Table of Contents

KAP 2.0 Admin Manual

Kyligence Analytics Platform（简称KAP）是领先的大数据分析平台。

本手册可以在Gitbook 浏览或下载。

上海跬智信息技术有限公司

Kyligence大数据分析平台

Kyligence大数据分析平台(Kyligence Analytics Platform, KAP)是基于Apache Kylin实现的、为可伸缩数据集提供分析能力的企业级大数据产品，在Apache Hadoop上为百亿及以上超大规模数据集提供亚秒级标准SQL查询能力，为业务用户、分析师及工程师提供简单、快捷的大数据分析能力。Apache Kylin在业界已经被多个公司采用已解决各种大数据分析挑战。

相较于其它技术，KAP具有以下特点：

- 亚秒级查询

在百亿及以上规模数据集上为业务用户及分析师提供亚秒级的查询速度，并同时支持高并发，使得在大数据平台上对超大规模数据进行交互式分析成为可能；支持ANSI SQL查询标准，使得业务用户及分析师无需重新学习新的技术即可掌握在海量超大规模数据集上快速分析的能力。

- 无缝集成

支持与企业级商业智能(BI)及可视化工具无缝集成，提供标准的ODBC、JDBC驱动及REST API接口等以连接流行的数据分析、展示工具，如Tableau、Microsoft PowerBI、Microsoft Excel、Apache Zeppelin、Saiku等

- 自助服务

Kyligence大数据分析平台使得分析师及用户能以简洁而快速的方式分析海量数据。易于使用的Web界面允许用户自己构建数据集市而无需知晓底层技术 。

- 可扩展架构

全新设计的可扩展架构从根本上解藕了对特定技术的依赖，将计算框架，数据源以及底层存储等扩展到更多的技术领域，为不同的技术栈提供可配置的优化解决方案。

- 非侵入式

Kyligence大数据分析平台的部署不需要在现有Hadoop集群上安装任何新的组件，更不需要在数据节点或其他节点上安装Agent等，所有与集群的操作都通过标准API完成，从而使得对现有集群的影响最小化，也为快速部署带来了可能。

更多信息，请访问Kyligence网站: http://kyligence.io/.

本节介绍了在KAP中使用的基础概念。对于某些领域术语，请参考章节“术语”。

CUBE

- **Table** - 表， 是Cube的数据源；在创建Cube之前，KAP需要从数据源（通常为Hive）同步表的元数据，包含表名、列名、列属性等。
- **Data Model** - 数据模型，定义了由若干张表的一个连接关系。KAP支持星型模型的多维分析；在创建Cube之前，用户需定义这么一个数据模型。
- **Cube** - 数据立方体，是一种多维分析的技术，通过预计算，将计算结果存储在某多个维度值所映射的空间中；在运行时通过对Cube的再处理而快速获取结果。
- **Partition** - 分区，用户可以定义一个分区日期或时间列，随后对Cube的构建按此列的值范围而进行，从而将Cube分成多个Segment。
- **Cube Segment** - 每个Cube Segment是对特定时间范围的数据计算而成的Cube。每个Segment对应一张HBase表。
- **Aggregation Group** - 聚合组，每个聚合组是全部维度的一个子集；通过将很多个维度分组，并把常一起使用的维度放在一起，可以有效降低Cube的组合数。

维度 & 度量

- **Mandotary** - 必需的维度：这种类型用于对Cube生成树做剪枝：如果一个维度被标记为“Mandatory”，会认为所有的查询都会包含此维度，故所有不含此维度的组合，在Cube构建时都会被剪枝（不计算）.
- **Hierarchy** - 层级维度：如果多个维度之间有层级（或包含）的关系，通过设置为“Hierarchy”，那些不满足层级的组合会被剪枝。如果A, B, C是层级，并且A>B>C，那么只需要计算组合A, AB, ABC; 其它组合如B, C, BC, AC将不做预计算。
- **Derived** - 衍生维度：维度表的列值，可以从它的主键值衍生而来，那么通过将这些列定义为衍生维度，可以仅将主键加入到Cube的预计算来，而在运行时通过使用维度表的快照，衍生出非PK列的值，从而起到降维的效果。
- **Count Distinct(HyperLogLog)** - 基于HyperLogLog的Count Distint：快速、精确的COUNT DISTINCT是较难计算的, 一个近似的轻量级算法 - HyperLogLog 为此而发明, 能够在大规模数据集上做去重并保持较低的误差率.
- **Count Distinct(Bitmap)** - 基于Bitmap的COUNT DISTINCT，可以精确去重，但是存储开销较大。目前只支持int的数据类型.
- **Top N** - 预计算最top的某些记录的度量，如交易量最大的1000个卖家。

CUBE 操作

- **BUILD** - 构建：给定一个时间范围，将源数据通过运算而生成一个新的Cube Segment。
- **REFRESH** - 刷新：对某个已经构建过的Cube Segment，重新从数据源抽取数据并构建，从而获得更新。
- **MERGE** - 合并：将多个Cube Segment合并为一个Segment。这个操作可以减少Segment的数量，同时减少Cube的存储空间。
- **PURGE** - 清空：将Cube的所有Cube Segment删除。

JOB 状态

- **NEW** - 新任务，刚刚创建。
- **PENDING** - 等待被调度执行的任务.
- **RUNNING** - 正在运行的任务。
- **FINISHED** - 正常完成的任务（终态）。
- **ERROR** - 执行出错的任务。
- **DISCARDED** - 丢弃的任务（终态）。

JOB 操作

- **RESUME** - 恢复：处于ERROR状态的任务，用户在排查或解决问题后，通过此操作来重试执行。
- **DISCARD** - 丢弃：放弃此任务，立即停止执行并不会再恢复。

KAP依赖的环境

KAP需要一个状态良好的Hadoop集群做为运行环境。为获得更好的稳定性，建议将KAP单独运行在一个或多个Hadoop客户机上。该客户机上已经配置了各个组件的客户端，如 `hive`，`hbase`，`hadoop`，`hdfs` 。

运行KAP的Linux账户，已经具备访问Hadoop集群的权限，包括读写HDFS, 创建和读取Hive表，递交MapReduce任务，创建和操作HBase表等。

认证的Hadoop企业版

- Cloudera CDH 5.7

兼容的Hadoop版本

- Hadoop: 2.4 - 2.7
- Hive: 0.13 - 1.2
- HBase: 0.98/0.99, 1.1.x
- JDK: 1.7+

YARN和MapReduce配置

KAP任务引擎递交到Hadoop的计算任务，需一定的内存资源，YARN的配置，需满足如下最小条件：

- Node Memory allocated (yarn.nodemanager.resource.memory-mb): 4096Mb
- Map Task Memory (mapreduce.map.memory.mb): 4096Mb
- Reduce Task Memory (mapreduce.reduce.memory.mb): 1024Mb
- Container Memory Maximum (yarn.scheduler.maximum-allocation-mb): 4096Mb

如果使用虚拟机的Hadoop Sandbox环境运行KAP时，请给虚拟机分配至少10GB内存和2个处理器。

推荐硬件配置

两路Intel至强处理器，6核（或8核）CPU, 主频2.3GHz或以上； 64GB ECC DDR3以上; 至少1个1TB的SAS硬盘(3.5寸), 7200RPM，RAID1； 至少两个1GbE以太网电口

推荐的Linux发行版

- Red Hat Enterprise Linux 6.4, 6.5
- CentOS 6.4, 6.5

在大多数情况下，KAP的SHELL脚本可以帮助您自动完成KAP的启动。

安装KAP

KAP二进制包拷贝至本地并解压

```
cd /usr/local
tar -zxvf kylin-kap-2.0-bin.tar.gz
```

设置环境变量KYLIN_HOME为KAP的解压缩目录

```
export KYLIN_HOME=/usr/local/kylin-kap-2.0-bin
```

确保运行KAP的shell账户，已经获得访问Hadoop各服务的权限。如果您还不确定的话，可以执行check-env.sh快速检测，如果有问题的话，它会将警告或错误信息输出在控制台。

```
${KYLIN_HOME}/bin/check-env.sh
```

启动KAP

执行 `bin/kylin.sh start` 命令，KAP将在后台开始启动，您可以用tail等命令观察logs/kylin.log文件，以了解启动进度。

```
${KYLIN_HOME}/bin/kylin.sh start
```

访问KAP

在KAP启动完成后，可以打开web浏览器，访问此KAP实例的web页面： http://host_name:7070/kylin

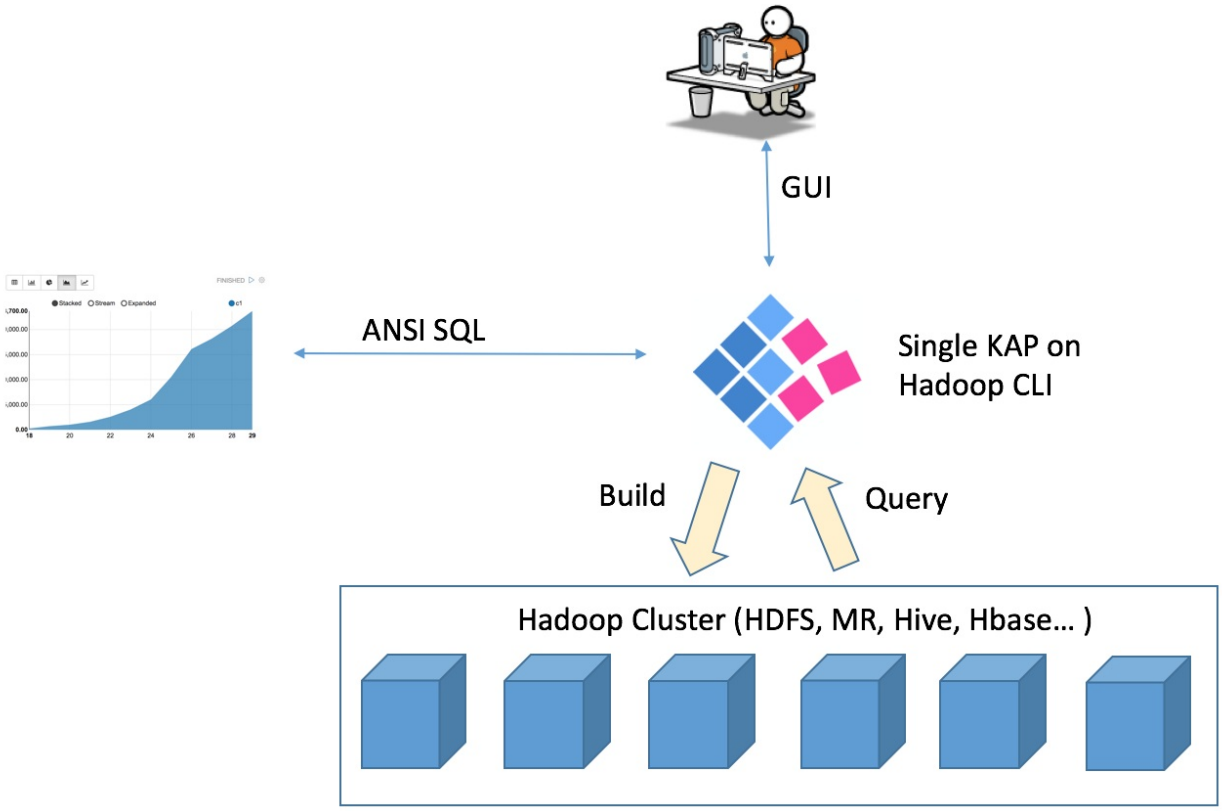
请替换“host_name”为具体的机器名、IP地址或域名。默认KAP启动在7070端口；一切正常的话，KAP的登录页面会显示。

停止KAP

执行 `bin/kylin.sh stop` 命令，停止KAP进程： `${KYLIN_HOME}/bin/kylin.sh stop`

单节点部署

通常部署单个KAP节点，已经能够满足中小规模(QPS<50)的查询需求；单节点部署具有简单、快速的特点。部署过程即上一节的安装过程，下图是一个单节点部署的示意图。



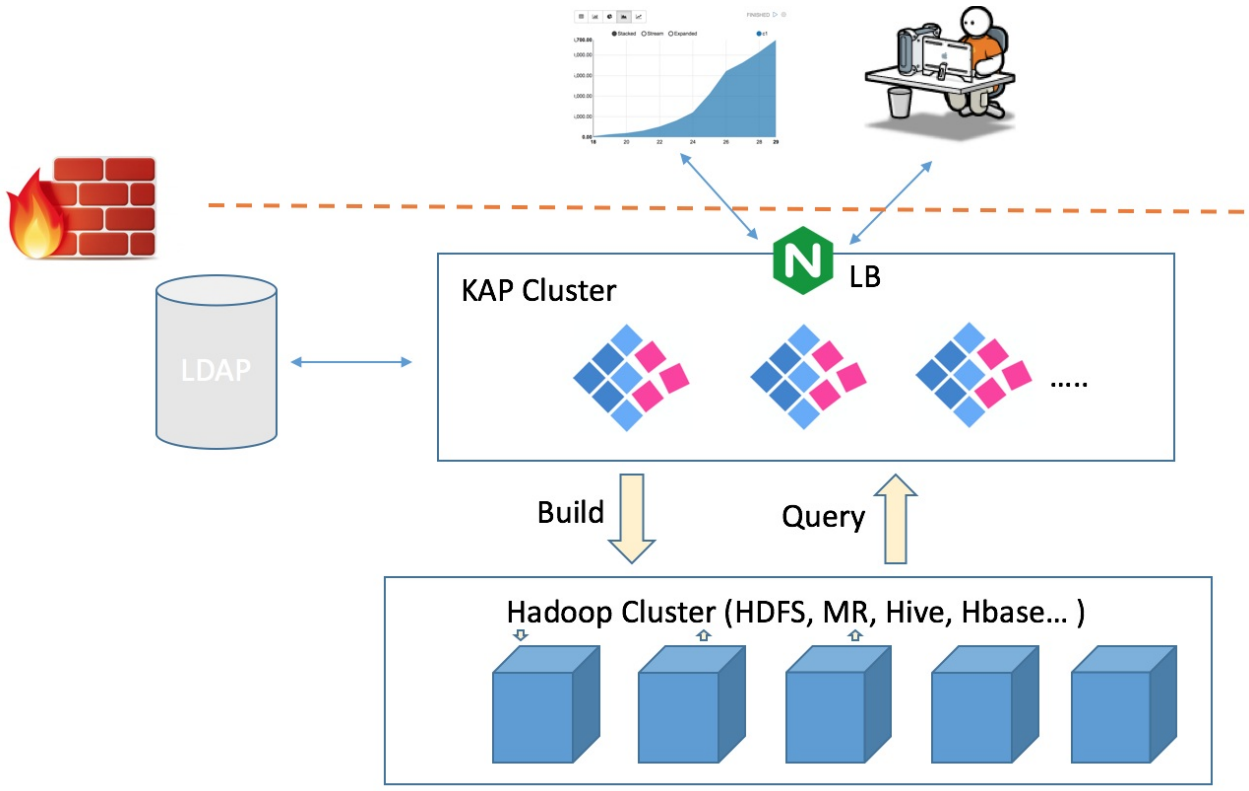
多节点（集群）部署

KAP实例是无状态的服务，所有的状态信息，都存储在HBase中；启用多个KAP节点的集群部署，让这些节点分担查询压力，互为备份，提供服务的高可用性。

将多个KAP节点组成集群，需要确保它们：

- 使用在同一Hadoop集群和HBase集群
- 没有端口冲突; 最好分开在不同服务器上，以做到互不影响。
- 使用同一张HBase metadata表，即 `kylin.metadata.url` 值相同
- 只有一个KAP实例运行任务引擎（ `kylin.server.mode=all` ），其它KAP实例都是查询引擎(`kylin.server.mode=query`)模式。或者启用任务引擎High Availability，参考下一章的配置方法.

为了将外部请求发给集群，而不是单个节点，需要部署一个负载均衡器（Load Balancer），如Apache HTTP Server或Nginx Server。负载均衡器通过一定策略决定将某个请求转给某个节点，并且在节点失效时重试其它节点。终端用户通过负载均衡器的地址来访问KAP。为了便于用户和角色的管理，通常此时会启用LDAP集成的安全验证。

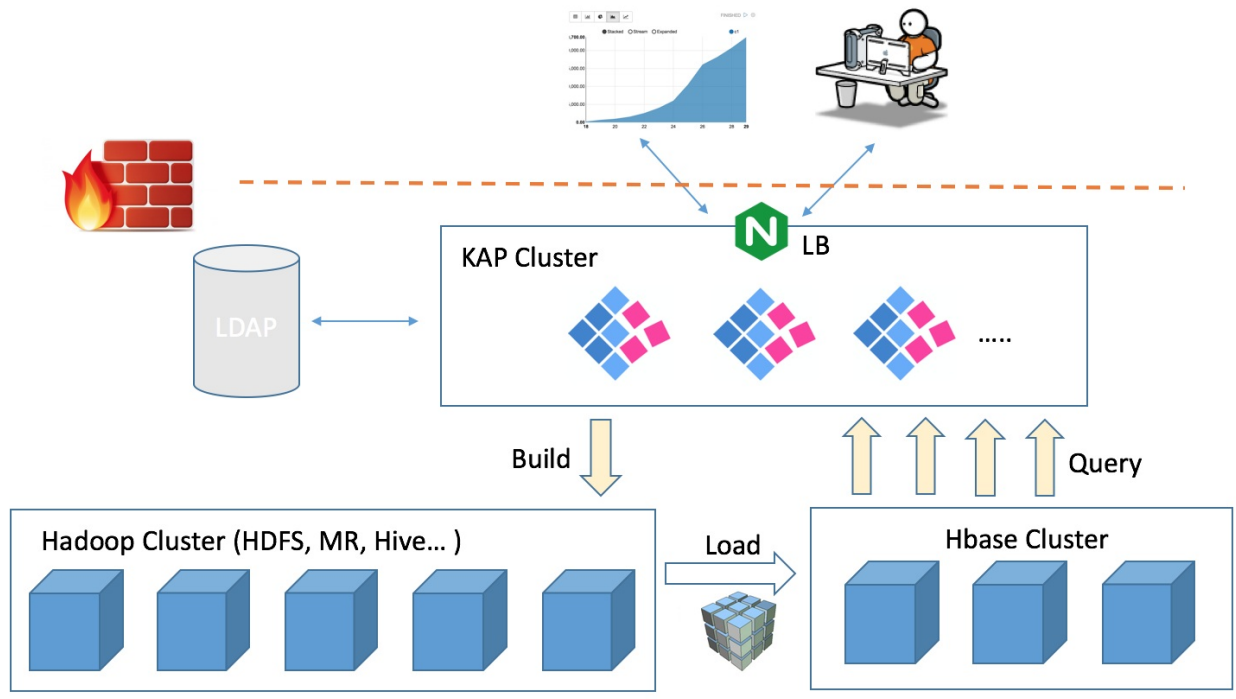


读写分离的部署

通常，KAP连接一个Hadoop集群(HBase也运行在这个集群上)，所有的负载，包括Cube构建和Cube查询都在这个集群上进行，相互影响，容易导致性能不稳定，特别是及时性要求很高的查询负载，要尽可能地独立运行。

采用Hadoop与HBase分开部署，将Cube运算递交到Hadoop集群，而Cube查询递交到HBase集群，可以完全隔离这两种工作负载。Cube构建过程中有很多写操作，而Cube查询则主要是只读操作，这样的部署也称为读写分离的部署。

读写分离的部署架构，参考下图。



部署通常有以下步骤：

- 分布部署Hadoop（计算）集群和HBase（存储）集群；两者的Hadoop核心版本要一致，分别有各自的HDFS、Zookeeper等组件；
- 在准备运行KAP的服务器上，安装和配置Hadoop（计算）集群的客户端；通过 `hadoop`，`hdfs`，`hive`，`mapred` 等命令，可以访问Hadoop集群上的服务和资源；
- 在准备运行KAP的服务器上，安装和配置HBase（存储）集群的HBase客户端；通过 `hbase` 命令，可以访问和操作HBase集群；
- 确保Hadoop（计算）集群和HBase（存储）集群的网络互通，且无需额外验证；可以从Hadoop（计算）集群的任一节点上，拷贝文件到HBase（存储）集群的任一节点。
- 确保在准备运行KAP的服务器上，通过hdfs命令行加HBase集群NameNode地址的方式，可以访问和操作存储集群的HDFS。
- 准备运行KAP的服务器，在网络上应靠近HBase集群，以确保密集查询时的网络低延迟；
- 编辑conf/kylin.properties，设置 `kylin.hbase.cluster.fs` 为HBase集群HDFS的url，例如：`kylin.hbase.cluster.fs=hdfs://hbase-cluster-nn01.example.com:8020`
- 重启KAP

卸载KAP

KAP以非侵入方式运行，故停止各个KAP服务进程，即停止在集群上的活动。

完整卸载KAP，并清除所有数据，请按如下步骤：

1. 在所有KAP节点上停止KAP实例： $KYLIN_HOME/bin/kylin.sh stop
2. 在某个KAP节点上备份元数据，随后建议将元数据拷贝到更可靠的存储器上： $KYLIN_HOME/bin/metastore.sh backup
3. 删除HBase存储；如果您的HBase集群只运行了一个KAP环境，且确定要卸载该KAP环境，可以使用hbase命令（如下）快速删除所有已构建的Cube; 否则，请谨慎使用，建议使用垃圾清理中介绍的方法有选择地删除Cube。

```
hbase shell
disable_all 'KYLIN_.*'
drop_all 'KYLIN_.*'
```

4. 删除KAP在HDFS上的工作目录：首先检查conf/kylin.properties文件，确定工作目录名，如“kylin.hdfs.working.dir=/kylin”， 使用 `hdfs` 命令行删除此目录：

```
hdfs fs -rmr /kylin
```

5. 删除KAP元数据表，首先检查conf/kylin.properties文件，确定元数据表名，如"kylin.metadata.url=kylin_metadata@hbase", 则元数据表名为“kylin_metadata”。使用hbase shell删除该表：

```
hbase shell
disable 'kylin_metadata'
drop 'kylin_metadata'
```

6. 在所有KAP节点上删除KAP安装：rm -rf $KYLIN_HOME

至此，卸载完成。

配置文件

把KAP安装到集群节点之后，往往还需要对KAP进行配置，一方面将KAP接入现有的Apache Hadoop、Apache HBase、Apache Hive环境，另一方面可以根据实际环境条件对KAP进行性能优化。

在KAP安装目录的conf目录里，保存了对KAP进行配置的所有配置文件。读者可以修改配置文件中的参数，以达到环境适配、性能调优等目的。在conf目录下默认存在以下配置文件：

kylin.properties

该文件是KAP服务所用的全局配置文件，和KAP有关的配置项都在此文件中。具体配置项在下文会有详细讲解。

kylin_hive_conf.xml

该文件包含了Apache Hive任务的配置项。在构建Cube的第一步通过Hive生成中间表时，会根据该文件的设置调整Hive的配置参数。

kylin_job_conf_inmem.xml

该文件包含了Map Reduce任务的配置项。当Cube构建算法是Fast Cubing时，会根据该文件的设置来调整构建任务中的Map Reduce参数。

kylin_job_conf.xml

该文件包含了Map Reduce任务的配置项。当kylin_job_conf_inmem.xml不存在，或Cube构建算法是Layer Cubing时，用来调整构建任务中的Map Reduce参数。

重要配置项

KAP配置文件中，最重要就是kylin.properties了。在本节，将对一些最常用的配置项作详细介绍。

kylin.metadata.url

指定KAP元数据库的URL。默认为HBase中kylin_metadata表，用户可以手动修改表名以使用HBase中的其他表保存元数据。在同一个HBase集群上部署多个KAP服务时，可以为每个KAP服务配置一个元数据库URL，以实现多个KAP服务间的隔离。例如，Production实例设置该值为kylin_metadata_prod，Staging实例设置该值为kylin_metadata_staging，在Staging实例中的操作不会对Production环境产生影响。

kylin.hdfs.working.dir

指定KAP服务所用的HDFS路径。默认为HDFS上/kylin的目录下，以元数据库URL中的HTable表名为子目录。例如，如果元数据库URL设置为kylin_metadata@hbase，那么该HDFS路径默认值就是/kylin/kylin_metadata。请预先确保启动KAP的用户有读写该目录的权限。

kylin.server.mode

指定KAP服务的运行模式，值可以是“all”，“job”，“query”中的一个，默认是“all”。Job模式指该服务仅用于Cube任务调度，不用于SQL查询。Query模式表示该服务仅用于SQL查询，不用于Cube构建任务的调度。“all“模式指该服务同时用于任务调度和SQL查询。

kylin.job.hive.database.for.intermediatetable

指定Hive中间表保存在哪个Hive数据库中，默认是default。如果执行KAP的用户没有操作default数据库的权限，可以修改此参数以使用其他数据库。

kylin.hbase.default.compression.codec

KAP创建的HTable所采用的压缩算法，配置文件中默认使用了snappy。如果实际环境不支持snappy压缩，可以修改该参数以使用其他压缩算法，如lzo、gzip、lz4等，删除该配置项即不启动任何压缩算法。

kylin.security.profile

指定KAP服务启用的安全方案，可以是 `ldap` 、 `saml` 、 `testing` 。默认值是testing，即使用固定的测试账号进行登录。用户可以修改此参数以接入已有的企业级认证体系，如ldap、saml。具体设置可以参考其他章节。

kylin.rest.timezone

指定KAP的Rest服务所使用的时区，默认是 `PST` 。用户可以根据具体应用的需要修改此参数。

kylin.hive.client

指定Hive命令行类型，可以是 `cli` 或 `beeline` 。默认是 `cli` ，即hive cli。如果实际系统只支持beeline作为Hive命令行，可以修改此配置为beeline。

kylin.hive.client

指定Hive命令行类型，可以是 `cli` 或 `beeline` 。默认是 `cli` ，即hive cli。如果实际系统只支持beeline作为Hive命令行，可以修改此配置为beeline。

kylin.hive.beeline.params

当使用beeline做为hive的client工具时，需要配置此参数，以提供更多信息给beeline；举例，如果您需要这样使用beeline来执行一个sql文件的话：

```
beeline -u root -u 'jdbc:hive2://localhost:10000' -f abc.sql
```

那么，请设置此参数：

```
kylin.hive.beeline.params=beeline -u root -u 'jdbc:hive2://localhost:10000'
```

deploy.env

指定KAP部署的用途，可以是DEV、PROD、QA。默认是QA; 在DEV模式下一些开发者功能将被启用, 在PROD模式下禁止创建新Cube。

任务引擎HA配置

KAP的任务引擎负责执行用户递交的构建任务，当任务完成时，通知用户和各个KAP节点。默认地，在KAP集群里只能有一个节点运行任务引擎，如果该节点失效，将导致任务引擎不能自动恢复。

要启用任务引擎高可用(HA)及自动恢复，需要额外配置。

- KAP的任务引擎HA依赖Zookeeper实现服务状态的监测和动态分配，故需要获取并在kylin.properties里配置Zookeeper信息；Zookeeper是Hadoop和HBase集群必需的服务，您可以选择使用现有的Zookeeper，或启动新的Zookeeper：

```
kylin.zookeeper.address=host1:2181,host2:2181,host3:2181
```

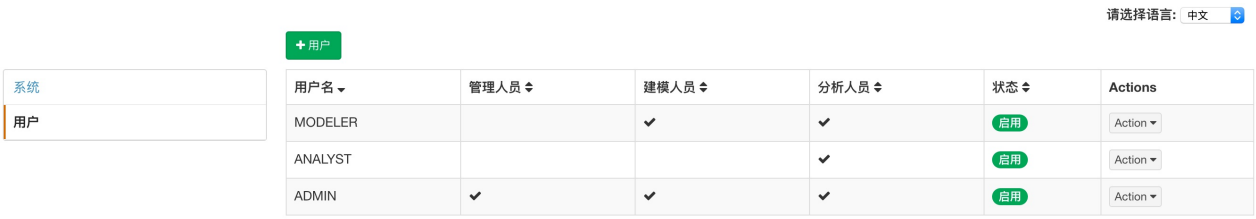
- 在kylin.properties里设置启用基于zookeeper的任务执行器：

```
kylin.enable.scheduler=1
```

- 重启所有KAP节点，任务引擎将在所有节点中动态分配。

用户管理

登录KAP后，点击导航栏的'系统'按钮，进入系统管理页面，点击左侧的用户栏进入用户管理页面。只有Admin能看到该页面。 KAP默认会初始化三个用户，对应帐号信息为 管理员(ADMIN/KYLIN), 建模人员(MODELER/MODELER), 分析人员(ANALYST/ANALYST)



角色

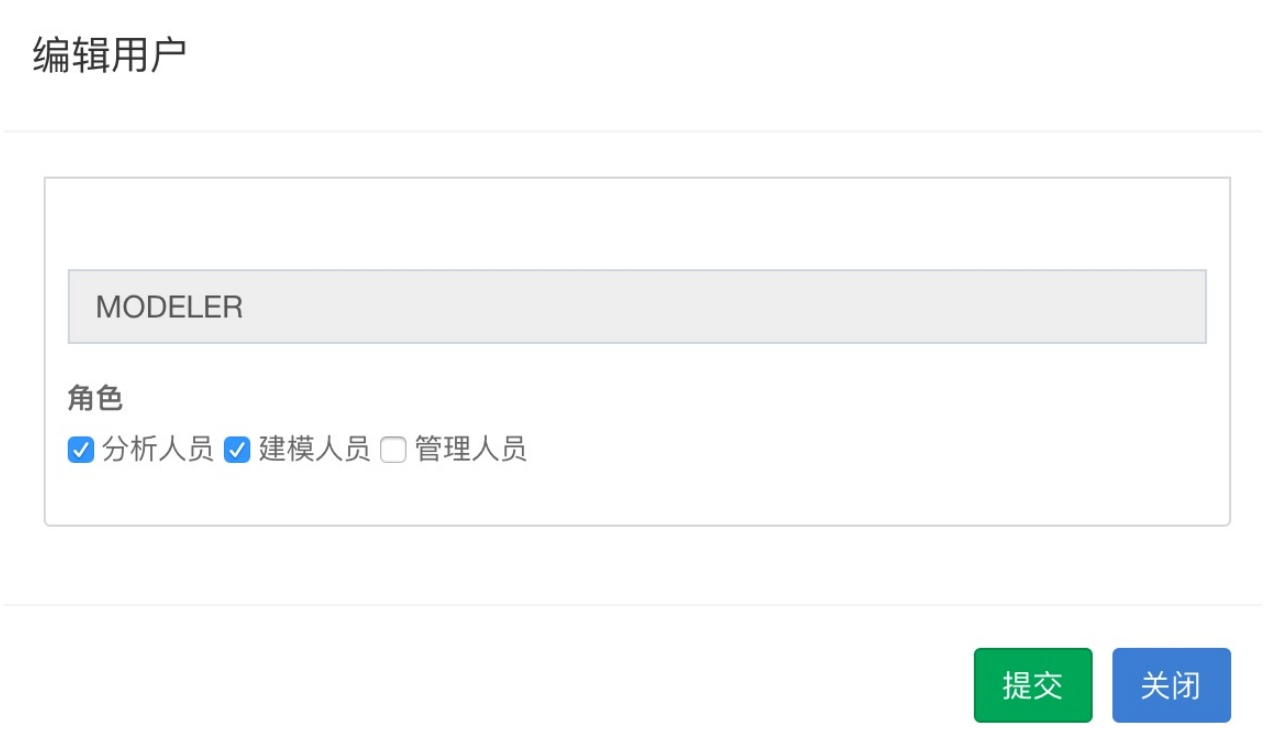
系统默认有三个角色，管理人员、建模人员和分析师。 分析师主要权限为查询功能，可以查看有权限的实体信息(项目，模型，Cube 等),可以查询构建的数据。 建模人员可以创建模型, Cube，并可以进行各类操作（构建，编辑，删除，清理，启用，禁用等，包含所有分析师具有的权限）。 管理人员具有系统所有相关的权限。

添加用户

管理员可以添加新的用户， 需要填写用户名，密码，角色信息。

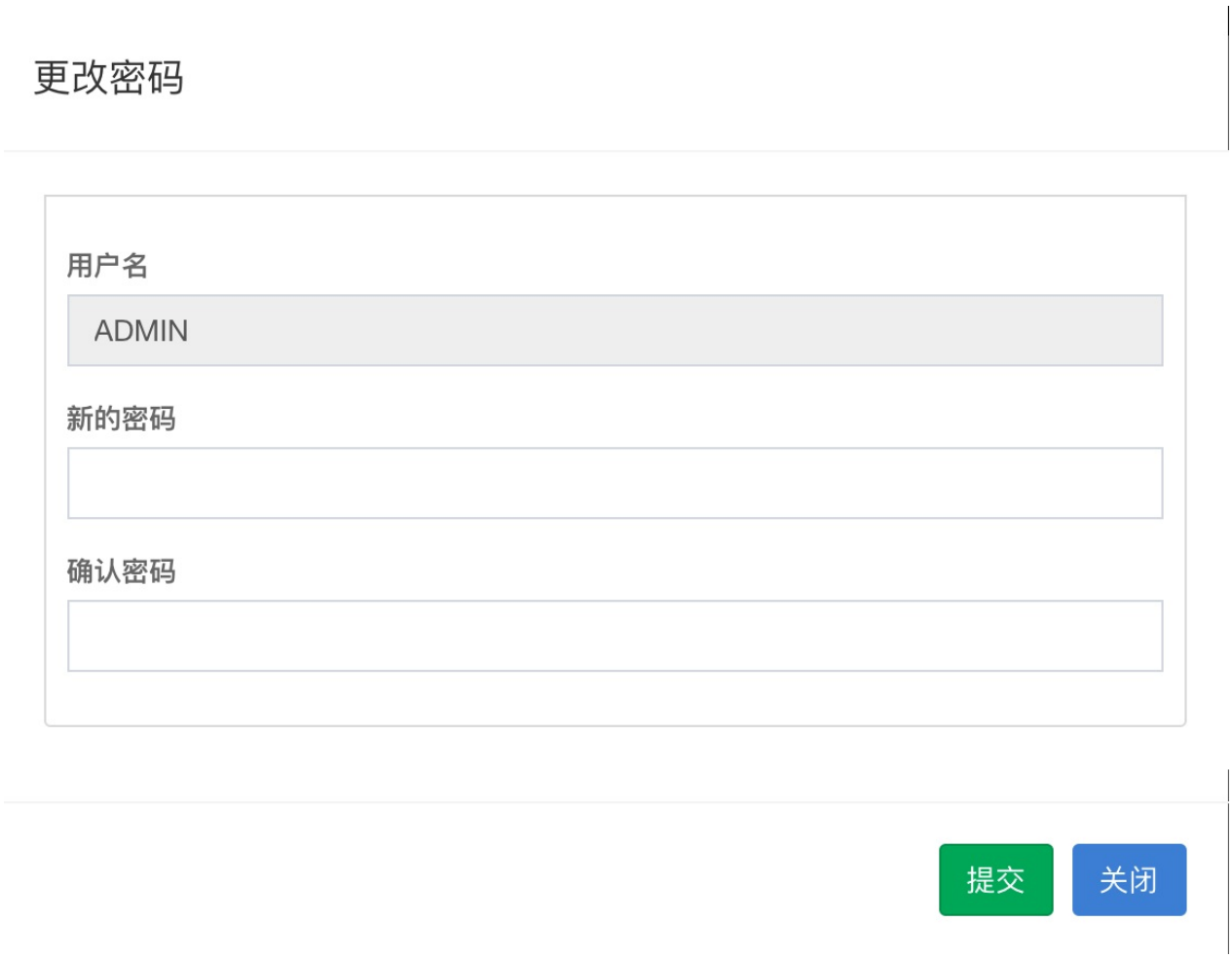
编辑用户

用户添加完成后，管理员可以编辑用户角色信息。



更改密码

管理员可以更改密码信息，需重复输入两次新密码。



删除用户

管理员可以在此页面删除用户，删除后将不能恢复。

启用/禁用用户

管理员可以启用或禁用用户，用户被禁用后将不可再登录系统。

普通用户修改密码

点击导航栏最右侧的欢迎标签，可以看到'设置'选项，点击进入个人更改密码页面。

更改密码

旧密码

新的密码

确认密码

更新密码

LDAP验证

KAP支持与LDAP服务器集成完成用户验证。这种验证是通过Spring Security框架实现的，所以具有良好的通用性。在启用LDAP验证之前，建议您联系LDAP管理员，以获取必要的信息。

配置LDAP服务器信息

首先，在conf/kylin.properties中，配置LDAP服务器的URL, 必要的用户名和密码（如果LDAP Server不是匿名访问）；为安全起见，这里的密码是需要加密（加密算法AES），您可以运行 "${KYLIN_HOME}/bin/kylin.sh io.kyligence.kap.tool.general.CryptTool AES your_password"来获得加密后的值，然后填写在kylin.properties中，如下：

```
ldap.server=ldap://<your_ldap_host>:<port>
ldap.username=<your_user_name>
ldap.password=<your_password_hash>
```

其次，提供检索用户信息的模式, 例如从某个节点开始查询，需要满足哪些条件等；下面是一个例子，供参考:

```
ldap.user.searchBase=OU=UserAccounts,DC=mycompany,DC=com
ldap.user.searchPattern=(&(AccountName={0})(memberOf=CN=MYCOMPANY-USERS,DC=mycompany,DC=com))
ldap.user.groupSearchBase=OU=Group,DC=mycompany,DC=com
```

如果您需要服务账户（供系统集成）可以访问KAP，那么依照上面的例子配置 `ldap.service.*` ，否则请将它们留空。

配置管理员群组和默认角色

KAP允许您将一个LDAP群组映射成管理员角色：在kylin.properties中，将"acl.adminRole"设置为"ROLE_" + GROUP_NAME形式. 例如，在LDAP中使用群组"KAP-ADMIN-GROUP"来管理所有KAP管理员，那么这里应该设置为:

```
acl.adminRole=ROLE_KAP-ADMIN-GROUP
acl.defaultRole=ROLE_ANALYST,ROLE_MODELER
```

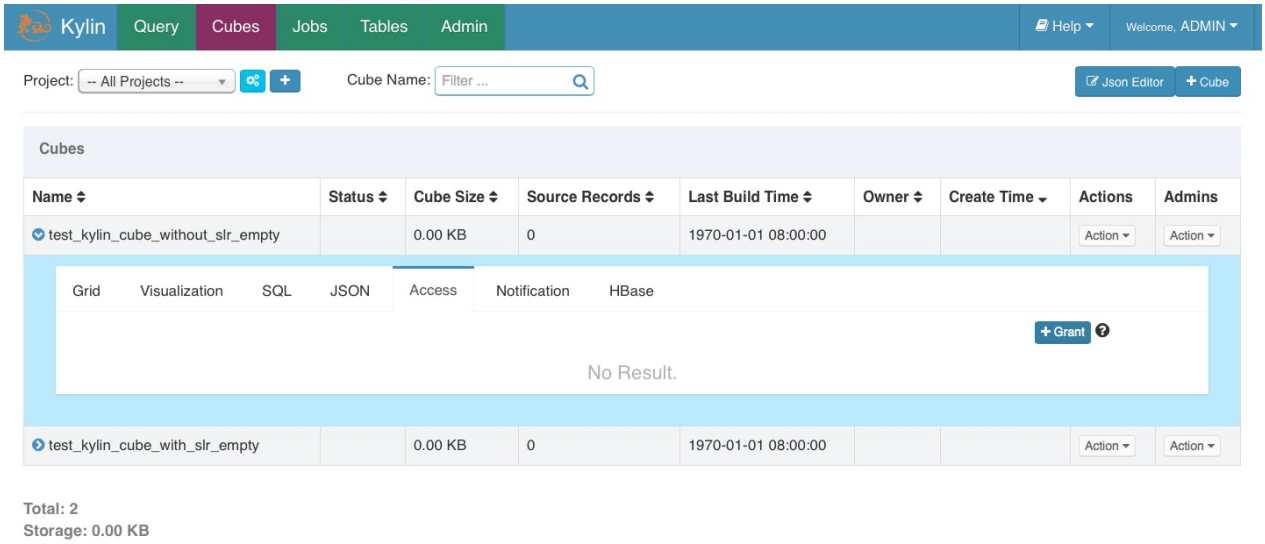
属性"acl.defaultRole"定义了赋予登录用户的权限，默认是分析师（ANALYST）和建模人员（MODELER）.

启用LDAP

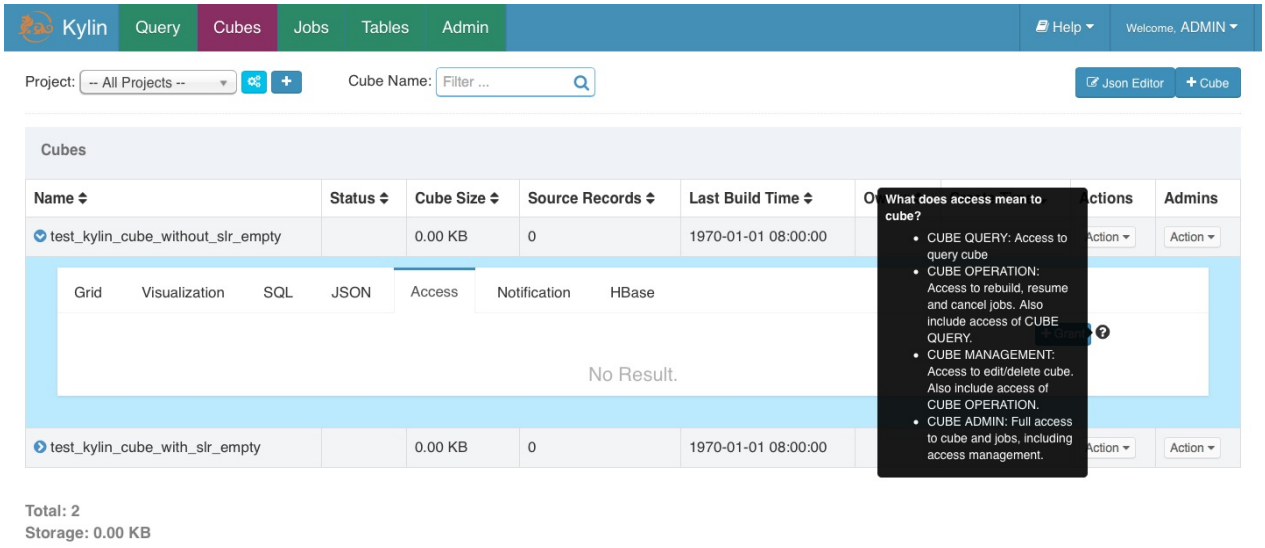
在conf/kylin.properties中，设置"kylin.security.profile=ldap"，然后重启KAP。

管理ACL

在 `Cubes` 页面，双击cube行查看详细信息。在这里我们关注 `Access` 标签。点击 `+Grant` 按钮进行授权。



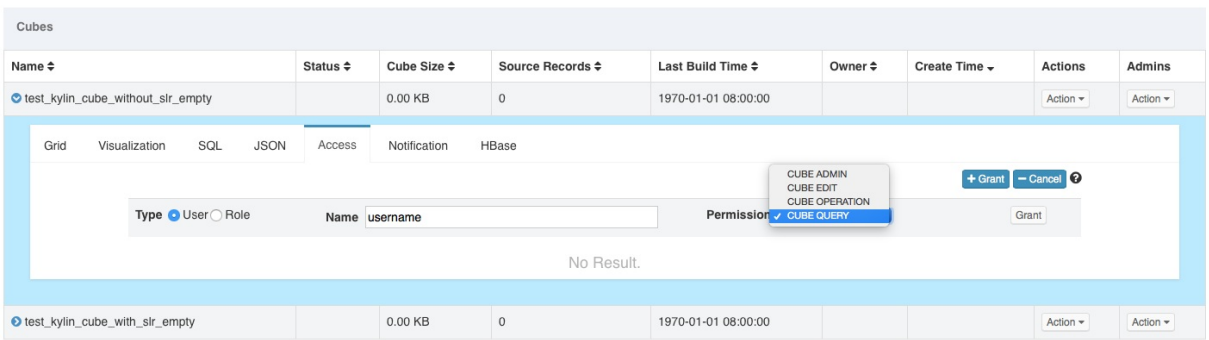
一个cube有四种不同的权限。将你的鼠标移动到 `?` 图标查看详细信息。



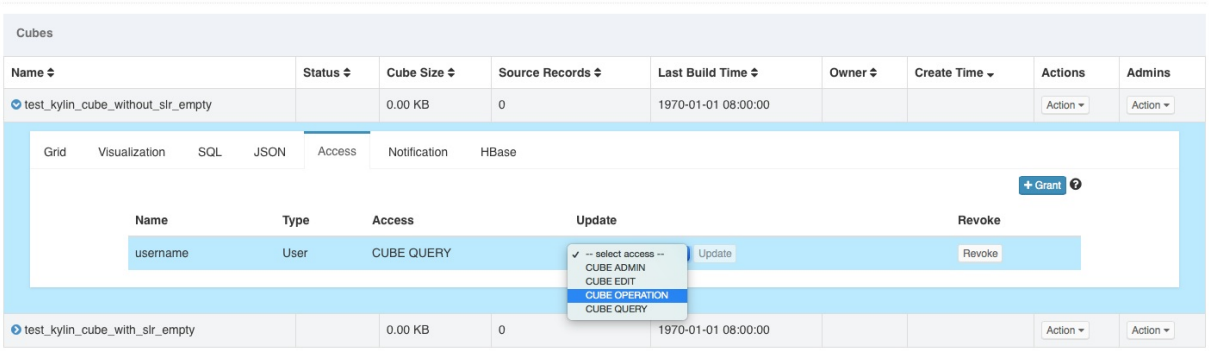
授权对象也有两种： `User` 和 `Role` 。 `Role` 是指一组拥有同样权限的用户。

授予用户权限

- 选择 `User` 类型，输入你想要授权的用户的用户名并选择相应的权限。



- 然后点击 `Grant` 按钮提交请求。在这一操作成功后，你会在表中看到一个新的表项。你可以选择不同的访问权限来修改用户权限。点击 `Revoke` 按钮可以删除一个拥有权限的用户。



授予角色权限

- 选择 `Role` 类型，通过点击下拉按钮选择你想要授权的一组用户并选择一个权限。
- 然后点击 `Grant` 按钮提交请求。在这一操作成功后，你会在表中看到一个新的表项。你可以选择不同的访问权限来修改组权限。点击 `Revoke` 按钮可以删除一个拥有权限的组。

当KAP顺利启动之后，默认会在安装目录下产生一个logs目录，该目录保存KAP运行过程中产生的所有日志文件。

日志文件

KAP的日志主要包含以下三个文件：

kylin.log

该文件是主要的日志文件，所有的logger默认写入该文件，其中与KAP相关的日志级别默认是DEBUG。日志随日期轮转，即每天0点时将前一天的日志存放到以日期为后缀的文件中（如kylin.log.2014-01-01），并把新一天的日志保存到全新的kylin.log文件中。

kylin.out

该文件是标准输出的重定向文件，一些非Apache Kylin产生的标准输出（如tomcat启动输出、Hive命令行输出等）将被重定向到该文件。

kylin.gc

该文件是Apache Kylin的Java进程记录的GC日志。为避免多次启动覆盖旧文件，该日志使用了进程号作为文件名后缀（如kylin.gc.9188）。

日志分析

在这里，我们以查询为例，简单介绍一下如何在日志中获取查询的更多信息。首先在Web UI执行一个查询，然后马上到kylin.log文件尾部查找相关日志。当查询结束，我们会看到如下的记录片段：

```
2016-06-10 10:03:03,800 INFO  [http-bio-7070-exec-10] service.QueryService:251 :
==========================[QUERY]===============================
SQL: select * from kylin_sales
User: ADMIN
Success: true
Duration: 2.831
Project: learn_kylin
Realization Names: [kylin_sales_cube]
Cuboid Ids: [99]
Total scan count: 9840
Result row count: 9840
Accept Partial: true
Is Partial Result: false
Hit Exception Cache: false
Storage cache used: false
Message: null
==========================[QUERY]===============================
```

下表是对上述片段中主要字段的介绍：

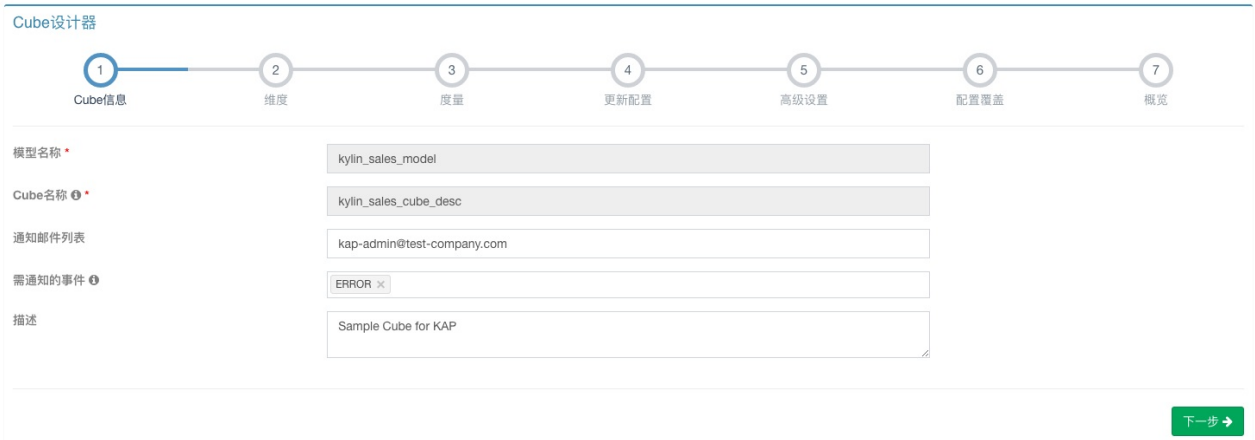
- SQL: 查询所执行的SQL语句
- User: 执行查询的用户名
- Success: 该查询是否成功
- Duration: 该查询所用时间（单位：秒）
- Project: 该查询所在项目
- Realization Names: 该查询所击中的Cube名称

日志配置

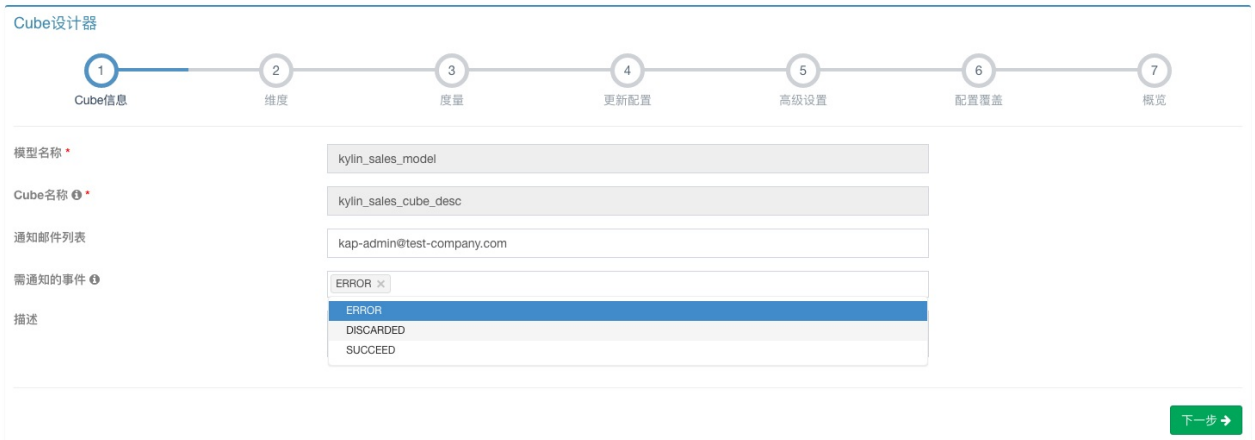
以上日志均可以通过修改配置文件进行调整。KAP使用log4j对日志进行配置，用户可以编辑KAP安装目录下conf目录中的kylin-server-log4j.properties文件，对日志级别、路径等进行修改。修改后，需要重启KAP服务才可生效。

报警设置

在KAP中，构建一个Cube往往至少需要花费几十分钟的时间。因此，当一个Cube构建任务完成或失败时，运维人员常常希望可以在第一时间得到通知，便于进行下一步的增量构建或故障排查。KAP中提供了一个邮件通知的功能，可以在Cube状态发生改变时，向Cube管理员发送电子邮件。 想要通过电子邮件实现任务报警，需要先在配置文件kylin.properties中进行设置： mail.enabled 设置该项为true，即可启动邮件通知功能 mail.host 设置该项为邮件的SMTP服务器地址 mail.username 设置为邮件的SMTP登录用户名 mail.password 设置为邮件的SMTP登录密码 mail.sender 设置为邮件的发送邮箱地址 设置完毕后，重新启动KAP服务，这些配置即可奏效。 接下来，需要对Cube进行配置。首先设置Cube的邮件联系人，作为邮件通知的收件人。如下图所示：



然后，选择一些状态作为邮件通知的触发条件。即当Cube构建任务切换到这些状态时，就给用户发送邮件通知。



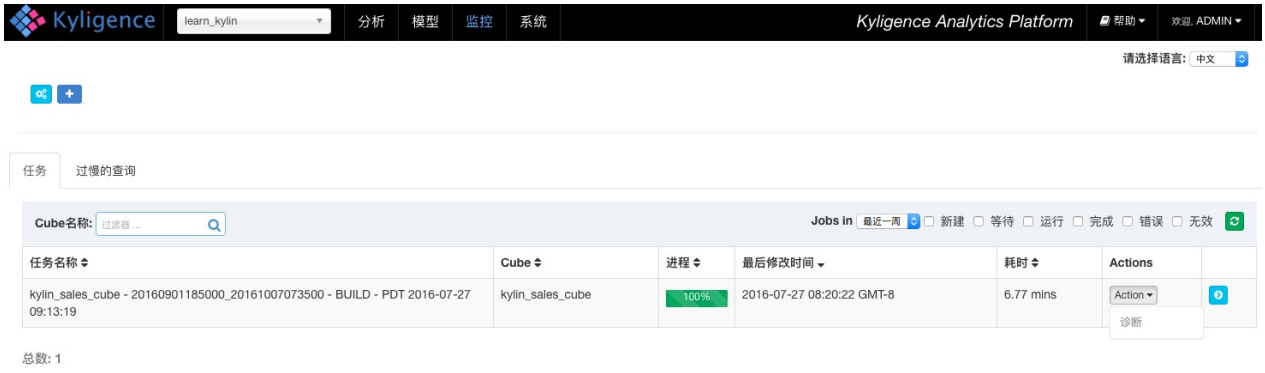
在KAP中，在Web UI上提供了一个“诊断”功能。该功能的入口总共有两处：项目诊断和任务诊断。

项目诊断

用户经常会遇到一些棘手的问题，例如Cube创建失败、SQL查询失败，或SQL查询时间过长等。运维人员需要抓取相关信息并进行分析，以找出问题的根本所在。这时候，运维人员可以选择错误所在的项目，然后单击System页面下的Diagnosis按钮，系统会自动生成一个zip格式的压缩包，包含了该项目下所有的有用信息，帮助运维人员缩小问题排查范围，并为问题的快速定位提供了方向。

任务诊断

当一个Cube构建任务执行失败或时间过长，运维人员还可以单击Job下的Diagnosis菜单项，以生成一个专门针对该任务的zip压缩包，帮助运维人员快速分析任务的失败原因。



基本运维

KAP服务器每天会接受不同用户提交的多个Cube构建任务，且往往因为Cube设计不当或集群环境异常等原因，导致Cube构建失败或时间过长，需要运维人员提高警惕；此外，KAP服务运行一段时间之后，一些存在于HBase或HDFS上的数据会成为无用数据，需要定时对存储器进行垃圾清理。

运维任务

作为KAP的运维人员，工作中需要明确：

- 确保KAP服务运行正常
- 确保KAP对集群资源的利用正常
- 确保Cube构建任务正常
- 灾难备份和恢复
- ……

以上列出了一些KAP的基本运维职责和工作，常言道“工欲善其事，必先利其器”。运维人员需要熟练地运用本章提到的各种工具，一方面对KAP服务的每日运行状况进行监控，另一方面在遇到问题时能够找到合理的解决途径。

为了保障KAP服务的正常运行，运维人员可以对KAP的日志进行监控，确保KAP进程的稳定运行。为了确保KAP对集群资源的正常利用，运维人员需要经常查看Map Reduce任务队列的闲忙程度，以及HBase存储的利用率（如HTable数量、Region数量等）。为了确保Cube构建任务的正常，请根据邮件通知或Web UI的监控页面对任务进行监控。

元数据备份

元数据是KAP中最重要的数据之一，备份元数据是运维工作中一个至关重要的环节。只有这样，在由于误操作导致整个KAP服务或某个Cube异常时，才能将KAP快速从备份中恢复出来。

一般的，在每次进行故障恢复或系统升级之前，对元数据进行备份是一个良好的习惯，这可以保证KAP服务在系统更新失败后依然有回滚的可能，在最坏情况下依然保持系统的鲁棒性。

KAP提供了一个工具用于备份KAP的元数据，具体用法是：

```
$KYLIN_HOME/bin/metastore.sh backup
```

当看到如下提示时即为备份成功：

```
metadata store backed up to /usr/local/kylin/meta_backups/meta_2016_06_10_20_24_50
```

在上面的例子里，这个命令会把KAP用到的所有元数据以文件形式下载到本地目录当中（如/usr/local/kylin/meta_backups/meta_2016_06_10_20_24_50）。目录结构如下表所示：

目录名	介绍
project	包含了项目的基本信息，项目所包含其他元数据类型的声明
model_desc	包含了描述数据模型基本信息、结构的定义
cube_desc	包含了描述Cube模型基本信息、结构的定义
cube	包含了Cube实例的基本信息，以及下属Cube Segment的信息
cube_statistics	包含了Cube实例的统计信息包含了Cube实例的统计信息
table	包含了表的基本信息，如Hive信息
table_exd	包含了表的扩展信息，如维度
table_snapshot	包含了Lookup表的镜像
dict	包含了使用字典列的字典
execute	包含了Cube构建任务的步骤信息
execute_output	包含了Cube构建任务的步骤输出

此外，元数据备份也是故障查找的一个工具，当系统出现故障导致前端频繁报错，通过该工具下载元数据并查看文件，往往能对确定元数据是否存在问题提供巨大帮助。

元数据恢复

有了元数据备份，往往用户还需要对元数据从备份进行恢复。和备份类似，KAP中提供了一个元数据恢复的工具，具体用法是：

```
$KYLIN_HOME/bin/metastore.sh restore /path/to/backup
```

如果从前一节元数据备份的例子中恢复，需要在命令行中执行：

```
cd $KYLIN_HOME
bin/metastore.sh restore meta_backups/meta_2016_06_10_20_24_50
```

等待恢复操作成功，用户可以在Web UI上单击“重新加载元数据”按钮对元数据缓存进行刷新，即可看到最新的元数据。

垃圾清理

在KAP运行一段时间之后，有很多数据因为不再使用而变成了垃圾数据，这些数据占据着大量HDFS、HBase等资源，当积累到一定规模会对集群性能产生影响。

这些垃圾数据主要包括：

- Purge之后原Cube的数据
- Cube合并之后原Cube Segment的数据
- 任务中未被正常清理的临时文件
- 很久之前Cube构建的日志和任务历史

为了对这些垃圾数据进行清理，KAP提供了两个常用的工具。请特别注意，数据一经删除将彻底无法恢复！建议使用前进行元数据备份，并对目标资源进行谨慎核对。

清理元数据

第一个是元数据清理工具，该工具有一个delete参数，默认是false。只有当delete参数为true，工具才会真正对无效的元数据进行删除。该工具的执行方式如下：

```
$KYLIN_HOMEbin/metastore.sh cleanup [--delete true]
```

第一次执行该工具时建议省去delete参数，这样会只列出所有可以被清理的资源供用户核对，而并不实际进行删除操作。当用户确认无误后，再添加delete参数并执行命令，才会进行实际的删除操作。 默认情况下，该工具会清理的资源列表如下：

- 2天前创建的已无效的Lookup表镜像、字典、Cube统计信息
- 30天前结束的Cube构建任务的步骤信息、步骤输出

清理存储器

第二个工具是存储器清理工具。顾名思义，就是对HBase和HDFS上的资源进行清理。同样的，该工具也有一个delete参数，默认是false。当且仅当delete参数的值是true，工具才会对存储器中的资源真正执行删除操作。该工具的执行方式如下：

```
$KYLIN_HOME/bin/kylin.sh storage cleanup [--delete true]
```

第一次执行该工具时建议省去delete参数，这样会只列出所有可以被清理的资源供用户核对，而并不实际进行删除操作。当用户确认无误后，再添加delete参数并执行命令，才会进行实际的删除操作。

默认情况下，该工具会清理的资源列表如下：

- 创建时间在2天前，且已无效的HTable
- 在Cube构建时创建的但未被正常清理的的Hive中间表、HDFS临时文件

系统升级

KAP是基于Apache Kylin进行二次开发的，因此支持从Apache Kylin升级到KAP。

从Apache Kylin 1.5.1/1.5.2升级到KAP 2.0

KAP 2.0与Kylin 1.5.1/1.5.2版本的元数据、Cube数据都是兼容的，不需要对数据进行升级。但是因为HBase协处理的通讯协议发生了改动，而已有的HTable仍然绑定了老版本的HBase协处理器Jar包，因此必须为现有的HTable重新部署HBase协处理器Jar包。在命令行中执行如下命令即可为所有的HTable重新部署HBase协处理器jar包：

```
$KYLIN_HOME/bin/kylin.sh org.apache.kylin.storage.hbase.util.DeployCoprocessorCLI $KYLIN_HOME/lib/kylin-coprocessor-*
.jar all
```

从Apache Kylin 1.5.0及之前版本升级到KAP 2.0

如果用户希望从Apache Kylin 1.5.0升级到KAP 2.0，需要先升级到Apache Kylin 1.5.1，然后再按照上文的介绍从Apache Kylin 1.5.1升级到KAP 2.0。

Apache Kylin 1.5.1和Apache Kylin 1.5.0及之前版本的元数据格式并不兼容，用户需要对元数据进行升级。在升级后，原有的已构建好的Cube仍然可以继续进行查询。以下是升级过程：

1) 备份元数据

根据上文介绍的元数据备份方法，在执行升级操作前对元数据进行备份，以便于在升级失败后进行回滚。

2) 停止正在运行的Kylin实例

3) 备份配置文件

将目前Kylin安装目录中的conf目录备份至其他目录

4) 安装Kylin 1.5.1

前往Kylin官网下载1.5.1版本的二进制包，并解压到一个新的目录中，并把上一步备份好的conf目录复制到该安装目录。特别注意，新版本可能引入了一些新的配置参数，用户需要对新老版本的配置文件进行手动比较和合并。

5) 执行升级

把新的安装目录设置为KYLIN_HOME，然后将第一步的元数据备份拷贝到该目录。接下来在命令行中执行如下命令，会对拷贝进来的元数据文件进行升级：

```
export KYLIN_HOME="<path_of_1_5_0_installation>"
$KYLIN_HOME/bin/kylin.sh  org.apache.kylin.cube.upgrade.entry.CubeMetadataUpgradeEntry_v_1_5_1 <path_of_BACKUP_FOLDER
>
```

6) 上传元数据

利用上文介绍的元数据恢复工具，将升级后的元数据恢复到Kylin实例的元数据仓库中。如果这新老版本实例的元数据仓库使用了同一个HTable，则需要先对元数据仓库进行重置。可以参考如下命令：

```
$KYLIN_HOME/bin/metastore.sh reset
$KYLIN_HOME/bin/metastore.sh restore <path_of_workspace>
```

7) 重新部署HBase协处理器Jar包

和1.5.1到1.5.2版本升级一样，此处也需要对HBase协处理器进行重新部署，新版本的HBase协处理器Jar包放置在安装目录的lib目录中。具体部署方法可以参考前文，在此不再赘述。

8) 启动新Kylin实例

升级失败后回滚

如果升级任务失败，导致生产系统长时间无法运行，往往会对业务系统产生严重的影响。因此，为了保证生产系统的正常运行，必须立即在升级失败后进行回滚，将系统恢复到升级前的状态。一下是回滚的操作步骤：

1) 停止新Kylin或KAP实例

2) 恢复元数据

正如上文所述，升级前必须对元数据进行备份。这一步就需要从升级前备份的元数据目录中对元数据仓库进行恢复。如果新老Kylin实例的元数据仓库使用同一个HTable，需要先对元数据进行重置。请参考以下命令：

```
export KYLIN_HOME="<path_of_old_installation>"
$KYLIN_HOME/bin/metastore.sh reset
$KYLIN_HOME/bin/metastore.sh restore <path_of_BACKUP_FOLDER>
```

3) 重新部署HBase协处理器

因为升级过程中给HTable部署了新版本的HBase协处理器Jar包，回滚时需要把老版本的HBase协处理Jar包部署到HTable中。老版本的HBase协处理器Jar包放置于Kylin安装目录的lib目录中。具体部署方法可以参考前文，在此不再赘述。

4) 启动老版本Kylin实例

常见问题

常见问题

联系我们

欢迎您通过任何方式与我们联系：

商务咨询：info@kyligence.io

加入我们：talent@kyligence.io

产品问题反馈：support@kyligence.io

微信公众号：kyligence

地址：上海市浦东新区亮秀路112号Y1座405室