Exemples de Jobs d'intégration de données
# Table des matières

**Copyright** ............................................................................................................................................. 4

**Exemple de Job comprenant un tMap** ............................................................................................... 5
- Données d’entrée ........................................................................................................................................... 5
- Données de sortie ........................................................................................................................................... 5
- Reference data ............................................................................................................................................... 6
- Transformer le scénario en Job ...................................................................................................................... 6

**Utilisation du Dynamic Schema pour charger dynamiquement des données dans les tables de base de données** .......................................................................................................................... 15
- Prérequis pour cet exemple de Job .................................................................................................................. 15
- Créer le Job parent .......................................................................................................................................... 16
- Créer le Job enfant .......................................................................................................................................... 16
-Configurer le Job parent ................................................................................................................................... 17
-Configurer le Job enfant.................................................................................................................................... 19
-Exécuter le Job parent et vérifier le résultat de chargement des données ....................................................... 21

**Utilisation de la fonctionnalité Output Stream** .................................................................................... 23
- Données d’entrée ........................................................................................................................................... 23
- Données de sortie ........................................................................................................................................... 23
- Transformer le scénario en Job ....................................................................................................................... 23

**Utilisation de la fonctionnalité de chargement implicite des contextes** .................................................. 30
- Créer le Job et définir des variables de contexte ............................................................................................ 30
-Configurer les composants ............................................................................................................................... 33
-Configurer la fonctionnalité de chargement implicite des contextes ................................................................ 34
-Exécuter le Job ................................................................................................................................................ 35

**Utilisation de la fonctionnalité Exécution en multi thread pour exécuter des Jobs en parallèle** ............ 36
- Préparer les Jobs pour lire des données des employés dans différents contextes .......................................... 36
- Mettre en place un Job parent pour exécuter les Jobs en parallèle ................................................................ 37
-Exécuter les Jobs ........................................................................................................................................... 38

**Appel du service Web REST de la MetaServlet pour exécuter une tâche** .......................................... 40
- Déposer et relier les composants .................................................................................................................... 40
-Configurer les variables de contexte .................................................................................................................. 40
-Configurer la Routine ....................................................................................................................................... 41
-Configurer les composants ............................................................................................................................... 42

**Exemple de Job comprenant des règles de validation** ......................................................................... 44
- Utiliser une règle de validation par référence ............................................................................................... 44
- Utiliser une règle de validation simple .......................................................................................................... 47
Utiliser le conneteur JDBC pour se connecter à Amazon Athena........................................ 52
  Configurer une connexion JDBC pour se connecter à Amazon Athena........................................ 52
  Configurer un Job pour se connecter à Amazon Athena.............................................................. 55

Ordonnancement et exécution d’un Job Talend sur le moteur Google Kubernetes......................................................... 57
  Prérequis à cet exemple.......................................................................................................................... 57
  Publier votre Job en tant qu’image Docker............................................................................................ 57
  Configurer Google Kubernetes Engine..................................................................................................... 61
  Ordonnancer le Job...................................................................................................................................... 62
Copyright

Convient à la version 7.3.1. Annule et remplace toute version antérieure de ce guide.

Copyright © Talend 2020. Tous droits réservés.

Le contenu de ce document est correct à la date de publication.

Cependant, des mises à jour peuvent être disponibles dans la version en ligne, sur Talend Help Center.

Mentions légales

Talend est une marque déposée de Talend, Inc.

Tous les noms de marques, de produits, les noms de sociétés, les marques de commerce et de service sont la propriété de leurs détenteurs respectifs.

Contrat-cadre de licence utilisateur final

Le progiciel décrit dans cette documentation est fourni sous les termes du contrat-cadre Talend de licence utilisateur final (progiciel et souscription) pour les produits commerciaux. Si vous utilisez le progiciel, vous indiquez que vous avez compris et accepté les termes et les conditions de ce contrat-cadre.

Exemple de Job comprenant un tMap

Pour illustrer le fonctionnement du Studio Talend, vous trouverez ci-dessous un scénario reflétant un cas d’utilisation réel. Dans ce scénario, vous devez charger un fichier dans une table MySQL en appliquant des transformations à la volée. Et dans une étape suivante, vous sélectionnez les données à charger en appliquant un filtre dynamique.

Avant de commencer le Job, vérifiez les données en entrée (Input) et les données attendues en sortie (Output).

Données d’entrée

Le contenu du fichier en entrée est une liste des clients de toutes les régions de l’état de Californie. Ces données seront donc chargées dans une table de données.

La structure du fichier appelée Schema dans le Studio Talend comprend les colonnes suivantes :
- First name (prénom)
- Last name (nom)
- Address (adresse)
- City (ville)

Données de sortie

Vous souhaitez charger uniquement les données des clients habitant dans certaines régions (Counties) de la Californie dans la nouvelle base de données : les régions d’Orange et de Los Angeles.

La structure de la table est légèrement différente, ainsi les données devant être chargées dans la table de données doivent être structurées de la manière suivante :
- Key (Clé, Type Integer)
- Name (Type chaîne, longueur max. 40)
- Address (Type chaîne, longueur max. 40)
- County (Type chaîne, longueur max. 40)

Pour charger cette table, vous devez utiliser les processus de mapping suivants :

La colonne Key est alimentée par un entier auto-incrémenté.

La colonne Name est renseignée avec une concaténation des données First Name (prénom) et Last Name (nom).

Les données de la colonne Address sont les mêmes que celles de la colonne County du fichier d’entrée et elles seront mises en majuscule avant d’être chargées.

La colonne County est alimentée par le nom de la région dans laquelle se situe la ville. Un fichier de référence vous aidera à filtrer les villes des régions d’Orange et de Los Angeles.
**Reference data**

Etant donné que les données des régions d’Orange et de Los Angeles doivent être chargées dans la base de données, vous devez mapper les villes de Californie avec leur région respective, afin de pouvoir filtrer uniquement les villes d’Orange et de Los Angeles.

Pour cela, utilisez un fichier de référence contenant la liste des villes situées dans ces régions, par exemple :

<table>
<thead>
<tr>
<th>City (ville)</th>
<th>County (comté)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agoura Hills</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>Alhambra</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>Aliso Viejo</td>
<td>Orange</td>
</tr>
<tr>
<td>Anaheim</td>
<td>Orange</td>
</tr>
<tr>
<td>Arcadia</td>
<td>Los Angeles</td>
</tr>
</tbody>
</table>

Le fichier de référence de ce Job se nomme *LosAngelesandOrangeCounties.txt*.

**Transformer le scénario en Job**

Pour mettre ce scénario en pratique, séparez ce Job en quatre étapes.

1. Création du Job, configuration des paramètres et lecture du fichier d’entrée
2. Mapping et transformations de données
3. Définition des paramètres du fichier de référence, mapping correspondant à l’aide du composant `tMap` et sélection du mode Inner Join.
4. Redirection des données en sortie dans une table MySQL

**Étape 1 : Création du Job, définition des données d’entrée, lecture du fichier**

**Procédure**

1. Après avoir lancé le Studio Talend, créez un projet en local ou importez un projet démo si vous lancez le Studio Talend pour la première fois.
2. Pour créer le Job, cliquez-droit sur **Job Designs** dans le **Repository** et sélectionnez **Create Job**.
3. Dans la boîte de dialogue qui s’ouvre, seul le premier champ (Name) est requis. Saisissez *California1* et cliquez sur **Finish**.

   Un Job vide s’ouvre ensuite dans la fenêtre principale et la Palette de composants techniques apparaît (par défaut, à droite du Studio) affichant une dizaine de familles de composants, notamment : Databases, Files, Internet, Data Quality, des centaines de composants sont disponibles.
4. Pour lire le fichier *California_Clients*, utilisez le composant `tFileInputDelimited`. Ce composant se trouve dans la famille **File** > **Input** de la Palette. Cliquez sur ce composant et placez-le à la gauche de l’espace de modélisation.
5. Définissez maintenant les propriétés de lecture de ce composant : chemin d’accès, séparateur de colonnes, encodage, etc. Pour ce faire, utilisez le Metadata Manager. Cet outil possède de nombreux assistants qui vous aideront à définir les paramètres nécessaires et vous permettront de conserver ces propriétés qui pourront être réutilisées en un seul clic dans futurs Jobs.


L’assistant spécifique aux fichiers délimités s’ouvre :

- Dans l’étape 1, seul le champ Name est requis : À l’étape 1, seul le champ Name est obligatoire : saisissez le nom California_clients et passez à l’étape suivante.
- A l’étape 2, sélectionnez le fichier d’entrée (California_Clients.csv) via le bouton Browse…. Un extrait du fichier apparaît immédiatement dans la zone File viewer en bas de l’assistant, afin que vous puissiez en vérifier le contenu. Cliquez sur Next.
- A l’étape 3, nous définirons les paramètres de fichiers suivants : encodage des fichiers, délimitations de lignes et de colonnes… Puisque votre fichier d’entrée est standard, vous pouvez garder la plupart des valeurs par défaut. La première ligne de votre fichier est un en-tête contenant les noms des colonnes. Pour récupérer automatiquement ces noms, cochez la case Set heading row as column names et cliquez sur Refresh Preview. Cliquez sur Next pour passer à l’étape suivante.
- A l’étape 4, définissez chaque colonne de votre fichier. L’assistant intègre des algorithmes qui essayent de deviner le type et la longueur des données contenues dans les colonnes du fichier en analysant les premières lignes. La description des données (appelé schéma dans le Studio Talend) peut être modifiée à tout moment. Pour ce scénario particulier, ces informations peuvent être gardées telles quelles.

La métadonnée California_clients est maintenant définie.

Vous pouvez donc l’utiliser dans votre composant d’entrée. Sélectionnez le composant tFileInputDelimited que vous avez déposé dans l’espace de modélisation et sélectionnez la vue Component Settings dans le bas de la fenêtre.

7. Sélectionnez l’onglet vertical Basic settings. Dans cet onglet, vous trouverez toutes les propriétés techniques nécessaires au composant. Au lieu de les saisir une à une, utilisez la métadonnée que vous venez de créer.


Notez que tous les paramètres sont automatiquement renseignés.
A cette étape, terminez votre flux en envoyant tout simplement les données lues dans le fichier d’entrée vers une sortie standard (StdOut).

9. Pour ce faire, ajoutez un composant **tLogRow** (de la famille **Logs & Errors**). Pour lier ces deux composants, cliquez-droit sur le composant d’entrée et sélectionnez **Row > Main**. Puis cliquez sur le composant de sortie **tLogRow**.

10. Ce Job est maintenant prêt à être exécuté. Pour l’exécuter, sélectionnez la vue **Run** dans le bas de la fenêtre.

11. Activez les statistiques en cochant la case **Statistics** dans l’onglet **Advanced settings** de la vue **Run**, puis exécutez le Job en cliquant sur le bouton **Run**, dans l’onglet **Basic Run**.
Étape 2 : Mapping et transformations

Vous allez maintenant enrichir votre Job en ajoutant des transformations à la volée. Pour effectuer ces transformations, utilisez le composant tMap dans votre Job.

Ce composant est multiple et peut gérer des :

- entrées et sorties multiples,
- recherches de référence (simple, produit cartésien, première et dernière correspondance, etc.),
- jointures (inner join, outer join),
- transformations,
- rejets,
- etc.

Procédure

1. Supprimez la connexion reliant vos deux composants via un clic-droit sur cette connexion et en sélectionnant l’option Delete. Puis placez le tMap entre les deux autres composants Processing et reliez le composant tMap au composant d’entrée comme vous l’avez fait précédemment.

2. Pour lier le composant tMap à la sortie standard, cliquez-droit sur le tMap et sélectionnez Row > *New Output* (Main) et cliquez sur le composant tLogRow pour créer la connexion. Saisissez
Exemple de Job comprenant un tMap

1. out1 dans la boîte de dialogue. Logiquement, une boîte de dialogue apparaît (pour la rétro-propagation des schémas), ignorez-la en cliquant sur No.

3. Double-cliquez sur le tMap pour accéder à son interface.

   A gauche, vous trouverez le schéma (description) de votre fichier d’entrée (row1). A droite, votre sortie est encore vide pour le moment (out1).

4. Déposez les colonnes FirstName et LastName de la gauche vers la droite dans la colonne Name, comme le montre la capture d’écran suivant. Puis déposez les autres colonnes Address et City dans leur ligne respective.

5. Effectuez les transformations suivantes sur chaque colonne :
   • Changez les données de la colonne Name de la manière suivante : row1.FirstName + " " + row1.LastName. Cette action concatène les colonnes Firstname et Lastname dans une seule colonne.
   • Changez les données de la colonne Address de la manière suivante : row1.Address.toUpperCase(). Cette action met l’adresse en majuscule.

6. Supprimez la colonne LastName de la table out1out1, et augmentez la longueur des colonnes restantes. Pour cela, cliquez sur l’onglet Schema Editor situé en bas de l’éditeur du Map Editor et procédez comme suit :

   1. Sélectionnez la colonne à supprimer du schéma, et cliquez sur l’icône représentant une croix rouge.
   2. Sélectionnez la colonne dont vous souhaitez augmenter la longueur.
   3. Saisissez la longueur que vous voulez dans la colonne Length. Dans cet exemple, modifiez la longueur de chaque colonne restante en 40.

   Remarque: Comme les noms et prénoms des clients sont concaténés, il est nécessaire d’augmenter la longueur de la colonne nom, afin de prendre en compte la longueur complète du nom. Aucune transformation n’est effectuée sur la colonne City.

7. Cliquez sur OK pour valider les modifications et fermer l’éditeur.

8. Exécutez votre Job.
**Exemple de Job comprenant un tMap**

**Résultats**
L’adresse a été mise en majuscule et les prénoms et noms ont été regroupés dans une seule colonne.

**Étape 3 : définition du fichier de référence, mapping des données de référence, sélection du mode Inner Join**

**Procédure**

1. Définissez la métadonnée correspondant au fichier `LosAngelesandOrangeCounties.txt` à l’aide de l’assistant, comme vous l’avez fait dans l’étape 1 avec le fichier `California_clients`.

   A l’étape 1 de l’assistant, nommez cette entrée de métadonnée : `LA_Orange_cities`.

2. Puis déposez cette nouvelle métadonnée en haut de l’espace de modélisation, cela créera automatiquement un composant de lecture pointant vers cette métadonnée.

3. Reliez ce composant au `tMap`. 
4. Double-cliquez de nouveau sur le composant **tMap** pour ouvrir son interface. Notez que la table de référence (**row2**) correspondant au fichier LosAngelesandOrangeCounties.txt, apparaît à gauche de la fenêtre dans la zone Input d’entrée, juste en dessus de votre flux d’entrée principal (**row1**).

5. Maintenant, définissez la jointure entre le flux principal et le flux de référence.

Dans ce scénario, la jointure est simple à définir puisque la colonne **City** est présente dans les deux fichiers d’entrée et que les données correspondent parfaitement. Mais si ça n’avait pas été le cas, il aurait été possible de rapprocher les données (padding, changement de casse, etc.) directement à ce niveau.

Pour établir la jointure, déposez la colonne **City** de la première table d’entrée vers la colonne **City** de la table de référence. Un lien violet apparaît pour matérialiser cette jointure.

Maintenant, vous pouvez utiliser la colonne **County** de la table de référence dans la table de sortie (**out1**).
Enfin, cliquez sur le bouton OK pour valider les modifications et exécutez ce nouveau Job.

La sortie suivante s’affichera dans la console :

```
Ulysses Taft | 1794 GRANDVIEW DRIVE | GARDEN GROVE | ORANGE
Theodore Grant | 1895 PACIFIC HWY S | ORANGE | ORANGE
John Johnson | 1554 SAN YSIDRO BLVD | NORCO
Warren Jackson | 897 MONROE STREET | VILLA PARK | ORANGE
Warren Van Buren | 1633 WESTSIDE FREEWAY | PLACENTIA | ORANGE
Rutherford Eisenhower | 418 BURNETT ROAD | CORONA
Zachary Taft | 385 W. RUSSELL ST | VORBA LINDA | ORANGE
Zachary Pierce | 1292 FONTAINE ROAD | VENTURA
George Garfield | 699 VIA REAL | CAMARILLO
Warren Taylor | 630 NORTH ATHERTON STREET | CARPINTERIA
```

Comme vous pouvez le voir, la dernière colonne ne contient que les villes des régions d’Orange et de Los Angeles. Pour les autres villes, cette colonne reste vide. Ceci est dû au fait que par défaut le tMap établit une jointure Left Outer Join. Si vous souhaitez appliquer un filtre permettant de n’afficher que les données pour lesquelles une correspondance a été trouvée par le tMap, ouvrez le tMap cliquez sur le bouton tMap settings et sélectionnez Inner Join dans la liste Join Model sur la table de référence (row2).

**Étape 4 : Sortie vers une table MySQL**

Votre Job fonctionne à merveille. Pour le finaliser, redirigez le flux de sortie vers une table MySQL.

**Procédure**

1. Pour cela, créez tout d’abord une métadonnée décrivant la connexion à la base de données MySQL. Double-cliquez sur DemoMySQL dans le répertoire Metadata > MySQL du Référentiel (à condition que vous ayez bien importé le projet Demo) pour lancer l’assistant Metadata.

Exemple de Job comprenant un tMap

3. Déposez cette métadonnée à droite de l’espace de modélisation en maintenant la touche Ctrl enfoncée pour créer automatiquement un composant tMysqlOutput.

4. Supprimez le composant tLogRow de votre Job.

5. Reconnectez le flux de sortie out1 du tMap vers le composant tMysqlOutput.

6. Dans l’onglet Basic settings de ce composant :
   a) Saisissez LA_Orange_clients dans le champ Table pour nommer votre table cible qui va être créée à la volée.
   b) Sélectionnez l’option Drop table if exists and create dans le champ Action on table.
   c) Cliquez sur Edit Schema et sur le bouton Reset DB type (le bouton en forme de base de données dans la barre d’outils) pour renseigner automatiquement le type de base de données, si nécessaire.

7. Exécutez à nouveau le Job.

Résultats

La table cible devrait être automatiquement créée et remplie en moins d’une seconde.

Dans ce scénario, seuls quatre composants différents sont utilisés, mais la Palette en contient plus de 450 (bases de données, Webservices, FTP, etc.).

D'autres composants, réalisés cette fois par la communauté, sont disponibles sur le site communautaire : talendforge.org.
Utilisation du Dynamic Schema pour charger dynamiquement des données dans les tables de base de données

Cet exemple décrit comment tirer profit de la fonctionnalité Dynamic Schema pour charger des données dynamiquement.

**Avertissement:** Cette documentation est pertinente uniquement si vous avez souscrit à Studio Talend.

Le contenu des trois fichiers plats ci-dessous sera chargé dans des tables de base de données MySQL, respectivement nommées customer, employee et delivery :

- **customer_details.csv** :

```
first_name;last_name;order_date
Reita;Leto;2012-05-15
Larae;Gudroe;2011-08-07
Clay;Hoa;2014-04-16
Irma;Wolfgramm;2012-09-10
Sylvia;Cousey;2013-02-25
```

- **employee_details.csv** :

```
id;name;gender;email;department;order_date
1;Sage;Male;sage_wieser@cox.net;R&D;2012-05-15
2;Kris;Female;kris@gmail.com;Test;2011-08-07
3;Abel;Male;amaclead@gmail.com;HR;2014-04-16
4;Mattie;Female;mattie@aol.com;Test;2012-09-10
5;Bernardo;Female;bfigeroa@aol.com;Marketing;2013-02-25
```

- **delivery_details.csv** :

```
Name;Datetime
Reita;2012-02-11 15:30:32
Larae;2015-08-28 09:30:32
```

Cet exemple implique deux Jobs, l'un appelant l'autre : Le Job parent mappe le nom des fichiers vers le nom des tables dans la base de données, et le Job enfant lit les données depuis les fichiers et écrit les données dans les tables de base de données correspondantes.

**Prérequis pour cet exemple de Job**

Avant de créer, de configurer et d’exécuter les Jobs dans cet exemple, assurez-vous que :

- Les tables cible ont été créées dans la base de données MySQL et la structure des tables et le nom des colonnes sont identiques dans les fichiers sources correspondants.

**Conseil:** Vous pouvez créer rapidement des tables de base de données à l’aide du composant tCreateTable.

- Les fichiers sources sont disponibles dans votre système de fichiers.
Créer le Job parent

Procédure
2. Déposez les composants suivants dans l’espace de modélisation graphique et libellez-les selon leur rôle dans le Job.

<table>
<thead>
<tr>
<th>Composant</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>tFileList</td>
<td>File_list</td>
</tr>
<tr>
<td>tIterateToFlow</td>
<td>File_name_flow</td>
</tr>
<tr>
<td>tMap</td>
<td>File_table_lookup</td>
</tr>
<tr>
<td>tFixedFlowInput</td>
<td>Mappings</td>
</tr>
<tr>
<td>tJavaRow</td>
<td>Set_context_variables</td>
</tr>
<tr>
<td>tRunJob</td>
<td>Call_child</td>
</tr>
</tbody>
</table>

3. Reliez les composants :
   a) tFileList et tIterateToFlow à l’aide d’une connexion Row > Iterate
   b) tIterateToFlow et tMap à l’aide d’une connexion Row > Main
   c) tFixedFlowInput et tMap à l’aide d’une connexion Row > Main, en tant que connexion de type Lookup
   d) tMap et tJavaRow à l’aide d’une connexion Row > Main (nommée out dans cet exemple)
   e) tJavaRow et tRunJob à l’aide d’une connexion Row > Main

Créer le Job enfant

Procédure
2. Déposez les composants suivants dans l’espace de modélisation graphique et libellez-les selon leur rôle dans le Job.

<table>
<thead>
<tr>
<th>Composant</th>
<th>Libellé</th>
</tr>
</thead>
<tbody>
<tr>
<td>tFileInputDelimited</td>
<td>Input_file</td>
</tr>
</tbody>
</table>
Utilisation du Dynamic Schema pour charger dynamiquement des données dans les tables de base de données

<table>
<thead>
<tr>
<th>Composant</th>
<th>Libellé</th>
</tr>
</thead>
<tbody>
<tr>
<td>tDBOutput</td>
<td>Write_to_DB</td>
</tr>
</tbody>
</table>

3. Dans la vue Basic settings du composant tDBOutput, sélectionnez MySQL dans la liste Database et cliquez sur Apply.

4. Reliez le composant tFileInputDelimited au composant tDBOutput à l'aide d’un lien de type Row > Main.

**Configurer le Job parent**

**Procédure**

1. Dans la vue Contexts :
   a) Ajoutez les trois variables suivantes :
      - filename, de type String
      - tablename, de type String
      - directory, de type Directory
   b) Spécifiez le répertoire en cliquant dans le champ Value de la variable directory. Cliquez sur le bouton qui apparaît et naviguez jusqu’au répertoire contenant les fichiers sources.

2. Dans la vue Basic settings du composant tFileList, renseignez le champ Directory avec context.directory.
   **Conseil:** Cliquez dans le champ et appuyez sur Ctrl+Maj pour accéder à une liste de variables disponibles.

3. Double-cliquez sur le composant tIterateToFlow pour ouvrir sa vue Basic settings et configurez le composant comme suit :
a) Ajoutez une colonne nommée filename (de type String) dans l’éditeur de schéma.
b) Renseignez le champ Value de la table Mapping avec la variable globale CURRENT_FILE générée par le composant tFileList, au format ((String)globalMap.get("tFileList_1_CURRENT_FILE")).

4. Double-cliquez sur le composant tFixedFlowInput pour ouvrir sa vue Basic settings et configurez le composant :
a) Ajoutez deux colonnes dans le schéma : file_name (de type String) et table_name (de type String).
b) Sélectionnez l’option Use Inline Table et définissez les mappings fichier à table suivants dans la table :

<table>
<thead>
<tr>
<th>file_name</th>
<th>table_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;customer_details.csv&quot;</td>
<td>&quot;customer&quot;</td>
</tr>
<tr>
<td>&quot;delivery_details.csv&quot;</td>
<td>&quot;delivery&quot;</td>
</tr>
<tr>
<td>&quot;employee_details.csv&quot;</td>
<td>&quot;employee&quot;</td>
</tr>
</tbody>
</table>

5. Double-cliquez sur le composant tMap pour ouvrir son éditeur de mapping, puis :
a) Glissez la colonne filename de la table row1 (qui vient du composant tIterateToFlow) et déposez-la dans la colonne file_name de la table row2 (qui vient du composant tFixedFlowInput), pour joindre les deux tables pour le Lookup du nom du fichier.
b) Cliquez sur le bouton à molette pour afficher les propriétés de jointure et définissez :
   • le Match Model sur Unique match
   • le Join Model sur Inner Join
c) Dans l’éditeur de schéma, ajoutez deux colonnes à la table out : tablename (de type String) et filename (de type String).
d) Glissez la colonne filename de la table row1 et déposez-la dans la colonne filename de la table out.
e) Glissez la colonne table_name de la table row2 et déposez-la dans la colonne filename de la table out.
Utilisation du Dynamic Schema pour charger dynamiquement des données dans les tables de base de données

f) Cliquez sur OK pour valider vos paramètres et l’éditeur de mapping.

6. Double-cliquez sur le composant tJavaRow, puis saisissez le code suivant dans le champ Code.

```java
context.tablename = out.tablename;
context.filename = out.filename;
```

7. Dans la vue Basic settings du composant tRunJob :
   a) Cliquez sur le bouton [...] à côté du champ Job et sélectionnez le Job enfant que vous souhaitez appeler depuis la boîte de dialogue Repository Content, LoadDynamicSchemaChild dans cet exemple.
   b) Cochez l’option Transmit whole context pour transmettre les variables de contexte au Job enfant.

8. Appuyez sur les touches Ctrl+S pour enregistrer le Job.

**Configurer le Job enfant**

**Procédure**

1. Dans la vue Contexts, ajoutez les trois variables suivantes, en laissant leur valeur indéfinie :
   - filename, de type String
   - tablename, de type String
   - directory, de type String

2. Double-cliquez sur le composant tFileInputDelimited (libellé Input_file) pour ouvrir sa vue Basic settings et configurez le composant comme suit :
• Ouvrez l’éditeur de schéma et ajoutez une colonne data, puis définissez son type sur Dynamic.

![Schema of Input_file](image)

• Renseignez le champ File name/Stream avec les variables de contexte définies pour l’accès au fichier : context.directory+context.filename.

**Conseil:** Cliquez dans le champ et appuyez sur Ctrl+Shift pour accéder à une liste de variables disponibles.

• Dans le champ Header, spécifiez le nombre de lignes au début des fichiers à passer.

Dans cet exemple, la première ligne de chaque fichier est la ligne d’en-tête.

• Laissez les autres paramètres tels qu’ils sont.

![Input_file(FileInputDelimited_1)](image)

3. Double-cliquez sur le composant tDBOutput (libellé Write_to_DB) pour ouvrir sa vue Basic settings et configurez le composant comme suit :

• Spécifiez les informations de connexion à la base de données, y compris le nom d’hôte ou l’adresse IP, le numéro de port, le nom de la base de données et les identifiants d’authentification.

• Renseignez le champ Table avec la variable de contexte définie pour le nom de la table, context.tablename dans cet exemple.

• Dans la liste Action on table, sélectionnez Default.

• Dans la liste Action on data, sélectionnez Insert.

• Cliquez sur Sync columns pour vous assurer que le schéma soit le même que le composant d’entrée : une seule colonne nommée data, de type Dynamic.
Utilisation du Dynamic Schema pour charger dynamiquement des données dans les tables de base de données

- Laissez les valeurs par défaut des autres options.

4. Appuyez sur les touches Ctrl+S pour enregistrer le Job.

**Exécuter le Job parent et vérifier le résultat de chargement des données**

**Procédure**

1. Rendez-vous dans le Job parent.
2. Appuyez sur F6 pour exécuter le Job et vérifier vos tables de base de données MySQL.

**Résultats**

Le contenu de chaque fichier plat est chargé dans la table de base de données correspondante.
Utilisation du Dynamic Schema pour charger dynamiquement des données dans les tables de base de données

<table>
<thead>
<tr>
<th>Name</th>
<th>Datetime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reita</td>
<td>2012-02-11 15:30:32</td>
</tr>
<tr>
<td>Larae</td>
<td>2015-08-28 09:30:32</td>
</tr>
<tr>
<td>(NULL)</td>
<td>(NULL)</td>
</tr>
</tbody>
</table>
Utilisation de la fonctionnalité Output Stream

Le scénario suivant a pour objectif de montrer comment utiliser la fonctionnalité de flux de sortie dans un certain nombre de composants, afin d’améliorer considérablement les performances en sortie.

Dans ce scénario, un fichier .csv prédéfini contenant des informations client est chargé dans une table d’une base de données. Les données chargées sont sélectionnées à l’aide d’un composant tMap et écrites dans un fichier de sortie local, ainsi que dans la console, via la fonctionnalité Output stream.


Données d’entrée

Le fichier d’entrée, dont les données seront chargées dans la table de la base de données, contient des informations clients variées.

La structure du fichier appelée Schema dans le Studio Talend comprend les colonnes suivantes :

- id (Type : Integer)
- CustomerName (Type : String)
- CustomerAge (Type : Integer)
- CustomerAddress (Type : String)
- CustomerCity (Type : String)
- RegisterTime (Type : Date)

Données de sortie

Le composant tMap est utilisé pour sélectionner les colonnes id, CustomerName et CustomerAge dans les données d’entrée. Les données sélectionnées sont écrites en sortie via la fonctionnalité de flux de sortie.

Les données attendues en sortie doivent avoir la structure suivante :

- id (Type : Integer)
- CustomerName (Type : String)
- CustomerAge (Type : Integer)

Ces trois colonnes proviennent des colonnes des données d’entrée.

Transformer le scénario en Job

Pour créer ce Job, vous devez effectuer les quatre étapes suivantes :

2. Définition de la commande activant la fonctionnalité de flux de sortie.
3. Mapping des données via le composant tMap.
4. Écriture en sortie du flux de données sélectionné.
Vous pouvez voir le Job terminé dans la capture d'écran ci-dessous. Pour consulter les instructions détaillées relatives à la création du Job, lisez les sections suivantes.

**Étape 1 : Lire les données d’entrée d'un fichier local**

Utilisez le composant `tFileInputDelimited` pour lire le fichier `customers.csv` contenant les données d'entrée. Ce composant se trouve dans la famille `File/Input` de la Palette.

**Procédure**


2. Cliquez sur le bouton [...] à côté du champ File name/Stream et parcourrez votre système jusqu’à votre fichier d'entrée. Vous pouvez également saisir manuellement le chemin d'accès à ce fichier.

3. Cliquez sur Edit schema pour ouvrir une boîte de dialogue dans laquelle configurer la structure du fichier d'entrée.

4. Cliquez six fois sur le bouton [+] pour ajouter six colonnes, puis, définissez le Type et le nom des colonnes comme suit:
5. Cliquez sur **OK** pour fermer la boîte de dialogue.

**Étape 2 : Configurer la commande pour activer la fonctionnalité Output Stream**

Utilisez le tJava pour définir la commande de création d’un fichier de sortie et un répertoire contenant le fichier de sortie.

**Procédure**

1. Déposez un composant tJava dans l’espace de modélisation graphique, double-cliquez dessus pour ouvrir sa vue Basic settings et définir ses propriétés.

```java
new java.io.File("C:/myFolder").mkdirs();
globalMap.put("out_file", new java.io.FileOutputStream("C:/myFolder/customerselection.txt", false));
```

**Remarque:**

La commande saisie ci-dessus créé un nouveau répertoire C:/myFolder pour sauvegarder le fichier de sortie customerselection.txt. Vous pouvez personnaliser la commande selon vos besoins.

2. Dans le champ **Code**, saisissez la commande suivante :

```java
new java.io.File("C:/myFolder").mkdirs();
globalMap.put("out_file", new java.io.FileOutputStream("C:/myFolder/customerselection.txt", false));
```

3. Reliez le tJava au tFileInputDelimited à l’aide d’un lien Trigger > On Subjob Ok.

Cela déclenche le sous-job commençant par le tFileInputDelimited lorsque le tJava est correctement exécuté.
Étape 3 : Mapper les données à l'aide du composant tMap

Procédure

1. Déposez un composant tMap dans l'espace de modélisation graphique, double-cliquez dessus afin d' afficher sa vue Basic settings et configurer ses propriétés simples.

2. Cliquez sur le bouton [...] à côté du Map Editor pour ouvrir une boîte de dialogue dans laquelle configurer le mapping.

3. Cliquez six fois sur le bouton [+] de gauche pour ajouter six colonnes au schéma d’entrée, ces colonnes devant être les suivantes :

4. Cliquez sur le bouton [+] à droite pour ajouter un flux de données de sortie.
5. Sélectionnez **New output** et cliquez sur **OK** pour sauvegarder le schéma de sortie.

Le schéma de sortie est vide.

6. Cliquez sur le bouton [+ ] sous la table **out1** pour ajouter trois colonnes aux données de sortie.

7. Déposez les colonnes **id**, **CustomerName** et **CustomerAge** de la gauche à la droite, dans leurs lignes respectives.

8. Cliquez sur **OK** pour sauvegarder les paramètres.
Étape 4 : Écrire en sortie le flux de données sélectionné

Procédure

1. Déposez un composant `tFileOutputDelimited` dans l’espace de modélisation graphique, double-cliquez dessus afin d’ouvrir sa vue Basic settings et configurez ses paramètres simples.

2. Cochez la case **Use Output Stream** pour activer le champ **Output Stream** et saisissez dans le champ **Output Stream** la commande suivante :

   ```java
   (java.io.OutputStream)globalMap.get("out_file")
   ```

   **Remarque:**
   Vous pouvez personnaliser la commande dans le champ **Output Stream** en appuyant sur les touches **Ctrl+Espace** pour utiliser l’autocomplétion et sélectionner des commandes built-in. Vous pouvez également saisir manuellement votre commande. Dans ce scénario, la commande utilisée dans le champ **Output Stream** appelle la classe `java.io.OutputStream` pour écrire le flux de données filtré dans un fichier local, spécifié dans la zone **Code** du tJava.

3. Reliez le `tFileInputDelimited` au `tMap` à l’aide d’un lien **Row > Main**, puis reliez le `tMap` au `tFileOutputDelimited` à l’aide du lien **Row > out1** défini dans le Map Editor du `tMap`.

4. Cliquez sur **Sync columns** pour récupérer le schéma du composant précédent.

5. Déposez un `tLogRow` dans l’espace de modélisation graphique et double-cliquez sur ce composant pour ouvrir sa vue Basic settings.

6. Sélectionnez l’option **Table** dans la zone **Mode**.

7. Reliez le `tFileOutputDelimited` au `tLogRow` à l’aide d’un lien **Row > Main**.

8. Cliquez sur **Sync columns** pour récupérer le schéma du composant précédent.

   Ce Job est maintenant prêt à être exécuté.

9. Appuyez sur les touches **Ctrl+S** afin de sauvegarder votre Job et appuyez sur **F6** pour l’exécuter.

   Le contenu de données sélectionnées s’affiche dans la console.
Starting job OutputStream at 17:31 19/10/2011.

[statistics] connecting to socket on port 4059
[statistics] connected

<table>
<thead>
<tr>
<th>id</th>
<th>CustomerName</th>
<th>CustomerAge</th>
</tr>
</thead>
<tbody>
<tr>
<td>10001</td>
<td>Warren</td>
<td>67</td>
</tr>
<tr>
<td>10002</td>
<td>Woodrow</td>
<td>68</td>
</tr>
<tr>
<td>10003</td>
<td>Grover</td>
<td>77</td>
</tr>
<tr>
<td>10004</td>
<td>Abraham</td>
<td>74</td>
</tr>
<tr>
<td>10005</td>
<td>Chester</td>
<td>78</td>
</tr>
<tr>
<td>10006</td>
<td>Calvin</td>
<td>63</td>
</tr>
<tr>
<td>10007</td>
<td>Zachary</td>
<td>53</td>
</tr>
<tr>
<td>10008</td>
<td>Chester</td>
<td>36</td>
</tr>
<tr>
<td>10009</td>
<td>Chester</td>
<td>60</td>
</tr>
<tr>
<td>10010</td>
<td>Woodrow</td>
<td>57</td>
</tr>
</tbody>
</table>

[statistics] disconnected
Job OutputStream ended at 17:31 19/10/2011. [exit code=0]

Les données sélectionnées sont également écrites dans le fichier local customerselection.txt.

<table>
<thead>
<tr>
<th>customerselection.txt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: id;CustomerName;CustomerAge</td>
</tr>
<tr>
<td>2: 10001;Warren;67</td>
</tr>
<tr>
<td>3: 10002;Woodrow;68</td>
</tr>
<tr>
<td>4: 10003;Grover;77</td>
</tr>
<tr>
<td>5: 10004;Abraham;74</td>
</tr>
<tr>
<td>6: 10005;Chester;78</td>
</tr>
<tr>
<td>7: 10006;Calvin;63</td>
</tr>
<tr>
<td>8: 10007;Zachary;53</td>
</tr>
<tr>
<td>9: 10008;Chester;36</td>
</tr>
<tr>
<td>10: 10009;Chester;60</td>
</tr>
<tr>
<td>11: 10010;Woodrow;57</td>
</tr>
</tbody>
</table>
Utilisation de la fonctionnalité de chargement implicite des contextes

La configuration d’un Job selon les variables de contexte vous permet d’orchestrer et d’exécuter vos Jobs dans différentes contextes et environnements. Vous pouvez définir les valeurs de vos variables de contexte lors de leur création ou charger dynamiquement vos paramètres de contexte, explicitement ou implicitement, lors de l’exécution des Jobs.

Le scénario ci-dessous décrit comment utiliser la fonctionnalité de chargement implicite des contextes (Implicit Context Load) de votre Studio Talend afin de charger dynamiquement les paramètres de contexte lors de l’exécution du Job.

Le Job de ce scénario comprend deux composants. Il lit les données des employés stockées dans deux bases de données MySQL, l’une en test et l’autre en production. Les paramètres de connexion pour accéder à ces bases de données sont stockés dans une autre base de données MySQL. Lorsqu’il est exécuté, le Job charge dynamiquement ces paramètres de connexion afin de se connecter aux deux bases de données.

**Créer le Job et définir des variables de contexte**

**Avant de commencer**

Créez deux tables nommées `db_testing` et `db_production`, respectivement, dans une base de données MySQL nommée `db_connections`, pour contenir les paramètres de connexion permettant d’accéder aux bases de données susmentionnées, `testing` et `production`. Chaque table doit contenir seulement deux colonnes : `key` et `value`, de type VARCHAR. Voici un exemple du contenu des tables des bases de données :

`db_testing` :

<table>
<thead>
<tr>
<th>clé</th>
<th>valeur</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hôte</td>
<td>localhost</td>
</tr>
<tr>
<td>Port</td>
<td>3306</td>
</tr>
<tr>
<td>username</td>
<td>root</td>
</tr>
<tr>
<td>password</td>
<td>talend</td>
</tr>
<tr>
<td>database</td>
<td>testing</td>
</tr>
</tbody>
</table>

`db_production` :

<table>
<thead>
<tr>
<th>clé</th>
<th>valeur</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hôte</td>
<td>localhost</td>
</tr>
<tr>
<td>Port</td>
<td>3306</td>
</tr>
<tr>
<td>username</td>
<td>root</td>
</tr>
<tr>
<td>password</td>
<td>talend</td>
</tr>
</tbody>
</table>
Vous pouvez créer ces tables de bases de données à l'aide d'un autre Job Talend contenant un tFixedFlowInput et un tMysqlOutput.

**Procédure**

1. Créez un Job et ajoutez un composant tMysqlInput et un tLogRow dans l'espace de modélisation graphique et reliez-les à l'aide d'un lien Row > Main.

2. Sélectionnez la vue Contexts du Job et cliquez cinq fois sur le bouton [+] au bas de la vue, pour ajouter cinq lignes à la table et définir les variables de contexte suivantes, toutes de type String. Ne configurez pas les valeurs, car elles seront chargées dynamiquement lors de l'exécution du Job : host, port, username, password, et database.

3. Créez une autre variable nommée db_connection de type List Of Value.

4. Cliquez dans le champ Value de la nouvelle variable créée et cliquez sur le bouton qui s'affiche dans la boîte de dialogue Configure Values. Cliquez sur New... pour ouvrir la boîte de dialogue New Value. Saisissez le nom d'une des tables des bases de données contenant les informations de connexion à la base de données puis cliquez sur OK.

<table>
<thead>
<tr>
<th>clé</th>
<th>valeur</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>production</td>
</tr>
</tbody>
</table>
5. Cliquez sur à nouveau sur **New...** pour définir l’autre table contenant les informations de connexion à la base de données. Cela fait, cliquez sur **OK** afin de fermer la boîte de dialogue **Configure Values**.

La variable `db_connection` contient une liste de valeurs `db_testing` et `db_production`, les tables de base de données desquelles charger les paramètres de connexion.

6. Cochez la case **Prompt** à côté du champ **Value** de la variable `db_connection` pour afficher les champs **Prompt** et saisissez le message à afficher lors de l’exécution.
Configurer les composants

Procédure

1. Double-cliquez sur le tMysqlInput pour ouvrir sa vue Basic settings.
2. Renseignez les champs Host, Port, Database, Username, Password et Table Name avec les variables correspondantes définies dans l'onglet Contexts : context.host, context.port, context.database, context.username et context.password dans cet exemple.
3. Renseignez le champ Table Name en saisissant employees, le nom de la table contenant les informations des employés, dans les deux bases de données, dans cet exemple.
4. Renseignez le Schema. Si vous avez stocké le schéma dans le Repository, vous pouvez le récupérer en sélectionnant Repository ainsi que l’entrée correspondante dans la liste.

Dans cet exemple, le schéma des deux tables des bases de données à lire se compose de six colonnes : id (INT, de longueur 2), name (VARCHAR, de longueur 20), email (VARCHAR, de longueur 25), sex (VARCHAR, de longueur 1), department (VARCHAR, de longueur 10) et position (VARCHAR, de longueur 10).
5. Cliquez sur **Guess Query** afin de récupérer toutes les colonnes de la table, qui seront affichées dans l’onglet **Run**, via le composant **tLogRow**.

6. Dans la vue **Basic settings** du composant **tLogRow**, sélectionnez l’option **Table** pour afficher les enregistrements sous forme de tableau.

### Configurer la fonctionnalité de chargement implicite des contextes

Vous pouvez configurer la fonctionnalité de chargement implicite des contextes, soit dans les paramètres du projet, pour pouvoir l’utiliser dans tous les Jobs du projet, soit dans l’onglet Job pour un Job spécifique.

L’exemple suivant explique comment configurer cette fonctionnalité dans la vue **Job** pour un Job en particulier. Si vous souhaitez configurer la fonctionnalité pour la réutiliser dans différents Jobs, sélectionnez **File > Edit Project properties** dans le menu pour ouvrir la boîte de dialogue **Project Settings**. Dans **Job Settings > Implicit context load**, cochez la case **Implicit tContextLoad** et configurez les paramètres en suivant les étapes 2 à 6, ci-dessous. Dans la vue **Job**, cochez la case **Use Project Settings** pour appliquer les paramètres au Job.

### Procédure

1. Dans la vue **Job**, cliquez sur l’onglet **Extra** et cochez la case **Implicit tContextLoad** pour activer le chargement explicite des contextes à l’aide du composant **tContextLoad** dans le Job.

2. Sélectionnez la source de laquelle charger les paramètres de contexte. Une source de contextes peut être un fichier plat à deux colonnes ou une table de base de données contenant deux colonnes. Dans ce scénario, les informations de connexion à la base de données sont stockées dans des tables de bases de données. Sélectionnez donc l’option **From Database**.

3. Configurez les détails de la connexion à la base de données comme vous configurez les paramètres simples d’un composant d’entrée de base de données.
Dans cet exemple, tous les paramètres de connexion sont utilisés pour ce Job en particulier, sélectionnez **Built-In** dans la liste **Property Type** et renseignez manuellement les informations de connexion.

4. Renseignez le champ **Table Name** avec la variable de contexte nommée `db_connection` définie dans la vue **Contexts** du Job, afin de pouvoir choisir la table de base de données de laquelle charger dynamiquement les paramètres de contexte lors de l’exécution du Job.

5. Vous allez récupérer sans condition tous les détails de la connexion depuis les tables de la base de données, laissez donc le champ **Query Condition** vide.

6. Cochez la case **Print operations** afin de lister les paramètres de contexte chargés lors de l’exécution du Job.

**Exécuter le Job**

**Procédure**

1. Appuyez sur les touches **Ctrl+S** afin de sauvegarder le Job et appuyez sur **F6** pour exécuter le Job.

2. Une boîte de dialogue s’ouvre et vous demande de sélectionner une base de données. Sélectionnez une base de données et cliquez sur **OK**.

   Les paramètres de contexte chargés ainsi que le contenu de la table “employees” de la base de données sélectionnée s’affichent dans la console **Run**.

3. Appuyez sur la touche **F6** pour exécuter le Job à nouveau et sélectionnez l’autre base de données lorsque cela vous est proposé.

   Les paramètres de contexte chargés, ainsi que le contenu de la table “employees” de l’autre base de données, sont affichés dans la console de la vue **Run**.
Utilisation de la fonctionnalité Exécution en multi thread pour exécuter des Jobs en parallèle

Basé sur le cas d'utilisation précédent Utilisation de la fonctionnalité Output Stream à la page 23, ce scénario donne un exemple d’utilisation de la fonctionnalité d’exécution en multi thread pour exécuter deux Jobs en parallèle afin d’afficher des informations des employés dans l’environnement de test en même temps que dans l’environnement de production. Lorsqu’il faut gérer de grandes quantités de données, cette fonctionnalité peut considérablement optimiser les performances d’exécution du Studio Talend.


Préparer les Jobs pour lire des données des employés dans différents contextes

Procédure

1. Dans la vue Repository, cliquez-droit sur le Job créé dans le cas d’utilisation Utilisation de la fonctionnalité Output Stream à la page 23 et sélectionnez Duplicate dans le menu contextuel. Puis, dans la boîte de dialogue Duplicate, saisissez un nouveau nom pour le Job, employees_testing dans cet exemple, puis cliquez sur OK.

2. Ouvrez le nouveau Job et renommez les composants pour mieux identifier le rôle de chacun.

3. Créez un autre Job nommé employees_production en répétant les étapes ci-dessus.
4. Dans la vue **Contexts** des deux Jobs, supprimez la variable **db_connection**.

5. Dans l’onglet **Extra** de la vue **Job** du Job **employees_testing**, renseignez le champ **Table Name** en saisissant **db_testing**. Dans l’onglet **Extra** de la vue **Job** du Job **employees_production**, renseignez le champ **Table Name** en saisissant **db_production**.

---

**Mettre en place un Job parent pour exécuter les Jobs en parallèle**

**Procédure**

1. Créez un nouveau Job, ajoutez deux composants **tRunJob** dans l’espace de modélisation graphique et renommez ces composants pour mieux identifier le rôle de chacun.
2. Dans la vue Component du premier composant tRunJob, cliquez sur le bouton [...] à côté du champ Job et spécifiez le Job à exécuter, employees_testing dans cet exemple.

3. Configurez l’autre composant tRunJob pour exécuter l’autre Job, employees_production.


**Exécuter les Jobs**

**Procédure**

1. Sauvegardez chaque Job en appuyant sur les touches Ctrl+S.
2. Dans le Job parent, appuyez sur F6 ou cliquez sur Run dans la vue Run pour démarrer l’exécution des Jobs enfants.

   Les Jobs enfants sont exécutés en parallèle, lisent les données des employés à partir des deux bases de données et affichent ces données dans la console.
Utilisation de la fonctionnalité Exécution en multi thread pour exécuter des Jobs en parallèle

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>email</th>
<th>sex</th>
<th>department</th>
<th>position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Elisa</td>
<td><a href="mailto:elisa@company.com">elisa@company.com</a></td>
<td>F</td>
<td>R&amp;D</td>
<td>Manager</td>
</tr>
<tr>
<td>2</td>
<td>Nicolas</td>
<td><a href="mailto:nicolas@company.com">nicolas@company.com</a></td>
<td>M</td>
<td>R&amp;D</td>
<td>Developer</td>
</tr>
<tr>
<td>3</td>
<td>Cedric</td>
<td><a href="mailto:cedric@company.com">cedric@company.com</a></td>
<td>M</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>4</td>
<td>Rabbit</td>
<td><a href="mailto:rabbit@company.com">rabbit@company.com</a></td>
<td>M</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>5</td>
<td>Mike</td>
<td><a href="mailto:mike@company.com">mike@company.com</a></td>
<td>M</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>6</td>
<td>Sabrina</td>
<td><a href="mailto:sabrina@company.com">sabrina@company.com</a></td>
<td>F</td>
<td>Community</td>
<td>Developer</td>
</tr>
<tr>
<td>7</td>
<td>Stephanie</td>
<td><a href="mailto:stephane@company.com">stephane@company.com</a></td>
<td>M</td>
<td>Sales</td>
<td>Manager</td>
</tr>
<tr>
<td>8</td>
<td>Jin</td>
<td><a href="mailto:jin@company.com">jin@company.com</a></td>
<td>M</td>
<td>Sales</td>
<td>Pre-sales</td>
</tr>
<tr>
<td>9</td>
<td>John</td>
<td><a href="mailto:john@company.com">john@company.com</a></td>
<td>M</td>
<td>null</td>
<td>null</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>email</th>
<th>sex</th>
<th>department</th>
<th>position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Herbert Pierce</td>
<td><a href="mailto:hp@talend.com">hp@talend.com</a></td>
<td>M</td>
<td>Finance</td>
<td>Manager</td>
</tr>
<tr>
<td>2</td>
<td>John Hoover</td>
<td><a href="mailto:jh@talend.com">jh@talend.com</a></td>
<td>M</td>
<td>Finance</td>
<td>Manager</td>
</tr>
<tr>
<td>3</td>
<td>Benjamin Harrison</td>
<td><a href="mailto:bh@talend.com">bh@talend.com</a></td>
<td>M</td>
<td>HR</td>
<td>Manager</td>
</tr>
<tr>
<td>4</td>
<td>George Harrison</td>
<td><a href="mailto:gh@talend.com">gh@talend.com</a></td>
<td>M</td>
<td>Sales</td>
<td>Manager</td>
</tr>
<tr>
<td>5</td>
<td>Helen Monroe</td>
<td><a href="mailto:hn@talend.com">hn@talend.com</a></td>
<td>F</td>
<td>R&amp;D</td>
<td>Developer</td>
</tr>
<tr>
<td>6</td>
<td>Anne Harrison</td>
<td><a href="mailto:ah@talend.com">ah@talend.com</a></td>
<td>F</td>
<td>Sales</td>
<td>Pre-sales</td>
</tr>
<tr>
<td>7</td>
<td>Thomas Nixon</td>
<td><a href="mailto:tn@talend.com">tn@talend.com</a></td>
<td>M</td>
<td>R&amp;D</td>
<td>Developer</td>
</tr>
<tr>
<td>8</td>
<td>James Lincoln</td>
<td><a href="mailto:ji@talend.com">ji@talend.com</a></td>
<td>M</td>
<td>R&amp;D</td>
<td>Developer</td>
</tr>
<tr>
<td>9</td>
<td>Rutherford Fillmore</td>
<td><a href="mailto:rf@talend.com">rf@talend.com</a></td>
<td>M</td>
<td>Finance</td>
<td>Accountant</td>
</tr>
<tr>
<td>10</td>
<td>Maria Pierce</td>
<td><a href="mailto:mp@talend.com">mp@talend.com</a></td>
<td>F</td>
<td>Finance</td>
<td>Accountant</td>
</tr>
</tbody>
</table>
Appel du service Web REST de la MetaServlet pour exécuter une tâche

Dans le scénario suivant, un Job créé dans le Studio invoque le service Web REST de la MetaServlet pour exécuter une tâche spécifique dans Talend Administration Center.

**Avertissement:** Cette documentation est pertinente uniquement si vous avez souscrit à Studio Talend.

**Prérequis :**
Assurez-vous qu’une tâche existante est disponible dans la page *Job Conductor* de Talend Administration Center.

**Déposer et relier les composants**

**Procédure**
1. Déposez les composants suivants de la *Palette* dans l’espace de modélisation graphique : un composant *tSetGlobalVar*, un composant *tREST* et un composant *tLogRow*.
2. Reliez le composant *tREST* au composant *tLogRow* à l’aide d’un lien de type *Row > Main*.
3. Reliez le composant *tSetGlobalVar* au composant *tREST* à l’aide d’un lien de type *Trigger > OnSubjobOK*.

**Configurer les variables de contexte**

Vous devez définir les variables qui seront utilisées pour appeler le service Web REST de la MetaServlet.

**Procédure**
1. Dans la vue *Contexts*, cliquez sur le bouton [*+*] à quatre reprises pour ajouter quatre lignes.
Appel du service Web REST de la MetaServlet pour exécuter une tâche

2. Nommez ces variables `tac_url`, `tac_user`, `tac_pwd` et `task_id`.

3. Dans le champ **Value**, sous le contexte **Default**, saisissez les valeurs des variables :
   - Pour la variable `tac_user`, saisissez le nom utilisateur administrateur de l'application Web Talend Administration Center, `admin@company.com` par exemple.
   - Pour la variable `tac_pwd`, saisissez le mot de passe administrateur de l'application Web Talend Administration Center, `admin` par exemple.
   - Pour la variable `task_id`, saisissez l'identifiant de la tâche que vous souhaitez générer, `1` par exemple.

4. Appuyez sur les touches **Ctrl+S** afin de sauvegarder vos modifications.

**Configurer la Routine**

Vous devez définir une routine personnalisée qui encode les paramètres de la MetaServlet en base64 afin de les transmettre à l'API REST. Cette routine sera appelée dans le Job.

**Procédure**

1. Dans la vue **Repository**, cliquez sur **Code** pour accéder aux **Routines**.
2. Puis cliquez-droit sur **Routines** et sélectionnez **Create Routine**.
3. La boîte de dialogue **New routine** s'ouvre. Saisissez les informations requises pour créer la routine, puis cliquez sur **Finish** pour aller à l'étape suivante.

   La nouvelle routine apparaît dans l'arborescence du **Repository**, directement sous le nœud **Routines**. L'espace de travail s'ouvre sur un modèle de routine contenant par défaut un exemple simple, composé d'un commentaire en bleu, suivi du code correspondant.

4. Au début, juste après la ligne de code `package routines` , ajoutez ce qui suit :
   ```java
   import com.sun.org.apache.xml.internal.security.utils.Base64;
   ```

   Pour ce faire, commencez à saisir ce qui suit et appuyez sur **Ctrl+Espace** pour ouvrir la liste des modèles, puis sélectionnez `com.sun.org.apache.xml.internal.security.utils.*`; et remplacez le signe `*` par `Base64`.

5. Modifiez la fin du modèle avec votre propre code :
   ```java
   public static String base64Encode(String message) {
       return message==null ? null : "" + Base64.encode(message.getBytes()).replace("\n", "");
   }

   public static void main(String[] args) {
       String tmp = "{""actionName":"runTask","taskId":"1","mode":"synchronous","authPass":"talend","authUser":"talend@talend.com""};
       System.out.println("Base 64: " + base64Encode(tmp));
   }
   ```
   
   Ceci vous permet d'encoder l'action `runTask` de la MetaServlet en base64.

   Pour plus d'informations concernant les paramètres et les actions disponibles dans la MetaServlet, consultez le Guide utilisateur de Talend Administration Center.
Configurer les composants

Procédure

1. Double-cliquez sur le composant **tSetGlobalVar** pour afficher sa vue **Basic settings** dans l’onglet **Component**.
2. Cliquez sur le bouton [+ ] pour ajouter une ligne dans le tableau **Variables** :
   - Dans le champ **Key**, saisissez **jsonEncoded**
   - Dans les champs **Value**, saisissez :

```java
MetaServlet.base64Encode("{"actionName":"runTask","taskId":"","mode":"synchronous","context":{"Default":"" + ((String)globalMap.get("tMsg Box_1_RESULT")) + ",","authPass":"" + context.tac_pwd + ","authUser":"" + context.tac_user + ""}
```

pour appeler la routine précédemment créée.

3. Double-cliquez sur le composant **tREST** pour afficher sa vue **Basic settings**.
4. Renseignez le champ **URL** avec l’URL du service Web que vous souhaitez invoquer. Pour ce scénario, saisissez :

```java
context.tac_url + "/metaServlet?" +((String)globalMap.get("jsonEncoded"))
```

pour appeler le service et encoder les paramètres de la MetaServlet au format JSON.
5. Dans la liste **HTTP Method**, sélectionnez **GET** pour envoyer la requête HTTP pour générer une tâche.
   Ainsi, la MetaServlet est invoquée via l’API REST de Talend Administration Center avec les paramètres requis.
6. Dans la vue **Basic settings** du composant **tLogRow**, sélectionnez l’option **Basic** pour afficher le résultat dans la console **Run**.
7. Enregistrez votre Job et appuyez sur **F6** pour l’exécuter.

Résultats

La console indique que le composant **tREST** a envoyé une requête HTTP côté serveur pour exécuter la tâche spécifiée et que cette tâche a été exécutée sans erreur.
Dans la page **Job Conductor** de Talend Administration Center, le statut de la tâche est désormais **ready to run**.

![Image of Job Conductor page](image-url)
Exemple de Job comprenant des règles de validation

Avertissement: Cette documentation est pertinente uniquement si vous avez souscrit à Studio Talend.

Si vous devez filtrer vos données afin d’envoyer uniquement des données valides dans vos flux de sortie, vous pouvez créer des règles de validation qui vérifieront vos données. Ces règles de validation peuvent effectuer une vérification simple de la validité des valeurs ou effectuer une vérification d’intégrité par référence. Elles sont stockées au niveau des métadonnées dans la vue Repository et peuvent donc être facilement réutilisées et modifiées.


Utiliser une règle de validation par référence

Après avoir configuré une règle de validation, vous pouvez l’appliquer dans votre Job. Par exemple, appliquez une règle de validation par référence lors de la mise à jour de la base de données avec de nouvelles données. Cependant, avant de charger ces nouvelles données, soyez sûr(e) qu’elles correspondent à celles qui existent déjà dans la base de données.

Ce scénario est basé sur la règle de validation par référence créée dans la section Règle de validation par référence du Guide utilisateur du Studio Talend.

Configurer l’environnement pour les Jobs

Procédure

1. Déposez les composants suivants de la Palette, dans l’espace de modélisation graphique : un composant d’entrée de base de données (Input) et un composant de sortie de base de données (Output), dans cet exemple un tMysqlInput et un tMysqlOutput, afin de charger les données, un composant tLogRow pour afficher les données rejetées dans la console et un composant tJava pour afficher le nombre de lignes traitées dans la console.

2. Reliez les composants d’entrée et de sortie de base de données à l’aide d’un lien Row > Main. Reliez le composant tMysqlInput au tJava à l’aide d’un lien Trigger > OnSubjobOk.

Remarque:

Vous pouvez créer le lien de rejet entre le tMysqlOutput et le tLogRow uniquement si vous avez appliqué la règle de validation au composant tMysqlOutput.
**Configurer les composants**

**Procédure**

1. Double-cliquez sur le composant `tMysqlInput` afin d’afficher sa vue *Basic settings*.

2. Sélectionnez *Repository* dans la liste *Property type* puis cliquez sur le bouton `...` à côté du champ afin de récupérer les propriétés de connexion correspondant à la métadonnée que vous souhaitez vérifier.

3. Sélectionnez *Repository* dans la liste déroulante *Schema* et cliquez sur le bouton `...` à côté du champ pour récupérer le schéma correspondant à votre table de base de données.

4. Cliquez sur le bouton `...` à côté du champ *Table* afin de sélectionner la table à vérifier.
5. Cliquez sur le bouton **Guess Query** pour récupérer automatiquement la requête correspondant au schéma de la table.

6. Double-cliquez sur le composant **tMysqlOutput** pour afficher ses paramètres (**Basic settings**).

7. Sélectionnez **Repository** comme **Property type** et cliquez sur le bouton [...] à côté du champ afin de récupérer les propriétés de connexion correspondant à la table de base de données dans laquelle vous souhaitez charger les nouvelles données.

8. Cliquez sur le bouton [...] à côté du champ **Table** pour sélectionner la table dans laquelle vous souhaitez charger les données.

9. Dans la liste **Action on table**, sélectionnez **Default** et dans la liste **Action on data**, sélectionnez l’action correspondant à celle(s) définie(s) dans la règle de validation appliquée(s) au Job. Ici, comme vous avez sélectionné **On insert** et **On update** dans la règle de validation par référence utilisée, sélectionnez **Update or insert** pour déclencher la règle.

10. Si le schéma des composants d’entrée et de sortie ne se synchronisent pas automatiquement, cliquez sur **Sync columns** et le schéma du flux d’entrée sera automatiquement récupéré.

**Appliquer la règle de validation et visualiser les résultats de l’exécution du Job**

**Procédure**

1. Cliquez sur l’onglet **Validation Rules** dans la vue **Component** du composant **tMysqlOutput**.

2. Cochez la case **Use an existing validation rule** pour appliquer la règle de validation au composant.

3. Dans la liste **Validation Rule Type**, sélectionnez **Repository** et cliquez sur le bouton [...] pour sélectionner la règle de validation dans la boîte de dialogue [Repository Content].

4. Cliquez-droit sur le **tMysqlOutput**, sélectionnez **Row > Rejects** dans le menu et glissez jusqu’au **tLogRow** pour créer un lien de rejet entre les deux composants.
Exemple de Job comprenant des règles de validation

Si vous avez activé l’option **Reject link** pour cette règle de validation, vous pouvez récupérer les données rejetées dans un flux de rejet.

5. Double-cliquez sur le composant **tJava** afin d’afficher ses paramètres (**Basic settings**).

6. Dans le champ **Code**, saisissez le code qui affichera le nombre de lignes traitées mises à jour, insérées et rejetées :

   ```java
   System.out.println("Updates: "+(Integer)globalMap.get("tMysqlOutput_1_NB_LINE_UPDATED")+"\nInserts: "+(Integer)globalMap.get("tMysqlOutput_1_NB_LINE_INSERTED")+"\nRejects: "+(Integer)globalMap.get("tLogRow_1_NB_LINE").
   ```

7. Sauvegardez votre Job et appuyez sur **F6** pour l’exécuter.

   ```text
   Starting job validation_rules_by_reference at 18:15
   12/04/2011.
   
   [statistics] connecting to socket on port 3651
   [statistics] connected
   \l\n   Andrew|Cleveland|South Roosevelt Drive|20/01/2005|55239|XX\n   |Ford|Cleveland Ave.|27/11/2001|78936|XX\n   Updates: 1
   Inserts: 1
   Rejects: 2
   [statistics] disconnected
   ```

**Résultats**

Les données valides sont insérées ou mises à jour dans la table de la base de données et la console affiche les lignes rejetées par la règle de validation, ainsi que le nombre de mises à jour, d’insertions et de rejets traités dans le Job.

**Utiliser une règle de validation simple**

Après avoir configuré une règle de validation, vous pouvez l’appliquer dans votre Job. Par exemple, appliquez une règle de validation des valeurs lors de la lecture d’une table de base de données.

Ce scénario se base sur la règle de validation simple créée dans la section **Règle simple** du Guide utilisateur du Studio Talend.

**Configurer l’environnement pour les Jobs**
**Exemple de Job comprenant des règles de validation**

**Procédure**

1. Déposez les composants suivants de la Palette, dans l'espace de modélisation graphique : un composant d'entrée de base de données, ici un `tMysqlInput`, duquel vous allez lire et vérifier les valeurs, deux `tFileOutputDelimited` pour extraire les données valides dans un fichier et les données rejetées dans un autre fichier et un `tJava` afin d’afficher le nombre de lignes traitées dans la console.

2. Reliez le composant d’entrée de base de données au premier `tFileOutputDelimited` à l’aide d’un lien `Row > Main`. Reliez le `tMysqlInput` au `tJava` à l’aide d’un lien `Trigger > OnSubjobOk`.

<table>
<thead>
<tr>
<th>Remarque:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vous pouvez créer un lien <code>Reject</code> entre le composant <code>tMysqlInput</code> et le second <code>tFileOutputDelimited</code> uniquement si vous avez appliqué la règle de validation au composant <code>tMysqlInput</code>.</td>
</tr>
</tbody>
</table>

![Diagramme du Job basic_phone_validation](image)

**Configurer le tMysqlInput**

**Procédure**

1. Double-cliquez sur le composant `tMysqlInput` afin d’afficher sa vue `Basic settings`. 

---

48
2. Sélectionnez Repository dans la liste Property type puis cliquez sur le bouton [...] à côté du champ afin de récupérer les propriétés de connexion correspondant à la métadonnée que vous souhaitez vérifier.

3. Sélectionnez Repository dans la liste déroulante Schema et cliquez sur le bouton [...] à côté du champ pour récupérer le schéma correspondant à votre table de base de données.

4. Cliquez sur le bouton [...] à côté du champ Table afin de sélectionner la table à vérifier.

5. Cliquez sur le bouton Guess Query pour récupérer automatiquement la requête correspondant au schéma de la table.

**Appliquer la règle de validation et créer un lien de rejet**

Comme l’option On select est sélectionnée pour cette règle de validation, la règle de validation doit être appliquée au composant d’entrée.

**Procédure**

2. Cochez la case **Use an existing validation rule** pour appliquer la règle de validation au composant.
3. Dans la liste **Validation Rule Type**, sélectionnez **Repository** puis cliquez sur le bouton [...] pour sélectionner la règle de validation dans la boîte de dialogue **[Repository Content]**.
4. Cliquez-droit sur le composant **tMysqlInput**, sélectionnez **Row > Reject** dans le menu et déposez le second composant **tFileOutputDelimited** afin de créer un lien de rejet entre les deux composants.

Si vous activez l’option **Reject link** pour cette règle de validation, vous pouvez récupérer les données rejetées dans un flux de rejet.

**Configurer les composants de sortie et visualiser le résultat de l’exécution du Job**

**Procédure**

1. Double-cliquez sur le premier composant **tFileOutputDelimited** afin d’afficher sa vue **Basic settings**.

2. Dans le champ **File Name**, spécifiez le chemin d’accès au fichier dans lequel écrire les données valides, ainsi que son nom.
3. Dans les champs Row Separator et Field Separator, saisissez respectivement le séparateur de lignes et le séparateur de champs.
4. Cochez la case **Include Header** afin d’inclure les en-têtes des colonnes dans les données de sortie.
5. Répétez les étapes ci-dessus pour le second composant **tFileOutputDelimited** afin de configurer la sortie des données rejetées.
6. Double-cliquez sur le composant **tJava** afin d’afficher ses paramètres (**Basic settings**).

7. Dans le champ **Code**, saisissez le code qui affichera le nombre de lignes traitées mises à jour, insérées et rejetées :

   ```java
   System.out.println("Valid data: "+((Integer)globalMap.get("tFileOutputDelimited_1_NB_LINE")));"nRejected data: "+((Integer)globalMap.get("tFileOutputDelimited_2_NB_LINE")));;
   ```

8. Sauvegardez votre Job et appuyez sur **F6** pour l’exécuter.

   ```
   Starting job Basic_phone_validation at 16:00 26/04/2011.
   [statistics] connecting to socket cn port 3510 [statistics] connected
   Valid data: 518
   Rejected data: 402 [statistics] disconnected
   Job Basic_phone_validation ended at 16:00 26/04/2011. [exit code=0]
   ```

**Résultats**

Les données valides sont écrites en sortie dans le premier fichier délimité et les rejets dans le second. La console affiche le nombre de lignes valides et le nombre de rejets traités dans le Job.
Utiliser le connecteur JDBC pour se connecter à Amazon Athena

Cet exemple explique comment utiliser le pilote JDBC Java pour se connecter au service Amazon Athena et tirer parti du service sur le Studio Talend.

Dans cet article, Talend 7.1.1 est utilisé pour l’implémentation. Cependant, n’importe quelle version de Talend qui supporte JDBC peut être utilisée pour se connecter à Amazon Athena. Amazon Athena n’est actuellement disponible que dans certaines régions AWS. Il est recommandé de vérifier la disponibilité du service dans le tableau des régions d’AWS.

Les résultats de la requête d’Amazon Athena doivent être sauvegardés sur Amazon S3. Un dossier de sortie S3 a été créé à cet effet.

Pour plus d’informations sur Amazon Athena, Amazon S3 et JDBC, consultez :
• Amazon Athena
• Amazon S3
• Java JDBC

Configurer une connexion JDBC pour se connecter à Amazon Athena

Avant de commencer
La bibliothèque additionnelle requise AthenaJDBC42-2.0.7.jar est prête. Si non, téléchargez le pilote Athena JDBC ici.

Pourquoi et quand exécuter cette tâche
Suivez la procédure ci-dessous pour configurer une connexion JDBC afin de connecter le Job Studio Talend à Amazon Athena.

Procédure
1. Dans la vue Repository de votre Studio Talend, développez le nœud Metadata, cliquez-droit sur DB Connections et sélectionnez Create connection.
   L’assistant Database Connection s’ouvre.
2. Saisissez le nom de la connexion et sa finalité puis cliquez sur Next.
3. Sélectionnez **JDBC** dans la liste **DB type**.
4. Dans le champ **JDBC URL**, saisissez les détails de l'URL JDBC, incluant vos identifiants AWS.

```
jdbc:awsathena://AwsRegion=<AWS Region>;User=<AWS Access Key>;Password=<AWS Secret Key>;S3OutputLocation=<S3 folder>
```

5. Installez la bibliothèque additionnelle requise.

a) Dans la table **Drivers**, cliquez sur le bouton [*] pour ajouter une ligne et cliquez sur le bouton [...] pour ouvrir la boîte de dialogue **Module**.

b) Sélectionnez l’option **Install a new module**, cliquez sur le bouton [...] pour parcourir les fichier .jar téléchargé.

c) Cliquez sur le bouton **Detect the module install status**, puis cliquez sur **OK** pour fermer la boîte de dialogue.
6. De retour sur l’assistant **Database connection**, cliquez sur le bouton **Select class name** et sélectionnez la classe du pilote principale qui autorise la communication avec Amazon Athena.

7. Dans les champs **User Id** et **Password**, renseignez votre identifiant et mot de passe d’authentification à la base de données.

8. Cliquez sur **Test connection** puis sur **Finish** pour fermer l’assistant.


**Configurer un Job pour se connecter à Amazon Athena**

**Procédure**

1. Créez un Job Talend standard et déposez deux composants dans son espace de modélisation graphique : un composant **tDBInput** et un **tLogRow**, puis nommez-les selon vos besoins.

2. Double-cliquez sur le composant **tDBInput**, sélectionnez **JDBC** à partir de la liste **Database** puis cliquez sur **Apply**.

3. Reliez le composant **tDBInput** au composant **tLogRow** à l’aide d’un lien de type **Row > Main**.


4. Dans l’arborescence de **Repository**, développez la connexion JDBC fraîchement créée, sélectionnez le schéma de la table qui vous intéresse puis déposez-le sur le composant **tDBInput**. Les informations de connexion, le nom de la table et le schéma sont configurés automatiquement.

5. Cliquez sur **Guess Query** pour générer la clause de requête puis la modifier selon vos besoins.

6. Dans la vue **Basic settings** du composant **tLogRow**, cliquez sur **Sync columns** pour vous assurer que son schéma est identique aux composants en entrée, puis sélectionnez l’option **Table (each is key/value list)** pour une meilleure lisibilité de l’affichage de la console.

7. Appuyez sur **F6** pour exécuter le Job.

**Résultats**
La console **Run** affiche les données lues à partir du service Amazon Athena.
Ordonnancement et exécution d’un Job Talend sur le moteur Google Kubernetes


Cet exemple vous présente comment :

• publier un Job Talend en tant qu’image Docker dans un Docker Registry,
• ordonnancer l’image Docker pour exécuter le Job dans un cluster Google Kubernetes Engine (GKE).

Prérequis à cet exemple

Pour publier un Job en tant qu’image Docker et l’exécuter sur Google Kubernetes Engine, vous devez avoir :

• Un Studio Platform Talend ou un Studio Talend Data Fabric, en version 7.1.1 ou supérieure, installé sur une machine cliente/locale,
• un compte Docker registry,
• un compte sous licence Google Cloud Platform (GCP).

Publier votre Job en tant qu’image Docker

Docker est un outil conçu pour simplifier la création, le déploiement et l’exécution d’applications grâce à l’utilisation de conteneurs. Suivez les étapes suivantes afin de publier un Job en tant qu’image Docker.

Procédure

1. Ouvrez le Studio Talend et créez un Job Standard simple comme ci-dessous et assurez-vous qu’il s’exécute correctement.

3. Dans la liste **Export Type**, sélectionnez **Docker Image** et cliquez sur **Next**.
4. Sélectionnez **Local** en tant qu’hôte Docker et fournissez les détails suivants avec vos identifiants Docker registry.

- **Image name** : nom pour l’image Docker. Un référentiel sera créé, avec le même nom dans le registre Docker.
- **Image tag** : valeur du tag à mettre sur votre imager Docker. Par défaut, la version du Job sera la valeur du tag.
- **Registry** : fournissez l’URL du registre Docker avec votre ID Docker. Pour Docker Hub, elle sera `hub.docker.com/dockeroId`.
- **Username et Password** : informations d’authentification utilisées pour vous connecter au registre Docker.
Configurer Google Kubernetes Engine

Google Kubernetes Engine (GKE) est un système de gestion et d’orchestration pour le conteneur Docker et les clusters de conteneurs s’exécutant au sein des services publics Cloud Google.

Procédure

1. Connectez-vous à votre environnement GCP.
2. Sélectionnez le moteur Kubernetes Engine dans la liste Google Services.
3. Cliquez sur le bouton Create Cluster et fournissez la configuration requise pour votre Kubernetes Engine.

Pour plus d’informations concernant les détails de configurations, consultez https://cloud.google.com/kubernetes-engine/docs/how-to/creating-a-cluster.

Une fois le cluster créé, la page ressemble à ceci :

4. Cliquez sur le bouton Connect à droite afin de vous connecter à ce cluster.
   Une fenêtre comme celle-ci s’ouvre.
5. Cliquez sur Run in Cloud Shell.

Google Cloud Shell s’ouvre avec une commande. Appuyez sur Entrée pour exécuter la commande.

Vous êtes connecté(e) au cluster.

**Ordonnancer le Job**

**Procédure**

2. Créez le Secret avec le nom `regcred` en exécutant la commande suivante.

```bash
kubectl create secret docker-registry regcred --docker-server=<docker-registry-server> --docker-username=<dockerId> --docker-password=<docker-password> --docker-email=<your-email>
```


3. Créez un fichier `config.yml` sur votre poste de travail avec les détails suivants. Assurez-vous de ne pas modifier l’alignement pour les paramètres.

```yaml
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: talend-standard-job
spec:
schedule: "/3 * * * *"
jobTemplate:
spec:
template:
spec:
  containers:
  - name: kubernetes-job
    image: dockerId/standard_job:latest
    restartPolicy: Never
    imagePullSecrets:
    - name: regcred
  backoffLimit: 4
```


4. Cliquez sur le bouton **Settings** pour la fenêtre comme ci-dessous et cliquez sur **Upload file** afin de choisir et charger le fichier `config.yml` créé.
5. Une fois le chargement du fichier terminé, exécutez la commande suivante dans Cloud Shell.

    kubectl create -f config.yml

Cela crée un Job cron nommé `talend-standard-job`, comme ci-dessous.

6. Une fois le Job cron créé, cliquez dessus pour voir les détails suivants.
Ce Job est ordonnancé pour s'exécuter toutes les trois minutes.

Pour voir les logs d'exécution du Job, cliquez sur les logs du conteneur dans la page Cron Job details.

Ordonnancement et exécution d'un Job Talend sur le moteur Google Kubernetes