Alibaba Cloud E-MapReduce

開源組件介紹

檔案版本：20200426

法律#明

阿里雲提醒您在閱讀或使用本文檔之前仔細閱讀、充分理解本法律聲明各條款的內容。如果您閱讀或使用本文檔，您的閱讀或使用行為將被視為對本聲明全部內容的認可。

1. 您應當通過阿里雲網站或阿里雲提供的其他授權通道下載、擷取本文檔，且僅能用於自身的合法合規的商務活動。本文檔的內容視為阿里雲的保密資訊，您應當嚴格遵守保密義務；未經阿里雲事先書面同意，您不得向任何第三方披露本手冊內容或提供給任何第三方使用。
2. 未經阿里雲事先書面許可，任何單位、公司或個人不得擅自摘抄、翻譯、複製本文檔內容的部分或全部，不得以任何方式或途徑進行傳播和宣傳。
3. 由於產品版本升級、調整或其他原因，本文檔內容有可能變更。阿里雲保留在沒有任何通知或者提示下對本文檔的內容進行修改的權利，並在阿里雲授權通道中不時發布更新後的使用者文檔。您應當即時關注使用者文檔的版本變更並通過阿里雲授權渠道下載、擷取最新版的使用者文檔。
4. 本文檔僅作為使用者使用阿里雲產品及服務的參考性指引，阿里雲以產品及服務的“現狀”、“有缺陷”和“當前功能”的狀態提供本文檔。阿里雲在現有技術的基礎上盡最大努力提供相應的介紹及操作指引，但阿里雲在此明確聲明對本文檔內容的準確性、完整性、適用性、可靠性等不作任何明示或暗示的保證。任何單位、公司或個人因為下載、使用或信賴本文檔而發生任何差錯或經濟損失的，阿里雲不承擔任何法律責任。在任何情況下，阿里雲均不對任何間接性、後果性、懲戒性、偶然性、特殊性或刑罰性的損害，包括使用者使用或信賴本文檔而遭受的利潤損失，承擔責任（即使阿里雲已被告知該等損失的可能性）。
5. 阿里雲文檔中所有內容，包括但不限於圖片、架構設計、頁面配置、文字描述，均由阿里雲和/或其關係企業依法擁有其智慧財產權，包括但不限於商標權、專利權、著作權、商業秘密等。非經阿里雲和/或其關係企業書面同意，任何人不得擅自使用、修改、複製、公開傳播、改變、散布、發行或公開發表阿里雲網站、產品程式或內容。此外，未經阿里雲事先書面同意，任何人不得為了任何營銷、廣告、促銷或其他目的使用、公布或複製阿里雲的名稱（包括但不限於單獨為或以組合形式包含“阿里雲”、“Aliyun”、“萬網”等阿里雲和/或其關係企業品牌，上述品牌的附屬標誌及圖案或任何類似公司名稱、商號、商標、產品或服務名稱、網域名稱、圖案標示、標誌、標識或通過特定描述使第三方能夠識別阿里雲和/或其關係企業）。
6. 如若發現本文檔存在任何錯誤，請與阿里雲取得直接聯絡。

通用#定

格式	說明	範例
	該類警示資訊將導致系統重大變更甚至故障，或者導致人身傷害等結果。	 **禁止:** 重設操作將丟失使用者配置資料。
	該類警示資訊可能會導致系統重大變更甚至故障，或者導致人身傷害等結果。	 **警告:** 重啟操作將導致業務中斷，恢複業務時間約十分鐘。
	用於警示資訊、補充說明等，是使用者必須瞭解的內容。	 **注意:** 權重設定為0，該伺服器不會再接受新請求。
	用於補充說明、最佳實務、竅門等，不是使用者必須瞭解的內容。	 **说明:** 您也可以通過按Ctrl + A選中全部檔案。
>	多級菜單遞進。	單擊**設定** > **網路** > **設定網路類型**。
粗體	表示按鍵、菜單、頁面名稱等UI元素。	在**結果確認**頁面，單擊**確定**。
Courier字型	命令。	執行cd /d C:/window命令，進入Windows系統檔案夾。
斜體	表示參數、變數。	bae log list --instanceid Instance_ID
[]或者[a|b]	表示可選項，至多選擇一個。	ipconfig [-all|-t]
{}或者{a|b}	表示必選項，至多選擇一個。	switch {active|stand}

目#

1 Oozie 使用#明

> **说明:**
> 阿里雲 E-MapReduce 在 2.0.0 及之後的版本中提供了對 Oozie 的支援，如果需要在叢集中使用 Oozie，請確認叢集的版本不低於 2.0.0。

準備工作

在叢集建立出來之後，需要打通 ssh 隧道，詳細步驟請參考#unique_4。

這裡以 MAC 環境為例，使用 Chrome 瀏覽器實現連接埠轉寄（假設叢集 master 節點公網 IP 為 **xx.xx.xx.xx**）：

1. 登入到 master 節點。

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

2. 輸入密碼。
3. 查看原生 **id_rsa.pub** 內容（注意在本機執行，不要在遠端 master 節點上執行）。

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

4. 將原生 **id_rsa.pub** 內容寫入到遠程 master 節點的 ~/.ssh/authorized_keys 中（在遠端 master 節點上執行）。

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

5. 然後將步驟 2 中看到的內容粘貼進來，現在應該可以直接使用 ssh root@xx.xx.xx.xx 免密登入 master 節點了。
6. 在本機執行以下命令進行連接埠轉寄。

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

7. 啟動 Chrome（在本機新開 terminal 執行）。

```
/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
```

訪問 Oozie UI 頁面

在進行連接埠轉寄的 Chrome 瀏覽器中訪問：xx.xx.xx.xx:11000/oozie，localhost:11000/oozie 或者內網 ip:11000/oozie。

提交workflow作業

運行 Oozie 需要先安裝 Oozie 的sharelib：https://oozie.apache.org/docs/4.2.0/WorkflowFunctionalSpec.html#ShareLib

在 E-MapReduce 叢集中，預設給 Oozie 使用者安裝了 sharelib，即如果使用 Oozie 使用者來提交 workflow 作業，則不需要再進行 sharelib 的安裝。

由於開啟 HA的叢集和沒有開啟 HA 的叢集，訪問 NameNode 和 ResourceManager 的方式不同，在提交 oozie workflow job 的時候，job.properties 檔案中需要指定不同的 NameNode 和 JobTracker （ResourceManager）。具體如下：

- 非 HA 叢集

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

- HA 叢集

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

下面操作樣本中，已經針對是否是 HA 叢集配置好了，即範例代碼不需要任何修改即可以直接運行。關於 workflow 檔案的具體格式，請參考 Oozie 官方文檔：https://oozie.apache.org/docs/4.2.0/。

- **在非 HA 叢集上提交 workflow 作業**

  1. 登入叢集的主 master 節點。

     ```
     ssh root@master公網Ip
     ```

  2. 下載範例程式碼。

     ```
     [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. 將 Oozie workflow 代碼同步到 hdfs 上。

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

4. 提交 Oozie workflow 範例作業。

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

執行成功之後，會返回一個 jobId，類似：

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

5. 訪問 Oozie UI 頁面，可以看到剛剛提交的 Oozie workflow job。

- **在 HA 叢集上提交 workflow 作業**

1. 登入叢集的主 master 節點。

```
ssh root@主master公網Ip
```

可以通過是否能訪問 Oozie UI 來判斷哪個 master 節點是當前的主 master 節點， Oozie server 服務預設是啟動在主 master 節點 `xx.xx.xx.xx:11000/oozie`。

2. 下載 HA 叢集的範例程式碼。

```
[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. 將 Oozie workflow 代碼同步到 hdfs 上。

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

4. 提交 Oozie workflow 範例作業。

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

執行成功之後，會返回一個 jobId，類似：

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

5. 訪問 Oozie UI 頁面，可以看到剛剛提交的 Oozie workflow job。

2 ZooKeeper 使用#明

目前 E-MapReduce 叢集中預設啟動了 ZooKeeper 服務。

注意事項

目前無論叢集內有多少台機器，ZooKeeper 只會有 3 個節點。目前還不支援更多的節點。

建立叢集

E-MapReduce 建立叢集的軟體配置頁面，會預設勾選 ZooKeeper。

節點資訊

叢集建立成功，狀態空閑後，查看叢集的詳情頁面，可以查到 ZooKeeper 的節點資訊，E-MapReduce 會啟動 3 個 ZooKeeper 節點。如下圖所示，應用進程一欄標有 ZooKeeper 節點對應的內網 IP (連接埠預設為 2181)，即可訪問 ZooKeeper 服務。

3 Kafka

3.1 Kafka 快速入門

從EMR-3.4.0版本開始將支援Kafka服務。

建立Kafka叢集

在E-MapReduce控制台建立叢集時，選擇叢集類型為Kafka，則會建立一個預設只包含Kafka組件的叢集，除了基礎組件外包括Zookeeper，Kafka和KafkaManager三個組件。每個節點將只部署一個Kafka broker。我們建議您的Kafka叢集是一個專用叢集，不要和Hadoop相關服務混部在一起。

本地碟Kafka叢集

為了更好地降低單位成本以及應對更大的儲存需求，E-MapReduce將在EMR-3.5.1版本開始支援基於本地碟（D1類簇機型，詳情參考ECS機型介紹文檔）的Kafka叢集。相比較雲端硬碟，本地碟Kafka叢集有如下特點：

- 執行個體配備大容量、高吞吐SATA HDD本地碟，單磁碟 190 MB/s 順序讀寫效能，單一實例最大 5 GB/s 儲存吞吐能力。
- 本機存放區價格比 SSD 雲端硬碟降低 97%。
- 更高網路效能，最大17 Gbit/s執行個體間網路頻寬，滿足業務高峰期執行個體間資料互動需求。

但本地碟機型也有特殊之處：

操作	本地磁碟資料狀態	說明
作業系統重啟/控制台重啟/強制重啟	保留	本地磁碟儲存卷保留，資料保留
作業系統關機/控制台停止/強制停止	保留	本地磁碟儲存卷保留，資料保留
控制台上釋放（執行個體）	擦除	本地磁碟儲存卷擦除，資料不保留

注意：

- 當宿主機宕機或者磁碟損壞時，磁碟中的資料將會丟失。
- 請勿在本地磁碟上儲存需要長期儲存的業務資料，並及時做好資料備份和採用高可用架構。如需長期儲存，建議將資料存放區在雲端硬碟上。

為了能夠在本地碟上部署Kafka服務，E-MapReduce預設要求：

1. **default.replication.factor** = 3，即要求topic的分區副本數至少為3。如果設定更小副本數，則會增加資料丟失風險。
2. **min.insync.replicas** = 2，即要求當producer設定acks為all(-1)時，每次至少寫入兩個副本才算寫入成功。

當出現本地碟損壞時，E-MapReduce會進行：

1. 將壞盤從Broker的配置中剔除，重啟Broker，在其他正常可用的本地碟上恢複壞盤丟失的資料。根據壞盤上已經寫入的資料量不等，恢複的總時間也不等。
2. 當機器磁碟損壞數目過多（超過20%）是，E-MapReduce將主動進行機器遷移，恢複異常的磁碟。
3. 如果當前機器上可用剩餘磁碟空間不足以恢複壞盤遺失資料時，Broker將異常Down掉。這種情況，您可以選擇清理一些資料，騰出磁碟空間並重啟Broker服務；也可以聯絡E-MapReduce進行機器遷移，恢複異常的磁碟。

參數說明

您可以在E-MapReduce的叢集組態管理中查看Kafka的軟體配置，當前主要有：

配置項	說明
zookeeper.connect	Kafka叢集的Zookeeper串連地址
kafka.heap.opts	Kafka broker的堆記憶體大小
num.io.threads	Kafka broker的IO線程數，預設為機器CPU核心數目的2倍
num.network.threads	Kafka broker的網路線程數，預設為機器的CPU核心數目

3.2 Kafka Ranger使用說明

從 EMR-3.12.0 版本開始，E-MapReduce Kafka 支援用Ranger進行許可權配置。

Kafka整合Ranger

前面簡介中介紹了E-MapReduce中建立啟動Ranger服務的叢集，以及一些準備工作，本節介紹Kafka整合Ranger的一些步驟流程。

- **Enable Kakfa Plugin**

  1. 在**叢集組態管理**頁面的 **Ranger** 服務下，單擊右側的**操作**下拉式功能表，選擇 **Enable Kafka PLUGIN**。

  2. 單擊右上方的**查看操作曆史**查看任務進度，等待任務完成100%。

- **重啟Kafka broker**

  上述任務完成後，需要重啟broker才會生效。

  1. 在叢集組態管理頁面中，從 **Ranger** 切換到 **Kafka**。
  2. 點擊右上方 **Actions** 中的 **RESTART Broker**。
  3. 點擊右上方**查看操作歷史**查看任務進度，等待重啟任務完成。

- **Ranger UI頁面添加Kafka Service**

  參見Ranger簡介的介紹進入Ranger UI頁面。

  在 **Ranger** 的UI頁面添加Kafka Service：

  配置Kafka Service：

許可權配置樣本

上面一節中已經將Ranger整合到Kafka，現在可以進行相關的使用權限設定。

> **注意：**
> 標準叢集中，在添加了 Kafka Service 後，ranger 會預設建置規則 **all - topic**，不作任何許可權限制（即允許所有使用者進行所有操作），此時ranger無法通過使用者進行許可權識別。

以test使用者為例，添加Publish許可權：

按照上述步驟設定添加一個Policy後，就實現了對 **test** 的授權，然後使用者 **test** 就可以對 **test** 的topic進行寫入操作。

> **说明：**
> Policy添加後需要1分鐘左右才會生效。

3.3 Kafka SSL使用說明

從 EMR-3.12.0 版本開始，E-MapReduce Kafka支援SSL功能。

建立叢集

具體的建立叢集操作，請參見#unique_12。

開啟SSL服務

預設Kafka叢集沒有開啟SSL功能，您可以在Kafka服務的配置頁面開啟SSL。

如圖，修改配置項**kafka.ssl.enable**為true，部署配置並重啟組件。

用戶端訪問Kafka

用戶端通過SSL訪問Kafka時需要設定 **security.protocol**、**truststore** 和 **keystore** 的相關配置。以非安全叢集為例，如果是在kafka叢集運行作業，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叢集以外的環境運行作業，可將 Kafka 叢集中的 truststore 和 keystore 檔案（位於叢集任意一個節點的/etc/ecm/kafka-conf/目錄中）拷貝至運行環境作相應配置。

以Kafka內建的producer和consumer程式，在 Kafka 叢集運行為例：

1. 建立設定檔 ssl.properties， 添加配置項。

   ```
   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. 建立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設定檔產生資料。

   ```
   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設定檔消費資料。

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

3.4 Kafka Manager 使用說明

從EMR-3.4.0版本開始將支援Kafka Manager服務進行Kafka叢集管理。

操作步驟

> **注意:**
> 建立Kafka叢集時將預設安裝Kafka Manager軟體服務，並開啟Kafka Manager的認證功能。我們強烈建議您首次使用Kafka Manager時修改預設密碼，且使用SSH隧道方式來訪問。不建議您公網暴露8085連接埠，否則需要做好IP白名單保護，避免資料泄漏。

- 建議使用SSH隧道方式訪問Web介面，參考#unique_4。
- 訪問《http://localhost:8085》。
- 輸入使用者名密碼，請參考Kafka Manager的配置資訊。

- 添加一個建立好的Kafka叢集，需要注意配置正確Kafka叢集的Zookeeper地址，不清楚的可以看Kakfa的配置資訊。選擇對應的Kafka版本，另外建議開啟JMX功能。

- 建立好之後即可使用一些常見的Kakfa功能。

3.5 Kafka 常見問題

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

  常見原因:

  - Kafka服務異常，叢集Broker進程都退出了，需要結合日誌排查問題。
  - Kafka服務的Zookeeper地址使用錯誤，請注意查看並使用叢集組態管理中Kafka組件的Zookeeper串連地址。
- `java.net.BindException: Address already in use (Bind failed)`

  當您使用kafka命令列工具時，有時會碰到 `java.net.BindException: Address already in use (Bind failed)` 異常，這一般由JMX連接埠被佔用導致，您可以在命令列前手動指定一個JMX連接埠即可。例如:

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

4 Druid使用#明

4.1 Druid簡介

Druid是Metamarkets公司（一家為線上媒體或廣告公司提供資料分析服務的公司）推出的一個分布式記憶體即時分析系統，用於解決如何在大規模資料集下進行快速的、互動查詢和分析。

基本特點

Druid具有以下特點：

- 亞秒級OLAP查詢，包括多維過濾、ad-hoc的屬性分組、快速彙總資料等等。
- 即時的資料消費，真正做到資料攝入即時、查詢結果即時。
- 高效的多租戶能力，最高可以做到幾千使用者同時線上查詢。
- 擴充性強，支援PB級資料、千億級事件快速處理，支援每秒數千查詢並發。
- 極高的高可用保障，支援滾動升級。

應用情境

即時資料分析是Druid最典型的使用情境。該情境涵蓋的面很廣，例如：

- 即時指標監控
- 推薦模型
- 廣告平台
- 搜尋模型

這些情境的特點都是擁有大量的資料，且對資料查詢的時延要求非常高。在即時指標監控中，系統問題需要在出現的一刻被檢測到並被及時給出警示。在推薦模型中，使用者行為資料需要即時採集，並及時反饋到推薦系統中。使用者幾次點擊之後系統就能夠識別其搜尋意圖，並在之後的搜尋中推薦更合理的結果。

Druid架構

Druid擁有優秀的架構設計，多個組件協同工作，共同完成資料從攝取到索引、儲存、查詢等一系列流程。

下圖是Druid工作層（資料索引以及查詢）包含的組件。

- Realtime 組件負責資料的即時攝入。
- Broker 階段負責查詢任務的分發以及查詢結果的匯總，並將結果返回給使用者。

- Historical節點負責索引後的曆史資料的儲存，資料存放區在deep storage。deep storage可以是本地，也可以是HDFS等分散式檔案系統。
- Indexing service 包含兩個組件（圖中未畫出）。
  - Overlord 組件負責索引任務的管理、分發。
  - MiddleManager 負責索引任務的具體執行。

下圖是 Druid segments（Druid 索引檔案）管理層所涉及的組件。

- Zookeeper 負責儲存叢集的狀態以及作為服務發現組件，例如叢集的拓撲資訊，overlord leader 的選舉，indexing task 的管理等等。
- Coordinator 負責 segments 的管理，如 segments 下載、刪除以及如何在 historical 之間做均衡等等。
- Metadata storage 負責儲存 segments 的元資訊，以及管理叢集各種各樣的持久化或臨時性資料，比如配置資訊、審計資訊等等。

產品優勢

E-MapReduce Druid 基於開源 Druid 做了大量的改進，包括與E-MapReduce、阿里雲周邊生態的整合、方便的監控與營運支援、易用的產品介面等等，真正做到了即買即用和7*24免營運。

EMR Druid 目前支援的特性如下所示：

- 支援以OSS作為deep storage
- 支援將OSS檔案作為批量索引的資料來源
- 支援將中繼資料存放區到 RDS
- 整合了Superset工具
- 方便地擴容、縮容（縮容針對 task 節點）
- 豐富的監控指標和警示規則
- 壞節點遷移
- 支援高安全
- 支援HA

4.2 資料格式描述檔案

本小節簡要介紹一下索引資料的描述檔案——Ingestion Spec檔案，更為詳細的資訊請參考Druid官方文檔。

Ingestion Spec（資料格式描述）是Druid對要索引資料的格式以及如何索引該資料格式的一個統一描述，它是一個JSON檔案，一般由三部分組成：

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

鍵	格式	描述	是否必須
dataSchema	JSON 對象	描述所要消費資料的schema資訊。dataSchema是固定的，不隨資料消費方式改變	是
ioConfig	JSON 對象	描述所要消費資料的來源和消費去向。資料消費方式不同，ioConfig也不相同	是
tuningConfig	JSON 對象	調節資料消費時的一些參數。資料消費方式不同，可調節的參數也不相同	否

DataSchema

第一部分的dataSchema描述了資料的格式，如何解析該資料，典型結構如下：

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

鍵	格式	描述	是否必須
dataSource	字串	資料來源的名稱	是
parser	JSON 對象	資料的解析方式	是
metricsSpec	JSON 對象數組	彙總器（aggregator）列表	是
granularitySpec	JSON 對象	資料彙總設定，如建立segments，彙總粒度等等	是

- **parser**

  parser部分決定了您的資料如何被正確地解析，metricsSpec定義了資料如何被聚集計算，granularitySpec定義了資料分區的粒度、查詢的粒度。

  對於parser，type有兩個選項：string和hadoopString，後者用於Hadoop索引的job。parseSpec是資料格式解析的具體定義。

鍵	格式	描述	是否必須
type	字串	資料格式，可以是 “json”，“jsonLowercase”，“csv”，“tsv”幾種格式	是
timestampSpec	JSON 對象	時間戳記和時間戳記類型	是
dimensionsSpec	JSON 對象	資料的維度（包含哪些列）	是

對於不同的資料格式，可能還有額外的 parseSpec 選項。下面的表是 timestampSpec 和 dimensionsSpec 的描述：

鍵	格式	描述	是否必須
column	字串	時間戳記對應的列	是
format	字串	時間戳記類型，可選”iso”，“millis”，“posix”，“auto” 和 joda time 支援的類型	是

鍵	格式	描述	是否必須
dimensions	JSON 數組	描述資料包含哪些維度。每個維度可以只是個字串，或者可以額外指明維度屬性，比如 “dimensions”:[“dimenssion1”，“dimenssion2”，“{ “type”： “long”，“name”： “dimenssion3”}]，預設是 string 類型。	是
dimensionExclusions	JSON 字串數組	資料消費時要剔除的維度	否
spatialDimensions	JSON 對象數組	空間維度	否

- **metricsSpec**

  metricsSpec是一個JSON對象數組，定義了一些彙總器（aggregators）。彙總器通常有如下的結構：

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

  官方提供了以下常用的彙總器：

類型	type 可選
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：

  後三個屬於進階彙總器，您需要參见 Druid 官方文檔學習如何使用它們。

- **granularitySpec**

  彙總支援兩種彙總方式：uniform和 arbitrary，前者以一個固定的時間間隔彙總資料，後者盡量保證每個segments大小一致，時間間隔是不固定的。目前uniform是預設選項。

鍵	格式	描述	是否必須
segmentGranularity	字串	segments 粒度。uniform方式使用。	否，預設是”DAY”
queryGranularity	字串	可供查詢的最小資料彙總粒度	否
rollup	bool值	是否彙總	否，預設值為”true”

鍵	格式	描述	是否必須
intervals	字串	資料消費時間間隔	對於batch是，對於realtime否

ioConfig

第二部分ioConfig描述了資料來源。以下是一個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"
  }
}
```

對於通過Tranquility處理的流式資料，這部分是不需要的。

Tunning Config

Tuning Config是指一些額外的設定。比如Hadoop對批量資料建立索引，可以在這裡指定一些MapReduce參數。Tunning Config的內容依賴於您的資料來源可能有不同的內容。

4.3 Superset

Druid叢集整合了Superset工具。Superset對Druid做了深度整合，同時也支援多種關係型資料庫。由於Druid也支援SQL，所以可以通過Superset以兩種方式訪問Druid，即Druid原生查詢語言或者SQL。

Superset預設安裝在emr-header-1節點，目前還不支援HA。在使用該工具前，確保您的主機能夠正常訪問emr-header-1。您可以通過打 SSH 隧道 的方式串連到主機。

1. 登入Superset。

   在瀏覽器地址欄中輸入 http://emr-header-1:18088 後斷行符號，開啟Superset登入介面，預設使用者名/密碼為 admin/admin，請您登入後及時修改密碼。

2. 添加 Druid 叢集。

   登入後預設為英文介面，可點擊右上方的國旗表徵圖選擇合適的語言。接下來在上方功能表列中依次選擇**資料來源** > **Druid 叢集**來添加一個 Druid 叢集。

   配置好協調機（Coordinator）和代理機（Broker）的地址，注意 E-MapReduce 中預設連接埠均為相應的開源連接埠前加數字1，例如開源 Broker 連接埠為 8082，E-MapReduce 中為18082。

3. 重新整理或者添加新資料來源。

   添加好 Druid 叢集之後，您可以單擊**資料來源** > **掃描新的資料來源**，這時 Druid 叢集上的資料來源（datasource）就可以自動被載入進來。

   您也可以在介面上單擊**資料來源** > **Druid 資料來源**自訂新的資料來源（其操作等同於寫一個 data source ingestion 的 json 檔案），步驟如下：

   自訂資料來源時需要填寫必要的資訊，然後儲存。

   儲存之後點擊左側三個小表徵圖中的第二個，編輯該資料來源，填寫相應的維度列與指標列等資訊。

4. 查詢 Druid。

   資料來源添加成功後，單擊資料來源名稱，進入查詢頁面進行查詢。

5. （可選）將 Druid 作為 資料庫使用。

   Superset 提供了 SQLAlchemy 以多種方言支援各種各樣的資料庫，其支援的資料庫類型如下表所示。

   Superset 亦支援該方式訪問 Druid，Druid 對應的 SQLAlchemy URI 為 “druid://emr-header-1:18082/druid/v2/sql”，如下圖所示，將 Druid 作為一個資料庫添加：

   接下來就可以在 SQL 工具箱裡用 SQL 進行查詢了：

4.4 常見問題

索引失敗問題分析思路

當發現索引失敗時，一般遵循如下排錯思路：

- **對於批量索引**

  1. 如果 curl 直接返回錯誤，或者不返回，檢查一下輸入檔案格式。或者 curl 加上 **-v** 參數，觀察 REST API的返回情況。
  2. 在 Overlord 頁面觀察作業執行情況，如果失敗，查看頁面上的 logs。
  3. 在很多情況下並沒有產生 logs。如果是 Hadoop 作業，開啟 YARN 頁面查看是否有索引作業產生，並查看作業執行 log。
  4. 如果上述情況都沒有定位到錯誤，需要登入到 Druid 叢集，查看 overlord 的執行日誌（位於 /mnt/disk1/log/druid/overlord—emr-header-1.cluster-xxxx.log），如果是 HA 叢集，查看您提交作業的那個 Overlord。
  5. 如果作業已經被提交到 Middlemanager，但是從 Middlemanager 返回了失敗，則需要從 Overlord 中查看作業提交到了那個 worker，並登入到相應的 worker，查看 Middlemanager 的日誌（位於/mnt/disk1/log/druid/middleManager-emr-header-1.cluster-xxxx.log）。

- **對於 Tranquility 即時索引**

  查看 Tranquility log，查看訊息是否被接收到了或者是否被丟棄（drop）掉了。

  其餘的排查步驟同批量索引的 2 ~ 5。

  錯誤多數情況為叢集配置問題和作業問題。叢集配置問題包括：記憶體參數是否合理、跨叢集聯通性是否正確、安全叢集訪問是否通過、principal 是否正確等等，作業問題包括作業描述檔案格式正確、輸入資料是否能夠正常被解析，以及一些其他的作業相關的配置（如ioConfig等）。

常見問題列表

- 組件啟動失敗

  此類問題多數是由於組件 JVM 運行參數配置問題，例如機器可能沒有很大的記憶體，而配置了較大的 JVM 記憶體或者較多的線程數量。

  解決方案：查看組件日誌並調整相關參數即可解決。JVM 記憶體涉及堆記憶體和直接記憶體。具體可參考http://druid.io/docs/latest/operations/performance-faq.html。

- 索引時 YARN task 執行失敗，顯示諸如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; 之類的 jar 包衝突錯誤。

  解決方案：在 indexing 的作業設定檔中加入如下配置：

  ```
  "tuningConfig" : {
     ...
     "jobProperties" : {
  ```

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

其中參數 **mapreduce.job.classloader** 讓MR job用獨立的 classloader，**mapreduce.job.user.classpath.first** 是讓MapReduce優先使用使用者的jar包，兩個配置項配置一個即可。 可參考：http://druid.io/docs/0.9.2-rc1/operations/other-hadoop.html。

- indexing作業的日誌中報reduce無法建立segments目錄

  解決方案：

  - 注意檢查 deep storage 的設定，包括 type 和 directory。當 type 為 local 時，注意 directory 的使用權限設定。當 type 為 HDFS 時，directory 盡量用完整的 HDFS 路徑寫法，如 hdfs://:9000/。hdfs_master 最好用 IP，如果用網域名稱，要用完整網域名稱，如 emr-header-1.cluster-xxxxxxxx，而不是 emr-header-1。
  - 用 Hadoop 批量索引時，要將 segments 的 deep storage 設定為 “hdfs”，“local” 的方式會導致 MR 作業處於 UNDEFINED 狀態，這是因為遠端 YARN 叢集無法在 reduce task 下建立 local 的 segments 目錄。（此針對獨立 Druid 叢集）。

- Failed to create directory within 10000 attempts...

  此問題一般為 JVM 設定檔中 java.io.tmp 設定的路徑不存在。設定該路徑並確保 Druid 賬戶有許可權訪問即可。

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

  此問題一般是 ZooKeeper 的串連問題。確保 Druid 與 Tranquility 對於 ZooKeeper 有相同的連接字串。注意：Druid 預設的ZooKeeper路徑為 /druid，因此確保 Tranquility 設定中 zookeeper.connect 包含路徑 /druid。（另注意 Tranquility Kafka 設定中有兩個 ZooKeeper 的設定，一個為 zookeeper.connect，串連的 Druid 叢集的 ZooKeeper，一個為 kafka.zookeeper.connect，串連的 Kafka 叢集的 ZooKeeper。這兩個 ZooKeeper 可能不是一個 ZooKeeper 叢集）。

- 索引時 MiddleManager 報找不到類 com.hadoop.compression.lzo.LzoCodec

  這是因為 EMR 的 Hadoop 叢集配置了 lzo 壓縮。

  解決方案：拷貝 EMR HADOOP_HOME/lib 下的 jar 包和 native 檔案夾到 Druid 的 druid.extensions.hadoopDependenciesDir（預設為 DRUID_HOME/hadoop-dependencies）

- 索引時報如下錯誤：

```
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:?]
```

這個問題還是因為 java.io.tmp 路徑不存在。設定該路徑並確保 Druid 賬戶有許可權訪問。

5 Presto

5.1 產品簡介

Presto是一款由FaceBook開源的一個分布式SQL-on—Hadoop分析引擎。Presto目前由開源社區和FaceBook內部工程師共同維護，並衍生出多個商業版本。

基本特性

Presto使用Java語言進行開發，具備易用、高效能、強擴充能力等特點，具體的：

- 完全支援ANSI SQL。
- 支援豐富的資料來源 Presto可接入豐富的資料來源，如下所示：
  - 與Hive數倉互操作
  - Cassandra
  - Kafka
  - MongoDB
  - MySQL
  - PostgreSQL
  - SQL Server
  - Redis
  - Redshift
  - 本地檔案
- 支援進階資料結構：
  - 支援數組和Map資料。
  - 支援JSON資料。
  - 支援GIS資料。
  - 支援顏色資料。
- 功能擴充能力強 Presto提供了多種擴充機制，包括：
  - 擴充資料連線器。
  - 自訂資料類型。
  - 自訂SQL函數。

  使用者可以根據自身業務特點擴充相應的模組，實現高效的業務處理。

- 基於Pipeline處理模型 資料在處理過程中即時返回給使用者。
- 監控介面完善：
  - 提供友好的WebUI，可視化的呈現查詢任務執行過程。
  - 支援JMX協議。

應用情境

Presto是定位在資料倉儲和資料分析業務的分布式SQL引擎，比較適合如下幾個應用情境：

- ETL
- Ad-Hoc查詢
- 海量結構化資料/半結構化資料分析
- 海量多維資料彙總/報表

特別需要注意的是，Presto是一個數倉類產品，其設計目標並不是為了替代MySQL、PostgreSQL等傳統的RDBMS資料庫，對事務對支援有限，不適合線上業務情境。

產品優勢

EMR Presto產品除了開源Presto本身具有的優點外，還具備如下優勢：

- 即買即用 分分鐘完成上百節點的Presto叢集搭建。
- 彈性擴容 簡單操作即可完成叢集的擴容和縮容。
- 與EMR軟體棧完美結合，支援處理儲存在OSS的資料。
- 免營運 7*24一站式服務。

6 Ranger

6.1 Ranger簡介

Apache Ranger提供集中式的許可權管理架構，可以對Hadoop生態中的HDFS/Hive/YARN/Kafka/Storm/Solr等組件進行細粒度的許可權存取控制，並且提供了Web UI方便管理員進行操作。

建立叢集

在E-MapReduce控制台建立EMR-2.9.2/EMR-3.9.0及以上的叢集，勾選Ranger組件即可, 如下圖所示:

如果已經建立EMR-2.9.2/EMR-3.9.0及以上的叢集，可以通過叢集的組態管理頁面添加Ranger服務:

Ranger UI

在安裝了Ranger的叢集後，單擊**組態管理**，然後在左側導覽列中選擇**訪問連結與連接埠**，可以通過快捷連結訪問Ranger UI，如下圖所示:

單擊連結後會進入Ranger UI登入介面，預設的使用者名/密碼是admin/admin，如下圖:

修改密碼

管理員首次登入後，需要修改admin的密碼，如下圖:

改完admin密碼後，單擊右上方**admin**下拉式功能表的**Log Out**，然後使用新的密碼登入即可。

組件整合Ranger

經過上述步驟，可以將叢集中的相關組件整合到Ranger，進行相關許可權的控制，詳情請參見:

- HDFS整合Ranger
- Hive整合Ranger
- HBase整合Ranger

7 #件授#

7.1 HDFS授權

HDFS開啟了許可權控制後，使用者訪問HDFS需要有合法的許可權才能正常操作HDFS，如讀取資料/建立檔案夾等。

添加配置

HDFS許可權相關的配置如下:

- dfs.permissions.enabled

  開啟許可權檢查，即使該值為false， chmod/chgrp/chown/setfacl操作還是會進行許可權檢查。

- dfs.datanode.data.dir.perm

  datanode使用的本地檔案夾路徑的許可權，預設755。

- fs.permissions.umask-mode
  - 許可權掩碼，在建立檔案/檔案夾的時候的預設使用權限設定。
  - 建立檔案: 0666 & ^umask。
  - 建立檔案夾: 0777 & ^umask。
  - 預設umask值為022，即建立檔案許可權為644(666&^022=644)，建立檔案夾許可權為755(777&^022=755)。
  - EMR的Kerberos安全叢集預設設定為027，對應建立檔案許可權為640，建立檔案夾許可權為750。
- dfs.namenode.acls.enabled
  - 開啟ACL控制，開啟後除了可以對owner/group進行許可權控制外，還可以對其它使用者進行設定。
  - 設定ACL相關命令:

    ```
    hadoop fs -getfacl [-R] <path>
    hadoop fs -setfacl [-R] [-b |-k -m |-x <acl_spec> <path>] |[--set <acl_spec>   <path>]
    ```

    如:

    ```
    su test
     #test使用者建立檔案夾
     hadoop fs -mkdir /tmp/test
     #查看建立的檔案夾的許可權
     hadoop fs -ls /tmp
    ```

```
drwxr-x---   - test   hadoop          0 2017-11-26 21:18 /tmp/test
#設定acl,授權給foo使用者rwx
hadoop fs -setfacl -m user:foo:rwx /tmp/test
#查看檔案許可權(+號表示設定了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

  超級使用者組，屬於該組的使用者都具有超級使用者的許可權

重啟HDFS服務

對於Kerberos安全叢集，已經預設設定了HDFS的許可權(umask設定為027)，無需配置和重啟服務。

對於非Kerberos安全叢集需要添加配置並重啟服務。

其它

- umask值可以根據需求自行修改
- HDFS是一個基礎的服務，Hive/HBase等都是基於HDFS，所以在配置其它上層服務時，需要提前配置好HDFS的許可權控制。
- 在HDFS開啟許可權後，需要設定好服務的(如spark的/spark-history、yarn的/tmp/$user/等)。
- sticky bit:

  針對檔案夾可設定sticky bit，可以防止除了superuser/file owner/dir owner之外的其它使用者刪除該檔案夾中的檔案/檔案夾(即使其它使用者對該檔案夾有rwx許可權)。如:

```
#即在第一位添加數字1
hadoop fs -chmod 1777 /tmp
hadoop fs -chmod 1777 /spark-history
hadoop fs -chmod 1777 /user/hive/warehouse
```

7.2 YARN授權

YARN的授權根據授權實體，可以分為服務等級的授權、隊列層級的授權。

服務等級的授權

詳見Hadoop官方文檔。

- 控制特定使用者訪問叢集服務，如提交作業。
- 配置在hadoop-policy.xml。
- 服務等級的許可權校正在其他許可權校正之前(如HDFS的permission檢查/yarn提交作業到隊列控制)。

说明:

一般設定了HDFS permission檢查/yarn隊列資源控制，可以不設定服務等級的授權控制，使用者可以根據自己需求進行相關配置。

隊列層級的授權

YARN可以通過隊列對資源進行授權管理，有兩種隊列調度 Capacity Scheduler和Fair Scheduler。

這裡以Capacity Scheduler為例。

- 添加配置

  隊列也有兩個層級的授權，一個是提交作業到隊列的授權，一個是管理隊列的授權。

  

  说明:

  - 隊列的ACL的控制對象為user/group，設定相關參數時，user和group可同時設定，中間用空格分開，user/group內部可用逗號分開，只有一個空格表示任何人都沒有許可權。
  - 隊列ACL繼承: 如果一個user/group可以向某個隊列中提交應用程式，則它可以向它的所有子隊列中提交應用程式，同理管理隊列的ACL也具有繼承性。所以如果要防止某個user/group提交作業到某個隊列，則需要設定該隊列以及該隊列的所有父隊列的ACL來限制該user/group的提交作業的許可權。

  - yarn.acl.enable

    ACL開關，設定為true

  - yarn.admin.acl

    ■ yarn的管理員設定，如可執行yarn rmadmin/yarn kill等命令，該值必須配置，否則後續的隊列相關的acl管理員設定無法生效。

    ■ 如上備忘，配置值時可以設定user/group：

    ```
    user1,user2 group1,group2 #user和group用空格隔開
    ```

```
group1,group2 #只有group情況下，必須在最前面加上空格
```

EMR叢集中需將has配置為admin的acl許可權

- yarn.scheduler.capacity.${queue-name}.acl_submit_applications
  - 設定能夠向該隊列提交的user/group。
  - 其中${queue-name}為隊列的名稱，可以是多級隊列，注意多級情況下的ACL繼承機制。

  如:

```
#queue-name=root
 <property>
    <name>yarn.scheduler.capacity.root.acl_submit_applications</name>
    <value> </value> #空格表示任何人都無法往root隊列提交作業
 </property>
#queue-name=root.testqueue
<property>
  <name>yarn.scheduler.capacity.root.testqueue.acl_submit_applications</
name>
    <value>test testgrp</value> #testqueue只允許test使用者/testgrp組提交作業
 </property>
```

- yarn.scheduler.capacity.${queue-name}.acl_administer_queue
  - 設定某些user/group管理隊列，比如kill隊列中作業等。
  - queue-name可以是多級，注意多級情況下的ACL繼承機制。

```
#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_administer_queue</name
>
    <value>test testgrp</value>
 </property>
```

- 重啟YARN服務
  - 對於Kerberos安全叢集已經預設開啟ACL，使用者可以根據自己需求配置隊列的相關ACL許可權控制。
  - 對於非Kerberos安全叢集根據上述開啟ACL並配置好隊列的許可權控制，重啟YARN服務。
- 配置樣本
  - 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隊列: 禁用default隊列，不允許任何使用者提交或管理。
- q1隊列: 只允許test使用者提交作業以及管理隊列(如kill)。
- q2隊列: 只允許foo使用者提交作業以及管理隊列。

```
<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.DefaultResourceCalculator</value>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.queues</name>
    <value>default,q1,q2</value>
    <!-- 3個隊列-->
    <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隊列狀態設定為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隊列禁止提交作業-->
    <description>The ACL of who can submit jobs to the default queue.</description>
  </property>
```

```
  <property>
    <name>yarn.scheduler.capacity.root.default.acl_administer_queue</name>
    <value> </value>
    <!-- 禁止管理default隊列-->
    <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>
    <!-- 隊列映射，test使用者自動對應到q1隊列-->
    <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設定的映射，是否覆蓋用戶端設定的隊列參數-->
    <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繼承性，父隊列需控制住許可權-->
    <description>
      The ACL of who can submit jobs to the root queue.
    </description>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.q1.acl_submit_applications</name>
    <value>test</value>
    <!-- q1隻允許test使用者提交作業-->
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.q2.acl_submit_applications</name>
    <value>foo</value>
    <!-- q2隻允許foo使用者提交作業-->
  </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繼承性，父隊列需控制住許可權-->
    </property>
    <property>
      <name>yarn.scheduler.capacity.root.q1.acl_administer_queue</name>
      <value>test</value>
      <!-- q1隊列只允許test使用者管理，如kill作業-->
    </property>
    <property>
      <name>yarn.scheduler.capacity.root.q2.acl_administer_queue</name>
      <value>foo</value>
      <!-- q2隊列只允許foo使用者管理，如kill作業-->
    </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>
```

7.3 Hive授權

Hive內建有兩種授權機制：

- 基於底層HDFS的許可權(Storage Based Authorization)
- 基於標準SQL的grant等命令(SQL Standards Based Authorization)

詳見Hive官方文檔

说明：

兩種授權機制可以同時配置，不衝突。

Storage Based Authorization(針對HiveMetaStore)

情境：

如果叢集的使用者可以直接通過HDFS/Hive Client訪問Hive的資料，需要對Hive在HDFS中的資料進行相關的許可權控制，通過HDFS許可權控制，進而可以控制Hive SQL相關的操作許可權。

詳見Hive文檔

添加配置

在叢集的組態管理頁面選擇**Hive** > **配置** > **hive-site.xml** > **添加自訂配置**

```
<property>
<name>hive.metastore.pre.event.listeners</name>
   <value>org.apache.hadoop.hive.ql.security.authorization.AuthorizationPreEven
tListener</value>
</property>
<property>
<name>hive.security.metastore.authorization.manager</name>
   <value>org.apache.hadoop.hive.ql.security.authorization.StorageBasedAuthoriz
ationProvider</value>
</property>
<property>
<name>hive.security.metastore.authenticator.manager</name>
   <value>org.apache.hadoop.hive.ql.security.HadoopDefaultMetastoreAuthenticator</
value>
</property>
```

重啟HiveMetaStore

在叢集組態管理頁面重啟HiveMetaStore

HDFS許可權控制

EMR的Kerberos安全叢集已經設定了Hive的warehouse的HDFS相關許可權；

對於非Kerberos安全叢集，使用者需要做如下步驟設定hive基本的HDFS許可權:

- 開啟HDFS的許可權
- 配置Hive的warehouse許可權

```
hadoop fs -chmod 1771 /user/hive/warehouse
也可以設定成，1表示stick bit(不能刪除別人建立的檔案/檔案夾)
hadoop fs -chmod 1777 /user/hive/warehouse
```

有了上述設定基礎許可權後，可以通過對warehouse檔案夾授權，讓相關使用者/使用者組能夠正常建立表/讀寫表等

```
sudo su has
    #授予test對warehouse檔案夾rwx許可權
    hadoop fs -setfacl -m user:test:rwx /user/hive/warehouse
    #授予hivegrp對warehouse檔案夾rwx許可權
    hadoo fs -setfacl -m group:hivegrp:rwx /user/hive/warehouse
```

經過HDFS的授權後，讓相關使用者/使用者組能夠正常建立表/讀寫表等，而且 不同的帳號建立的hive表在HDFS中的資料只能自己的帳號才能訪問。

驗證

- test使用者建表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.AccessCont
rolException Permission denied: user=test, access=WRITE, inode="/user/hive/
warehouse/testtbl":hadoop:hadoop:drwxrwx--t
at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissi
onChecker.java:320)
at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissi
onChecker.java:292)
```

上面顯示錯誤沒有許可權，需要給test使用者添加許可權

```
#從root帳號切到has帳號
su has
#給test帳號添加acl，對warehouse目錄的rwx許可權
 hadoop fs -setfacl -m user:test:rwx /user/hive/warehouse
```

test帳號再建立database，成功

```
hive> create table testtbl(a string);
OK
Time taken: 1.371 seconds
#查看hdfs中testtbl的目錄，從許可權可以看出test使用者建立的表資料只有test和hadoop組
可以讀取，其他使用者沒有任何許可權
hadoop fs -ls /user/hive/warehouse
drwxr-x---   - test hadoop          0 2017-11-25 14:51 /user/hive/warehouse/testtbl
#插入一條資料
hive>insert into table testtbl select "hz"
```

- foo使用者訪問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.checkPermi
ssion(FSPermissionChecker.java:219)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermi
ssion(FSPermissionChecker.java:190)
#alter table
hive> alter table testtbl add columns(b string);
FAILED: SemanticException Unable to fetch table testtbl. java.security.AccessCont
rolException: 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.checkPermi
ssion(FSPermissionChecker.java:219)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermi
ssion(FSPermissionChecker.java:190)
    at org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPermission(
FSDirectory.java:1720)
#select
hive> select * from testtbl;
FAILED: SemanticException Unable to fetch table testtbl. java.security.AccessCont
rolException: 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.checkPermi
ssion(FSPermissionChecker.java:219)
```

可見foo使用者不能對test使用者建立的表做任何的操作，如果想要授權給foo，需要通過HDFS的授權來實現

```
su has
#只授權讀的許可權，也可以根據情況授權寫入權限(比如alter)
#備忘: -R 將testtbl檔案夾下的檔案也設定可讀
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:

一般可以根據需求建立一個hive使用者的group，然後通過給group授權，後續將新使用者添加到group中，同一個group的資料許可權都可以訪問。

SQL Standards Based Authorization

- 情境

  如果叢集的使用者不能直接通過hdfs/hive client訪問，只能通過HiveServer(beeline/jdbc等)來執行hive相關的命令，可以使用 SQL Standards Based Authorization的授權方式。

  如果使用者能夠使用hive shell等方式，即使做下面的一些設定作業，只要使用者用戶端的hive-site.xml沒有相關配置，都可以正常訪問hive。

  詳見Hive文檔

- 添加配置

  - 配置是提供給HiveServer
  - 在叢集的組態管理頁面選擇**Hive** > **配置** > **hive-site.xml** > **添加自訂配置**

  ```
  <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>
  ```

- 重啟HiveServer2

  在叢集組態管理頁面重啟HiveServer2

- 許可權操作命令

  具體命令操作詳見

- 驗證

  - 使用者foo通過beeline訪問test使用者的testtbl表

    ```
    2: jdbc:hive2://emr-header-1.cluster-xxx:10> select * from testtbl;
    Error: Error while compiling statement: FAILED: HiveAccessControlException
    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許可權

    ```
    切到test帳號執行grant給foo授權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)
    ```

  - 回收許可權

    ```
    切換到test帳號，回收許可權foo的select許可權
      hive> revoke select from user foo;
        OK
        Time taken: 1.094 seconds
    ```

  - foo無法select testtbl的資料

    ```
    0: jdbc:hive2://emr-header-1.cluster-xxxxx:10> select * from testtbl;
    Error: Error while compiling statement: FAILED: HiveAccessControlException
    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)
    ```

7.4 HBase授權

HBase在不開啟授權的情況下，任何帳號對HBase叢集可以進行任何操作，比如disable table/drop table/major compact等等。

说明:

對於沒有Kerberos認證的叢集，即使開啟了HBase授權，使用者也可以偽造身份訪問叢集服務。所以建議建立高安全模式(即支援Kerberos)的叢集，詳見Kerberos簡介。

添加配置

在HBase叢集的組態管理頁面選擇**HBase** > **配置** > **hbase-site** > **自訂配置**

添加如下幾個參數:

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

重啟HBase叢集

在HBase叢集的組態管理頁面選擇**HBase** > **操作** > **RESTART All Components**。

授權(ACL)

- 基本概念

  授權就是將對 [某個範圍的資源] 的 [操作許可權] 授予[某個實體]。

  在HBase中，上述對應的三個概念分別為:

  - 某個範圍(Scope)的資源
    - Superuser

      超級帳號可以進行任何操作，運行HBase服務的帳號預設是Superuser。也可以通過在hbase-site.xml中配置hbase.superuser的值可以添加超級帳號。
    - Global

      Global Scope擁有叢集所有table的Admin許可權。
    - Namespace

      在Namespace Scope進行相關許可權控制。
    - Table

      在Table Scope進行相關許可權控制。
    - ColumnFamily

      在ColumnFamily Scope進行相關許可權控制。
    - Cell

      在Cell Scope進行相關許可權控制。
  - 操作許可權
    - Read (R)

      讀取某個Scope資源的資料。
    - Write (W)

      寫資料到某個Scope的資源。
    - Execute (X)

      在某個Scope執行副處理器。
    - Create (C)

      在某個Scope建立/刪除表等操作。
    - Admin (A)

      在某個Scope進行叢集相關操作，如balance/assign等。

- 某個實體
  - User

    對某個使用者授權。
  - Group

    對某個使用者組授權。

- 授權命令
  - grant 授權

    ```
    grant <user> <permissions> [<@namespace> [<table> [<column family> [<column qualifier>]]]
    ```

    说明:

    - user/group的授權方式一樣，group需要加一個首碼@

      ```
      grant 'test','R','tbl1'  #給使用者test授予表tbl1的讀許可權
       grant '@testgrp','R','tbl1' #給使用者組testgrp授予表tbl1的讀許可權
      ```

    - namespace需要加一個首碼@

      ```
      grant 'test 'C','@ns_1'  #給使用者test授予namespace ns_1的CREATE許可權
      ```

  - revoke 回收
  - user_permissions 查看許可權

7.5 Kafka授權

Kafka授權

如果沒有開啟Kafka認證(如Kerberos認證或者簡單的使用者名密碼)，即使開啟了Kafka授權，使用者也可以偽造身份訪問服務。所以建議建立高安全模式(即支援Kerberos)的Kafka叢集。

说明:

本文的許可權配置只針對E-MapReduce的高安全模式叢集，即Kafka以Kerberos的方式啟動，詳見Kerberos簡介。

添加配置

1. 在**叢集列表**頁面，在需要添加配置的Kafka叢集後單擊**詳情**。
2. 在左側導覽列中單擊**叢集與服務管理**頁簽，然後單擊服務列表中的**Kafka**。
3. 在上方單擊**配置**頁簽。

4. 在**服務配置**列表的右上方單擊自訂配置，添加如下幾個參數：

key	value	備忘
authorizer.class.name	kafka.security.auth.SimpleAclAuthorizer	無
super.users	User:kafka	User:kafka是必須的，可添加其它使用者用分號(;)隔開

说明：

zookeeper.set.acl用來設定Kafka在ZooKeeper中資料的許可權，E-MapReduce叢集中已經設定為true，所以此處不需要再添加該配置。該配置設定為true後，在Kerberos環境中，只有使用者名稱為kafka且通過Kerberos認證後才能執行kafka-topics.sh命令(kafka-topics.sh會直接讀寫/修改ZooKeeper中的資料)。

重啟Kafka叢集

1. 在**叢集列表**頁面，在需要添加配置的Kafka叢集後單擊**詳情**。
2. 在左側導覽列中單擊**叢集與服務管理**頁簽，然後單擊服務列表中的Kafka右側的操作。
3. 在下拉式功能表中選擇**RESTART All Components**，輸入記錄資訊後單擊**確定**。

授權(ACL)

- 基本概念

  Kafka官方文檔定義：

  > Kafka acls are defined in the general format of "Principal P is [Allowed/Denied] Operation O From Host H On Resource R"

  即ACL過程涉及Principal、Allowed/Denied、Operation、Host和Resource。

  - Principal：使用者名

安全性通訊協定	value
PLAINTEXT	ANONYMOUS
SSL	ANONYMOUS
SASL_PLAINTEXT	mechanism為PLAIN時，使用者名是client_jaas.conf指定的使用者名，mechanism為GSSAPI時，使用者名為client_jaas.conf指定的principal

安全性通訊協定	value
SASL_SSL	

- Allowed/Denied：允許/拒絕。
- Operation： 操作，包括Read、Write、Create、DeleteAlter、Describe、ClusterAction、AlterConfigs、DescribeConfigs、IdempotentWrite和All。
- Host： 針對的機器。
- Resource： 許可權作用的資來源物件，包括Topic、Group、Cluster和TransactionalId。

Operation/Resource的一些詳細對應關係，如哪些Resource支援哪些Operation的授權，詳見KIP-11 - Authorization Interface。

- 授權命令

我們使用指令碼kafka-acls.sh (/usr/lib/kafka-current/bin/kafka-acls.sh) 進行Kafka授權，您可以直接執行 kafka-acls.sh --help查看如何使用該命令。

操作樣本

在已經建立的E-MapReduce高安全Kafka叢集的master節點上進行相關樣本操作。

1. 建立使用者test，執行以下命令。

   useradd test

2. 建立topic。

   第一節添加配置的備忘中提到zookeeper.set.acl=true，kafka-topics.sh需要在kafka帳號下執行，而且kafka帳號下要通過Kerberos認證。

   ```
   # kafka_client_jaas.conf中已經設定了kafka的Kerberos認證相關資訊
   export KAFKA_HEAP_OPTS="-Djava.security.auth.login.config=/etc/ecm/kafka-conf/kafka_client_jaas.conf"
   # zookeeper地址改成自己叢集的對應地址(執行`hostnamed`後即可擷取)
   kafka-topics.sh --create --zookeeper emr-header-1:2181/kafka-1.0.0 --replication-factor 3 --partitions 1 --topic test
   ```

3. test使用者執行kafka-console-producer.sh。

   a. 建立test使用者的keytab檔案，用於zookeeper/kafka的認證。

   ```
   su root
   sh /usr/lib/has-current/bin/hadmin-local.sh /etc/ecm/has-conf -k /etc/ecm/has-conf/admin.keytab
   HadminLocalTool.local: #直接按斷行符號可以看到一些命令的用法
   HadminLocalTool.local: addprinc #輸入命令按斷行符號可以看到具體命令的用法
   HadminLocalTool.local: addprinc -pw 123456 test #添加test的principal,密碼設定為123456
   ```

```
HadminLocalTool.local: ktadd -k /home/test/test.keytab test #匯出keytab檔案，後續可使用該檔案
```

b. 添加kafka_client_test.conf。

如檔案放到/home/test/kafka_client_test.conf，內容如下:

```
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。

如檔案放到/home/test/producer.conf，內容如下:

```
security.protocol=SASL_PLAINTEXT
sasl.mechanism=GSSAPI
```

d. 執行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
```

由於沒有設定ACL，所以上述會報錯:

```
org.apache.kafka.common.errors.TopicAuthorizationException: Not authorized to access topics: [test]
```

e. 設定ACL。

同樣kafka-acls.sh也需要kafka帳號執行。

```
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. 再執行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
```

正常情況下顯示如下:

```
[2018-02-28 22:25:36,178] INFO Kafka commitId : aaa7af6d4a11b29d (org.apache.
kafka.common.utils.AppInfoParser)
>alibaba
>E-MapReduce
>
```

4. test使用者執行kafka-console-consumer.sh。

上面成功執行kafka-console-producer.sh,並往topic裡面寫入一些資料後，就可以執行kafka-console-consumer.sh進行消費測試.

a. 添加consumer.conf。

如檔案放到/home/test/consumer.conf，內容如下:

```
security.protocol=SASL_PLAINTEXT
sasl.mechanism=GSSAPI
```

b. 執行kafka-console-consumer.sh。

```
su test
#kafka_client_test.conf跟上面producer使用的是一樣的
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
```

由於未設定許可權，會報錯:

```
org.apache.kafka.common.errors.GroupAuthorizationException: Not authorized to
access group: test-group
```

c. 設定ACL。

```
su kafka
export KAFKA_HEAP_OPTS="-Djava.security.auth.login.config=/etc/ecm/kafka-conf
/kafka_client_jaas.conf"
# test-group許可權
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許可權
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. 再執行kafka-console-consumer.sh。

```
su test
# kafka_client_test.conf跟上面producer使用的是一樣的
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
```

正常輸出:

```
alibaba
E-MapReduce
```

8 Kerberos##

8.1 Kerberos簡介

E-MapReduce從EMR-2.7.x/EMR-3.5.x版本開始支援建立安全類型的叢集，即叢集中的開源組件以Kerberos的安全模式啟動,在這種安全環境下只有經過認證的用戶端(Client)才能訪問叢集的服務(Service，如HDFS)。

前置

目前E-MapReduce版本中支援的Kerberos的組件列表如下所示：

組件名稱	組件版本
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目前版本不支援Kerberos。

建立安全叢集

在叢集建立頁面的軟體配置下開啟**安全**按鈕即可，如下所示:

Kerberos身份認證原理

Kerberos是一種基於對稱金鑰技術的身份認證協議，它作為一個獨立的第三方的身份認證服務，可以為其它服務提供身份認證功能，且支援SSO(即用戶端身份認證後，可以訪問多個服務如HBase/HDFS等)。

Kerberos協議過程主要有兩個階段，第一個階段是KDC對Client身份認證，第二個階段是Service對Client身份認證。

- KDC

  Kerberos的服務端程式

- Client

  需要訪問服務的使用者(principal)，KDC和Service會對使用者的身份進行認證

- Service

  整合了Kerberos的服務，如HDFS/YARN/HBase等

- KDC對Client身份認證

  當用戶端使用者(principal)訪問一個整合了Kerberos的服務之前，需要先通過KDC的身份認證。

  若身份認證通過則用戶端會拿到一個TGT(Ticket Granting Ticket)，後續就可以拿該TGT去訪問整合了Kerberos的服務。

- Service對Client身份認證

  當2.1中使用者拿到TGT後，就可以繼續訪問Service服務。它會使用TGT以及需要訪問的服務名稱(如HDFS)去KDC擷取SGT(Service Granting Ticket)，然後使用SGT去訪問Service，Service會利用相關資訊對Client進行身份認證，認證通過後就可以正常訪問Service服務。

EMR實踐

EMR的Kerberos安全叢集中的服務在建立叢集的時候會以Kerberos安全模式啟動。

- Kerberos服務端程式為HasServer
  - 登入EMR 控制台，選擇**叢集** > **組態管理** > **HAS**，執行查看/修改配置/重啟等操作。
  - 非HA叢集部署在emr-header-1,HA叢集部署在emr-header-1/emr-header-2兩個節點
- 支援四種身份認證方式

  HasServer可同時支援以下4種身份認證方式，用戶端可以通過配置相關參數來指定HasServer使用哪種方式進行身份認證

  - 相容MIT Kerberos的身份認證方式

    用戶端配置:

    如果在叢集的某個節點上執行用戶端命令，則需要將/etc/ecm/hadoop-conf/core-site.xml中hadoop.security.authentication.use.has設定為false.

如果有通過控制台的執行計畫跑作業，則不能修改master節點上面/etc/ecm/hadoop-conf/core-site.xml中的值，否則執行計畫的作業認證就不通過而失敗，可以使用下面的方式
export HADOOP_CONF_DIR=/etc/has/hadoop-conf臨時export環境變數，該路徑下的hadoop.security.authentication.use.has已經設定為false

訪問方式:Service的用戶端包完全可使用開源的，如HDFS用戶端等。詳見

- RAM身份認證

  用戶端配置:

  如果在叢集的某個節點上執行用戶端命令，則需要將/etc/ecm/hadoop-conf/core-site.xml中hadoop.security.authentication.use.has設定為true，/etc/has/has-client.conf中auth_type設定為RAM.
  如果有通過控制台的執行計畫跑作業，則不能修改master節點上面/etc/ecm/hadoop-conf/core-site.xml以及/etc/has/has-client.conf中的值，否則執行計畫的作業認證就不通過而失敗，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臨時export環境變數，其中HAS_CONF_DIR檔案夾下的has-client.conf的auth_type設定為RAM

  訪問方式： 用戶端需要使用叢集中的軟體包(如Hadoop/HBase等)，詳見

- LDAP身份認證

  用戶端配置:

  如果在叢集的某個節點上執行用戶端命令，則需要將/etc/ecm/hadoop-conf/core-site.xml中hadoop.security.authentication.use.has設定為true，/etc/has/has-client.conf中auth_type設定為LDAP.
  如果有通過控制台的執行計畫跑作業，則不能修改master節點上面/etc/ecm/hadoop-conf/core-site.xml以及/etc/has/has-client.conf中的值，否則執行計畫的作業認證就不通過而失敗，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臨時export環境變數，其中HAS_CONF_DIR檔案夾下的has-client.conf的auth_type設定為LDAP

  訪問方式：用戶端需要使用叢集中的軟體包(如Hadoop/HBase等)，詳見

- 執行計畫認證

  如果使用者有使用EMR控制台的執行計畫提交作業，則emr-header-1節點的配置必須不能被修改(預設配置)。

  用戶端配置:

  emr-header-1上面的/etc/ecm/hadoop-conf/core-site.xml中hadoop.security.authentication.use.has設定為true，/etc/has/has-client.conf中auth_type設定為EMR.

  訪問方式：跟非Kerberos安全叢集使用方式一致。詳見

- 其他

  登陸master節點訪問叢集

  叢集管理員也可以登陸master節點訪問叢集服務，登陸master節點切換到has帳號(預設使用相容MIT Kerberos的方式)即可訪問叢集服務，方便做一些排查問題或者營運等。

  ```
  >sudo su has
  >hadoop fs -ls /
  ```

  说明:
  也可以登入其他帳號操作叢集，前提是該帳號可以通過Kerberos認證。另外，如果在master節點上需要使用 相容MITKerberos的方式，需要在該帳號下先export一個環境變數。

  `export HADOOP_CONF_DIR=/etc/has/hadoop-conf/`

8.2 相容MIT Kerberos認證

相容MIT Kerberos的身份認證方式

EMR叢集中Kerberos服務端啟動在master節點，涉及一些管理操作需在master節點(emr-header-1)的root帳號執行。

下面以test使用者訪問HDFS服務為例介紹相關流程。

- Gateway上執行`hadoop fs -ls /`
  - 配置krb5.conf

    ```
    Gateway上面使用root帳號
     scp root@emr-header-1:/etc/krb5.conf /etc/
    ```

  - 添加principal
    - 登入叢集emr-header-1節點，切到root帳號。
    - 進入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: #直接按斷行符號可以看到一些命令的用法
      HadminLocalTool.local: addprinc #輸入命令按斷行符號可以看到具體命令的用法
      ```
      ```

```
HadminLocalTool.local: addprinc -pw 123456 test #添加test的princippal,密碼設定為123456
```

- 匯出keytab檔案

  使用Kerberos的admin工具可以匯出principal對應的keytab檔案。

  ```
  HadminLocalTool.local: ktadd -k /root/test.keytab test #匯出keytab檔案,後續可使用該檔案
  ```

- kinit擷取Ticket

  在執行hdfs命令的用戶端機器上面，如Gateway。

  ■ 添加linux帳號test

  ```
  useradd test
  ```

  ■ 安裝MITKerberos 用戶端工具。

  可以使用MITKerberos tools進行相關操作(如kinit/klist等)，詳細使用方式參考MITKerberos文檔

  ```
  yum install krb5-libs krb5-workstation -y
  ```

  ■ 切到test帳號執行kinit

  ```
  su test
  #如果沒有keytab檔案,則執行
  kinit #直接斷行符號
  Password for test: 123456 #即可
  #如有有keytab檔案,也可執行
  kinit -kt test.keytab test
  #查看ticket
  klist
  ```

  说明:

  MITKerberos工具使用執行個體

- 執行hdfs命令

  擷取到Ticket後，就可以正常執行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作業，需要提前在叢集中所有節點添加對應的linux帳號(詳見下文中[EMR叢集添加test帳號]。

- java代碼訪問HDFS
  - 使用本地ticket cache

    说明:

    需要提前執行kinit擷取ticket,且ticket到期後程式會訪問異常。

```
public static void main(String[] args) throws IOException {
  Configuration conf = new Configuration();
  //載入hdfs的配置,配置從emr叢集上複製一份
  conf.addResource(new Path("/etc/ecm/hadoop-conf/hdfs-site.xml"));
  conf.addResource(new Path("/etc/ecm/hadoop-conf/core-site.xml"));
  //需要在程式所在linux帳號下,提前kinit擷取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檔案(推薦)

    说明:

    keytab長期有效，跟本地ticket無關。

```
public static void main(String[] args) throws IOException {
 String keytab = args[0];
 String principal = args[1];
 Configuration conf = new Configuration();
 //載入hdfs的配置,配置從emr叢集上複製一份
 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檔案,該檔案從emr叢集master-1上面執行相關命令擷取[文檔前面有介紹命令]
 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依賴:

```
<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>
```

8.3 RAM認證

RAM身份認證

EMR叢集中的Kerberos服務端除了可以支援第一種MIT Kerberos相容的使用方式，也可以支援Kerberos用戶端使用RAM作為身份資訊進行身份認證。

RAM產品可以建立/管理子帳號，通過子帳號實現對雲上各個資源的存取控制。

主帳號的管理員可以在RAM的使用者管理介面建立一個子帳號(子賬戶名稱必須符合linux使用者的規範)，然後將子帳號的AccessKey下載下來提供給該子帳號對應的開發人員，後續開發人員可以通過配置AccessKey，從而通過Kerberos認證訪問叢集服務。

使用RAM身份認證不需要像第一部分MIT Kerberos使用方式一樣，提前在Kerberos服務端添加principle等操作

下面以已經建立的子帳號test在Gateway訪問為例:

- EMR叢集添加test帳號

  EMR的安全叢集的yarn使用了LinuxContainerExecutor，在叢集上跑yarn作業必須要在叢集所有節點上面添加跑作業的使用者帳號，LinuxContainerExecutor執行程式過程中會根據使用者帳號進行相關的許可權校正。

  EMR叢集管理員在EMR叢集的master節點上執行:

  ```
  sudo su hadoop
  sh adduser.sh test 1 2
  ```

  附:adduser.sh代碼

  ```
  #添加的賬戶名稱
   user_name=$1
  #叢集master節點個數,如HA叢集有2個master
  master_cnt=$2
  #叢集worker節點個數
  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管理員在Gateway機器上添加test使用者

```
useradd test
```

- Gateway管理員配置Kerberos基礎環境

```
sudo su root
 sh config_gateway_kerberos.sh 10.27.230.10 /pathto/emrheader1_pwd_file
 #確保Gateway上面/etc/ecm/hadoop-conf/core-site.xml中值為true
 <property>
   <name>hadoop.security.authentication.use.has</name>
   <value>true</value>
 </property>
```

附: config_gateway_kerberos.sh指令碼代碼

```
#EMR叢集的emr-header-1的ip
 masterip=$1
 #儲存了masterip對應的root登入密碼檔案
 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用戶端配置,將預設的auth_type從EMR改為RAM
 #也可以手工修改該檔案
 sed -i 's/EMR/RAM/g' /etc/has/has-client.conf
```

- test使用者登入Gateway配置AccessKey

```
#登入Gateway的test帳號
#執行指令碼
sh add_accesskey.sh test
```

附: add_accesskey.sh指令碼(修改一下AccessKey)

```
user=$1
 if [[ `cat /home/$user/.bashrc | grep 'export AccessKey'` == "" ]];then
 echo "
 #修改為test使用者的AccessKeyId/AccessKeySecret
 export AccessKeyId=YOUR_AccessKeyId
 export AccessKeySecret=YOUR_AccessKeySecret
 " >>~/.bashrc
 else
   echo $user AccessKey has been added to .bashrc
```

```
fi
```

- test使用者執行命令

  經過以上步驟，test使用者可以執行相關命令訪問叢集服務了。

  執行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作業

```
[test@gateway ~]$ hadoop jar /usr/lib/hadoop-current/share/hadoop/mapreduce/
hadoop-mapreduce-examples-2.7.2.jar pi 10 1
```

  跑spark作業

```
[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
```

8.4 LDAP認證

LDAP身份認證

EMR叢集還支援基於LDAP的身份認證，通過LDAP來管理帳號體系，Kerberos用戶端使用LDAP中的帳號資訊作為身份資訊進行身份認證。

LDAP帳號可以是和其它服務共用，比如Hue等，只需要在Kerberos服務端進行相關配置即可。使用者可以使用EMR叢集中已經配置好的LDAP服務(ApacheDS)，也可以使用已經存在的LDAP服務，只需要在Kerberos服務端進行相關配置即可。

下面以叢集中已經預設啟動的LDAP服務(ApacheDS)為例:

- Gateway管理對基礎環境進行配置(跟第二部分RAM中的一致，如果已經配置可以跳過)

  區別的地方只是/etc/has/has-client.conf中的auth_type需要改為LDAP

  也可以不修改/etc/has/has-client.conf,使用者test在自己的帳號下拷貝一份該檔案進行修改auth_type，然後通過環境變數指定路徑，如:

  `export HAS_CONF_DIR=/home/test/has-conf`

- EMR控制台配置LDAP管理使用者名/密碼到Kerberos服務端(HAS)

  進入EMR控制台叢集的組態管理-HAS軟體下，將LDAP的管理使用者名和密碼配置到對應的bind_dn和bind_password欄位，然後重啟HAS服務。

  此例中，LDAP服務即EMR叢集中的ApacheDS服務，相關欄位可以從ApacheDS中擷取。

- EMR叢集管理員在LDAP中添加使用者資訊

  - 擷取ApacheDS的LDAP服務的管理使用者名和密碼在EMR控制台叢集的組態管理/ApacheDS的配置中可以查看manager_dn和manager_password
  - 在ApacheDS中添加test使用者名和密碼

    ```
    登入叢集emr-header-1節點root帳號
     建立test.ldif檔案,內容如下:
     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 對應修改成密碼manager_password
     ldapmodify -x -h localhost -p 10389 -D "uid=admin,ou=system" -w "Ns1aSe" -a -f
    test.ldif
     #刪除test.ldif
     rm test.ldif
    ```

    將添加的使用者名/密碼提供給test使用

- 使用者test配置LDAP資訊

  ```
  登入Gateway的test帳號
   #執行指令碼
   sh add_ldap.sh test
  ```

  附: add_ldap.sh指令碼(修改一下LDAP帳號資訊)

  ```
  user=$1
   if [[ `cat /home/$user/.bashrc | grep 'export LDAP_'` == "" ]];then
   echo "
   #修改為test使用者的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
  ```

- 使用者test訪問叢集服務

  執行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作業等

8.5 執行計畫認證

E-MapReduce控制台-執行計畫

主帳號訪問

在主帳號登陸E-MapReduce控制台的情況下，在執行計畫頁面運行相關執行計畫，將作業提交到安全叢集上面執行，以hadoop使用者訪問作業中涉及的相關開源元件服務。

子帳號訪問

在RAM子帳號登陸E-MapReduce控制台的情況下，在執行計畫頁面運行相關執行計畫，將作業提交到安全叢集上面執行，以RAM子帳號對應的使用者名訪問作業中涉及的相關開源元件服務。

樣本

- 主帳號管理員根據需求建立多個子帳號(如A/B/C),在RAM控制台頁面給子帳號授予AliyunEMRFullAccess的許可權後，子帳號就能正常登陸並使用E-MapReduce控制台上的相關功能。
- 主帳號管理員將子帳號提供給相關開發人員。
- 開發人員完成作業的建立/執行計畫的建立,然後啟動運行執行計畫提交作業到叢集運行，最終在叢集上面以該子帳號對應的使用者名(A/B/C)去訪問相關的元件服務。

  说明:
  周期調度的執行計畫目前統一以hadoop帳號執行
- 元件服務使用子賬戶的使用者名進行相關的許可權控制，如子賬戶A是否有許可權訪問hdfs中的某個檔案等。

8.6 跨域互信

E-MapReduce中的Kerberos支援跨域訪問(cross-realm)，即不同的Kerberos叢集之間可以互相訪問。

下面以Cluster-A跨域去訪問Cluster-B中的服務為例:

- 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上面執行命令hostname擷取
  - realm可以在emr-header-1上面的/etc/krb5.conf擷取

添加principal

Cluster-A和Cluster-B兩個叢集的emr-header-1節點分別執行以下完全一樣的命令:

```
# root帳號
    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是密碼,可自行修改
- EMR.6789.COM是Cluster-B的realm，即被訪問的叢集的realm
- EMR.1234.COM是Cluster-A的realm，即發起訪問的叢集realm

配置Cluster-A的/etc/krb5.conf

在Cluster-A叢集上配置[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 = .
    }
```

將上述/etc/krb5.conf同步到Cluster-A所有節點

將Cluster-B節點的/etc/hosts檔案中綁定資訊(只需要長網域名稱emr-xxx-x.cluster-xxx)拷貝到Cluster-A的所有節點/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上面如果要跑作業訪問Cluster-B，需要先重啟yarn
- Cluster-A的所有節點配置Cluster-B的host綁定資訊

訪問Cluster-B服務

在Cluster-A上面可以用Cluster-A的Kerberos的keytab檔案/ticket緩衝，去訪問Cluster-B的服務。

如訪問Cluster-B的hdfs服務:

```
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
```