好的集群， 单击在可用集群列表中选择一个。注意关联的集群必须是EMR-2.3以上版本，不小于三节点，4核8G即以上配置。



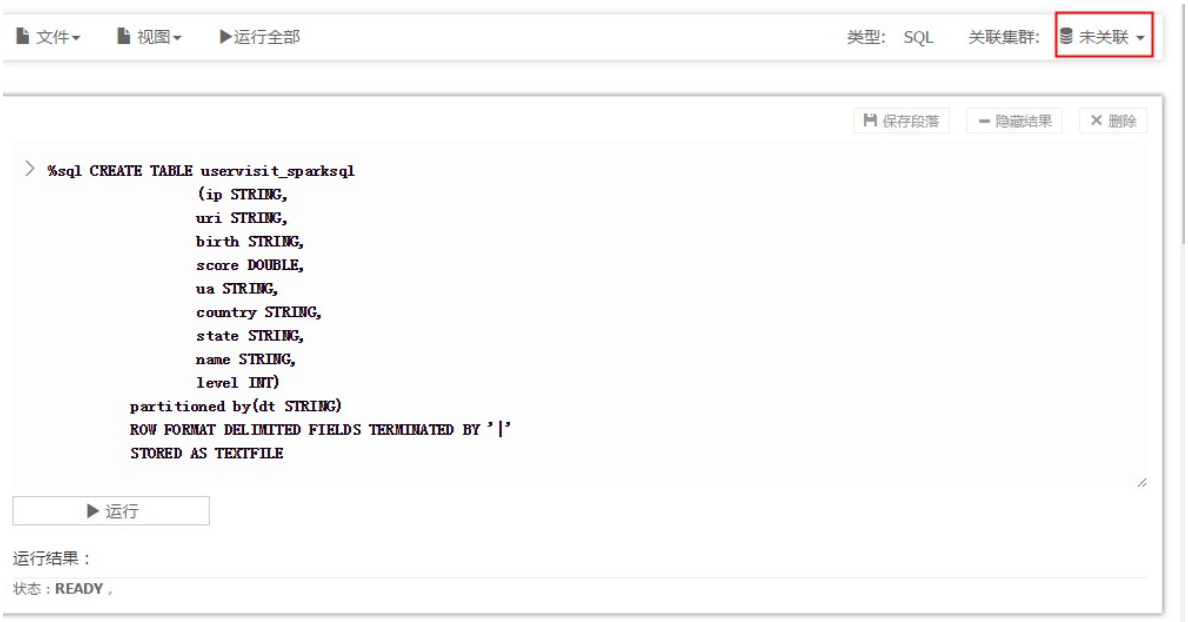
2. 关联后，单击运行。关联的集群第一次执行Spark/SparkSQL交互式任务时会额外花费一些时间构建Spark上下文和运行环境，大概

要1分钟，后续的执行就不需要再耗时构建了。运行结果显示在下方



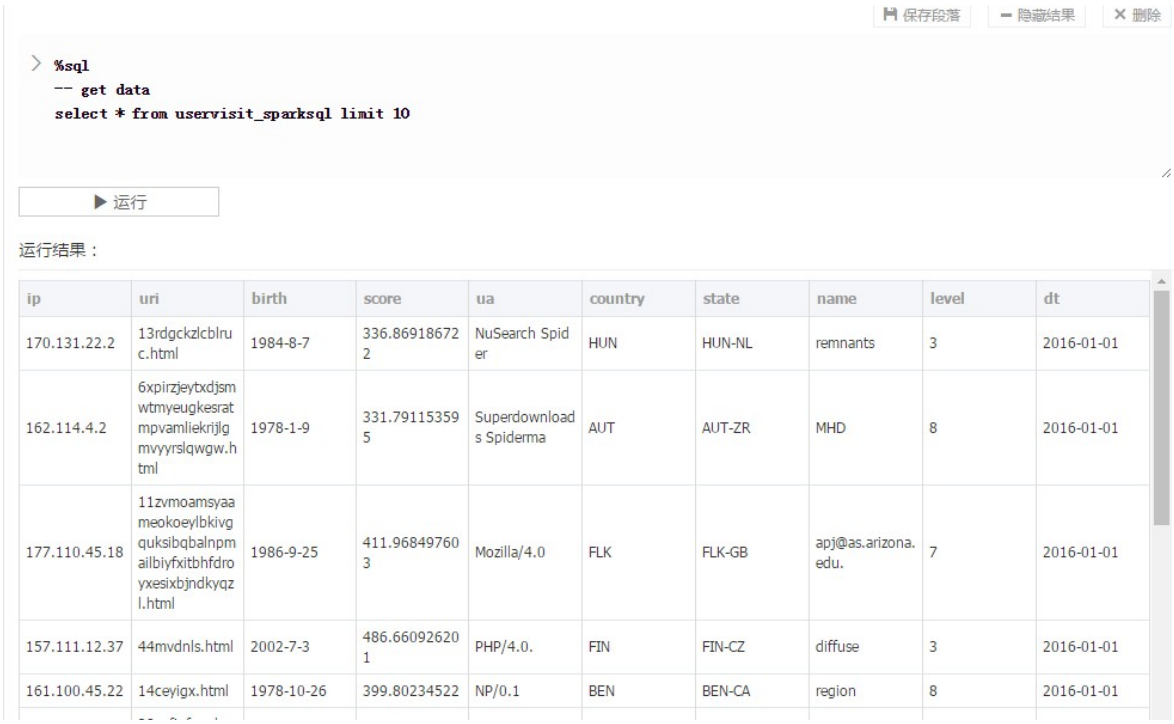
运行**SparkSQL**演示任务

1. 单击**EMR-Spark-Demo**，显示SparkSQL的交互式示例。运行之前依然要先关联一个已经创建好的集群， 单击右上角在可用集群列表中选择一个。



2. SparkSQL的演示任务有好几个演示段落，每个段落可以单独运行，也可以通过运行全部运行。运行后可以看到各段落返回的数据结果。

> **说明：**
> 创建表的段落如果运行多次会报错提示表已存在。

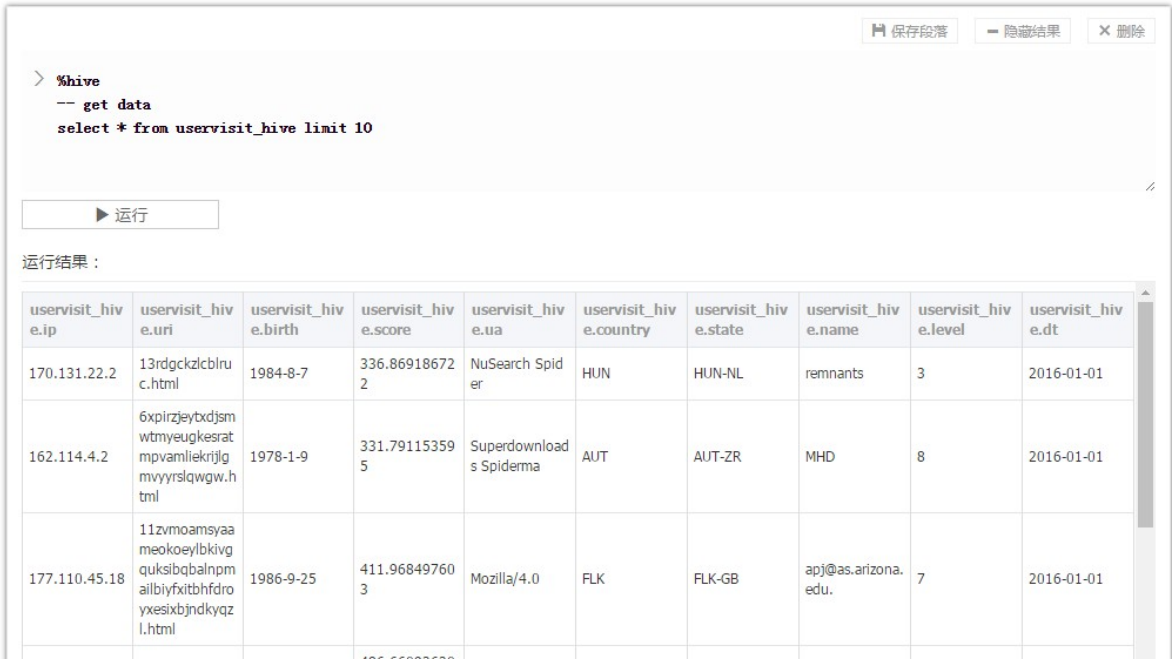


ip	uri	birth	score	ua	country	state	name	level	dt
170.131.22.2	13rdgckzlcblruc.html	1984-8-7	336.869186722	NuSearch Spider	HUN	HUN-NL	remnants	3	2016-01-01
162.114.4.2	6xpirzjeytxdjsmwtmyeugkesratmpvamliekrijlgmvyyrslqwgw.html	1978-1-9	331.791153595	Superdownloads Spiderma	AUT	AUT-ZR	MHD	8	2016-01-01
177.110.45.18	11zvmoamsyaameokoeylbkivgquksibqbalnpmailbiyfxitbhfdroyxesixbjndkyqzl.html	1986-9-25	411.968497603	Mozilla/4.0	FLK	FLK-GB	apj@as.arizona.edu.	7	2016-01-01
157.111.12.37	44mvdnls.html	2002-7-3	486.660926201	PHP/4.0.	FIN	FIN-CZ	diffuse	3	2016-01-01
161.100.45.22	14ceyigx.html	1978-10-26	399.80234522	NP/0.1	BEN	BEN-CA	region	8	2016-01-01

运行Hive演示任务

1. 单击**EMR-Hive-Demo**，显示Hive的交互式示例。运行之前依然要先关联一个已经创建好的集群， 单击右上角在可用集群列表中选择一个。
2. Hive的演示任务有好几个演示段落，每个段落可以单独运行，也可以通过运行全部运行。运行后可以看到各段落返回的数据结果。

> **说明：**
>
> - 关联的集群第一次执行hive交互式任务时会额外花费一些时间构建hive客户端运行环境，大概要几十秒，后续的执行就不需要再耗时构建了。
> - 创建表的段落如果运行多次会报错提示表已存在。



uservisit_hive.ip	uservisit_hive.uri	uservisit_hive.birth	uservisit_hive.score	uservisit_hive.ua	uservisit_hive.country	uservisit_hive.state	uservisit_hive.name	uservisit_hive.level	uservisit_hive.dt
170.131.22.2	13rdgckzlcblruc.html	1984-8-7	336.869186722	NuSearch Spider	HUN	HUN-NL	remnants	3	2016-01-01
162.114.4.2	6xpirzjeytxdjsmwtmyeugkesratmpvamliekrijlgmvyyrslqwgw.html	1978-1-9	331.791153595	Superdownloads Spiderma	AUT	AUT-ZR	MHD	8	2016-01-01
177.110.45.18	11zvmoamsyaameokoeylbkivgquksibqbalnpmailbiyfxitbhfdroyxesixbjndkyqzl.html	1986-9-25	411.968497603	Mozilla/4.0	FLK	FLK-GB	apj@as.arizona.edu.	7	2016-01-01

取消关联集群

集群运行过交互式任务后，为了再次执行时能够快速响应，会创建进程缓存一些上下文运行环境。如果您暂时不再执行交互式任务，想要释放缓存占用的集群资源，可以把运行过的交互式任务都取消关联，会释放掉原关联集群上占用的内存资源。

16.2 交互式工作台操作说明

本文向您介绍，如何在EMR控制台上新建交互式任务，并指导您完成任务的创建和运行。

新建交互式任务

> 说明：
> 要运行交互式任务的集群的配置，必须满足EMR-2.3及以上版本，不小于三节点，4核8G及以上配置。

1. 登录阿里云 *E-MapReduce* 控制台。
2. 单击上方的老版作业调度。
3. 在左侧导航栏中单击交互式工作台。
4. 单击右侧新建交互式任务或文件 > -新建交互式任务。

5. 填入名称，选择默认类型，关联集群可选，单击确认新建一个交互式任务。

类型目前支持三类，Spark可以编写scala spark代码，Spark SQL可以写spark支持的sql语句，Hive可以写Hive支持的sql语句。

6. 关联集群，需要是一个创建好的集群，且必须是EMR-2.3及以上版本，不小于三节点，4核8G及以上配置。也可以先不关联，在运行前再关联。

目前一个账户最多创建20个交互式任务。

填写保存段落

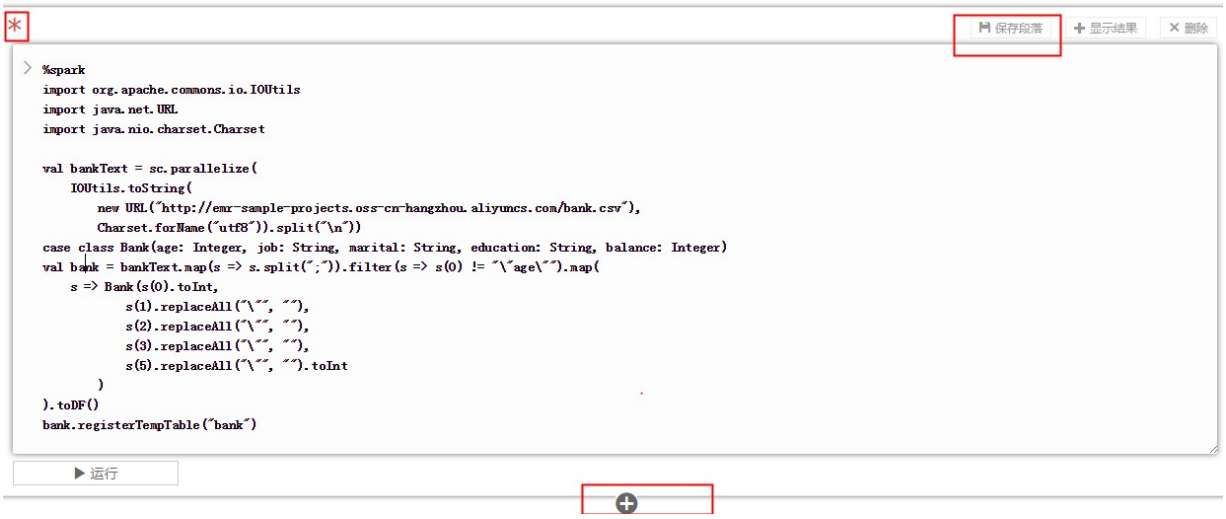
段落是运行任务的最小单元，1个交互式任务可以填写多个段落。每个段落可以在内容开头写`%spark`，`%sql`，`%hive`表明该段落是scala spark代码段，spark sql，还是hive sql。类型前缀以空格或换行和实际内容分割，不写类型前缀则以交互式任务的默认类型作为该段落的运行类型。

一个创建spark临时表的示例如下：

将如下代码粘贴进段落内，会显示一个红*提醒有修改，通过**保存段落**按钮或**运行**按钮可以保存对段落内容的修改，单击段落下方的**+**可以新建一个段落。目前一个交互式任务最多可以创建30个段落。

```
%spark
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
// load bank data
val bankText = sc.parallelize(
    IOUtils.toString(
        new URL("http://emr-sample-projects.oss-cn-hangzhou.aliyuncs.
com/bank.csv"),
        Charset.forName("utf8")).split("\n"))
```

```
case class Bank(age: Integer, job: String, marital: String, education
: String, balance: Integer)
val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "\"age
\"").map(
    s => Bank(s(0).toInt,
            s(1).replaceAll("\"", ""),
            s(2).replaceAll("\"", ""),
            s(3).replaceAll("\"", ""),
            s(5).replaceAll("\"", "").toInt
        )
).toDF()
bank.registerTempTable("bank")
```

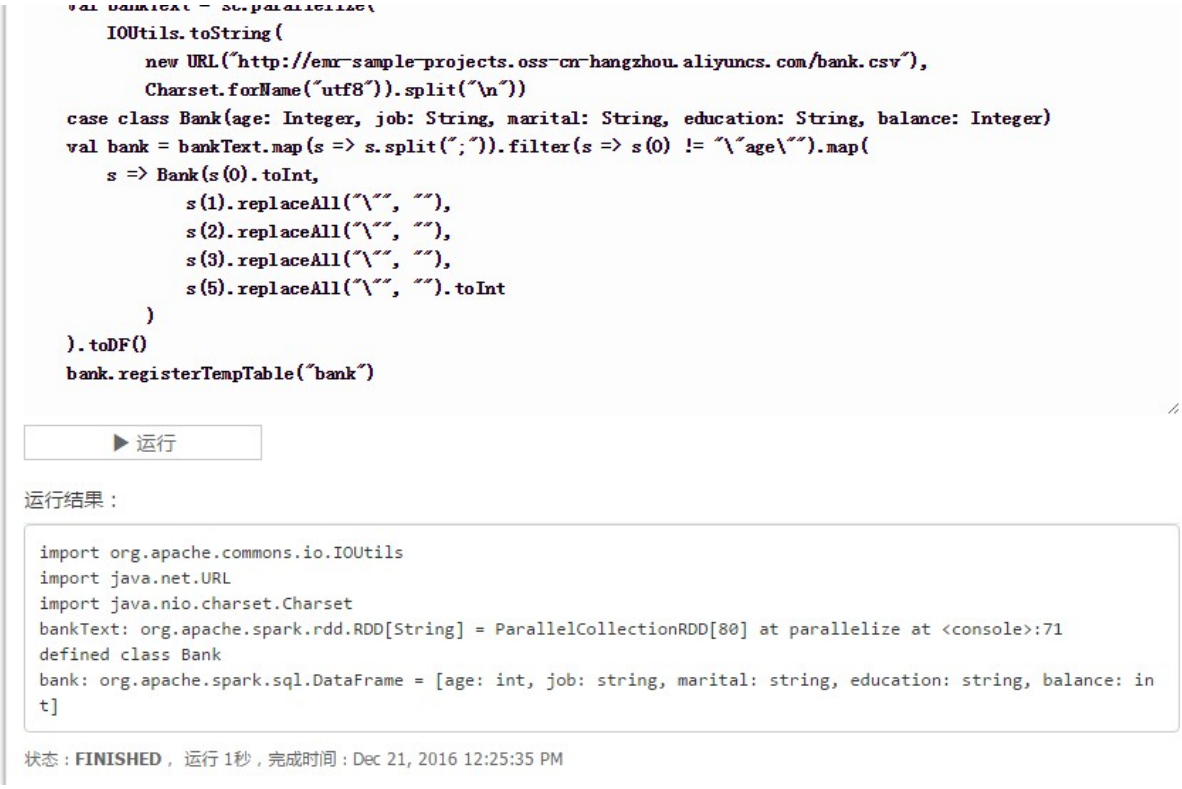


运行段落

运行之前首先要关联一个已经创建好的集群，如果创建交互式任务时未关联，右上角显示未关联，单击在可用集群列表中选择一个。注意关联的集群必须是EMR-2.3以上版本，不小于三节点，4核8G即以上配置。

单击运行按钮，会自动保存当前段落，运行内容，如果这是最后一个段落会自动新建一个段落。

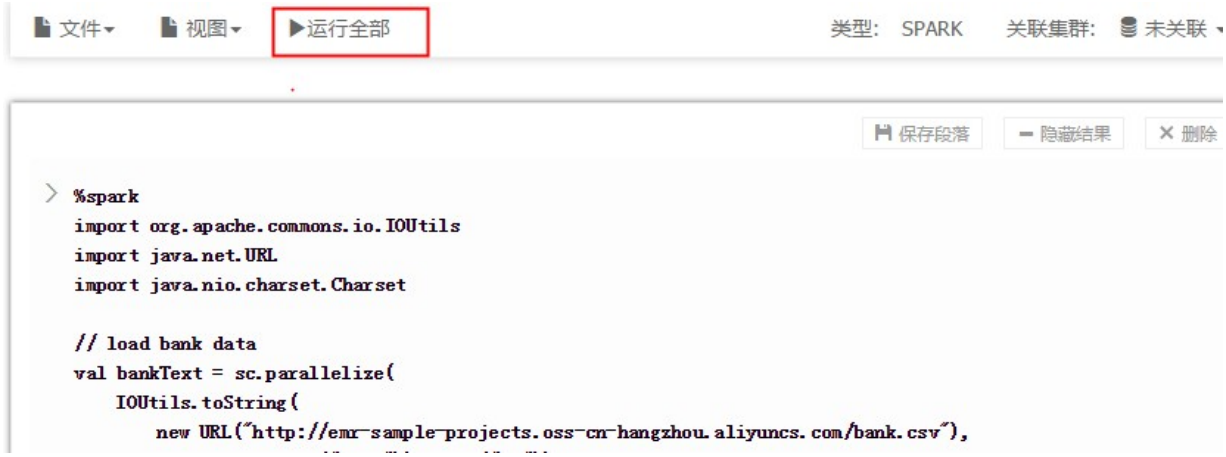
运行后会显示当前的运行状态，还未实际运行的是**PENDING**，运行后是**RUNNING**。运行完成是**FINISHED**，如果有错误是**ERROR**，运行结果会显示在段落的运行按钮下方。运行时可以点运行按钮下方的取消按钮取消运行，取消的状态显示**ABORT**。



段落可以反复多次运行，只保留最后一次运行的结果。运行时不能修改段落的输入内容，运行后可以修改。

运行全部

交互式任务可以单击菜单栏上的运行全部运行所有的段落，段落会顺序提交运行。不同的类型有独立的执行队列,如果一个交互式任务包含多种段落类型，顺序提交运行后，实际在集群上的执行顺序是按照类型划分的。Spark和Spark SQL类型是顺序一个个的执行。HIVE支持并发执行，同一个集群交互式段落最大并发数是10。注意并发运行的作业同时受集群资源限制，集群规模小并发很多依然要在yarn上排队。

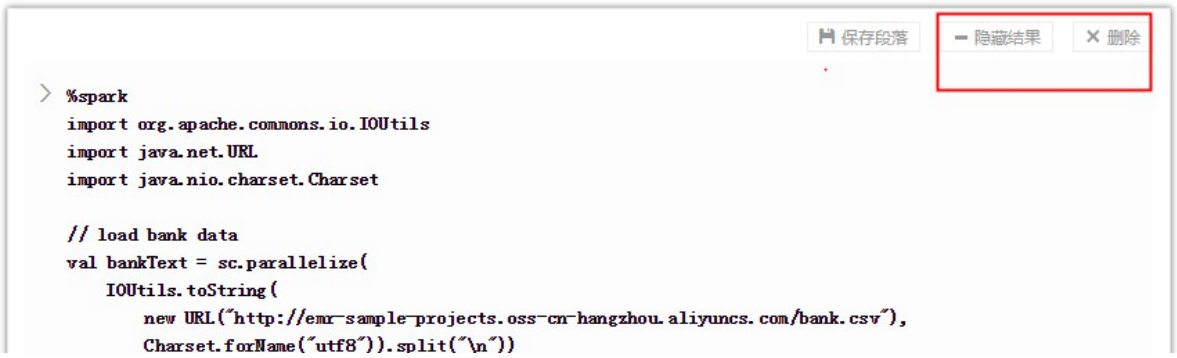


取消关联集群

集群运行过交互式任务后，为了再次执行时能够快速响应，会创建进程缓存一些上下文运行环境。如果您暂时不再执行交互式任务，想要释放缓存占用的集群资源，可以把运行过的交互式任务都取消关联，会释放掉原关联集群上占用的内存资源。

其他操作项

- 段落操作



— 隐藏结果/显示结果

可以将段落的结果隐藏掉，只显示段落的输入内容。

— 删除

删除当前段落，运行中的段落也可以删除。

- 文件菜单

— 新建交互式任务

新建一个交互式任务，并切换界面到新建的交互式任务上。

— 新建段落

在交互式任务的尾部添加一个新段落，一个交互式任务最多有30个段落。

— 保存所有段落

所有修改过的段落都会保存

— 删除交互式任务

删除掉当前的交互式任务。如果关联了集群会同时取消关联。

- 视图

只显示代码/显示代码和结果

所有段落只显示输入的代码，还是同时显示结果内容。

16.3 交互式工作台示例

16.3.1 银行员工信息查询示例

段落1创建临时表

```
%spark
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
// Zeppelin creates and injects sc (SparkContext) and sqlContext (
HiveContext or SqlContext)
// So you don't need create them manually
// load bank data
val bankText = sc.parallelize(
    IOUtils.toString(
        new URL("http://emr-sample-projects.oss-cn-hangzhou.aliyuncs.
com/bank.csv"),
        Charset.forName("utf8")).split("\n"))
case class Bank(age: Integer, job: String, marital: String, education
: String, balance: Integer)
val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "\"age
\"").map(
    s => Bank(s(0).toInt,
            s(1).replaceAll("\"", ""),
            s(2).replaceAll("\"", ""),
            s(3).replaceAll("\"", ""),
            s(5).replaceAll("\"", "").toInt
        )
).toDF()
bank.registerTempTable("bank")
```

段落2查询表结构

```
%sql
desc bank
```

段落3查询年龄小于30各年龄段员工人数

```
%sql select age, count(1) value from bank where age < 30 group by age
order by age
```

段落4 查询年龄小于等于20岁的员工信息

```
%sql select * from bank where age <= 20
```

16.3.2 视频播放数据示例

数据准备

本示例需要您从oss上下载数据，并上传到您自己的oss bucket上。数据包含

- 用户表示例数据

- 视频表示例数据
- 播放表示例数据

分别上传到您oss bucket指定目录的userinfo子目录，videoinfo目录，playvideo目录。例如 bucket example 下的demo/userinfo目录。

将下面创建表的sql中[bucketname]替换成您的bucket名字例如example，[region]替换成您用的oss地域名如hangzhou,[bucketpath]替换成您oss的指定的路径前缀例如demo。

段落1创建用户表

```
%hive
CREATE EXTERNAL TABLE user_info(id int,sex int,age int, marital_st
atus int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 'oss
://[bucketname].oss-cn-[region]-internal.aliyuncs.com/[bucketpath]/
userinfo'
```

段落2创建视频表

```
%hive
CREATE EXTERNAL TABLE video_info(id int,title string,type string) ROW
 FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 'oss://[bucketname
].oss-cn-[region]-internal.aliyuncs.com/[bucketpath]/videoinfo'
```

段落3创建播放表

```
%hive
CREATE EXTERNAL TABLE play_video(user_id int,video_id int, play_time
 bigint) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 'oss
```

```
://[bucketname].oss-cn-[region]-internal.aliyuncs.com/[bucketpath]/
playvideo'
```

段落4用户表计数

```
%sql select count(*) from user_info
```

段落5 视频表计数

```
%sql select count(*) from video_info
```

段落6 播放表计数

```
%sql select count(*) from play_video
```

段落7统计各类型视频播放数

```
%sql select video.type, count(video.type) as count from play_video
 play join video_info video on (play.video_id = video.id) group by
video.type order by count desc
```

段落8播放数top10的视频信息

```
%sql select video.id, video.title, video.type, video_count.count from
 (select video_id, count(video_id) as count from play_video group by
 video_id order by count desc limit 10) video_count join video_info
video on (video_count.video_id = video.id) order by count desc
```

段落9播放数最高视频观看者的年龄分布

```
%sql select age , count(*) as count  from (select distinct(user_id)
from play_video  where video_id =49 ) play join user_info userinfo on
 (play.user_id = userinfo.id) group by userinfo.age
```

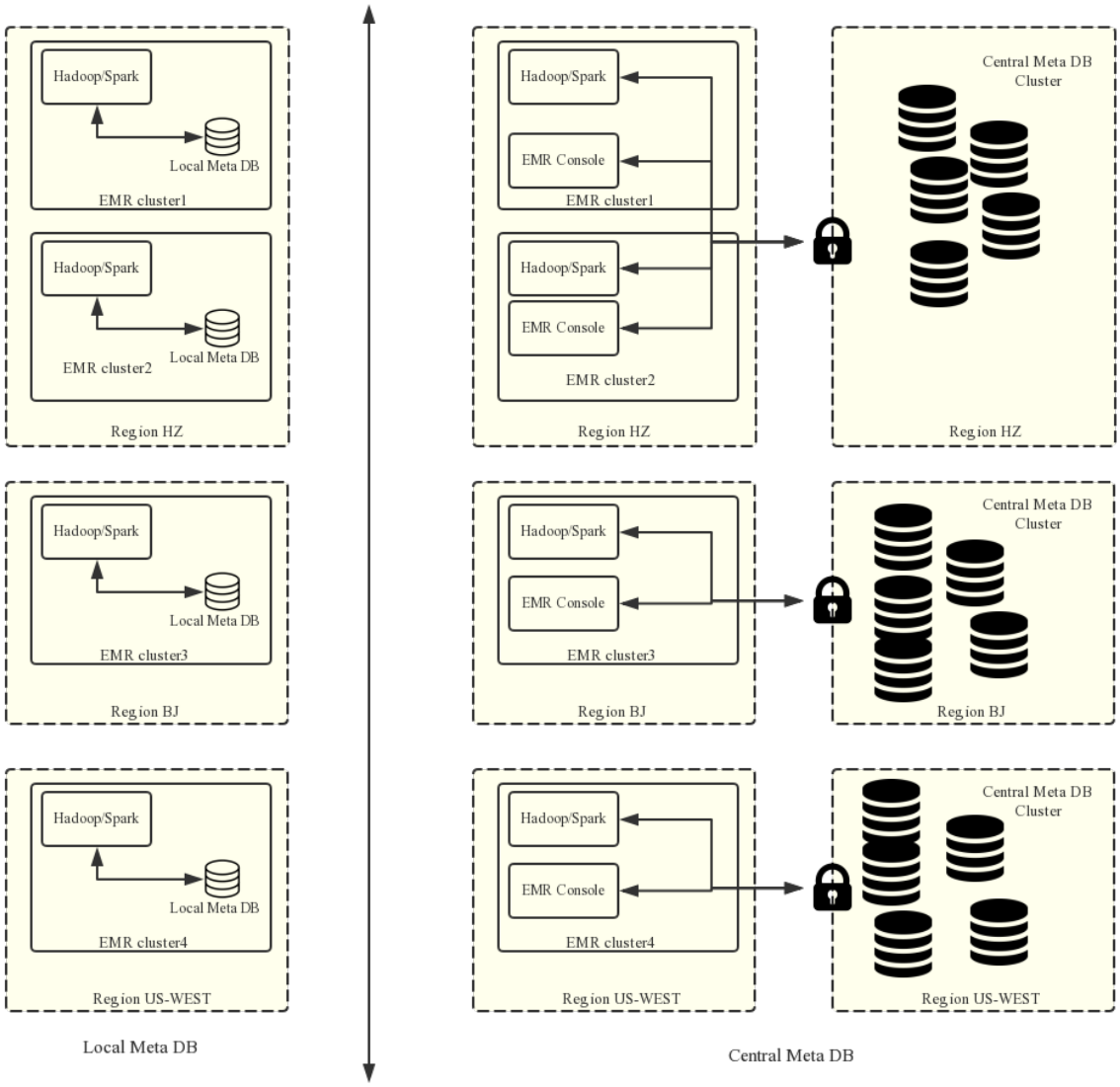
段落10播放数最高视频观看者的性别，年龄段，婚姻状态分布汇总

```
%sql select if(sex=0,'女','男') as title, count(*) as count, '性别' as
 type from (select distinct(user_id) from play_video  where video_id
 =49 ) play join user_info userinfo on (play.user_id = userinfo.id)
group by userinfo.sex
union all
select case when userinfo.age<15 then '小于15' when age<25 then '15-25
' when age<35 then '25-35' else '大于35' end  , count(*) as count, '年
龄段' as type from (select distinct(user_id) from play_video  where
video_id =49) play join user_info userinfo on (play.user_id = userinfo
.id) group by case when userinfo.age<15 then '小于15' when age<25 then
 '15-25' when age<35 then '25-35' else '大于35' end
union all
select if(marital_status=0,'未婚','已婚') as title, count(*) as count
, '婚否' as type from (select distinct(user_id) from play_video
where video_id =49 ) play join user_info userinfo on (play.user_id =
userinfo.id) group by marital_status
```

17 表管理

从EMR-2.4.0版本开始，EMR支持了统一元数据管理，在EMR-2.4.0版本之前，用户所有集群均采用的是集群本地的mysql数据库作为hive元数据库，在EMR-2.4.0版本以及之后的版本中，EMR会支持统一的高可靠的hive元数据库。

介绍



用户可以选择在创建集群的时候，打开我们的元数据库的开关，从而使用外部的元数据库。

说明：

- 当前元数据库需要使用公网IP来连接，所以集群必须要有公网IP，同时请不要随意的切换公网IP地址，防止对应的数据库白名单失效。
- 只有在创建集群的时候打开了统一元数据库这个开关才能使用表管理的功能，如果是本地的元数据库，目前还不支持进行管理。可以使用集群上的 Hue 工具来进行管理。

有了统一的元数据管理之后，就可以实现：

1. 提供持久化的元数据存储

   之前元数据都是在集群内部的Mysql数据库，元数据会随着集群的释放而丢失，特别是EMR提供了灵活按量模式，集群可以按需创建用完就释放。如果用户需要保留现有的元数据信息，必须登录上集群手动将元数据信息导出。支持统一的元数据管理之后，不再存在该问题。

2. 能更方便地实现计算存储分离

   EMR上可以支持将数据存放在阿里云OSS中，在大数据量的情况下将数据存储在OSS上会大大降低使用的成本，EMR集群主要用来作为计算资源，在计算完成之后机器可以随时释放，数据在OSS上，同时也不用再考虑元数据迁移的问题。

3. 更方便地实现数据共享

   使用统一的元数据库，如果用户的所有数据都存放在OSS之上，则不需要做任何元数据的迁移和重建所有集群都是可以直接访问数据，这样每个EMR集群可以做不同的业务，但是可以很方便地实现数据的共享。

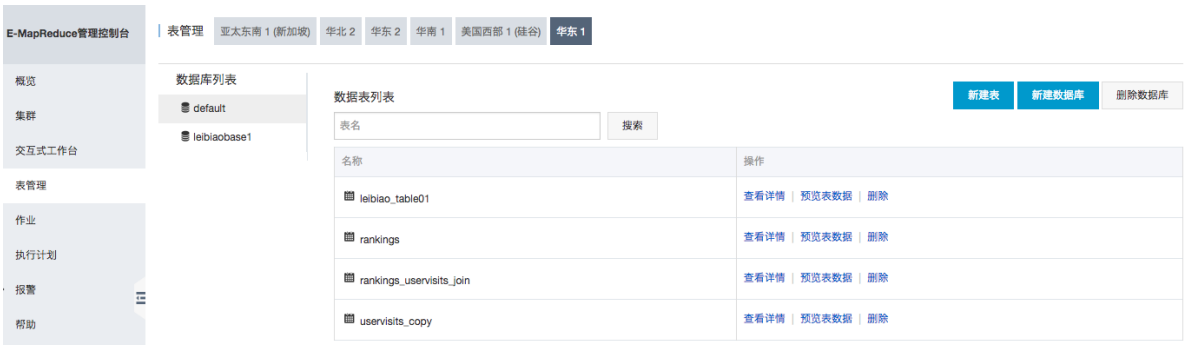
 注意：

在支持统一元数据之前，元数据是存储在每个集群本地环境的Mysql数据库中，所以元数据会随着集群的释放消亡。在支持统一元数据之后，释放集群不会清理元数据信息。所以，在任何时候删除OSS上的数据或者集群HDFS上的数据（包括释放集群操作）的时候，需要先确认该数据对应的元数据已经删除（即要drop掉数据对应的表和数据库）。否则元数据库中可能出现一些脏的元数据信息。

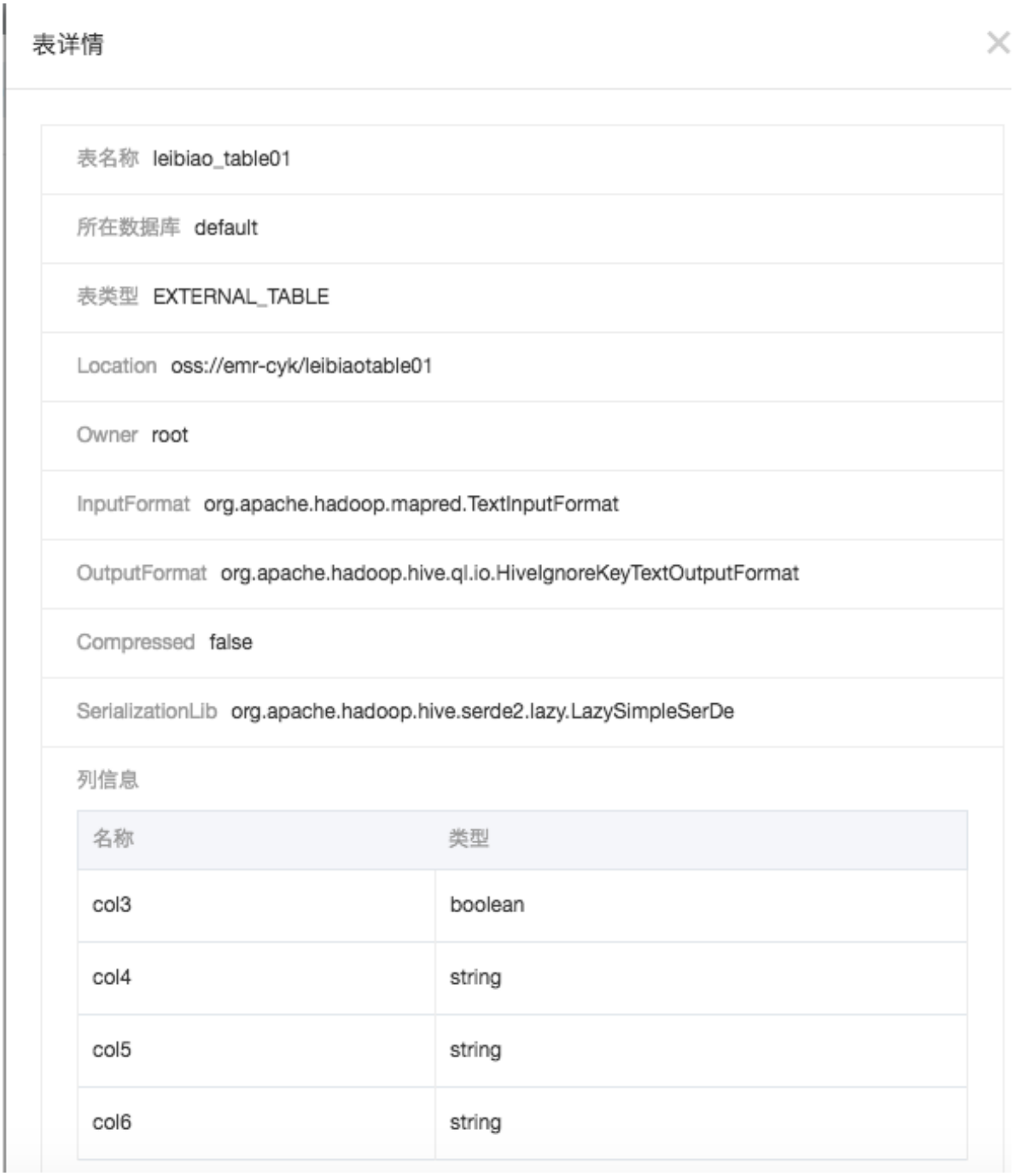
表管理操作

在EMR集群支持统一元数据支持之前，客户对集群上表的查看、增删操作必须要登录到集群内部环境操作，如果有多个集群，则需要每个集群上分别操作，非常不方便。在EMR支持统一元数据管理之后，为了用户更方便地对表进行管理，EMR在控制台上提供了表管理功能。包括对对数据库和表列表的查看、表详情的查看、新建数据库和表、删除数据库和表、数据预览等操作。

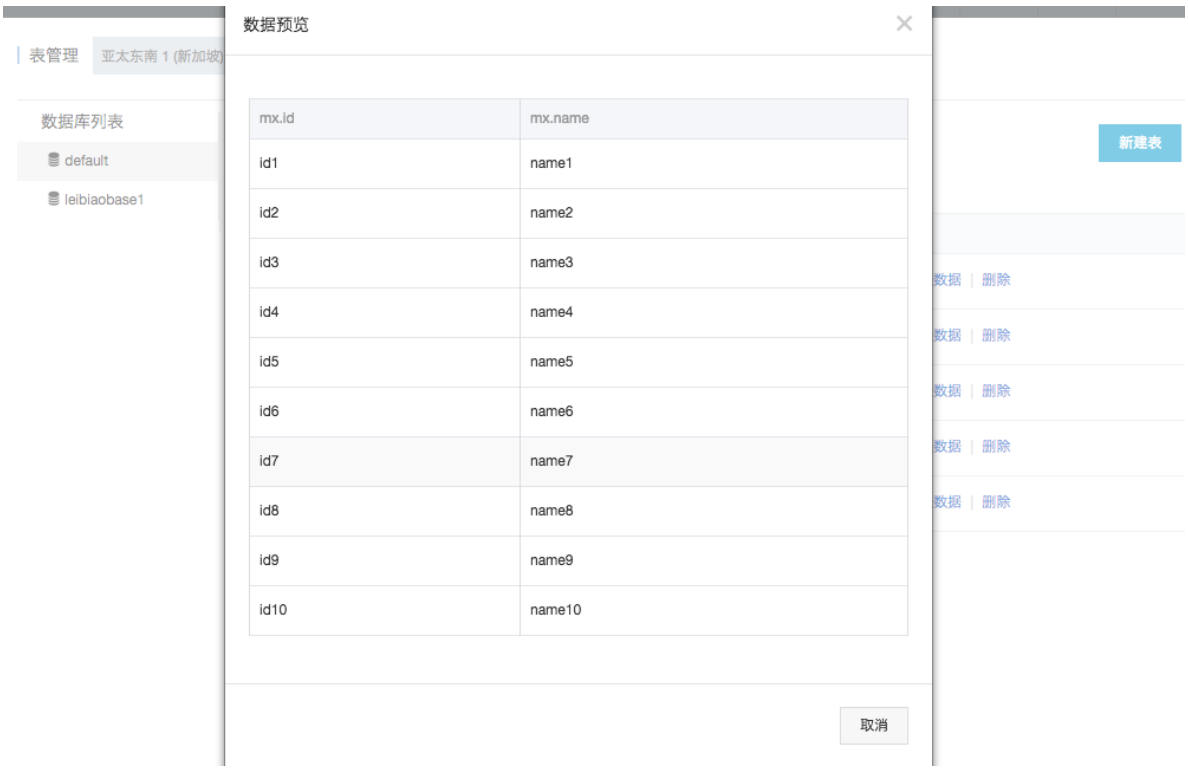
- 数据库和表列表


E-MapReduce管理控制台
表管理
亚太东南 1 (新加坡)
华北 2
华东 2
华南 1
美国西部 1 (硅谷)
华东 1
概览
集群
交互式工作台
表管理
作业
执行计划
报警
帮助
数据库列表
default
leibiaobase1
数据表列表
新建表
新建数据库
删除数据库
表名
搜索
名称
操作
leibiao_table01
查看详情 | 预览表数据 | 删除
rankings
查看详情 | 预览表数据 | 删除
rankings_uservisits_join
查看详情 | 预览表数据 | 删除
uservisits_copy
查看详情 | 预览表数据 | 删除

- 表详情


表详情
表名称 leibiao_table01
所在数据库 default
表类型 EXTERNAL_TABLE
Location oss://emr-cyk/leibiaotable01
Owner root
InputFormat org.apache.hadoop.mapred.TextInputFormat
OutputFormat org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed false
SerializationLib org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
列信息
名称
类型
col3
boolean
col4
string
col5
string
col6
string

- 数据预览



- 创建数据库

- 创建表

创建表的时候，有两种选项：手动创建表和从文件创建表

- 手动创建表，表示还没有对应的业务数据，手动输入表结构建一个空的表；
- 从文件创建表，是指在已经有业务数据的情况下，可以直接将这些业务数据作为一个表，表接口是从文件内容中解析的。注意创建表时候设置的分隔符需要跟数据文件中的分隔符对应以保证表结构正确。

分隔符不仅支持普通的逗号、空格等字符，还支持了四种特殊字符：TAB、^A、^B和^C。

注意：

1. 如果没有任何EMR集群，不支持进行对数据库和表的创建和删除操作。
2. 由于HDFS是每个集群内部文件系统，在没有进行特殊的网络环境设置的情况下，不同集群之间的HDFS不无法相互访问的，所以EMR表管理功能对数据库和表的创建只支持基于OSS文件系统的。
3. 数据库和表的location都不能选择整个OSS bucket，需要选择到OSS bucket下面的目录。

常见问题

1. Wrong FS: *oss://yourbucket/xxx/xxx/xxx*

   出现这个问题，是由于删除OSS上的表数据之前，没有删除数据表对应的元数据。导致表的schema还在，但实际的数据已不存在或已移动到别的路径。可以先修改表的location为一个存在的路径，然后再删除表。

   ```
   alter table test set location 'oss://your_bucket/your_folder'
   ```

   可以直接在EMR控制台的交互式控制台中完成：

   说明：

   *oss://your_bucket/your_folder*必须是一个存在的oss路径。

2. Wrong FS: hdfs://yourhost:9000/xxx/xxx/xxx

   出现这个问题是删除了表在HDFS上的数据，但是没有删除表对应的schema信息，解决方法同上。

3. 删除hive database的时候出现：java.lang.IllegalArgumentException: java.net.UnknownHostException: xxxxxxx

   出现该问题的原因，是因为在之前的集群之上创建了hive的数据库，并且数据库的位置是落在之前集群的hdfs之上，但是在集群释放的时候，没有清理掉对应的hive database，导致新建集群之后，没法访问到之前已经释放集群的hdfs数据。所以如果是手动创建了在hdfs之上的数据库和表，在释放集群的时候请记得清理。

   解决方案：

   首先，通过命令行登录到集群master节点上，找到hive meta db的访问地址和用户名密码信息，在*$HIVE_CONF_DIR/hive-site.xml*中，找到对应属性。

   ```
   javax.jdo.option.ConnectionUserName //对应数据库用户名;
   javax.jdo.option.ConnectionPassword //对应数据库访问密码;
   ```

```
javax.jdo.option.ConnectionURL //对应数据库访问地址和库名;
```

```
<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>${ConnectionUserName}</value>
  <description>Username to use against metastore database</description>
</property>
<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>${ConnectionPassword}</value>
  <description>password to use against metastore database</description>
</property>
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://${DBConnectionURL}/${DBName}?createDatabaseIfNotExist=true&amp;amp;characterEncoding=UTF-8</value>
  <description>JDBC connect string for a JDBC metastore</description>
</property>
```

在集群master节点上登录hive meta db:

```
mysql -h ${DBConnectionURL} -u ${ConnectionUserName} -p [回车]
[输入密码]${ConnectionPassword}
```

登录上hive meta db之后，修改对应hive database的location为该region真实存在的oss路径即可：

```
mysql> show databases;
+-------------------------------------------------+
| Database                                        |
+-------------------------------------------------+
| information_schema                              |
| xxxxxxxxx77ac43c3bd0efae77e0bf1947d45fb4c896fb99 |
+-------------------------------------------------+

mysql> use xxxxxxxxx77ac43c3bd0efae77e0bf1947d45fb4c896fb99;

mysql> select * from dbs;
+-------+------------------------+-----------------------------------+----------+------------+------------+
| DB_ID | DESC                   | DB_LOCATION_URI                   | NAME     | OWNER_NAME | OWNER_TYPE |
+-------+------------------------+-----------------------------------+----------+------------+------------+
|     1 | Default Hive database | oss://mybucket/hive/warehouse     | default  | public     | ROLE       |
|     6 | NULL                   | hdfs://dirty-hostname/warehouse   | dirty_db | NULL       | USER       |
+-------+------------------------+-----------------------------------+----------+------------+------------+

mysql> update dbs set DB_LOCATION_URI = 'oss://your-bucket/your-db-folder' where DB_ID = 6;
```

18 Kerberos认证

18.1 Kerberos简介

E-MapReduce从EMR-2.7.x/EMR-3.5.x版本开始支持创建安全类型的集群，即集群中的开源组件以Kerberos的安全模式启动,在这种安全环境下只有经过认证的客户端(Client)才能访问集群的服务(Service，如HDFS)。

前置

目前E-MapReduce版本中支持的Kerberos的组件列表如下所示：

组件名称	组件版本
YARN	2.7.2
SPARK	2.1.1/1.6.3
HIVE	2.0.1
TEZ	0.8.4
ZOOKEEPER	3.4.6
HUE	3.12.0
ZEPPELIN	0.7.1
OOZIE	4.2.0
SQOOP	1.4.6
HBASE	1.1.1
PHOENIX	4.7.0

说明：

Kafka/Presto/Storm目前版本不支持Kerberos。

创建安全集群

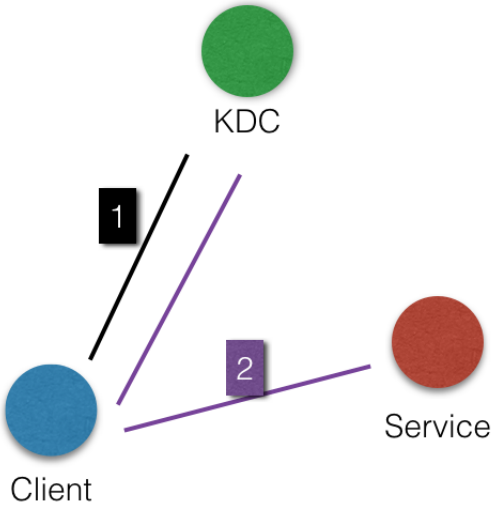
在集群创建页面的软件配置下打开安全按钮即可，如下所示:


创建集群
软件配置
硬件配置
版本配置
产品版本:
EMR-3.14.0
集群类型:
Hadoop
Druid
Data Science
必选服务:
Knox (0.13.0)
ApacheDS (2.0.0)
Tez (0.9.1)
Sqoop (1.4.7)
Pig (0.14.0)
YARN (2.7.2)
HDFS (2.7.2)
Ganglia (3.7.2)
可选服务:
Superset (0.27.0)
Ranger (1.0.0)
Flink (1.4.0)
Phoenix (4.10.0)
HBase (1.1.1)
Presto (0.208)
Impala (2.10.0)
请点击选择
高安全模式:
软件自定义配置:

Kerberos身份认证原理

Kerberos是一种基于对称密钥技术的身份认证协议，它作为一个独立的第三方的身份认证服务，可以为其它服务提供身份认证功能，且支持SSO(即客户端身份认证后，可以访问多个服务如HBase/HDFS等)。

Kerberos协议过程主要有两个阶段，第一个阶段是KDC对Client身份认证，第二个阶段是Service对Client身份认证。



- KDC

  Kerberos的服务端程序

- Client

  需要访问服务的用户(principal)，KDC和Service会对用户的身份进行认证

- Service

  集成了Kerberos的服务，如HDFS/YARN/HBase等

- KDC对Client身份认证

  当客户端用户(principal)访问一个集成了Kerberos的服务之前，需要先通过KDC的身份认证。

  若身份认证通过则客户端会拿到一个TGT(Ticket Granting Ticket)，后续就可以拿该TGT去访问集成了Kerberos的服务。

- Service对Client身份认证

当2.1中用户拿到TGT后，就可以继续访问Service服务。它会使用TGT以及需要访问的服务名称(如HDFS)去KDC获取SGT(Service Granting Ticket)，然后使用SGT去访问Service，Service会利用相关信息对Client进行身份认证，认证通过后就可以正常访问Service服务。

EMR实践

EMR的Kerberos安全集群中的服务在创建集群的时候会以Kerberos安全模式启动。

- Kerberos服务端程序为HasServer
  - 登录阿里云 *E-MapReduce* 控制台，选择集群管理 > 管理 > **Has**，执行查看/修改配置/重启等操作。
  - 非HA集群部署在emr-header-1,HA集群部署在emr-header-1/emr-header-2两个节点
- 支持四种身份认证方式

  HasServer可同时支持以下4种身份认证方式，客户端可以通过配置相关参数来指定HasServer使用哪种方式进行身份认证

  - 兼容MIT Kerberos的身份认证方式

    客户端配置:

    ```
    如果在集群的某个节点上执行客户端命令，则需要将/etc/ecm/hadoop-conf/core-site.xml中hadoop.security.authentication.use.has设置为false.
    如果有通过控制台的执行计划跑作业，则不能修改master节点上面/etc/ecm/hadoop-conf/core-site.xml中的值，否则执行计划的作业认证就不通过而失败，可以使用下面的方式
    export HADOOP_CONF_DIR=/etc/has/hadoop-conf临时export环境变量，该路径下的hadoop.security.authentication.use.has已经设置为false
    ```

    访问方式:Service的客户端包完全可使用开源的，如HDFS客户端等。详见
  - RAM身份认证

    客户端配置:

    ```
    如果在集群的某个节点上执行客户端命令，则需要将/etc/ecm/hadoop-conf/core-site.xml中hadoop.security.authentication.use.has设置为true，/etc/has/has-client.conf中auth_type设置为RAM.
    如果有通过控制台的执行计划跑作业，则不能修改master节点上面/etc/ecm/hadoop-conf/core-site.xml以及/etc/has/has-client.conf中的值，否则执行计划的作业认证就不通过而失败，可以使用下面的方式
    ```

```
export HADOOP_CONF_DIR=/etc/has/hadoop-conf; export HAS_CONF_DIR=/
path/to/has-client.conf临时export环境变量，其中HAS_CONF_DIR文件夹下的
has-client.conf的auth_type设置为RAM
```

访问方式： 客户端需要使用集群中的软件包(如Hadoop/HBase等)，详见

- LDAP身份认证

客户端配置：

```
如果在集群的某个节点上执行客户端命令，则需要将
/etc/ecm/hadoop-conf/core-site.xml中hadoop.security.authentication
.use.has设置为true，/etc/has/has-client.conf中auth_type设置为LDAP.
如果有通过控制台的执行计划跑作业，则不能修改master节点上面/etc/ecm/hadoop-
conf/core-site.xml以及/etc/has/has-client.conf中的值，否则执行计划的作
业认证就不通过而失败，可以使用下面的方式
export HADOOP_CONF_DIR=/etc/has/hadoop-conf; export HAS_CONF_DIR=/
path/to/has-client.conf临时export环境变量，其中HAS_CONF_DIR文件夹下的
has-client.conf的auth_type设置为LDAP
```

访问方式：客户端需要使用集群中的软件包(如Hadoop/HBase等)，详见。

- 执行计划认证

如果用户有使用EMR控制台的执行计划提交作业，则emr-header-1节点的配置必须不能被修改(默认配置)。

客户端配置:

```
emr-header-1上面的/etc/ecm/hadoop-conf/core-site.xml中hadoop.
security.authentication.use.has设置为true，/etc/has/has-client.conf
中auth_type设置为EMR.
```

访问方式：跟非Kerberos安全集群使用方式一致，详见

。

- 其他

登陆master节点访问集群

集群管理员也可以登陆master节点访问集群服务，登陆master节点切换到has账号(默认使用兼容MIT Kerberos的方式)即可访问集群服务，方便做一些排查问题或者运维等。

```
>sudo su has
>hadoop fs -ls /
```

说明：

> 也可以登录其他账号操作集群，前提是该账号可以通过Kerberos认证。另外，如果在master节点上需要使用 兼容MITKerberos的方式，需要在该账号下先export一个环境变量。

```
export HADOOP_CONF_DIR=/etc/has/hadoop-conf/
```

18.2 兼容MIT Kerberos认证

本文将通过HDFS服务介绍兼容MIT Kerberos认证流程。

兼容MIT Kerberos的身份认证方式

EMR集群中Kerberos服务端启动在master节点，涉及一些管理操作需在master节点(emr-header-1)的root账号执行。

下面以test用户访问HDFS服务为例介绍相关流程。

- Gateway上执行`hadoop fs -ls /`
  - 配置krb5.conf

```
Gateway上面使用root账号
 scp root@emr-header-1:/etc/krb5.conf /etc/
```

  - 添加principal
    - 登录集群emr-header-1节点，切到root账号。
    - 进入Kerberos的admin工具。
    -

```
  sh /usr/lib/has-current/bin/hadmin-local.sh /etc/ecm/has-conf
 -k /etc/ecm/has-conf/admin.keytab
  HadminLocalTool.local: #直接按回车可以看到一些命令的用法
  HadminLocalTool.local: addprinc #输入命令按回车可以看到具体命令的用法
  HadminLocalTool.local: addprinc -pw 123456 test #添加test的princippal,密码设置为123456
```

  - 导出keytab文件

    使用Kerberos的admin工具可以导出principal对应的keytab文件。

```
HadminLocalTool.local: ktadd -k /root/test.keytab test #导出keytab文件,后续可使用该文件
```

  - kinit获取Ticket

    在执行hdfs命令的客户端机器上面，如Gateway。

- 添加linux账号test

```
useradd test
```

- 安装MITKerberos 客户端工具。

  可以使用MITKerberos tools进行相关操作(如kinit/klist等)，详细使用方式参考MITKerberos文档

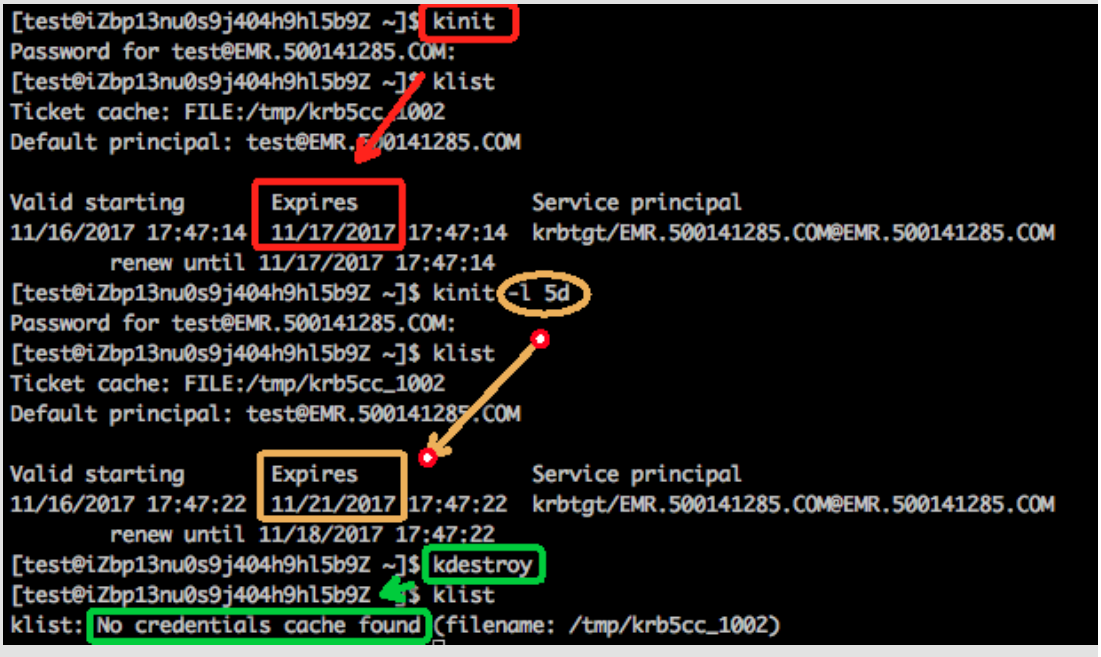
```
yum install krb5-libs krb5-workstation -y
```

- 切到test账号执行kinit

```
su test
#如果没有keytab文件,则执行
kinit #直接回车
Password for test: 123456 #即可
#如有有keytab文件,也可执行
kinit -kt test.keytab test
#查看ticket
klist
```

说明：

MITKerberos工具使用实例



- 执行hdfs命令

  获取到Ticket后，就可以正常执行hdfs命令了。

```
hadoop fs -ls /
      Found 5 items
```

```
     drwxr-xr-x     - hadoop hadoop          0 2017-11-12 14:23 /
apps
     drwx------      - hbase  hadoop           0 2017-11-15 19:40
 /hbase
     drwxrwx--t+   - hadoop hadoop         0 2017-11-15 17:51 /
spark-history
     drwxrwxrwt   - hadoop hadoop          0 2017-11-13 23:25 /tmp
     drwxr-x--t      - hadoop hadoop          0 2017-11-13 16:12 /
user
```

说明：

跑yarn作业，需要提前在集群中所有节点添加对应的linux账号(详见下文中[EMR集群添加test账号]。

- java代码访问HDFS
  - 使用本地ticket cache

说明：

需要提前执行kinit获取ticket,且ticket过期后程序会访问异常。

```
public static void main(String[] args) throws IOException {
   Configuration conf = new Configuration();
   //加载hdfs的配置,配置从emr集群上复制一份
   conf.addResource(new Path("/etc/ecm/hadoop-conf/hdfs-site.xml
"));
   conf.addResource(new Path("/etc/ecm/hadoop-conf/core-site.xml
"));
   //需要在程序所在linux账号下,提前kinit获取ticket
   UserGroupInformation.setConfiguration(conf);
   UserGroupInformation.loginUserFromSubject(null);
   FileSystem fs = FileSystem.get(conf);
   FileStatus[] fsStatus = fs.listStatus(new Path("/"));
   for(int i = 0; i < fsStatus.length; i++){
       System.out.println(fsStatus[i].getPath().toString());
   }
}
```

  - 使用keytab文件(推荐)

说明：

keytab长期有效，跟本地ticket无关。

```
public static void main(String[] args) throws IOException {
  String keytab = args[0];
  String principal = args[1];
  Configuration conf = new Configuration();
  //加载hdfs的配置,配置从emr集群上复制一份
  conf.addResource(new Path("/etc/ecm/hadoop-conf/hdfs-site.xml
"));
```

```
  conf.addResource(new Path("/etc/ecm/hadoop-conf/core-site.xml
"));
  //直接使用keytab文件,该文件从emr集群master-1上面执行相关命令获取[文档前
面有介绍命令]
  UserGroupInformation.setConfiguration(conf);
  UserGroupInformation.loginUserFromKeytab(principal, keytab);
  FileSystem fs = FileSystem.get(conf);
  FileStatus[] fsStatus = fs.listStatus(new Path("/"));
  for(int i = 0; i < fsStatus.length; i++){
      System.out.println(fsStatus[i].getPath().toString());
  }
  }
```

附pom依赖:

```
<dependencies>
            <dependency>
                <groupId>org.apache.hadoop</groupId>
                <artifactId>hadoop-common</artifactId>
                <version>2.7.2</version>
            </dependency>
            <dependency>
                <groupId>org.apache.hadoop</groupId>
                <artifactId>hadoop-hdfs</artifactId>
                <version>2.7.2</version>
            </dependency>
        </dependencies>
```

18.3 RAM认证

EMR集群中的Kerberos服务端除了可以支持第一种MIT Kerberos兼容的使用方式，也可以支持Kerberos客户端使用RAM作为身份信息进行身份认证。

RAM身份认证

*RAM*产品可以创建/管理子账号，通过子账号实现对云上各个资源的访问控制。

主账号的管理员可以在RAM的用户管理界面创建一个子账号(子账户名称必须符合linux用户的规范)，然后将子账号的AccessKey下载下来提供给该子账号对应的开发人员，后续开发人员可以通过配置AccessKey，从而通过Kerberos认证访问集群服务。

使用RAM身份认证不需要像第一部分MIT Kerberos使用方式一样，提前在Kerberos服务端添加principle等操作

下面以已经创建的子账号test在Gateway访问为例:

- EMR集群添加test账号

EMR的安全集群的yarn使用了LinuxContainerExecutor，在集群上跑yarn作业必须要在集群所有节点上面添加跑作业的用户账号，LinuxContainerExecutor执行程序过程中会根据用户账号进行相关的权限校验。

EMR集群管理员在EMR集群的master节点上执行:

```
sudo su hadoop
sh adduser.sh test 1 2
```

附:adduser.sh代码

```
#添加的账户名称
 user_name=$1
 #集群master节点个数,如HA集群有2个master
 master_cnt=$2
 #集群worker节点个数
 worker_cnt=$3
 for((i=1;i<=$master_cnt;i++))
 do
   ssh -o StrictHostKeyChecking=no emr-header-$i sudo useradd $
user_name
 done
 for((i=1;i<=$worker_cnt;i++))
 do
   ssh -o StrictHostKeyChecking=no emr-worker-$i sudo useradd $
user_name
done
```

- Gateway管理员在Gateway机器上添加test用户

```
useradd test
```

- Gateway管理员配置Kerberos基础环境

```
sudo su root
 sh config_gateway_kerberos.sh 10.27.230.10 /pathto/emrheader1
_pwd_file
 #确保Gateway上面/etc/ecm/hadoop-conf/core-site.xml中值为true
 <property>
    <name>hadoop.security.authentication.use.has</name>
    <value>true</value>
 </property>
```

附: config_gateway_kerberos.sh脚本代码

```
#EMR集群的emr-header-1的ip
 masterip=$1
 #保存了masterip对应的root登录密码文件
 masterpwdfile=$2
 if ! type sshpass >/dev/null 2>&1; then
    yum install -y sshpass
fi
  ## Kerberos conf
 sshpass -f $masterpwdfile scp root@$masterip:/etc/krb5.conf /etc/
```

```
 mkdir /etc/has
 sshpass -f $masterpwdfile scp root@$masterip:/etc/has/has-client.
conf /etc/has
 sshpass -f $masterpwdfile scp root@$masterip:/etc/has/truststore /
etc/has/
 sshpass -f $masterpwdfile scp root@$masterip:/etc/has/ssl-client.
conf /etc/has/
 #修改Kerberos客户端配置,将默认的auth_type从EMR改为RAM
 #也可以手工修改该文件
 sed -i 's/EMR/RAM/g' /etc/has/has-client.conf
```

- test用户登录Gateway配置AccessKey

```
登录Gateway的test账号
 #执行脚本
 sh add_accesskey.sh test
```

附: add_accesskey.sh脚本(修改一下AccessKey)

```
user=$1
 if [[ `cat /home/$user/.bashrc | grep 'export AccessKey'` == "" ]];
then
 echo "
 #修改为test用户的AccessKeyId/AccessKeySecret
 export AccessKeyId=YOUR_AccessKeyId
 export AccessKeySecret=YOUR_AccessKeySecret
 " >>~/.bashrc
 else
    echo $user AccessKey has been added to .bashrc
 fi
```

- test用户执行命令

经过以上步骤，test用户可以执行相关命令访问集群服务了。

执行hdfs命令

```
[test@gateway ~]$ hadoop fs -ls /
  17/11/19 12:32:15 INFO client.HasClient: The plugin type is: RAM
  Found 4 items
  drwxr-x---   - has    hadoop          0 2017-11-18 21:12 /apps
  drwxrwxrwt   - hadoop hadoop          0 2017-11-19 12:32 /spark-
history
  drwxrwxrwt   - hadoop hadoop          0 2017-11-18 21:16 /tmp
  drwxrwxrwt   - hadoop hadoop          0 2017-11-18 21:16 /user
```

跑hadoop作业

```
[test@gateway ~]$ hadoop jar /usr/lib/hadoop-current/share/hadoop/
mapreduce/hadoop-mapreduce-examples-2.7.2.jar pi 10 1
```

跑spark作业

```
[test@gateway ~]$ spark-submit --conf spark.ui.view.acls=* --class
 org.apache.spark.examples.SparkPi --master yarn-client --driver-
memory 512m --num-executors 1 --executor-memory 1g --executor-cores
```

```
2 /usr/lib/spark-current/examples/jars/spark-examples_2.11-2.1.1.jar
 10
```

18.4 LDAP认证

EMR集群还支持基于LDAP的身份认证，通过LDAP来管理账号体系，Kerberos客户端使用LDAP中的账号信息作为身份信息进行身份认证。

LDAP身份认证

LDAP账号可以是和其它服务共用，比如Hue等，只需要在Kerberos服务端进行相关配置即可。用户可以使用EMR集群中已经配置好的LDAP服务(ApacheDS)，也可以使用已经存在的LDAP服务，只需要在Kerberos服务端进行相关配置即可。

下面以集群中已经默认启动的LDAP服务(ApacheDS)为例:

- Gateway管理对基础环境进行配置(跟第二部分RAM中的一致，如果已经配置可以跳过)

  区别的地方只是*/etc/has/has-client.conf*中的*auth_type*需要改为LDAP

  也可以不修改*/etc/has/has-client.conf*,用户test在自己的账号下拷贝一份该文件进行修改*auth_type*，然后通过环境变量指定路径，如:

  ```
  export HAS_CONF_DIR=/home/test/has-conf
  ```

- EMR控制台配置LDAP管理员用户名/密码到Kerberos服务端(HAS)

  进入EMR控制台集群的配置管理-HAS软件下，将LDAP的管理员用户名和密码配置到对应的*bind_dn*和*bind_password*字段，然后重启HAS服务。

  此例中，LDAP服务即EMR集群中的ApacheDS服务，相关字段可以从ApacheDS中获取。

- EMR集群管理员在LDAP中添加用户信息

  - 获取ApacheDS的LDAP服务的管理员用户名和密码在EMR控制台集群的配置管理/ApacheDS的配置中可以查看*manager_dn*和*manager_password*
  - 在ApacheDS中添加test用户名和密码

    ```
    登录集群emr-header-1节点root账号
     新建test.ldif文件,内容如下：
     dn: cn=test,ou=people,o=emr
     objectclass: inetOrgPerson
     objectclass: organizationalPerson
     objectclass: person
     objectclass: top
     cn: test
     sn: test
     mail: test@example.com
     userpassword: test1234
    ```

```
 #添加到LDAP，其中-w 对应修改成密码manager_password
 ldapmodify -x -h localhost -p 10389 -D "uid=admin,ou=system" -w "
Ns1aSe" -a -f test.ldif
 #删除test.ldif
 rm test.ldif
```

将添加的用户名/密码提供给test使用

- 用户test配置LDAP信息

```
登录Gateway的test账号
 #执行脚本
 sh add_ldap.sh test
```

附: add_ldap.sh脚本(修改一下LDAP账号信息)

```
user=$1
 if [[ `cat /home/$user/.bashrc | grep 'export LDAP_'` == "" ]];then
 echo "
 #修改为test用户的LDAP_USER/LDAP_PWD
 export LDAP_USER=YOUR_LDAP_USER
 export LDAP_PWD=YOUR_LDAP_USER
 " >>~/.bashrc
 else
    echo $user LDAP user info has been added to .bashrc
 fi
```

- 用户test访问集群服务

执行hdfs命令

```
[test@iZbp1cyio18s5ymggr7yhrZ ~]$ hadoop fs -ls /
  17/11/19 13:33:33 INFO client.HasClient: The plugin type is: LDAP
  Found 4 items
  drwxr-x---   - has    hadoop          0 2017-11-18 21:12 /apps
  drwxrwxrwt   - hadoop hadoop          0 2017-11-19 13:33 /spark-
history
  drwxrwxrwt   - hadoop hadoop          0 2017-11-19 12:41 /tmp
  drwxrwxrwt   - hadoop hadoop          0 2017-11-19 12:41 /user
```

跑Hadoop/Spark作业等

18.5 执行计划认证

EMR集群支持执行计划认证，用户可以通过主账号授权子账号进行执行计划的访问。

主账号访问

在主账号登陆E-MapReduce控制台的情况下，在执行计划页面运行相关执行计划，将作业提交到安全集群上面执行，以hadoop用户访问作业中涉及的相关开源组件服务。

子账号访问

在RAM子账号登陆E-MapReduce控制台的情况下，在执行计划页面运行相关执行计划，将作业提交到安全集群上面执行，以RAM子账号对应的用户名访问作业中涉及的相关开源组件服务。

示例

- 主账号管理员根据需求创建多个子账号(如A/B/C),在RAM控制台页面给子账号授予*AliyunEMRFullAccess*的权限后，子账号就能正常登陆并使用E-MapReduce控制台上的相关功能。
- 主账号管理员将子账号提供给相关开发人员
- 开发人员完成作业的创建/执行计划的创建,然后启动运行执行计划提交作业到集群运行，最终在集群上面以该子账号对应的用户名(A/B/C)去访问相关的组件服务。

  > 说明：
  > 周期调度的执行计划目前统一以hadoop账号执行

- 组件服务使用子账户的用户名进行相关的权限控制，如子账户A是否有权限访问hdfs中的某个文件等。

18.6 跨域互信

E-MapReduce中的Kerberos支持跨域访问(cross-realm)，即不同的Kerberos集群之间可以互相访问。

下面以Cluster-A跨域去访问Cluster-B中的服务为例:

- Cluster-A的emr-header-1的hostname -> emr-header-1.cluster-1234 ; realm -> EMR.1234.COM
- Cluster-B的emr-header-1的hostname -> emr-header-1.cluster-6789 ; realm -> EMR.6789.COM
- > 说明：
  > - hostname可以在emr-header-1上面执行命令hostname获取
  > - realm可以在emr-header-1上面的/etc/krb5.conf获取

添加principal

Cluster-A和Cluster-B两个集群的emr-header-1节点分别执行以下完全一样的命令:

```
# root账号
      sh /usr/lib/has-current/bin/hadmin-local.sh /etc/ecm/has-conf -
k /etc/ecm/has-conf/admin.keytab
```

```
        HadminLocalTool.local: addprinc -pw 123456 krbtgt/EMR.6789.COM@
EMR.1234.COM
```

说明：

- 123456是密码,可自行修改
- EMR.6789.COM是Cluster-B的realm，即被访问的集群的realm
- EMR.1234.COM是Cluster-A的realm，即发起访问的集群realm

配置Cluster-A的/etc/krb5.conf

在Cluster-A集群上配置[realms]/[domain_realm]/[capaths]，如下所示:

```
[libdefaults]
    kdc_realm = EMR.1234.COM
    default_realm = EMR.1234.COM
    udp_preference_limit = 4096
    kdc_tcp_port = 88
    kdc_udp_port = 88
    dns_lookup_kdc = false
[realms]
    EMR.1234.COM = {
                kdc = 10.81.49.3:88
    }
    EMR.6789.COM = {
                kdc = 10.81.49.7:88
    }
[domain_realm]
    .cluster-1234 = EMR.1234.COM
    .cluster-6789 = EMR.6789.COM
[capaths]
    EMR.1234.COM = {
       EMR.6789.COM = .
    }
    EMR.6789.COM = {
       EMR.1234.COM = .
    }
```

将上述/etc/krb5.conf同步到Cluster-A所有节点

将Cluster-B节点的/etc/hosts文件中绑定信息(只需要长域名emr-xxx-x.cluster-xxx)拷贝复制到Cluster-A的所有节点/etc/hosts

```
10.81.45.89  emr-worker-1.cluster-xxx
 10.81.46.222  emr-worker-2.cluster-xx
 10.81.44.177  emr-header-1.cluster-xxx
```

说明：

- Cluster-A上面如果要跑作业访问Cluster-B，需要先重启yarn

- Cluster-A的所有节点配置Cluster-B的host绑定信息

访问Cluster-B服务

在Cluster-A上面可以用Cluster-A的Kerberos的keytab文件/ticket缓存，去访问Cluster-B的服务。

如访问Cluster-B的hdfs服务:

```
su has;
hadoop fs -ls hdfs://emr-header-1.cluster-6789:9000/
Found 4 items
-rw-r-----   2 has    hadoop         34 2017-12-05 18:15 hdfs://emr-
header-1.cluster-6789:9000/abc
drwxrwxrwt   - hadoop hadoop          0 2017-12-05 18:32 hdfs://emr-
header-1.cluster-6789:9000/spark-history
drwxrwxrwt   - hadoop hadoop          0 2017-12-05 17:53 hdfs://emr-
header-1.cluster-6789:9000/tmp
drwxrwxrwt   - hadoop hadoop          0 2017-12-05 18:24 hdfs://emr-
header-1.cluster-6789:9000/user
```

19 组件授权

19.1 HDFS授权

HDFS开启了权限控制后，用户访问HDFS需要有合法的权限才能正常操作HDFS，如读取数据/创建文件夹等。

添加配置

HDFS权限相关的配置如下:

- dfs.permissions.enabled

  开启权限检查，即使该值为false， chmod/chgrp/chown/setfacl操作还是会进行权限检查
- dfs.datanode.data.dir.perm

  datanode使用的本地文件夹路径的权限，默认755
- fs.permissions.umask-mode
  - 权限掩码，在新建文件/文件夹的时候的默认权限设置
  - 新建文件: 0666 & ^umask
  - 新建文件夹: 0777 & ^umask
  - 默认umask值为022，即新建文件权限为644(666&^022=644)，新建文件夹权限为755(777&^022=755)
  - EMR的Kerberos安全集群默认设置为027，对应新建文件权限为640，新建文件夹权限为750
- dfs.namenode.acls.enabled
  - 打开ACL控制，打开后除了可以对owner/group进行权限控制外，还可以对其它用户进行设置。
  - 设置ACL相关命令:

    ```
    hadoop fs -getfacl [-R] <path>
    hadoop fs -setfacl [-R] [-b |-k -m |-x <acl_spec> <path>] |[--set
     <acl_spec>   <path>]
    ```

    如:

    ```
    su test
     #test用户创建文件夹
     hadoop fs -mkdir /tmp/test
     #查看创建的文件夹的权限
     hadoop fs -ls /tmp
    ```

```
 drwxr-x---   - test   hadoop          0 2017-11-26 21:18 /tmp/
test
 #设置acl,授权给foo用户rwx
 hadoop fs -setfacl -m user:foo:rwx /tmp/test
 #查看文件权限(+号表示设置了ACL)
 hadoop fs -ls  /tmp/
 drwxrwx---+  - test   hadoop          0 2017-11-26 21:18 /tmp/
test
 #查看acl
  hadoop fs -getfacl  /tmp/test
  # file: /tmp/test
# owner: test
# group: hadoop
user::rwx
user:foo:rwx
group::r-x
mask::rwx
other::---
```

- dfs.permissions.superusergroup

  超级用户组，属于该组的用户都具有超级用户的权限

重启HDFS服务

对于Kerberos安全集群，已经默认设置了HDFS的权限(umask设置为027)，无需配置和重启服务。

对于非Kerberos安全集群需要添加配置并重启服务

其它

- umask值可以根据需求自行修改
- HDFS是一个基础的服务，Hive/HBase等都是基于HDFS，所以在配置其它上层服务时，需要提前配置好HDFS的权限控制。
- 在HDFS开启权限后，需要设置好服务的(如spark的/spark-history、yarn的/tmp/$user/等)
- sticky bit:

  针对文件夹可设置sticky bit，可以防止除了superuser/file owner/dir owner之外的其它用户删除该文件夹中的文件/文件夹(即使其它用户对该文件夹有rwx权限)。如:

```
#即在第一位添加数字1
hadoop fs -chmod 1777 /tmp
hadoop fs -chmod 1777 /spark-history
```

```
hadoop fs -chmod 1777 /user/hive/warehouse
```

19.2 YARN授权

YARN的授权根据授权实体，可以分为服务级别的授权、队列级别的授权。

服务级别的授权

详见*Hadoop*官方文档

- 控制特定用户访问集群服务，如提交作业
- 配置在hadoop-policy.xml
- 服务级别的权限校验在其他权限校验之前(如HDFS的permission检查/yarn提交作业到队列控制)

> 说明：
> 一般设置了HDFS permission检查/yarn队列资源控制，可以不设置服务级别的授权控制，用户可以根据自己需求进行相关配置。

队列级别的授权

YARN可以通过队列对资源进行授权管理，有两种队列调度 Capacity Scheduler和Fair Scheduler。

这里以Capacity Scheduler为例。

- 添加配置

  队列也有两个级别的授权，一个是提交作业到队列的授权，一个是管理队列的授权

  > 说明：
  > - 队列的ACL的控制对象为user/group，设置相关参数时，user和group可同时设置，中间用空格分开，user/group内部可用逗号分开，只有一个空格表示任何人都没有权限。
  > - 队列ACL继承: 如果一个user/group可以向某个队列中提交应用程序，则它可以向它的所有子队列中提交应用程序，同理管理队列的ACL也具有继承性。所以如果要防止某个user/group提交作业到某个队列，则需要设置该队列以及该队列的所有父队列的ACL来限制该user/group的提交作业的权限。

  - yarn.acl.enable

    ACL开关，设置为true

  - yarn.admin.acl

- yarn的管理员设置，如可执行`yarn rmadmin/yarn kill`等命令，该值必须配置，否则后续的队列相关的acl管理员设置无法生效。
- 如上备注，配置值时可以设置user/group：

```
user1,user2 group1,group2 #user和group用空格隔开
  group1,group2 #只有group情况下，必须在最前面加上空格
```

EMR集群中需将has配置为admin的acl权限

- yarn.scheduler.capacity.${queue-name}.acl_submit_applications
  - 设置能够向该队列提交的user/group。
  - 其中${queue-name}为队列的名称，可以是多级队列，注意多级情况下的ACL继承机制。如:

```
#queue-name=root
  <property>
      <name>yarn.scheduler.capacity.root.acl_submit_applicati
ons</name>
      <value> </value> #空格表示任何人都无法往root队列提交作业
  </property>
 #queue-name=root.testqueue
 <property>
   <name>yarn.scheduler.capacity.root.testqueue.acl_submit
_applications</name>
      <value>test testgrp</value> #testqueue只允许test用户/
testgrp组提交作业
  </property>
```

- yarn.scheduler.capacity.${queue-name}.acl_administer_queue
  - 设置某些user/group管理队列，比如kill队列中作业等。
  - queue-name可以是多级，注意多级情况下的ACL继承机制。

```
#queue-name=root
  <property>
      <name>yarn.scheduler.capacity.root.acl_administer_queue</
name>
      <value> </value>
  </property>
 #queue-name=root.testqueue
 <property>
   <name>yarn.scheduler.capacity.root.testqueue.acl_admini
ster_queue</name>
      <value>test testgrp</value>
  </property>
```

- 重启YARN服务

- 对于Kerberos安全集群已经默认开启ACL，用户可以根据自己需求配置队列的相关ACL权限控制。
- 对于非Kerberos安全集群根据上述开启ACL并配置好队列的权限控制，重启YARN服务。

- 配置示例
  - yarn-site.xml

```
<property>
        <name>yarn.acl.enable</name>
        <value>true</value>
 </property>
<property>
        <name>yarn.admin.acl</name>
        <value>has</value>
 </property>
```

  - capacity-scheduler.xml
  - default队列: 禁用default队列，不允许任何用户提交或管理。
  - q1队列: 只允许test用户提交作业以及管理队列(如kill)。
  - q2队列: 只允许foo用户提交作业以及管理队列。

```
<configuration>
    <property>
        <name>yarn.scheduler.capacity.maximum-applications</name>
        <value>10000</value>
        <description>Maximum number of applications that can be
pending and running.</description>
    </property>
    <property>
        <name>yarn.scheduler.capacity.maximum-am-resource-percent</
name>
        <value>0.25</value>
        <description>Maximum percent of resources in the cluster
which can be used to run application masters i.e.
            controls number of concurrent running applications.
        </description>
    </property>
    <property>
        <name>yarn.scheduler.capacity.resource-calculator</name>
        <value>org.apache.hadoop.yarn.util.resource.DefaultRes
ourceCalculator</value>
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.queues</name>
        <value>default,q1,q2</value>
        <!-- 3个队列-->
        <description>The queues at the this level (root is the root
queue).</description>
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.default.capacity</name>
        <value>0</value>
        <description>Default queue target capacity.</description>
```

```
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.default.user-limit-factor
</name>
        <value>1</value>
        <description>Default queue user limit a percentage from 0.0
to 1.0.</description>
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.default.maximum-capacity
</name>
        <value>100</value>
        <description>The maximum capacity of the default queue.</
description>
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.default.state</name>
        <value>STOPPED</value>
        <!-- default队列状态设置为STOPPED-->
        <description>The state of the default queue. State can be
one of RUNNING or STOPPED.</description>
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.default.acl_submit
_applications</name>
        <value> </value>
        <!-- default队列禁止提交作业-->
        <description>The ACL of who can submit jobs to the default
queue.</description>
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.default.acl_admini
ster_queue</name>
        <value> </value>
        <!-- 禁止管理default队列-->
        <description>The ACL of who can administer jobs on the
default queue.</description>
    </property>
    <property>
        <name>yarn.scheduler.capacity.node-locality-delay</name>
        <value>40</value>
    </property>
    <property>
        <name>yarn.scheduler.capacity.queue-mappings</name>
        <value>u:test:q1,u:foo:q2</value>
        <!-- 队列映射，test用户自动映射到q1队列-->
        <description>A list of mappings that will be used to assign
jobs to queues. The syntax for this list is
            [u|g]:[name]:[queue_name][,next mapping]* Typically this
 list will be used to map users to queues,for
            example, u:%user:%user maps all users to queues with the
 same name as the user.
        </description>
    </property>
    <property>
        <name>yarn.scheduler.capacity.queue-mappings-override.enable
</name>
        <value>true</value>
        <!-- 上述queue-mappings设置的映射，是否覆盖客户端设置的队列参数-->
        <description>If a queue mapping is present, will it override
 the value specified by the user? This can be used
```

```
            by administrators to place jobs in queues that are
different than the one specified by the user. The default
            is false.
        </description>
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.acl_submit_applications</
name>
        <value> </value>
        <!-- ACL继承性，父队列需控制住权限-->
        <description>
            The ACL of who can submit jobs to the root queue.
        </description>
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.q1.acl_submit_applicati
ons</name>
        <value>test</value>
        <!-- q1只允许test用户提交作业-->
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.q2.acl_submit_applicati
ons</name>
        <value>foo</value>
        <!-- q2只允许foo用户提交作业-->
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.q1.maximum-capacity</name
>
        <value>100</value>
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.q2.maximum-capacity</name
>
        <value>100</value>
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.q1.capacity</name>
        <value>50</value>
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.q2.capacity</name>
        <value>50</value>
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.acl_administer_queue</
name>
        <value> </value>
        <!-- ACL继承性，父队列需控制住权限-->
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.q1.acl_administer_queue</
name>
        <value>test</value>
        <!-- q1队列只允许test用户管理，如kill作业-->
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.q2.acl_administer_queue</
name>
        <value>foo</value>
        <!-- q2队列只允许foo用户管理，如kill作业-->
```

```
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.q1.state</name>
        <value>RUNNING</value>
    </property>
    <property>
        <name>yarn.scheduler.capacity.root.q2.state</name>
        <value>RUNNING</value>
    </property>
</configuration>
```

19.3 Hive授权

Hive内置有基于底层HDFS的权限(Storage Based Authorization)和基于标准SQL的grant等命令(SQL Standards Based Authorization)两种授权机制，详见*Hive*官方文档。

说明：

两种授权机制可以同时配置，不冲突。

Storage Based Authorization(针对HiveMetaStore)

场景:

如果集群的用户直接通过HDFS/Hive Client访问Hive的数据，需要对Hive在HDFS中的数据进行相关的权限控制，通过HDFS权限控制，进而可以控制Hive SQL相关的操作权限。详见*Hive*文档

添加配置

在集群的配置管理页面选择**Hive** > 配置 > **hive-site.xml** > 自定义配置

```
<property>
<name>hive.metastore.pre.event.listeners</name>
    <value>org.apache.hadoop.hive.ql.security.authorization.Authorizat
ionPreEventListener</value>
</property>
<property>
<name>hive.security.metastore.authorization.manager</name>
    <value>org.apache.hadoop.hive.ql.security.authorization.StorageBas
edAuthorizationProvider</value>
</property>
<property>
<name>hive.security.metastore.authenticator.manager</name>
    <value>org.apache.hadoop.hive.ql.security.HadoopDefaultMetasto
reAuthenticator</value>
</property>
```

重启HiveMetaStore

在集群配置管理页面重启HiveMetaStore

HDFS权限控制

EMR的Kerberos安全集群已经设置了Hive的warehouse的HDFS相关权限；

对于非Kerberos安全集群，用户需要做如下步骤设置hive基本的HDFS权限:

- 打开HDFS的权限
- 配置Hive的warehouse权限

```
hadoop fs -chmod 1771 /user/hive/warehouse
也可以设置成，1表示stick bit(不能删除别人创建的文件/文件夹)
hadoop fs -chmod 1777 /user/hive/warehouse
```

有了上述设置基础权限后，可以通过对warehouse文件夹授权，让相关用户/用户组能够正常创建表/读写表等

```
sudo su has
      #授予test对warehouse文件夹rwx权限
      hadoop fs -setfacl -m user:test:rwx /user/hive/warehouse
      #授予hivegrp对warehouse文件夹rwx权限
      hadoo fs -setfacl -m group:hivegrp:rwx /user/hive/warehouse
```

经过HDFS的授权后，让相关用户/用户组能够正常创建表/读写表等，而且 不同的账号创建的hive表在HDFS中的数据只能自己的账号才能访问。

验证

- test用户建表testtbl

```
hive> create table testtbl(a string);
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive
.ql.exec.DDLTask. MetaException(message:Got exception: org.apache.
hadoop.security.AccessControlException Permission denied: user=test
, access=WRITE, inode="/user/hive/warehouse/testtbl":hadoop:hadoop:
drwxrwx--t
at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(
FSPermissionChecker.java:320)
at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(
FSPermissionChecker.java:292)
```

上面显示错误没有权限，需要给test用户添加权限

```
#从root账号切到has账号
su has
#给test账号添加acl，对warehouse目录的rwx权限
 hadoop fs -setfacl -m user:test:rwx /user/hive/warehouse
```

test账号再创建database，成功

```
hive> create table testtbl(a string);
```

```
OK
Time taken: 1.371 seconds
#查看hdfs中testtbl的目录，从权限可以看出test用户创建的表数据只有test和hadoop
组可以读取，其他用户没有任何权限
hadoop fs -ls /user/hive/warehouse
drwxr-x---   - test hadoop          0 2017-11-25 14:51 /user/hive/
warehouse/testtbl
#插入一条数据
hive>insert into table testtbl select "hz"
```

- foo用户访问testtbl

```
#drop table
hive> drop table testtbl;
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.
ql.exec.DDLTask. MetaException(message:Permission denied: user=foo,
access=READ, inode="/user/hive/warehouse/testtbl":test:hadoop:drwxr-
x---
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.
check(FSPermissionChecker.java:320)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.
checkPermission(FSPermissionChecker.java:219)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.
checkPermission(FSPermissionChecker.java:190)
#alter table
hive> alter table testtbl add columns(b string);
FAILED: SemanticException Unable to fetch table testtbl. java.
security.AccessControlException: Permission denied: user=foo, access
=READ, inode="/user/hive/warehouse/testtbl":test:hadoop:drwxr-x---
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.
check(FSPermissionChecker.java:320)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.
checkPermission(FSPermissionChecker.java:219)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.
checkPermission(FSPermissionChecker.java:190)
    at org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPermi
ssion(FSDirectory.java:1720)
#select
hive> select * from testtbl;
FAILED: SemanticException Unable to fetch table testtbl. java.
security.AccessControlException: Permission denied: user=foo, access
=READ, inode="/user/hive/warehouse/testtbl":test:hadoop:drwxr-x---
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.
check(FSPermissionChecker.java:320)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.
checkPermission(FSPermissionChecker.java:219)
```

可见foo用户不能对test用户创建的表做任何的操作，如果想要授权给foo，需要通过HDFS的授权来实现

```
su has
#只授权读的权限，也可以根据情况授权写权限(比如alter)
#备注: -R 将testtbl文件夹下的文件也设置可读
hadoop fs -setfacl -R -m user:foo:r-x /user/hive/warehouse/testtbl
#可以select成功
hive> select * from testtbl;
OK
hz
```

```
Time taken: 2.134 seconds, Fetched: 1 row(s)
```

说明：

一般可以根据需求新建一个hive用户的group，然后通过给group授权，后续将新用户添加到group中，同一个group的数据权限都可以访问。

SQL Standards Based Authorization

- 场景

  如果集群的用户不能直接通过hdfs/hive client访问，只能通过`HiveServer(beeline/jdbc等)`来执行hive相关的命令，可以使用 SQL Standards Based Authorization的授权方式。

  如果用户能够使用hive shell等方式，即使做下面的一些设置操作，只要用户客户端的hive-site.xml没有相关配置，都可以正常访问hive。

  详见*Hive*文档

- 添加配置

  - 配置是提供给HiveServer
  - 在集群与服务管理页面选择**Hive** > 配置 > **hive-site.xml** > 自定义配置

```
<property>
<name>hive.security.authorization.enabled</name>
 <value>true</value>
</property>
 <property>
<name>hive.users.in.admin.role</name>
 <value>hive</value>
</property>
 <property>
<name>hive.security.authorization.createtable.owner.grants</name>
 <value>ALL</value>
</property>
```

- 重启HiveServer2

  在集群配置管理页面重启HiveServer2

- 权限操作命令

  具体命令操作详见

- 验证

  - 用户foo通过beeline访问test用户的testtbl表

```
2: jdbc:hive2://emr-header-1.cluster-xxx:10> select * from testtbl
;
```

```
Error: Error while compiling statement: FAILED: HiveAccess
ControlException Permission denied: Principal [name=foo, type=USER
] does not have following privileges for operation QUERY [[SELECT
] on Object [type=TABLE_OR_VIEW, name=default.testtbl]] (state=
42000,code=40000)
```

- grant权限

```
切到test账号执行grant给foo授权select操作
hive> grant select on table testtbl to user foo;
 OK
 Time taken: 1.205 seconds
```

- foo可以正常select

```
0: jdbc:hive2://emr-header-1.cluster-xxxxx:10> select * from
testtbl;
INFO  : OK
+------------+--+
| testtbl.a  |
+------------+--+
| hz         |
+------------+--+
1 row selected (0.787 seconds)
```

- 回收权限

```
切换到test账号，回收权限foo的select权限
  hive> revoke select from user foo;
    OK
    Time taken: 1.094 seconds
```

- foo无法select testtbl的数据

```
0: jdbc:hive2://emr-header-1.cluster-xxxxx:10> select * from
testtbl;
Error: Error while compiling statement: FAILED: HiveAccess
ControlException Permission denied: Principal [name=foo, type=USER
] does not have following privileges for operation QUERY [[SELECT
] on Object [type=TABLE_OR_VIEW, name=default.testtbl]] (state=
42000,code=40000)
```

19.4 HBase授权

HBase在不开启授权的情况下，任何账号对HBase集群可以进行任何操作，比如disable table/drop table/major compact等等。

说明：

对于没有Kerberos认证的集群，即使开启了HBase授权，用户也可以伪造身份访问集群服务。所以建议创建高安全模式(即支持Kerberos)的集群，详见*Kerberos*安全文档。

添加配置

在HBase集群的集群与服务管理页面选择**HBase** > 配置 > **hbase-site** > 自定义配置

添加如下几个参数:

```
<property>
     <name>hbase.security.authorization</name>
     <value>true</value>
</property>
<property>
     <name>hbase.coprocessor.master.classes</name>
     <value>org.apache.hadoop.hbase.security.access.AccessController</
value>
</property>
<property>
     <name>hbase.coprocessor.region.classes</name>
 <value>org.apache.hadoop.hbase.security.token.TokenProvider,org.
apache.hadoop.hbase.security.access.AccessController</value>
</property>
<property>
  <name>hbase.coprocessor.regionserver.classes</name>
  <value>org.apache.hadoop.hbase.security.access.AccessController,org.
apache.hadoop.hbase.security.token.TokenProvider</value>
</property>
```

重启HBase集群

在HBase集群的集群与服务管理页面选择**HBase** > 操作 > **RESTART All Components**

授权(ACL)

- 基本概念

  授权就是将对 [某个范围的资源] 的 [操作权限] 授予[某个实体]

  在HBase中，上述对应的三个概念分别为:

  - 某个范围(Scope)的资源
    - Superuser

      超级账号可以进行任何操作，运行HBase服务的账号默认是Superuser。也可以通过在hbase-site.xml中配置hbase.superuser的值可以添加超级账号
    - Global

      Global Scope拥有集群所有table的Admin权限
    - Namespace

      在Namespace Scope进行相关权限控制
    - Table

在Table Scope进行相关权限控制

- ColumnFamily

  在ColumnFamily Scope进行相关权限控制

- Cell

  在Cell Scope进行相关权限控制

- 操作权限
  - Read (R)

    读取某个Scope资源的数据

  - Write (W)

    写数据到某个Scope的资源

  - Execute (X)

    在某个Scope执行协处理器

  - Create (C)

    在某个Scope创建/删除表等操作

  - Admin (A)

    在某个Scope进行集群相关操作，如balance/assign等

- 某个实体
  - User

    对某个用户授权

  - Group

    对某个用户组授权

- 授权命令
  - grant 授权

```
grant <user> <permissions> [<@namespace> [<table> [<column family>
 [<column qualifier>]]]
```

说明：

- user/group的授权方式一样，group需要加一个前缀@

```
grant 'test','R','tbl1'    #给用户test授予表tbl1的读权限
  grant '@testgrp','R','tbl1' #给用户组testgrp授予表tbl1的读权限
```

- namespace需要加一个前缀@

```
grant 'test 'C','@ns_1'  #给用户test授予namespace ns_1的CREATE权限
```

- revoke 回收
- user_permissions 查看权限

19.5 Kafka授权

如果没有开启Kafka认证(如Kerberos认证或者简单的用户名密码)，即使开启了Kafka授权，用户也可以伪造身份访问服务。所以建议创建高安全模式(即支持Kerberos)的Kafka集群，详见*Kerberos安全文档*。

说明：

本文的权限配置只针对E-MapReduce的高安全模式集群，即Kafka以Kerberos的方式启动。

添加配置

1. 在集群管理页面，在需要添加配置的Kafka集群后单击**详情**。
2. 在左侧导航栏中单击**集群与服务管理**页签，然后单击服务列表中的**Kafka**。
3. 在上方单击**配置**页签。
4. 在**服务配置**列表的右上角单击**自定义配置**，添加如下几个参数:

key	value	备注
authorizer.class.name	kafka.security.auth.SimpleAclAuthorizer	无
super.users	User:kafka	User:kafka是必须的，可添加其它用户用分号(;)隔开

说明：

zookeeper.set.acl用来设置Kafka在ZooKeeper中数据的权限，E-MapReduce集群中已经设置为true，所以此处不需要再添加该配置。该配置设置为true后，在Kerberos环境中，只有用户

名称为kafka且通过Kerberos认证后才能执行kafka-topics.sh命令(kafka-topics.sh会直接读写/修改ZooKeeper中的数据)。

重启Kafka集群

1. 在集群管理页面，在需要添加配置的Kafka集群后单击详情。
2. 在左侧导航栏中单击集群与服务管理页签，然后单击服务列表中的Kafka右侧的操作。
3. 在下拉菜单中选择**RESTART All Components**，输入记录信息后单击确定。

授权(ACL)

- 基本概念

  Kafka官方文档定义:

  ```
  Kafka acls are defined in the general format of "Principal P is [
  Allowed/Denied] Operation O From Host H On Resource R"
  ```

  即ACL过程涉及Principal、Allowed/Denied、Operation、Host和Resource。

  - Principal：用户名

安全协议	**value**
PLAINTEXT	ANONYMOUS
SSL	ANONYMOUS
SASL_PLAINTEXT	mechanism为PLAIN时，用户名是client_jaas.conf指定的用户名，mechanism为GSSAPI时，用户名为client_jaas.conf指定的principal
SASL_SSL	

  - Allowed/Denied：允许/拒绝。
  - Operation： 操作，包括Read、Write、Create、DeleteAlter、Describe、ClusterAction、AlterConfigs、DescribeConfigs、IdempotentWrite和All。
  - Host： 针对的机器。
  - Resource： 权限作用的资源对象，包括Topic、Group、Cluster和TransactionalId。

  Operation/Resource的一些详细对应关系，如哪些Resource支持哪些Operation的授权，详见 *KIP-11 - Authorization Interface*。

- 授权命令

  我们使用脚本kafka-acls.sh (/usr/lib/kafka-current/bin/kafka-acls.sh) 进行Kafka授权，您可以直接执行 `kafka-acls.sh --help`查看如何使用该命令。

操作示例

在已经创建的E-MapReduce高安全Kafka集群的master节点上进行相关示例操作。

1. 新建用户test，执行以下命令。

```
useradd test
```

2. 创建topic。

第一节添加配置的备注中提到zookeeper.set.acl=true，kafka-topics.sh需要在kafka账号下执行，而且kafka账号下要通过Kerberos认证。

```
# kafka_client_jaas.conf中已经设置了kafka的Kerberos认证相关信息
export KAFKA_HEAP_OPTS="-Djava.security.auth.login.config=/etc/ecm/
kafka-conf/kafka_client_jaas.conf"
# zookeeper地址改成自己集群的对应地址(执行`hostnamed`后即可获取)
kafka-topics.sh --create --zookeeper emr-header-1:2181/kafka-1.0.0
 --replication-factor 3 --partitions 1 --topic test
```

3. test用户执行kafka-console-producer.sh。

a. 创建test用户的keytab文件，用于zookeeper/kafka的认证。

```
su root
sh /usr/lib/has-current/bin/hadmin-local.sh /etc/ecm/has-conf -k /
etc/ecm/has-conf/admin.keytab
HadminLocalTool.local: #直接按回车可以看到一些命令的用法
HadminLocalTool.local: addprinc #输入命令按回车可以看到具体命令的用法
HadminLocalTool.local: addprinc -pw 123456 test #添加test的
principal,密码设置为123456
HadminLocalTool.local: ktadd -k /home/test/test.keytab test #导出
keytab文件，后续可使用该文件
```

b. 添加kafka_client_test.conf。

如文件放到*/home/test/kafka_client_test.conf*，内容如下:

```
KafkaClient {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
storeKey=true
serviceName="kafka"
keyTab="/home/test/test.keytab"
principal="test";
};
// Zookeeper client authentication
Client {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
useTicketCache=false
serviceName="zookeeper"
keyTab="/home/test/test.keytab"
principal="test";
```

```
};
```

c. 添加producer.conf。

如文件放到/home/test/producer.conf，内容如下:

```
security.protocol=SASL_PLAINTEXT
sasl.mechanism=GSSAPI
```

d. 执行kafka-console-producer.sh。

```
su test
export KAFKA_HEAP_OPTS="-Djava.security.auth.login.config=/home/
test/kafka_client_test.conf"
kafka-console-producer.sh --producer.config /home/test/producer.
conf --topic test --broker-list emr-worker-1:9092
```

由于没有设置ACL，所以上述会报错:

```
org.apache.kafka.common.errors.TopicAuthorizationException: Not
authorized to access topics: [test]
```

e. 设置ACL。

同样kafka-acls.sh也需要kafka账号执行。

```
su kafka
export KAFKA_HEAP_OPTS="-Djava.security.auth.login.config=/etc/ecm
/kafka-conf/kafka_client_jaas.conf"
kafka-acls.sh --authorizer-properties zookeeper.connect=emr-header
-1:2181/kafka-1.0.0 --add --allow-principal User:test --operation
Write --topic test
```

f. 再执行kafka-console-producer.sh。

```
su test
export KAFKA_HEAP_OPTS="-Djava.security.auth.login.config=/home/
test/kafka_client_test.conf"
kafka-console-producer.sh --producer.config /home/test/producer.
conf --topic test --broker-list emr-worker-1:9092
```

正常情况下显示如下:

```
[2018-02-28 22:25:36,178] INFO Kafka commitId : aaa7af6d4a11b29d (
org.apache.kafka.common.utils.AppInfoParser)
>alibaba
>E-MapReduce
>
```

4. test用户执行kafka-console-consumer.sh。

上面成功执行kafka-console-producer.sh,并往topic里面写入一些数据后，就可以执行kafka-console-consumer.sh进行消费测试.

a. 添加consumer.conf。

如文件放到*/home/test/consumer.conf*，内容如下:

```
security.protocol=SASL_PLAINTEXT
sasl.mechanism=GSSAPI
```

b. 执行kafka-console-consumer.sh。

```
su test
#kafka_client_test.conf跟上面producer使用的是一样的
export KAFKA_HEAP_OPTS="-Djava.security.auth.login.config=/home/
test/kafka_client_test.conf"
kafka-console-consumer.sh --consumer.config consumer.conf  --topic
 test --bootstrap-server emr-worker-1:9092 --group test-group  --
from-beginning
```

由于未设置权限，会报错:

```
org.apache.kafka.common.errors.GroupAuthorizationException: Not
authorized to access group: test-group
```

c. 设置ACL。

```
su kafka
export KAFKA_HEAP_OPTS="-Djava.security.auth.login.config=/etc/ecm
/kafka-conf/kafka_client_jaas.conf"
# test-group权限
kafka-acls.sh --authorizer-properties zookeeper.connect=emr-header
-1:2181/kafka-1.0.0 --add --allow-principal User:test --operation
Read --group test-group
# topic权限
kafka-acls.sh --authorizer-properties zookeeper.connect=emr-header
-1:2181/kafka-1.0.0 --add --allow-principal User:test --operation
Read --topic test
```

d. 再执行kafka-console-consumer.sh。

```
su test
# kafka_client_test.conf跟上面producer使用的是一样的
export KAFKA_HEAP_OPTS="-Djava.security.auth.login.config=/home/
test/kafka_client_test.conf"
kafka-console-consumer.sh --consumer.config consumer.conf  --topic
 test --bootstrap-server emr-worker-1:9092 --group test-group  --
from-beginning
```

正常输出:

```
alibaba
```

```
E-MapReduce
```

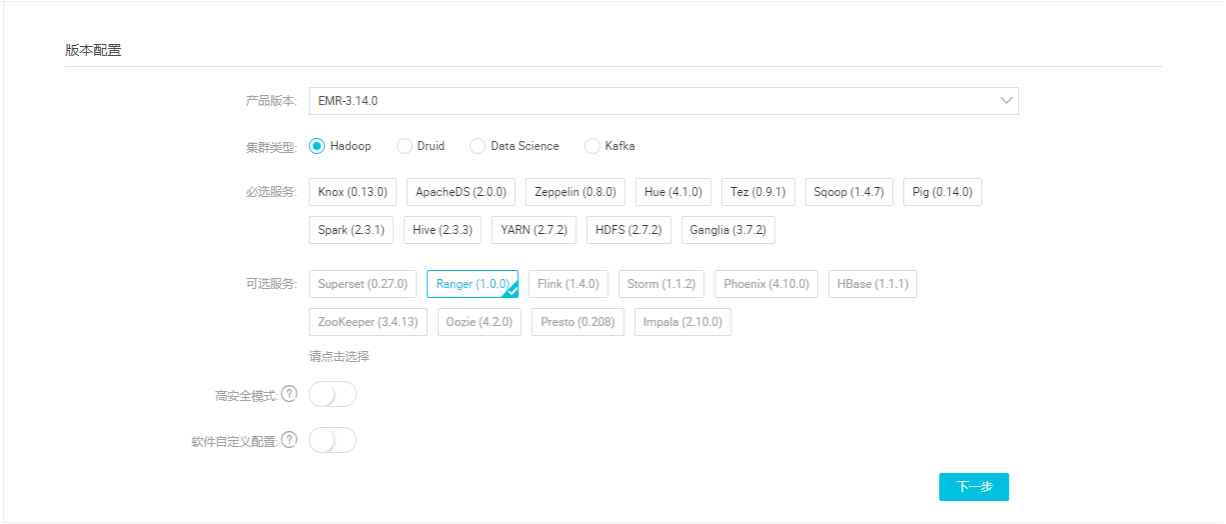
19.6 RANGER

19.6.1 Ranger简介

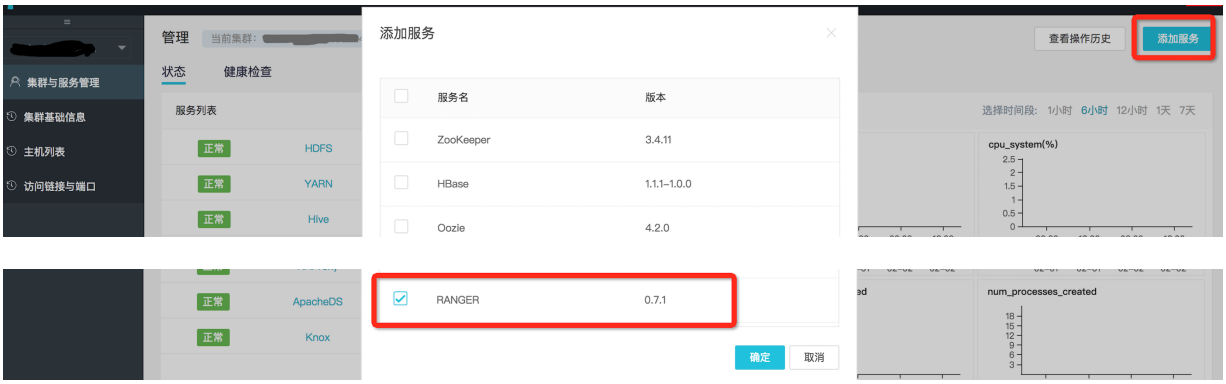
Apache Ranger提供集中式的权限管理框架，可以对Hadoop生态中的HDFS/Hive/YARN/Kafka/Storm/Solr等组件进行细粒度的权限访问控制，并且提供了Web UI方便管理员进行操作。

创建集群

在E-MapReduce控制台创建EMR-2.9.2/EMR-3.9.0及以上的集群，勾选Ranger组件即可，如下图所示:



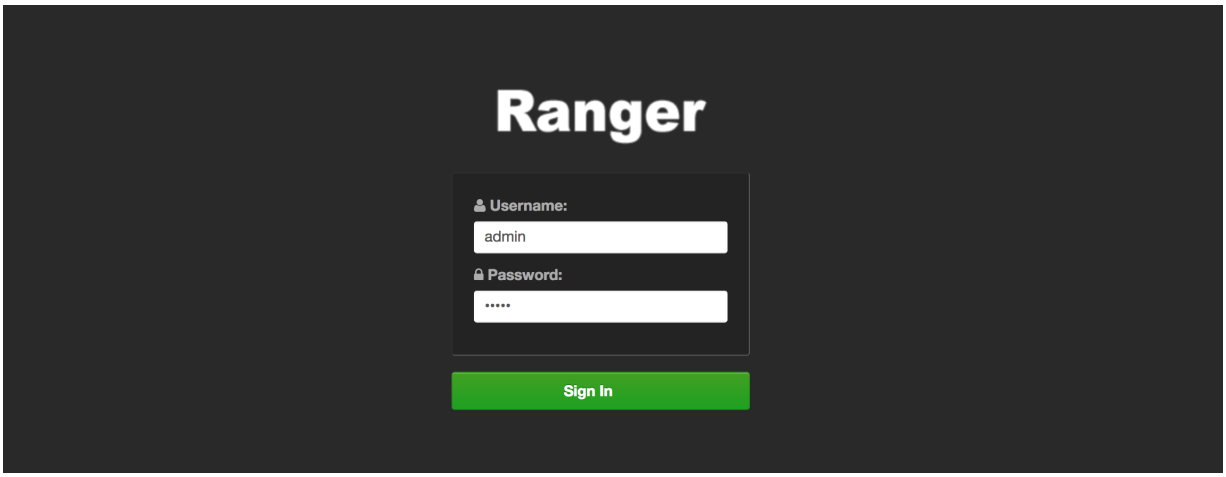
如果已经创建EMR-2.9.2/EMR-3.9.0及以上的集群，可以在集群的集群与服务管理页面添加Ranger服务:

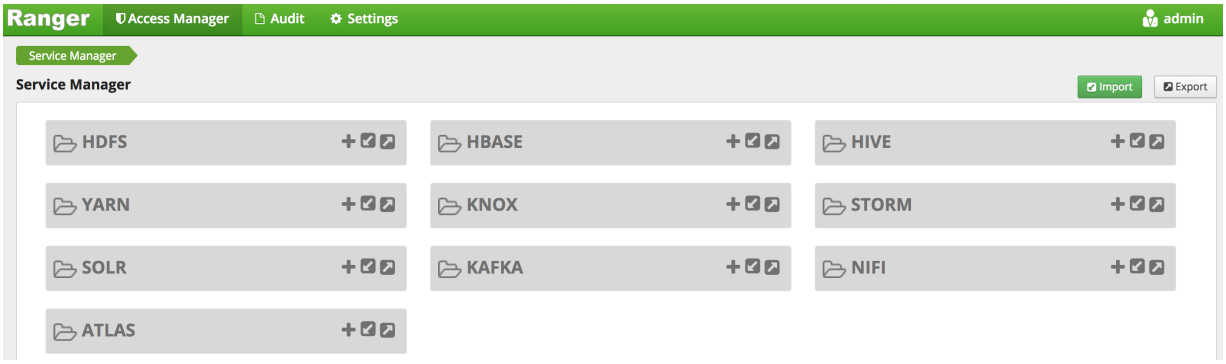


Ranger UI

在安装了Ranger的集群后，单击管理，然后在左侧导航栏中选择**访问链接与端口**，可以通过快捷链接访问Ranger UI，如下图所示:

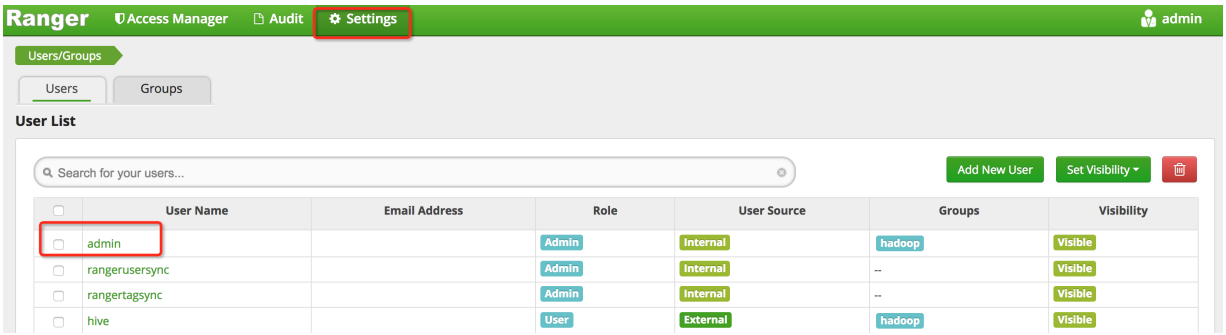
单击链接后会进入Ranger UI登录界面，默认的用户名/密码是admin/admin，如下图：

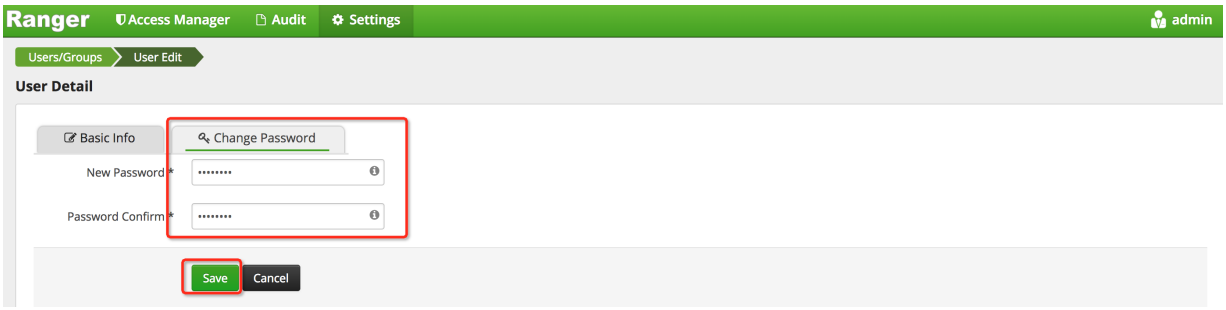




修改密码

管理员首次登录后，需要修改admin的密码，如下图：





改完admin密码后，单击右上角**admin**下拉菜单的**Log Out**，然后使用新的密码登录即可。

组件集成Ranger

经过上述步骤，可以将集群中的相关组件集成到Ranger，进行相关权限的控制，详情请参见：

- *HDFS集成Ranger*
- *Hive集成Ranger*
- *HBase集成Ranger*

19.6.2 HDFS配置

前面简介中介绍了E-MapReduce中创建启动Ranger服务的集群，以及一些准备工作，本节介绍HDFS集成Ranger的一些步骤流程。

HDFS集成Ranger

- Enable HDFS Plugin

1. 在集群管理页面，在需要操作的集群后单击管理
2. 在服务列表中单击**Ranger**进入Ranger配置页面
3. 在Ranger配置页面，单击右侧的操作下拉菜单，选择**Enable HDFS PLUGIN**。

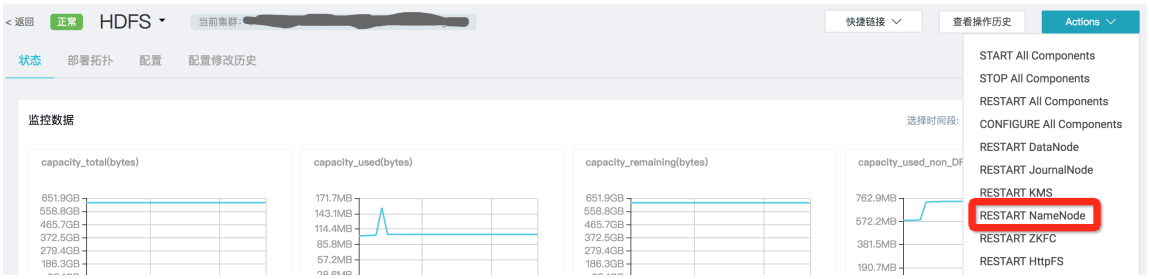
4. 在弹出框输入执行Commit记录，然后单击**确定**。

   单击右上角**查看操作历史**查看任务进度，等待任务完成。

- 重启NameNode

  上述任务完成后，需要重启NameNode才生效。

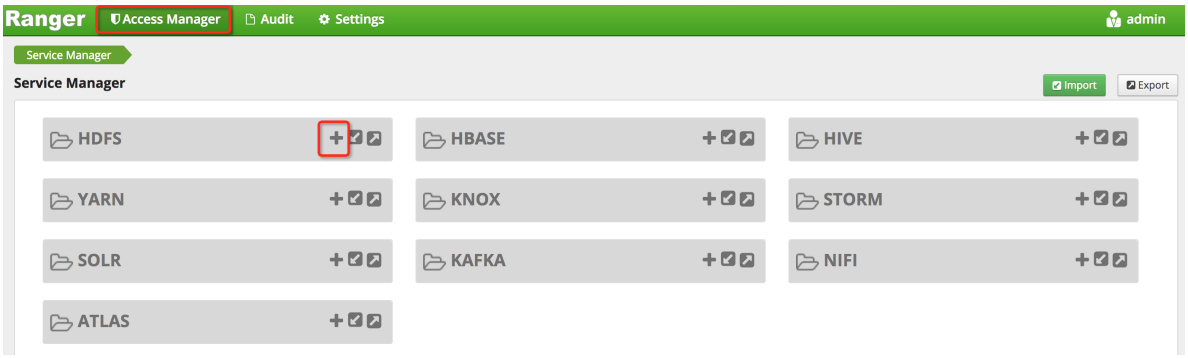
  1. 在Ranger配置页面中，单击左上角Ranger后的倒三角，从Ranger切换到HDFS。
  2. 单击右上角**操作**，从下拉菜单中选择**RESTART NameNode**。
  3. 在弹出框输入执行Commit记录，然后单击**确定**。
  4. 单击右上角的**查看操作历史**查看任务进度，等待重启任务完成。



- Ranger UI添加HDFS service

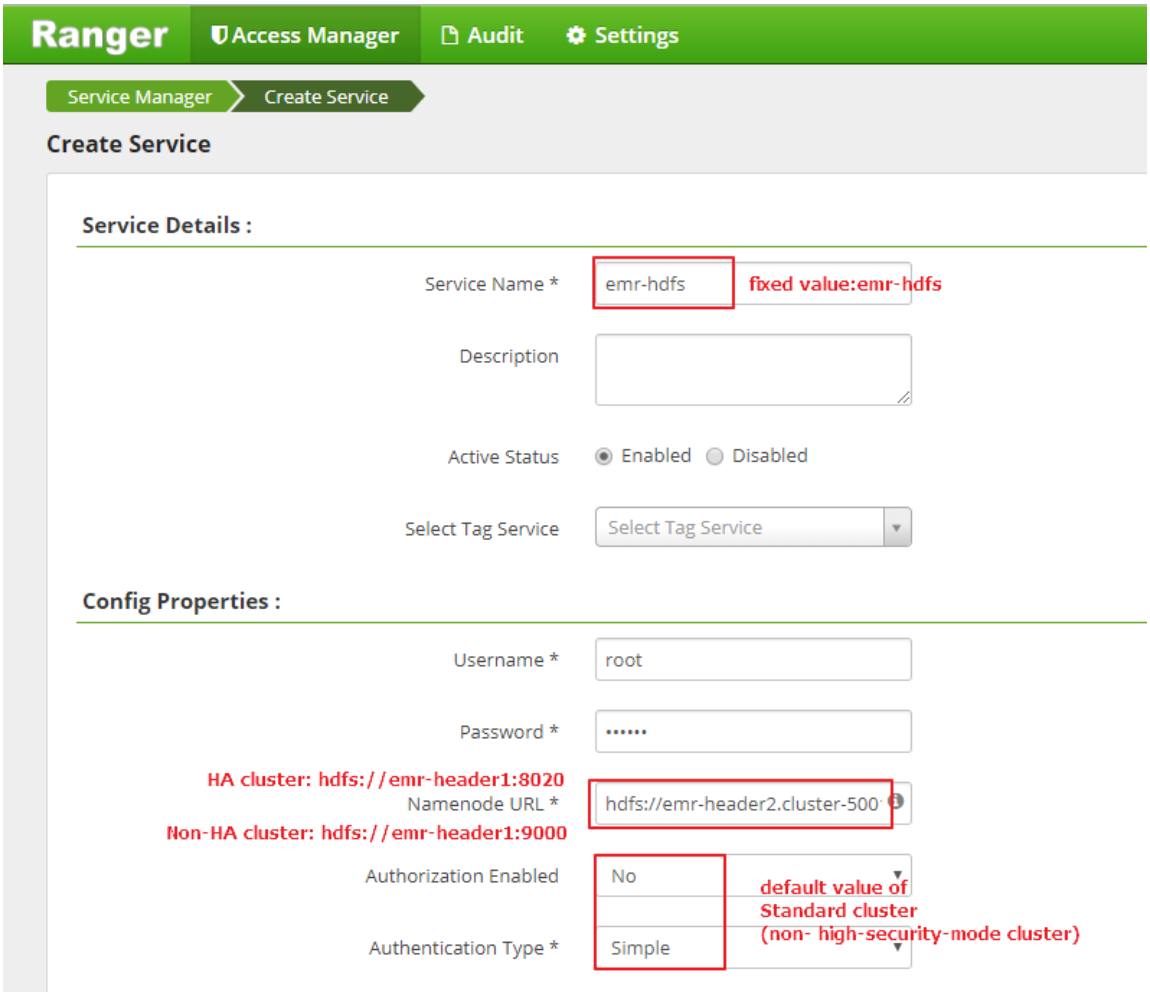
  参见*Ranger简介*的介绍进入Ranger UI。

  在Ranger UI页面添加HDFS Service:



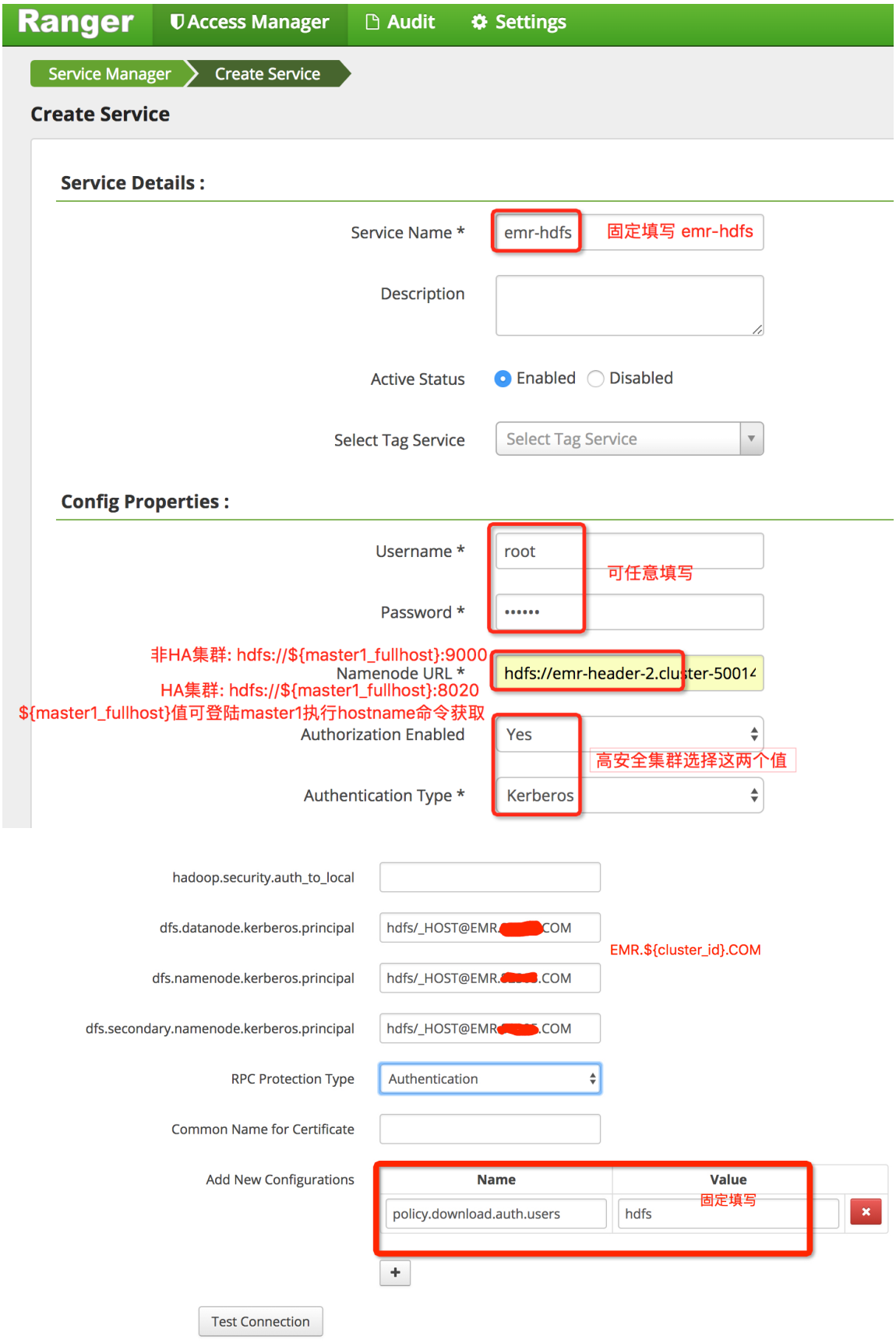
- 标准集群

如果您创建的是标准集群(可到详情页面查看安全模式)，请参考下图进行配置：



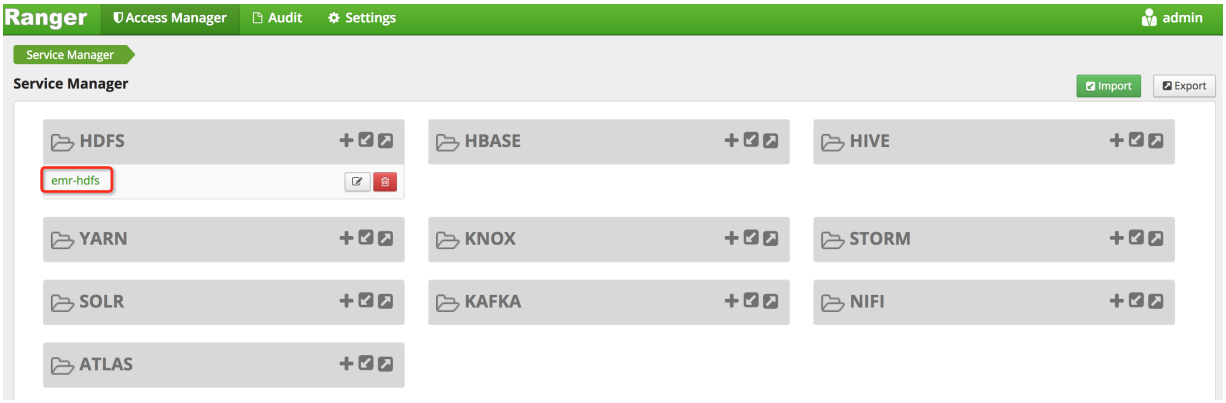
- 高安全集群

如果您创建的是高安全集群(可到详情页面查看安全模式)，请参考下图进行配置：

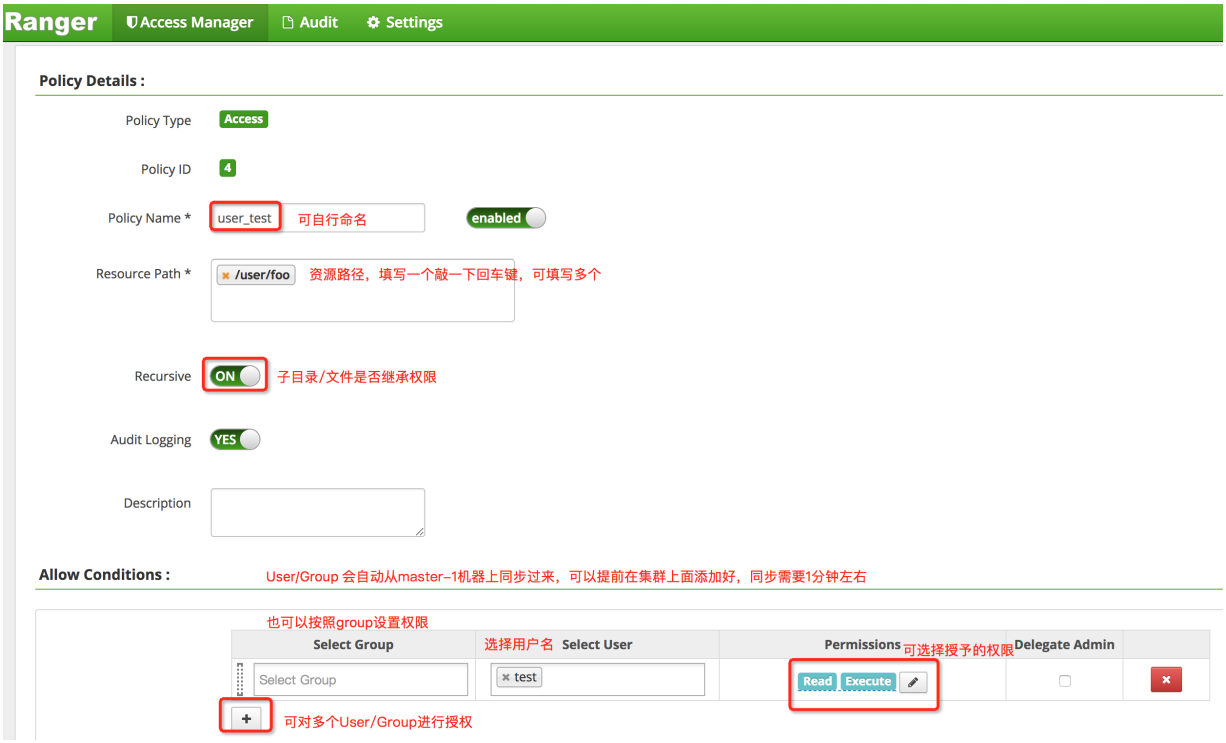

Ranger
Access Manager
Audit
Settings
Service Manager
Create Service
Create Service
Service Details :
Service Name *
emr-hdfs
固定填写 emr-hdfs
Description
Active Status
Enabled
Disabled
Select Tag Service
Select Tag Service
Config Properties :
Username *
root
可任意填写
Password *
非HA集群: hdfs://${master1_fullhost}:9000
Namenode URL *
HA集群: hdfs://${master1_fullhost}:8020
${master1_fullhost}值可登陆master1执行hostname命令获取
hdfs://emr-header-2.cluster-50014
Authorization Enabled
Yes
高安全集群选择这两个值
Authentication Type *
Kerberos
hadoop.security.auth_to_local
dfs.datanode.kerberos.principal
hdfs/_HOST@EMR.COM
EMR.${cluster_id}.COM
dfs.namenode.kerberos.principal
hdfs/_HOST@EMR.COM
dfs.secondary.namenode.kerberos.principal
hdfs/_HOST@EMR.COM
RPC Protection Type
Authentication
Common Name for Certificate
Add New Configurations
Name
Value
固定填写
policy.download.auth.users
hdfs
Test Connection

权限配置示例

上面一节中已经将Ranger集成到HDFS，现在可以进行相关的权限设置。例如给用户test授予/*user/foo*路径的写/执行权限：



单击上图中的**emr-hdfs**进入配置页面，配置相关权限。



按照上述步骤设置添加一个Policy后，就实现了对test的授权，然后用户test就可以对*/user/foo*的HDFS路径进行访问了。

> 说明：
> Policy被添加1分钟左右后，HDFS才会生效。

19.6.3 Hive配置

*Ranger简介*中介绍了E-MapReduce中创建启动Ranger服务的集群，以及一些准备工作，本节介绍Hive集成Ranger的步骤流程。

Hive集成Ranger

- Hive访问模型

  用户可以通过三种方式访问Hive数据，包括HiveServer2、Hive Client和HDFS。

  - HiveServer2方式
    - 场景: 用户只能通过HiveServer2访问Hive数据，不能通过Hive Client和HDFS访问。
    - 方式: 使用beeline客户端或者JDBC代码通过HiveServer2去执行相关的Hive脚本。
    - 权限设置:

      Hive官方自带的(*SQL Standards Based Authorization*)就是针对HiveServer2使用场景进行权限控制的。

      Ranger中对Hive的表/列级别的权限控制也是针对HiveServer2的使用场景，如果用户还可以通过Hive Client或者HDFS访问Hive数据，仅仅对表/列层面做权限控制还不够，还需要下面两种方式的一些设置进一步控制权限。
  - Hive Client方式
    - 场景: 用户可以通过Hive Client访问Hive数据。
    - 方式: 使用Hive Client访问。
    - 权限设置:

      Hive Client会请求HiveMetaStore进行一些DDL操作(如Alter Table Add Columns等)，也会通过提交MapReduce作业读取HDFS中的数据进行处理。

      Hive官方自带的(*Storage Based Authorization*)就是针对Hive Client使用场景进行的权限控制，它会根据SQL中表的HDFS路径的读写权限来决定该用户是否可以进行相关的DDL/DML操作，如`ALTER TABLE test ADD COLUMNS(b STRING)`

      Ranger中可以对Hive表的HDFS路径进行权限控制，加上HiveMetaStore配置Storage Based Authorization，从而可以实现对Hive Client访问场景的权限控制。

      说明：

Hive Client场景的DDL操作权限实际也是通过底层HDFS权限控制，所以如果用户有HDFS权限，则也会有对应表的DDL操作权限(如Drop Table/Alter Table等)。

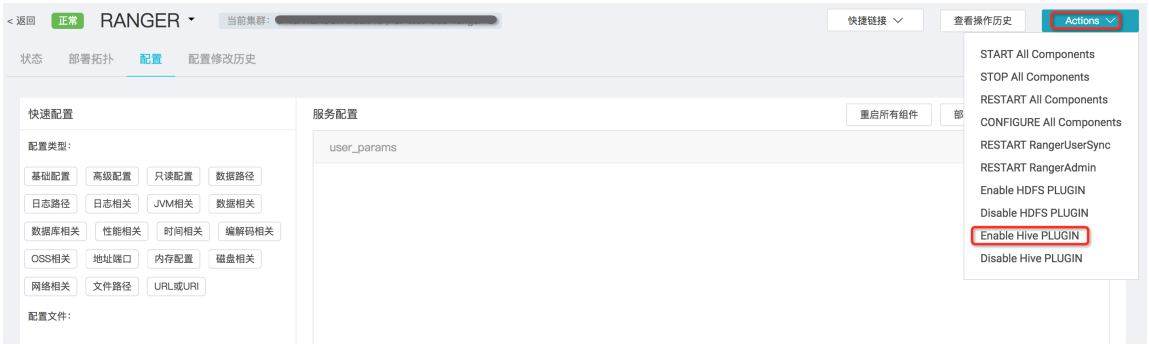
- HDFS方式
  - 场景: 用户可以直接访问HDFS。
  - 方式: HDFS客户端/代码等。
  - 权限设置:

    用户可以直接访问HDFS，则需要对Hive表的底层HDFS数据增加HDFS的权限控制。

    通过Ranger对Hive表底层的HDFS路径进行权限控制。

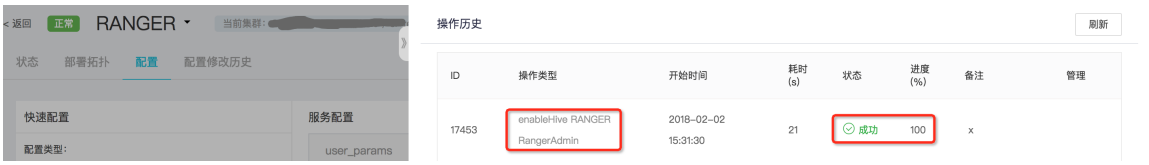
- Enable Hive Plugin

1. 在集群配置页面，在你想操作的集群后的操作栏中单击管理。
2. 在服务列表中单击**Ranger**进入Ranger配置界面。
3. 在Ranger配置页面，单击右侧的操作下拉菜单，选择**Enable Hive PLUGIN**，然后单击确定。



4. 在弹出框输入备注信息，然后单击确定。

   单击右上角查看操作历史查看任务进度，等待任务完成。



说明：

Enable Hive Plugin并重启Hive后，则HiveServer2场景/HiveClient场景(Storage Based Authorization)的都进行了相关的配置，HDFS的权限可参见*HDFS*配置进行开启。

- 重启Hive

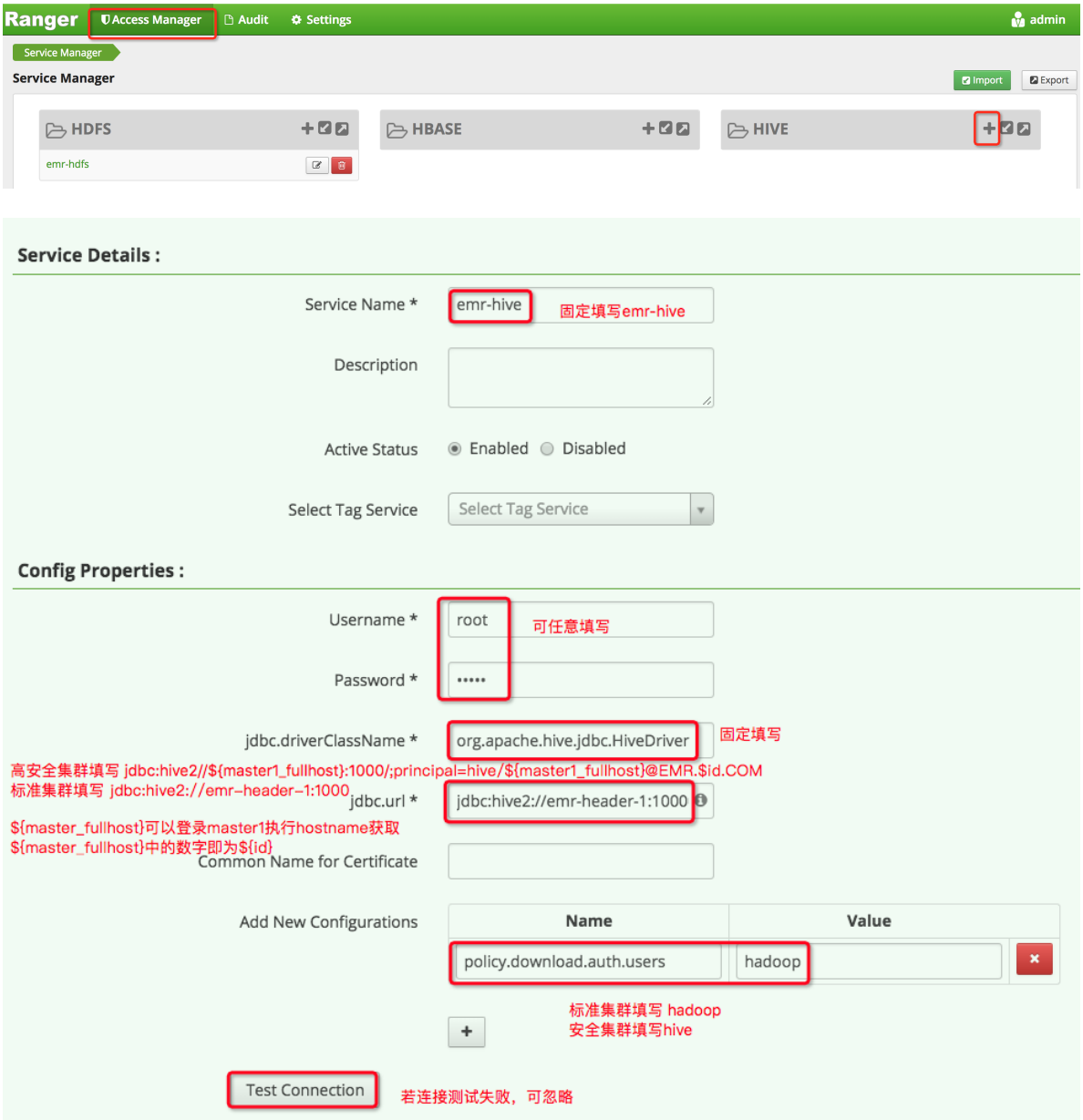
上述任务完成后，需要重启Hive才生效。

1. 在Ranger配置页面中，单击左上角Ranger后的倒三角，从Ranger切换到Hive。
2. 单击右上角操作，从下拉菜单中选择**RESTART All Components**，然后单击确定。
3. 单击右上角的查看操作历史查看任务进度，等待重启任务完成。

- Ranger UI添加Hive service

  参见*Ranger*简介的介绍进入Ranger UI。

  在Ranger UI页面添加Hive Service:



  — 填写说明

上述截图中相关的固定填写内容如下:

名称	值
Service Name	emr-hive
jdbc.driverClassName	org.apache.hive.jdbc.HiveDriver

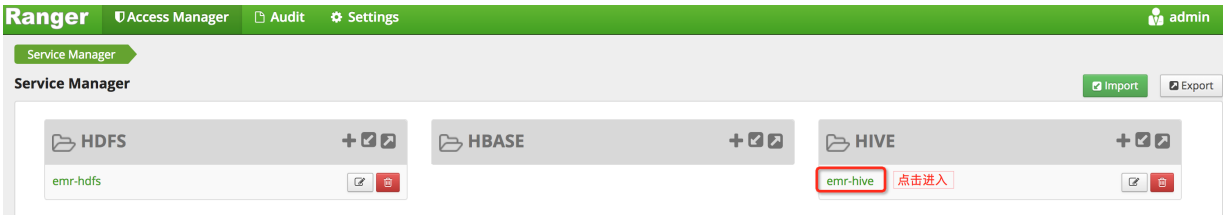
- 相关的变量值:

名称	值
jdbc.url	标准集群: jdbc:hive2://emr-header-1:10000/ 高安全集群:jdbc:hive2://${master1_fullhost}:10000/;principal=hive/${master1_fullhost}@EMR.$id.COM
policy.download.auth.users	hadoop(非高安全集群)/hive(高安全集群)

${master1_fullhost} 为master1的长域名，可登录master1执行`hostname`命令获取，${master1_fullhost}中的数字即为$id的值。

权限配置示例

上面一节中已经将Ranger集成到Hive，现在可以进行相关的权限设置。例如给用户foo授予表testdb.test的a列Select权限：



单击上图中的**emr-hive**进入配置页面，配置相关权限。

按照上述步骤设置添加一个Policy后，就实现了对foo的授权，然后用户foo就可以对testdb.test的表进行访问了。

> 说明：
> Policy被添加1分钟左右后，Hive才会生效。

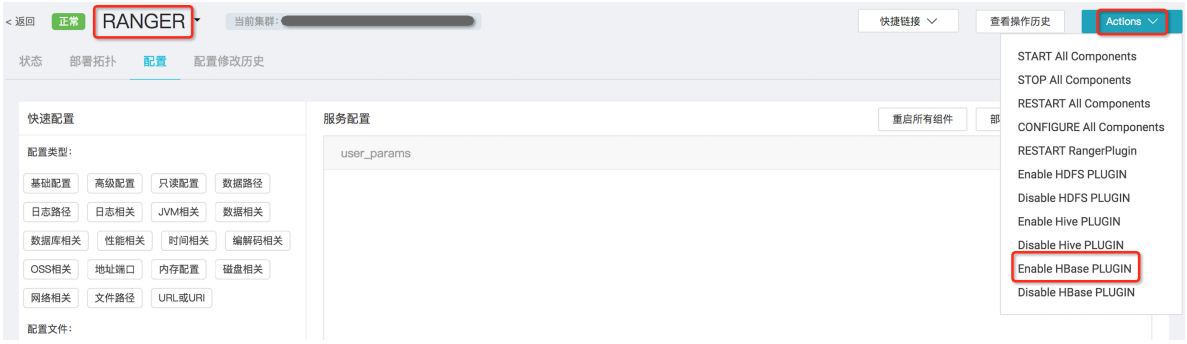
19.6.4 HBase配置

前面简介中介绍了E-MapReduce中创建启动Ranger服务的集群，以及一些准备工作，本节介绍HBase集成Ranger的一些步骤流程。

HBase集成Ranger

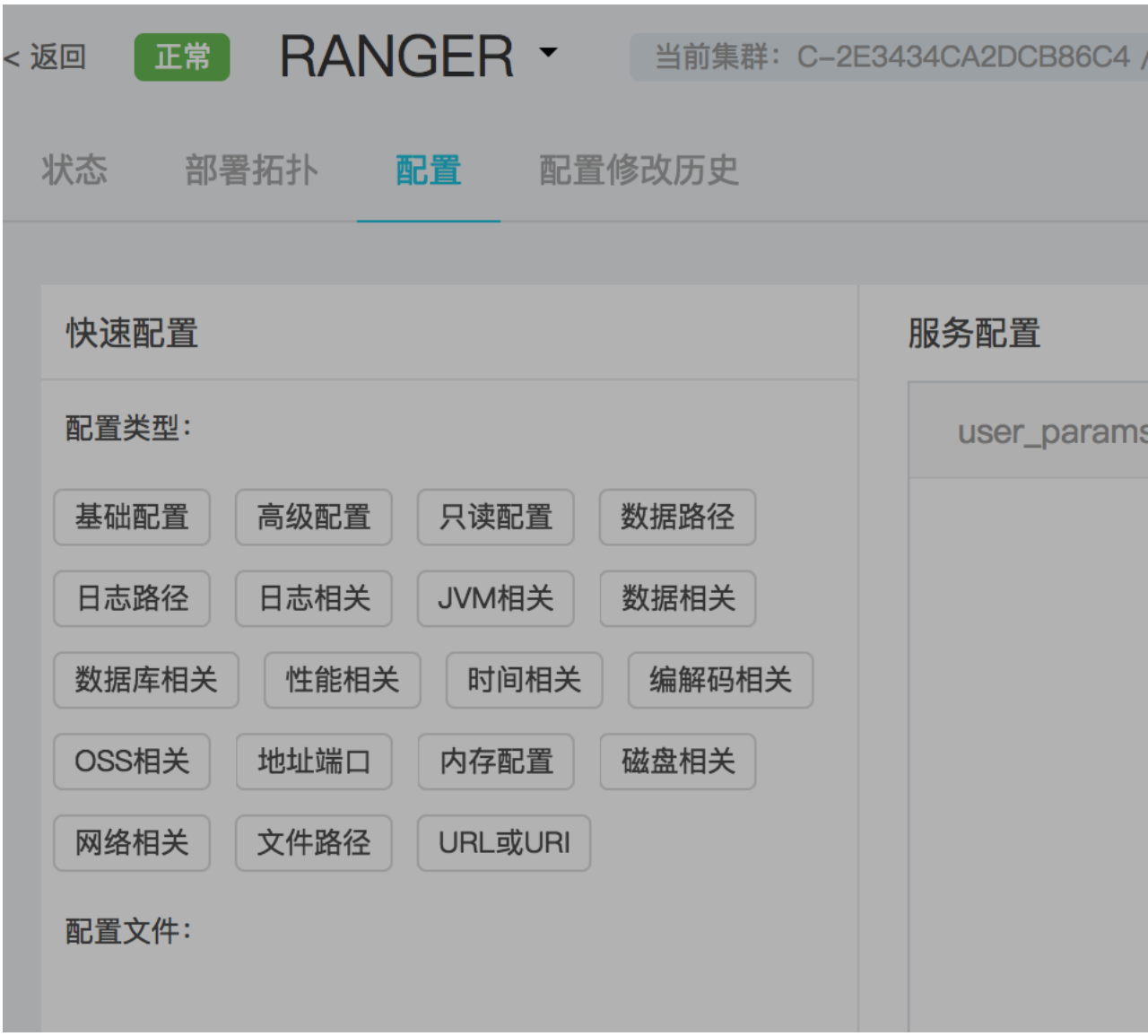
Enable HBase Plugin

1. 在集群管理页面，在你想操作的集群后的操作栏中单击管理
2. 在服务列表中单击**Ranger**进入Ranger配置页面
3. 在Ranger配置页面，单击右侧的操作下拉菜单，选择**Enable HBase PLUGIN**。



4. 在弹出框输入执行Commit记录，然后单击确定

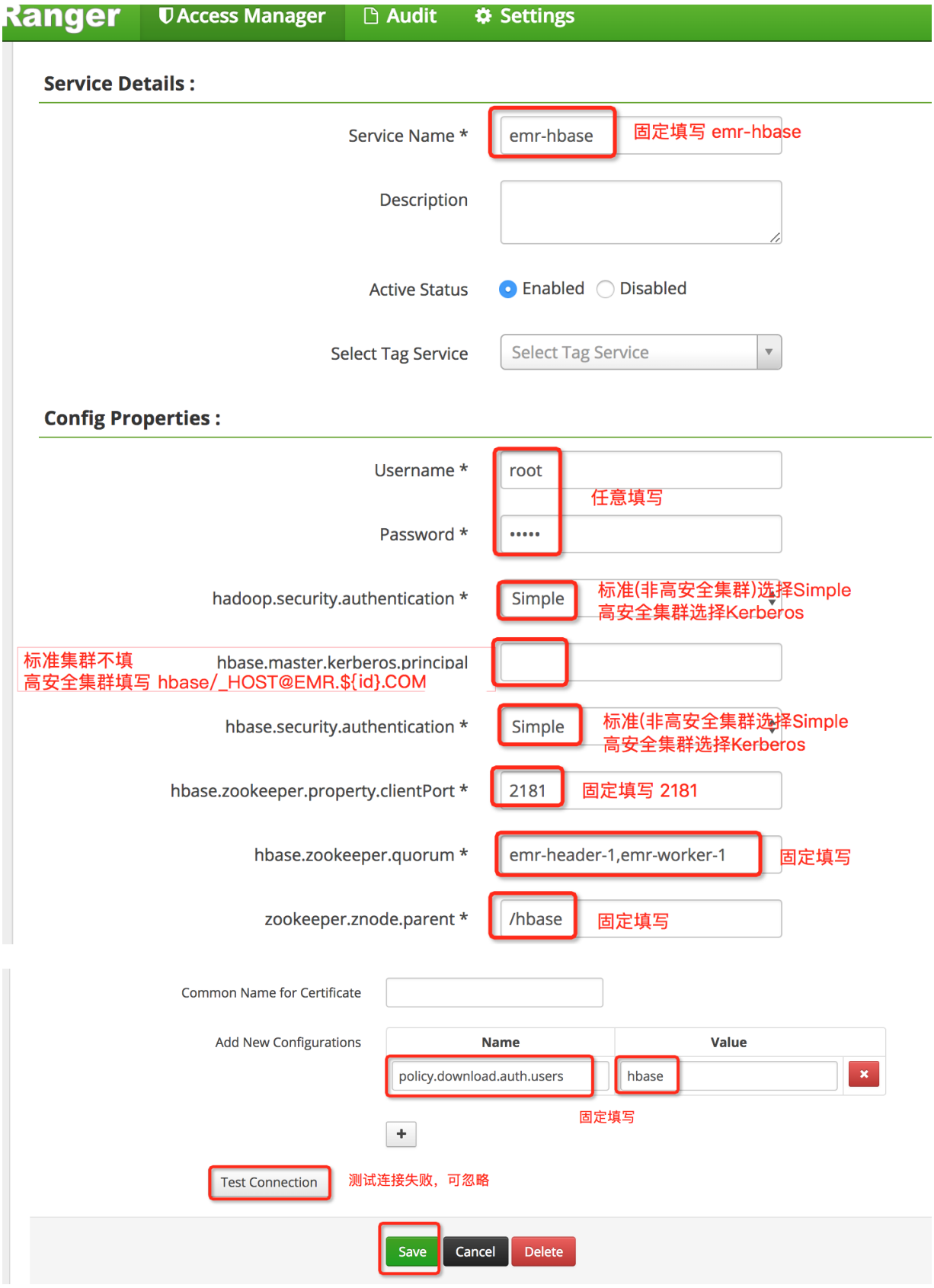
5. 单击右上角查看任务进度，等待任务完成。



Ranger UI添加HBase service

参见*Ranger*简介的介绍进入Ranger UI。

在Ranger的UI页面添加Hase Service：



> 说明：
> ${id}： 可登录机器执行host命令，hostname中的数字即为${id}的值。

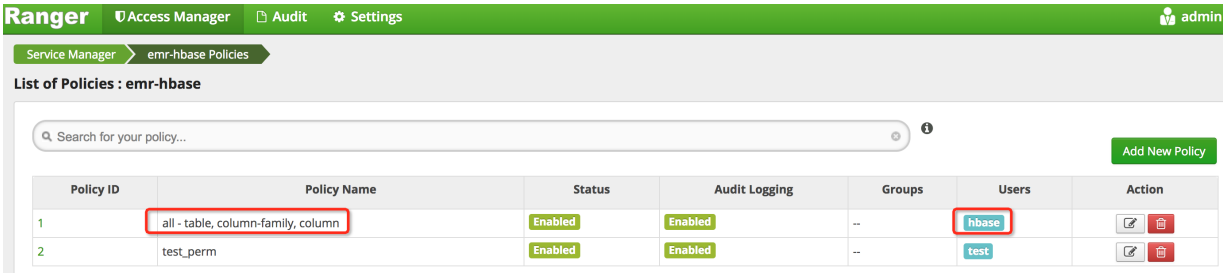
重启HBase

上述任务完成后，需要重启HBase才生效。重启HBase，请按照以下步骤操作：

1. 在Ranger配置页面中，单击左上角Ranger后的倒三角，从Ranger切换到HBase。
2. 单击右上角操作下拉菜单，选择**RESTART All Components**。
3. 在弹出框输入执行Commit记录，然后单击确定。
4. 单击右上角查看操作历史查看任务进度，等待重启任务完成。

设置管理员账号

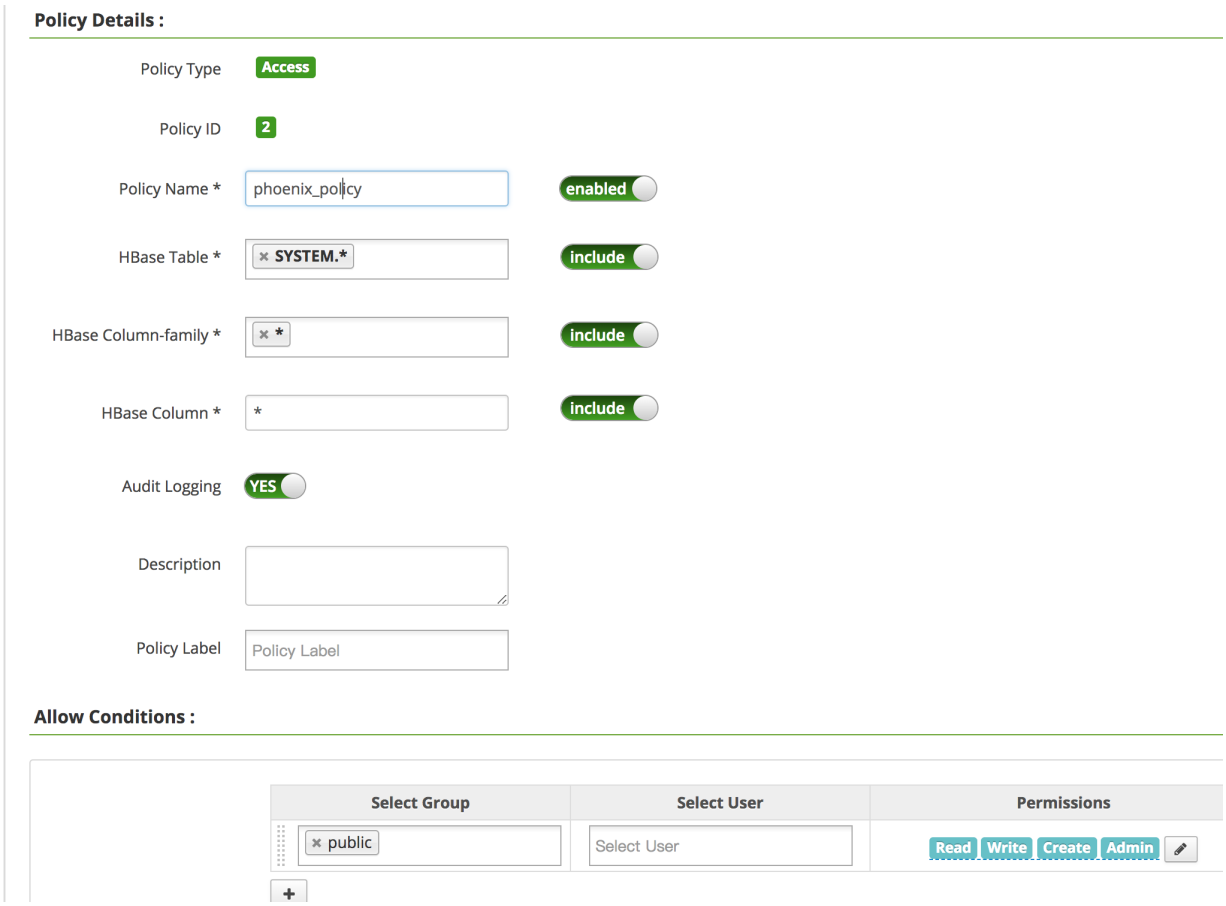
用户需要设置管理员账号的权限(admin权限)，用于执行一些管理命令，如`balance/compaction/flush/split`等等。



上图中已经存在权限策略，您只需要单击右侧的编辑按钮，将user改成自己想要设置的账号即可，另外也可以修改其中的权限(如只保留Admin权限)。hbase账号必须要默认设置为管理员账号。

若使用Phoenix，则还需在ranger的HBase中新增如下policy：

Table	SYSTEM.*
Column Family	*
Column	*
Groups	public
Permissions	Read Write, Create, Admin



权限配置示例

上面一节中已经将Ranger集成到HBase，可以进行相关的权限设置。例如给用户test授予表foo_ns:test的`Create/Write/Read`权限。

单击上图中的**emr-hbase**进入配置页面，配置相关权限。

User/Group 会自动从集群中同步过来，大约需要一分钟，用户可以事先将它们添加到集群。

按照上述步骤设置添加一个Policy后，就实现了对test用户的授权，然后test用户就可以对foo_ns:test表进行访问了。

> 说明：
> Policy被添加1分钟左右后，HBase才会生效。

19.6.5 Kafka配置

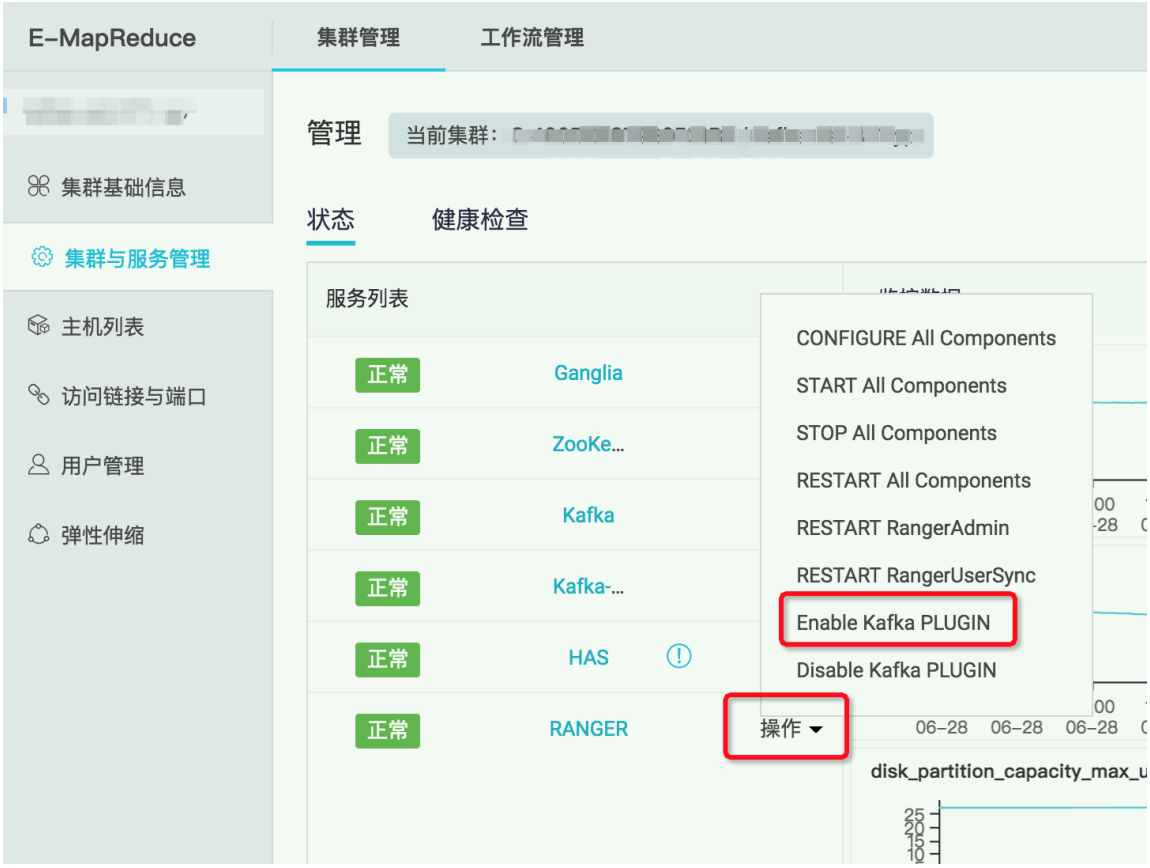
从EMR-3.12.0版本开始，E-MapReduce Kafka支持用Ranger进行权限配置。

Kafka集成Ranger

前面简介中介绍了E-MapReduce中创建启动Ranger服务的集群，以及一些准备工作，本节介绍Kafka集成Ranger的一些步骤流程。

- Enable Kakfa Plugin

1. 在集群管理页面，在需要操作的集群后单击管理
2. 在服务列表中单击**Ranger**进入Ranger配置页面
3. 在集群配置管理页面的Ranger服务下，单击右侧的操作下拉菜单，选择**Enable Kafka PLUGIN**。



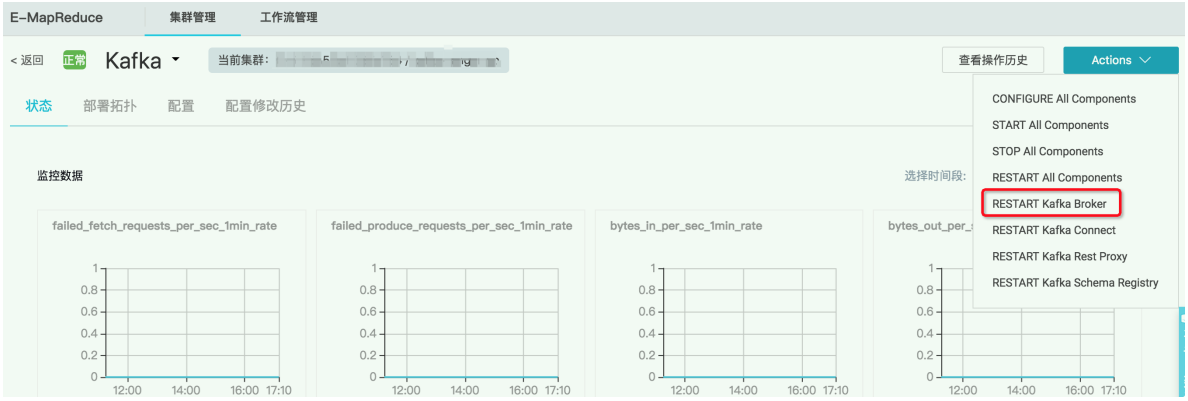
4. 在弹出框输入执行Commit记录，然后单击**确定**。

   单击右上角的**查看操作历史**查看任务进度，等待任务完成100%。

- 重启Kafka broker

  上述任务完成后，需要重启broker才能生效。

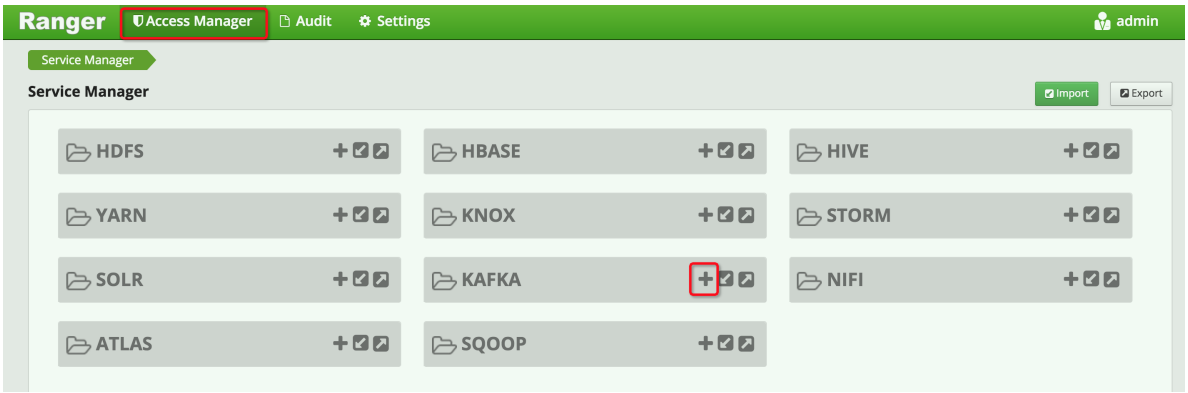
  1. 在Ranger配置页面中，单击左上角Ranger后的倒三角，从Ranger切换到Kafka。
  2. 单击右上角**操作**下拉菜单，选择**RESTART Broker**。
  3. 在弹出框输入执行Commit记录，然后单击**确定**。
  4. 单击右上角**查看操作历史**查看任务进度，等待重启任务完成。



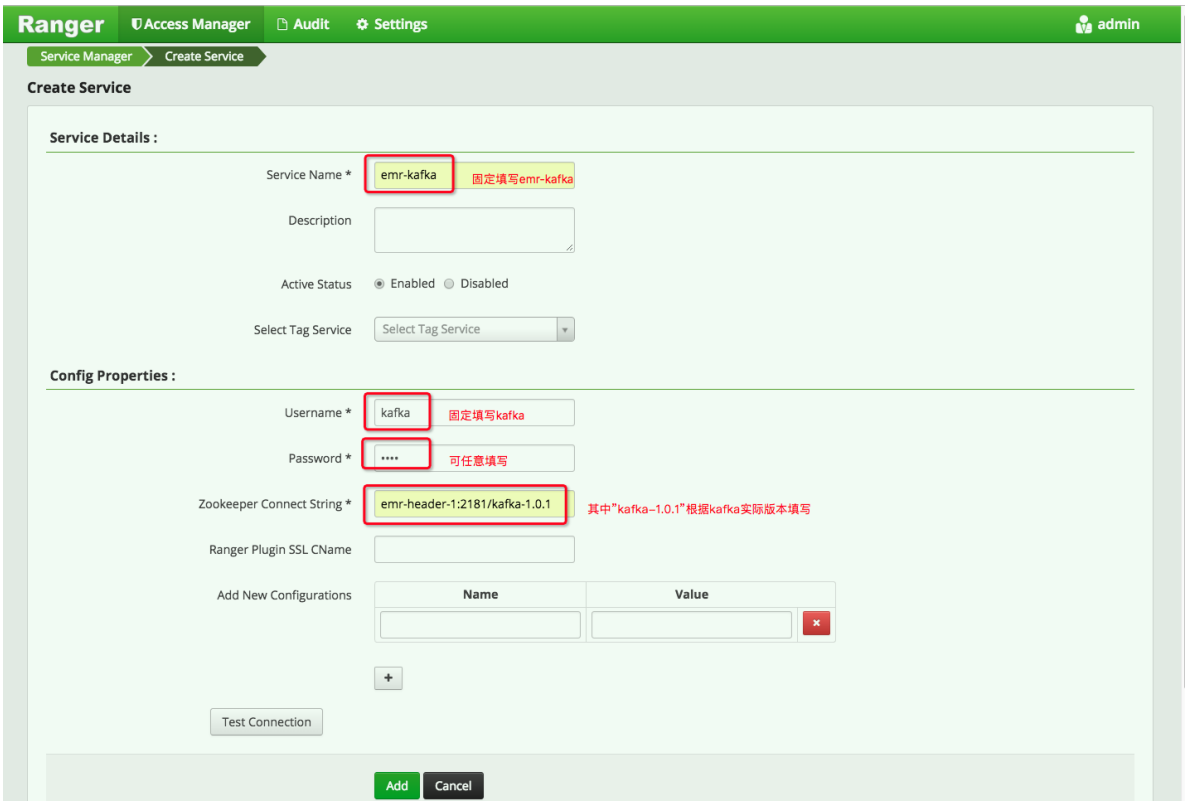
- Ranger UI页面添加Kafka Service

  参见*Ranger*简介的介绍进入Ranger UI页面。

  在Ranger的UI页面添加Kafka Service:



  配置Kafka Service:

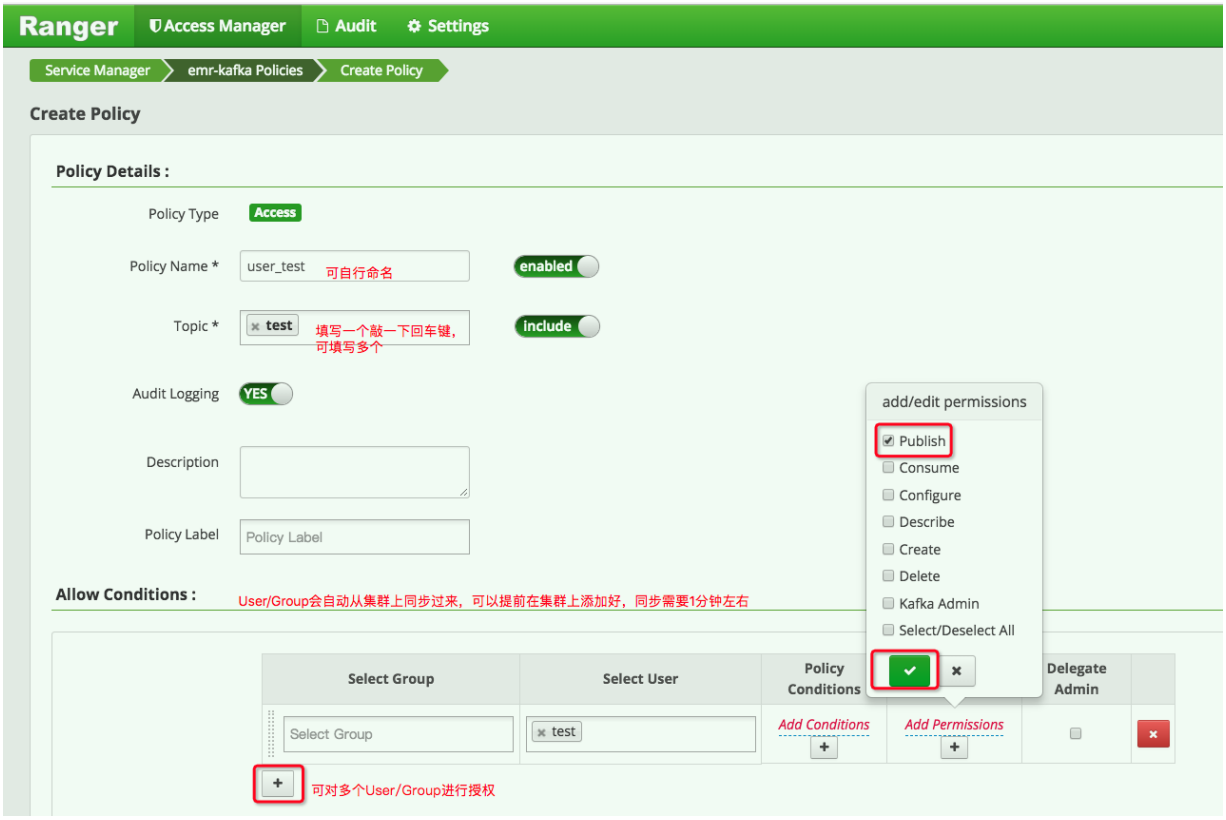


权限配置示例

上面一节中已经将Ranger集成到Kafka，现在可以进行相关的权限设置。

> 说明：
> 标准集群中，在添加了Kafka Service后，ranger会默认生成规则all - topic，不作任何权限限制（即允许所有用户进行所有操作），此时ranger无法通过用户进行权限识别。

以test用户为例，添加Publish权限：



按照上述步骤设置添加一个Policy后，就实现了对test的授权，然后用户test就可以对test的topic进行写入操作。

 说明：

Policy添加后需要1分钟左右才会生效。

20 集群容灾能力说明

20.1 EMR集群容灾能力

EMR集群容灾能力介绍

数据容灾

Hadoop分布式文件系统(HDFS)将每一个文件的数据进行分块存储，同时每一个数据块又保存有多个副本(系统默认为每一个数据块存放3个副本),尽量保证这些数据块副本分布在不同的机架之上（在大多数情况下，副本系数是3，HDFS的存放策略是将一个副本存放在本地机架节点上，一个副本存放在同一个机架的另一个节点上，最后一个副本放在不同机架的节点上）。

HDFS会定期扫描数据副本，若发现数据副本发生丢失，则会快速的进行数据的复制以保证副本的数量。若发现节点丢失，则节点上的所有数据也会快速的进行复制恢复。在阿里云上，如果是使用云盘的技术，则在后台每一个云盘都会对应三个数据副本，当其中的任何一个出现问题时，副本数据都会自动进行切换并恢复，以保证数据的可靠性。

Hadoop HDFS是一个经历了长时间考验且具有高可靠性的数据存储系统，已经能够实现海量数据的高可靠性存储。同时基于云上的特性，也可以在OSS等服务上进行数据的额外备份，来达到更高的数据可靠性。

服务容灾

Hadoop的核心组件都会进行HA的部署，即有至少2个节点的服务互备，如YARN，HDFS，Hive Server，Hive Meta，以保证在任何时候，其中任何一个服务节点挂掉时，当前的服务节点都能自动的进行切换，保证服务不会受到影响。

21 资源池使用说明

资源池（Dynamic Resource Pools）是一种对YARN应用的使用策略。EMR YARN默认采用Capacity Scheduler。

开启资源池

1. 登录阿里云 *E-MapReduce* 控制台，进入集群列表页面。
2. 在需要配置的集群后单击管理。
3. 在服务列表中单击**YARN**，进入YARN配置页面。
4. 在页面上方选择**资源池**页签，进入资源池配置页面。
5. 单击**初始化资源池**，选择调度策略。支持容量调度（*Capacity Scheduler*）和公平调度（*Fair Scheduler*）。

> **说明：**
> 开启资源池功能后，容量调度capacity_Scheduler或公平调度Fair_Scheduler一旦选择完成，后续将不可更改，用户只能在资源池中配置相关参数。

配置资源池

调度策略选定后，单击**更多设置**，在下拉菜单中选择后进行配置。下拉菜单中的选项为全局配置，建议由系统管理员统一配置。

- Capacity Scheduler

表 21-1: 更多配置

EMR参数项	Hadoop YARN参数项
默认配置-最大应用数	配置全局的参数yarn.scheduler.capacity.maximum-applications / yarn.scheduler.capacity.<queue-path>.maximum-applications
默认配置-最大am比	配置全局的参数yarn.scheduler.capacity.maximum-am-resource-percent / yarn.scheduler.capacity.<queue-path>.maximum-am-resource-percent
默认配置-资源计算类	yarn.scheduler.capacity.resource-calculator
默认配置-节点延迟等待次数	yarn.scheduler.capacity.node-locality-delay
放置规则	配置用户/组和队列的对应关系

EMR参数项	Hadoop YARN参数项
放置规则-队列映射覆盖	yarn.scheduler.capacity.queue-mappings-override.enable
ACL设置-启用ResourceManager ACL	yarn-site中的yarn.acl.enable
ACL设置-管理 ACL	yarn-site中的yarn.admin.acl
用户限制-acl_submit_applications	配置全局的参数yarn.scheduler.capacity.root.<queue-path>.acl_submit_applications
用户限制-acl_administer_queue	配置全局的参数yarn.scheduler.capacity.root.<queue-path>.acl_administer_queue

表 **21-2:** 创建资源池

EMR参数项	Hadoop YARN参数项
资源池名称	YARN中队列名称
资源池限制-比重	yarn.scheduler.capacity.<queue-path>.capacity
资源池限制-miniUserLimit	yarn.scheduler.capacity.<queue-path>.minimum-user-limit-percent
资源池限制-maximumCapacity	yarn.scheduler.capacity.<queue-path>.maximum-capacity
资源池限制-单个用户限制比例	yarn.scheduler.capacity.<queue-path>.user-limit-factor
资源池限制-最大内存	yarn.scheduler.capacity.<queue-path>.maximum-allocation-mb
资源池限制-最大核数	yarn.scheduler.capacity.<queue-path>.maximum-allocation-vcores
资源池限制-最大应用数	优先级高于全局变量yarn.scheduler.capacity.maximum-applications / yarn.scheduler.capacity.<queue-path>.maximum-applications
资源池限制-AM最大资源占比	优先级高于全局变量yarn.scheduler.capacity.maximum-am-resource-percent / yarn.scheduler.capacity.<queue-path>.maximum-am-resource-percent
提交访问控制	优先级高于全局变量yarn.scheduler.capacity.root.<queue-path>.acl_submit_applications
管理访问控制	优先级高于全局变量yarn.scheduler.capacity.root.<queue-path>.acl_administer_queue

- Fair Scheduler

  表 **21-3:** 更多配置

EMR参数项	**Hadoop YARN**参数项
ACL设置-启用ResourceManager ACL	yarn-site中的yarn.acl.enable
ACL设置-管理 ACL	yarn-site中的yarn.admin.acl

  表 **21-4:** 创建资源池

EMR参数项	**Hadoop YARN**参数项
开启抢占模式	yarn.scheduler.fair.preemption
资源池名称	YARN中队列名称
抢占-开启抢占模式	yarn.scheduler.fair.preemption
抢占-公平份额抢占阈值	yarn.scheduler.fair.preemption.cluster-utilization-threshold

  资源池配置完成后，单击页面右上角的**同步配置到集群**，使配置生效。

关闭资源池

如果您想通过XML直接配置，请先在资源池页签下单击**关闭资源池**按钮，然后到**配置**页签下进行配置。

22 弹性伸缩

22.1 弹性伸缩概述

本文将介绍如何开启和关闭弹性伸缩功能。

在以下场景中，您可以通过E-MapReduce弹性伸缩功能节省成本，提高执行效率。

- 临时需要按照时间段添加计算节点，补充计算能力。
- 确保重要作业按时完成，按照某些集群指标扩充计算节点。

说明：

- 弹性伸缩功能仅能对Task节点进行扩容或缩容。
- 弹性伸缩仅支持Hadoop包年包月、按量付费集群。

开启弹性伸缩

1. 登录阿里云 *E-MapReduce* 控制台，单击**前往集群列表**进入集群列表页面。
2. 单击集群ID右侧的**管理**。
3. 在左侧导航栏中单击**弹性伸缩**。
4. 单击页面右上角的**开启弹性伸缩**按钮。

   如果当前账号是首次使用弹性伸缩功能，需对E-MapReduce授予弹性伸缩（ESS）的默认角色。
5. 在ESS的授权界面中单击**确定**即可。

关闭弹性伸缩

单击**关闭弹性伸缩**，当前已经通过弹性伸缩功能扩充的Task节点将会被全部释放，HDFS存储的数据位于Core节点，不会受影响。

22.2 按时间伸缩规则配置

如果Hadoop集群计算量在一定的周期内存在明显的波峰、波谷，您可以设置在每天、每周或每月的固定时间段扩出一定量的Task节点来补充计算能力，这样在保证作业完成的同时，可以节省您的成本。

由于弹性伸缩节点均为按量付费的购买方式，并且按量付费和包年包月的同等计算能力价格大概在3：1左右，所以需要根据您的弹性伸缩时间来设计包年包月计算能力和按量付费计算能力的比例。

例如业务波峰时间段每天持续8小时，包年包月和按量付费的价格大致相同，当大于8小时时，包年包月比弹性伸缩的购买方式更优惠。

配置伸缩实例数

- 最大节点数：弹性伸缩的Task节点上限。一旦达到上限，即使满足弹性伸缩的规则，也不会继续进行弹性伸缩的动作。目前可设置的弹性伸缩最大上限为1000。
- 最小节点数：弹性伸缩的Task节点下限。如果弹性伸缩规则中设置的增加或减少Task节点数小于此处的最小节点数，那么在首次执行时，集群会以最小节点数为准进行伸缩。

  例如，设置弹性扩容规则为每天零点动态添加1个节点，但最小节点数为3。那么系统在第一天的零点时会添加3个节点，以满足最小节点数的要求。

配置伸缩规则

伸缩规则分为扩容规则和缩容规则。集群关闭弹性伸缩功能后，所有规则会被清空，再次开启弹性伸缩功能时，需要重新配置伸缩规则。

- 规则名称：在同一个集群中，伸缩规则名称（包括扩容规则和缩容规则）不允许重复。

- 规则执行周期：
  - 只执行一次：集群在指定的时间点执行一次弹性伸缩动作。
  - 重复执行：用户可以选择每天、每周或每月的某一特定时间点执行一次弹性伸缩动作。
- 重试过期时间：弹性伸缩在到达指定时间时可能由于各种原因不能执行，通过设置重试过期时间，系统会在该时间范围内每隔30秒一直检测可以执行伸缩的时机，直到在满足条件时执行伸缩。设置范围为0到21600秒。

  假设在指定时间段需要进行弹性伸缩动作A，如果有其他弹性伸缩动作B或正处在冷却期，则动作A无法执行。在您设置的重试过期时间内，每隔30秒会重试一次，尝试执行A，一旦条件满足，集群会立刻执行弹性伸缩。
- 增加或减少Task节点数：规则被触发时，集群每次执行增加或减少的Task节点数量。
- 冷却时间：每次弹性伸缩动作执行完成，到可以再次进行弹性伸缩的时间间隔。在冷却时间内，不会发生弹性伸缩动作。

配置伸缩规格

弹性伸缩配置可以指定伸缩的节点的硬件规格。用户只能在开启弹性伸缩功能时配置，保存后不能更改。如出于特殊情况确实需要修改，可以关闭弹性伸缩功能后，再次开启。

- 选择vCPU和内存规格时，系统会根据您的选择自动匹配出满足条件的实例，显示在下面的备选实例列表中。您需要添加备选的实例到右侧列表中，以便集群按照已选的实例规格进行伸缩。
- 为避免由于ECS库存不足造成的弹性伸缩失败，您最多可以选择3种ECS实例。
- 无论是选择高效云盘还是SSD云盘，数据盘最小设置为40G。

22.3 弹性伸缩抢占式实例

EMR的抢占式实例适用于大数据作业执行成功与否没有强需求，但对计算资源价格非常敏感的场景。可以通过开通弹性伸缩功能，购买抢占式实例提升集群的计算资源。

开启弹性伸缩

开启弹性伸缩并设置伸缩规则可以参考以下步骤：

1. 登录阿里云 *E-MapReduce* 控制台。
2. 单击集群管理。
3. 单击需要添加抢占式实例集群ID右侧的管理。
4. 在左侧导航栏中单击弹性伸缩。

5. 如果集群没有开启过弹性伸缩，单击开启弹性伸缩。
6. 配置伸缩规则请参见按时间伸缩规则配置。
7. 在伸缩配置中，选择抢占式实例。

配置抢占式实例

> 说明：
> 抢占式实例价格相对于普通按量付费实例有较大优惠，但阿里云会根据 供需资源或市场成交价的变化可能随时释放您的抢占式实例。

配置抢占式实例参考以下步骤：

1. 选择实例的vCPU和内存大小
2. 选择实例类型，EMR会筛选出符合规格的所有实例类型，为保证您尽可能购买到抢占式实例，您可以选择3种实例规格。
3. 选中对应的实例规格后，设置单台实例规格的上限价格，单击确认。该实例类型会从可选实例到已选实例。如需改变出价，可再次在可选实例列表中选中实例规则，修改单台实例上限价格（按小时计算）。当您的出价高于当前市场价格时，您的实例就会运行。最终实例规格会按照市场价格计费。
4. 系统盘主要部署OS和EMR基础服务，无需您调整大小。可根据您的需要调整数据盘大小。
5. 实例出价上限+系统盘价格+数据盘价格为最终的配置价格。单击保存完成配置。

关于抢占式实例的更多信息，可以查看抢占式实例*FAQ*。

22.4 弹性伸缩记录

弹性伸缩执行完成后，您可以单击**弹性伸缩**页面上方的**弹性伸缩记录**页签，查看弹性伸缩活动的执行记录，以及弹性伸缩活动执行完成后的节点数量等信息。

弹性伸缩的执行状态包括以下4类：

- **执行中**：弹性伸缩活动正在执行。
- **成功**：根据伸缩规则，所有弹性伸缩中的所有节点被加入或移出集群。
- **部分成功**：根据伸缩规则，有部分节点成功被加入或移出集群，但是受磁盘配额管理或ECS库存的影响，部分节点执行失败。
- **全部失败**：根据伸缩规则，没有一个节点被加入或移出集群。

23 数据开发

23.1 项目管理

创建E-MapReduce集群后，用户可以创建工作流项目，使多个作业可以同时或者按照先后顺序运行，以便更好的管理作业的运行。

创建项目

1. 通过主账号登录阿里云 *E-MapReduce* 控制台。
2. 单击上方的**数据开发**页签，进入**项目列表**页面。

   主账号下可以查看该账号下的所有项目（包括所有子账号），子账号仅可以查看具有开发权限的项目。如需添加项目开发权限，需要通过主账号来配置，请参见用户管理。
3. 单击右上角的**新建项目**按钮，弹出**新建项目**对话框。
4. 输入项目名称和项目描述，单击**创建**。

   说明：

   只有主账号才能创建项目，即**新建项目**按钮只对主账号管理员可见。

用户管理

创建新的项目后，您可以为RAM子账号添加该项目的操作权限。

1. 在**项目列表**页面，单击项目右侧的**详情**。
2. 单击**用户管理**页签。
3. 单击**添加用户**，添加该主账号下的RAM子账号到该项目。

   被添加的子账号将成为该项目的成员，并能查看、开发该项目下的作业和工作流。如果不想将子账号继续设置为所选项目成员，单击用户右侧的**删除**即可。

   说明：

   只有主账号才能添加项目成员，即项目列表页面中的**用户管理**功能只对主账号管理员可见。

关联集群资源

创建新的项目后，您需要为项目关联集群，使得该项目中的工作流可以运行在关联的集群上。

1. 在**项目列表**页面，单击项目右侧的**详情**。

2. 单击集群设置页签。
3. 单击添加集群，从下拉菜单中可以选择已购买的包年包月和按量付费集群（执行临时作业创建的集群此处不会列示）。
4. 单击确定。

   单击集群右侧的删除，可以取消关联该集群资源。

   说明：

   只有主账号才能添加集群资源，即项目列表页面中的集群设置功能只对主账号管理员可见。

单击集群右侧的修改配置，可以设置提交作业到该集群的队列和用户。具体配置项说明如下：

- 提交作业默认用户：设置项目使用所选集群提交作业时的默认Hadoop用户，默认值是hadoop，默认用户只能有一个。
- 提交作业默认队列：设置项目使用所选集群提交作业时的默认队列，如果此处不填写，则作业会提交到default队列。
- 提交作业用户白名单：设置可以提交作业的Hadoop用户，如果有多个用户，可以通过英文半角逗号（,）分隔。
- 提交作业队列白名单：用于设置项目中的作业可以运行在所选集群的队列，若果有多个队列，可以通过英文半角逗号（,）分隔。
- 配置客户端白名单：配置可以提交作业的客户端，用户可以使用EMR的Master节点或EMR购买的Gateway，ECS自建Gateway暂不支持在此处配置。

23.2 作业编辑

在项目中，您可以创建 Shell、Hive、Spark、SparkSQL、MapReduce、Sqoop、Pig 、Spark Streaming等类型的作业。

新建作业

1. 通过主账号登录阿里云 *E-MapReduce* 控制台。
2. 单击上方的数据开发页签，进入项目列表页面。
3. 单击对应项目右侧的工作流设计，进入作业编辑页面。
4. 在页面左侧，在需要操作的文件夹上单击右键，选择新建作业。
5. 在新建作业对话框中，输入作业名称、作业描述，选择作业类型。

   创建作业时作业类型一经确定，不能修改。

6. 单击确定。

> 说明：
> 您还可以通过在文件夹上单击右键，进行创建子文件夹、重命名文件夹和删除文件夹操作。

开发作业

关于各类作业的具体开发，请参见《*EMR*用户指南》的作业部分。

> 说明：
> 插入OSS路径时，如果选择 OSSREF 文件前缀，系统会把OSS文件下载到集群本地，并添加到classpath中。

- 作业基础设置

  单击页面右上角的作业设置，弹出作业设置页面。

  - 失败重试次数：设置在工作流运行到该作业失败时，重试的次数。直接在作业编辑页面运行作业，该选项不会生效。
  - 失败策略：设置在工作流运行到该作业失败时，继续执行下一个节点还是暂停当前工作流。
  - 添加运行资源：如添加作业执行需依赖的Jar包或UDF等资源，需将资源先上传至OSS。在作页的运行资源中选中该资源后，可以直接在作业中引用该资源。
  - 配置参数：指定作业代码中所引用变量的值。用户可以在代码中引用变量，格式为：*${*变量名*}*。单击右侧的加号图标添加key和value，key为变量名，value为变量的值。另外，您还可以根据调度启动时间自定义时间变量， 规则如下：
    - yyyy 表示4位的年份。
    - MM 表示月份。
    - dd 表示天。
    - hh24 表示24小时制，12小时制使用 hh。
    - mm 表示分钟。
    - ss 表示秒。

      时间变量可以是包含 yyyy 年份的任意时间组合，同时支持用+和-方式来分别表示提前和延后。例如，变量 *${yyyy-MM-dd}*表示当前日期，则：

      - 后1年的表示方式： *${yyyy+Ny}* 或者 *${yyyy-MM-dd hh:mm:ss+1y}*。
      - 后3月的表示方式： *${yyyyMM+Nm}*或者 *${hh:mm:ss yyyy-MM-dd+3m}*。

■ 前5天的表示方式： *${yyyyMMdd-Nd}*或者 *${hh:mm:ss yyyy-MM-dd-5d}*。

- 作业高级设置

  在**作业设置**页面，单击**高级设置**页签。

  — 模式：包括YARN和LOCAL两种模式。YARN模式下，作业通过Launcher在YARN上分配资源进行提交。LOCAL模式下，作业在分配的机器上直接运行。

  — 环境变量：添加作业执行的环境变量，也可以在作业脚本中直接export环境变量。

  — 调度参数：设置作业运行yarn队列、CPU、内存和Hadoop用户等信息，可以不设置，作业会直接采用Hadoop集群的默认值。

作业执行

作业开发和配置完成后，您可以单击右上角的**运行**按钮执行作业。

查看日志

作业运行后，您可以在页面下方的**运行记录**页签中查看作业的运行日志。单击**详情**跳转到运行记录中该作业的详细日志页面，可以看到作业的提交日志、YARN Container日志。

23.3 临时查询

临时查询是adhoc 即席查询的场景，只支持HiveSQL SparkSQL和Shell三种类型，运行临时查询的语句，在页面下方显示日志和查询结果。

新建作业

作业编辑页中运行作业，单击对应作业**详情**会跳转到详情页面显示提交日志和运行日志。作业与两者的区别主要是运行场景不同，临时查询针对数据科学家和数据分析师，主要用SQL为工具。

1. 通过主账号登录阿里云 *E-MapReduce* 控制台。
2. 单击上方的**数据开发**页签，进入**项目列表**页面。
3. 单击对应项目右侧的**工作流设计**，进入**作业编辑**页面。
4. 单击页面左侧的**临时查询**页签，进入**临时查询**页面。
5. 在页面左侧，在需要操作的文件夹上单击右键，选择**新建作业**。
6. 在**新建作业**对话框中，输入作业名称、作业描述，选择作业类型。

   创建作业时作业类型一经确定，不能修改。
7. 单击**确定**。

> 说明：
> 您还可以通过在文件夹上单击右键，进行创建子文件夹、重命名文件夹和删除文件夹操作。

开发作业

关于HiveSQL SparkSQL和shell作业的具体开发，请参见EMR用户指南《作业》部分。

> 说明：
> 插入OSS路径时，如果选择 OSSREF 文件前缀，系统会把OSS文件下载到集群本地，并添加到classpath中。

- 作业基础设置

  单击页面右上角的作业设置，弹出作业设置页面。

  - 添加运行资源：如添加作业执行需依赖的Jar包或UDF等资源，需将资源先上传至OSS。在作页的运行资源中选中该资源后，可以直接在作业中引用该资源。
  - 配置参数：指定作业代码中所引用变量的值。用户可以在代码中引用变量，格式为：*${*变量名*}*。单击右侧的加号图标添加key和value，key为变量名，value为变量的值。另外，您还可以根据调度启动时间自定义时间变量， 规则如下：
    - yyyy 表示4位的年份。
    - MM 表示月份。
    - dd 表示天。
    - hh24 表示24小时制，12小时制使用 hh。
    - mm 表示分钟。
    - ss 表示秒。

      时间变量可以是包含 yyyy 年份的任意时间组合，同时支持用+和-方式来分别表示提前和延后。例如，变量 *${yyyy-MM-dd}*表示当前日期，则：

      - 后1年的表示方式： *${yyyy+Ny}* 或者 *${yyyy-MM-dd hh:mm:ss+1y}*。
      - 后3月的表示方式： *${yyyyMM+Nm}*或者 *${hh:mm:ss yyyy-MM-dd+3m}*。
      - 前5天的表示方式： *${yyyyMMdd-Nd}*或者 *${hh:mm:ss yyyy-MM-dd-5d}*。

- 作业高级设置

  在作业设置页面，单击高级设置页签。

- 模式：包括YARN和LOCAL两种模式。YARN模式下，作业通过Launcher在YARN上分配资源进行提交。LOCAL模式下，作业在分配的机器上直接运行。
- 调度参数：设置作业运行yarn队列、CPU、内存和Hadoop用户等信息，可以不设置，作业会直接采用Hadoop集群的默认值。

作业执行

作业开发和配置完成后，您可以单击右上角的运行按钮执行作业。

查看日志

作业运行后，您可以在页面下方的日志页签中查看作业的运行日志。

23.4 工作流编辑

E-MapReduce工作流支持通过DAG的方式并行执行大数据作业，用户可以暂停、停止、重新运行工作流，还可以在Web UI查看工作流的执行状态。

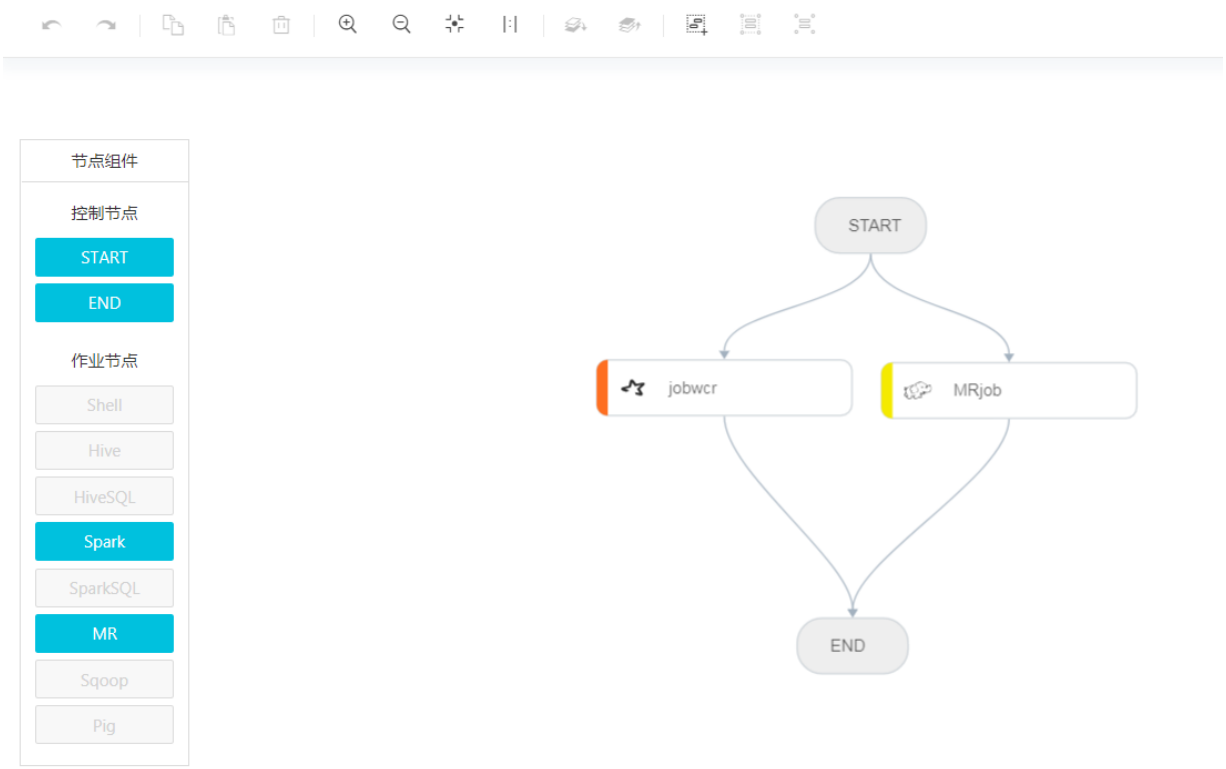
新建工作流

1. 通过主账号登录阿里云 *E-MapReduce* 控制台。
2. 单击上方的数据开发页签，进入项目列表页面。
3. 单击对应项目右侧的工作流设计，然后单击左侧的工作流设计页签，进入工作流设计页面。
4. 在页面左侧，在需要操作的文件夹上单击右键，选择新建工作流。
5. 在新建工作流对话框中，输入工作流名称、工作流描述，选择执行集群。

   用户可以选择已经创建的且被关联到该项目的预付费和后付费EMR集群用于执行工作流，也可以通过集群模板的方式新建一个临时集群用于执行该工作流。
6. 单击确定。

编辑工作流

用户可以通过拖拽方式将不同类型的作业拉到工作流编辑画布，将不同作业节点通过连线的方式指定工作流的流转。作业拖拽完成后，从控制节点处拖拽**END**组件到画布中，表示整个工作流设计完成。



配置工作流

在工作流设计页面的右侧，单击**配置**按钮，可以进行工作流调度配置。

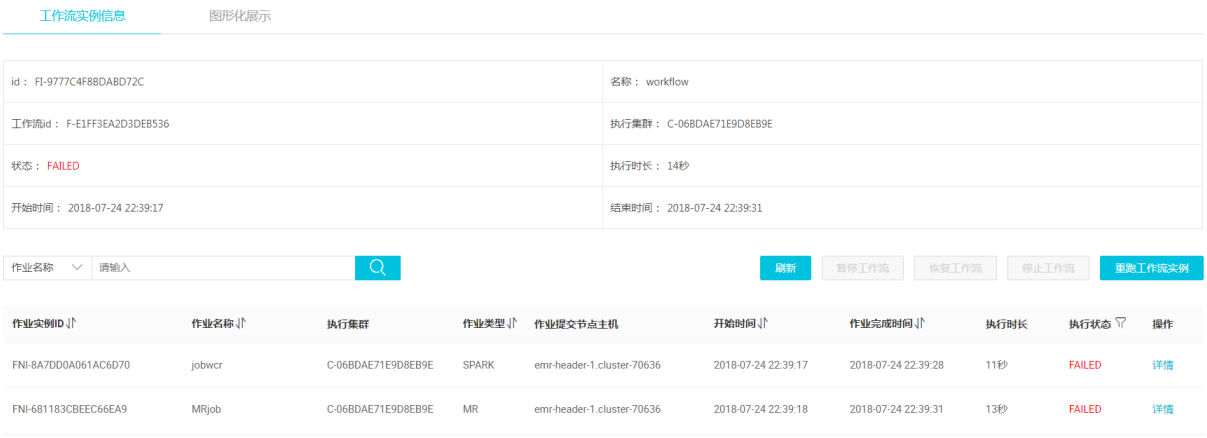
- 执行集群：用户可以修改工作流的执行集群。
- 调度策略：在开启工作流调度后，您可以选择一种调度策略，时间调度或依赖调度。
  - 时间调度：设置工作流调度的开始时间和结束时间，在此时间范围内，系统会根据您设置的周期执行工作流。
  - 依赖调度：从所选项目中，选择当前工作流的前续工作流。当前续工作流执行完成后，当前工作流才会被调度执行。目前依赖调度只能选择一个工作流。

执行工作流

工作流设计和配置完成后，您可以单击右上角的**运行**按钮执行工作流。

查看并操作工作流实例

工作流运行后，单击左侧的**运行记录**页签，可以查看工作流实例的运行状态。单击工作流实例对应的**详情**，可以查看作业实例的运行情况，也可以暂停、恢复、停止和重跑工作流实例。



- 暂停工作流后：正在运行的作业节点会继续执行，但后续的作业节点不再执行，可以单击恢复工作流，系统将继续执行暂停作业节点之后的作业。
- 取消工作流：所有正在运行的作业节点立即停止。
- 重跑工作流实例：系统将从工作流的start节点从头开始执行工作流。