Alibaba Cloud Elasticsearch

ベストプラクティス

Document Version20190716

目次

1 Beats による可視化された O&M システムの構築

背景

Beats はデータシッパー専用のプラットフォームです。 Beats をインストールすると、軽量の Beats エージェントから Logstash や Elasticsearch などのターゲットオブジェクトにインスタンスのデータを送信できます。

Beats エージェントおよび軽量シッパーとして、Metricbeat はシステムやサービスからメトリックを収集し、そのメトリックを Elasticsearch などのターゲットオブジェクトに送信するように設計されています。 Metricbeat は、システムやサービスの統計情報を CPU からメモリ、Redis から NGINX などに送信するための軽量な方法です。

このトピックでは、Metricbeat を使用して MacBook からメトリックを収集し、そのメトリックを Elasticsearch インスタンスに送信し、Kibana にダッシュボードを生成する方法について説明します。

 注：
Linux または Windows システムを実行しているコンピューターからメトリックを収集し、そのメトリックを Elasticserach インスタンスに送信する手順もほぼ同じです。

1. Elasticsearch インスタンスの購入と設定

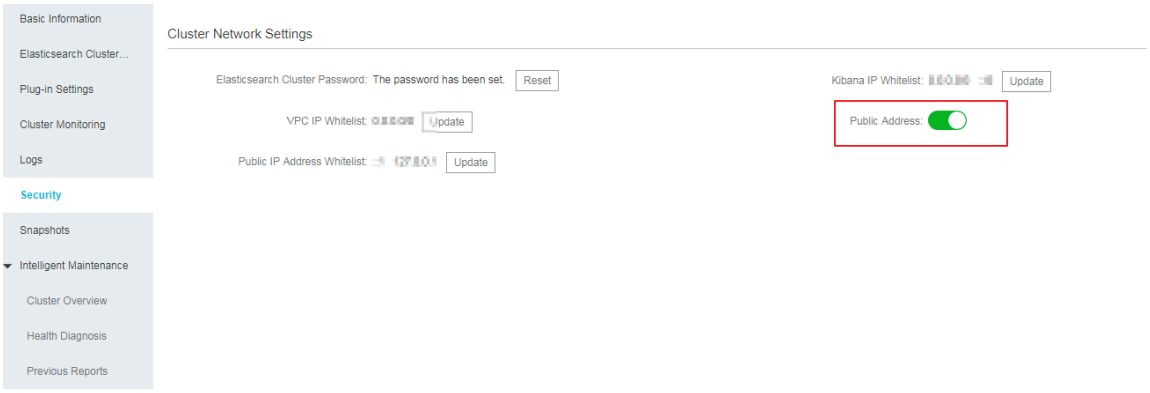
   Elasticsearch インスタンスがない場合は、Elasticsearch を有効化してインスタンスを作成する必要があります (前提条件)。 その後、インスタンスの内部またはパブリック IP アドレスを介して、MacBook で収集したデータを Elasticsearch インスタンスに送信できるようになります。

    注：

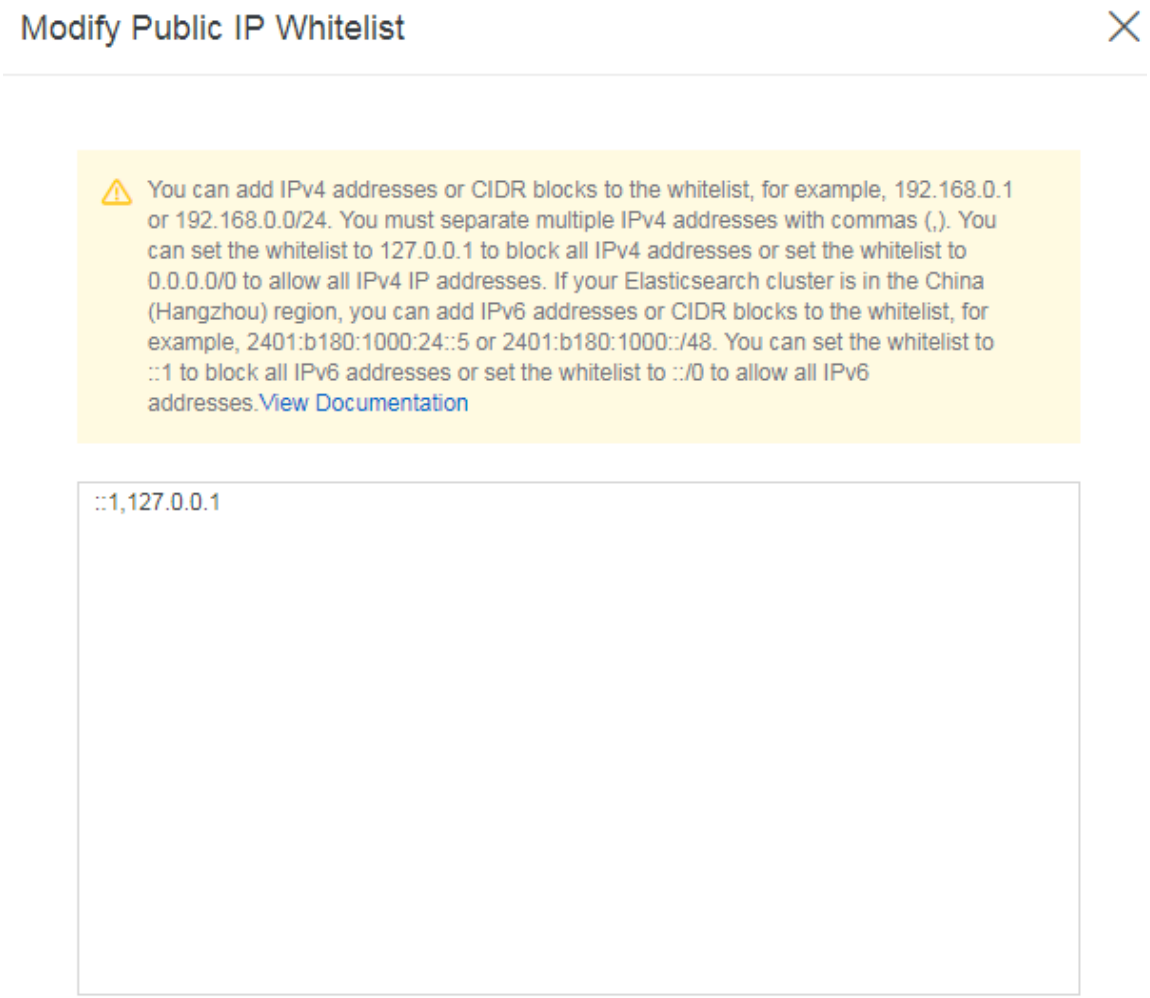
   - パブリック IP アドレスを介して Elasticsearch インスタンスにアクセスする場合、[セキュリティ] ページで [パブリックアドレス] のスイッチをオンにし、[パブリック IP アドレスのホワイトリスト] を設定する必要があります。

- 内部 IP アドレスを介して Elasticsearch インスタンスにアクセスする場合、Elasticsearch インスタンスと同じ VPC とリージョン に Elastic Compute Service (ECS) を作成して、Elasticsearch へのアクセスを管理する必要があります。

a. Elasticsearch コンソールにログインし、インスタンスの名前か ID をクリックしてから、左側のナビゲーションウィンドウで [セキュリティ] をクリックします。 [セキュリティ] ページで、[パブリックアドレス] のスイッチをオンにします。



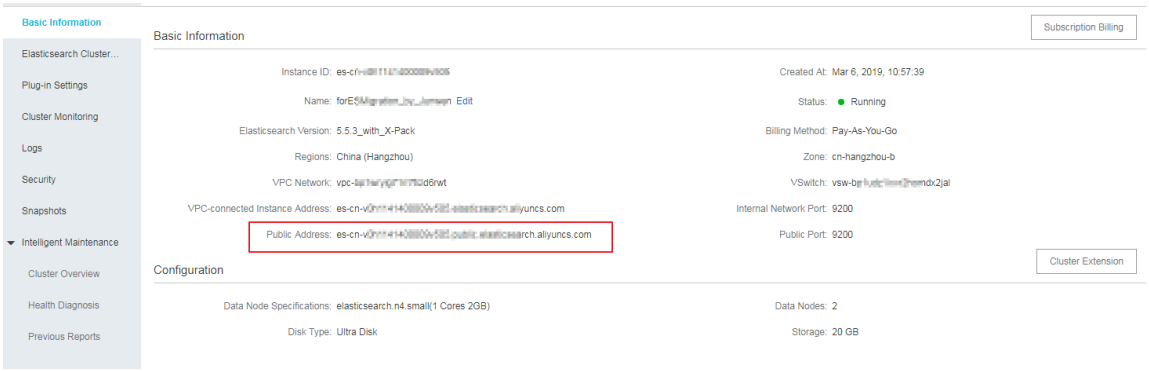
b. MacBook のパブリック IP アドレスをホワイトリストに追加します。



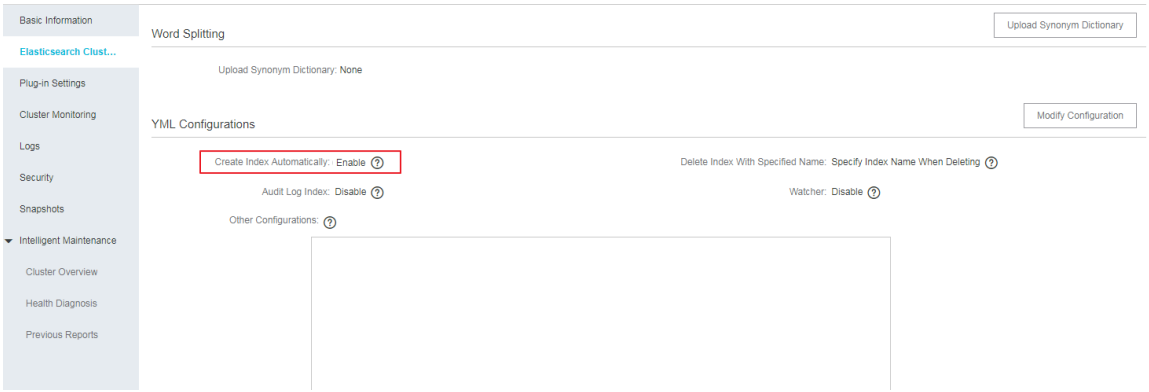
：

パブリックネットワークを使用する場合、パブリックネットワークのアウトバウンドネットワークトラフィックを制御するジャンプサーバーの IP アドレスをホワイトリストに追加します。 ジャンプサーバーの IP アドレスを取得できない場合、 `0.0.0.0/1`, `128.0.0.0/1` をホワイトリストに追加して、特定の IP アドレスを許可します。 この設定により、Elasticsearch インスタンスはパブリックネットワークに公開されます。 リスクを評価し、慎重に進めてください。

c. 設定が完了したら、左側のナビゲーションペインで [基本情報]をクリックし、Elaticsearch インスタンスのパブリック IP アドレスをコピーします。



d. YML 設定を変更します。 [YML 設定] ページで、[自動インデックス] を有効化します。 デフォルトで、この機能は無効化されています。 この操作により、Elasticsearch インスタンスが再起動され、有効になるまでに時間がかかります。



2. Metricbeat のダウンロードと設定

- Metricbeat インストールパッケージ (Mac オペレーティングシステム用)
- Metricbeat インストールパッケージ (32 ビット Linux オペレーティングシステム用)
- Metricbeat インストールパッケージ (64 ビット Linux オペレーティングシステム用)
- Metricbeat インストールパッケージ (32 ビット Windows オペレーティングシステム用)
- Metricbeat インストールパッケージ (64 ビット Windows オペレーティングシステム用)

a. Metricbeat ファイルをダウンロード、解凍し、開きます。

```
zhaohongyangdeMacBook-Pro:Desktop zhaohongyang$ cd metricbeat-6.3.1-darwin-x86_64
zhaohongyangdeMacBook-Pro:metricbeat-6.3.1-darwin-x86_64 zhaohongyang$ ls
LICENSE.txt                 data                        logs                        metricbeat.yml
NOTICE.txt                  fields.yml                  metricbeat                  modules.d
README.md                   kibana                      metricbeat.reference.yml
zhaohongyangdeMacBook-Pro:metricbeat-6.3.1-darwin-x86_64 zhaohongyang$
```

b. metricbeat.yml ファイルの Elasticsearch output セクションを開き、編集します。 該当するコンテンツのコメントを解除する必要があります。

```
#================================== Outputs ===================================

# Configure what output to use when sending the data collected by the beat.

#-------------------------- Elasticsearch output ------------------------------
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["es-cn-[redacted].public.elasticsearch.aliyuncs.com:9200"]

  # Optional protocol and basic auth credentials.
  protocol: "http"
  username: "[redacted]"
  password: "[redacted]"
```

注 :

Elasticsearch のアクセス制御情報は、以下のとおりです。

- hosts：Elasticsearch インスタンスのパブリックまたは内部 IP アドレス。 この例では、パブリック IP アドレスを使用しています。
- protocol： `http` を設定します。
- username：デフォルトのユーザー名は elastic です。
- password：Elasticsearch へのログインに使用されるパスワード。

3. Metricbeat の有効化

   次のコマンドを実行して Metricbeat を有効化し、Metricbeat を使用して Elasticsearch インスタンスにデータを送信します。

```
./ metricbeat  - e  - c   metricbeat . yml
```

```
zhaohongyangdeMacBook-Pro:metricbeat-6.3.1-darwin-x86_64 zhaohongyang$ ./metricbeat -e -c metricbeat.yml
```

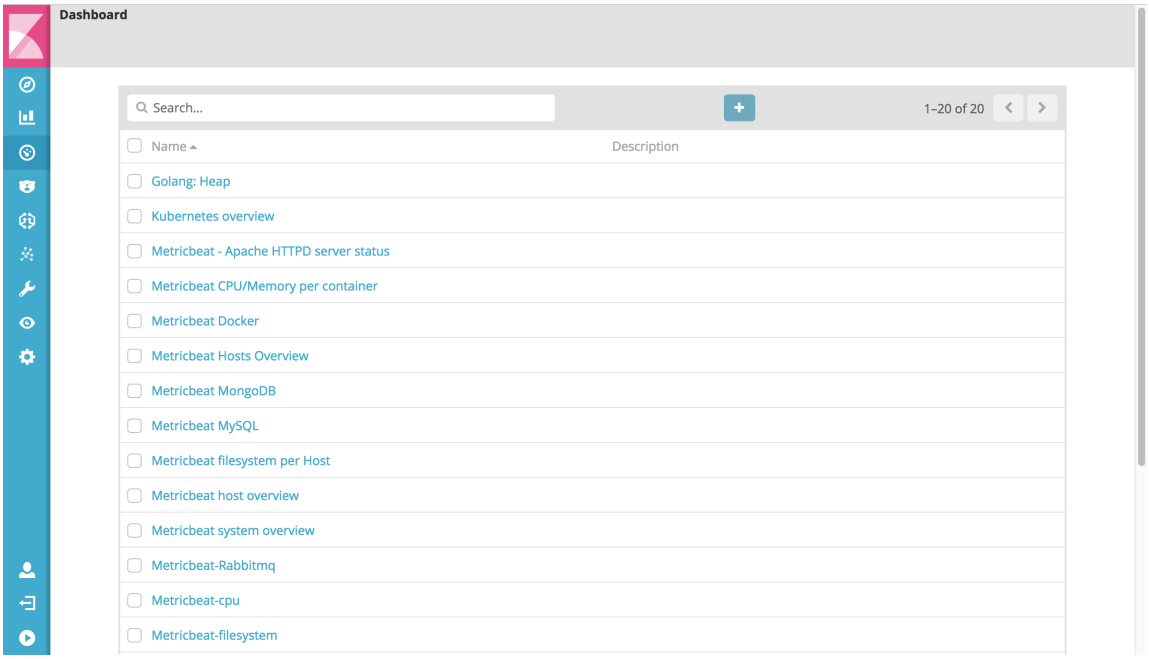
4. Kibana でのダッシュボードの表示

   Elasticsearch コンソールの右上にある Kibana コンソールをクリックします。 Dashboard ページに移動します (次図)。

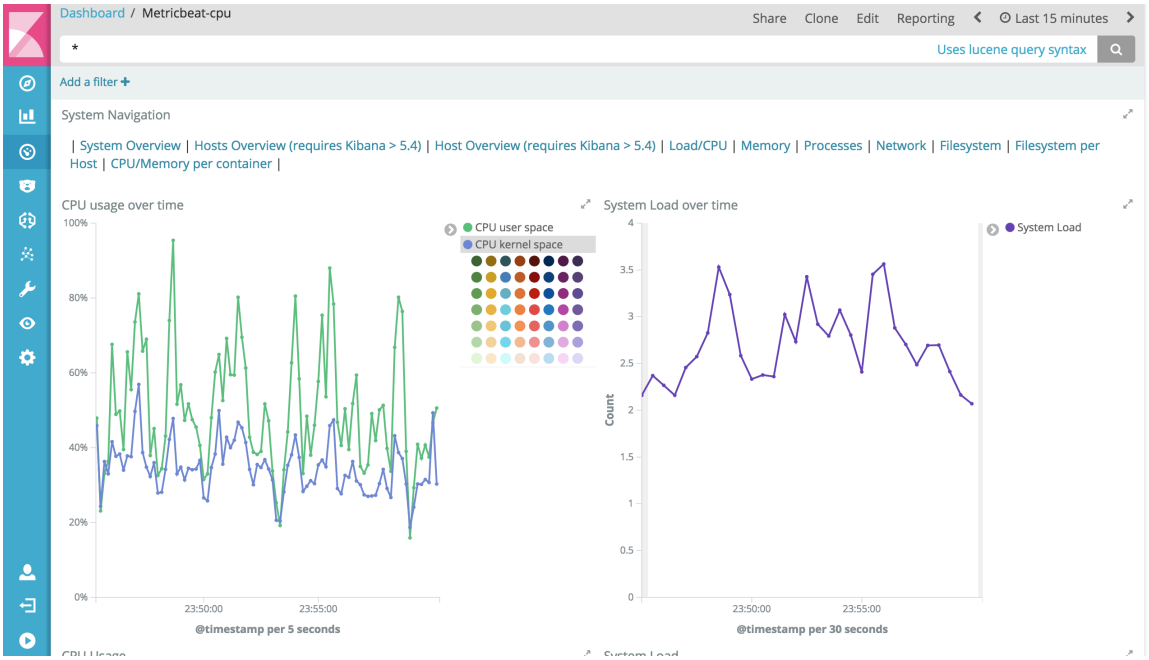
注 :

Kibana コンソールでインデックスパターンを作成していない場合、ダッシュボードに情報が表示されないことがあります。 この問題を解決するには、インデックスパターンを作成して、[ダッシュボード]ページで情報をもう一度表示します。

a. メトリックのリスト。



b. CPU メトリック。



注：

5 秒ごとにデータを更新してレポートを生成するようシステムをスケジュールしたり、例外が発生したときにアラートを送信するよう webhook を設定したりすることができます。

2 Curator の使用

Elasticsearch Curator のインストール

1. Elasticsearch インスタンスと同じ VPC ネットワークにある ECS インスタンスを購入します。 この例では、CentOS 7.3 64 ビットを実行する ECS インスタンスを使用します。
2. 以下のコマンドを実行します。

   a. Elasticsearch Curator のインストール

   ```
   pip install elasticsearch-curator
   ```

   注：

   - Elasticsearch Curator 5.6.0 をインストールすることを推奨します。 このバージョンは、Elasticsearch 5.5.3 と 6.3.2 をサポートします。
   - Curator と Elasticsearch のバージョンの互換性

   b. Curator のバージョンの表示

   ```
   curator --version
   ```

   返されるバージョン情報

   ```
   curator, version 5.6.0
   ```

Singleton コマンドラインインターフェイス

- `curator_cli i` を使用して、操作を実行します。
- Singleton コマンドラインインターフェイス

注：

- 一度に実行できる操作は 1 つだけです。
- Curator を使用してすべての操作を実行できるわけではありません (Alias や Restore など)。

Crontab によるタスクのスケジュール

crontab コマンドと curator コマンドを使用して、複数の操作を実行するようにタスクをスケジュールすることができます。

Curator コマンド：

```
 curator  [ OPTIONS ]  ACTION_FIL  E
オプション：
  -- config   PATH  設定ファイルへのパス。  デフォルト：~/. curator /
 curator . yml
  -- dry - run  何も変更しません。
  -- version    バージョンを表示して終了します。
  -- help       このメッセージを表示して終了します。
```

- curator コマンドを実行するとき、config.yml ファイル (公式リファレンス) を指定する必要があります。
- curator コマンドを実行するとき、 action.yml ファイル (公式リファレンス) を指定する必要があります。

Hot-Warm アーキテクチャの演習

Curator を使用して hot ノードから warm ノードにインデックスを移行します (公式リファレンス)。

hot ノードから warm ノードへのインデックスの移行

1. 以下のように、config.yml ファイルをパス / *usr* / *curator* / に作成します。

：

- hosts ：アクセスする Elasticsearch インスタンスのアドレスに置き換えます。 この例では、Elasticsearch インスタンスのプライベートアドレスを使用しています。
- http_auth ：Elasticsearch インスタンスへのログインに使用するユーザー名とパスワードに置き換えます。

```
client :
  hosts :
   -  http :// es - cn - 0pp0z9p2v0  0031234 . elasticsea  rch .
aliyuncs . com
  port :  9200
  url_prefix :
  use_ssl :  False
  certificat  e :
  client_cer  t :
  client_key :
  ssl_no_val  idate :  False
  http_auth :  user : password
  timeout :  30
  master_onl  y :  False
logging :
  loglevel :  INFO
  logfile :
  logformat :  default
```

```
    blacklist : [' elasticsea  rch ', ' urllib3 ']
```

2. 以下のように、action.yml ファイルをパス / *usr / curator /* に作成します。

注：

- 以下のコンテンツでは、30 分前に作成され、 logstash - で始まるインデックスを hot ノードから warm ノードに移行します。
- 以下のコンテンツは、ビジネスニーズに合わせてカスタマイズできます。

```
actions :
  1 :
    action :  allocation
    descriptio  n : " Apply   shard   allocation   filtering
rules   to   the   specified   indices "
    options :
      key :  box_type
      value :  warm
      allocation  _type :  require
      wait_for_c  ompletion :  true
      timeout_ov  erride :
      continue_i  f_exceptio  n :  false
      disable_ac  tion :  false
    filters :
    -  filtertype :  pattern
       kind :  prefix
       value :  logstash -
    -  filtertype :  age
       source :  creation_d  ate
       direction :  older
       timestring : '% Y -% m -% dT % H :% M :% S '
       unit :  minutes
       unit_count :  30
```

3. curator コマンドが正常に実行されるかどうか確認します。

```
curator  -- config  / usr / curator / config . yml  / usr / curator
/ action . yml
```

コマンドが正常に実行されると、以下の情報が返されます。

```
2019 - 02 - 12   20 : 11 : 30 , 607   INFO        Preparing
Action   ID :  1 , " allocation "
2019 - 02 - 12   20 : 11 : 30 , 612   INFO        Trying   Action
ID :  1 , " allocation ":  Apply   shard   allocation   filtering
rules   to   the   specified   indices
2019 - 02 - 12   20 : 11 : 30 , 693   INFO        Updating   index
  setting  {' index . routing . allocation . require . box_type ':
' warm '}
2019 - 02 - 12   20 : 12 : 57 , 925   INFO        Health   Check
for   all   provided   keys   passed .
2019 - 02 - 12   20 : 12 : 57 , 925   INFO        Action   ID :  1
, " allocation "  completed .
```

```
 2019 - 02 - 12   20 : 12 : 57 , 925   INFO        Job   completed .
```

4. **crontab コマンドを実行して、15 分間隔で curator コマンドを実行します。**

```
*/ 15  * * * *  curator  -- config  / usr / curator / config . yml
 / usr / curator / action . yml
```

3 データの同期と移行

3.1 クラウドデータのインポート

Alibaba Cloud から ES へのデータのインポート (オフライン)

Alibaba Cloud は、豊富なクラウドストレージとデータベースプロダクトを提供しています。これらのプロダクトのデータの分析や検索をする場合、Data Integration を使用してください。Data Integration では、オフラインデータを 5 分ごとに Elasticsearch に同期させることができます。

サポートされているデータソース

- Alibaba Cloud データベース (MySQL、PostgreSQL、SQL Server、PPAS、MongoDB、HBase)
- DRDS
- MaxCompute
- OSS
- Table Store
- 自社開発の HDFS、Oracle、FTP、DB2、および以前のクラウドデータベースの自社開発バージョン

> 注 :
> データ同期により、パブリックネットワークのトラフィック料金が発生することがあります。

手順

オフラインデータをインポートするには、次の手順を実行します。

- VPC 内で Elasticsearch とやり取り可能な ECS インスタンスを準備します。 この ECS インスタンスはデータソースを取得し、ES データを書き込むジョブを実行します (このジョブは Data Integration によって一元的に実行されます)。
- Data Integration サービスを有効化して、実行可能ジョブリソースとして ECS インスタンスを Data Integration サービスに登録する必要があります。
- データ同期スクリプトを設定し、定期的に実行させます。

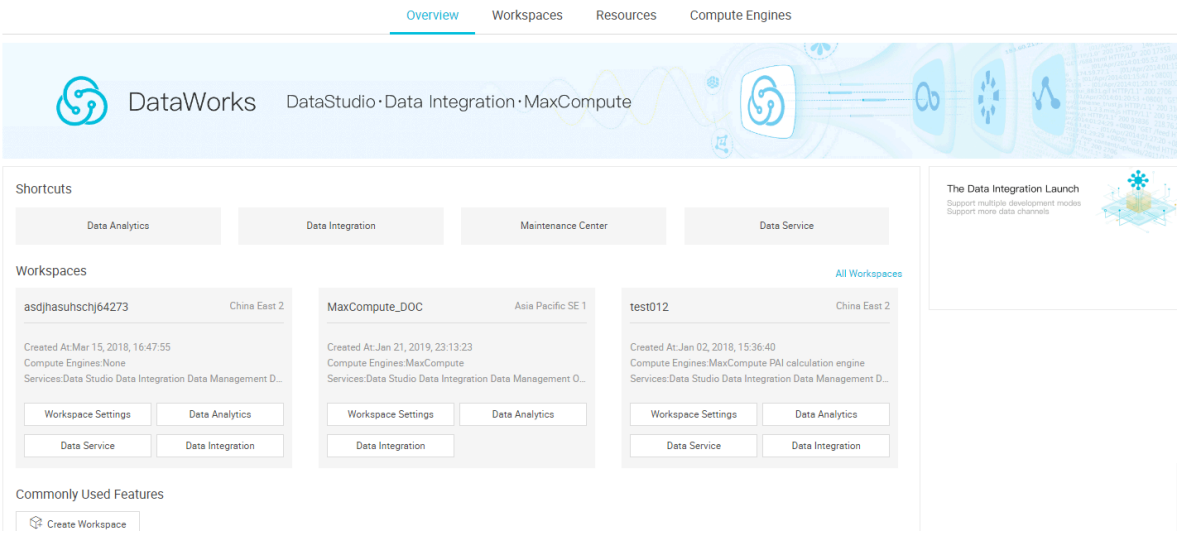
ステップ

1. Elasticsearch サービスと同じ VPC にある ECS インスタンスを購入します。 パブリック IP アドレスを ECS インスタンスに割り当てるか、ECS インスタンスの Elastic IP アドレスを有効にします。 コストを抑えるため、既存の ECS インスタンスを使用できます。 ECS インスタンスの購入方法は、「手順 2: インスタンスの作成」をご参照ください。

   > 注 :
   >
   > - CentOS 6、CentOS 7、AliyunOS を推奨します。
   > - 追加された ECS インスタンスで MaxCompute または同期タスクを実行する必要がある場合、ECS インスタンスの現在の Python バージョンが 2.6 または 2.7 かどうかを確認します (CentOS 5 の Python バージョンは 2.4 ですが、他のオペレーティングシステムの Python バージョンは 2.6 以降)。
   > - ECS インスタンスにパブリック IP アドレスが割り当てられていることを確認します。

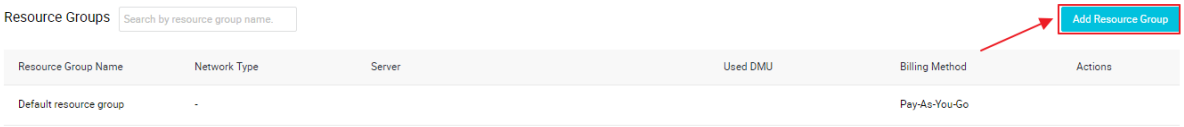
2. Data Integration コンソールにログインして、ワークベンチを開きます。

   Data Integration または DataWorks が有効化されている場合、次のように表示されます。



   Data Integration または DataWorks が有効化されていない場合、次のメッセージが表示されます。 手順に従って、Data Integration サービスを有効化します。 有料サービスなので、見積もり価格を予算と照らし合わせてチェックしてください。

3. Data Integration サービスの [Project Management-Scheduling Resource Management] ページに移動し、VPC の ECS インスタンスをスケジューリングリソースとして設定します。 詳細は、「スケジューリングリソースの追加」をご参照ください。



4. Data Integration サービスでデータ同期スクリプトを設定します。 設定手順は、「スクリプトモードの設定」をご参照ください。 Elasticsearch の設定方法は、「ElasticSearch Writer の設定」をご参照ください。

   注：

   - 同期スクリプトの設定は 3 つの部分に分けられます。Reader は、アップストリームデータソース (データ同期のクラウドプロダクト) の設定、Writer は ES の設定、setting はパケットロス率や最大同時性などの同期の設定です。
   - ES Writer の accessId と accessKey は、それぞれ Elasticsearch のユーザー名とパスワードです。

5. スクリプトを設定したら、データ同期ジョブを送信します。 ジョブの実行サイクルを設定し、[OK] をクリックします。

   注：

   - 定期スケジュールを設定する場合、このポップアップウィンドウでジョブ開始時間、実行間隔、ジョブライフサイクルなどのパラメーターを設定します。
   - 設定したルールに従って、翌日の 00:00 に定期ジョブが実行されます。

6. 送信後、[O&M Center-Task Scheduling] ページに移動して送信されたジョブを見つけ、デフォルトのスケジューリングリソースから、設定したスケジューリングリソースに変更します。

リアルタイムデータのインポート

この機能は現在開発中です。今後利用可能になる予定です。

3.2 DataWorks による Hadoop と ES データの同期

このトピックでは、DataWorks のデータ同期機能を使用して、Hadoop から Elasticsearch (ES) へデータを移行し、分析する方法を説明します。 Java コードを使用してデータを同期することもできます。 詳細は、「ES-Hadoop と Elasticsearch 間のデータ相互接続」と「E-MapReduce で ES-Hadoop を使用」をご参照ください。

前提条件

1. Hadoop クラスターの作成

   データを移行するには、Hadoop クラスターを作成する必要があります。 このトピックでは、E-MapReduce サービス (EMR) を使用して、Hadoopクラスターを作成します。 詳細は、「クラスターの作成」をご参照ください。

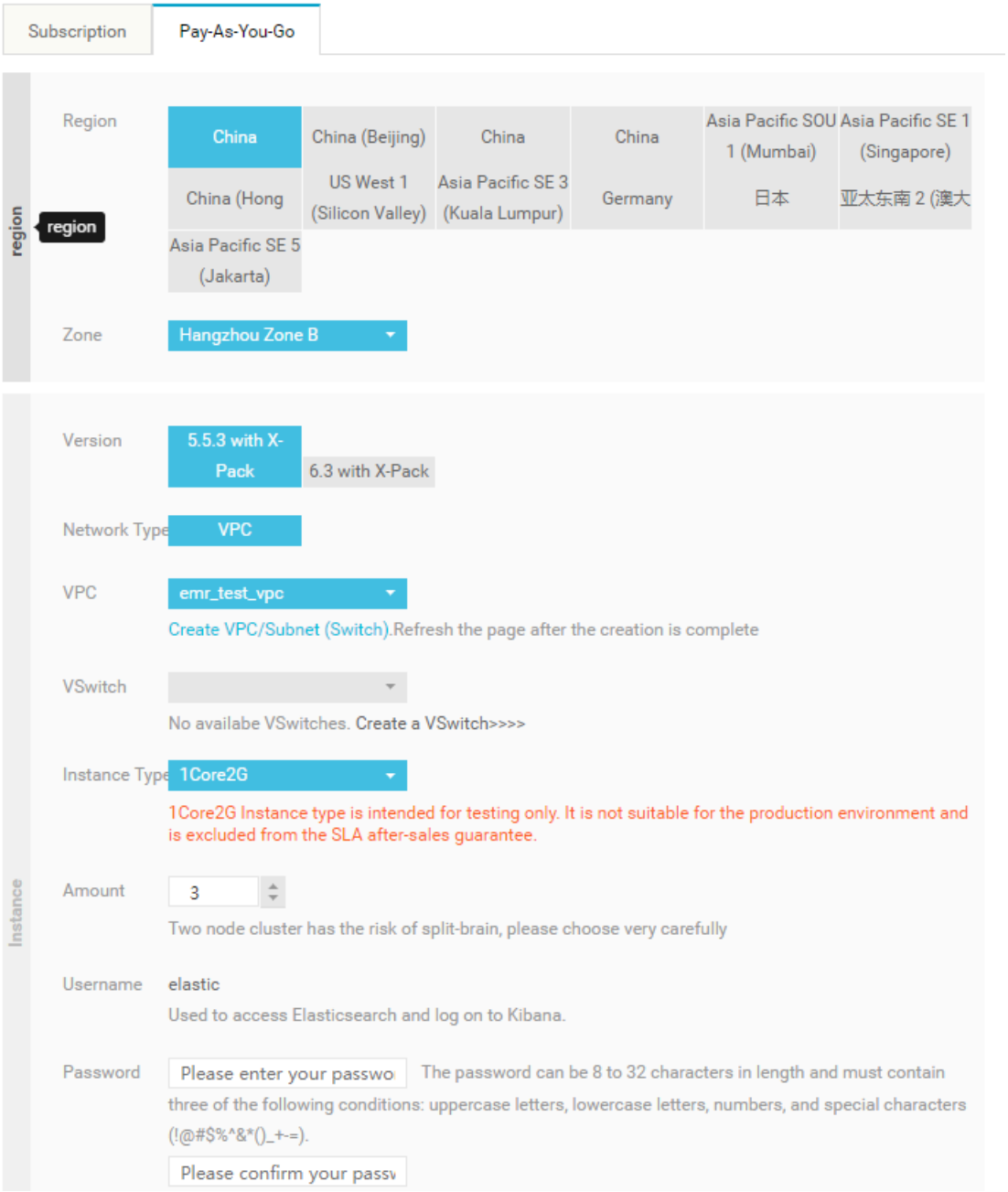
   以下の EMR Hadoop バージョン情報を使用します。

   - EMR バージョン： EMR-3.11.0
   - クラスタータイプ：HADOOP
   - サービス：HDFS2.7.2 / YARN2.7.2 / Hive2.3.3 / Ganglia3.7.2 / Spark2.3.1 / HUE4.1.0 / Zeppelin0.8.0 / Tez0.9.1 / Sqoop1.4.7 / Pig0.14.0 / ApacheDS2.0.0 / Knox0.13.0

   また、このトピックでは、Hadoop クラスターに VPC ネットワークを使用し、リージョンを [中国 (杭州)] に設定し、ECS マスターノードにパブリック IP とプライベート IP を設定し、非高可用性 (非HA) モードを選択します。

2. Elasticsearch

   Elasticsearch コンソール にログインし、EMR クラスターと同じリージョンと VPC ネットワークを選択します。 ES インスタンスの購入方法は、「購入と設定」をご参照ください。



3. DataWorks

   ワークスペースの作成 を行い、リージョンを [中国 (杭州)] に設定します。 次の例では、プロジェクト bigdata_DOC を使用しています。

データの準備

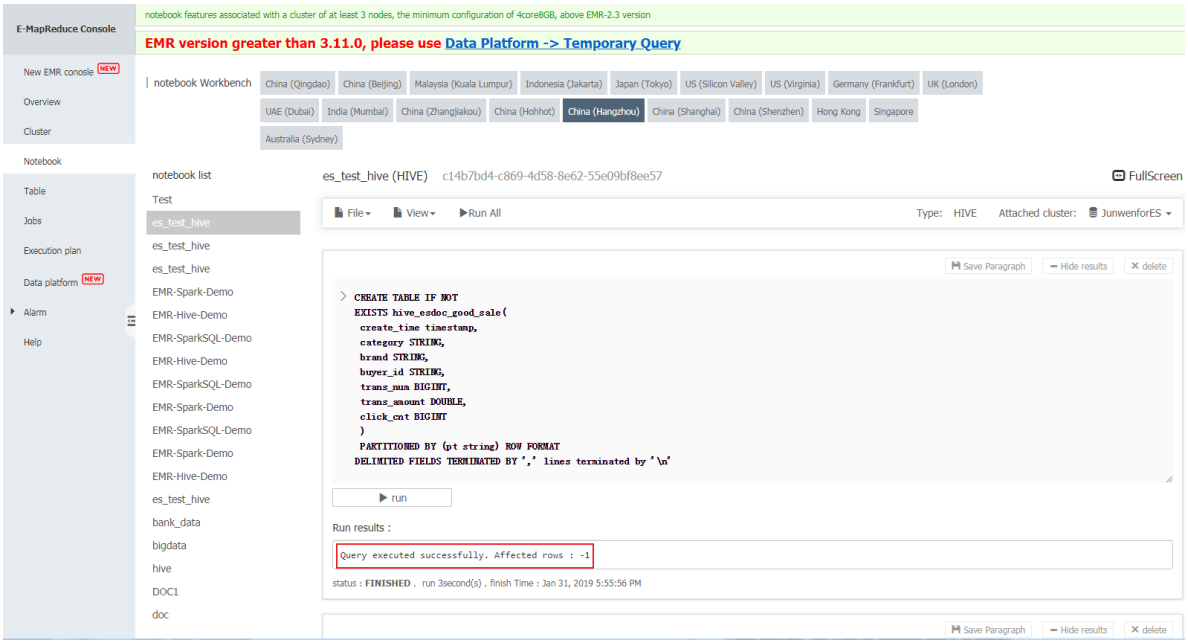
Hadoop クラスターにテストデータを作成するには、次の手順に従います。

1. EMR コンソールにログインし、[旧版ジョブスケジューラ] に移動し、左側のナビゲーションウィンドウで、[ノートブック] をクリックします。
2. [ファイル] > [新しいノートブック] をクリックします。 この例では、es_test_hive という名前のノートブックを作成します。 デフォルトタイプを [Hive] に設定します。 接続するクラスターは、作成済みの EMR Hadoop クラスターです。
3. Hive テーブルを作成する構文を入力します。

```
CREATE   TABLE   IF   NOT

EXISTS   hive_esdoc  _good_sale (
 create_tim  e   timestamp ,
 category   STRING ,
 brand   STRING ,
 buyer_id   STRING ,
 trans_num   BIGINT ,
 trans_amou  nt   DOUBLE ,
 click_cnt   BIGINT
)
 PARTITIONE  D   BY  ( pt   string )  ROW   FORMAT
DELIMITED   FIELDS   TERMINATED   BY  ','  lines   terminated   by
'\ n '
```

4. [実行] をクリックします。 クエリは正常に実行されましたというメッセージが表示された場合、テーブル hive_esdoc_good_sale は EMR Hadoop クラスターに作成されています (次図)。



5. テストデータを挿入します。 OSS や他のデータソースからデータをインポートしたり、手動でデータを挿入したりできます。 この例では、データを手動で挿入します。 データを挿入するスクリプトは、次のとおりです。

```
insert   into
```

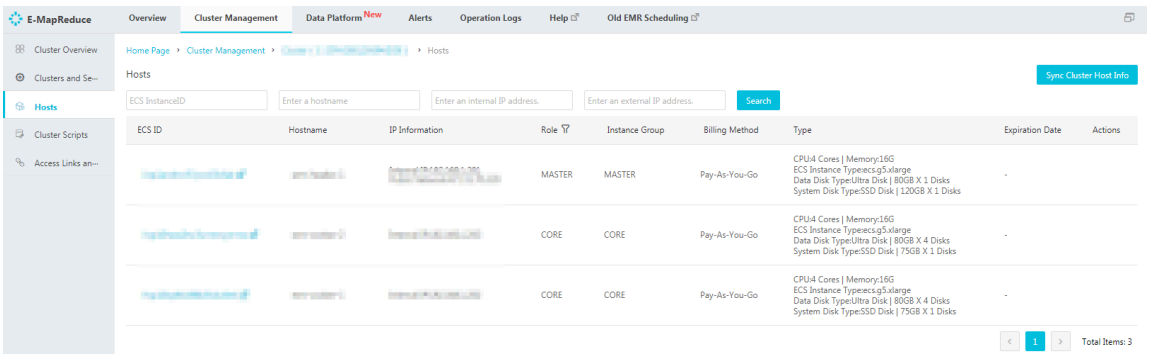
```
hive_esdoc _good_sale PARTITION ( pt = 1 ) values (' 2018 -
08 - 21 ',' Jacket ',' Brand A ',' lilei ', 3 , 500 . 6 , 7 ),('
2018 - 08 - 22 ',' Fresh food ',' Brand B ',' lilei ', 1 , 303
, 8 ),(' 2018 - 08 - 22 ',' Jacket ',' Brand C ',' hanmeimei ',
2 , 510 , 2 ),( 2018 - 08 - 22 ,' Bathroom accessory ',' Brand
A ',' hanmeimei ', 1 , 442 . 5 , 1 ),(' 2018 - 08 - 22 ',' Fresh
food ',' Brand D ',' hanmeimei ', 2 , 234 , 3 ),(' 2018 - 08
- 23 ',' Jacket ',' Brand B ',' jimmy ', 9 , 2000 , 7 ),(' 2018
- 08 - 23 ',' Fresh food ',' Brand A ',' jimmy ', 5 , 45 . 1
, 5 ),(' 2018 - 08 - 23 ',' Jacket ',' Brand E ',' jimmy ', 5 ,
100 . 2 , 4 ),(' 2018 - 08 - 24 ',' Fresh food ',' Brand G ','
peiqi ', 10 , 5560 , 7 ),(' 2018 - 08 - 24 ',' Bathroom accessory
',' BrandF ',' peiqi ', 1 , 445 . 6 , 2 ),(' 2018 - 08 - 24 ','
Jacket ',' Brand A ',' ray ', 3 , 777 , 3 ),(' 2018 - 08 - 24
',' Bathroom accessory ',' Brand G ',' ray ', 3 , 122 , 3 ),('
2018 - 08 - 24 ',' Jacket ',' Brand C ',' ray ', 1 , 62 , 7 ) ;
```

6. データの挿入後、`select * from hive_esdoc _good_sale where pt = 1 ;` 文を実行し、データが EMR Hadoop クラスターテーブルに存在することを確認します。

データの同期

> 注：
> 通常、DataWorks プロジェクトのネットワーク環境は Hadoop クラスターコアノードのネットワーク環境に接続されていないため、Hadoop クラスターマスターノードで DataWorks の同期タスクを実行するようにリソースグループをカスタマイズできます (Hadoop クラスターのマスターノードとコアノードは相互接続されていることが多いため)。

1. EMR Hadoop クラスターのコアノードの表示
   a. EMR コンソールのメニューバー上部にある [クラスター管理] をクリックします。
   b. ターゲットクラスターを見つけ、右側の [管理] をクリックします。
   c. 左側のナビゲーションウィンドウで、[ホスト] をクリックしてマスターノードとコアノードを表示します (次図)。



> 注：

通常、非 HA EMR Hadoop クラスターのマスターノード名は emr-header-1、コアノード名は emr-worker-X です。

d. 上の図のマスターノードの ECS ID をクリックして、[インスタンスの詳細] ページに移動します。 [接続] をクリックして、ECS インスタンスに接続します。 `hadoop dfsadmin -report` コマンドを実行して、コアノード情報を表示することもできます。

 注：

ECS マスターノードのログインパスワードは、EMR Hadoop クラスターを作成したときに設定したパスワードです。

```
DFS Remaining: 665931456512 (620.20 GB)
DFS Used: 209780736 (200.06 MB)
DFS Used%: 0.03%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0

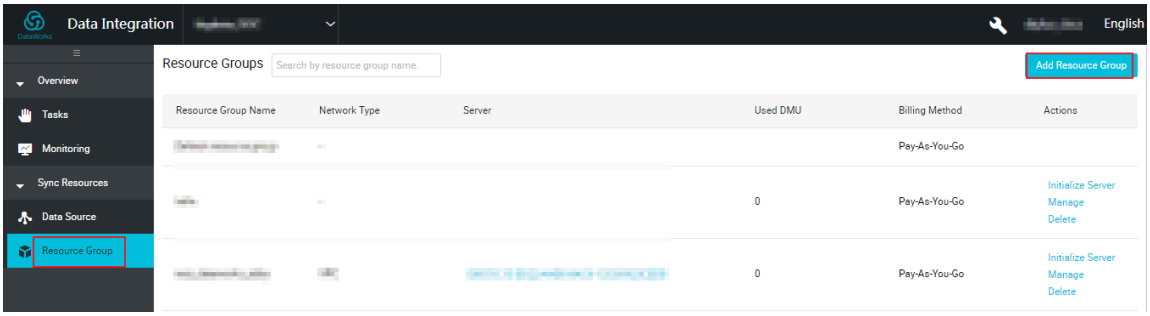
-------------------------------------------------
Live datanodes (2):

Name: 192.168.1.206:50010 (emr-worker-2.cluster-77026)
Hostname: emr-worker-2.cluster-77026
Decommission Status : Normal
Configured Capacity: 333373341696 (310.48 GB)
DFS Used: 104890368 (100.03 MB)
Non DFS Used: 302723072 (288.70 MB)
DFS Remaining: 332965728256 (310.10 GB)
DFS Used%: 0.03%
DFS Remaining%: 99.88%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Sat Sep 29 17:37:46 CST 2018

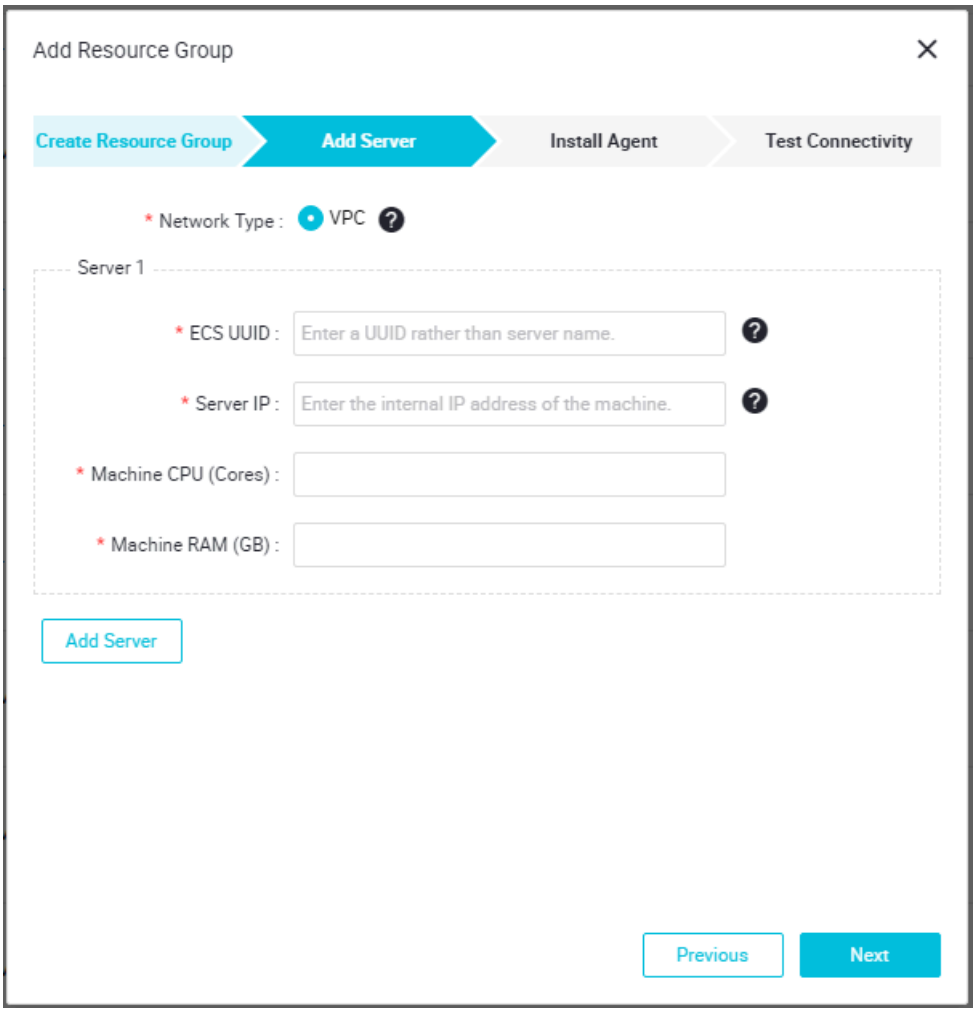

Name: 192.168.1.205:50010 (emr-worker-1.cluster-77026)
Hostname: emr-worker-1.cluster-77026
Decommission Status : Normal
Configured Capacity: 333373341696 (310.48 GB)
DFS Used: 104890368 (100.03 MB)
Non DFS Used: 302723072 (288.70 MB)
DFS Remaining: 332965728256 (310.10 GB)
DFS Used%: 0.03%
DFS Remaining%: 99.88%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Sat Sep 29 17:37:46 CST 2018
```

2. カスタムリソースグループの作成

a. DataWorks コンソールで、[Data Integration] ページに移動し、[Resource Group > [Add resources Group] を選択します。 カスタムリソースグループの詳細は、「スケジューリングリソースの追加」をご参照ください。



b. リソースグループの名前とサーバー情報を入力します。 サーバーは、EMR クラスターのマスターノードです。



・ [Network type] は、[proprietary network (VPC)] です。

注：

VPC ネットワークの場合、ECS インスタンスの UUID を入力する必要があります。 クラシックネットワークの場合、インスタンス名を入力する必要があります。 現在、[中国 (上海)] リージョンの DataWorks 2.0 のみが、クラシックネットワークスケジューリングリソースの追加をサポートしています。 他のリージョンでは、使用しているのがクラシックネットワークか VPC ネットワークかにかかわらず、スケジューリングリソースグループを追加する場合は VPC ネットワークタイプを選択する必要があります。

- ECS UUID：EMR クラスターマスターノードにログインし、`dmidecode | grep UUID` を実行して戻り値を取得します。
- Machine IP：マスターノードのパブリックIP。Machine CPU：マスターノードの CPU。Memory size：マスターノードのメモリ。ECS コンソールでマスターノード ID をクリックすると、設定情報セクションで上記の情報を取得できます。

c. [Add server] ステップの完了後、マスターノードと DataWorks のネットワークが相互接続されていることを確認する必要があります。 ECS サーバーを使用している場合、サーバーセキュリティグループを設定する必要があります。 プライベート IP を使用している場合、「セキュリティグループの追加」をご参照ください。 パブリック IP アドレスを使用している場合、[Security Group Rules] でインターネットの入力と出力を直接設定できます。 この例では、DataWorks と同じリージョンにある VPC ネットワークで EMR クラスターを使用しているので、セキュリティグループを設定する必要はありません。

d. プロンプトに従ってエージェントをインストールします。 available ステータスが表示されたら、リソースグループが正常に追加されたことを示します。

注：

この例では、VPC ネットワークを使用しているので、ポート 8000 を開く必要はありません。

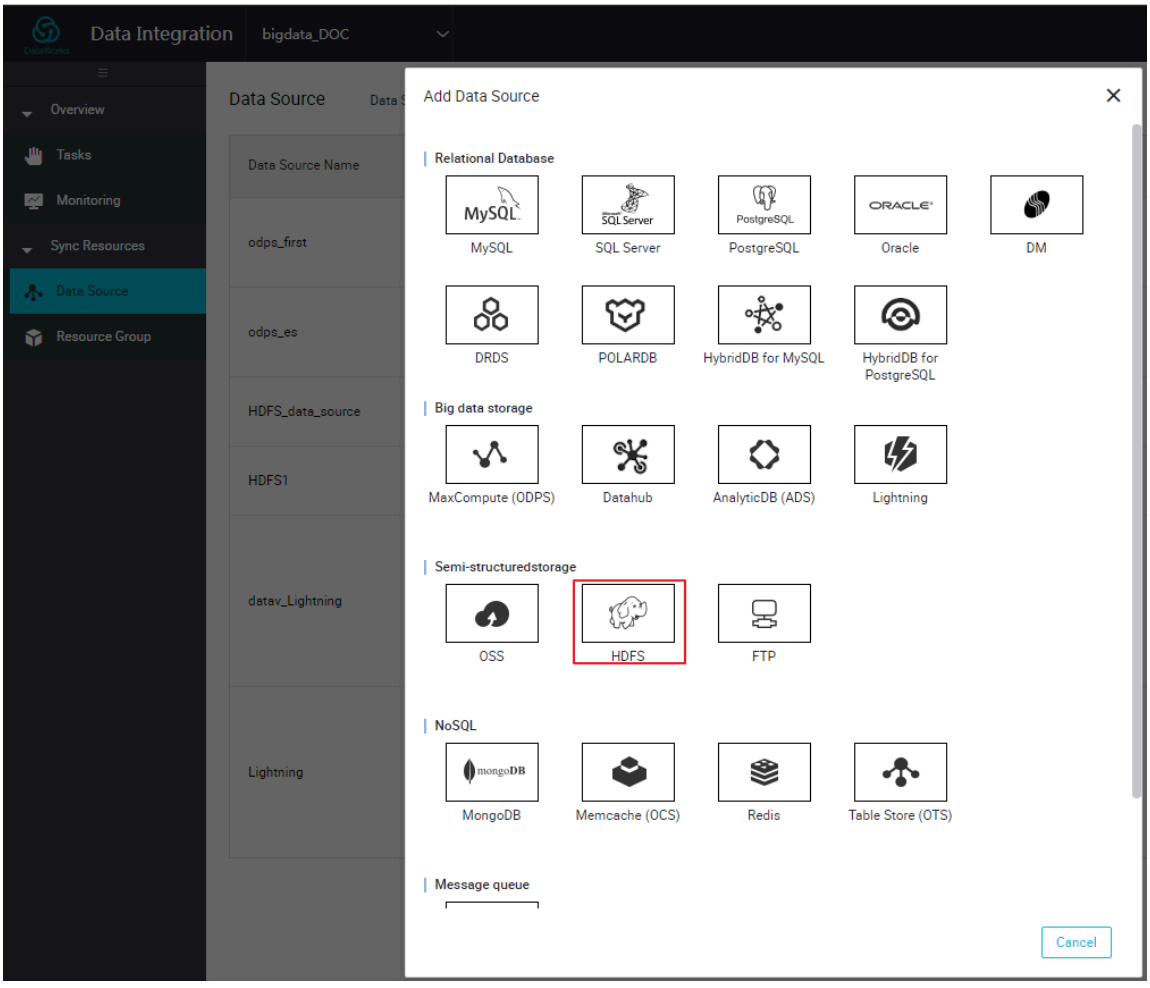
ステータスが unavailable の場合、マスターノードにログインし、`tail -f /home/admin/alisatasknode/logs/heartbeat.log` コマンドを実行し

て、DataWorks とマスターノードの間のハートビートメッセージがタイムアウトしているかどうかを確認します。

```
[root@emr-header-1 logs]# hdfs dfs -ls /user/hive/warehouse/hive_doc_good_sale/
Found 1 items
drwxr-x--x   - hive hadoop          0 2018-09-03 17:46 /user/hive/warehouse/hive_doc_good_sale/pt=1
[root@emr-header-1 logs]# tail -f /home/admin/alisatasknode/logs/heartbeat.log
2018-09-06 21:47:34,440 INFO  [pool-6-thread-1] [HeartbeatReporter.java:104] [] - heartbeat start, current status:2
2018-09-06 21:47:34,465 INFO  [pool-6-thread-1] [HeartbeatReporter.java:133] [] - heartbeat end■ cost time:0.025s
2018-09-06 21:47:39,465 INFO  [pool-6-thread-1] [HeartbeatReporter.java:104] [] - heartbeat start, current status:2
2018-09-06 21:47:39,491 INFO  [pool-6-thread-1] [HeartbeatReporter.java:133] [] - heartbeat end■ cost time:0.026s
2018-09-06 21:47:44,491 INFO  [pool-6-thread-1] [HeartbeatReporter.java:104] [] - heartbeat start, current status:2
2018-09-06 21:47:44,515 INFO  [pool-6-thread-1] [HeartbeatReporter.java:133] [] - heartbeat end■ cost time:0.024s
2018-09-06 21:47:49,516 INFO  [pool-6-thread-1] [HeartbeatReporter.java:104] [] - heartbeat start, current status:2
2018-09-06 21:47:49,538 INFO  [pool-6-thread-1] [HeartbeatReporter.java:133] [] - heartbeat end■ cost time:0.022s
2018-09-06 21:47:54,539 INFO  [pool-6-thread-1] [HeartbeatReporter.java:104] [] - heartbeat start, current status:2
2018-09-06 21:47:54,555 INFO  [pool-6-thread-1] [HeartbeatReporter.java:133] [] - heartbeat end■ cost time:0.016s
^C
```

3. データソースの作成

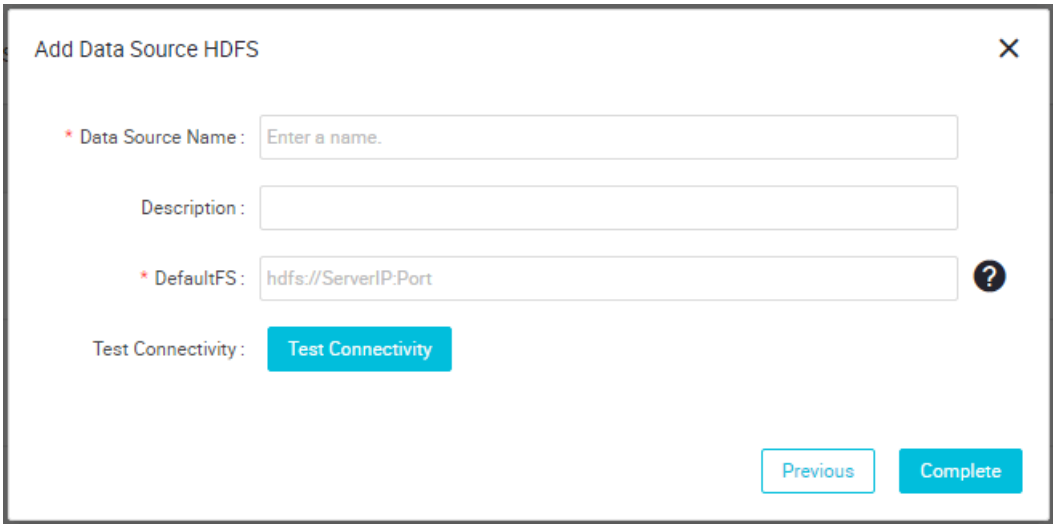
   a. DataWorks の [Data Integration] ページで、[Data Sources] > [New source] をクリックし、[HDFS] を選択します。



   b. [New HDFS Data Sources] パネルで、Name と defaultFS パラメーターを設定します。

> **注：**
> EMR Hadoop クラスターでは、非 HA クラスターの場合、アドレスは `hdfs :// emr - header - 1 的 IP : 9000` に設定されます。HA クラスターの場合、アドレ

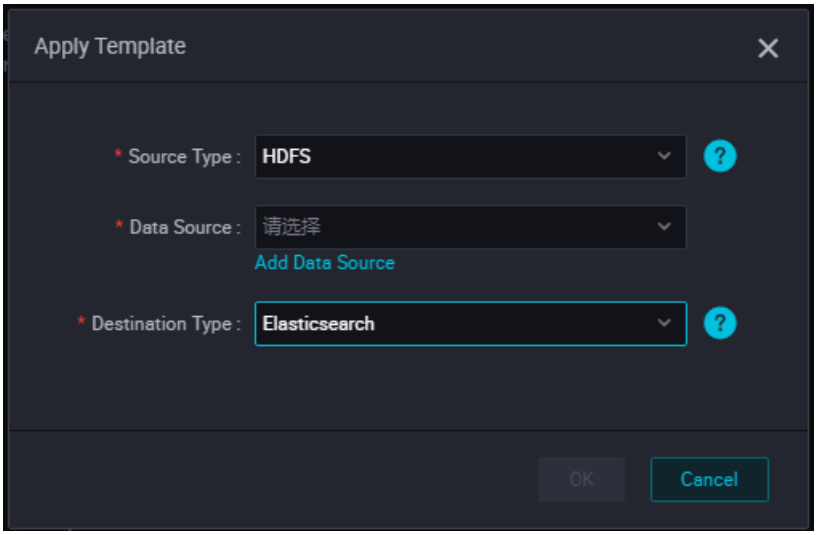
スは `hdfs://emr-header-1的IP:8020` に設定されます。この例では、emr-header-1 と DataWorks が VPC ネットワークを介して接続されているため、イントラネット IP が設定され、[Test Connectivity] は利用できません。



4. データ同期タスクの設定

   a. [Data Integration] ページの左側のナビゲーションウィンドウで、[Sync Tasks] をクリックし、[New] > [Script Mode] を選択します。

   b. [Import template] パネルで、次のデータソースタイプを選択します。



   c. テンプレートがインポートされた後、同期タスクはスクリプトモードに変換されます。次の図は、このトピックで使用されている設定スクリプトを示します。詳細は、「ス

クリプトモードの設定」をご参照ください。 Elasticsearch 設定ルールの詳細は、「ElasticSearch Writer の設定」をご参照ください。

Hdfs Reader

```
{
  "configuration": {
    "reader": {
      "plugin": "hdfs",
      "parameter": {
        "path": "/user/hive/warehouse/hive_esdoc_good_sale/",
        "datasource": "HDFS_data_source",
        "column": [
          {
            "index": 0,
            "type": "string"
          },
          {
            "index": 1,
            "type": "string"
          },
          {
            "index": 2,
            "type": "string"
          },
          {
            "index": 3,
            "type": "string"
          },
          {
            "index": 4,
            "type": "long"
          },
          {
            "index": 5,
            "type": "double"
          },
          {
            "index": 6,
            "type": "long"
          }
        ],
        "defaultFS": "hdfs://[redacted]:9000",
        "fieldDelimiter": ",",
        "encoding": "UTF-8",
        "fileType": "text"
      }
    },
    "writer": {
      "plugin": "elasticsearch",
      "parameter": {
        "accessId": "[redacted]",
        "endpoint": "http://es-cn-[redacted].com:9200",
        "indexType": "elasticsearch",
        "accessKey": "[redacted]",
        "cleanup": true,
        "discovery": false,
        "column": [
          {
            "name": "create_time",
            "type": "string"
          },
          {
            "name": "category",
            "type": "string"
          },
          {
            "name": "brand",
            "type": "string"
          },
          {
            "name": "buyer_id",
            "type": "string"
          },
          {
            "name": "trans_num",
            "type": "long"
          },
          {
            "name": "trans_amount",
            "type": "double"
          },
          {
            "name": "click_cnt",
            "type": "long"
          }
        ],
        "index": "hive_doc_esgood_sale",
```

- 同期スクリプトの設定は 3 つの部分に分けられます。Reader はアップストリームデータソース (データ同期用のクラウドプロダクト) の設定、Writer は ES の設定、setting はパケットロス率や最大同時性などの同期の設定です。
- *path* パラメーターは、Hadoop クラスターでデータが保存されている場所を示します。マスターノードにログインし、`hdfs dfs -ls /user/hive/warehouse/hive_doc_good_sale` コマンドを実行して場所を確認します。パーティションテーブルの場合、パーティションを指定する必要はありません。DataWorks のデータ同期機能では、パーティションパスを自動的に繰り返し処理します。
- Elasticsearch は timestamp 型をサポートしていないため、このトピックの例では、`creat_time` フィールドの型を string に設定しています。
- `endpoint` は、Elasticsearch インスタンスのイントラネットまたはインターネットの IP アドレスです。イントラネットアドレスを使用している場合、Elasticsearch クラスター設定ページの Elasticsearch ホワイトリストに IP を追加する必要があります。インターネット IP を使用している場合、Elasticsearch パブリックネットワークアクセスホワイトリスト (DataWorks のサーバー IP アドレスと、使用するリソースグループの IP を含む) を設定する必要があります。
- Elasticsearch Writer の `accessId` と `accessKey` は、それぞれ Elasticsearch アクセスユーザー名 (デフォルトでは elastic) とパスワードです。
- `index` はElasticsearch データへのアクセスに必要な Elasticsearch インスタンスのインデックスです。
- 同期タスクを作成するとき、DataWorks のデフォルトの設定スクリプトでは、`errorLimit` の `record` フィールド値に 0 が設定されています。より大きな数値 (1,000 など) に変更する必要があります。

5. 前述の設定が完了した後、右上隅の [configuration tasks resources group] をクリックし、[Run] をクリックします。

   タスクの実行に成功しましたというメッセージが表示されたら、タスクが同期されたことを示します。タスクの実行に失敗した場合、トラブルシューティングのためにエラーログをコピーしてください。

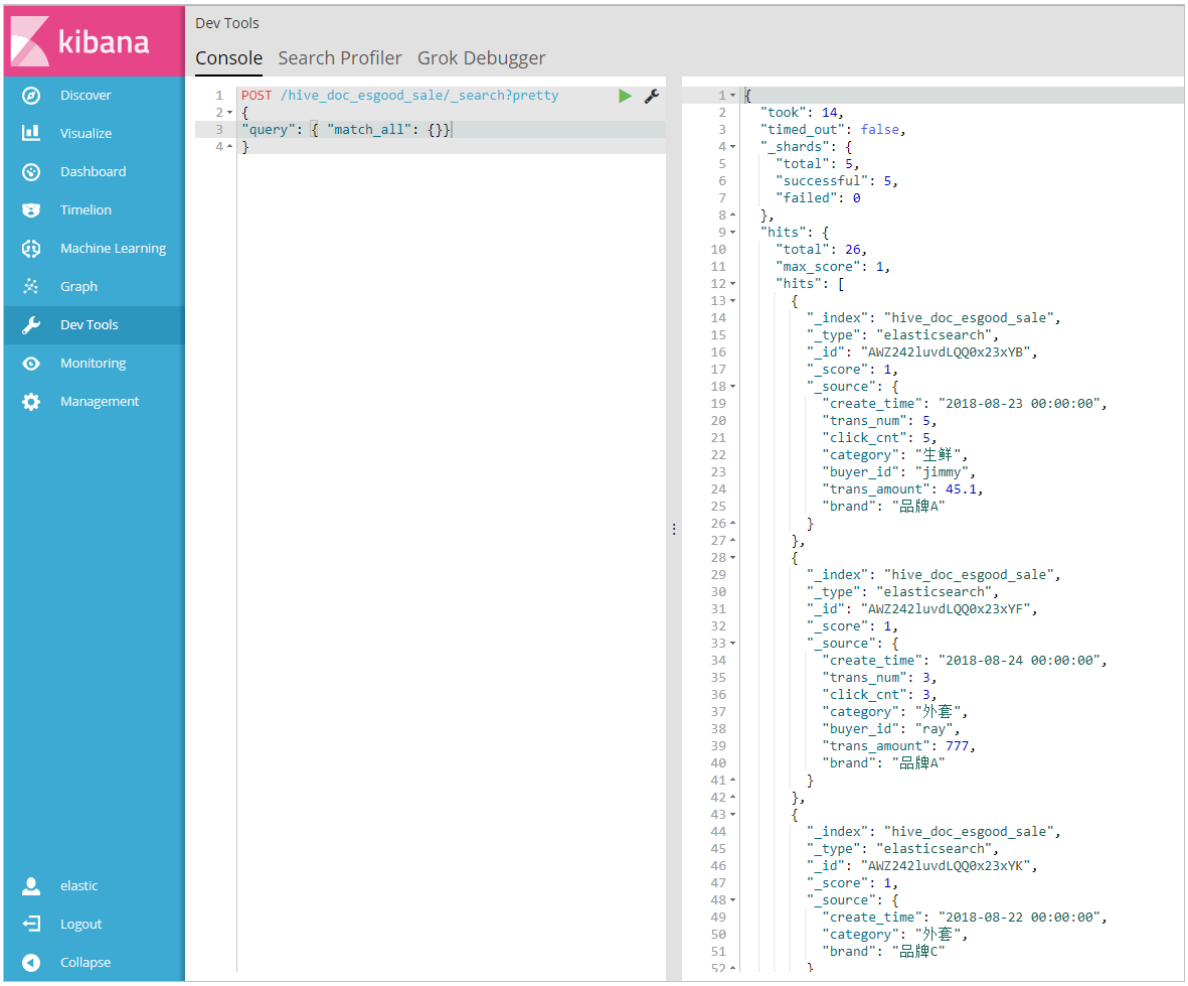
同期結果の確認

1. Elasticsearch コンソールに移動し、右上隅にある Kibana コンソールをクリックし、[Dev Tools] を選択します。
2. 次のコマンドを実行して、同期されたデータを表示します。

```
POST /hive_doc_good_sale/_search?pretty
```

```
{
" query ": { " match_all ": {}}
}
```

`hive_doc_e sgood_sale` は、データを同期するときの `index` フィールドの値です。

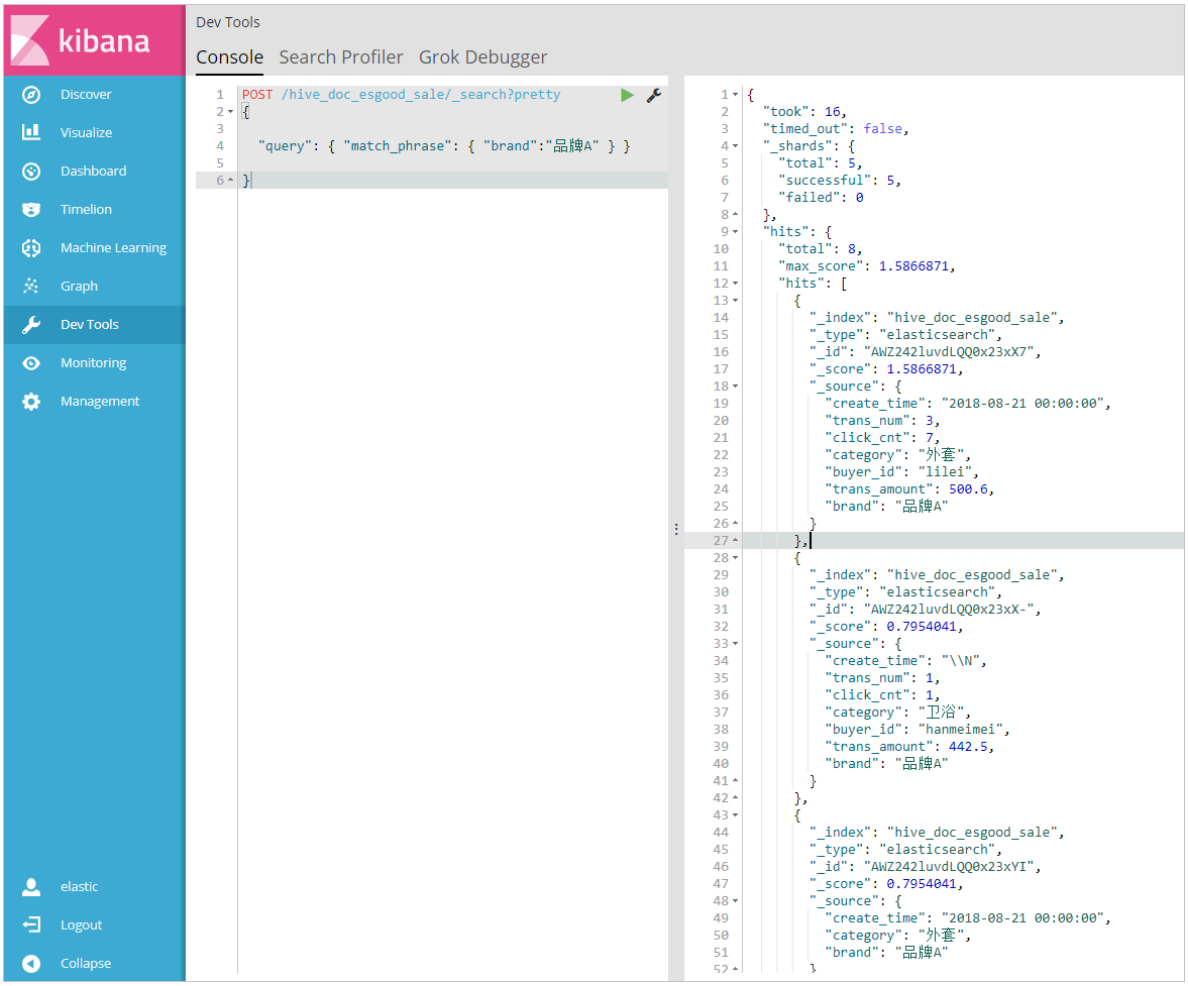


データのクエリと分析

1. 次の例では、Brand A のすべてのドキュメントが返されます。

```
 POST  / hive_doc_e  sgood_sale / _search ?  pretty
{
  " query ": { " match_phra  se ": { " brand ":" Brand   A " } }
```

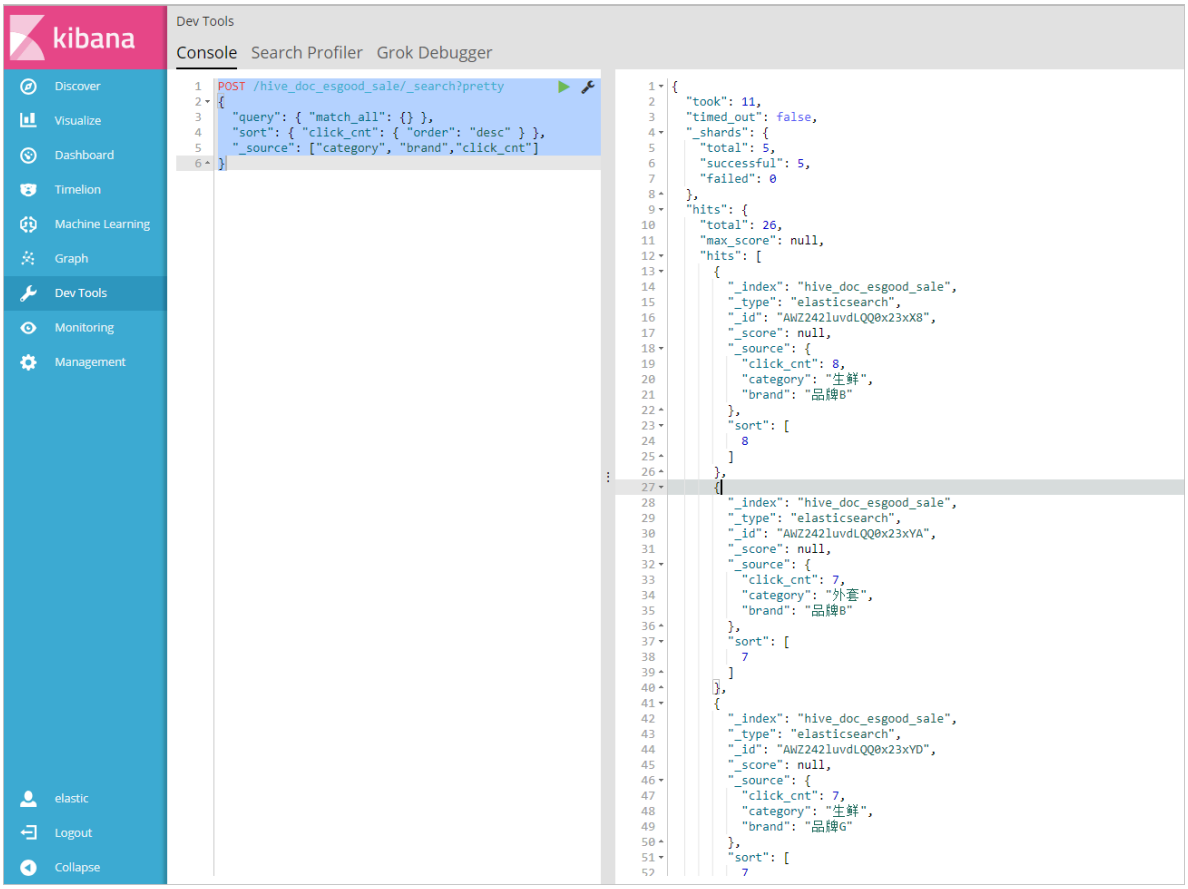
```
}
```



2. **次の例では、すべてのブランドの注目度を表示するため、さまざまなドキュメントをクリック回数でソートしています。**

```
 POST  / hive_doc_e  sgood_sale / _search ?  pretty
{
" query ": { " match_all ": {} },
" sort ": { " click_cnt ": { " order ": " desc " } },
" _source ": [" category ", " brand "," click_cnt "]
```

```
}
```



コマンドやアクセス方法の詳細は、Elasticsearch ドキュメントと Elastic.co help center をご参照ください。

3.3 ApsaraDB RDS for MySQL データベースから Elasticsearch インスタンスへのデータの同期、およびデータのクエリと分析

Alibaba Cloud は、幅広いクラウドストレージとデータベースサービスを提供します。 これらのサービスに保存されているデータの分析や検索をする場合、Data Integration を使用してデータを Elasticsearch にレプリケートしてから、データのクエリや分析をします。 Data Integration を使用すると、最小 5 分間隔でデータをレプリケートできます。

> 注：
> データのレプリケートにより、パブリックネットワークトラフィックが生成され、課金が発生する可能性があります。

前提条件

オンプレミスデータを分析またはクエリする前に、次のタスクを実行してください。

- データベースを作成します。 ApsaraDB RDS for MySQL データベースを使用するか、ローカルサーバーにデータベースを作成することができます。 この例では、ApsaraDB RDS for MySQL を使用します。 次の図は、データベースに保存されているデータセットを示しています。

create_time	category	brand	buyer_id	trans_num	trans_amount	click_cnt
2018-08-21 00:00:00	Outside	B	1	3	500.6	7
2018-08-22 00:00:00	Raw	B	1	1	303	8
2018-08-22 00:00:00	Outside	B	h	2	510	2
1970-01-01 08:00:00	Guard	B	h	1	442.5	1
2018-08-22 00:00:00	Raw	B	h	2	234	3
2018-08-23 00:00:00	Outside	B	j	9	2000	7
2018-08-23 00:00:00	Raw	B	j	5	45.1	5
2018-08-23 00:00:00	Outside	B	j	5	100.2	4
2018-08-24 00:00:00	Raw	B	p	10	5560	7
2018-08-24 00:00:00	Guard	B	p	1	445.6	2
2018-08-24 00:00:00	Outside	B	r	3	777	3
2018-08-24 00:00:00	Guard	B	r	3	122	3
2018-08-24 00:00:00	Outside	B	r	1	62	7

- Elasticsearch インスタンスと同じ VPC ネットワークに接続されている Elastic Compute Service (ECS) インスタンスを購入します。 この ECS インスタンスを使用して、データソースからデータを取得し、そのデータを Elasticsearch インスタンスに書き込むタスクを実行します。 タスクは Data Integration によって送信されます。
- Data Integration を有効化し、同期タスクを実行するためのリソースとして ECS インスタンスを Data Integration に追加します。
- データ同期スクリプトを設定し、定期的に実行します。
- Data Integration で同期されたデータを保存するための Elasticsearch インスタンスを作成します。

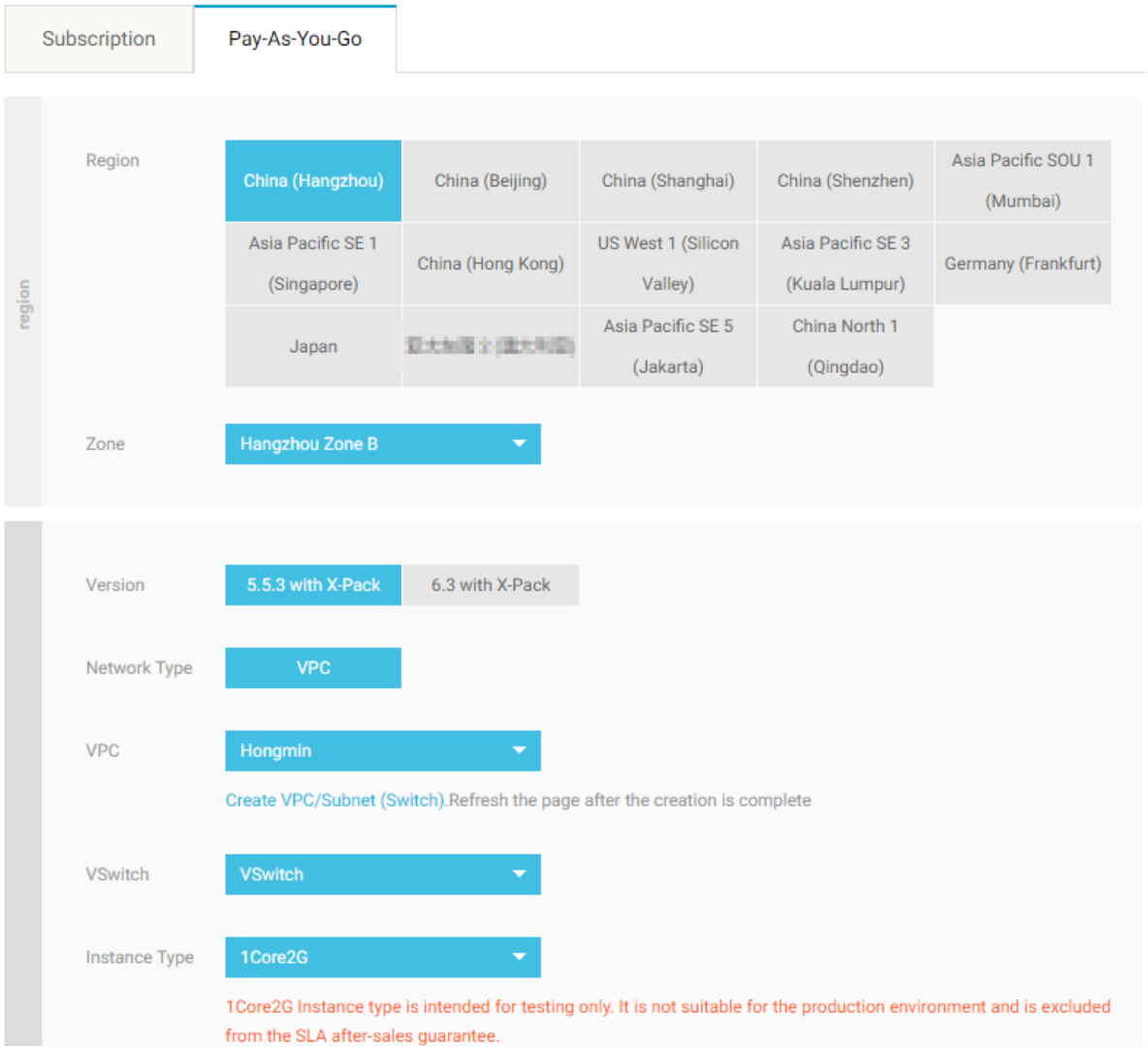
手順

データの同期

1. VPCの作成を行います。
2. Elasticsearch コンソールにログインし、[作成] をクリックして、Elasticsearch インスタンスを作成します。

注：

Elaticsearch インスタンスに指定するリージョン、VPC ネットワーク、vSwitch は、手順 1 で作成した VPC ネットワークと同じでなければなりません。



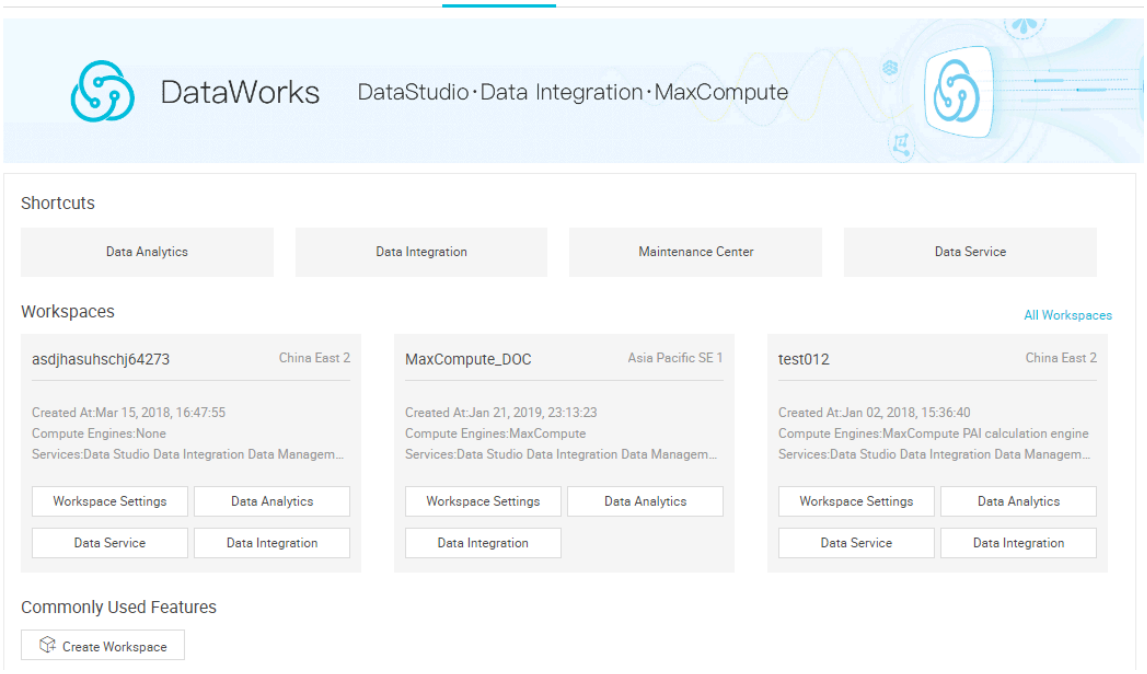
3. Elasticsearch インスタンスと同じ VPC ネットワークに接続されている ECS インスタンスを購入し、パブリック IP アドレスを ECS インスタンスに割り当てるか、Elastic IP Address (EIP) を有効化します。 コストを節約するため、要件を満たす既存の ECS インスタンスを使用することを推奨します。

注：

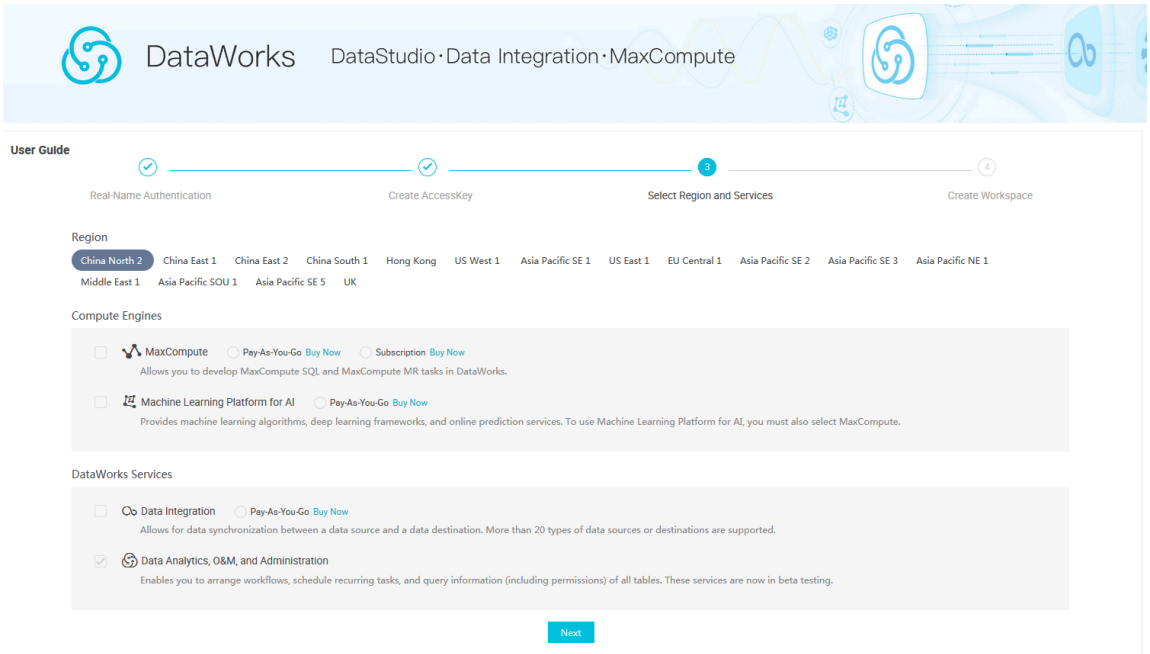
- CentOS 6、CentOS 7、または AliyunOS を使用することを推奨します。
- ECS インスタンスで MaxCompute タスクまたはデータ同期タスクを実行する場合、ECS インスタンスの Python バージョンが 2.6 か 2.7 であることを確認する必要があります。CentOS 5 の Python バージョンは 2.4 ですが、他の CentOS バージョンの Python バージョンは 2.6 以降です。
- ECS インスタンスにパブリック IP アドレスが割り当てられていることを確認します。

4. DataWorks コンソールにログインします。

   - Data Integration が既に有効化されている場合、次のページが表示されます。

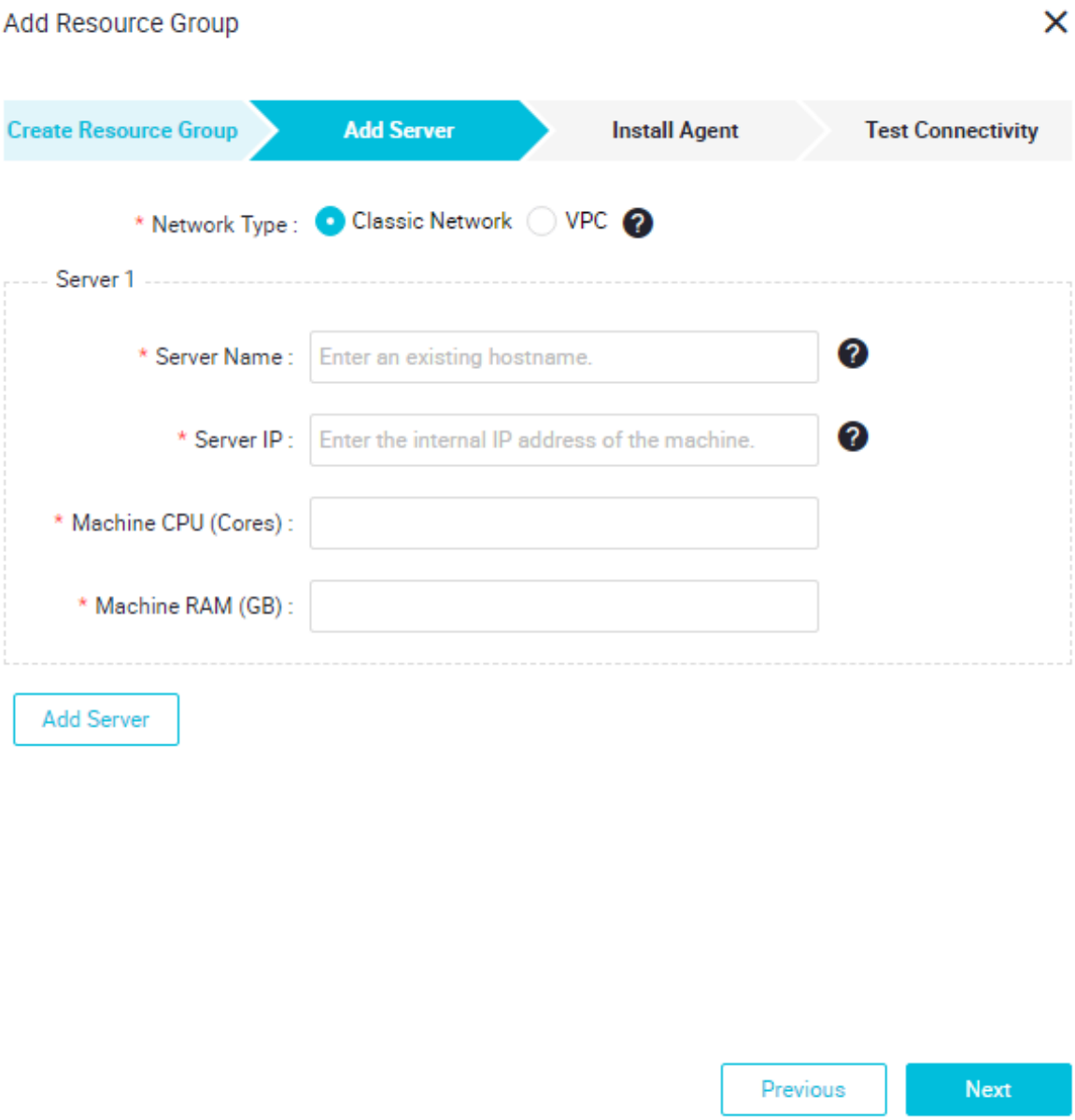


   - Data Integration が有効化されていない場合、次のページが表示されます。Data Integration を有効化するには、次の手順を実行します。 Data Integration を有効化すると、サービス料金が発生します。 課金項目に基づいて料金を見積もることができます。



5. [Data Integration] をクリックします。
6. [Data Integration] ページの左側のナビゲーションウィンドウで [Resource Group] をクリックし、右上隅の [Add Resource Group] をクリックします。

7. 必要に応じてリソースグループ名とサーバー情報を入力します。 このページで追加するサーバーは、購入した ECS インスタンスです。 次の情報を入力します。



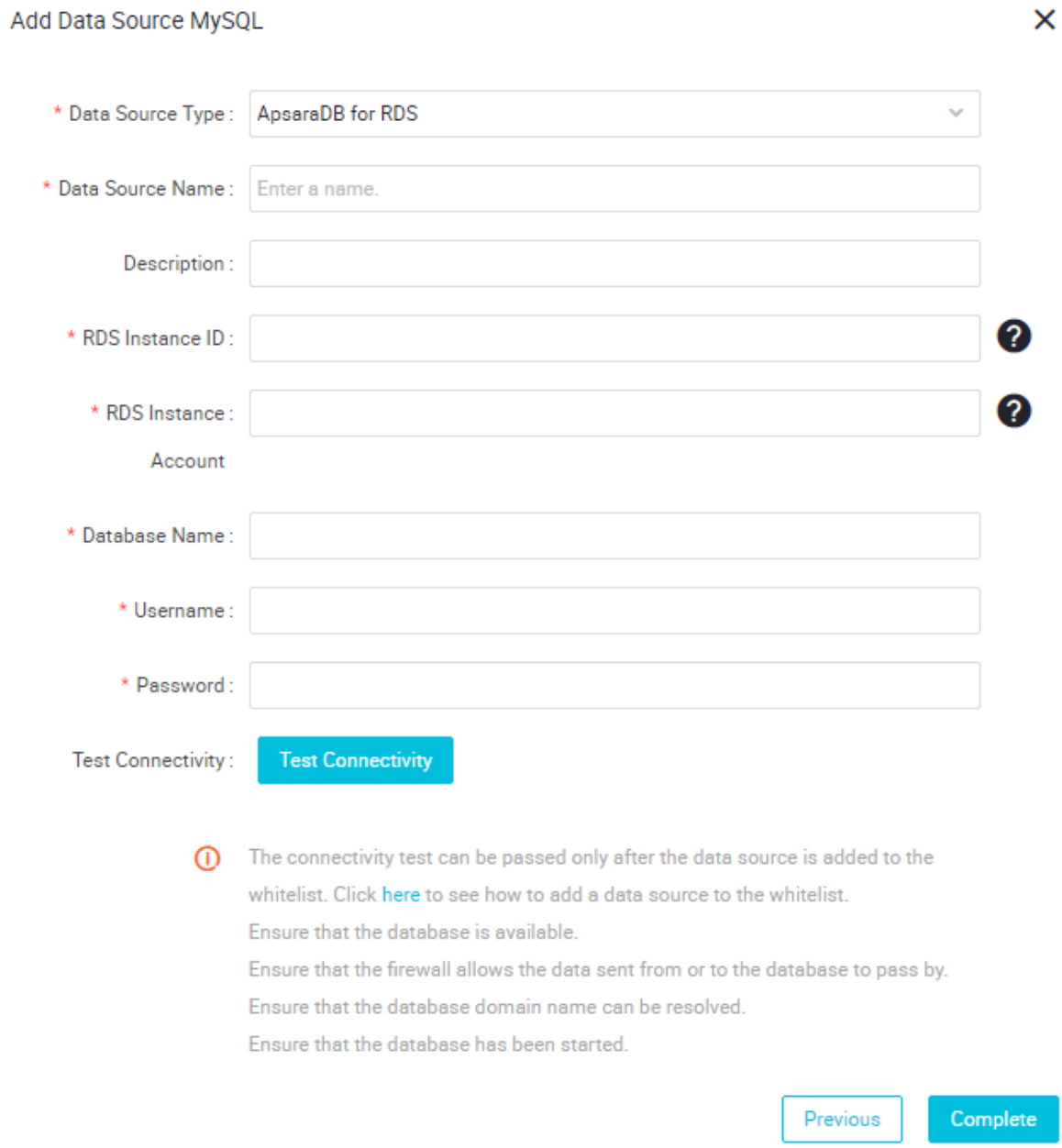
- ECS UUID：ECS インスタンスの UUID を入力します。 ECS インスタンスにログインし、`dmidecode | grep UUID` コマンドを実行して UUID を取得します。 詳細は、「手順 3: インスタンスへの接続」をご参照ください。
- Server IP、Machine CPU (Cores)、Machine RAM (GB)：ECS インスタンスのパブリック IP アドレス、CPU サイズ、メモリサイズを入力します。 情報を取得するに

は、ECS コンソールにログインし、ECS インスタンス名をクリックします。 [設定情報] フィールドに、情報が表示されます。

- ページの指示に従って、エージェントをインストールします。 手順 5 で、ECS インスタンスのポート 8000 は開かれています。 デフォルトの設定を使用すると、この手順をスキップできます。

8. データベースホワイトリストを設定します。 リソースグループの IP アドレスと ECS インスタンスの IP アドレスをホワイトリストに追加します。 ホワイトリスト設定の詳細は、「ホワイトリストの追加」をご参照ください。
9. リソースグループを作成した後、左側のナビゲーションウィンドウで [Data Source] をクリックし、右上隅の [Add Data Source] をクリックします。

10.[MySQL] を選択します。 [Add Data Source MySQL] ページで、必要な情報を入力します (次図)。



Data Source Type：この例では、ApsaraDB RDS for MySQL データベースを使用します。 [Public IP Address Available] または [Public IP Address Unavailable] を選択します。 パラメーターの詳細は、「MySQL データソースの設定」をご参照ください。

11.左側のナビゲーションウィンドウで [Sync Resources] をクリックし、[Create Task] をクリックします。 [Script Mode] を選択します。

12.[Apply Template] ダイアログボックスで、[Source Type] > [MySQL] を選択します。 手順 10 で [Data Source] フィールドに追加したデータソースの名前を入力し、[Destination

Type] ドロップダウンリストから [Elasticsearch] を選択します。 情報を入力して [OK] をクリックします。

13.データ同期スクリプトを設定します。設定の詳細は、「スクリプトモードの設定」をご参照ください。 Elasticsearch インスタンス設定ルールの詳細は、「ElasticSearch Writer の設定」をご参照ください。

```
{
  "configuration": {
    "reader": {
      "plugin": "mysql",
      "parameter": {
        "datasource": "es_test_rdsmysql",
        "column": [
          "create_time",
          "category",
          "brand",
          "buyer_id",
          "trans_num",
          "trans_amount",
          "click_cnt"
        ],
        "where": "",
        "splitPk": "",
        "table": "good_sale"
      }
    },
    "writer": {
      "plugin": "elasticsearch",
      "parameter": {
        "accessId": "elastic",
        "endpoint": "http://es-cn-                                    aliyuncs.com:9200",
        "indexType": "elasticsearch",
        "accessKey": "          ",
        "cleanup": false,
        "discovery": false,
        "column": [
          {
            "name": "create_time",
            "type": "date"
          },
          {
            "name": "category",
            "type": "string"
          },
          {
            "name": "brand",
            "type": "string"
          },
          {
            "name": "buyer_id",
            "type": "string"
          },
          {
            "name": "trans_num",
            "type": "long"
          },
          {
            "name": "trans_amount",
            "type": "double"
          },
          {
            "name": "click_cnt",
            "type": "long"
          }
        ],
        "index": "testrds",
        "batchSize": 1000,
        "splitter": ","
      }
    },
    "setting": {
      "errorLimit": {
        "record": "0"
      },
      "speed": {
        "throttle": false,
        "concurrent": 1,
        "mbps": "1",
        "dmu": 1
      }
    }
  }
}
```

注：

- データ同期スクリプトには、reader、writer、settings の 3 つのセクションがあります。 reader セクションには、同期するデータを保存するデータソース (クラウドリソース) の設定が含まれています。 writer セクションには、Elasticsearch インスタンスの設定が含まれています。 settings セクションには、パケットロスしきい値や最大同時数などのデータ同期設定が含まれています。
- endpoint を Elasticsearch インスタンスの内部またはパブリック IP アドレスに設定できます。 内部 IP アドレスを使用する場合、[Elasticsearch Cluster Configuration] ページで Elasticsearch インスタンスのシステムホワイトリストを設定する必要があります。 パブリック IP アドレスを使用する場合、パブリック IP アドレスからアクセスできるように [Security] ページで Elasticsearch インスタンスのホワイトリストを設定する必要があります。 ホワイトリストには、DataWorks に追加された ECS インスタンスの IP アドレスと、使用するリソースグループの IP アドレスを含める必要があります。
- writer セクションの accessId と accessKey パラメーターを、それぞれ Elasticsearch インスタンスのユーザー名とパスワードに設定します。
- writer セクションの index パラメーターを Elasticsearch インスタンスのインデックスに設定します。 このインデックスは、Elasticsearch インスタンスに保存されているデータへのアクセスに使用されます。

14.同期スクリプトを設定した後、ページの右側にある [Configure Resource Group] をクリックし、手順 7 で作成したリソースグループを選択します。 確認し、[Run] をクリックして、MySQL データベースから Elasticsearch インスタンスにデータをレプリケートします。

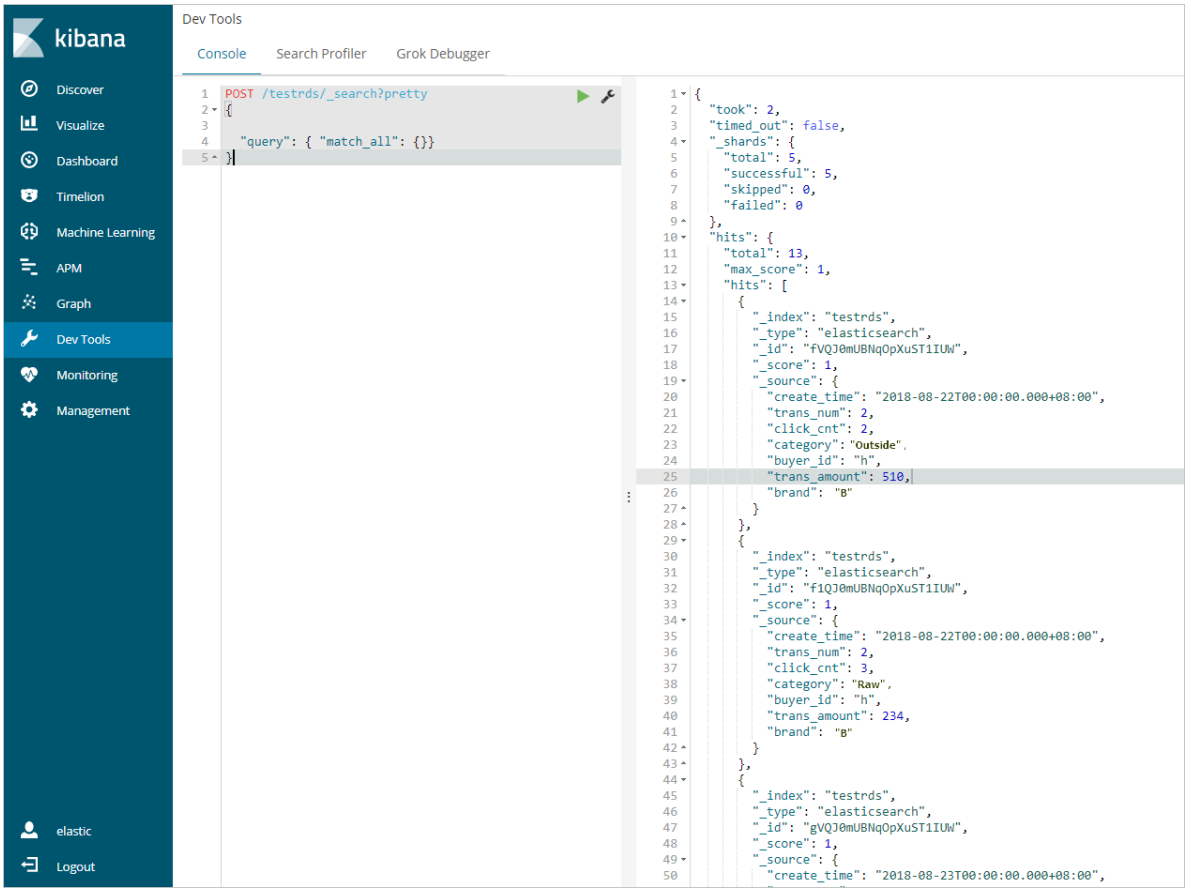
データのクエリと分析

1. Elasticsearch コンソールにログインし、右上隅の Kibana コンソールをクリックし、[Dev Tools] をクリックします。
2. 以下のコマンドを実行し、同期されたデータを表示します。

```
 POST  / testrds / _search ?  pretty
{
" query ": { " match_all ": {}}
```

```
}
```

testrds は、データ同期スクリプトの index パラメーターに指定された値です。



3. 次のコマンドを実行して、 trans_num 列を基準にデータをソートします。

```
 POST  / testrds / _search ?  pretty
{
" query ": { " match_all ": {} },
" sort ": { " trans_num ": { " order ": " desc " } }
}
```

4. 次のコマンドを実行して、データ内の category 列と brand 列をクエリします。

```
 POST  / testrds / _search ?  pretty
{
" query ": { " match_all ": {} },
" _source ": [" category ", " brand "]
}
```

5. 次のコマンドを実行して、 category 列が Raw に設定されているデータエントリをクエリします。

```
 POST  /  testrds  /  _search ?  pretty
{
" query ": { " match ": {" category ":" Raw "} }
```

```
}
```


```
{
  "took": 10,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 4,
    "max_score": 0.6931472,
    "hits": [
      {
        "_index": "testrds",
        "_type": "elasticsearch",
        "_id": "f1QJ0mUBNqOpXuST1IUW",
        "_score": 0.6931472,
        "_source": {
          "create_time": "2018-08-22T00:00:00.000+08:00",
          "trans_num": 2,
          "click_cnt": 3,
          "category": "Raw",
          "buyer_id": "h",
          "trans_amount": 234,
          "brand": "B"
        }
      },
      {
        "_index": "testrds",
        "_type": "elasticsearch",
        "_id": "gVQJ0mUBNqOpXuST1IUW",
        "_score": 0.6931472,
        "_source": {
          "create_time": "2018-08-23T00:00:00.000+08:00",
          "trans_num": 5,
          "click_cnt": 5,
          "category": "Raw",
          "buyer_id": "j",
          "trans_amount": 45.1,
          "brand": "B"
        }
      },
      {
        "_index": "testrds",
        "_type": "elasticsearch",
        "_id": "g1QJ0mUBNqOpXuST1IUW",
        "_score": 0.6931472,
        "_source": {
          "create_time": "2018-08-24T00:00:00.000+08:00",
          "trans_num": 10,
```

Elasticsearch へのアクセス方法の詳細は、「Elasticsearch アクセスのテスト」と「Elastic ドキュメント」をご参照ください。

よくある質問

- データベースへのアクセス中にエラーが発生しました。

  解決策：リソースグループ内の ECS インスタンスの内部 IP アドレスとパブリック IP アドレスを DataWorks データベースホワイトリストに追加します。

- Elasticsearch インスタンスへのアクセス中にエラーが発生しました。

  解決策：次の手順を実行します。

  1. 前の手順で作成したリソースグループを [Configure Resource Group] で選択しているかどうかを確認します。
     - 正しいリソースグループを選択している場合は、次の手順に進みます。
     - 正しいリソースグループを選択していない場合は、[Configure Resource Group] をクリックして正しいリソースグループを選択します。 確認し、[Run] をクリックします。
  2. ECS インスタンスの IP アドレスとリソースグループの IP アドレスを Elasticsearch インスタンスのホワイトリストに追加しているかどうかを確認します。
     - これらの IP アドレスをホワイトリストに追加している場合は、次の手順に進みます。
     - これらの IP アドレスをホワイトリストに追加していない場合は、ECS インスタンスの IP アドレスとリソースグループの IP アドレスを Elasticsearch インスタンスのホワイトリストに追加します。

       
       注：
       内部 IP アドレスを使用する場合、 Security ページで、Elasticsearch インスタンスのシステムホワイトリストを設定します。 パブリック IP アドレスを使用する場合、パブリック IP アドレスからアクセスできるように [Security] ページで Elasticsearch インスタンスのホワイトリストを設定します。 ホワイトリストには、ECS インスタンスの IP アドレスとリソースグループの IP アドレスを含める必要があります。
  3. スクリプトの設定が正しいかどうかを確認します。endpoint、accessId、accessKey を確認してください。 endpoint には、Elasticsearch インスタンスの内部またはパブリック IP アドレスを設定する必要があります。 accessId には、Elasticsearch インスタンスのユーザー名を設定する必要があります。 デフォルトの名前は、elastic です。 accessKey には、Elasticsearch インスタンスのパスワードを設定する必要があります。

3.4 RDS for MySQL から ES へのリアルタイムデータ同期

このセクションでは、Data Transmission Service (DTS) を使用して、RDS for MySQL インスタンスから Elasticsearch (ES) インスタンスへのリアルタイムデータ同期タスクを迅速に作成する方法を説明します。 DTS はこの同期機能を使用して RDS for MySQL データを ES インスタンスに同期し、データをリアルタイムでクエリします。

リアルタイム同期タイプ

同じ Alibaba Cloud アカウントの DTS インスタンスで、RDS for MySQL から ES 。

SQL 操作タイプ

サポートされている主な SQL 操作タイプは次のとおりです。

- Insert
- Delete
- Update

> 注 :
> DTS では、DDL 文を使用してデータを同期することはできません。 データを同期する場合、DDL 操作は無視されます。

RDS for MySQL インスタンスで DDL を使用するテーブルが検出されると、対応するテーブルの DML 操作が失敗する可能性があります。 この問題を解決するには、以下の手順を実行します。

1. 同期リストからオブジェクトを削除します。 詳細は、「」「同期オブジェクトの削除」「」をご参照ください。
2. ES インスタンスのこのテーブルに対応するインデックスを削除します。
3. テーブルを同期リストに追加し直し、再初期化してください。 詳細は、「」「同期オブジェクトの追加」「」をご参照ください。

DDL を使用して列を追加したりテーブルを変更したりする場合、DDL 操作順序は次のとおりです。

1. ES インスタンスのマッピングと新しい列を手動で変更します。
2. テーブルスキーマを変更し、ソース RDS for MySQL インスタンスに新しいスキーマを追加します。
3. DTS でインスタンスの同期を停止します。DTS 同期インスタンスを再起動して ES で変更されたマッピング関係を再ロードします。

データ同期の設定

RDS for MySQL インスタンスから ES インスタンスにデータを同期するには、以下の手順を実行します。

1. DTS 同期インスタンスの購入

   Data Transmission Service コンソールにログインし、[データ同期] ウィンドウに移動します。 右上隅にある [同期タスクの作成] をクリックして同期インスタンスを購入します。 同期インスタンスを設定できるようになります。

    注 :

設定する前に同期インスタンスを購入する必要があります。 サブスクリプションと従量課金の 2 つの課金方法がサポートされています。.

購入ページのパラメーター

- 機能

  [データ同期] を選択します。
- ソースインスタンス

  [MySQL] を選択します。
- ソースリージョン

  - この例では、RDS for MySQL インスタンスを使用するため、RDS for MySQL インスタンスが存在するリージョンを選択する必要があります。
- ターゲットインスタンス

  [Elasticsearch] を選択します。
- ターゲットリージョン

  Elasticsearch インスタンスが存在するリージョンを選択します。 同期インスタンスを購入した後、リージョンを変更することはできません。
- 仕様

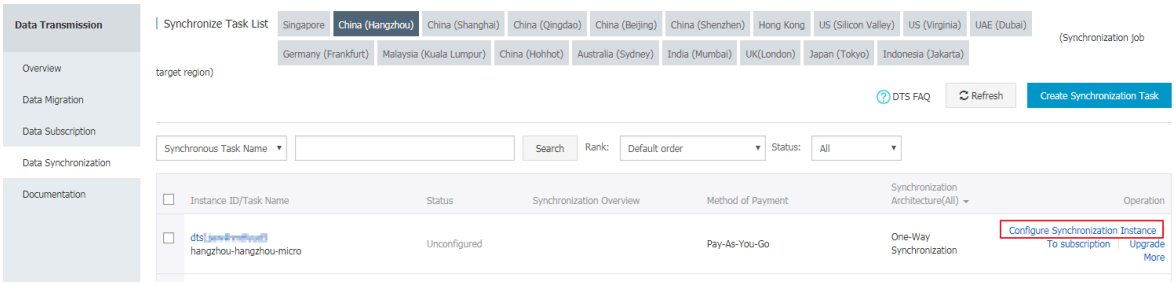
  インスタンス仕様は、同期インスタンスのパフォーマンスに対応しています。 詳細は、「」「データ同期仕様」「」をご参照ください。
- 購入期間

  - 同期インスタンスがサブスクリプションの場合、デフォルトの購入期間は 1 ヶ月です。
- 数

  デフォルトの数量は 1 です。

 注 :
DTS 同期インスタンスのリージョンは、選択したターゲットリージョンです。 たとえば、[中国 (杭州)] リージョンの RDS for MySQL から [中国 (杭州)] リージョンの Elasticsearch への同期インスタンスの場合、DTS 同期インスタンスのリージョンは [中国 (杭州)] です。 同期インスタンスを設定するには、DTS でそのリージョンのインスタンスリストに移動し、購入したばかりの同期インスタンスを検索して、右上隅の [同期インスタンスの設定] をクリックします。

2. 同期インスタンスの設定

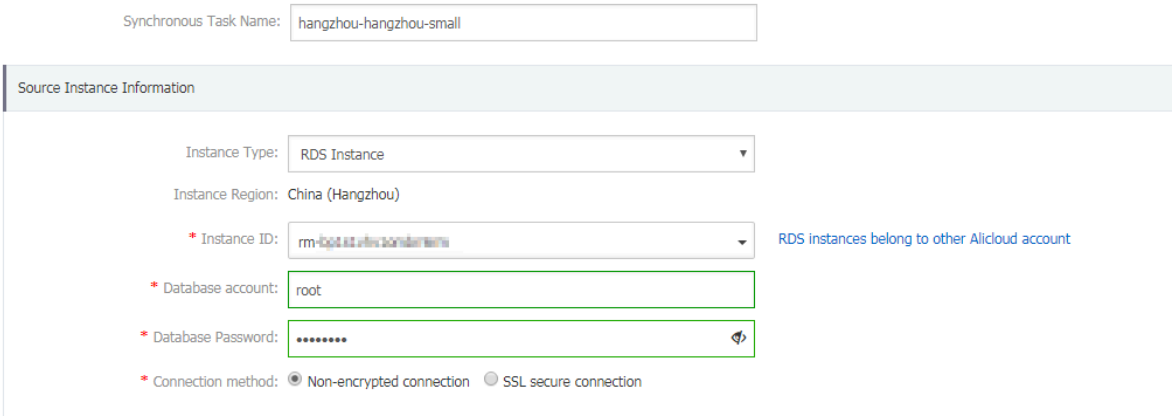


同期タスク名

同期インスタンスの名前に関する要件はありません。

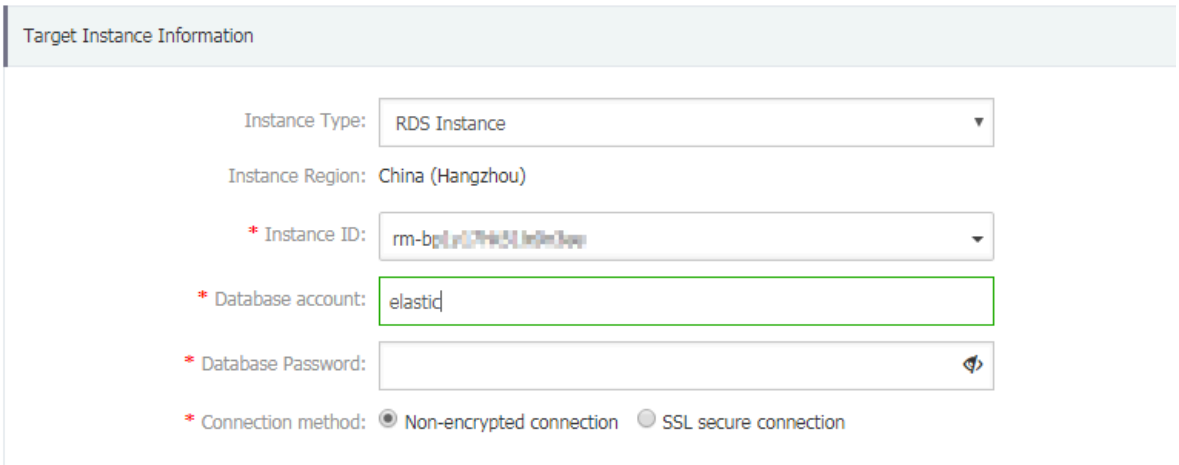
ソースインスタンス

この例では、データソースとして RDS for MySQL を使用します。 インスタンスのタイプ、リージョン、ID、およびデータベースのアカウントとパスワードを設定する必要があります。



ターゲットインスタンス

ES インスタンスの ID、アカウント、パスワードを設定する必要があります。



これらの設定が完了したら、[ホワイトリストを承認して次のステップに進む] をクリックして、RDS for MySQL と ES インスタンスのホワイトリストに IPを追加します。

3. インスタンスのホワイトリストの承認

 注：
ソースインスタンスが RDS for MySQL の場合、自動的に IP がホワイトリストに追加されるか、セキュリティグループが追加されます。

ソースインスタンスが RDS for MySQL の場合、インスタンス IP が RDS インスタンスのホワイトリストのセキュリティグループに追加されます。 これにより、同期タスクを作成するとき、DTS インスタンスと RDS データベース間の切断によって引き起こされるエラーを回避することができます。 同期タスクの安定性を確保するため、RDS インスタンスからインスタンス IP を削除しないでください。

ホワイトリストの承認後、[次へ] をクリックして同期アカウントを作成します。

4. 同期オブジェクトの選択

同期オブジェクトとインデックスの命名規則を設定するには、次の手順を実行します。

a. インデックスの命名規則を [テーブル名] または [データベース名_テーブル名] から選択します。

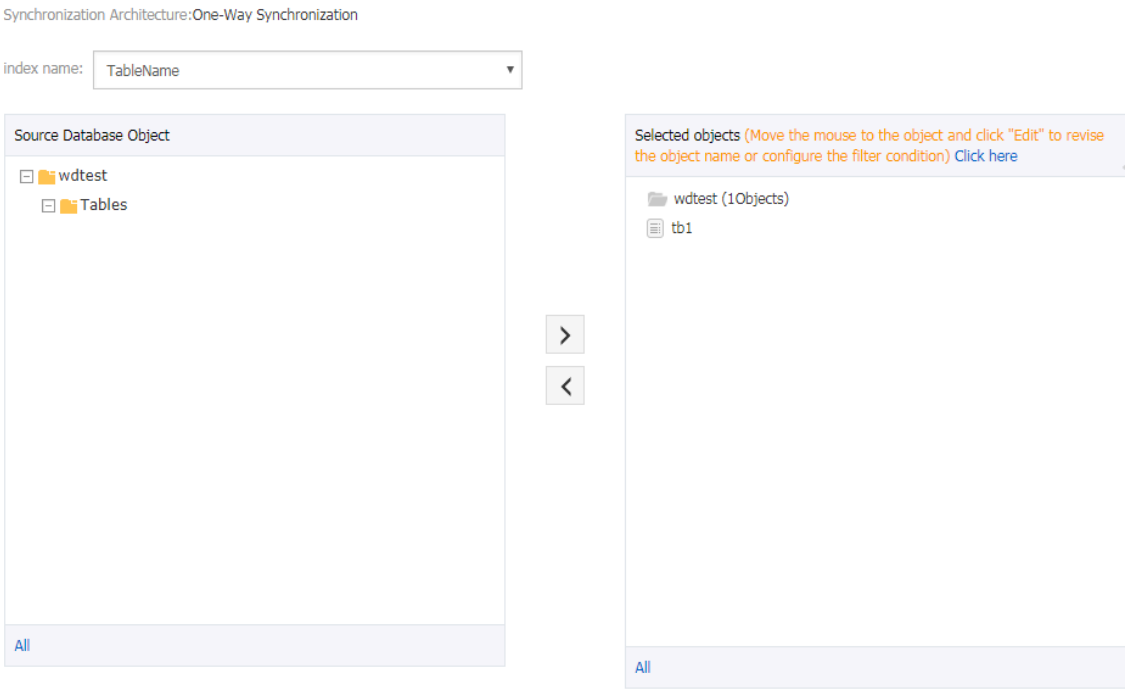
- [テーブル名] を選択した場合、インデックスの名前はテーブルの名前です。
- [データベース名_テーブル名] を選択した場合、インデックスの命名規則はデータベース名_テーブル名です。 たとえば、データベース名が dbtest、テーブル名が sbtest1 の

場合、テーブルが ES インスタンスと同期された後にインデックス名は dbtest_sbtest1 になります。

- 同じ名前の 2 つのテーブルが、それぞれ異なるデータベースにある場合、インデックス名をデータベース名_テーブル名に設定することを推奨します。

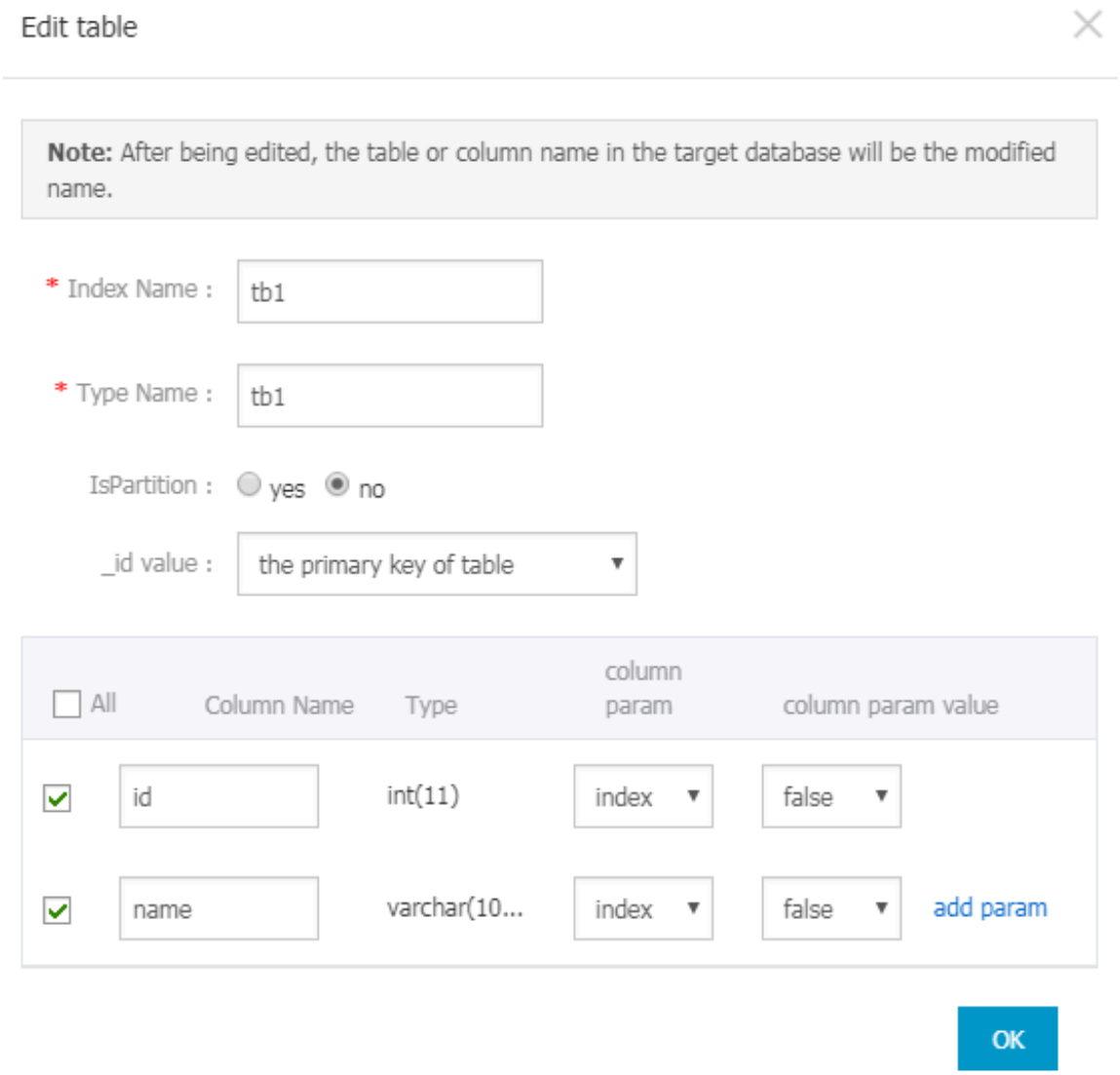
b. 特定のデータベース、テーブル、列を選択します。 同期オブジェクトの選択可能粒度は、テーブルレベルの操作をサポートします。 つまり、複数のデータベースとテーブルを同期できます。

同期オブジェクトの選択可能粒度は、テーブルレベルの操作をサポートします。 つまり、複数のデータベースとテーブルを同期できます。



c. デフォルトでは、すべてのテーブルの docid はプライマリキーです。 プライマリキーを持たないテーブルの場合、ソーステーブルの列に対応する docid を設定します。 右側の [選

択したオブジェクト] のボックスで、対応するテーブルの上にポインターを移動し、[編集] をクリックして [詳細設定] ウィンドウに移動します。



d. [詳細設定] では、インデックス名、タイプ名、パーティション列、数量、および _id 値列を設定できます。 _id の値がビジネスプライマリキーに設定されている場合、対応するビジネスプライマリキー列を選択する必要があります。

e. 同期オブジェクトを設定したら、詳細設定に進みます。

5. 詳細設定

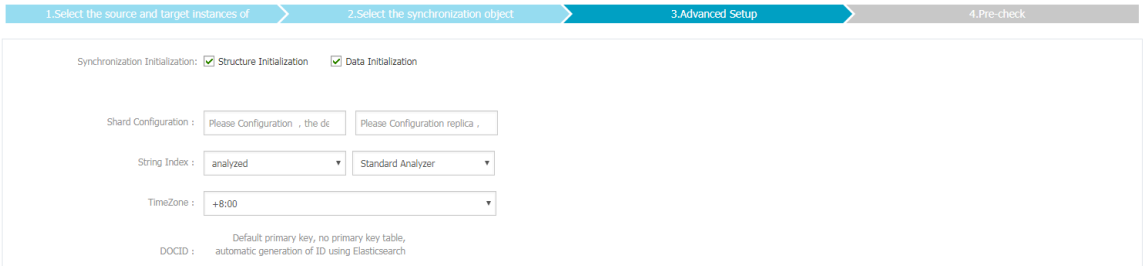
主な設定

a. 同期の初期化：[構造体の初期化] と [データの初期化] を選択することを推奨します。これにより、自動的にインデックスが作成され、データが初期化されます。 [スキーマの初期化] を選択しない場合、同期する前に ES のインデックスのマッピングを手動で定義する必

要があります。 [全データの初期化] を選択しない場合、増分 DTS データ同期の開始時刻は同期が開始される時刻です。

b. シャード設定：デフォルトで 5 つのパーティションと 1 つのレプリカがあります。 設定の調整後、すべてのインデックスがこの設定に従ってパーティションを定義します。

c. 文字列インデックス：文字列を選択できるアナライザーです。 デフォルトは、[Standard Analyzer] です。 その他の値は、[Simple Analyzer]、[Whitespace Analyzer]、[Stop Analyzer]、[Keyword Analyzer]、[English Analyzer]、[Fingerprint Analyzer] です。 すべてのインデックスの文字列フィールドは、この設定に従ってアナライザーを定義します。



d. Time Zone：ES インスタンスに同期している時間フィールドが保存されている場所です。 中国のデフォルトのタイムゾーンは UTC (UTC + 8) です。

6. 事前チェック

同期タスクの設定が完了したら、DTS は事前チェックを実行します。 事前チェックが確認されたら、[開始] をクリックして同期タスクを開始します。

同期タスクが開始したら、同期ジョブリストに移動し、タスクのステータスが [同期初期化] かどうかを確認します。 初期化にかかる時間は、ソースインスタンス内の同期オブジェクトのデータ量によって異なります。 初期化が完了すると、同期インスタンスのステータスは [同期中] になります。 ソースインスタンスとターゲットインスタンス間に同期リンクが確立されます。

7. データの検証

上記のすべての手順を完了したら、ES コンソールにログインして、ES インスタンスに作成されたインデックスと同期データを確認します。

3.5 DataWorks による MaxCompute と Elasticsearch 間のデータ同期

Alibaba Cloud は、幅広いクラウドストレージとデータベースサービスを提供します。 これらのサービスに保存されているデータの分析や検索をする場合、Data Integration を使用し

てデータを Elasticsearch にレプリケートしてから、データのクエリや分析をします。 Data Integration を使用すると、最小 5 分間隔でデータをレプリケートできます。

> 注：
> データのレプリケートにより、パブリックネットワークトラフィックが生成され、課金が発生する可能性があります。

前提条件

オンプレミスデータの分析や検索をするには、次の手順に従います。

- テーブルの作成と表示、およびデータのインポートを行います。 Hadoop から MaxCompute へデータを移行してから、データを同期します。 この例では、以下のテーブルスキームとデータを使用します。

Column Name	Type
create_time	STRING
category	STRING
brand	STRING
buyer_id	STRING
trans_num	BIGINT
trans_amount	DOUBLE
click_cnt	BIGINT
pt	STRING

create_time	category	brand	buyer_id	trans_num	trans_amount	click_cnt	pt
2018-08-21 00:00:00	[illegible]	[illegible]A	null	null	null	null	1
2018-08-22 00:00:00	[illegible]	[illegible]B	null	null	null	null	1
2018-08-22 00:00:00	[illegible]	[illegible]C	null	null	null	null	1
	[illegible]	[illegible]A	null	null	null	null	1
2018-08-22 00:00:00	[illegible]	[illegible]D	null	null	null	null	1
2018-08-23 00:00:00	[illegible]	[illegible]B	null	null	null	null	1
2018-08-23 00:00:00	[illegible]	[illegible]A	null	null	null	null	1
2018-08-23 00:00:00	[illegible]	[illegible]E	null	null	null	null	1
2018-08-24 00:00:00	[illegible]	[illegible]G	null	null	null	null	1
2018-08-24 00:00:00	[illegible]	[illegible]F	null	null	null	null	1
2018-08-24 00:00:00	[illegible]	[illegible]A	null	null	null	null	1
2018-08-24 00:00:00	[illegible]	[illegible]G	null	null	null	null	1
2018-08-24 00:00:00	[illegible]	[illegible]C	null	null	null	null	1

- Data Integration でレプリケートされたデータを保存するための Elasticsearch インスタンスを作成します。

- Elasticsearch と同じ VPC を共有する ECS インスタンスを購入します。 この ECS インスタンスでデータを取得し、Elasticsearch タスクを実行します (これらのタスクは、Data Integration によって送信されます)。
- Data Integration を有効化し、タスクを実行可能なリソースとして ECS インスタンスを Data Integration に登録します。
- データ同期スクリプトを設定し、定期的に実行します。

手順

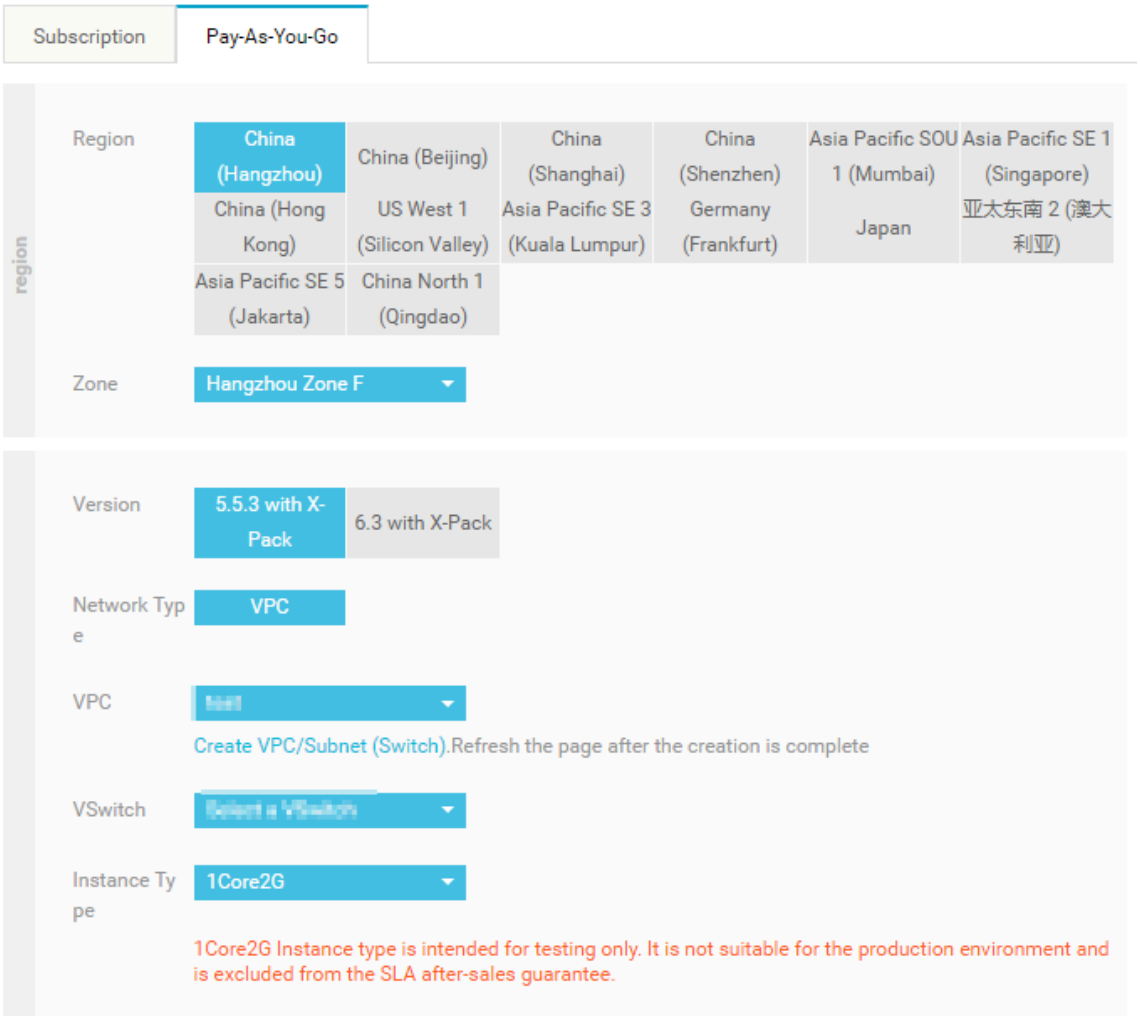
1. Elasticsearch インスタンスと ECS インスタンスの作成

   a. VPCの作成を行います。 この例では、VPC を中国 (杭州) リージョンに作成します。 インスタンス名は es_test_vpc、VSwitch 名は es_test_switch です。

   b. Elasticsearch コンソールにログインし 、Elasticsearch インスタンスを作成します。

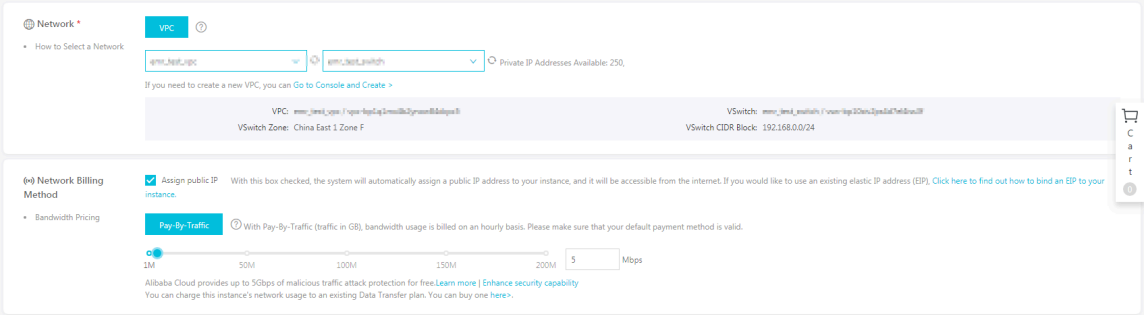
    注 :

前の手順で作成した VPC と同じリージョン、VPC、 VSwitch を選択するようにします。



c. Elasticsearch インスタンスと同じ VPC にある ECS インスタンスを購入し、パブリック IP アドレスを割り当てるか、EIP を有効化します。 コストを節約するため、要件を満たす既存の ECS インスタンスを使用することを推奨します。

この例では、杭州 (中国東部 1) ゾーン F に ECS インスタンスを作成します。 [64-bit CentOS 7.4] と [パブリック IP の割り当て] を選択して、ネットワークを設定します (次図)。



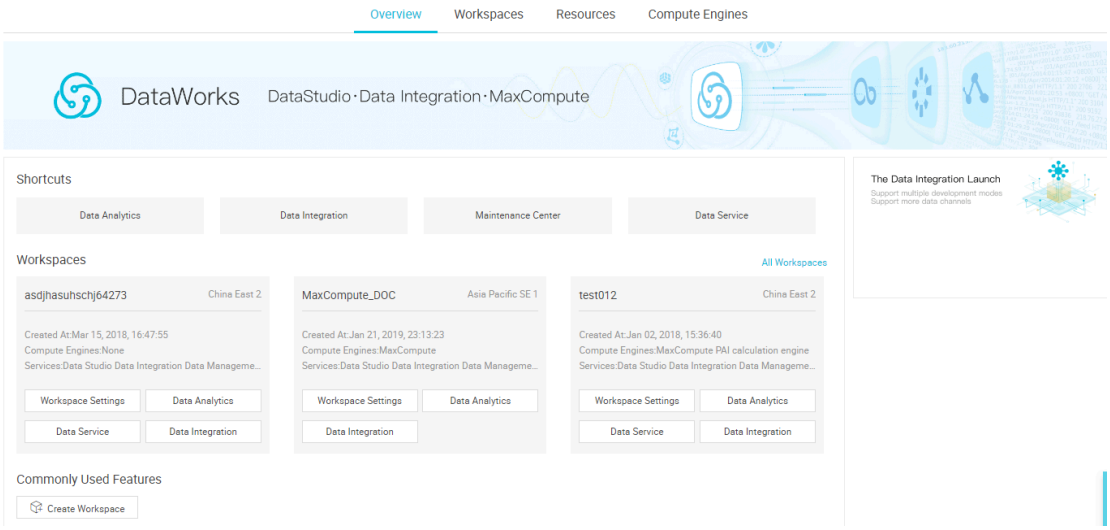
注：

- CentOS 6、CentOS 7、または Aliyun Linux を使用することを推奨します。
- ECS インスタンスで MaxCompute タスクまたはデータ同期タスクを実行する場合、ECS インスタンスの Python バージョンが 2.6 か 2.7 であることを確認する必要があります。 CentOS 5 をインストールすると、Python 2.4 もインストールされます。 他の CentOS バージョンの Python は 2.6 以降です。
- ECS インスタンスにパブリック IP アドレスが割り当てられていることを確認します。

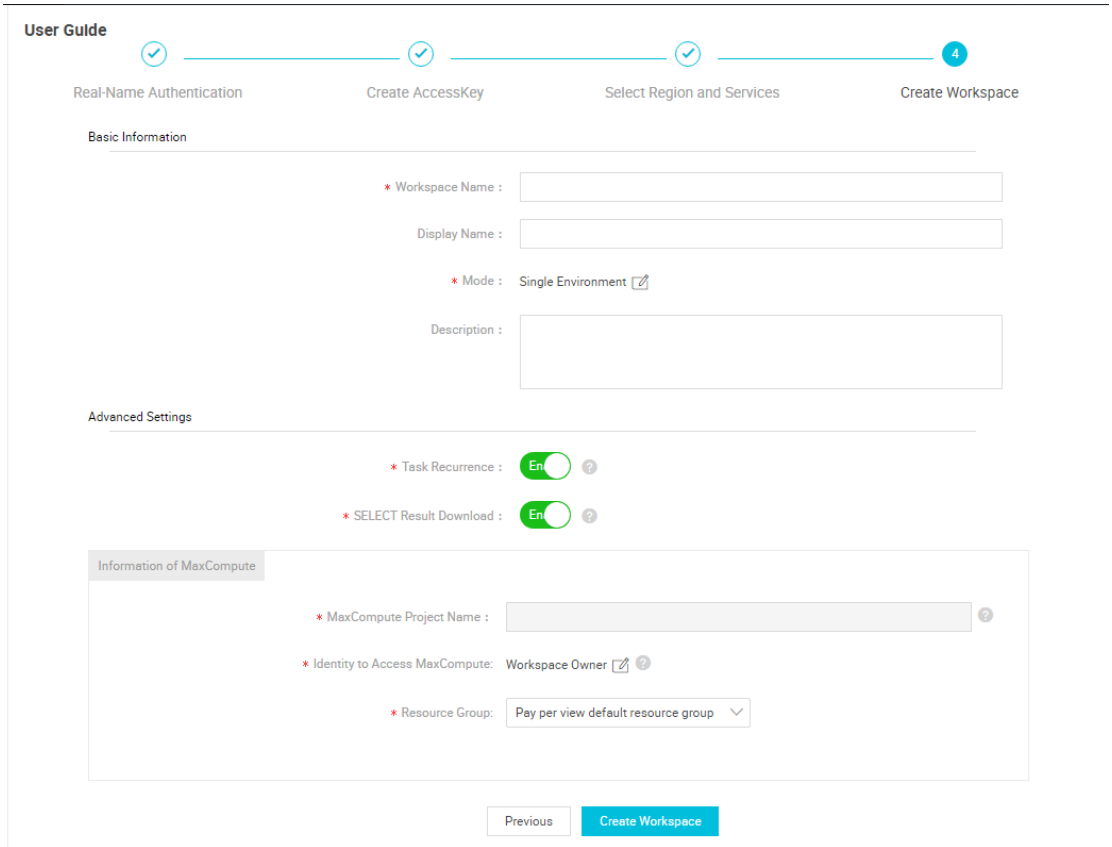
2. データ同期タスクの設定

a. DataWorks コンソールにログインしてプロジェクトを作成します。 この例では、bigdata_DOC という名前の DataWorks プロジェクトを使用します。

・ Data Integration が既に有効化されている場合、次のページが表示されます。



・ Data Integration が有効化されていない場合、次のページが表示されます。Data Integration を有効化するには、次の手順を実行します。 このサービスを有効化すると、料金が発生します。料金設定ルールに基づいて見積もることができます。



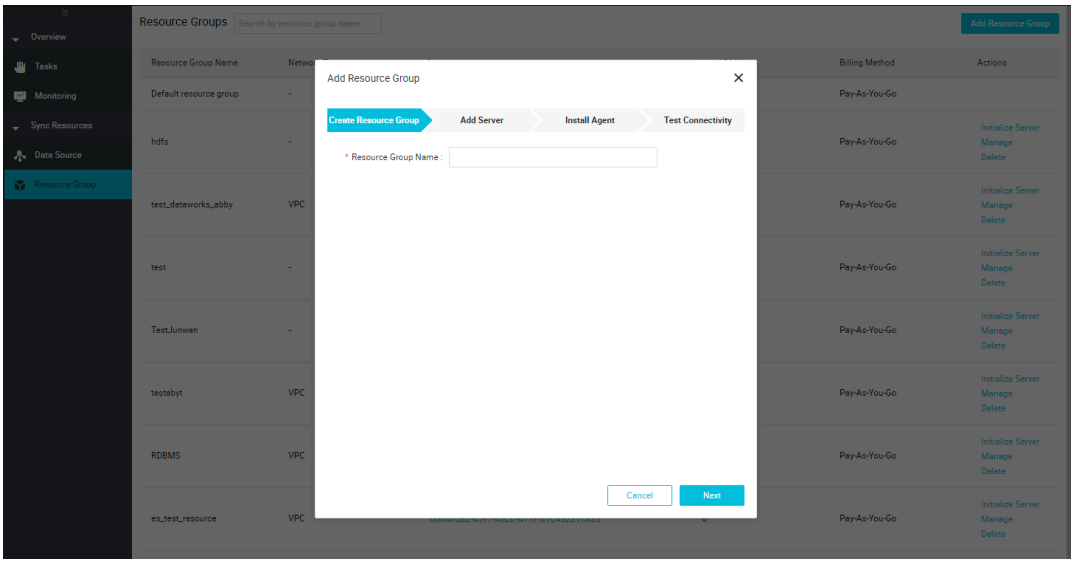
b. DataWorks プロジェクトの下にある [Data Integration] をクリックします。

c. リソースグループを作成します。

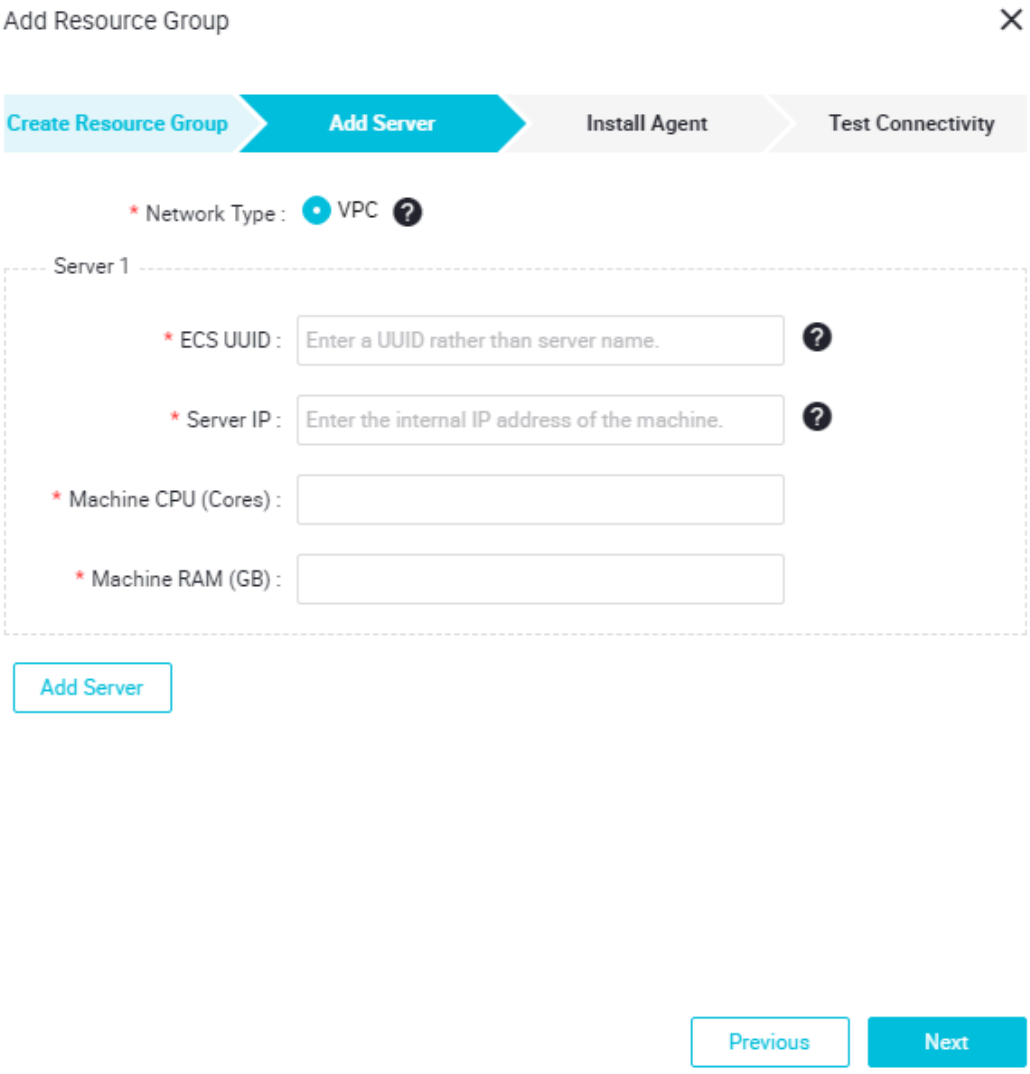
A. [Data Integration] ページの左側のナビゲーションウィンドウで、[Resource Group] を選択し、[Add Resource Group] をクリックします。

B. リソースグループを追加するには、次の手順に従います。

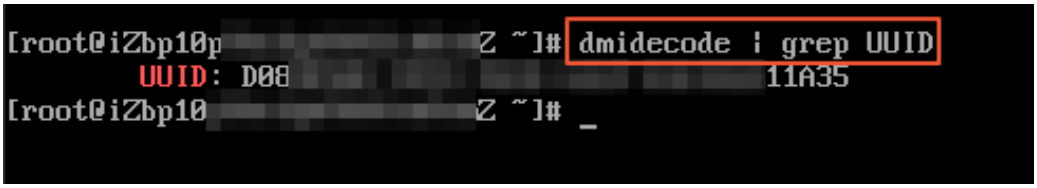
A. リソースグループを作成します。リソースグループ名を入力します。 この例では、リソースグループに es_test_resource という名前を付けます。



B. サーバーを追加します。



- ECS UUID：手順 3: インスタンスへの接続。 ECS インスタンスにログインし、`dmidecode | grep UUID` コマンドを実行して戻り値を取得します。



- Machine IP/Machine CPU (Cores)/Memory Size (GB)：ECS インスタンスのパブリック IP アドレス、CPU コア、メモリサイズを指定します。 ECS コンソー

にログインし、ECS インスタンス名をクリックすると、[設定情報] に、情報が表示されます。

C. エージェントをインストールします。次の手順に従って、エージェントのインストールを完了します。 この例では、VPC を使用します。 したがって、インスタンスのポート 8000 を開く必要はありません。

D. 接続を確認します。接続が正常に確立されると、ステータスが [Available] に変わります。 ステータスが [Unavailable] の場合、ECS インスタンスにログインし、

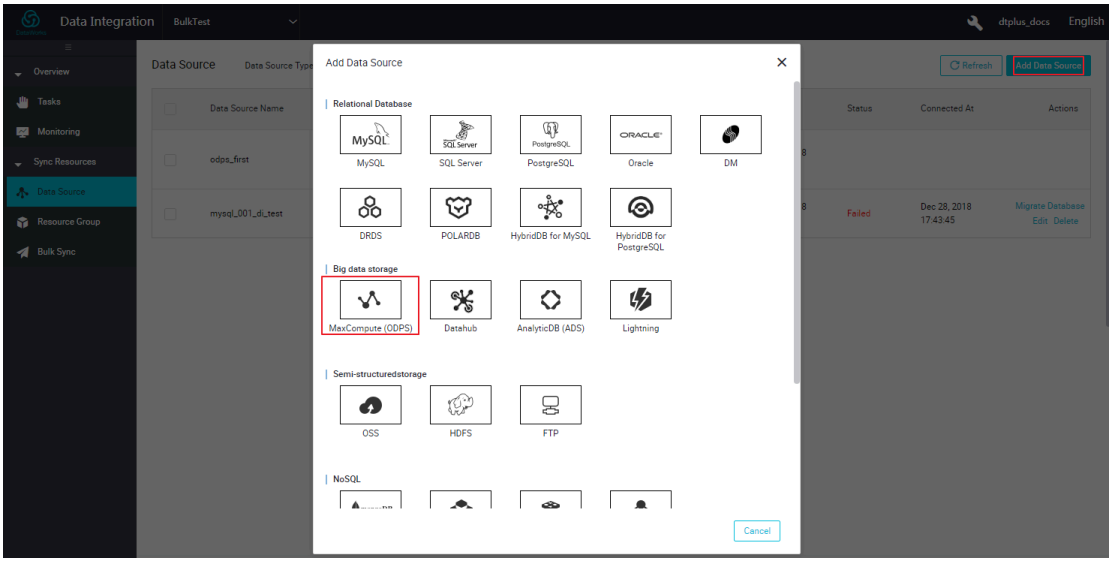
```
tail  - f  / home / admin / alisataskn  ode / logs / heartbeat
```

`. log` コマンドを実行して DataWorks と ECS インスタンス間のハートビートメッセージがタイムアウトしたかどうかを確認します。

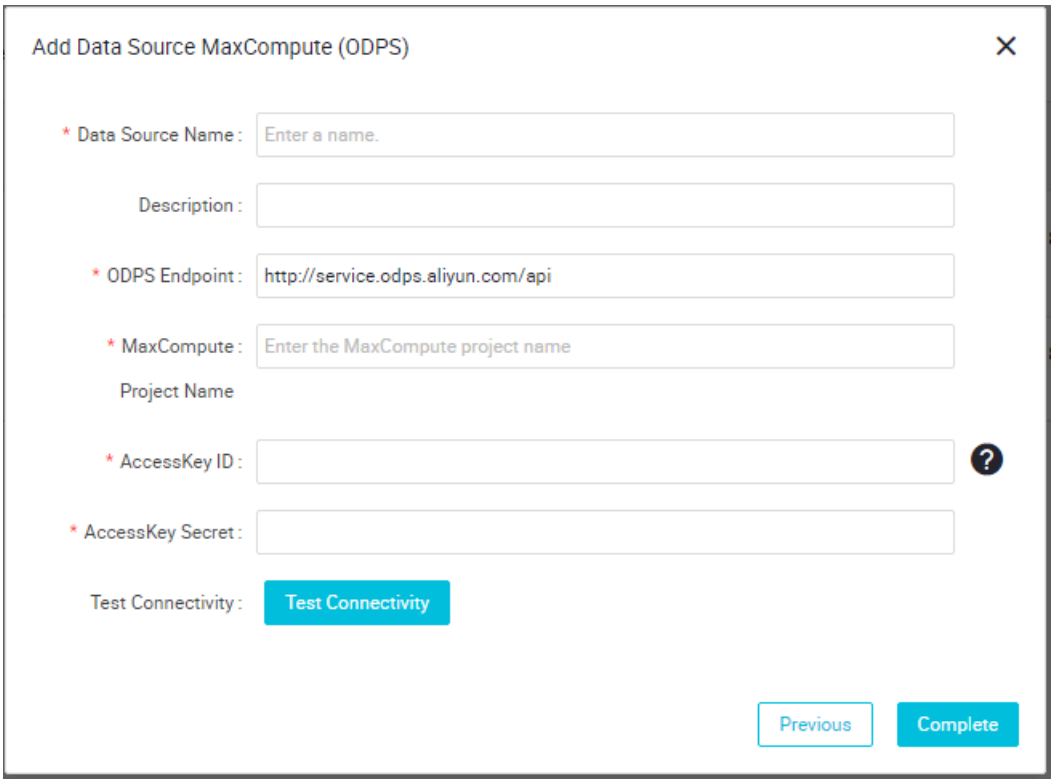
d. データソースを追加します。

A. [Data Integration] ページの左側のナビゲーションウィンドウで、[Data Source] を選択し、[Add Data Source] をクリックします。

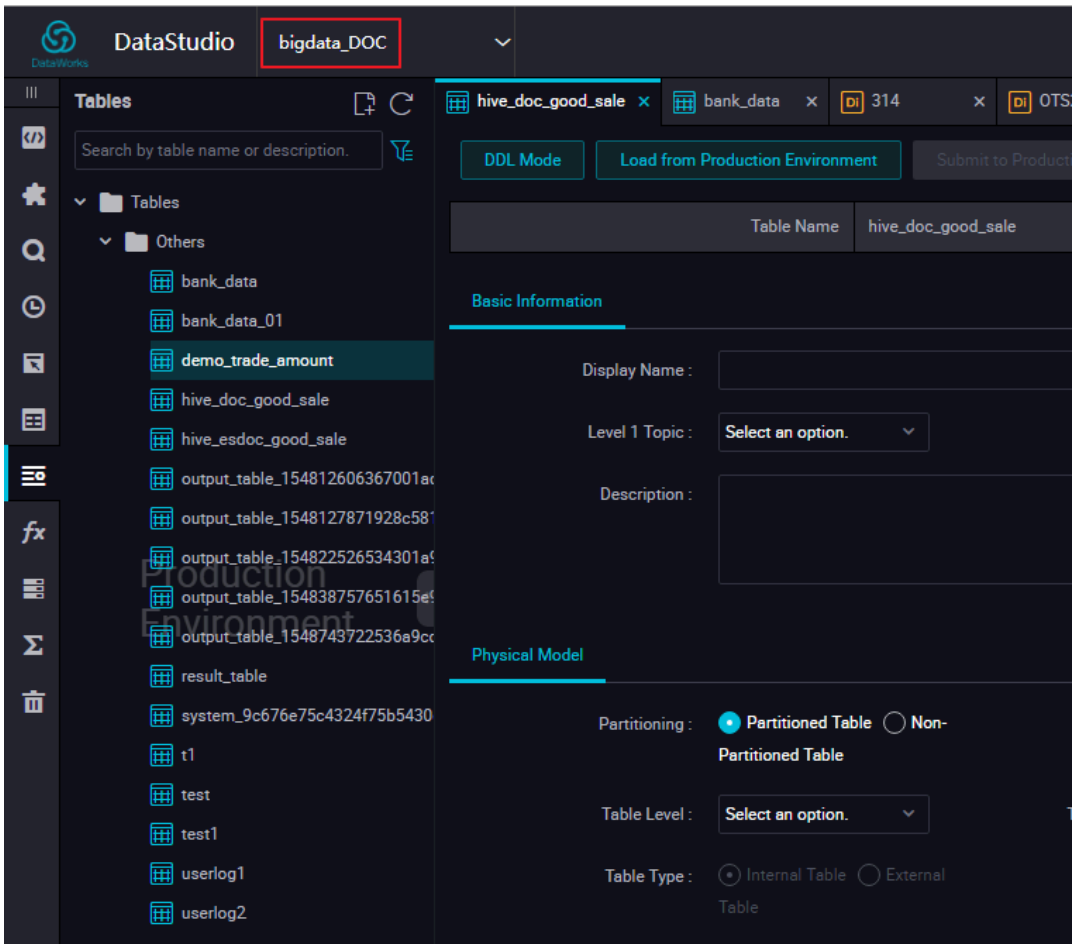
B. ソースタイプとして [MaxCompute] を選択します。



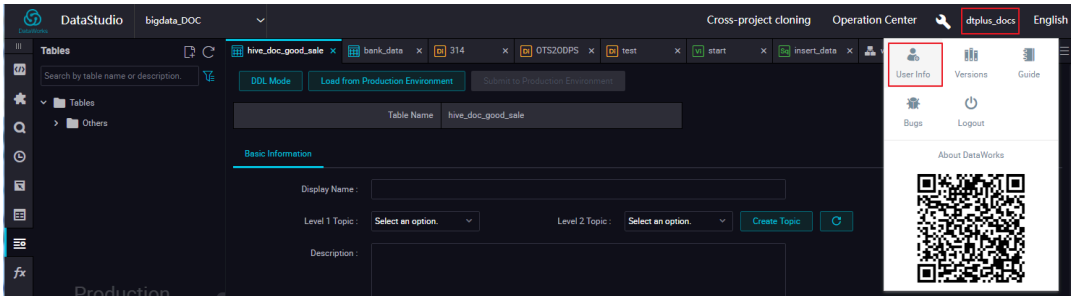
C. データソースに関する情報を入力します。 この例では、odps_es という名前のデータソースを作成します (次図)。



- ODPS workspace name：DataWorks の [Data Analytics] ページの左上隅のアイコンの右側に、テーブルのワークスペース名が表示されます (次図) 。



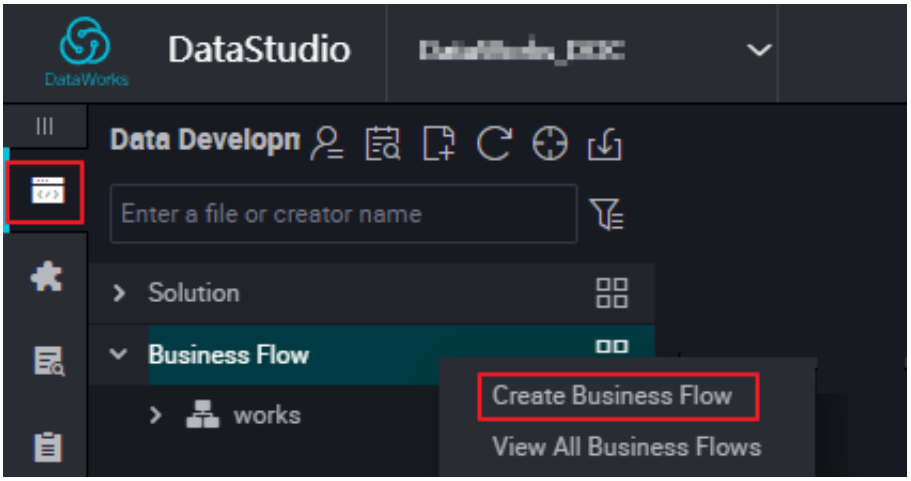
- AccessKeyId /AccessKeySecrete：ユーザー名の上にポインターを移動し、[User Info] を選択します (次図)。



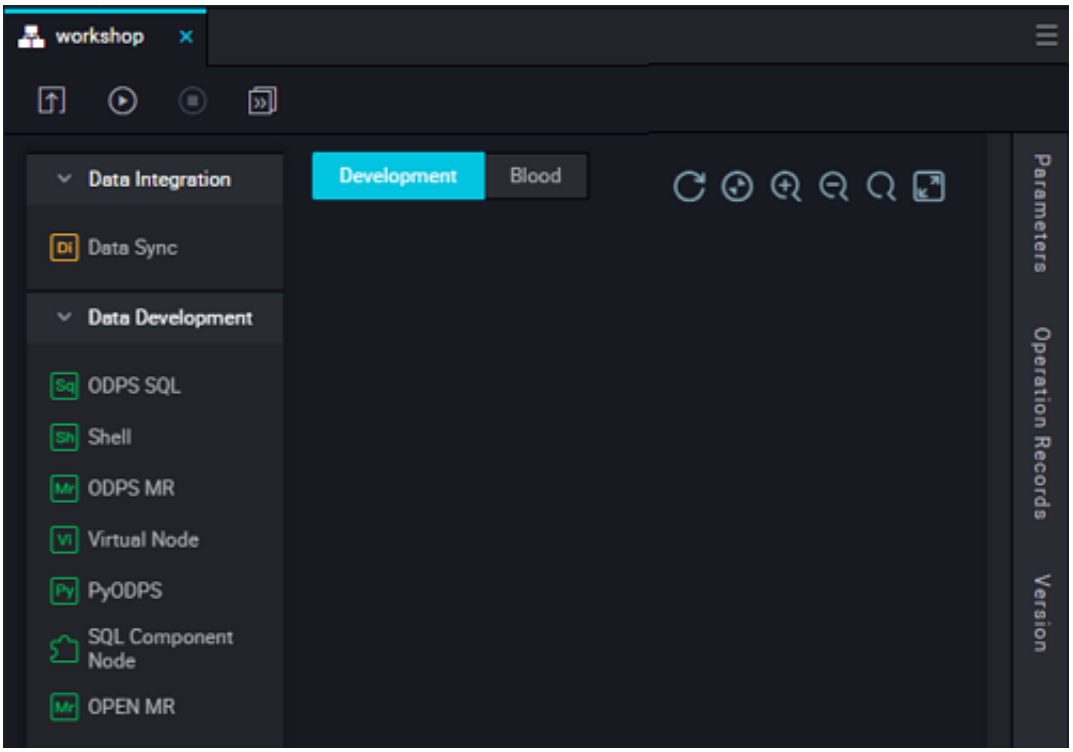
[Personal Account] ページでアバターの上にポインターを移動し、[accesskeys] をクリックします (次図)。

e. 同期タスクを設定します。

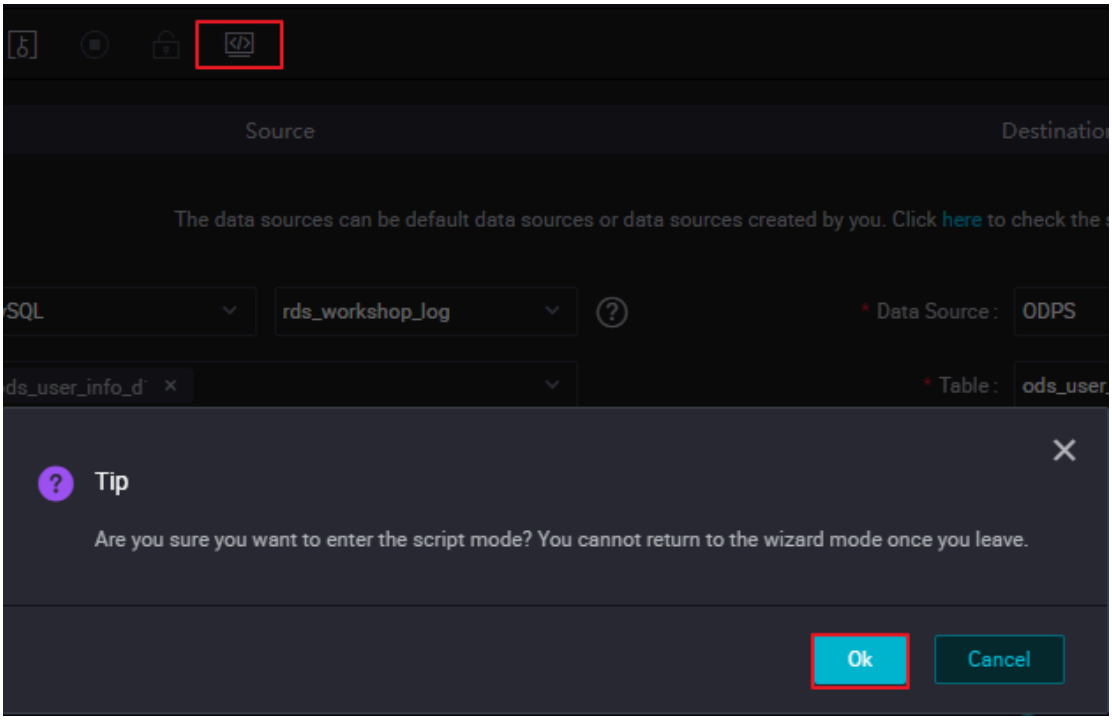
A. [Data Analytics] ページの左側のナビゲーションウィンドウで、[Data Analytics] アイコンをクリックし、[Business Flow] をクリックします。



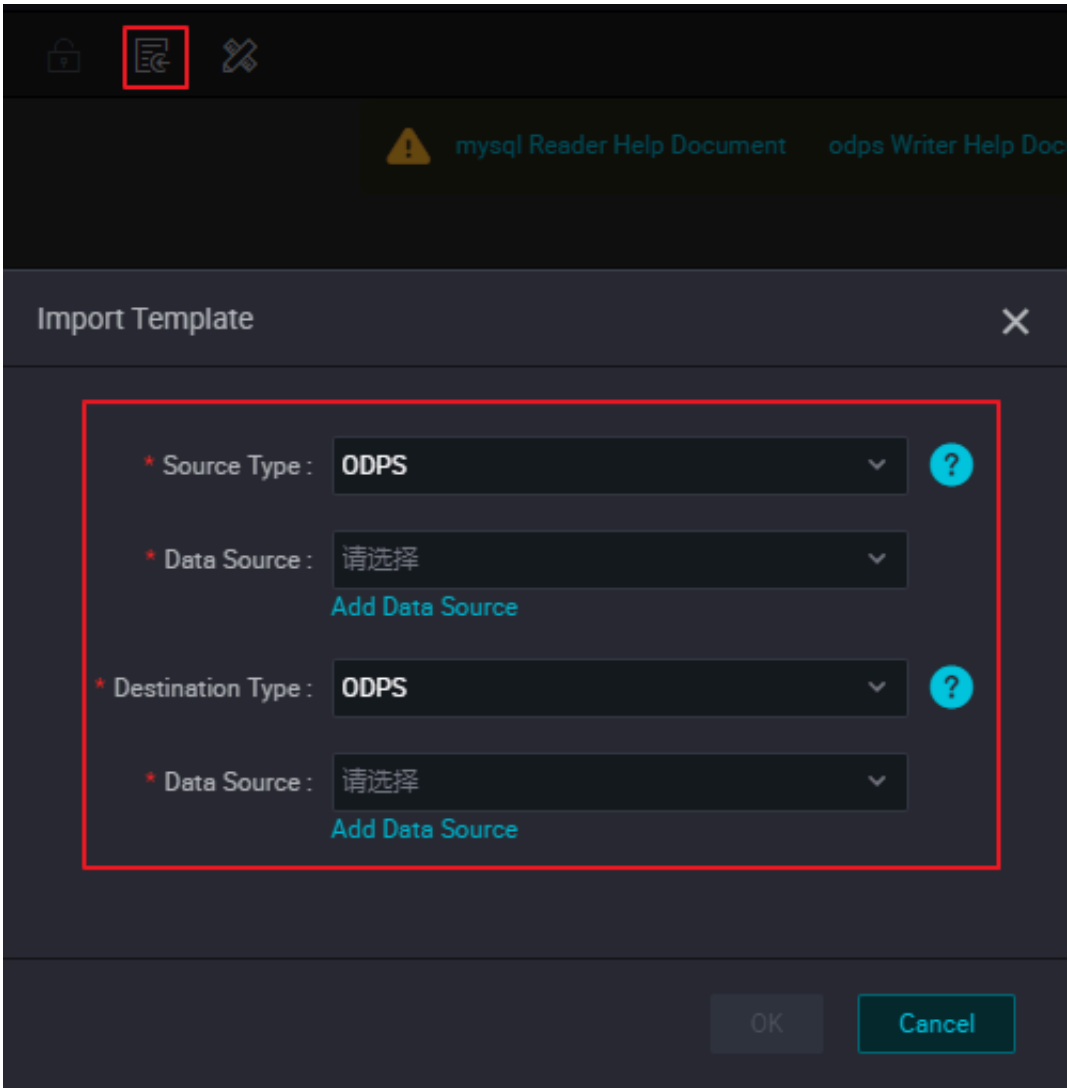
B. ターゲットビジネスフローをクリックし、[Data Integration] を選択し、[Create Data Integration Node] > [Data Sync] を選択してから、同期ノード名を入力します。



C. 同期ノードが正常に作成されたら、新しい同期ノードページの上部にある [Switch to Script Mode] アイコンをクリックし、[Confirm] を選択します。



D. [Script Mode] ページの上部にある [Apply Template] アイコンをクリックします。[Source Type]、[Data Source]、[Destination Type]、[Data Source] のオプションに情報を入力し、[OK] をクリックして初期スクリプトを生成します。



E. データ同期スクリプトの設定を行います。 Elasticsearch の設定ルールの詳細は、「writer プラグインの設定」をご参照ください。

```
"reader": {
  "plugin": "odps",
  "parameter": {
    "partition": "pt=1",
    "datasource": "odps_es",
    "column": [
      "create_time",
      "category",
      "brand",
      "buyer_id",
      "trans_num",
      "trans_amount",
      "click_cnt"
    ],
    "table": "hive_doc_good_sale"
  }
},
"writer": {
  "plugin": "elasticsearch",
  "parameter": {
    "accessId": "elastic",
    "endpoint": "http://es-cn-mp[redacted].elasticsearch.aliyuncs.com:9200",
    "indexType": "elasticsearch",
    "accessKey": "[redacted]",
    "cleanup": true,
    "discovery": false,
    "column": [
      {
        "name": "create_time",
        "type": "string"
      },
      {
        "name": "category",
        "type": "string"
      },
      {
        "name": "brand",
        "type": "string"
      },
      {
        "name": "buyer_id",
        "type": "string"
      },
      {
        "name": "trans_num",
        "type": "long"
      },
      {
        "name": "trans_amount",
        "type": "double"
      },
      {
        "name": "click_cnt",
        "type": "long"
      }
    ],
    "index": "es_index",
    "batchSize": 1000,
```

注：

- **同期スクリプトの設定には、Reader、Writer、Setting の 3 つの部分があります。 `Reader` では、データを同期するソースクラウドサービスを設定します。 `Write` では、Elasticsearch の設定ファイルを設定します。 `Setting` では、パケットロスと最大同時タスクを設定します。**

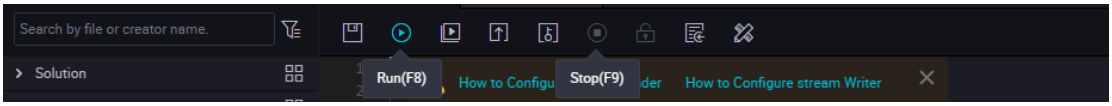
- `Endpoint` には、Elasticsearch インスタンスのプライベートまたはパブリック IP アドレスを指定します。 この例では、プライベート IP アドレスを使用します。 したがって、ホワイトリストは不要です。 外部 IP アドレスを使用する場合、Elasticsearch の [ネットワークとスナップショット] ページで、Elasticsearch へのアクセスが許可されているパブリック IP アドレスを含むホワイトリストを設定する必要があります。 ホワイトリストには、DataWorks サーバーの IP アドレスと、使用するリソースグループを指定する必要があります。
- Elasticsearch インスタンスへのログインに使用されるユーザー名とパスワードを Elasticsearch Writer の `accessId` と `accesskey` に設定する必要があります。
- Elasticsearch インスタンスのインデックス名を `index` に入力します。 Elasticsearch インスタンスのデータにアクセスするには、このインデックス名を使用する必要があります。 この例では、 `es_index` という名前の インデックスを使用します。
- MaxCompute テーブルがパーティションテーブルの場合、 `partition` フィールドにパーティション情報を設定する必要があります。 この例のパーティション情報は、pt=1 です。

サンプル設定コード：

```
{
" configurat  ion ": {
" reader ": {
" plugin ": " odps ",
" parameter ": {
  " partition ": " pt = 1 ",
  " datasource ": " odps_es ",
  " column ": [
    " create_tim  e ",
    " category ",
    " brand ",
    " buyer_id ",
    " trans_num ",
    " trans_amou  nt ",
    " click_cnt "
  ],
  " table ": " hive_doc_g  ood_sale "
}
},
" writer ": {
" plugin ": " elasticsea  rch ",
" parameter ": {
  " accessId ": " elastic ",
  " endpoint ": " http :// es - cn - mpXXXXXXX . elasticsea
 rch . aliyuncs . com : 9200 ",
  " indexType ": " elasticsea  rch ",
  " accessKey ": " XXXXXX ",
  " cleanup ":  true ,
  " discovery ":  false ,
```

```
  " column ": [
    {
      " name ": " create_tim  e ",
      " type ": " string "
    },
    {
      " name ": " category ",
      " type ": " string "
    },
    {
      " name ": " brand ",
      " type ": " string "
    },
    {
      " name ": " buyer_id ",
      " type ": " string "
    },
    {
      " name ": " trans_num ",
      " type ": " long "
    },
    {
      " name ": " trans_amou  nt ",
      " type ": " double "
    },
    {
      " name ": " click_cnt ",
      " type ": " long "
    }
  ],
  " index ": " es_index ",
  " batchSize ":  1000 ,
  " splitter ": ",",
}
},
" setting ": {
" errorLimit ": {
  " record ": " 0 "
},
" speed ": {
  " throttle ":  false ,
  " concurrent ":  1 ,
  " mbps ": " 1 ",
  " dmu ":  1
}
}
},
" Type ": " job  ",
" version ": " 1 . 0 "
}
```

F. スクリプトが同期されたら、[Run] をクリックして、ODPS データを Elasticsearch と同期させます。



3. 結果の検証

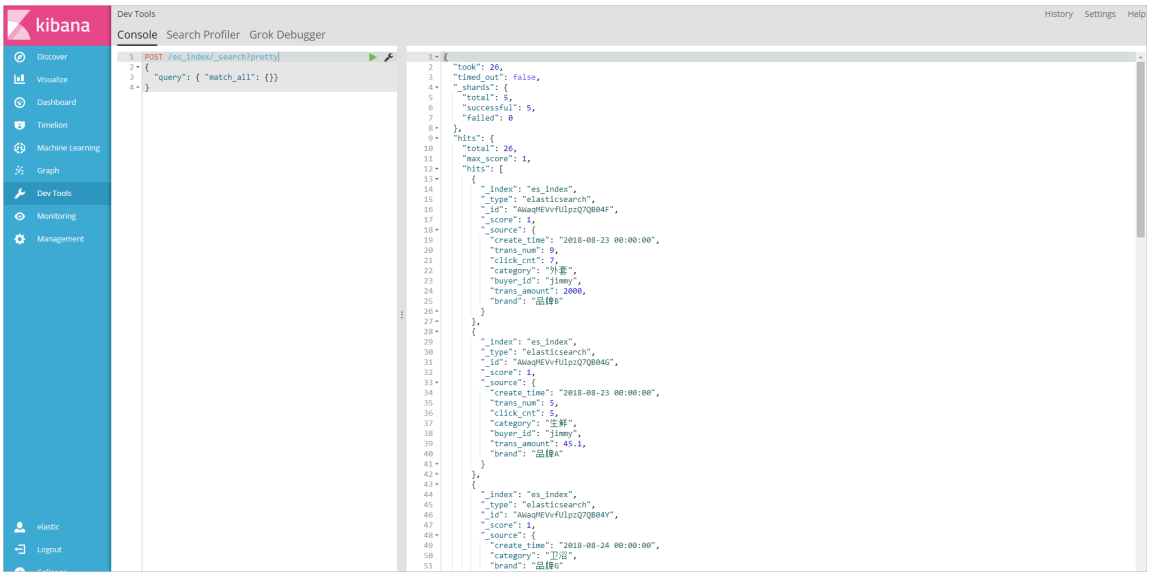
a. Elasticsearch コンソールにログインし、右上隅の Kibana コンソールをクリックし、[Dev Tools] を選択します。

b. 次のコマンドを実行して、データが Elasticsearch に正常にレプリケートされたことを確認します。

```
 POST  / es_index / _search ?  pretty
{
" query ": { " match_all ": {}}
}
```

`es_index` は、データ同期中の `index` フィールドの値です。

データが正常に同期されると、次のページが表示されます。



c. 次のコマンドを実行して、 `trans_num` フィールドを基準にドキュメントをソートします。

```
 POST  / es_index / _search ?  pretty
{
" query ": { " match_all ": {} },
" sort ": { " trans_num ": { " order ": " desc " } }
}
```

d. 次のコマンドを実行して、ドキュメント内の `category` フィールドと `brand` フィールドを検索します。

```
 POST  / es_index / _search ?  pretty
{
" query ": { " match_all ": {} },
" _source ": [" category ", " brand "]
```

```
}
```

e. 次のコマンドを実行して、 category が fresh のドキュメントをクエリします。

```
 POST  / es_index / _search ?  pretty
{
" query ": { " match ": {" category ":" fresh "} }
}
```

詳細は、「Elasticsearch アクセスのテスト」と『Elastic help center』をご参照ください。

よくある質問

Elasticsearch インスタンスに接続するときにエラーが発生します

1. 同期スクリプトを実行する前に、前の手順で作成したリソースグループが、右側の [configuration tasks resources group] メニューで選択されているかどうかを確認してください。
   - リソースグループが選択されている場合、次の手順に進みます。
   - リソースグループが選択されていない場合、右側の [configuration tasks resources group] メニューで、作成したリソースグループを選択し、[Run] をクリックします。
2. endpoint、accessId、accesskey など、同期スクリプトの設定が正しいかどうかを確認してください。 endpoint には、Elasticsearch インスタンスのプライベートまたはパブリック IP アドレスを指定します。 パブリック IP アドレスを使用する場合、ホワイトリストを設定してください。 accessId には、Elasticsearch インスタンスへのアクセスに使用されるユーザー名を指定します。デフォルトは elastic です。 accesskey には、Elasticsearch インスタンスへのアクセスに使用されるパスワードを指定します。

3.6 ES-Hadoop と Elasticsearch 間のデータ相互接続

Elasticsearch と E-MapReduce をベースとする ES-Hadoop を介して、Elasticsearch に直接データを書き込むことができます。

バージョン

Elasticsearch 5.5.3 with X-Pack がサポートされています。

注 :

Elasticsearch 6.3.2 with X-Pack がサポートされていません。

Elasticsearch の有効化

この例では、次の Alibaba Cloudサービスを使用します。

- VPC：パブリックネットワークでのデータ送信は、安全ではありません。 Elasticsearch インスタンスへの安全な接続を確保するため、特定のリージョンに VPC と VSwitch をデプロイする必要があります。 そのため、VPC を有効化する必要があります。
- OSS：この例では、OSS を使用して E-MapReduce ログを保存します。 E-MapReduce を有効化する前に、OSS を有効化し、バケットを作成する必要があります。
- Elasticsearch
- E-MapReduce

上記の Alibaba Cloud サービスを有効化するには、次の手順に従います。

1. VPC の有効化
   a. Alibaba Cloud Web サイトで、[プロダクト] > [ネットワークサービス] > [Virtual Private Cloud] を選択し、[無料で始める] をクリックします。
   b. VPC コンソールにログインし、[VPC の作成] をクリックして VPC を作成します。
   c. 作成した VPC をコンソールで管理できます。

   > 注：
   > VPC の詳細は、「」「Virtual Private Cloud (VPC)」「 」をご参照ください。

2. Object Storage Service の有効化
   a. コンソールにログインし、[プロダクト] > [ストレージと CDN] > [Object Storage Service] を選択し、[今すぐ購入] をクリックします。
   b. OSS コンソールにログインし、[バケットを作成] をクリックしてバケットを作成します。

   > 注：
   > E-MapReduce クラスターと同じリージョンにバケットを作成する必要があります。 この例では、[中国 (杭州)] リージョンを選択します。

   c. ページに表示される指示に従ってバケットを作成します。
3. Elasticsearch の有効化
   a. Alibaba Cloud Web サイトで、[プロダクト] > [分析とビッグデータ] > [Elasticsearch] を選択し、プロダクトページを表示します。

   > 注：
   > 30 日間の無料トライアルを入手できます。

   b. Elasticsearch を正常に有効化した後、新しく作成した Elasticsearch インスタンスは Elasticsearch コンソールに表示されます。

4. E-MapReduce の有効化

   a. Alibaba Cloud Web サイトで、[プロダクト] > [分析とビッグデータ] > [E-MapReduce] を選択し、プロダクトページを表示します。

   b. [使用を開始する] をクリックし、設定を完了します。

   c. 作成した E-MapReduce クラスターは、クラスターリストに表示されます。作成状況を確認するには、次の操作を行います。

      ・ パブリック IP アドレスを介して、リモートからクラスターにログインできます。

      ```
      ssh   root @ your   public   IP   address
      ```

      ・ バックグラウンドプロセスを表示するには、jps コマンドを実行します。

      ```
      [ root @ emr - header - 1  ~]#  jps
       16640   Bootstrap
       17988   RunJar
       19140   HistorySer  ver
       18981   WebAppProx  yServer
       14023   Jps
       15949   gateway . jar
       16621   ZeppelinSe  rver
       1133   EmrAgent
       15119   RunJar
       17519   ResourceMa  nager
       1871   Applicatio  n
       19316   JobHistory  Server
       1077   WatchDog
       17237   SecondaryN  ameNode
       16502   NameNode
       16988   ApacheDsTa  nukiWrappe  r
       18429   Applicatio  nHistorySe  rver
      ```

E-MapReduce から Elasticsearch にデータを書き込む MR ジョブの作成

Maven を使用してプロジェクトを管理することを推奨します。 Maven を使用するには、次の手順に従います。

1. Maven をインストールします。

   コンピューターに Maven がインストールされていることを確認します。

2. エンジニアリングフレームワークを生成します。

   プロジェクトのルートディレクトリで、次のコマンドを実行します。

```
mvn   archetype : generate  - DgroupId = com . aliyun . emrtoes
 - DartifactI  d = emrtoes  - Darchetype  ArtifactId = maven -
archetype - quickstart  - Dinteracti  veMode = false
```

   空のサンプルプロジェクト emrtoes (指定した artifactId と同じ) が自動的に生成されます。プロジェクトには、 *pom . xml* ファイルとアプリケーションクラスが含まれています。クラスパッケージのパスは、指定した groupId と同じです。

3. Hadoop と ES-Hadoop の依存関係を追加します。

   このプロジェクトを任意の IED で開始し、 *pom . xml* ファイルを編集します。次の内容を依存関係に追加します。

```
< dependency >
      < groupId > org . apache . hadoop </ groupId >
      < artifactId > hadoop - mapreduce - client - common </
artifactId >
      < version > 2 . 7 . 3 </ version >
  </ dependency >
  < dependency >
      < groupId > org . apache . hadoop </ groupId >
      < artifactId > hadoop - common </ artifactId >
      < version > 2 . 0 . 3 </ version >
  </ dependency >
   < dependency >
       < groupId > org . elasticsea  rch </ groupId >
       < artifactId > elasticsea  rch - hadoop - mr </ artifactId >
       < version > 2 . 5 . 0 </ version >
   </ dependency >
```

4. パッケージングプラグインを追加します。

   サードパーティのデータベースを使用しているため、このデータベースを JAR パッケージにパッケージ化する必要があります。次の maven-assembly-plugin を *pom . xml* ファイルに追加します。

```
< plugins >
   < plugin >
     < artifactId > maven - assembly - plugin </ artifactId >
     < configurat  ion >
       < archive >
         < manifest >
           < mainClass > com . aliyun . emrtoes . EmrToES </
mainClass >
         </ manifest >
       </ archive >
       < descriptor  Refs >
         < descriptor  Ref > jar - with - dependenci  es </
descriptor  Ref >
       </ descriptor  Refs >
     </ configurat  ion >
     < executions >
```

```
      < execution >
        < id > make - assembly </ id >
        < phase > package </ phase >
        < goals >
          < goal > single </ goal >
        </ goals >
      </ execution >
    </ executions >
  </ plugin >
  < plugin >
    < groupId > org . apache . maven . plugins </ groupId >
    < artifactId > maven - shade - plugin </ artifactId >
    < version > 2 . 1 . 0 </ version >
    < executions >
      < execution >
        < phase > package </ phase >
        < goals >
          < goal > shade </ goal >
        </ goals >
        < configurat  ion >
          < transforme  rs >
            < transforme  r   implementa  tion =" org . apache
. maven . plugins . shade . resource . ApacheLice  nseResourc
eTransform  er ">
            </ transforme  r >
          </ transforme  rs >
        </ configurat  ion >
      </ execution >
    </ executions >
  </ plugin >
</ plugins >
```

5. コードを記述します。

   アプリケーションクラスと並行する新しいクラス EmrToES.java を com.aliyun.emrtoes パッケージに追加します。 次の内容を追加します。

```
package   com . aliyun . emrtoes ;
 import   org . apache . hadoop . conf . Configurat  ion ;
 import   org . apache . hadoop . fs . Path ;
 import   org . apache . hadoop . io . NullWritab  le ;
 import   org . apache . hadoop . io . Text ;
 import   org . apache . hadoop . mapreduce . Job ;
 import   org . apache . hadoop . mapreduce . Mapper ;
 import   org . apache . hadoop . mapreduce . lib . input .
FileInputF  ormat ;
 import   org . apache . hadoop . mapreduce . lib . input .
TextInputF  ormat ;
 import   org . apache . hadoop . util . GenericOpt  ionsParser ;
 import   org . elasticsea  rch . hadoop . mr . EsOutputFo  rmat ;
 import   java . io . IOExceptio  n ;
 public   class   EmrToES  {
     public   static   class   MyMapper   extends   Mapper < Object
,  Text ,  NullWritab  le ,  Text > {
         private   Text   line  =  new   Text ();
        @ Override
         protected   void   map ( Object   key ,  Text   value ,
Context   context )
                 throws   IOExceptio  n ,  Interrupte  dException
{
             if  ( value . getLength () >  0 ) {
                 line . set ( value );
```

```
                    context . write ( NullWritab  le . get (),  line
);
                }
            }
        }
    public   static   void   main ( String []  args )  throws
  IOExceptio  n ,  ClassNotFo  undExcepti  on ,  Interrupte
dException  {
        Configurat  ion   conf  =  new   Configurat  ion ();
        String []  otherArgs  =  new   GenericOpt  ionsParser (
conf ,  args ). getRemaini  ngArgs ();
        // Alibaba   Cloud   Elasticsea  rch   X - PACK   username
  and   password
        conf . set (" es . net . http . auth . user ", " X - PACK
  username ");
        conf . set (" es . net . http . auth . pass ", " X - PACK
  password ");
        conf . setBoolean (" mapred . map . tasks . speculativ  e
. execution ",  false );
        conf . setBoolean (" mapred . reduce . tasks . speculativ
e . execution ",  false );
        conf . set (" es . nodes ", " The   private   address   of
  your   Elasticsea  rch   instance ");
        conf . set (" es . port ", " 9200 ");
        conf . set (" es . nodes . wan . only ", " true ");
        conf . set (" es . resource ", " blog / yunqi ");
        conf . set (" es . mapping . id ", " id ");
        conf . set (" es . input . json ", " yes ");
        Job   job  =  Job . getInstanc  e ( conf , " EmrToES ");
        job . setJarByCl  ass ( EmrToES . class );
        job . setMapperC  lass ( EsMapper . class );
        job . setInputFo  rmatClass ( TextInputF  ormat . class );
        job . setOutputF  ormatClass ( EsOutputFo  rmat . class );
        job . setMapOutp  utKeyClass ( NullWritab  le . class );
        job . setMapOutp  utValueCla  ss ( Text . class );
        FileInputF  ormat . setInputPa  ths ( job ,  new   Path (
otherArgs [ 0 ]));
        System . exit ( job . waitForCom  pletion ( true ) ?  0
:  1 );
    }
```

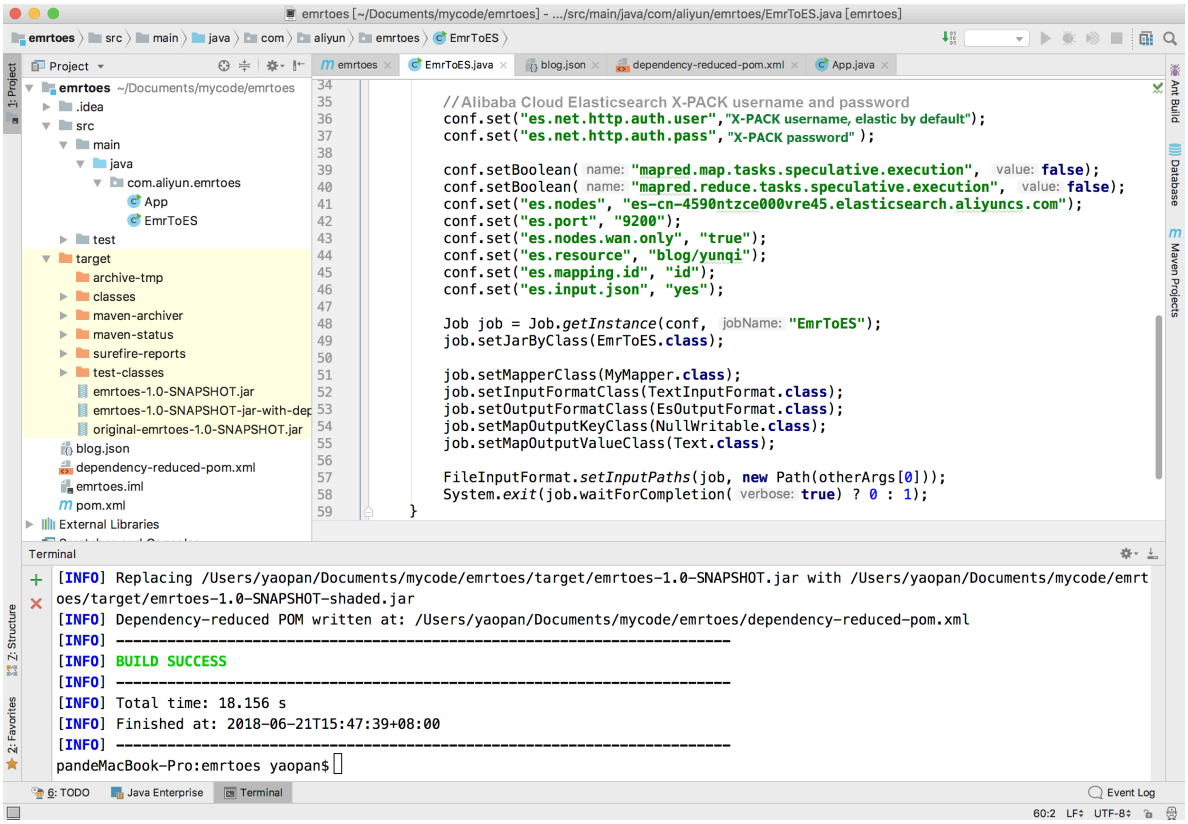
```
}
```

6. コンパイルしてパッケージ化します。

   プロジェクトディレクトリで次のコマンドを実行します。

```
mvn clean package
```

   コマンドを実行した後、プロジェクトのターゲットディレクトリで、ジョブの JAR パッケージ emrtoes-1.0-SNAPSHOT-jar-with-dependencies.jar を確認できます。



E-MapReduce でのジョブの実行

1. データのテスト

   a. 次のデータを blog.json に書き込みます。

```
{" id ":" 1 "," title ":" git introducti on "," posttime ":"
 2016 - 06 - 11 "," content ":" The main difference between
   svn and git ..."}
{" id ":" 2 "," title ":" Introducti on and simple use of
   Java Generics "," posttime ":" 2016 - 06 - 12 "," content ":"
 Basic operations : CRUD ..."}
{" id ":" 3 "," title ":" Basic operations of SQL ","
 posttime ":" 2016 - 06 - 13 "," content ":" The main
 difference between svn and git ..."}
{" id ":" 4 "," title ":" Basic Hibernate framework ","
 posttime ":" 2016 - 06 - 14 "," content ":" Basic Hibernate
 framework ..."}
```

```
{" id ":" 5 "," title ":" Basics   of   Shell "," posttime ":"
 2016 - 06 - 15 "," content ":" What   is   Shell ?..."}
```

b. 次の scp リモートコピーコマンドを実行して、ファイルを EMR クラスターにアップロードします。

```
scp   blog . json   root @ your   EIP :/ root
```

c. blog.json を HDFS にアップロードします。

```
hadoop   fs  - mkdir  / work
```

```
hadoop   fs  - put   blog . json  / work
```

2. JAR パッケージのアップロード

   Maven プロジェクトのターゲットディレクトリに保存されている JAR パッケージを EMR クラスターにアップロードします。

```
scp   target / emrtoes - 1 . 0 - SNAPSHOT - jar - with -
dependenci  es . jar   root @ YourIP :/ root
```

3. MR ジョブの実行

   以下のコマンドを実行します。

```
hadoop   jar   emrtoes - 1 . 0 - SNAPSHOT - jar - with -
dependenci  es . jar  / work / blog . json
```

   ジョブが正常に実行されると、次のメッセージがコンソールに表示されます。

```
1. root@emr-header-1:~ (ssh)
[root@emr-header-1 ~]# hadoop jar emrtoes-1.0-SNAPSHOT-jar-with-dependencies.jar /work/blog.json
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/apps/ecm/service/hadoop/2.7.2-1.2.11/package/hadoop-2.7.2-1.2.11/share/had
oop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/apps/ecm/service/tez/0.8.4/package/tez-0.8.4/lib/slf4j-log4j12-1.7.10.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/06/21 15:53:18 INFO impl.TimelineClientImpl: Timeline service address: http://emr-header-1.cluster-67561:8188/
ws/v1/timeline/
18/06/21 15:53:18 INFO client.RMProxy: Connecting to ResourceManager at emr-header-1.cluster-67561/192.168.0.103:
8032
18/06/21 15:53:19 INFO input.FileInputFormat: Total input paths to process : 1
18/06/21 15:53:19 INFO mapreduce.JobSubmitter: number of splits:1
18/06/21 15:53:19 INFO Configuration.deprecation: mapred.reduce.tasks.speculative.execution is deprecated. Instea
d, use mapreduce.reduce.speculative
18/06/21 15:53:19 INFO Configuration.deprecation: mapred.map.tasks.speculative.execution is deprecated. Instead,
use mapreduce.map.speculative
18/06/21 15:53:19 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1529566866753_0001
18/06/21 15:53:19 INFO impl.YarnClientImpl: Submitted application application_1529566866753_0001
18/06/21 15:53:20 INFO mapreduce.Job: The url to track the job: http://emr-header-1.cluster-67561:20888/proxy/app
lication_1529566866753_0001/
18/06/21 15:53:20 INFO mapreduce.Job: Running job: job_1529566866753_0001
18/06/21 15:53:28 INFO mapreduce.Job: Job job_1529566866753_0001 running in uber mode : false
18/06/21 15:53:28 INFO mapreduce.Job:  map 0% reduce 0%
18/06/21 15:53:34 INFO mapreduce.Job:  map 100% reduce 0%
18/06/21 15:53:40 INFO mapreduce.Job:  map 100% reduce 14%
18/06/21 15:53:41 INFO mapreduce.Job:  map 100% reduce 57%
18/06/21 15:53:42 INFO mapreduce.Job:  map 100% reduce 71%
18/06/21 15:53:43 INFO mapreduce.Job:  map 100% reduce 86%
18/06/21 15:53:44 INFO mapreduce.Job:  map 100% reduce 100%
18/06/21 15:53:44 INFO mapreduce.Job: Job job_1529566866753_0001 completed successfully
18/06/21 15:53:44 INFO mapreduce.Job: Counters: 66
        File System Counters
                FILE: Number of bytes read=412
                FILE: Number of bytes written=1024771
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=635
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=2
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
```

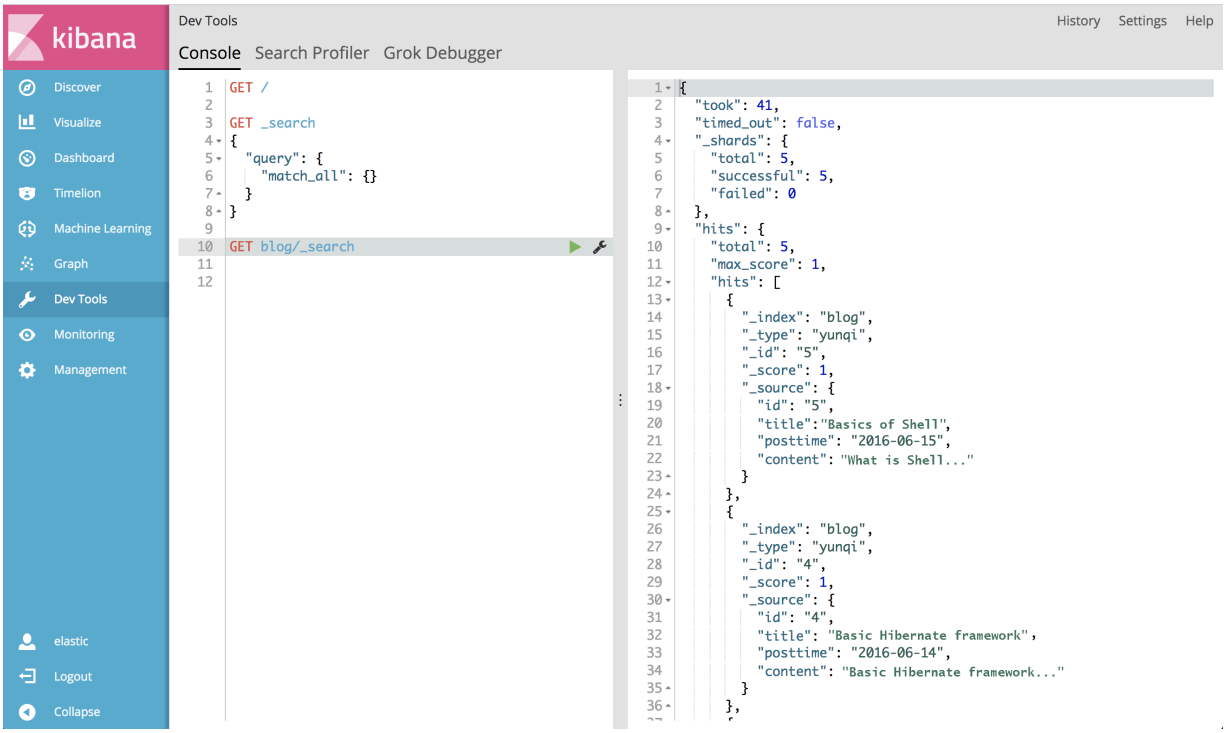
結果の確認

次のコマンドを実行して、データが Elasticsearch に正常に書き込まれたことを確認します。

```
curl -u elastic -XGET es-cn-v0h0jdp990001rta9.elasticsearch.aliyuncs.com:9200/blog/_search?pretty
```


```
[root@emr-header-1 ~]# curl -u elastic -XGET es-cn-4590nukw4000xuig3.elasticsearch.aliyuncs.com:9200/blog
/_search?pretty
Enter host password for user 'elastic':
{
  "took" : 17,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 5,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "blog",
        "_type" : "yunqi",
        "_id" : "5",
        "_score" : 1.0,
        "_source" : {
          "id" : "5",
          "title" : "Basics of Shell",
          "posttime" : "2016-06-15",
          "content" : "What is Shell..."
        }
      },
      {
        "_index" : "blog",
        "_type" : "yunqi",
        "_id" : "4",
        "_score" : 1.0,
        "_source" : {
          "id" : "4",
          "title" : "Basic Hibernate framework",
          "posttime" : "2016-06-14",
          "content" : "Basic Hibernate framework..."
        }
      },
```

Kibana で結果を確認することもできます。



API 分析

Map プロセス中、データは行単位で読み書きされます。 入力キーの型は object です。 入力値の型はテキストです。 出力キーの型は NullWritable です。これは、長さゼロにシリアライズする特殊な Writable 型です。 ストリームとの間でバイトの読み書きは行われません。 プレースホルダーとして使用されます。

たとえば、MapReduce では、キーや値を使用する必要がない場合、NullWritable として宣言できます。 この例では、出力キーを NullWritable に設定します。 出力値が BytesWritable に設定されている場合、JSON 文字列をシリアライズします。

データの書き込みのみが行われるため、Reduce プロセスは不要です。

パラメーターの説明

- conf.set(“es.net.http.auth.user”, “X-PACK username”)

  X-PACK のユーザー名を指定します。
- conf.set(“es.net.http.auth.pass”, “X-PACK password”)

  X-PACK のパスワードを指定します。
- conf.setBoolean(“mapred.map.tasks.speculative.execution”, false)

  リデューサーの投機的実行を無効にします。
- Conf.setBoolean(“mapred.reduce.tasks.speculative.exe cution ", false)

  マッパーの投機的実行を無効にします。

- conf.set(“es.nodes”,“The internal network address of your Elasticsearch”)

  Elasticsearch インスタンスにログインするための IP アドレスとポートを指定します。
- conf.set(“es.resource”,“blog/yunqi”)

  Elasticsearch インスタンスに書き込まれるデータのインデックス付けに使用されるインデックス名とタイプを指定します。
- conf.set(“es.mapping.id”,“id”)

  ドキュメント ID を指定します。"id" はドキュメントの ID 列です。
- conf.set(“es.input.json”,“yes”)

  入力ファイルの形式を JSON として指定します。
- job.setInputFormatClass(TextInputFormat.class);

  入力ストリームの形式をテキストとして指定します。
- job.setOutputFormatClass(EsOutputFormat.class)

  出力形式を EsOutputFormat として指定します。
- job.setMapOutputKeyClass(NullWritable.class)

  Map の出力キー形式を NullWritable として指定します。
- job.setMapOutputValueClass(BytesWritable.class)

  このパラメーターには、Map の出力値形式を BytesWritable として指定します。
- FileInputFormat.setInputPaths(job, new Path(otherArgs[0]))

  HDFS にアップロードする必要があるファイルのパスを指定します。

3.7 Logstash のデプロイメント

環境の準備

1. ES インスタンス、および 自己構築クラスターと ES にアクセス可能な ECS インスタンスを購入します。 要件を満たす ECS インスタンスが既にある場合、追加の ECS インスタンスを購入する必要はありません。 バージョン 1.8 以降の JDK を準備します。

   クラシックネットワーク上の ECS インスタンスは、クラシックネットワークに関する問題 を介して VPC 内の ES サービスにアクセスできるのであれば、使用できます。
2. Logstash v5.5.3 をダウンロードします。

   Elasticsearch に対応するバージョンの Logstash を Elastic の Web サイトでダウンロードします (v5.5.3 を推奨)。

3. ダウンロードした Logstash パッケージを解凍します。

```
tar -xzvf logstash-5.5.3.tar.gz
# A stringent configuration file checking feature is added to Elasticsearch later than version 5.x.
```

テストケース

1. データアクセス用のユーザー名とパスワードの作成

   - ロールの作成

```
curl -XPOST -H "Content-Type: application/json" -u elastic:es-password http://***instanceId***.elasticsearch.aliyuncs.com:9200/_xpack/security/role/***role-name*** -d '{"cluster":["manage_index_template","monitor"],"indices":[{"names":["logstash-*"],"privileges":["write","delete","create_index"]}]}'
# es-password is the Kibana logon password.
# ***instanceId*** is the ES instance ID.
# ***role-name*** is the role name.
# The default index name of Logstash is in the format of logstash-current date. Therefore, the read and write permissions on the Logstash-* index must be assigned when you add a user role.
```

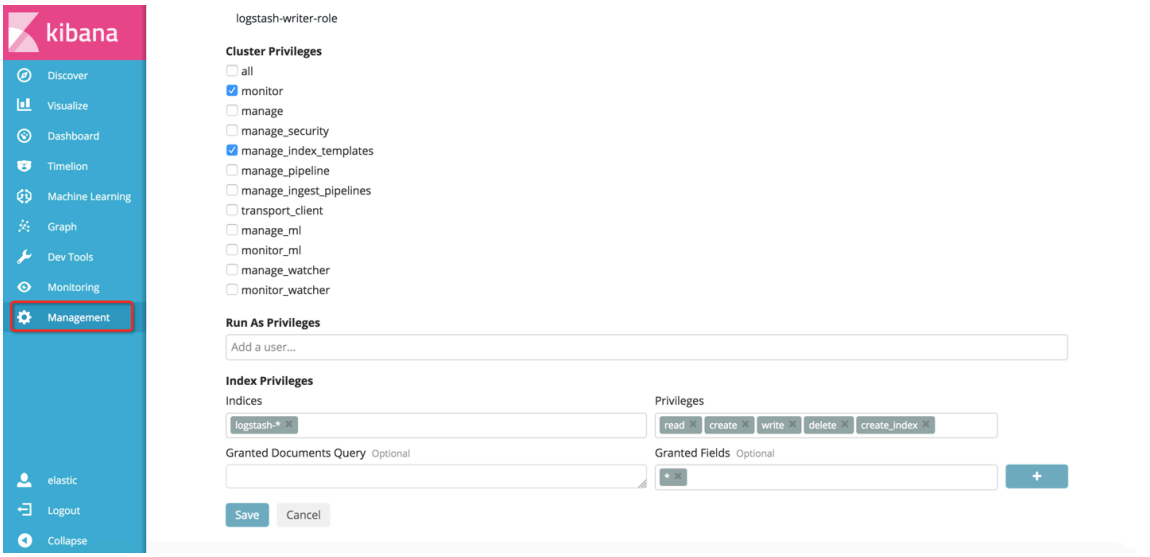
   - ユーザーの作成

```
curl -XPOST -H "Content-Type: application/json" -u elastic:es-password http://***instanceId***.elasticsearch.aliyuncs.com:9200/_xpack/security/user/***user-name*** -d '{"password":"***logstash-password***","roles":["***role-name***"],"full_name":"***your full name***"}'
# es-password is the Kibana logon password.
# ***instanceId*** is the ES instance ID.
# ***user-name*** is the user name for data access.
# ***logstash-password*** is the password for data access.
# ***role-name*** is the role name you created earlier.
# ***your full name*** is the full name of the current user.
```

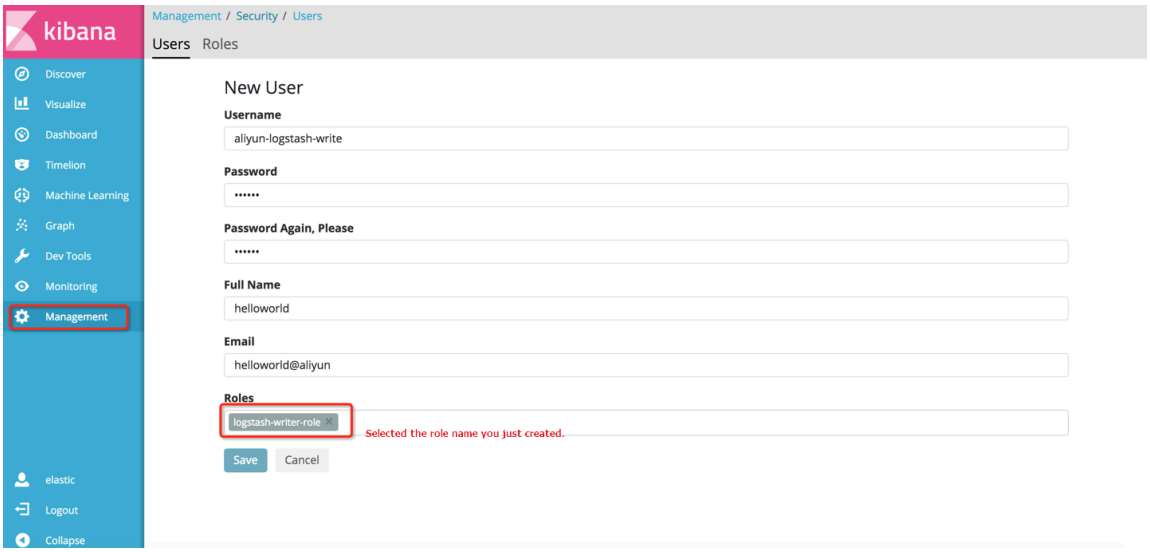
注：

ロールとユーザーは、Kibana ページでも作成できます。

- ロールの追加



- ユーザーの追加



2. conf ファイルの準備

詳細は、『Configuration file structure』をご参照ください。

例：

ECS インスタンスに `test.conf` ファイルを作成し、以下の設定を追加します。

```
 input  {
     file  {
         path  => "/ your / file / path / xxx "
        }
}
 filter  {
}
 output  {
   elasticsea  rch  {
```

```
    hosts  => [" http ://*** instanceId ***. elasticsea  rch .
 aliyuncs . com : 9200 "]
    user  => "*** user - name ***"
    password  => "*** logstash - password ***"
  }
}
# *** instanceId ***  is  the  ES  instance  ID .
# *** user - name ***  is  the  user  name  for  data  access
 .
# *** logstash - password ***  is  the  password  for  data
 access .
#  Place  the  user  name  and  password  in  quotation
 marks  to  prevent  errors  in  Logstash  startup  caused
 by  special  characters .
```

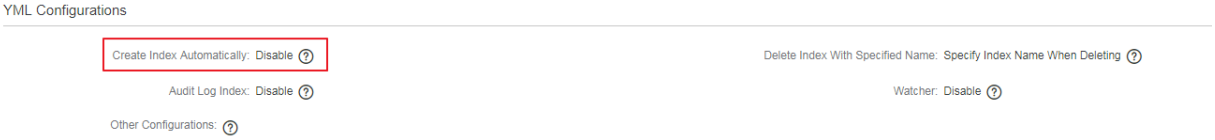
実行

conf ファイルに従って Logstash を実行します。

```
 bin / logstash  - f  path / to / your / test . conf
#  Logstash  provides  many  input ,  filter ,  and  output
 plugins .  Only  simple  configurat  ions  are  required  for
   data  transfer .
 This  example  shows  how  to  obtain  file  changes
 through  Logstash  and  submit  the  changed  data  to  the
   Elasticsea  rch  cluster .  All  the  new  contents  in
 the  monitored  file  can  be  automatica  lly  indexed  to
   the  Elasticsea  rch  cluster  by  Logstash .
```

よくある質問

クラスターによるインデックス自動作成を設定するには、どうすればよいですか。



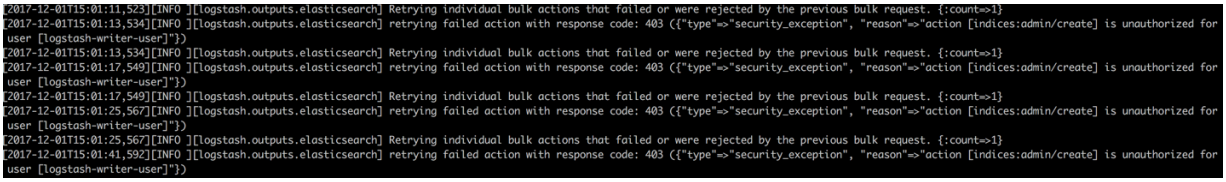
Elasticsearch では、ユーザーのデータ操作中のセキュリティを確保するため、デフォルトでインデックスの自動作成を許可していません。

Logstash は、 `create index` API を使用する代わりに、データのアップロード時にデータを送信することで、インデックスを作成します。 このため、Logstash を使用してデータをアップロードする前に、インデックスの自動作成を許可してください。

注：

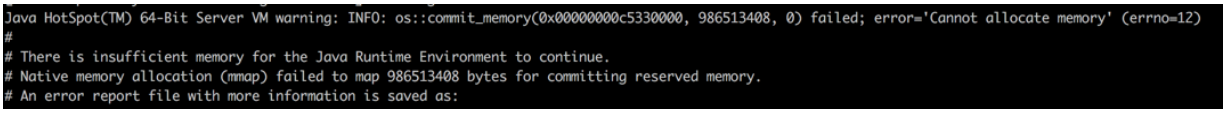
設定を変更し、確定後、ES クラスターが再起動します。

インデックスの作成権限がないというメッセージが表示されました。



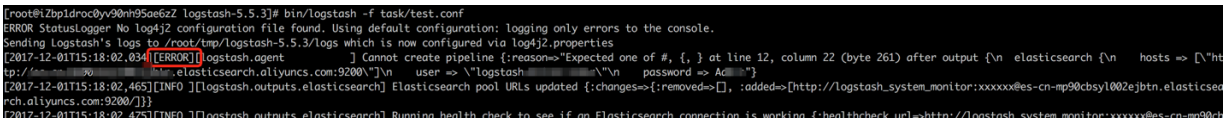
データアクセス用に作成したロールに `write`、`delete`、`create_index` 権限があるかどうかを確認してください。

メモリが不足しているというメッセージが表示されました。



デフォルトでは、Logstash のメモリは 1 GB です。ECS のメモリが不足している場合、`config/jvm.options` のメモリ設定を変更して、Logstash のメモリ使用を減らしてください。

test.conf 内のユーザー名とパスワードが引用符に囲まれていないというメッセージが表示されました。



`test.conf` ファイル内で、特殊文字を含むユーザー名またはパスワードが引用符で囲まれていないと、前述のエラーメッセージが表示されます。

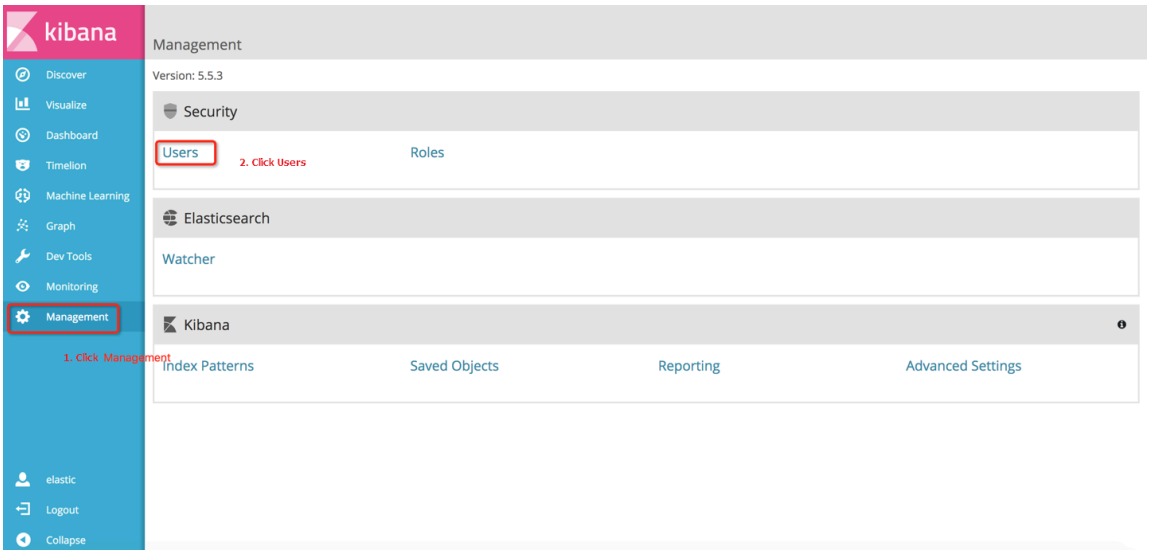
補足情報

Logstash ノードをモニタリングしてログを収集するには、以下の手順に従います。

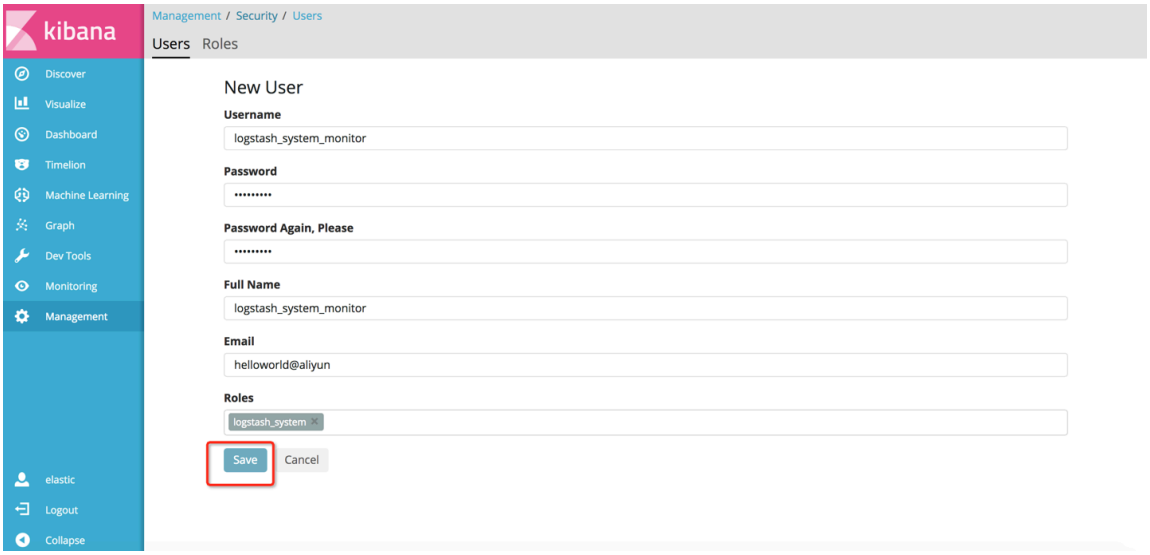
- Logstash 用の X-Pack プラグインをインストールします。詳細は、download link をご参照ください。
- ダウンロード後に X-Pack をデプロイします。
- ```
  bin/logstash-plugin install file:///path/to/file/x-pack-5.5.3.zip
  ```
- Logstash のモニタリングユーザーを追加します。Elasticsearch クラスターは、デフォルトで logstash_system ユーザーを無効にしています。logstash_system というロールを持つユーザーを作成する必要があります。ユーザー名を logstash_system にすることはできません。ユーザー名は変更できます。この例では、ユーザー名は logstash_system_monitor です。ユーザーを作成するには、次の 2 つの方法を推奨します。
  ```

- Kibana モジュールでモニタリングユーザーを作成する

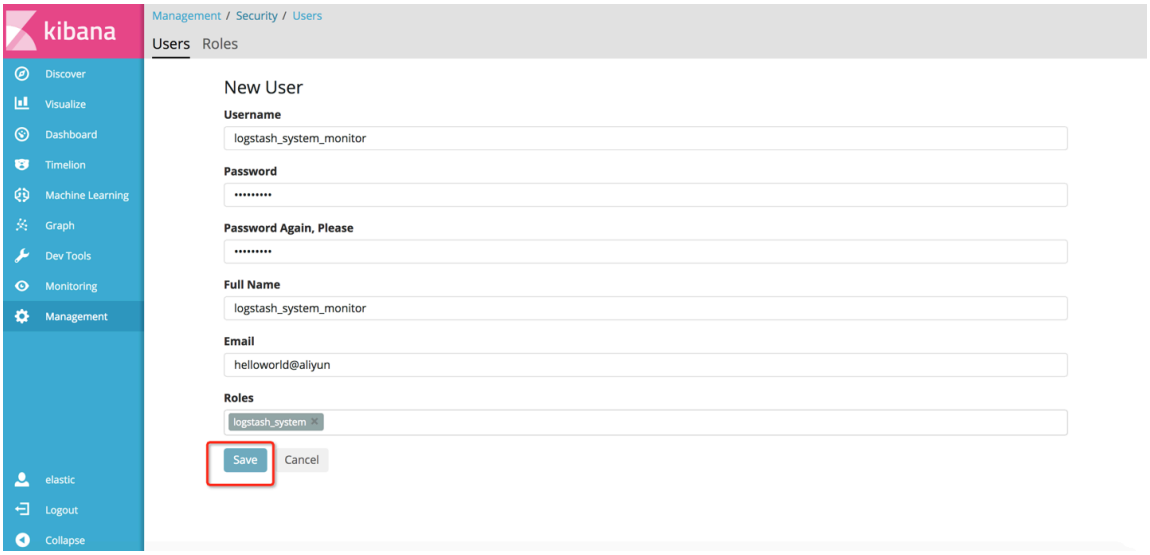
1. Kibana の [management] ページにログインし、次の図に従って操作します。



2. [Create User] ボタンをクリックします。



3. 必要な情報を入力します。 情報を保存して、送信します。



- コマンドでユーザーを追加する

```
curl -u elastic:es-password -XPOST http://***instanceId***.elasticsearch.aliyuncs.com:9200/_xpack/security/user/logstash_system_monitor -d '{"password":"***logstash-monitor-password***","roles":["logstash_system"],"full_name":"your full name"}'
#  es-password is the Kibana logon password.
# ***instanceId*** is the ES instance ID.
# ***logstash-monitor-password*** is the password of logstash_system_monitor.
```

3.8 ECS でホストされている ES インスタンスの移行

前提条件

このドキュメントでは、ECS でホストされている Elasticsearch インスタンスから Elasticsearch インスタンスにデータを移行する方法について説明します。 データを移行する前に、以下の要件を満たす必要があります。 以下の要件を満たしていない場合、「Logstash のデプロイメント」を参照し、他の移行ソリューションを使用してデータを移行してください。

- 自己構築 Elasticsearch インスタンスをホストする ECS インスタンスは、VPC ネットワークに接続されている必要があります。 ClassicLink を介して VPC ネットワークに接続された ECS インスタンスは、サポートされていません。 ECS インスタンスと Elasticsearch インスタンスは、同じ VPC ネットワークに接続されている必要があります。
- ECS インスタンスを使用して、reindex.sh スクリプトを実行することができます。 このタスクを実行するには、自己構築 Elasticsearch インスタンスと Elasticsearch インスタンス上で ECS インスタンスがポート `9200` にアクセスできる必要があります。
- VPC セキュリティグループは、IP ホワイトリスト内のすべての IP アドレスに対して ECS インスタンスへのアクセスを許可し、ポート `9200` を開く必要があります。

- VPC セキュリティグループは、すべての Elasticsearch インスタンスノードの IP アドレスに対して ECS インスタンスへのアクセスを許可する必要があります。 これらの IP アドレスは、Kibana コンソールで確認できます。
- スクリプトを実行する ECS インスタンスがソースおよびターゲット Elasticsearch インスタンス上のポート 9200 にアクセスできるかどうかを確認するには、ECS インスタンスで `curl -XGET http://<host>:9200` コマンドを実行します。

手順

1. インデックスを作成します。
2. データを移行します。

インデックスの作成

ソースクラスターのインデックスに基づき、ターゲット Elasticsearch インスタンスにインデックスを作成する必要があります。 動的インデックス作成と動的マッピング (非推奨) を有効にすることで、ターゲットクラスターにインデックスを作成することもできます。 動的インデックス作成を有効にする前に、自動インデックス作成を有効にする必要があります。

以下のセクションでは、Python スクリプト (`indiceCreate.py`) を示します。 ソースクラスターからターゲットクラスターにすべてのインデックスをコピーできます。 `シャード数`と`レプリカ数`のみが設定されています。 これ以外を設定する必要があります。

注:

cURL コマンドを実行したときに以下のエラーが発生した場合、コマンドに `-H "Content-Type: application/json"` パラメーターを追加し、コマンドを再実行してください。

```
`{" error ":" Content - Type header [ application / x - www - form - urlencoded ] is not supported "," status ": 406 }`
```

```
// Obtain all the indexes on the source cluster. If you do not have the required permissions, remove the "-u user:pass" parameter. Make sure that you have replaced oldCluster Host with the name of the ECS instance that hosts the source cluster.
curl -u user:pass -XGET http://oldCluster Host/_cat/indices | awk '{print $3}'
// Based on the returned indexes, obtain the setting and mapping of the index that you need to migrate for the specified user. Make sure that you have replaced indexName with the index name that you need to query.
curl -u user:pass -XGET http://oldCluster Host/indexName/_settings,_mapping?pretty=true
```

```
  // Create a new index in the target cluster
according to the _settings and _mapping settings that
  you have obtained from the preceding step . You
can set the number of index replicas to zero to
  accelerate the data synchroniz ation process , and
change the number to one after the migration has
completed .
 // ewClusterH ost indicates the ECS instance that
hosts the target cluster , testindex indicates the name
  of the index that you have created , and testtype
indicates the type of the index .
  curl - u user : pass - XPUT http ://< newCluster Host >/<
testindex > - d '{
   " testindex " : {
       " settings " : {
           " number_of_ shards " : " 5 ", // Set the number
  of shards for the correspond ing index on the
source cluster , for example , 5
           " number_of_ replicas " : " 0 " // Set the number
of index replicas to zero
         }
       },
       " mappings " : { // Set the mapping for the index
  on the source cluster . For example , you can set
the mapping as follows
           " testtype " : {
               " properties " : {
                   " uid " : {
                       " type " : " long "
                   },
                   " name " : {
                       " type " : " text "
                   },
                   " create_tim e " : {
                     " type ": " long "
                   }
               }
           }
       }
   }
}'
```

同期プロセスの加速化

注:

インデックスが大きすぎる場合、移行前にレプリカの数を 0 に、更新間隔を -1 に設定できます。 データを移行後、レプリカと更新の設定を以前の値に設定します。 これにより、同期プロセスが高速化されます。

```
// You can set the number of index replicas to zero
   and disable refresh , to accelerate the migration
 process .
 curl - u user : password - XPUT ' http ://< host : port >/
 indexName / _settings ' - d ' {
       " number_of_ replicas " : 0 ,
       " refresh_in terval " : "- 1 "
}'
```

```
//  After   the   data   has   been   migrated ,  set   the   number
   of   index   replicas   to   ` 1 `   and   the   refresh   interval
 to   ` 1 ` ( default   value ,  which   means   1   second ).
 curl  - u   user : password  - XPUT  ' http ://< host : port >/
 indexName / _settings ' - d ' {
        " number_of_  replicas " :  1 ,
        " refresh_in  terval " : " 1s "
}'
```

データ移行

移行後のデータ整合性を確保するため、ソースクラスターでの書き込み操作を停止する必要があります。 読み取り操作を停止する必要はありません。 移行プロセスが完了したら、読み取り操作と書き込み操作をターゲットクラスターに切り替えます。 ソースクラスターで書き込み操作を停止しないと、データの不整合が発生する可能性があります。

注：

- 次の方法でデータを移行するとき、 `IP` アドレスとポートを使用してソースクラスターにアクセスする場合、ターゲットクラスターの `YML` ファイルに `reindex` ホワイトリストを設定し、ソースクラスターの IP アドレスを追加する必要があります。例： `reindex . remote . whitelist :  1 . 1 . 1 . 1 : 9200 , 1 . 2 . *. *:* *. *:*`
- ドメイン名を使用してソースクラスターにアクセスする場合、 `http :// host : port / path` の形式を使用しないでください。 ドメイン名にパスを含めることはできません。

- 少量のデータを移行する

  `reindex . sh` スクリプトを実行します。

```
#! / bin / bash
#  file : reindex . sh
 indexName =" The   name   of   the   index "
 newCluster  User =" The   username   that   is   used   to   log
 on   to   the   target   cluster "
 Newcluster  pass  = " The   password   that   is   used   to   log
   on   to   the   target   cluster "
 newCluster  Host =" The   ECS   instance   that   hosts   the
 target   cluster "
 Oldcluster  user  = " The   username   that   is   used   to   log
   on   to   the   source   cluster "
 Oldcluster  pass  = " The   password   that   is   used   to   log
   on   to   the   source   cluster "
#  The   address   of   the   ECS   instance   that   hosts   the
 source   cluster   must   be   in   this   format : [ scheme ]://[
 host ]:[ port ].  Example :  http :// 10 . 37 . 1 . 1 : 9200 .
 Oldcluster  host  = " The   ECS   instance   that   hosts   the
 source   cluster "
 curl  - u  ${ newCluster  User }:${ newCluster  Pass } - XPOST  "
 http ://${ newCluster  Host }/ _reindex ?  pretty " - H  " Content
 - Type :  applicatio  n / json " - d '{
    " source ": {
        " remote ": {
            " host ": "'${ oldCluster  Host }'",
```

```
            " username ": "'${ oldCluster  User }'",
            " password ": "'${ oldCluster  Pass }'"
        },
        " index ": "'${ indexName }'",
        " query ": {
            " match_all ": {}
        }
    },
    " dest ": {
       " index ": "'${ indexName }'"
    }
}'
```

- **大量のデータを移行する (削除操作なし、更新時間ありの場合)**

  大量のデータがあり、かつ削除操作がない場合は、ローリング移行を使用することで書き込み操作の停止時間を短縮できます。 ローリング移行では、更新時刻を示す時系列属性がデータスキーマに必要です。 データの移行が完了したら、書き込み操作を停止し、増分データを移行します。 読み取り操作と書き込み操作をターゲットクラスターに切り替えます。

```
#! / bin / bash
#  file :  circleRein  dex . sh
#  CONTROLLIN  G   STARTUP :
#  This   script   is   used   to   remotely   rebuild   the   index
   using   the   reindex   operation .  Requiremen  ts :
# 1 .  You   have   created   the   index   on   the   target
 cluster ,  or   the   target   cluster   supports   automatic
 index   creation   and   dynamic   mapping .
#  2   You   must   configure   an   IP   whitelist   in   the
 YML   file   of   the   target   cluster :  reindex . remote .
 whitelist :  172 . 16 . 123 . *: 9200
# 3 .  You   need   to   specify   the   ECS   instance   address
 in   the   following   format : [ scheme ]://[ host ]:[ port ].
 USAGE =" Usage :  sh   circleRein  dex . sh  < count >
        count :  The   number   of   executions .  A   negative
 number   indicates   loop   execution .  You   can   set   this
 parameter   to   perform   the   reindex   operation   only   once
   or   multiple   times .
 For   example :
         sh   circleRein  dex . sh   1
         sh   circleRein  dex . sh   5
         sh   circleRein  dex . sh  - 1 "
 indexName =" The   name   of   the   index "
 newCluster  User =" The   username   that   is   used   to   log
 on   to   the   target   cluster "
 newCluster  Pass =" The   password   that   is   used   to   log
 on   to   the   target   cluster "
 oldCluster  User =" The   username   that   is   used   to   log
 on   to   the   source   cluster "
 oldCluster  Pass =" The   password   that   is   used   to   log
 on   to   the   source   cluster "
##  http :// myescluste  r . com
 newCluster  Host =" The   host   of   the   target   cluster "
#  You   need   to   address   of   the   ECS   instance   that
 hosts   the   source   cluster   in   the   following   format : [
 scheme ]://[ host ]:[ port ].  Example :  http :// 10 . 37 . 1 . 1
 : 9200
 oldCluster  Host =" The   ECS   instance   that   hosts   the
 source   cluster "
```

```
timeField =" The   field   that   specifies   the   time   window
during   which   the   incrementa  l   data   is   migrated "
reindexTim  es = 0
lastTimest  amp = 0
curTimesta  mp =` date  +% s `
hasError = false
function   reIndexOP () {
    reindexTim  es =$[${ reindexTim  es } +  1 ]
    curTimesta  mp =` date  +% s `
    ret =` curl  - u  ${ newCluster  User }:${ newCluster  Pass } -
XPOST  "${ newCluster  Host }/ _reindex ?  pretty " - H  " Content
- Type :  applicatio  n / json " - d  '{
       " source ": {
           " remote ": {
               " host ": "'${ oldCluster  Host }'",
               " username ": "'${ oldCluster  User }'",
               " password ": "'${ oldCluster  Pass }'"
           },
           " index ": "'${ indexName }'",
           " query ": {
               " range " : {
                   "'${ timeField }'" : {
                       " gte " : '${ lastTimest  amp }',
                       " lt " : '${ curTimesta  mp }'
                   }
               }
           }
       },
       " dest ": {
           " index ": "'${ indexName }'"
       }
   }'`
    lastTimest  amp =${ curTimesta  mp }
    echo  "${ reindexTim  es }  reindex   operations   have   been
  performed .  The   last   reindex   operation   is   completed
at  ${ lastTimest  amp }  Result :${ ret }"
    if  [[ ${ ret } == * error * ]];  then
        hasError = true
        echo  " An   unknown   error   occurred   while
performing   this   operation .  All   subsequent   operations
have   been   suspended ."
    fi
}
function   start () {
   ##  A   negative   number   indicates   loop   execution .
    if  [[ $ 1  - lt   0  ]];  then
        while  :
        do
            reIndexOP
        done
    elif  [[ $ 1  - gt   0  ]];  then
        k = 0
        while  [[  k  - lt  $ 1  ]] && [[ ${ hasError } ==  false
]];  do
            reIndexOP
            let  ++ k
        done
    fi
}
##  main
if  [ $# - lt   1  ];  then
    echo  "$ USAGE "
    exit   1
fi
```

```
echo " Start the reindex operation for index ${
indexName }"
start $ 1
echo " You have performed ${ reindexTim es } reindex
operations "
```

- 大量のデータを移行する (削除操作または更新時間なしの場合)

  大量のデータを移行する必要があり、マッピングに更新時間フィールドが定義されていない場合、ソースクラスターへのアクセスに使用されるコードに更新時刻フィールドを追加する必要があります。フィールドを追加した後、既存のデータを移行してから、前述のデータ移行計画で説明されているローリング移行を使用して増分データを移行できます。

  以下のスクリプトは、更新時刻フィールドなしで既存のデータを移行する方法を示しています。

```
#! / bin / bash
# file : miss . sh
indexName =" The name of the index "
newCluster User =" The username that is used to log
on to the target cluster "
Newcluster pass = " The password that is used to log
on to the target cluster "
newCluster Host =" The ECS instance that hosts the
target cluster "
Oldcluster user = " The username that is used to log
on to the source cluster "
Oldcluster pass = " The password that is used to log
on to the source cluster "
# The address of the ECS instance that hosts the
source cluster must be in this format : [ scheme ]://[
host ]:[ port ]. Example : http :// 10 . 37 . 1 . 1 : 9200 .
oldCluster Host =" The ECS instance that hosts the
source cluster "
timeField =" updatetime "
curl - u ${ newCluster User }:${ newCluster Pass } - XPOST "
http ://${ newCluster Host }/ _reindex ? pretty " - H " Content
- Type : applicatio n / json " - d '{
   " source ": {
       " remote ": {
           " host ": "'${ oldCluster Host }'",
           " username ": "'${ oldCluster User }'",
           " password ": "'${ oldCluster Pass }'"
       },
       " index ": "'${ indexName }'",
       " query ": {
           " bool ": {
               " must_not ": {
                   " exists ": {
                       " field ": "'${ timeField }'"
                   }
               }
           }
       }
   },
   " dest ": {
      " index ": "'${ indexName }'"
   }
```

```
}'
```

- 書き込み操作を停止せずにデータを移行する

  この機能は、今後提供予定です。

バッチ作成操作によるソースクラスターのインデックスのレプリケート

以下は、ソースクラスターからターゲットクラスターにインデックスをレプリケートする Python スクリプトです。 新しく作成されたインデックスレプリカのデフォルト数は 0 です。

```
#! / usr / bin / env   python
# -*-  coding :  UTF - 8  -*-
#  File   name  :  indiceCrea  te . py
 import   sys
 import   base64
 import   time
 Import   httplib
 import   json
##  The   ECS   instance   that   hosts   the   source   cluster  ( ip
 + port )
 oldCluster  Host  = " old - cluster . com "
#  The   username   that   is   used   to   log   on   to   the
 source   cluster .  The   username   field   can   be   left   empty
 oldCluster  UserName  = " old - username "
##  The   password   that   is   used   to   log   on   to   the
 source   cluster .  The   password   field   can   be   left   empty
 oldCluster  Password  = " old - password "
##  The   ECS   instance   that   hosts   the   target   cluster  ( ip
 + port )
 newCluster  Host  = " new - cluster . com "
##  The   username   that   is   used   to   log   on   to   the
 target   cluster .  The   username   field   can   be   left   empty
 newCluster  User  = " new - username "
##  The   password   that   is   used   to   log   on   to   the
 target   cluster .  The   password   field   can   be   left   empty
 newCluster  Password  = " new - password "
 DEFAULT_RE  PLICAS  =  0
 def   httpReques  t ( method ,  host ,  endpoint ,  params ="",
 username ="",  password =""):
     conn  =  httplib . HTTPConnec  tion ( host )
     headers  = {}
     if  ( username  ! = "") :
        ' Hello  { name },  your   age   is  { age } !'. format ( name
  = ' Tom ',  age  = ' 20 ')
         base64stri  ng  =  base64 . encodestri  ng ('{ username }:{
 password }'. format ( username  =  username ,  password  =  password
 )). replace ('\ n ', '')
         headers [" Authorizat  ion "] = " Basic  % s " %  base64stri
 ng ;
     if  " GET " ==  method :
         Content - Type :  applicatio  n / x - www - form -
 urlencoded
         conn . request ( method = method ,  url = endpoint ,  headers
 = headers )
     else  :
         Headers  [" Content - Type "] = " applicatio  n / JSON "
         conn . request ( method = method ,  url = endpoint ,  body =
 params ,  headers = headers )
     response  =  conn . getrespons  e ()
     res  =  response . read ()
```

```
    return   res
def   httpGet ( host ,  endpoint ,  username ="",  password =""):
    return   httpReques  t (" GET ",  host ,  endpoint , "",
username ,  password )
def   httpPost ( host ,  endpoint ,  params ,  username ="",
password =""):
    return   httpReques  t (" POST ",  host ,  endpoint ,  params ,
username ,  password )
def   httpPut ( host ,  endpoint ,  params ,  username ="",  password
=""):
    return   httpReques  t (" PUT ",  host ,  endpoint ,  params ,
username ,  password )
def   getIndices ( host ,  username ="",  password =""):
    endpoint  = "/ _cat / indices "
    indicesRes  ult  =  httpGet ( oldCluster  Host ,  endpoint ,
oldCluster  UserName ,  oldCluster  Password )
    indicesLis  t  =  indicesRes  ult . split ("\ n ")
    indexList  = []
    for   indices   in   indicesLis  t :
        if  ( indices . find (" open ") >  0 ):
            indexList . append ( indices . split ()[ 2 ])
    return   indexList
def   getSetting  s ( index ,  host ,  username ="",  password =""):
    endpoint  = "/" +  index  + "/ _settings "
    indexSetti  ngs  =  httpGet ( host ,  endpoint ,  username ,
password )
    print   index  + "   The   original   settings : \ n " +
indexSetti  ngs
    settingsDi  ct  =  json . loads ( indexSetti  ngs )
   ##  The   number   of   shards   equals   the   number   of
indexes   on   the   source   cluster   by   default
    number_of_  shards  =  settingsDi  ct [ index ][" settings "]["
index "][" number_of_  shards "]
   ##  The   default   number   of   replicas   is   0
    number_of_  replicas  =  DEFAULT_RE  PLICAS
    newSetting  = "\" settings \": {\" number_of_  shards \": %
s , \" number_of_  replicas \": % s }" % ( number_of_  shards ,
number_of_  replicas )
    return   newSetting
def   getMapping ( index ,  host ,  username ="",  password =""):
    endpoint  = "/" +  index  + "/ _mapping "
    indexMappi  ng  =  httpGet ( host ,  endpoint ,  username ,
password )
    print   index  + " The   original   mappings : \ n " +
indexMappi  ng
    mappingDic  t  =  json . loads ( indexMappi  ng )
    mappings  =  json . dumps ( mappingDic  t [ index ][" mappings
"])
    newMapping  = "\" mappings \" : " +  mappings
    return   newMapping
def   createInde  xStatement ( oldIndexNa  me ):
    settingStr  =  getSetting  s ( oldIndexNa  me ,  oldCluster  Host
,  oldCluster  UserName ,  oldCluster  Password )
    mappingStr  =  getMapping ( oldIndexNa  me ,  oldCluster  Host ,
oldCluster  UserName ,  oldCluster  Password )
    createstat  ement  = "{\ n " +  str ( settingStr ) + ",\ n " +
str ( mappingStr ) + "\ n }"
    return   createstat  ement
def   createInde  x ( oldIndexNa  me ,  newIndexNa  me =""):
    if  ( newIndexNa  me  == "") :
        newIndexNa  me  =  oldIndexNa  me
    createstat  ement  =  createInde  xStatement ( oldIndexNa  me )
    print  " new   index " +  newIndexNa  me  + " settings   and
mappings : \ n " +  createstat  ement
```

```
    endpoint  = "/" +  newIndexNa  me
    createResu  lt  =  httpPut ( newCluster  Host ,  endpoint ,
 createstat  ement ,  newCluster  User ,  newCluster  Password )
    print  " new   index " +  newIndexNa  me  + " creation   result
 :" +  createResu  lt
##  main
 indexList  =  getIndices ( oldCluster  Host ,  oldCluster  UserName
 ,  oldCluster  Password )
 systemInde  x  = []
 for   index   in   indexList :
    if  ( index . startswith ("")):
        systemInde  x . append ( index )
    else  :
        createInde  x ( index ,  index )
 if  ( len ( systemInde  x ) >  0 ) :
    for   index   in   systemInde  x :
        print   index  + " It   may   be   a   system   index   that
  will   not   be   recreated .  Create   the   index   based   on
 your   needs ."
```