Talend Cloud Migration use case – Hands-On Training

Use case description

Flora's Flowers sells cut flowers across the Southeastern US. Flora, a Talend Studio 6.4 customer, has a small on-premises Talend footprint and now wants to modernize her company's IT and move to the cloud. Flora has already been sold on Talend Cloud and is a happy Talend Customer. Your task is to evaluate her on-premises Talend installation and migrate all of it to Talend Cloud.

Recommended Talend Academy training

- Talend Cloud Administration
- Migration to Talend Cloud Introduction
- Migration to Cloud for CSMs 02

Prerequisites

- Basic knowledge of Windows
- Basic knowledge of Linux
- Basic knowledge of SQL
- Talend Cloud account - you can sign up for a 14-day free trial account here: https://iam.us.cloud.talend.com/idp/trial-registration
- GitHub account: https://github.com/

Training environment

Talend versions

- Talend Studio 6.4
- Talend Studio 7.3
- Talend Cloud
- Talend Cloud Remote Engines 2.8.1

Third-party software

- My SQL 5.7
- Notepad++
- Visuals
- GitHub (SaaS)

Technical Prerequisites

- System requirements (as provided by Talend Academy):

OS	CPU	RAM	SSD Disk Size
Windows	Intel i7 Processor 2 Cores or equivalent	13 GB	50 GB
Linux	Intel i7 Processor 2 Cores or equivalent	13 GB	50 GB

Contents

Overview

To modernize her entire IT department, Flora has retained your services to help her take her on-premises Talend infrastructure into Talend Cloud. She is looking forward to moving to a cost-effective platform that facilitates growth.

Your job as a consultant:

- Evaluate Flora's on-premises Talend footprint. What does she have today? Are there challenges to moving her Talend 6.4.1 administrative functions and Jobs to Talend Cloud?
- After your evaluation is complete, you may begin the process of moving her on-premises Talend 6.4.1 items to Talend Cloud.
    - Start with administrative items
        - Users
        - Projects
        - SDLC environments
    - Finish with Flora's Sales and Analytics Talend Jobs
- Along the way, build out her new Talend Cloud environment, including the installation of two Remote Engines in her on-premises server.
- Finally, complete her new SDLC process by promoting Talend Cloud Artifacts and Tasks in her Development environment to her Production environment.

Working environment configuration

Your Talend Academy sandbox is provisioned with a Windows 2012 R2 server and an Ubuntu Linux server. The Windows server represents Flora's development client, and the Linux server represents her infrastructure back end.

You may need to edit the host files on each server to enable seamless communication between the two servers. This is covered in the next section.

Windows server

Your Talend Academy Windows server contains the following installed and, where noted, configured software:

Talend Studio 6.4.1

You have to create a connection to the Talend Administration Center running on the Linux server in Talend Studio 6.4.1.

Talend Studio 7.3.1

In this use case, Studio 7.3.1 is solely intended to work with Talend Cloud. You are required to create a connection to your Talend Cloud account from this copy of Studio.

VisualSVN

Source control for Flora's on-premises Talend 6.4.1 system is Subversion. It runs on the Windows server and has Flora's Talend repository installed.

MySQL Server

MySQL Server 5.7 is installed and running on your Windows server. Talend Administration Center that runs on the Linux server accesses this instance of MySQL. A Talend Administration Center database is configured and installed in your MySQL server.

MySQL Workbench

The standard Windows client for MySQL, called MySQL Workbench 6.0, is installed on your Windows server. You can use this tool to verify Flora's Sales and Analytics tasks.

Tools

Your Windows server contains tools such as Notepad++, and the Google Chrome and Mozilla Firefox browsers.

Linux Server

The Linux server runs Ubuntu Linux 16.04, and the following items are installed:

Talend Administration Center 6.4.1

You have to associate it with the Talend Administration Center database found in the MySQL server on your Windows server. This database contains all the users, configurations, and licenses necessary to complete this use case.

Talend JobServer 6.4.1

While Flora's architecture calls for two JobServers: one for development and one for production, to save space and memory, only one is installed (development). The second JobServer is configured in your Talend Administration Center Job Conductor, but it is disabled. It is not necessary to instantiate a production JobServer to complete this use case, just be aware Flora is configured for two JobServers.

Talend Cloud Remote Engine 2.8.1

For your convenience, two copies of the Talend Cloud Remote Engine are installed on your Linux server. While Flora only has a single Talend 6.4.1 JobServer, you deploy both a Development and a Production Remote Engine to match her SDLC. You can use the Talend Cloud Remote Engine pairing URL when it comes time to pair these with your Talend Cloud account.

Talend Cloud

This use case does not include a Talend Cloud account, which is required. However, you can use the trial Talend Cloud account you registered for in the Prerequisites section.

Setting up the working environment

Configuring Talend Administration Center 6.4.1

From the Linux command prompt, here is how you would start and stop the Talend Administration Center:

Start:

```
$ sudo systemctl start talend-tac-6.4.1
```

Stop:

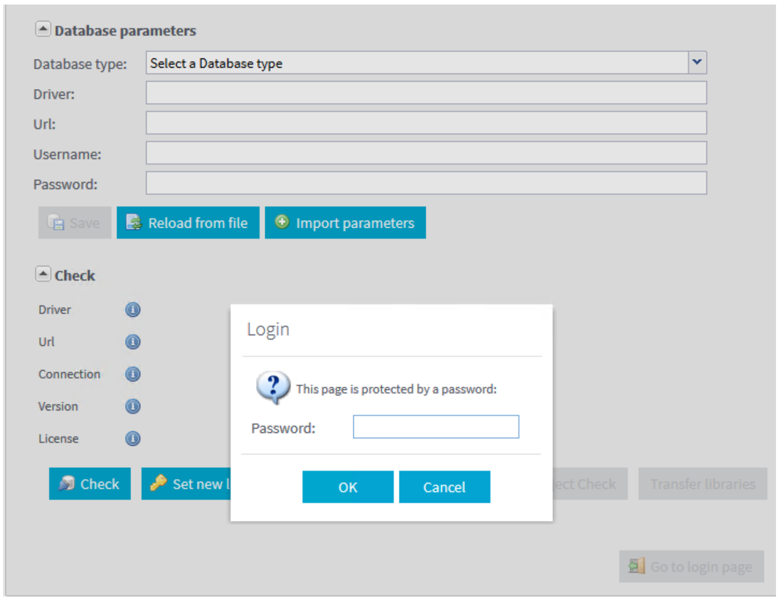
```
$ sudo systemctl stop talend-tac-6.4.1
```

To connect your copy to the preconfigured Talend Administration Center database in your MySQL server, open a browser and access the Talend Administration Center URL:

```
http://tac-server:8080/org.talend.administrator/
```

NOTE: Your Talend 6.4.1 on-premises configuration may already be provisioned with a student license and configured for database access. If so, you will be taken to the Talend Administration Center login page, and you can skip the rest of this section.

The Talend Administration Center database configuration page should open. If not, click the **Go to db config page** link at the bottom of the main login page.


Database parameters
Database type: Select a Database type
Driver:
Url:
Username:
Password:

Login
This page is protected by a password:
Password:

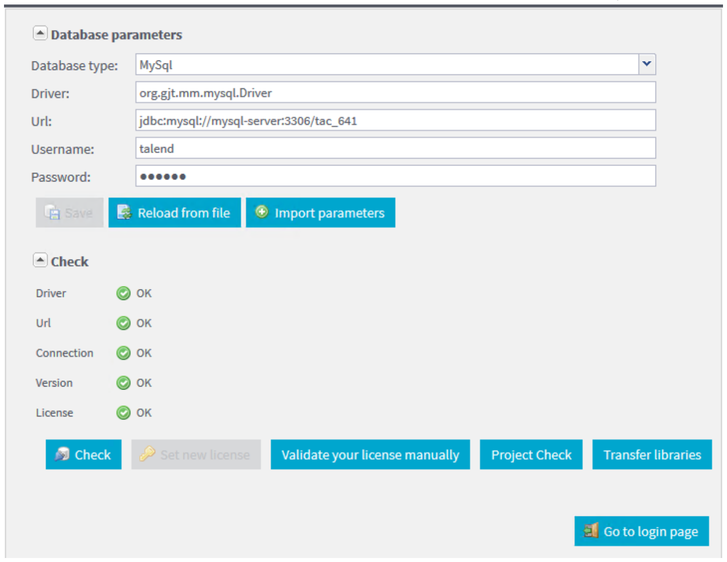
Access the page by providing the default Talend admin password: *admin*

- Select **MySQL** from the **Database type** drop-down.
- The **Driver** field auto-populates.
- Add the **Url** for the MySQL server:

```
Jdbc:mysql://mysql-server:3306/tac_641
```

- Enter the MySQL Username: *talend*
- Enter the MySQL password: *talend*

If needed, click **Save** then click **Check**. All checks should pass with **OK** status. If successful, click **Go to login page**.


Database parameters
Database type: MySql
Driver: org.gjt.mm.mysql.Driver
Url: jdbc:mysql://mysql-server:3306/tac_641
Username: talend
Password: ●●●●●●

Check
Driver ✓ OK
Url ✓ OK
Connection ✓ OK
Version ✓ OK
License ✓ OK

You can now log in to Talend Administration Center user interface with Flora's production user:

```
User Name:  prod@company.com
Password:   Talend2020
```

Configuring Talend JobServer 6.4.1

From the Linux command prompt, start and stop the JobServer as follows:

Start:

```
$ sudo systemctl start talend-rjs-6.4.1
```

Stop:

```
$ sudo systemctl stop talend-rjs-6.4.1
```

Configuring Talend Studio 6.4.1

Connect Talend Studio 6.4.1 to your Talend Administration Center, on your Windows workstation. Start Talend Studio 6.4.1. and add a remote connection to your Talend Administration Center server as follows:

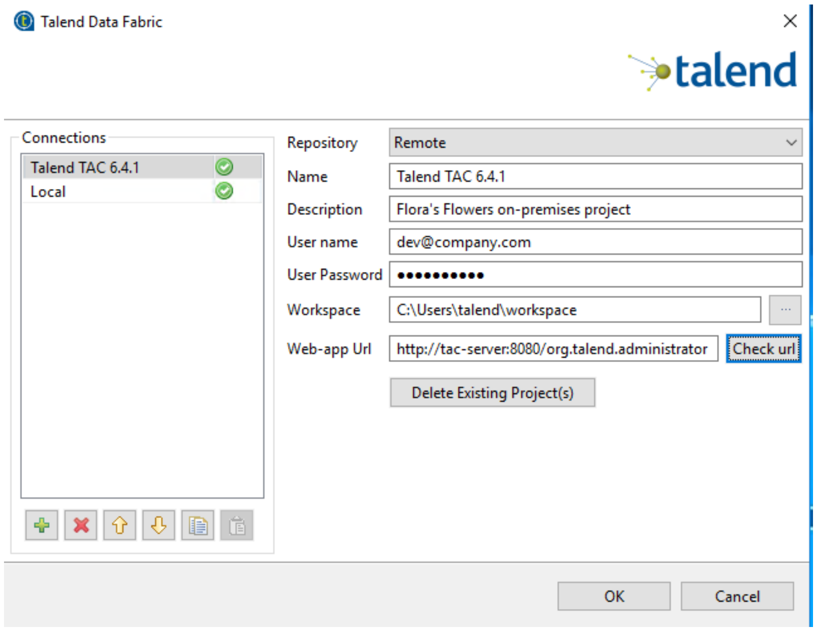
- Select **Remote** from the **Repository** drop-down list.
- Add a name, in this case, **Talend TAC 6.4.1**, and a description.
- Enter Flora's developer's username and password:

```
User Name: dev@company.com
Password:  Talend2020
```

- Leave the **Workspace** field the same.
- The **Web-app Url** is the URL for your Talend Administration Center server:

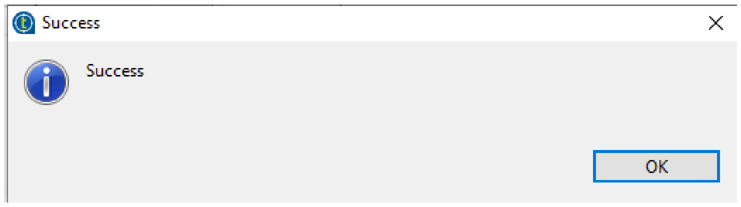
```
http://tac-server:8080/org.talend.administrator
```

Click **Check URL**. If everything is configured correctly, the **Success** dialog box opens. Click **OK**.


Talend Data Fabric

Connections: Talend TAC 6.4.1, Local

Repository: Remote
Name: Talend TAC 6.4.1
Description: Flora's Flowers on-premises project
User name: dev@company.com
User Password: ••••••••••
Workspace: C:\Users\talend\workspace
Web-app Url: http://tac-server:8080/org.talend.administrator


Success

Success

Configuring Talend Studio 7.3.1

In this use case, Talend Studio 7.3.1 is exclusively for use with Talend Cloud. The configuration is covered later in this document.

Starting Talend Cloud Remote Engines

The two copies of the Talend Cloud Remote Engine version 2.8.1, installed for you, will become Flora's development and production engines.

Start the remote engines with the following Linux command:

```
$ sudo systemctl start Talend-Remote-Engine-dev
$ sudo systemctl start Talend-Remote-Engine-prod
```

Stop the remote engines with the following Linux command:

```
$ sudo systemctl stop Talend-Remote-Engine-dev
$ sudo systemctl stop Talend-Remote-Engine-prod
```

NOTE: You must start and stop these separately.

The paring URLs for the remote engines are as follows:

```
Development: http://re-server:8043/configuration
Production:  http://re-server:8044/configuration
```

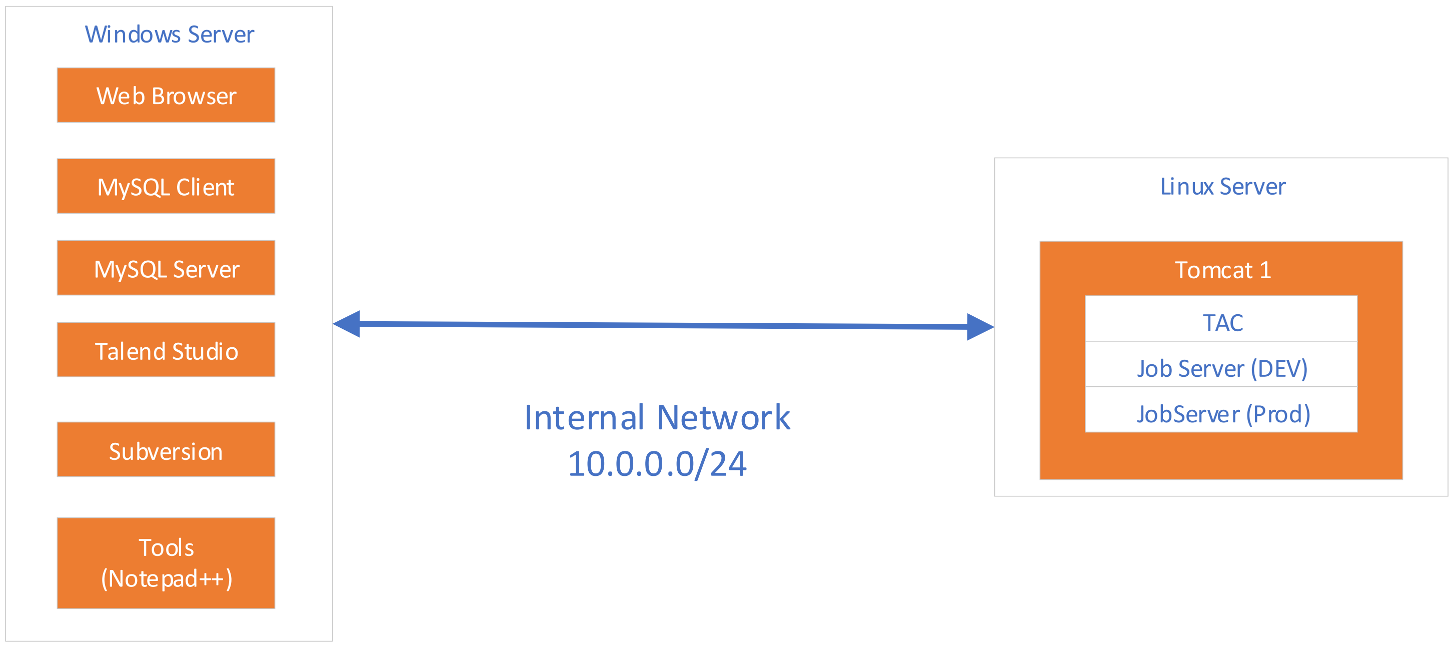
Creating a Git repository

Creation of the Git repository in GitHub, where you'll store the on-premises Talend project, is covered later in this use case. This repository will be added to your Talend Cloud account.

Preparing for migration

Reviewing the Talend 6.4.1 environment

The first thing to do when preparing to migrate to Talend Cloud is to review the on-premises Talend installation. The physical architecture of the on-premises Talend Academy sandbox is shown below:



Talend Administration Center

Start a browser on your Windows VM and login to the on-premises Talend Administration Center using the **Talend 6.4.1 TAC** connection in your Talend Academy sandbox as described in Talend Administration Center 6.4.1, in the Set up required section.

Select the **Licenses** menu. Notice that Flora has five named Talend users.

Click the **Users** menu to see that Flora uses three of her five Talend users:

User Name	Description
security@company.com	The security admin user, by default, is only granted the Security Administrator role.
dev@company.com	Flora's developer uses this account. This user does not have access to run any Jobs in production and is granted the Operation Manager and Designer roles.
prod@company.com	Flora's production manager uses this account. This user can run Jobs in production and is granted the Security Administrator, Administrator, and Operation Manager roles.

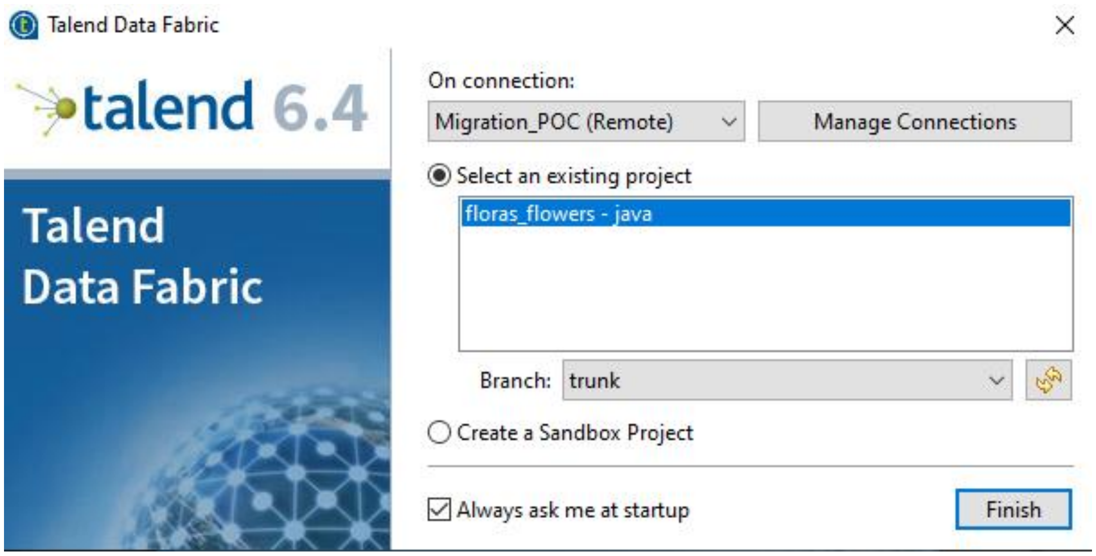
Next, click **Projects**. Flora has a single Talend project called **floras_flowers**. Click **Project Authorizations** to see the access her users are granted:

- dev@company.com has read/write access to the project.
- prod@company.com has read-only access to the project.

On **Talend Administration Center** menu, navigate to the **Conductor** section and click **Servers**. Flora has two JobServers, one for development and one for production. Click each JobServer and read the **Description** field to differentiate between them. In this use case, only the development JobServer exists on your Linux virtual machine. The production JobServer is configured but deactivated. It is not necessary to run any Jobs in either JobServer. In a real-world use case, it is common for development and production JobServers to be separated on different computing resources.

Exporting 6.4.1 project from Studio

Log out of Talend Administration Center. Start Talend Studio 6.4.1 on your Windows VM. Log in to the remote **floras_flowers** project.



On connection: Migration_POC (Remote)

Select an existing project: floras_flowers - java

Branch: trunk

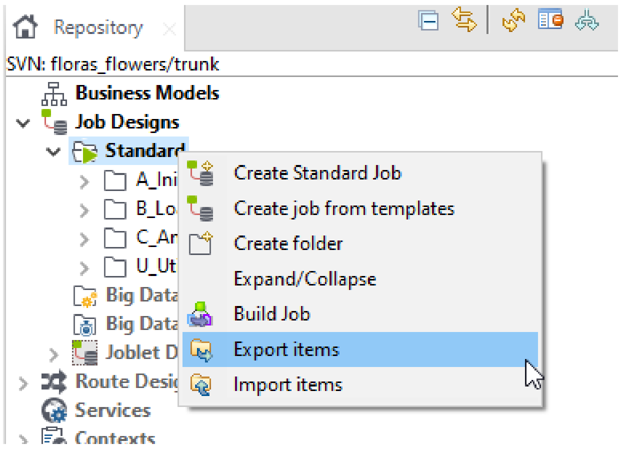
Always ask me at startup ✓

Take a brief look at the Talend Jobs provided. Don't edit any!

NOTE: Flora uses Subversion as the source control system for her Talend project, and she follows best practices for older versions of Talend on-premises products. However, Talend Cloud does not support Subversion; it only supports Git-based source control. The easiest way to move Flora's Talend Jobs from Subversion to Git (in this case, GitHub), is to export them from Talend Studio 6.4.1, and import them into the new version of Studio connected to the new Git type of source control.

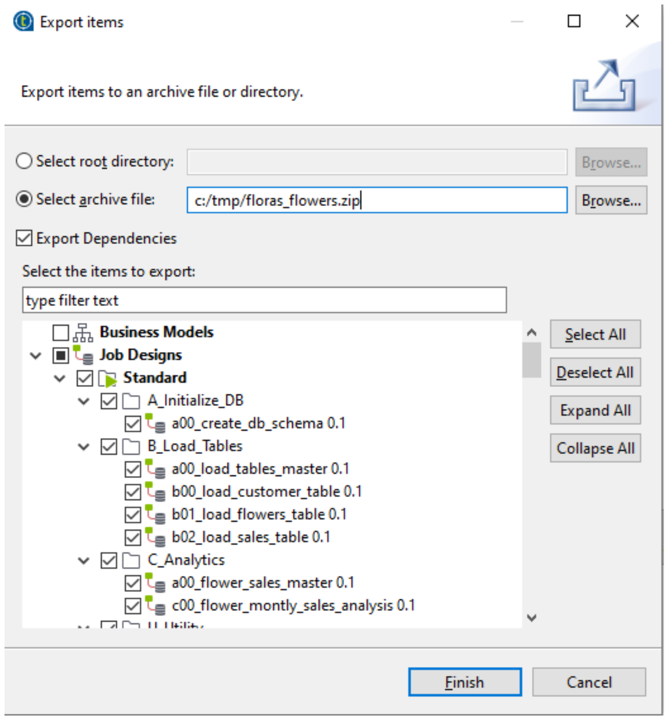
To begin the process of moving Flora's Talend Jobs to Talend Cloud and Studio 7.3.1, you need to export the Jobs and all their dependencies from Studio 6.4.1 to a .zip file.

To export all artifacts from the **floras_flowers** project, in the **Repository** view, expand **Job Designs**. Right-click **Standard**, then select **Export items**.



In the **Export items** dialog box:

- Click the **Select archive file** radio button.
- Enter the path and filename for your **.zip** file.
- Select the **Export Dependencies** check box.
- Click **Finish**.



Double-check that the .zip file was created in the directory you specified.

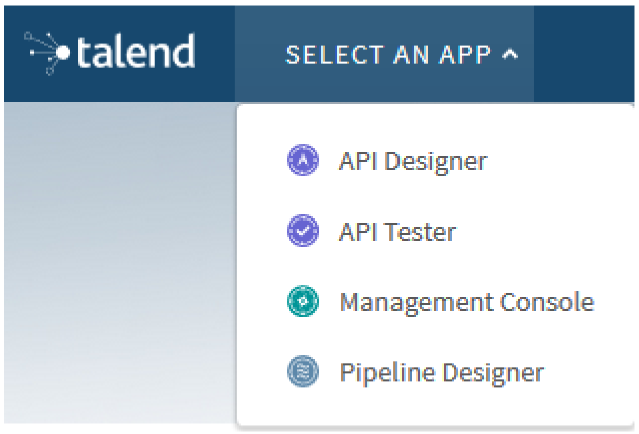
Finally, close Talend Studio 6.4.1 and log in to your Linux server and stop Talend Administration Center. It won't be needed again in this use case.

```
$ sudo systemctl stop talend-tac-6.4.1
```

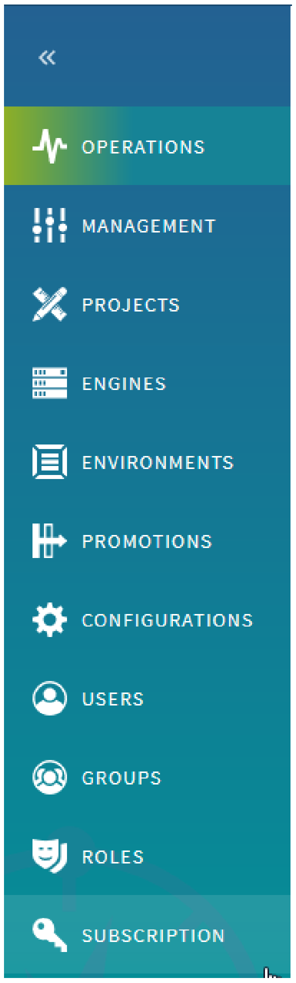
Reviewing Talend Cloud counts and licenses

Log in to your Talend Cloud account and check the user and token counts.

Select **Management Console** from the list of applications.



Select **Subscription** from the menu on the left to view your subscription license.



Notice the user and token counts. Using the Talend Cloud trial account, you will have at least two users and 18,000 tokens. You use tokens when you create a Cloud Engine or a Remote Engine to run Talend Cloud tasks. In this use case, you use only Remote Engines.

Migrating on-premises users to Talend Cloud

In this section, you create users in Talend Cloud that improve on the Talend users managed by the on-premises Talend Administration Center. To review, in Flora's Talend Administration Center, the following users are defined and active:

- security@company.com
- dev@company.com
- prod@company.com

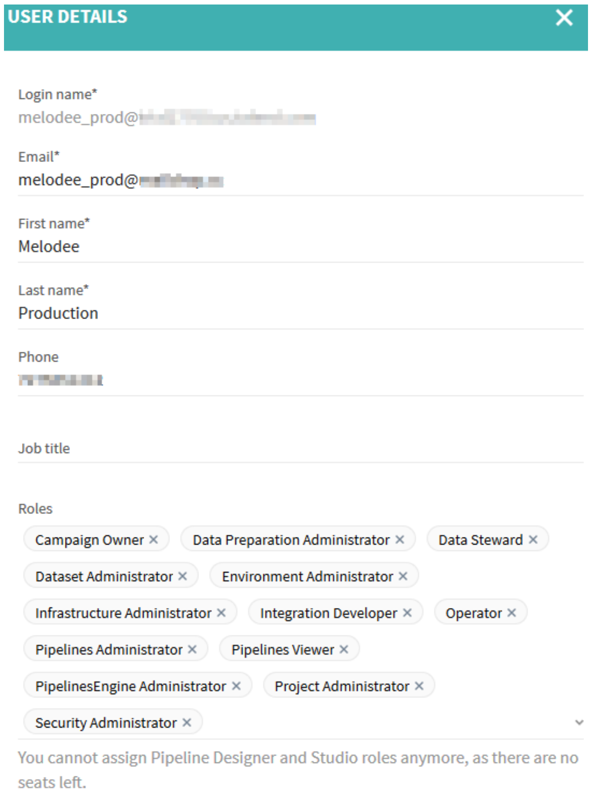
In Talend Cloud, each named user must have an active email account. You cannot share user IDs among a pool of people. For the trial Talend Cloud account referred to as Talend Cloud account for the remainder of this use case, you use a set of preset email addresses.

You use two user IDs to mirror what is on-premises for Flora's Flowers. Be sure you have one developer and one user as an administrator and production user. To complete the use case, your Talend Cloud production user ID needs the **Security Administrator** role assigned.

The names may be different, as all must match valid email addresses. Email domains are masked because they will be different for each person. This use case uses the following:

- Angus_dev<@domain>
- Melodee_prod<@domain>

Log in to your Talend Cloud account with the user ID provided. In this case, **Melodee** requested an evaluation account. In Talend Cloud, go to the **Management Console** and select **Users** from the menu on the left. You see, one user-defined. Click the username to display displays details about this user ID.



Login name*
melodee_prod@[redacted]

Email*
melodee_prod@[redacted]

First name*
Melodee

Last name*
Production

Phone
[redacted]

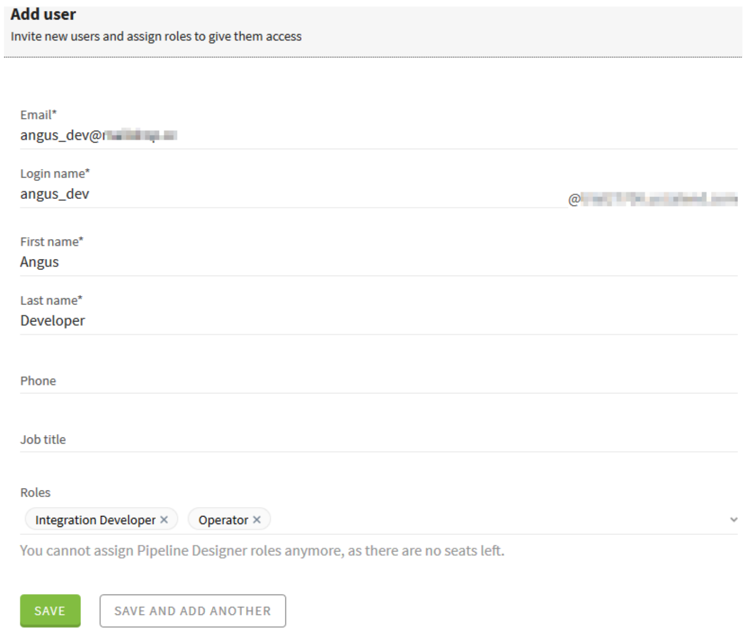
Job title

Roles
Campaign Owner ×, Data Preparation Administrator ×, Data Steward ×, Dataset Administrator ×, Environment Administrator ×, Infrastructure Administrator ×, Integration Developer ×, Operator ×, Pipelines Administrator ×, Pipelines Viewer ×, PipelinesEngine Administrator ×, Project Administrator ×, Security Administrator ×

You cannot assign Pipeline Designer and Studio roles anymore, as there are no seats left.

Notice that your initial user has all relevant roles granted. Revoke the **Integration Developer** role from this user. You will reassign this to the developer user. Save the user details. Click the **X** in the upper-right corner to close this dialog.

Click the **Add User** button at the top of the **Users** page and create a developer user. Be sure to provide a valid email address. Assign the **Integration Developer** and **Operator** roles to this user.



Add user
Invite new users and assign roles to give them access

Email*
angus_dev@[redacted]

Login name*
angus_dev @[redacted]

First name*
Angus

Last name*
Developer

Phone

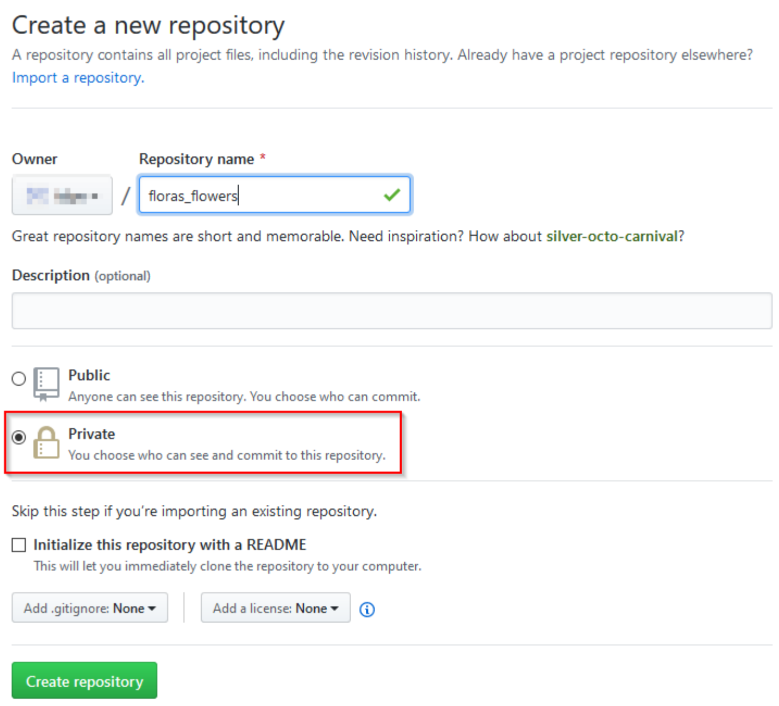
Job title

Roles
Integration Developer × Operator ×
You cannot assign Pipeline Designer roles anymore, as there are no seats left.

Click **Save**, and you are prompted to send an email to this user. This allows the user to log in for the first time and set a password. Click **Send Email**. Retrieve the email from the user's email account and log in to Talend Cloud to set the password and test the ID.

Creating a GitHub repository

If you do not have one already, create a free GitHub account. Use this account to create a new private Git repository for use with Talend Cloud.



Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner [redacted] / Repository name * floras_flowers

Great repository names are short and memorable. Need inspiration? How about silver-octo-carnival?

Description (optional)

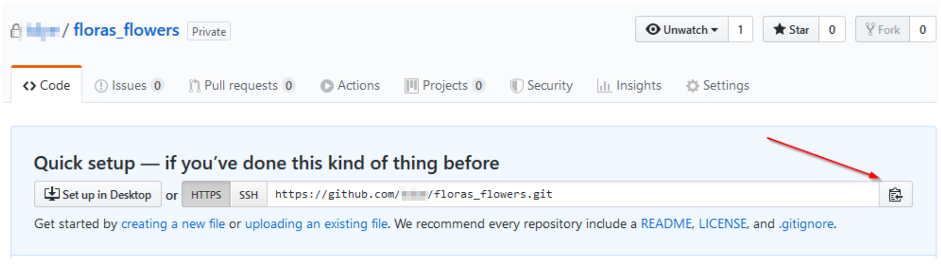
Public
Anyone can see this repository. You choose who can commit.

[✓ Private]
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer.

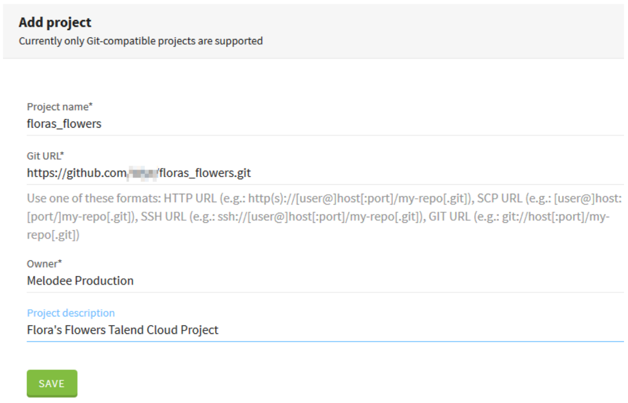
On the GitHub page that displays after you create a project, be sure **HTTPS** is selected, then click the **clipboard** icon to copy the project URL.


/ floras_flowers Private
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Actions
Projects 0
Security
Insights
Settings
Quick setup — if you've done this kind of thing before
Set up in Desktop or HTTPS SSH
https://github.com/ /floras_flowers.git
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

Creating and publishing projects

Create a new empty GitHub project in Talend Cloud. Return to Talend Cloud as the user with the **Project Administrator** role. In this use case, it is **Melodee_Prod**.

Select **Projects** from the **Management Console** menu, click **Add Project** at the top of the page. Provide a project name and paste the Git URL you copied in the previous step. Add a description and click **Save**.



Add project
Currently only Git-compatible projects are supported

Project name*
floras_flowers

Git URL*
https://github.com/ /floras_flowers.git

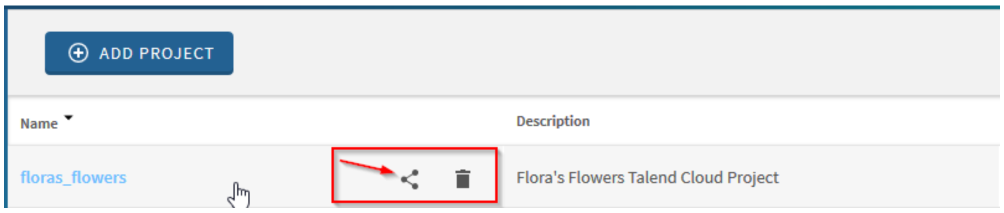
Use one of these formats: HTTP URL (e.g.: http(s)://[user@]host[:port]/my-repo[.git]), SCP URL (e.g.: [user@]host:[port/]my-repo[.git]), SSH URL (e.g.: ssh://[user@]host[:port]/my-repo[.git]), GIT URL (e.g.: git://host[:port]/my-repo[.git])

Owner*
Melodee Production

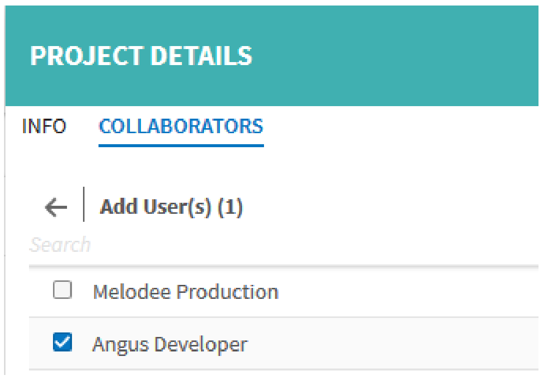
Project description
Flora's Flowers Talend Cloud Project

SAVE

After you save the project, you are returned to the main project page. Hover your mouse pointer over the project name, and a set of additional controls appears. Click the **Sharing** icon.



On the **Project Details** tab, select **Collaborators** and add the **developer** user to the project.



Logging in to Talend Cloud project from Studio

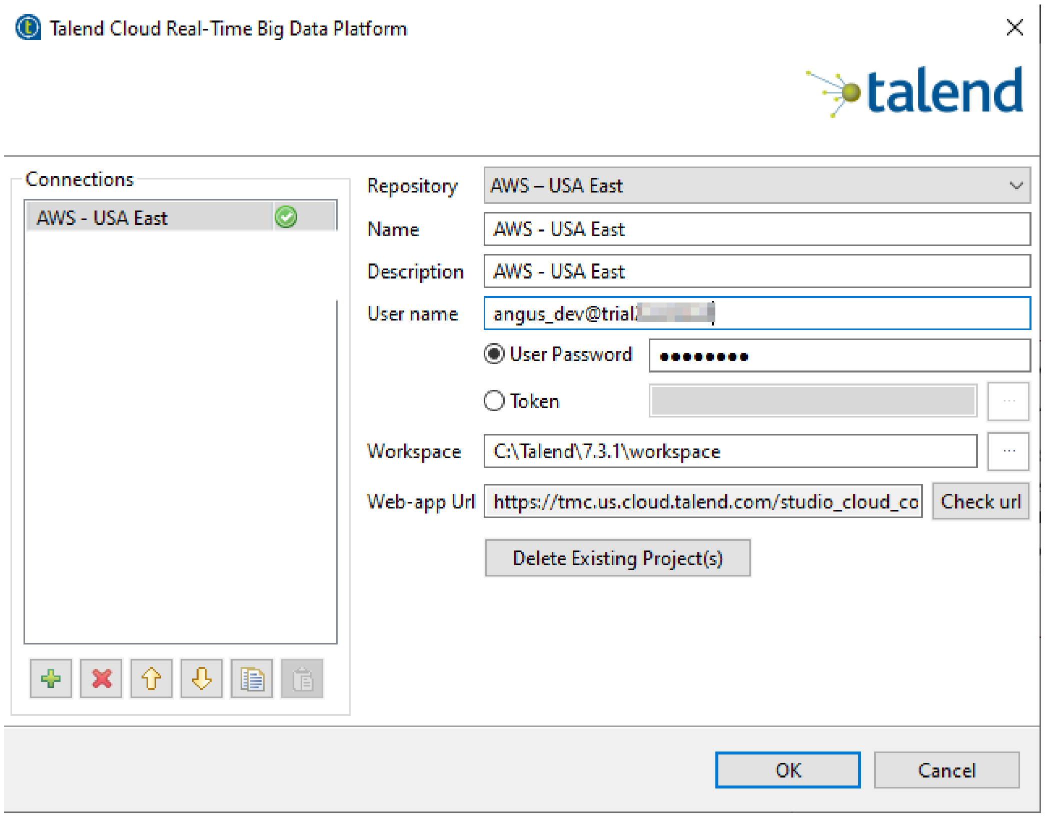
In this section, you configure Talend Studio 7.3.1 to access your Talend Cloud project. To do this, you add your Talend Cloud access credentials and configure Studio to allow publication of Talend Jobs to Talend Cloud. You then log in to your project in Studio and ensure the project is configured correctly in Talend Cloud.

Start Talend Studio 7.3.1 on your Windows VM in your Talend Academy sandbox. Make sure you're not using the on-premises 6.4.1 version.

In the login window, click the **Manage Connections** button. The **Connections** dialog box opens. Click the green plus sign to add a new connection.

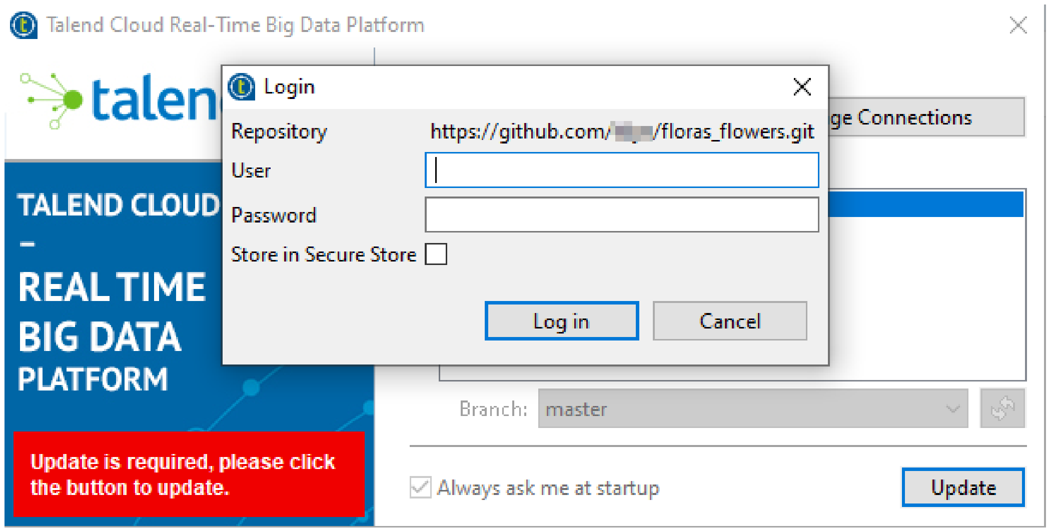
- For **Repository**, select the Talend Cloud data center where your account is provisioned. You can find this in the email notification you received when you registered for Talend Cloud or on the login screen when you access Talend Cloud.
- Supply a meaningful name and, optionally, a description.
- Enter the **User name** found in Talend Cloud on the Users page. It is not the user's email address, but a string assigned to the user when they are created in Talend Cloud. It looks like an email address, but probably longer. Be sure you are using your developer's Talend Cloud ID. Flora's production user does not get Studio access to the project.
- Accept the default values for the rest of the fields.

Click **Check url** to test your login credentials. If you get a success message, click **OK**. If not, recheck your login parameters.


Talend Cloud Real-Time Big Data Platform
talend
Connections
AWS - USA East
Delete Existing Project(s)
Check url
OK
Cancel

Repository: AWS – USA East
Name: AWS - USA East
Description: AWS - USA East
User name: angus_dev@trial[redacted]
[✓ User Password] ●●●●●●●●●
Token
Workspace: C:\Talend\7.3.1\workspace
Web-app Url: https://tmc.us.cloud.talend.com/studio_cloud_co

After successfully testing the connection to Talend Cloud and saving the connection parameters, Studio attempts to log you into your Talend Cloud account. If prompted, enter your Git/GitHub user ID and password (or token), then select **Store in Secure Store** to prevent being prompted for these again.


Talend Cloud Real-Time Big Data Platform
talend
TALEND CLOUD – REAL TIME BIG DATA PLATFORM
Login
ge Connections
Log in
Cancel
Update

Repository https://github.com/[redacted]/floras_flowers.git
User
Password
Store in Secure Store ☐

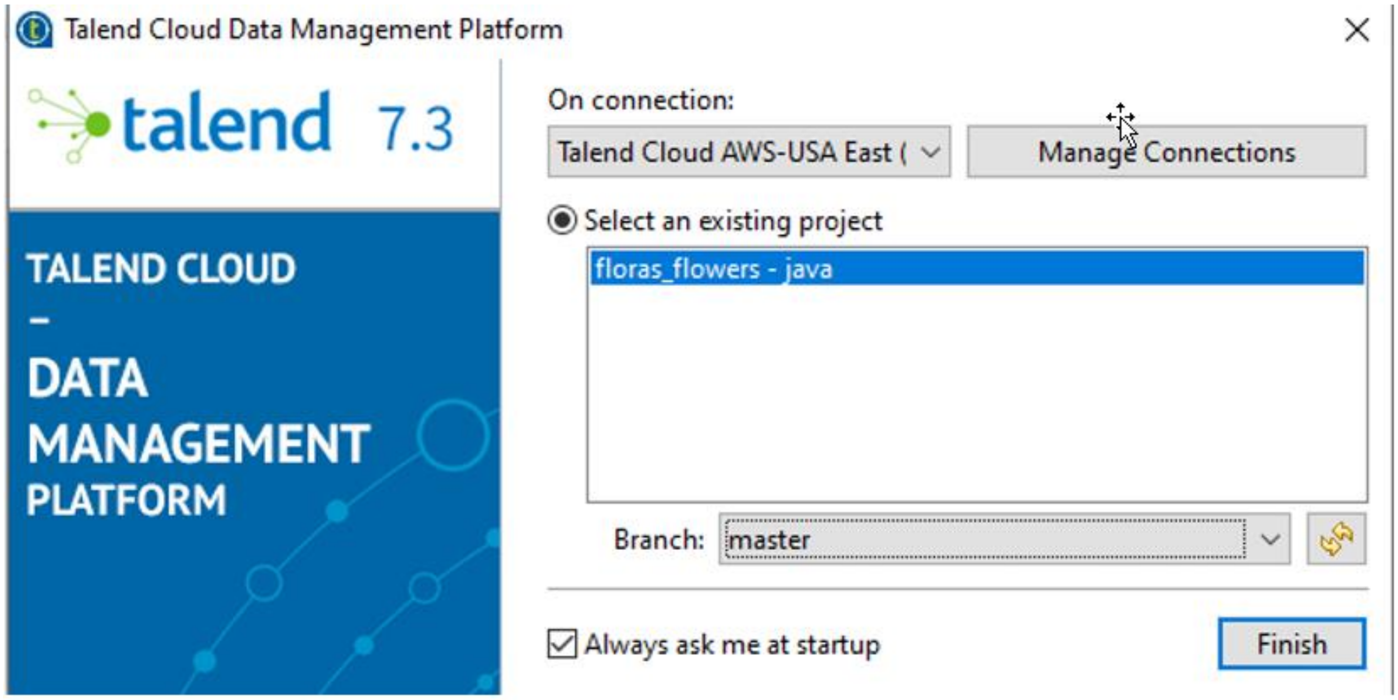
Update is required, please click the button to update.

Branch: master

☑ Always ask me at startup

You may also be prompted to update your workspace. If so, your connection button changes to **Update**. Click it and wait until Studio resets.

You are eventually presented with a page like this:


Talend Cloud Data Management Platform
talend 7.3
TALEND CLOUD – DATA MANAGEMENT PLATFORM
Manage Connections
Finish

On connection: Talend Cloud AWS-USA East (

(●) Select an existing project

floras_flowers - java

Branch: master

☑ Always ask me at startup

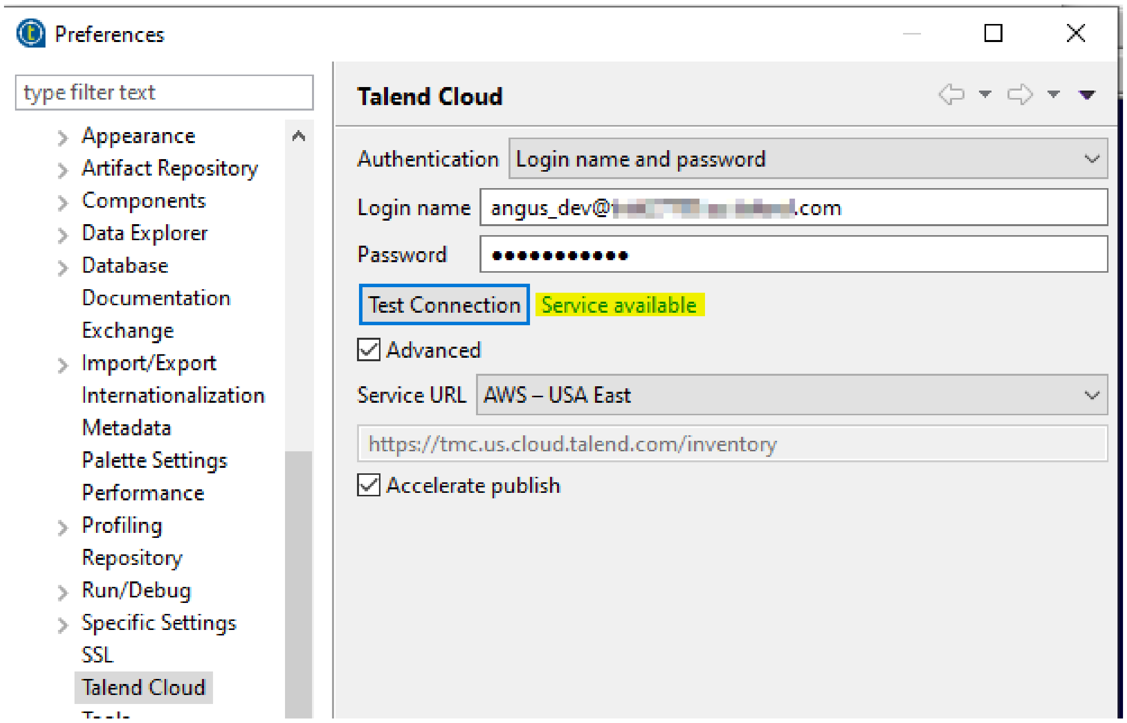
Click **Finish** to log in to Studio with your Talend Cloud project.

Configuring Studio to publish to Talend Cloud

Now, you can log in to your project in Talend Cloud, but you need to supply the configuration information that enables you to publish a Job to Talend Cloud.

In Studio, navigate to **Window** > **Preferences** > expand **Talend**, and select **Talend Cloud**.

Enter the developer user's login name and password, then select the **Advanced** check box and be sure your **Service URL** is set to the Talend Cloud data center where your Talend Cloud account is registered. Click **Test Connection**. If everything is configured correctly, the words **Service available** appear to the right of the **Test Connection** button. Click the **Apply and Close** button.



Talend Cloud

Authentication: Login name and password

Login name: angus_dev@[redacted].com

Password: ●●●●●●●●●●●●

Test Connection   Service available

☑ Advanced

Service URL: AWS – USA East

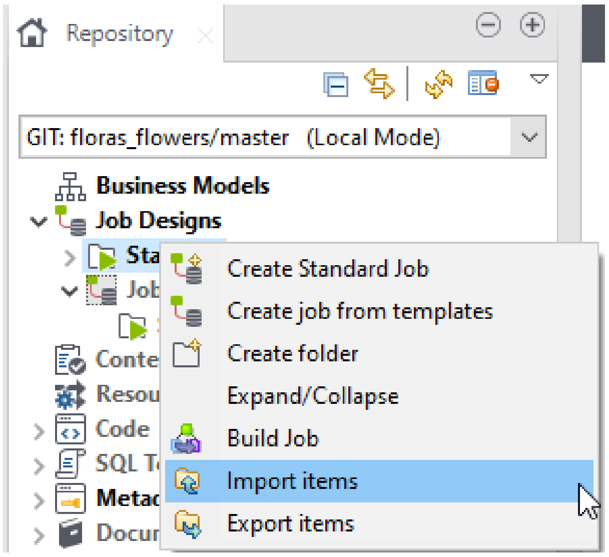
https://tmc.us.cloud.talend.com/inventory

☑ Accelerate publish

Importing the project to Studio 7.3

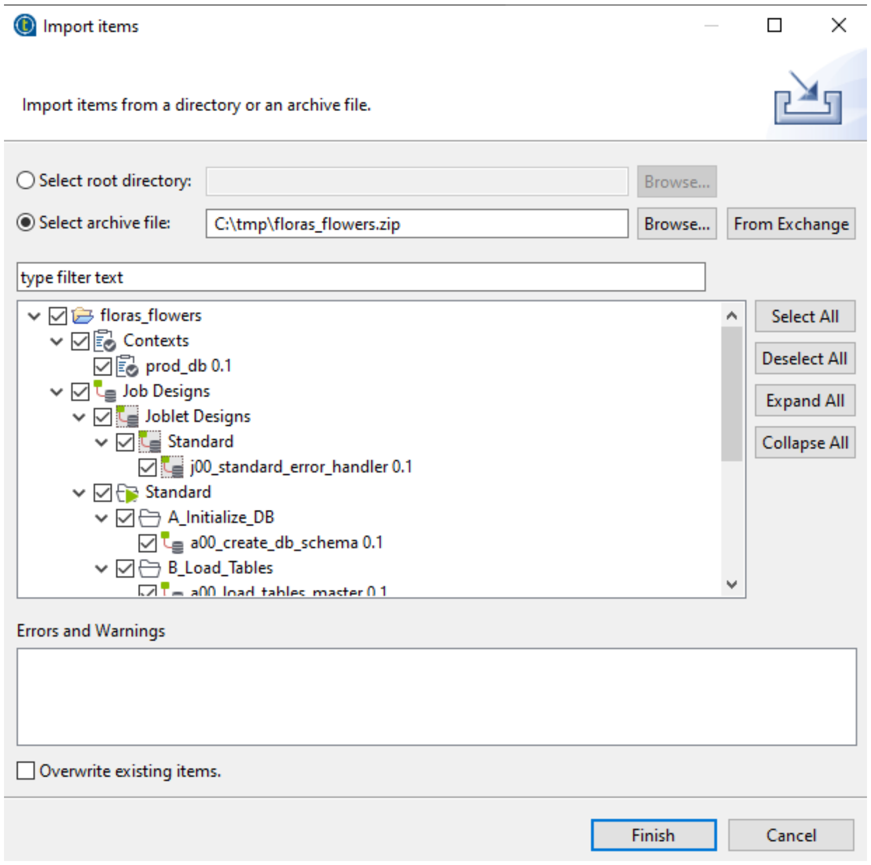
Now that you are logged into your Talend Cloud project in Studio, you can import your on-premises Talend 6.4.1 project artifacts into your new 7.3.1 project and store them in the Git/GitHub repository.

In Studio, in the **Repository** view, expand **Job Designs,** right-click **Standard**, and select **Import items**.



In the **Import items** dialog box, enter the path and filename for the project **.zip** file you created in the Migration Preparation section.

Click the **Select All** button and click **Finish**. All your 6.4.1 project artifacts are imported into Talend 7.3.1 and stored in your GitHub repository.


Import items

Import items from a directory or an archive file.

Select root directory:

Select archive file: C:\tmp\floras_flowers.zip

type filter text

floras_flowers
Contexts
prod_db 0.1
Job Designs
Joblet Designs
Standard
j00_standard_error_handler 0.1
Standard
A_Initialize_DB
a00_create_db_schema 0.1
B_Load_Tables
a00_load_tables_master 0.1

Errors and Warnings

Overwrite existing items.

Talend Cloud SDLC Environments and Workspaces

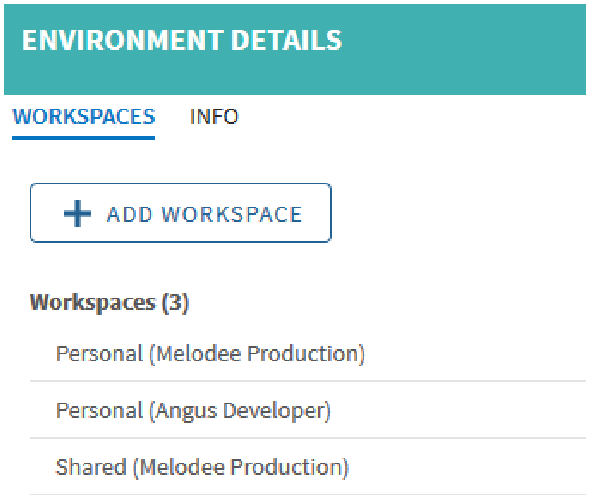
Before you can migrate and publish any Jobs to Talend Cloud, you must configure your environments for SDLC. Earlier, you discovered that Flora's Flowers uses two lifecycle environments: development and production. In real situations, many organizations use three or more environments with one Talend Administration Center for each environment, but Flora uses a single Talend Administration Center for both development and production. Her developer stages Jobs in the development JobServer where they are tested and run with test data, then they are promoted to the production JobServer for execution with Flora's production sales and analytics databases.

Mirror her promotion process and improve on it in Talend Cloud.

In Talend Management Console, set up Development and Production environments, each with a Remote Engine to execute Talend Cloud tasks.

From the Talend Management Console, select the **Environments** menu item. You see some default items configured, including an environment named **Default** with some workspaces listed as configured. Click the default workspace to see what is there.

With the Talend Cloud trial account, you see something like this:



ENVIRONMENT DETAILS

WORKSPACES INFO

ADD WORKSPACE

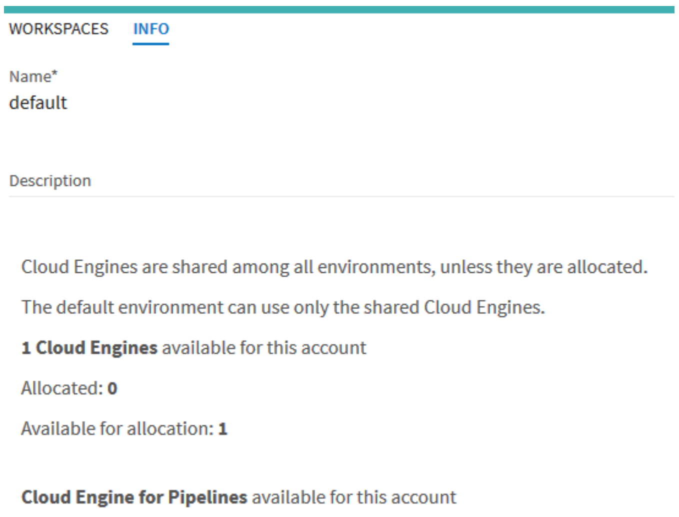
Workspaces (3)

Personal (Melodee Production)

Personal (Angus Developer)

Shared (Melodee Production)

Click the **INFO** link to get details.



WORKSPACES INFO

Name*
default

Description

Cloud Engines are shared among all environments, unless they are allocated.

The default environment can use only the shared Cloud Engines.

1 Cloud Engines available for this account

Allocated: **0**

Available for allocation: **1**

Cloud Engine for Pipelines available for this account

You may have a Cloud Engine available for use and other informational text, but for this use case, you won't use any Cloud Engines or Cloud Engines for Pipelines.

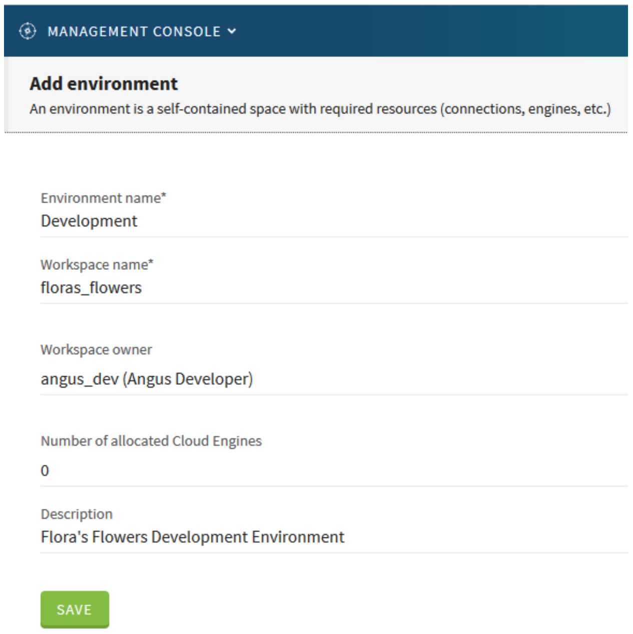
Creating a development environment

Go back to the main **Environments** page and click **Add Environment**.

Configure the **Add Environment** page as follows:

- **Environment name**: *Development*
- **Workspace name**: *floras_flowers*, the same name as your Talend project. Environments can have one or more workspaces. By default, this is a shared workspace, meaning more than one developer can publish to this workspace. While this use case only has a single developer, most companies use a team to do Talend development work.
- **Workspace owner**: your developer user
- **Number of allocated Cloud Engines**: *0*

- **Description:** provide a brief description of the purpose of the environment



Add environment

An environment is a self-contained space with required resources (connections, engines, etc.)

Environment name*: Development

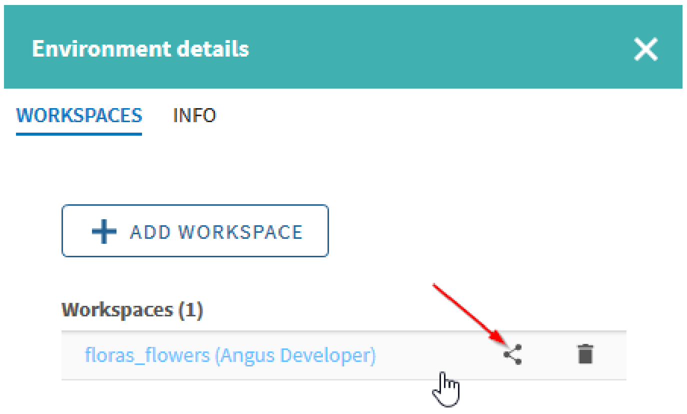
Workspace name*: floras_flowers

Workspace owner: angus_dev (Angus Developer)

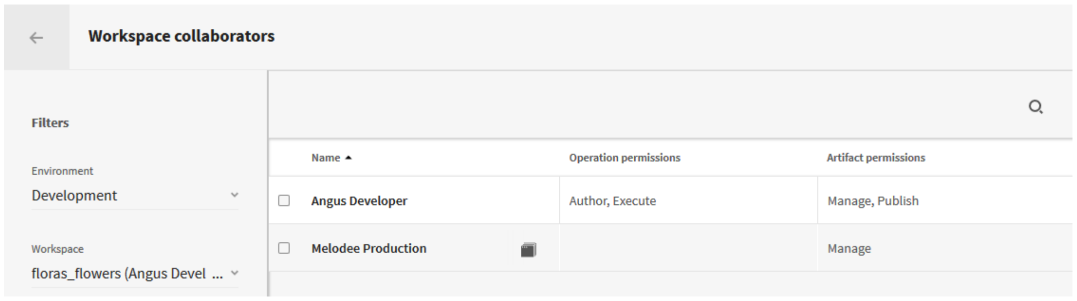
Number of allocated Cloud Engines: 0

Description: Flora's Flowers Development Environment

Click **Save**. By default, the workspace owner (your developer user) has all available permissions on the workspace.



Add the **Manage** permission to your production user. This enables your production user to promote artifacts and tasks to the production environment.



Environment: Development

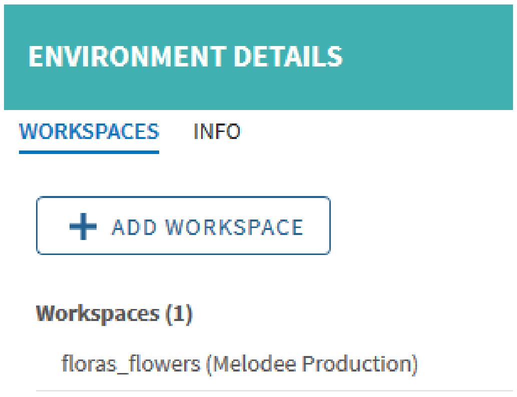
Workspace: floras_flowers (Angus Devel ...

Name	Operation permissions	Artifact permissions
Angus Developer	Author, Execute	Manage, Publish
Melodee Production		Manage

Creating a production environment

You need to create a production environment the same as you did for the development environment. Create a new environment called **Production**, and create a workspace, called **floras_flowers**. This time assign your production user as the owner of the workspace. Do not grant your developer user any access to the production environment or workspace.

The environment details for your Workspaces should look like this:

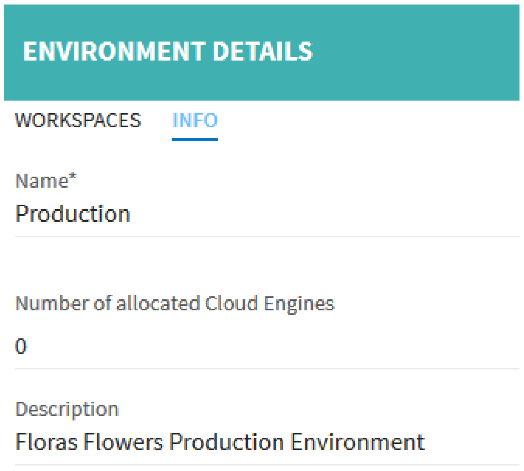

ENVIRONMENT DETAILS

WORKSPACES INFO

Workspaces (1)

floras_flowers (Melodee Production)

The environment details for your Info should look like this:


ENVIRONMENT DETAILS

WORKSPACES INFO

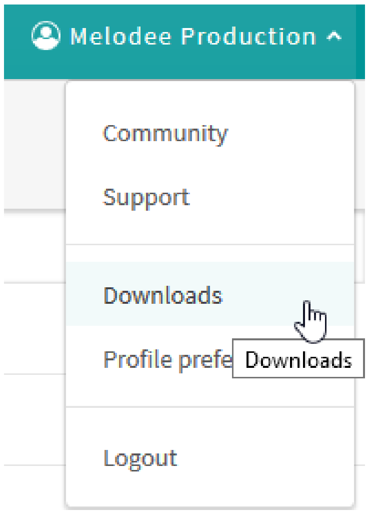
Name*
Production

Number of allocated Cloud Engines
0

Description
Floras Flowers Production Environment

Setting up Remote Engines

Typically, you would download supporting software such as Remote Engines from Talend Cloud from here:



To simplify your tasks, two copies of the Remote Engine software are already installed on your Linux Server.

There are many possible deployments of Remote Engines for use with Talend Cloud, for example:

- Remote Engines support clustering, so it is possible to deploy two or more to act as a single-engine in an environment.
- Remote Engines can be deployed anywhere, including on-premises in your VM or in cloud provider compute resources such as AWS EC2 or Azure VM.
- Remote Engines can be found and installed by accessing the AWS or Azure marketplaces. In this case, all you need to have is your pairing key.

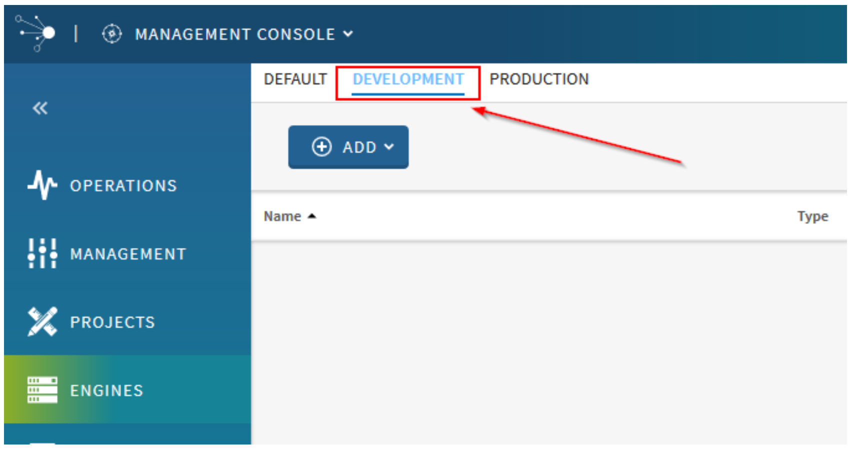
You need to pair each copy of the Remote Engine—pair one with the development environment and the other with the production environment.

Pairing Remote Engines

Your next task is to pair each Remote Engine using the Remote Engines pairing URL.

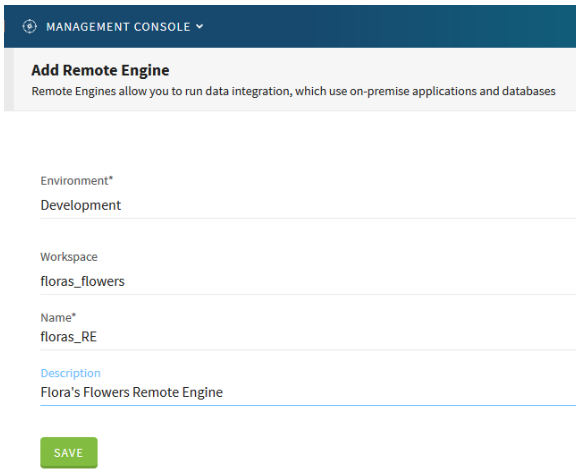
To begin, log in to Talend Cloud with your production users' credentials. Access the **Engines** menu from the **Management Console**.

This step is important! Switch the page to the **Development** environment. If you omit this step, your engines will be created in the default environment. This is not where you want them for use with Flora's defined SDLC.



Click the **Add** button to display a drop-down list with options for Remote Engine and Remote Engine Cluster. Select **Remote Engine**.

Enter the values as shown below. You do not need to name your Remote Engines to indicate development or production.



Add Remote Engine

Remote Engines allow you to run data integration, which use on-premise applications and databases

Environment*
Development

Workspace
floras_flowers

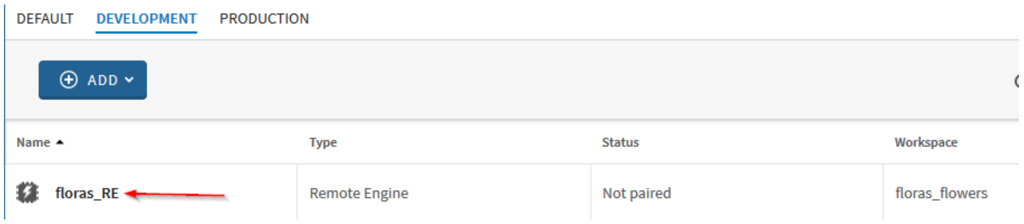
Name*
floras_RE

Description
Flora's Flowers Remote Engine

SAVE

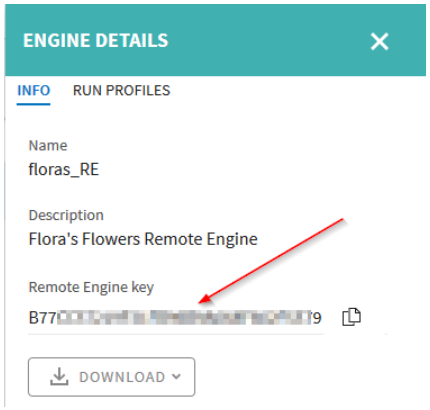
Click **Save**.

Click the **Remote Engine** name to display the pairing key.



Name	Type	Status	Workspace
floras_RE	Remote Engine	Not paired	floras_flowers

Copy the **Remote Engine key** and use it when installing your Remote Engine in the next steps.



Name
floras_RE

Description
Flora's Flowers Remote Engine

Remote Engine key
B77[redacted]9

First, log in to your Linux server and start the **development** Remote Engine.

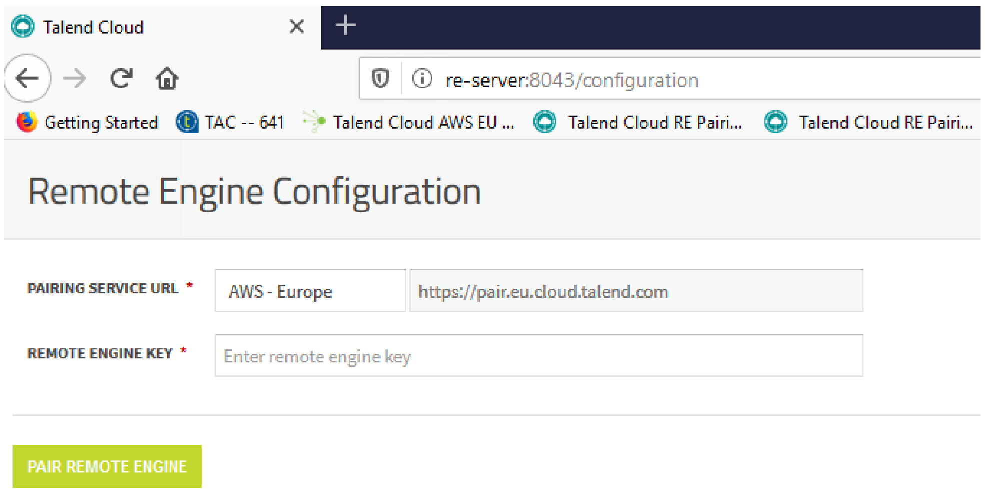
```
$ sudo systemctl start Talend-Remote-Engine-dev
```

Wait a few minutes for it to finish.

Open a browser on your Windows server, and go to the **development** Remote Engine pairing URL:

```
Development:  http://re-server:8043/configuration
```

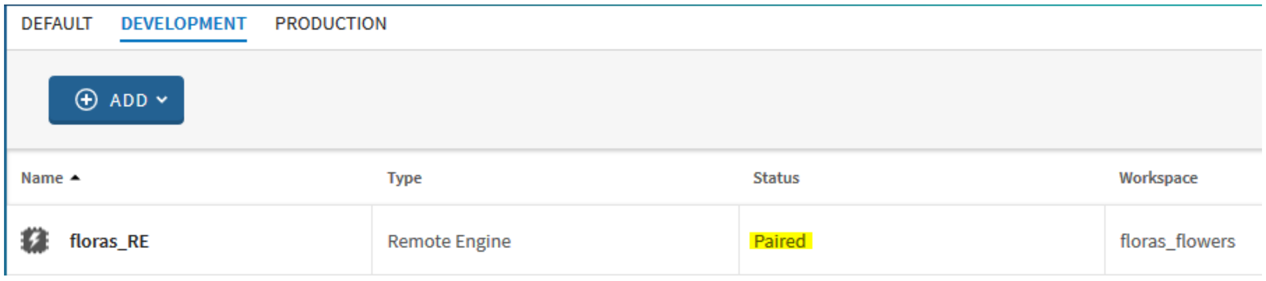
On the page that displays, select your pairing service URL. This matches the Talend Cloud data center where your Talend Cloud account is provisioned. Paste the pairing key you copied above and click the **Pair Remote Engine** link.



Remote Engine Configuration

PAIRING SERVICE URL *	AWS - Europe	https://pair.eu.cloud.talend.com
REMOTE ENGINE KEY *	Enter remote engine key	

After your Remote Engine is installed, running, and successfully paired, the Engines page in the Management Console shows that the Remote Engine status as **Paired**.



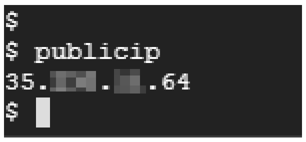
Name	Type	Status	Workspace
floras_RE	Remote Engine	Paired	floras_flowers

Debugging Jobs in Studio using Remote Engines

Enable the Remote Engine to run Jobs directly from Talend Studio for debugging purposes, just like an on-premises JobServer.

From the **Engines** page in the **Management Console**, click the Remote Engine name to display the **Engine Details** dialog. Enable the slider for **Use Remote Engine for debugging Jobs in Studio** and supply the public IP address of the server where your Remote Engine is installed.

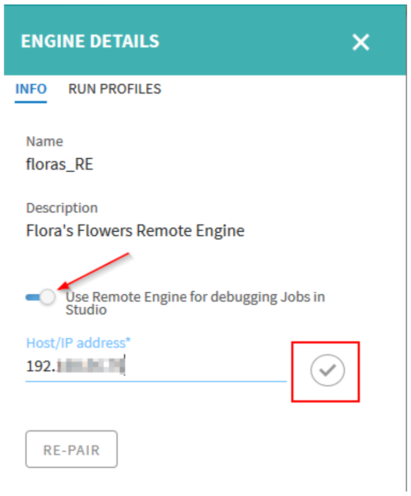
NOTE: To find the public IP address of your Linux server, log in to the Linux server and execute the **publicip** command:



```
$
$ publicip
35.[redacted].[redacted].64
$
```

After you add the public IP address of your development Remote Engine, click the **checkmark** in the circle to confirm. If everything is correct, Talend Cloud displays a dialog stating **Update Successful**.

NOTE: Talend does not recommend enabling production Remote Engines to run Jobs in Studio.



Name
floras_RE

Description
Flora's Flowers Remote Engine

Use Remote Engine for debugging Jobs in Studio

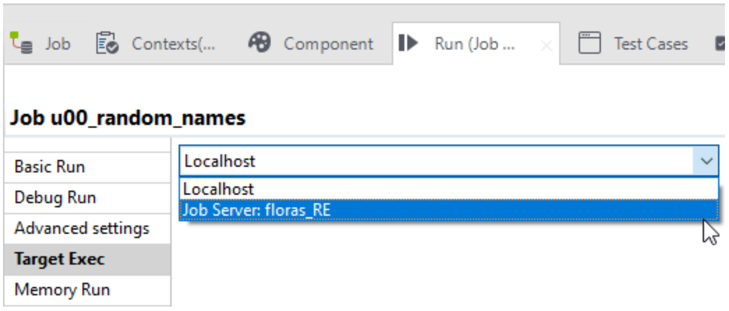
Host/IP address*
192.[redacted]

Testing a Job on a Remote Engine

This section familiarizes you with the end-to-end process of publishing a Job from Talend Studio to Talend Cloud, creating a task for that Job in the Management Console, orchestrating the task, and then running it and viewing the output.

Start Talend Studio 7.3.1 and log in to your new Talend Cloud project. Next, navigate to the **U_Utility** folder and open the Job named **u00_random_names**. This Job produces a set of random first and last names and sends them to a **tLogRow** component for output.

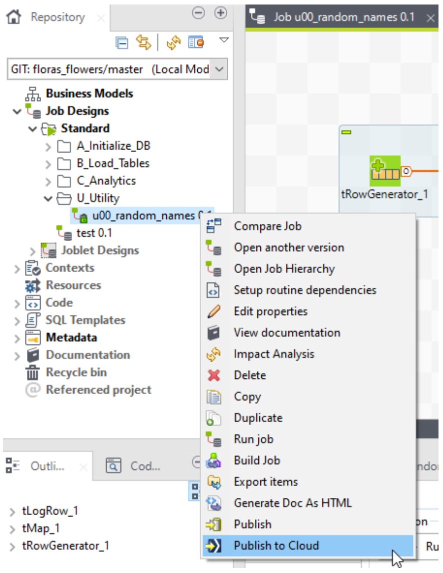
First, run this Job in Studio on Flora's development Remote Engine. To do this, go to the **Run** tab for the Job, then select **Target Exec** from the menu on the left. Select the **development** Remote Engine you configured for remote access in the section on Use Remote Engine for Debugging Jobs in Studio from the drop-down list.



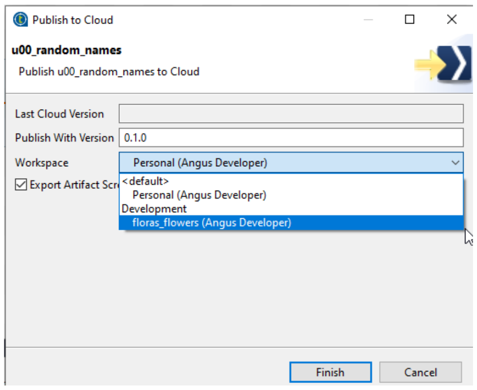
Return to the **Basic Run** tab and run the Job. The compiled Job is uploaded to the Remote Engine and runs. The output displays on the Studio output console. This is a good way to test Jobs before you publish them to Talend Cloud. Although this Job is simple, you can test more complex Jobs with the database I/O on the development Remote Engine before publishing them to Talend Cloud. This saves some time in your testing cycle.

Publishing your first Job to Talend Cloud

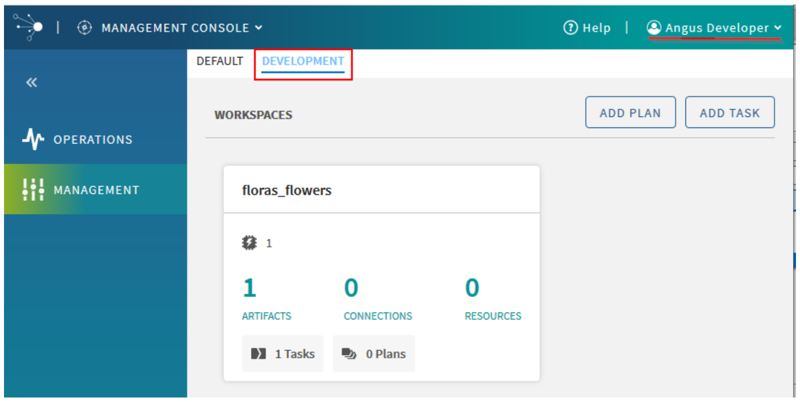
To publish your Job to Talend Cloud, right-click the Job name in the project repository and select **Publish to Cloud**.



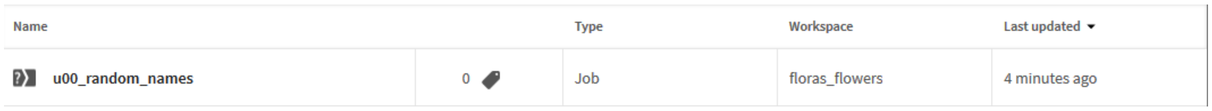
Because this is the first time, this Job is published the **Last Cloud Version** field is blank. The **Publish with Version** field is initialized to **0.1.0**. Select the workspace associated with the **Development** environment rather than the default personal workspace. Be sure the **Export Artifact Screenshot** check box is selected and click **Finish** to publish the Job to Talend Cloud.



Log in to Talend Cloud as your developer user. Access the **Management Console** app and go to the **Management** menu. Select the **Development** environment. Notice that there is one artifact and one task in **floras_flowers** workspace in the Development environment.

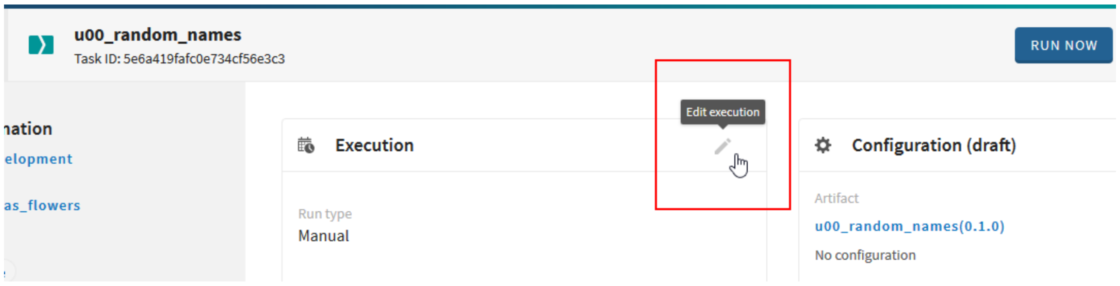


Click the **Tasks** link to display the tasks list.



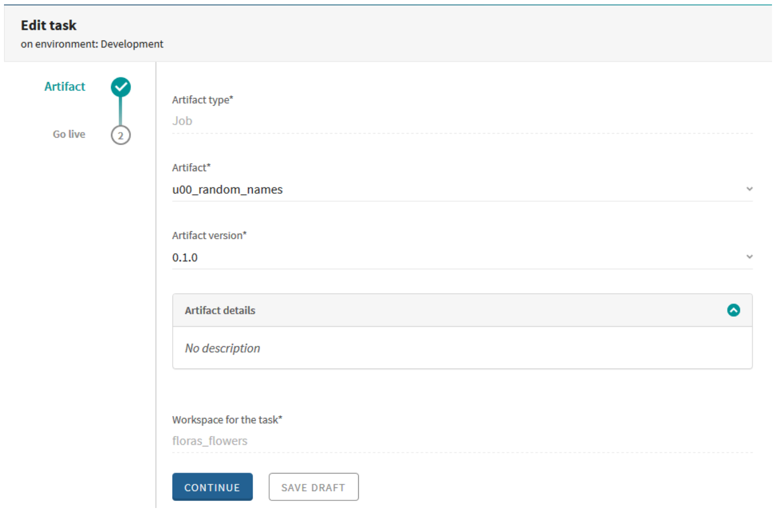
Name		Type	Workspace	Last updated
u00_random_names	0	Job	floras_flowers	4 minutes ago

Click the name of the task to display the task configuration page.



Hover your mouse near the word **Execution** to display a **pencil** icon. Click the icon to configure the execution parameters.

On the first task configuration page, you can change the artifact and the artifact version. This isn't necessary in this use case, so click **Continue**.



Edit task
on environment: Development

Artifact
Go live

Artifact type*
Job

Artifact*
u00_random_names

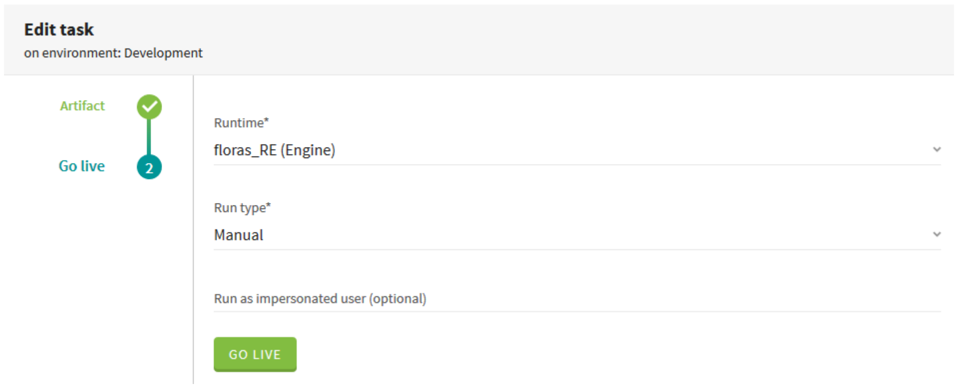
Artifact version*
0.1.0

Artifact details
No description

Workspace for the task*
floras_flowers

CONTINUE   SAVE DRAFT

On the Go live page, you can select the runtime. Click the **Runtime** drop-down and select Flora's **development** Remote Engine. Leave the **Run type** set to **Manual**.

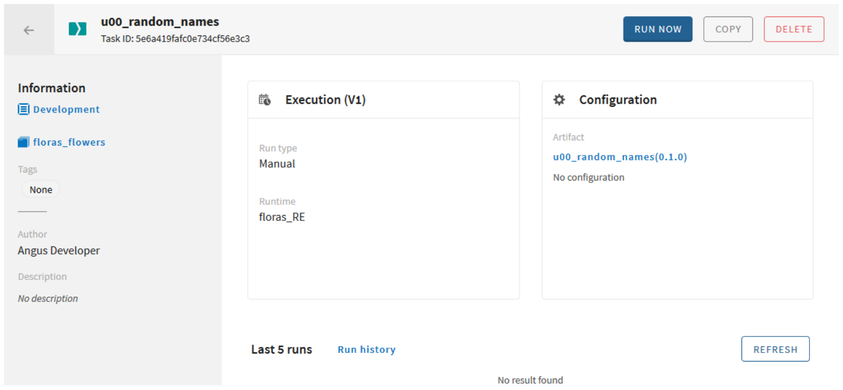

Edit task
on environment: Development

Runtime*
floras_RE (Engine)

Run type*
Manual

Run as impersonated user (optional)

Click **Go Live** to see the task summary screen.


u00_random_names
Task ID: 5e6a419fafc0e734cf56e3c3

Information
Development
floras_flowers
Tags
None
Author
Angus Developer
Description
No description

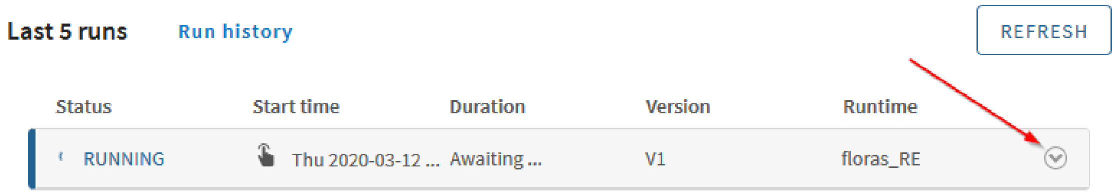
Execution (V1)
Run type
Manual
Runtime
floras_RE

Configuration
Artifact
u00_random_names(0.1.0)
No configuration

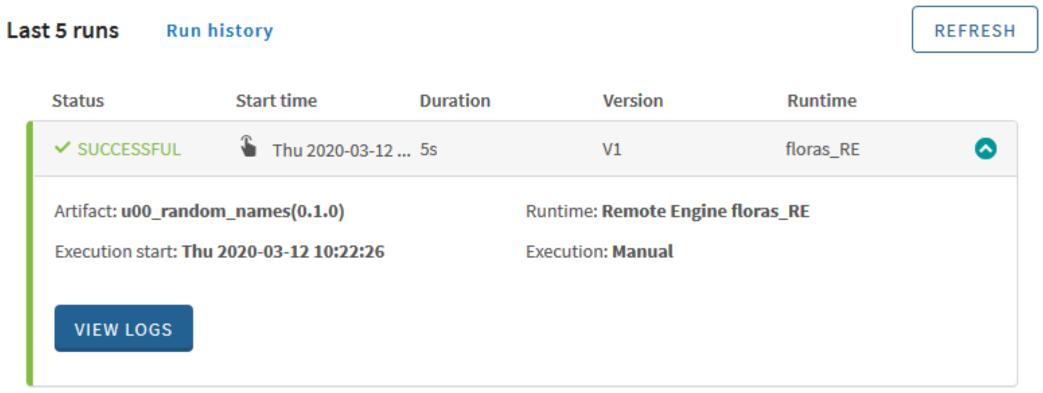
Last 5 runs Run history

No result found

Click **Run Now** to run the Job. Click the down arrow to view the run progress and to access the logs. You can click the **Refresh** button to update the run status.


Last 5 runs Run history

Status	Start time	Duration	Version	Runtime
RUNNING	Thu 2020-03-12 ...	Awaiting ...	V1	floras_RE


Last 5 runs Run history

Status	Start time	Duration	Version	Runtime
✓ SUCCESSFUL	Thu 2020-03-12 ...	5s	V1	floras_RE

Artifact: **u00_random_names(0.1.0)**
Runtime: **Remote Engine floras_RE**
Execution start: **Thu 2020-03-12 10:22:26**
Execution: **Manual**

Click **View Logs** to see the Job output. The Job output is formatted differently than when the Job is run in Studio, but the data generated by the Job is shown.

Talend Job Migration Preparation

Flora's Talend Jobs were built by a single in-house developer. They are well organized but could be improved in certain areas. You need to make changes as necessary to migrate them successfully to Talend Cloud while following general migration best practices.

In Studio, you can see that Flora's Jobs are organized in folders as follows:

Folder Name	Description
A_Initialize_DB	This folder contains a single Job that initializes a MySQL DB schema. It is required for this use case.
B_Load_Tables	This folder contains four Jobs: a master Job and three subJobs that perform an initial load of Flora's flower customer and sales data tables.
C_Analytics	This folder contains one master Job and one subJob. The subJob calculates the analysis of monthly flower sales.
U_Utility	This folder contains Jobs that are used for test or demonstration purposes only.
Joblet	This folder contains a basic Joblet that is used as a common error handling module.

Look at each Job and review the context parameters. Notice that there isn't much consistency with context parameters between Jobs, even though Jobs commonly read or write to the same two databases.

Migrating the DB Initialize Jobs

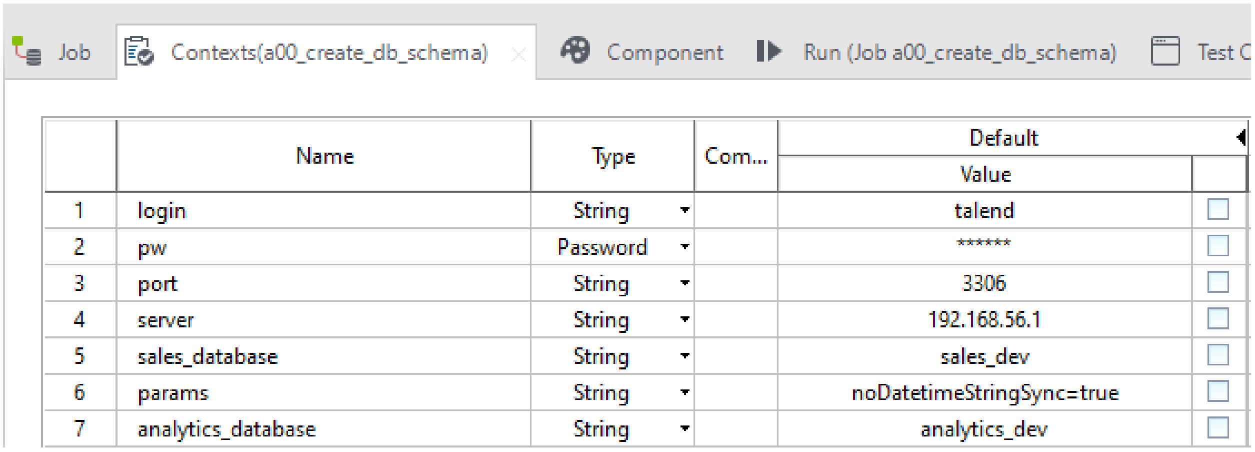
In Talend Studio 7.3.1, open the **a00_create_db_schema** Job in the **A_Initalize_DB** folder. Review the Job, components, and flow.

The Job does the following:

- Connects to your MySQL database based on context connection parameters
- Creates a Sales database based on context values if it does not already exist
- Creates an Analytics database based on context values if it does not already exist
- Notifies the user that the Job is complete, and all databases are created successfully using a standard Java **System.out.println** message.
- Traps errors from all significant components using **on component error** triggers and a **tDie** component. A Joblet catches these errors and sends output to two different logs (in this use case, it is the same output: **tLogRow**).

Review the context variables for this Job.

NOTE: Context values in your Talend Academy sandbox may be different.


Job
Contexts(a00_create_db_schema)
Component
Run (Job a00_create_db_schema)
Test C

	Name	Type	Com...	Default Value	
1	login	String		talend	☐
2	pw	Password		******	☐
3	port	String		3306	☐
4	server	String		192.168.56.1	☐
5	sales_database	String		sales_dev	☐
6	params	String		noDatetimeStringSync=true	☐
7	analytics_database	String		analytics_dev	☐

These primarily are concerned with a single connection to a MySQL database with two data schema names for the sales and analytics databases. The context parameters do not conform to the Talend Cloud connection parameter naming standard. If you migrate the Job as it is, all context variables move to Talend Cloud as local task advanced parameters.

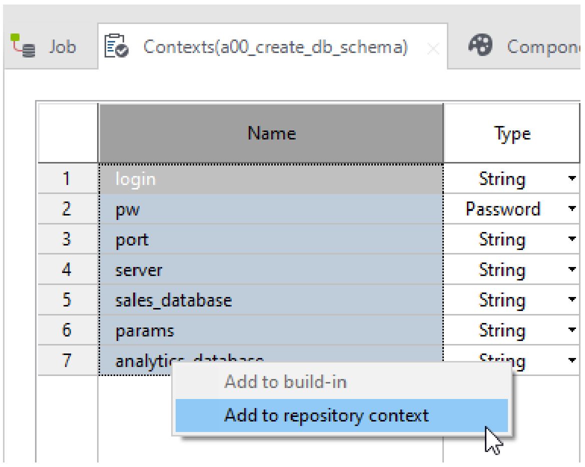
There are three benefits to using the Talend Cloud naming standard for connection parameters:

- **Password encryption**: Password values can only be hidden for context variables that conform to the connection parameter naming convention.
- **Sharing parameters across environments**: The parameters themselves can be shared across Talend Cloud environments, but not the values. Each environment can have a different set of values germane to that environment.
- **Sharing parameters across tasks**: Connection parameters that conform to the naming convention can be shared across Talend Management Console tasks.

Later you'll see other Jobs use the same connection parameters. These connection parameters are reusable across Jobs and environments.

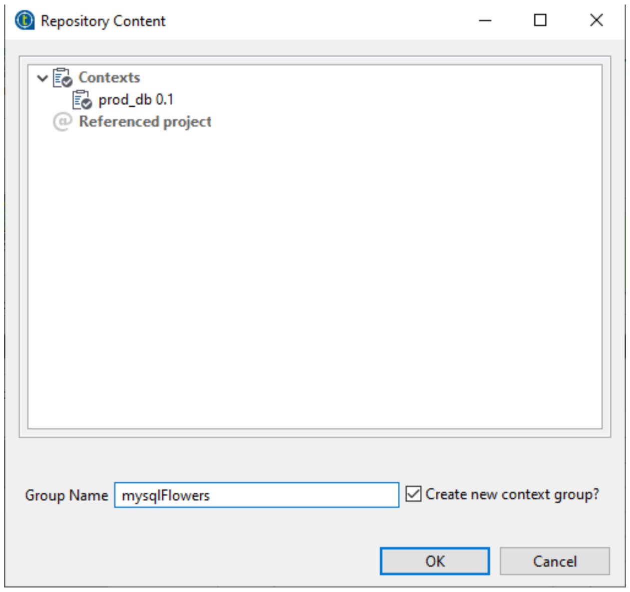
The following process prepares these context variables for migration:

First, add the context variables to the context repository. Select all context variables in the Job, then right-click the selected context variables, and select **Add to repository context**.



	Name	Type
1	login	String
2	pw	Password
3	port	String
4	server	String
5	sales_database	String
6	params	String
7	analytics_database	String

Give the context group a meaningful name that combines the purpose with the database technology, for example, *mysqlFlowers*.



Group Name: mysqlFlowers

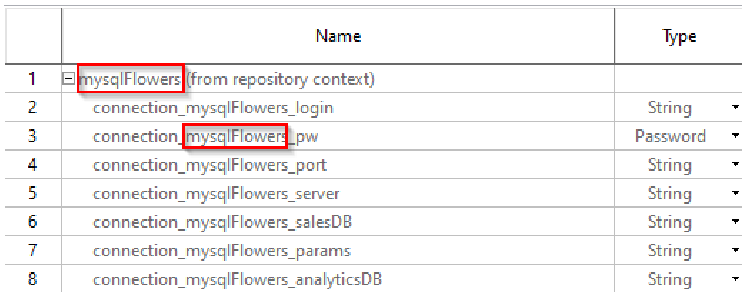
☑ Create new context group?

After you save the contexts in the repository, rename each context variable to conform to Talend Cloud connection parameter naming convention:

connection_<application_name>_<parameter_name>

where

connection	A fixed prefix
application name	The name of the system you want to connect to (which matches the context repository group name)
parameter_name	The variable name of the connection parameter



	Name	Type
1	mysqlFlowers (from repository context)	
2	connection_mysqlFlowers_login	String
3	connection_mysqlFlowers_pw	Password
4	connection_mysqlFlowers_port	String
5	connection_mysqlFlowers_server	String
6	connection_mysqlFlowers_salesDB	String
7	connection_mysqlFlowers_params	String
8	connection_mysqlFlowers_analyticsDB	String

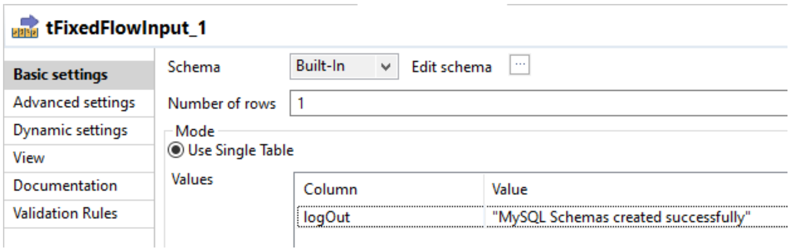
After you rename and save the new context parameters in the context repository, the contexts used in the Job are automatically updated.

Logging

The Job uses a Java **System.out.println** standard utility method to notify the user that all databases were created successfully, and the Job is complete. These standard output routines are problematic in Talend Cloud because the output is not directed to the Talend Cloud logs.

To preserve this functionality, you need to replace the **tJava** component that contains this line of Java code with something that works with Talend Cloud. In this use case, you use two other components: **tFixedFlowInput** and **tLogRow**. Add one of each of these to the **a00_create_db_schema** Job near the existing **tJava** component.

Edit the **tFixedFlowInput** component and add one column to its schema. Then copy and paste the output text from the **tJava** component to the column value in the **tFixedFlowInput** component.



tFixedFlowInput_1

Basic settings | Advanced settings | Dynamic settings | View | Documentation | Validation Rules

Schema: Built-In — Edit schema

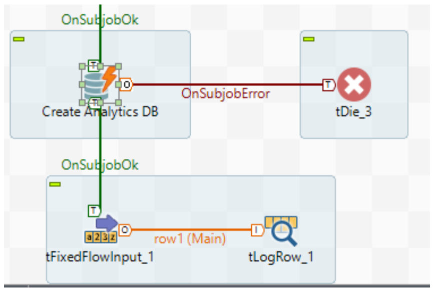
Number of rows: 1

Mode: ● Use Single Table

Values

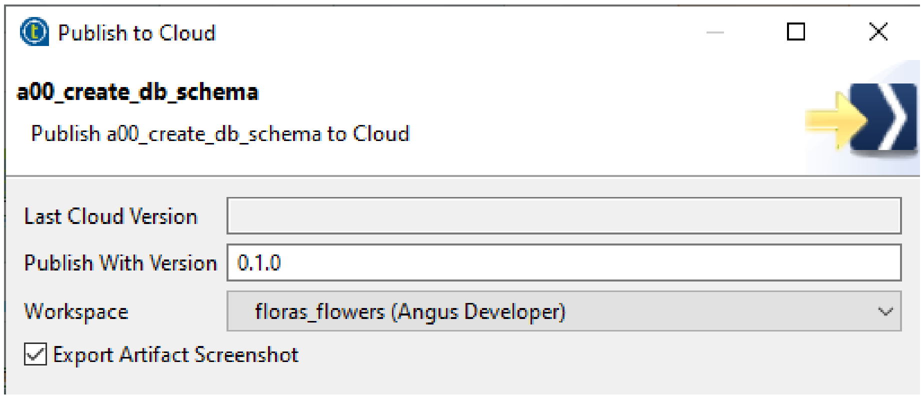
Column	Value
logOut	"MySQL Schemas created successfully"

Connect the main row from **tFixedFlowInput** to the new **tLogRow** component. Delete the **tJava** component from the Job. Now, connect the **tFixedFlowInput** component to the **Create Analytics DB tMySQLRow_2** component on the **OnSubjob OK** trigger. Save the Job.



Publishing to Talend Cloud

Right-click the Job name in the project repository and select **Publish to Cloud**, just like you did in the Publish your first Job to Talend Cloud section. Be sure you publish to the shared project workspace in the Development environment.


Publish to Cloud

a00_create_db_schema

Publish a00_create_db_schema to Cloud

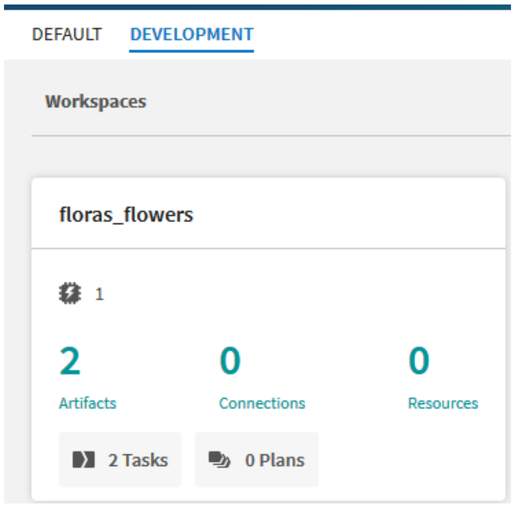
Last Cloud Version:

Publish With Version: 0.1.0

Workspace: floras_flowers (Angus Developer)

☑ Export Artifact Screenshot

Log in to Talend Cloud as the developer user. From the **Management** menu of the **Management Console**, you now have two artifacts and two tasks in the shared project workspace in the Development environment.


DEFAULT DEVELOPMENT

Workspaces

floras_flowers

1

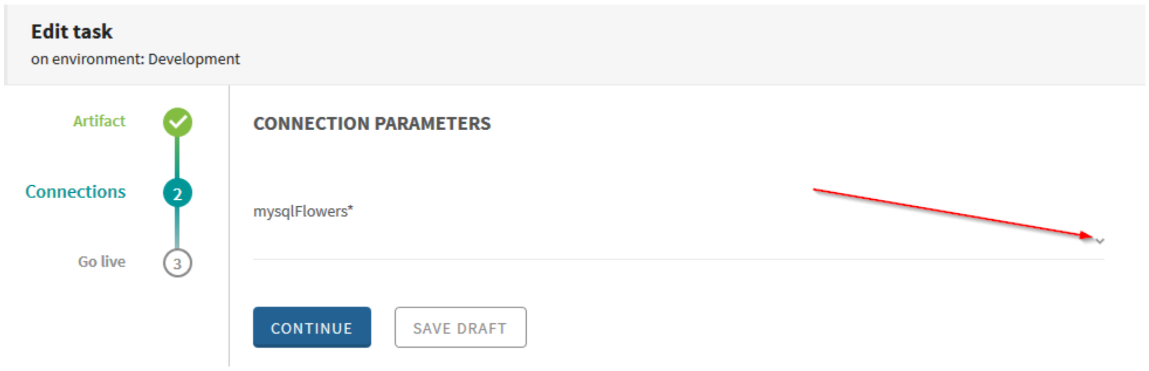
2 Artifacts 0 Connections 0 Resources

2 Tasks 0 Plans

Click the **Tasks** link, then click the **a00_create_db_schema** task to deploy. Click the **pencil** icon (hover your mouse near the word **Execution** to find it).

Different from the initial task you deployed, there are now three pages to navigate. The second page is concerned with connection parameters. Because you changed the Talend Studio Jobs context variables to conform with Talend Cloud connection parameter naming format, you set the values for these parameters once per environment.

Click the drop-down for the Connections Parameters, **mysqlFlowers**:


Edit task

on environment: Development

Artifact

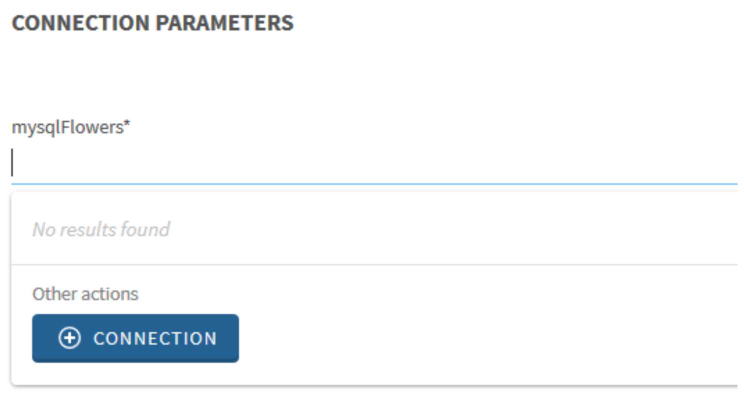
Connections

Go live

CONNECTION PARAMETERS

mysqlFlowers*

Then click **+ Connection**.

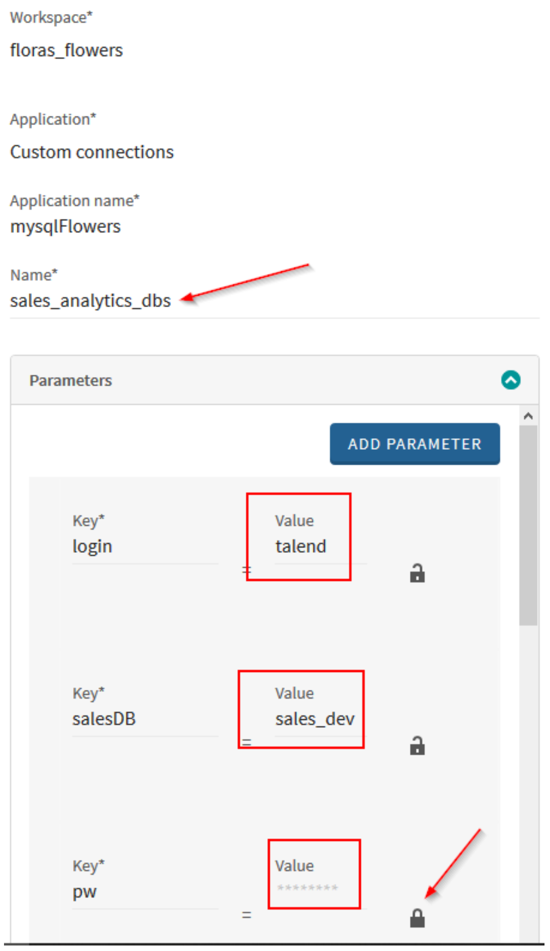

CONNECTION PARAMETERS

mysqlFlowers*

No results found

Other actions

Enter an environment-specific name for this set of connection parameter values and add the values to the parameters themselves.


Workspace*
floras_flowers

Application*
Custom connections

Application name*
mysqlFlowers

Name*
sales_analytics_dbs

Parameters

Key*		Value
login	=	talend
salesDB	=	sales_dev
pw	=	********

To hide parameter values like passwords, click the **lock** icon after entering a value.

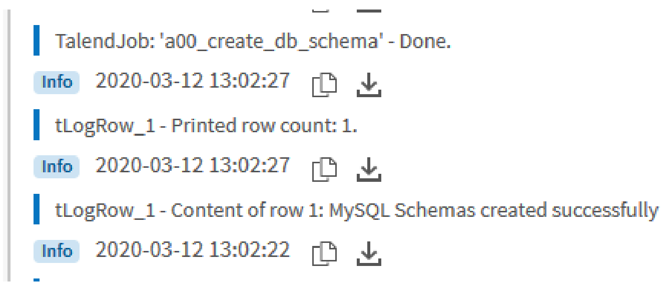
After adding all the connection parameter values, click **Save**.

Click **Continue** when you arrive back on the **Connection Parameters** page.

On the final task configuration page, select the **development** Remote Engine, and click **Go Live**.

On the **Task run** page, click **Run Now**.

- As before, you can click **Refresh** to view the status of the Job and click the down arrow to view the logs. When the Job finishes, examine the task log.



```
TalendJob: 'a00_create_db_schema' - Done.
Info 2020-03-12 13:02:27
tLogRow_1 - Printed row count: 1.
Info 2020-03-12 13:02:27
tLogRow_1 - Content of row 1: MySQL Schemas created successfully
Info 2020-03-12 13:02:22
```

Somewhere in the log, you should see the final message if the Job was successful, or you may see an error message if the Job failed.

You can view the data schemas as created by starting the MySQL Workbench on your Windows server and connecting to the MySQL database.

Migrating the Load Tables Jobs to Talend Cloud

In the **B_Load_Tables** folder, there are four Jobs, a master Job that calls three subJobs. The purpose of these Jobs is to perform an initial load on Flora's Flowers data tables in the sales and analytics databases.

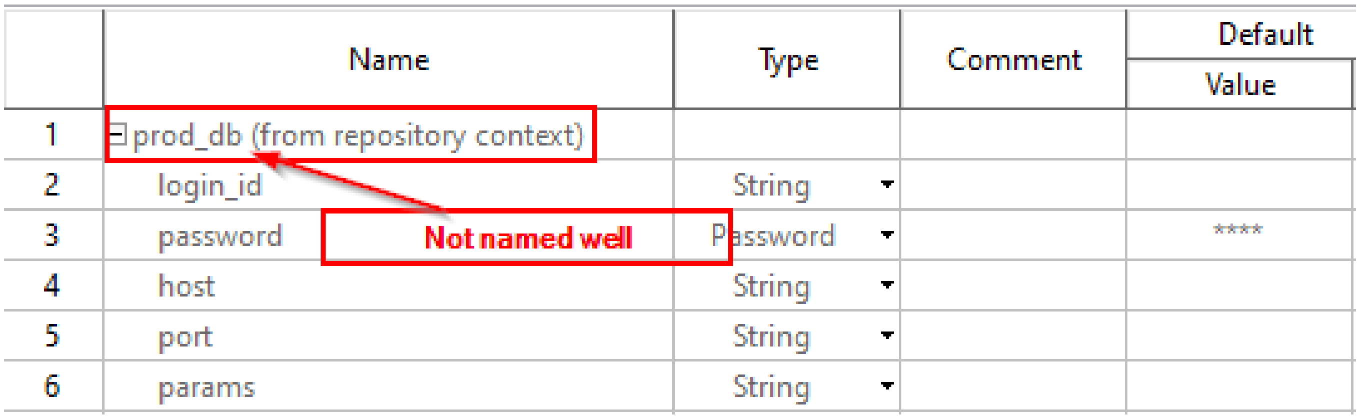
The master Job, **a00_load_tables_master**, loads context variables in key-value pairs from a CSV file in the filesystem that is local to the JobServers. There are two versions of the CSV file, one that connects to the development data server and one that connects to the production data server (for your purposes, it is the same database server).

Context variables are explicitly loaded at runtime in the master Job using a **tContextLoad** component. These are then shared with each subJob. Contexts are not standardized to any naming standard, nor are they harmonized among all the subJobs called by the master. It is <u>not necessary</u> to rename explicit or implicit context variables for a successful migration to Talend Cloud.

In this section, you leave the contexts as they are and migrate the Jobs to Talend Cloud by doing the following:

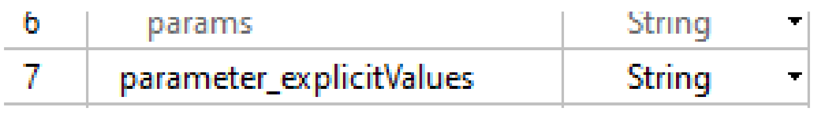
Review the master and each subJob in the **B_Load_Tables** folder.

- Notice that context variables are not harmonized or standardized in any of the Jobs.



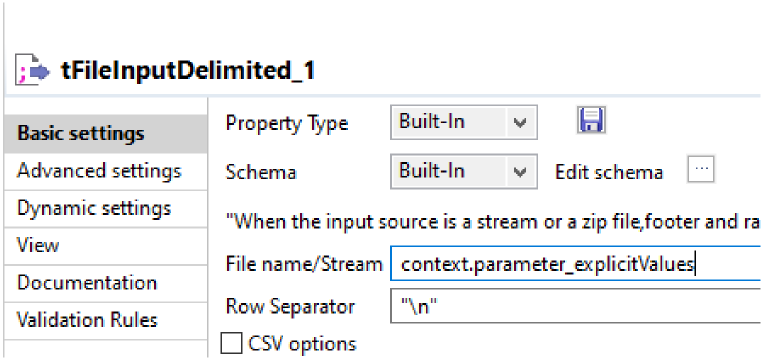
	Name	Type	Comment	Default Value
1	⊟prod_db (from repository context)			
2	login_id	String		
3	password	Password		****
4	host	String		
5	port	String		
6	params	String		

- Add a context variable called **parameter_explicitValues** to the master Job. Replace the hard-coded file path and name in the **tContextLoad** component with this context variable.



6	params	String
7	parameter_explicitValues	String

It is not necessary to provide a value for this parameter in Studio. You'll pass the context filename and path to the task in the Talend Management Console.



tFileInputDelimited_1

Basic settings | Advanced settings | Dynamic settings | View | Documentation | Validation Rules

Property Type: Built-In

Schema: Built-In — Edit schema

"When the input source is a stream or a zip file,footer and ra

File name/Stream: context.parameter_explicitValues

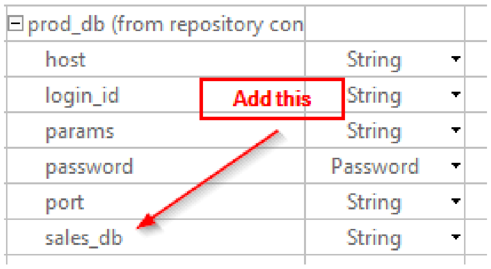
Row Separator: "\n"

CSV options: ☐

- In each subJob, no changes are necessary. There may be an unused context variable **new1** in the **b01** subJob. It can be ignored.

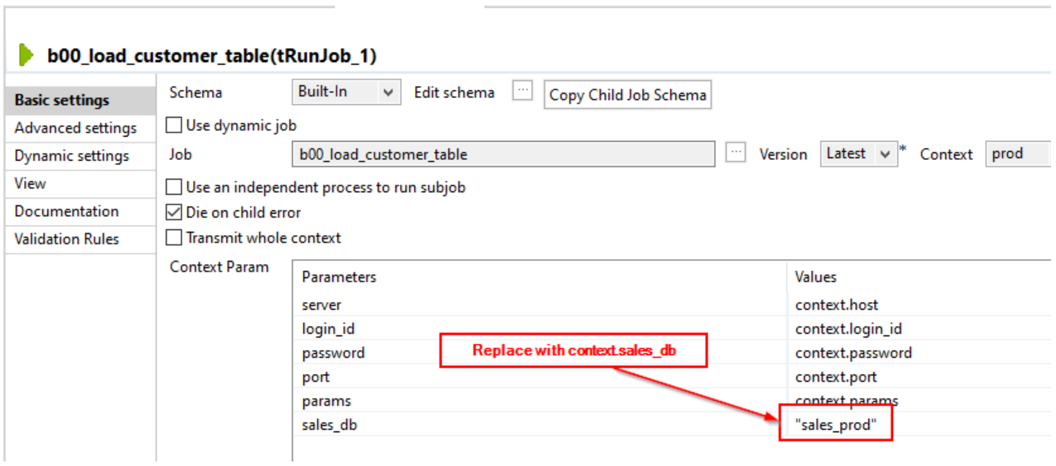
The master hard codes the database to **sales_prod** in the **tRunJob** components to access the production sales database. This is not a good practice, to say the least, so you need to fix it.

- Add a database parameter to the context group.



⊟ prod_db (from repository con	
host	String
login_id	String
params	String
password	Password
port	String
sales_db	String

- For each **tRunJob** component in the master Job, remove the hard-coded database parameter and replace it with the **sales_db** context parameter.



b00_load_customer_table(tRunJob_1)

Basic settings | Advanced settings | Dynamic settings | View | Documentation | Validation Rules

Schema: Built-In — Edit schema — Copy Child Job Schema

☐ Use dynamic job

Job: b00_load_customer_table — Version: Latest — Context: prod

☐ Use an independent process to run subjob

☑ Die on child error

☐ Transmit whole context

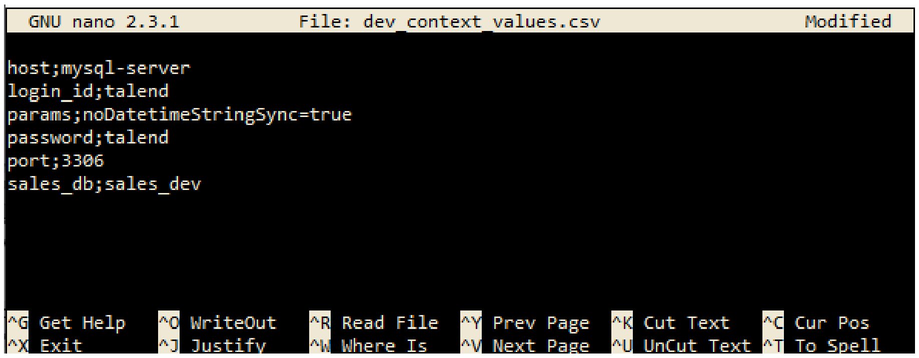
Context Param

Parameters	Values
server	context.host
login_id	context.login_id
password	context.password
port	context.port
params	context.params
sales_db	"sales_prod"

- To preserve the explicit loading of the context values from the CSV file. You need to find both the development and production versions of the context values CSV file and edit it to verify the context keys and values. The CSV files can be found in the **/var/tmp** directory on your Linux server:

```
/var/tmp/dev_context_values.csv
/var/tmp/prod_context_values.csv
```

- Edit each file on the Linux server and double-check your context values and keys are correct:

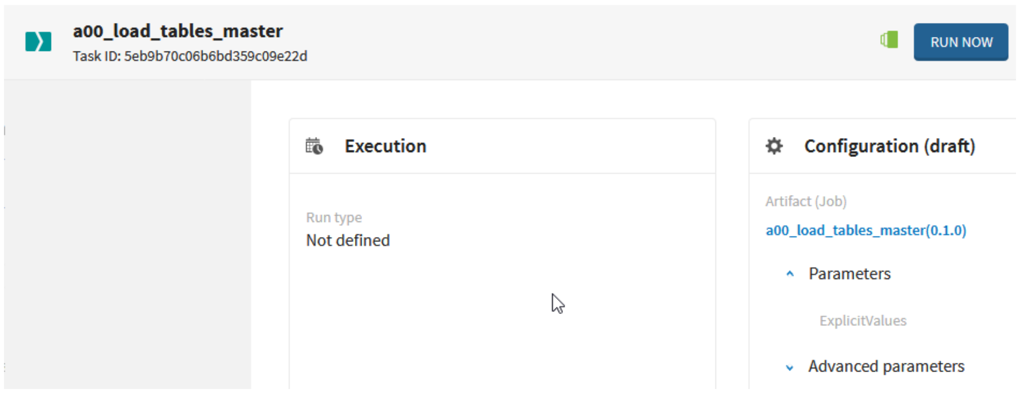


```
  GNU nano 2.3.1            File: dev_context_values.csv                Modified

host;mysql-server
login_id;talend
params;noDatetimeStringSync=true
password;talend
port;3306
sales_db;sales_dev




^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

Remember, the sales DB for production is **sales_prod** and **sales_dev** for development.

When you're finished, you can test the master Job in Studio by running it in the Remote Engine that is set up as a JobServer. Double-check the path to the CSV file to be sure you are pointing to the development version of the file.

As the compiled master Job will include object code for all the subJobs, you only need to publish the master Job to Talend Cloud. Remember to publish to the development environment.



a00_load_tables_master
Task ID: 5eb9b70c06b6bd359c09e22d

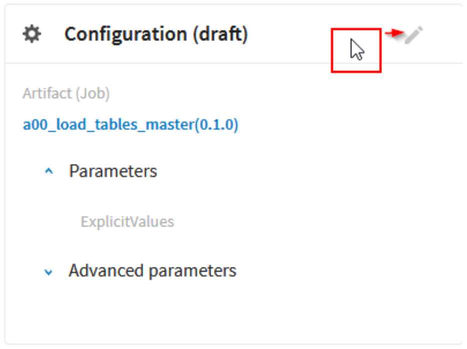
Run type
Not defined

Artifact (Job)
a00_load_tables_master(0.1.0)

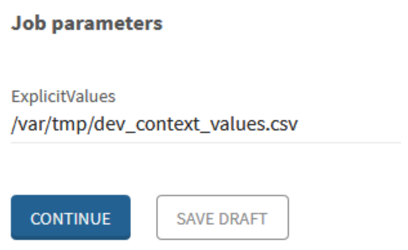
Notice that this Job has Talend Cloud Parameters and Advanced Parameters. There is only one parameter here, and it is from the explicit context values parameter we added. Next, you supply a value for this parameter in the Talend Management Console, so the Job may find its context parameter values.

Talend Cloud Advanced Parameters here include all the database parameters and are local to the task and are not shared with other Jobs or environments. Values for these are loaded explicitly at runtime; you will not supply values in the Talend Management Console.

Assign the explicit values parameter value (and only that one) created previously by clicking the **pencil** icon that displays when hovering the mouse pointer near the word **Configuration** on the right side of the page.

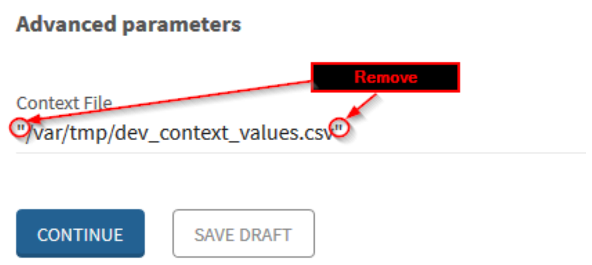


Enter a path and filename for the **ExplicitValues** parameter and click **Continue**.


Job parameters

ExplicitValues
/var/tmp/dev_context_values.csv

NOTE: Double quoted strings are not acceptable to Talend Cloud but work in Studio and Talend Administration Center. When you set the value for the CSV file, be sure the string is unquoted.

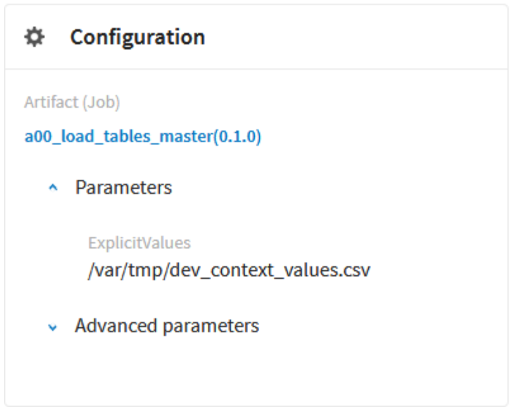

Advanced parameters

Context File
"/var/tmp/dev_context_values.csv"

When the ExplicitValues parameter is set correctly, click **Continue**. Do not set any values for the database connection parameters.

Assign the task to the **development** Remote Engine and click **Go Live**.

Now back on the main **Task** page, you have set a path and filename for the explicit context variable CSV file. The data in this file supplies your master and subJobs with context values at runtime.


Configuration

Artifact (Job)
a00_load_tables_master(0.1.0)

Parameters

ExplicitValues
/var/tmp/dev_context_values.csv

Advanced parameters

Click **Run Now** and view the logs when the Job completes. View the row values in the development sales database in your MySQL Workbench utility.

Deploy Load Tables as a Talend Cloud Plan (Optional)

To decouple the load table Jobs from the master and each other, publish each subJob separately to Talend Cloud. Create a plan in Talend Management Console to run each subJob and link them all together to run consecutively. You do not need to include the master Job in the plan. Assign it to the **development** Remote Engine and execute the plan.

Migrating the analytics Jobs to Talend Cloud

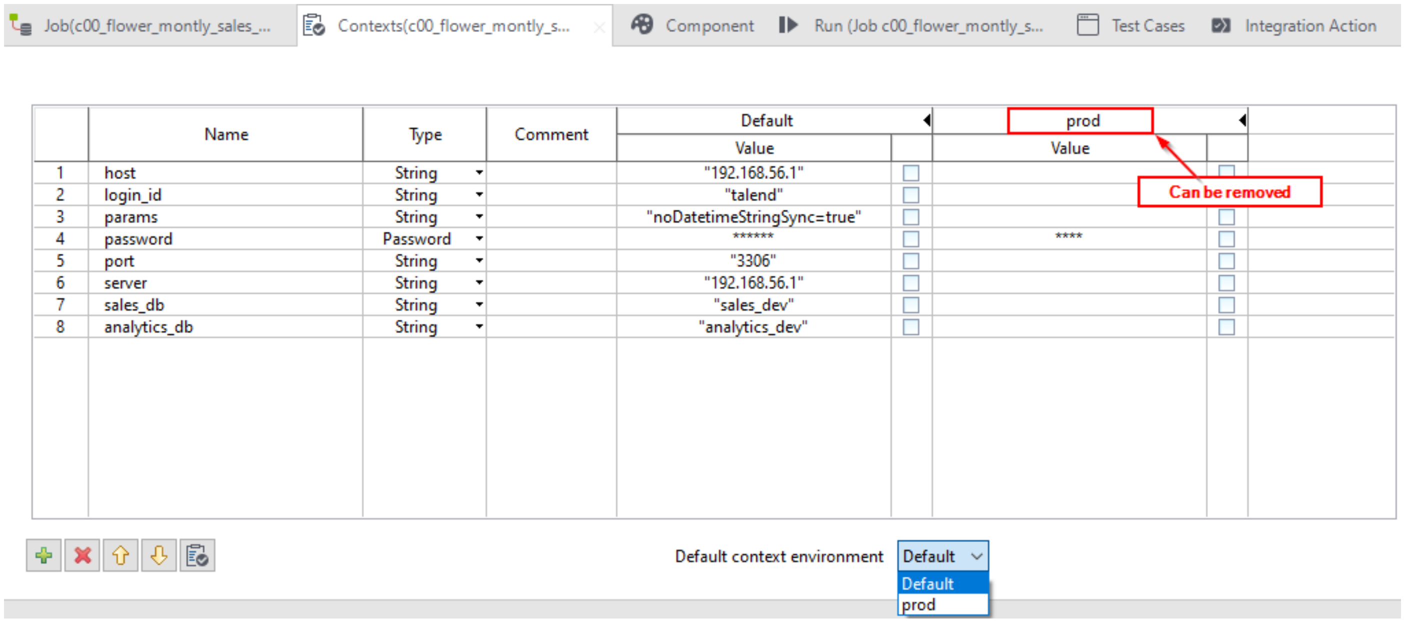
Review the two Jobs in the **C_Analytics** folder. There is one master and one subJob.

Your goal is to standardize the context variables to the connection parameter formatted MySQL set you created in the context repository.

Change the context variables in both the master and the subJob to use the Talend Cloud connection parameter standardized set **mysqlFlowers** you created previously. Remove the old context variables. Don't forget to update the **tRunJob** component in the master Job.

NOTE: In the subJob there are two MySQL connection components that must be updated.

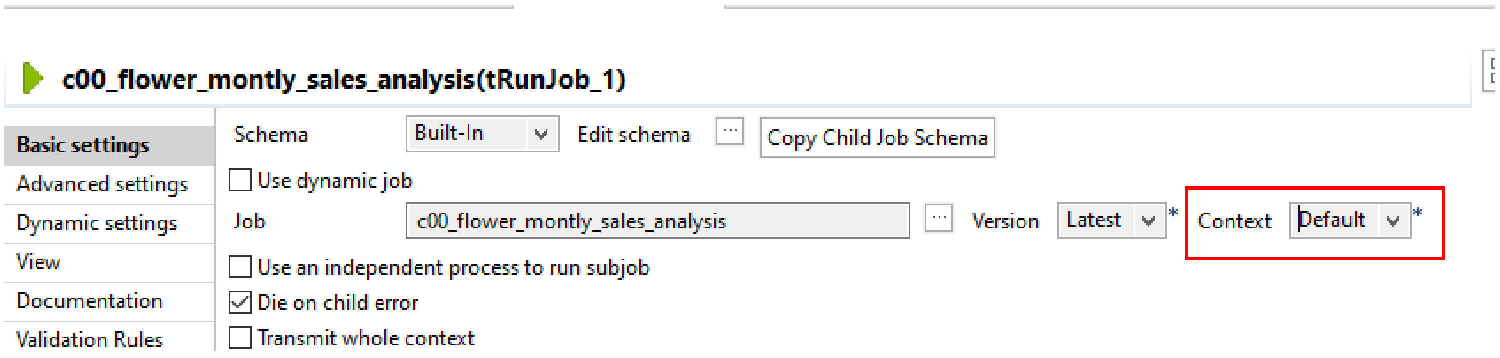
The **c00** subJob has two context environments **Default** and **prod**. Multiple context environments in Studio aren't necessary for use in Talend Cloud. SDLC connection parameters per environment will be handled in the cloud. The Prod context environment should be removed in the **c00** subJob.


Job(c00_flower_montly_sales_...
Contexts(c00_flower_montly_s...
Component
Run (Job c00_flower_montly_s...
Test Cases
Integration Action
Can be removed

	Name	Type	Comment	Default		prod	
				Value		Value	
1	host	String		"192.168.56.1"			
2	login_id	String		"talend"			
3	params	String		"noDatetimeStringSync=true"			
4	password	Password		******		****	
5	port	String		"3306"			
6	server	String		"192.168.56.1"			
7	sales_db	String		"sales_dev"			
8	analytics_db	String		"analytics_dev"			

Default context environment: Default (Default, prod)

Be sure the context in the **c00 flower monthly sales analysis** master Job is set to **Default**.



c00_flower_montly_sales_analysis(tRunJob_1)

Schema: Built-In

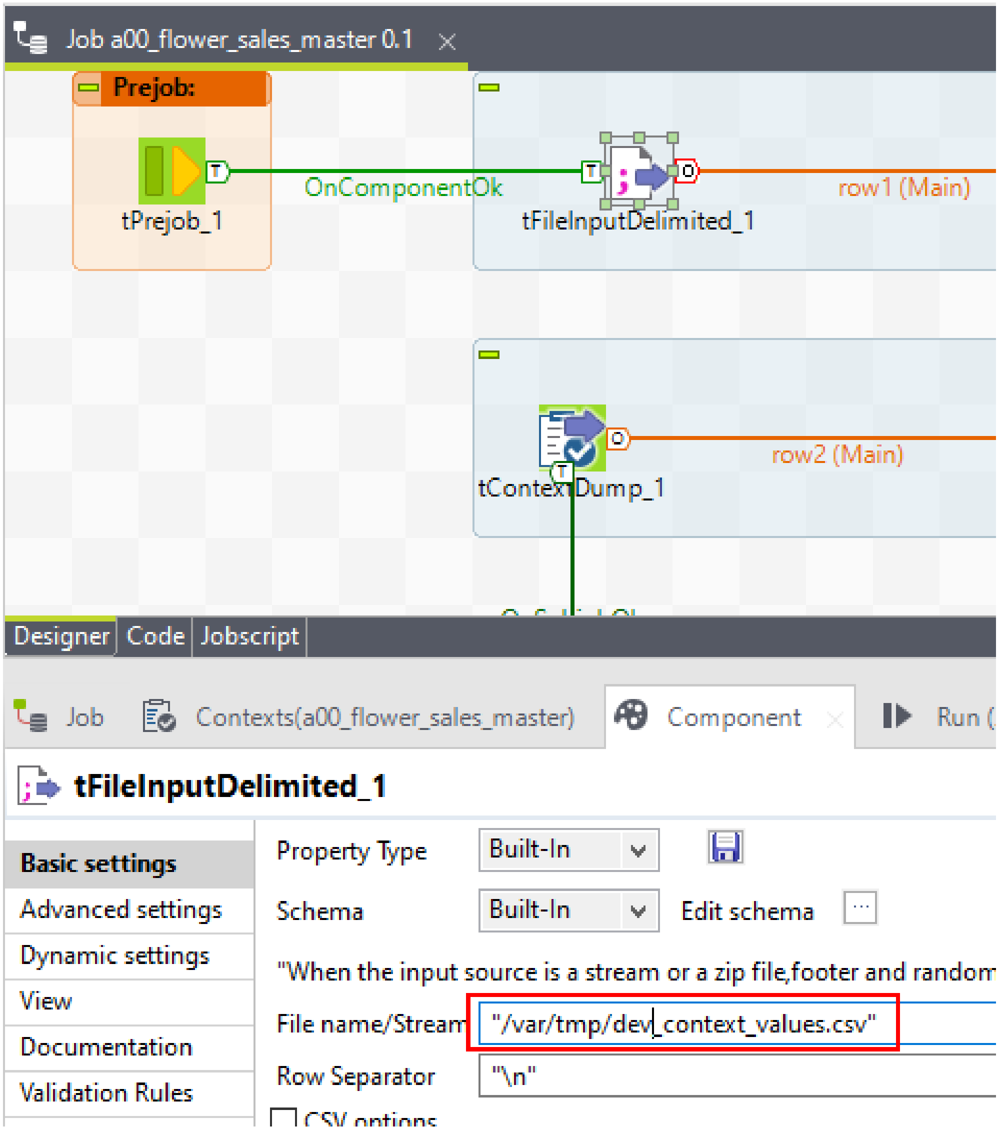
- [ ] Use dynamic job

Job: c00_flower_montly_sales_analysis   Version: Latest   Context: Default

- [ ] Use an independent process to run subjob
- [x] Die on child error
- [ ] Transmit whole context

Notice in the master Job for this folder that the **tFileInputDelimited** component that has the production CSV file hard-coded into the filename parameter. This is not a good practice as production databases and files should never be accessible from a developer's workstation. Of course, this is only a training sandbox, so no harm is done.

Change the hard-coded file value to the context variable **context_file**.



tFileInputDelimited_1

Property Type: Built-In
Schema: Built-In
"When the input source is a stream or a zip file,footer and random
File name/Stream: "/var/tmp/dev_context_values.csv"
Row Separator: "\n"

Publish the Job to the Development environment and workspace in Talend Cloud and deploy. Assign the advanced and connection parameters. Set the context file advanced parameter to the development CSV file, as shown in the red box in the screenshot above. Assign it to the **development** Remote Engine. Run the Job and view the log files.

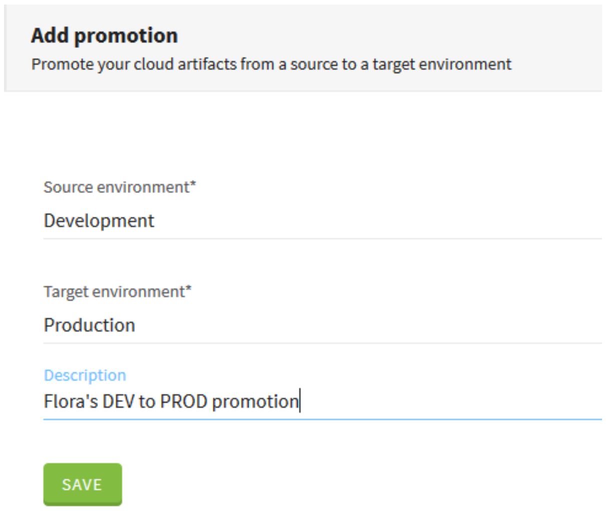
Defining the SDLC

Earlier, you created both a development and a production environment but did not create a process to copy objects from one to the other. In this section, you create a Talend Cloud promotion process to formally copy development artifacts and tasks to your production environment.

Configuring promotions

Login to Talend Cloud as the production user. From the **Management Console**, select the **Promotions** menu, then click **Add Promotion**.

The **Source environment** should be **Development**, and the **Target environment** should be **Production**. Add a meaningful description and click **Save**.



Add promotion

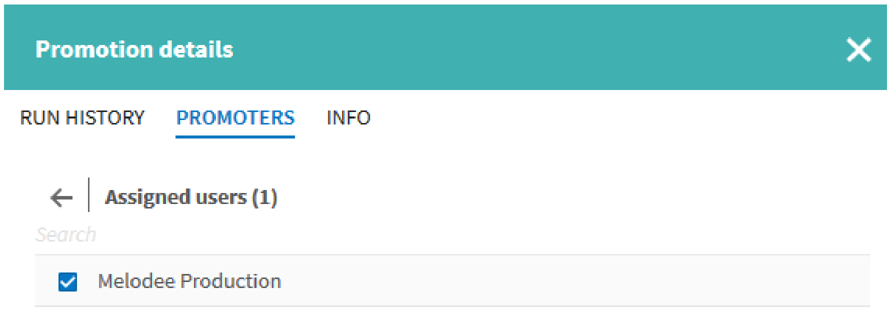
Promote your cloud artifacts from a source to a target environment

Source environment*: Development

Target environment*: Production

Description: Flora's DEV to PROD promotion

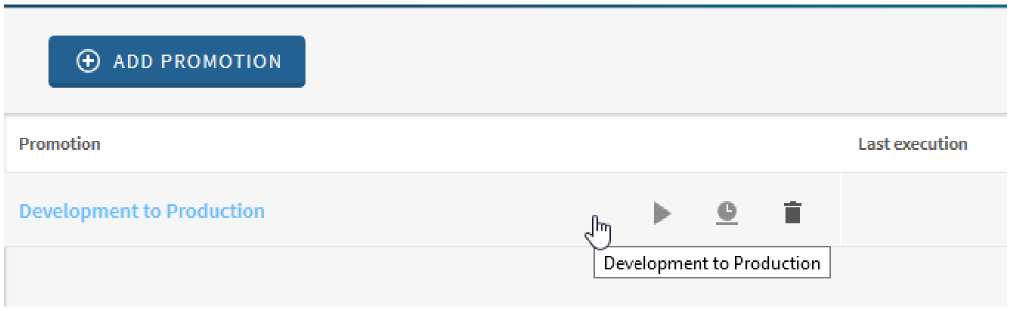
Click the name of the promotion to open the **Promotion details** tab.



☑ Melodee Production

Select the **Promoters** menu, then select your production user from the list. Close **Promotion details**, then hover the mouse in the cell with the promotion name to get the controls to appear.

Click the left **triangle** icon to execute the promotion.

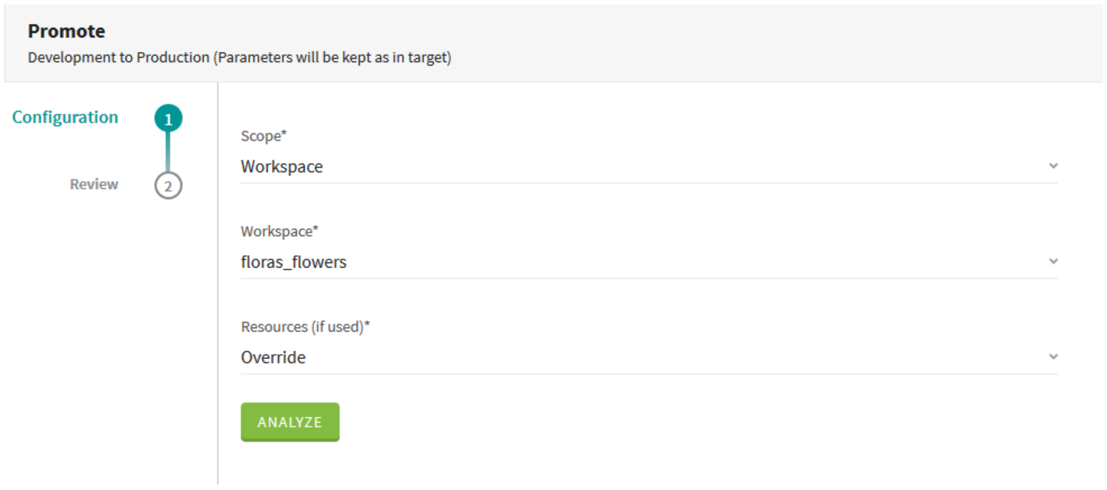


Add the details about the promotion between development and production you want done. Select **Workspace** as the **Scope**, and then add the development project workspace.

The final parameter, **Resources**, can be either Override or Keep target.

Override	Promoted resources take precedence over the target environment's resources.
Keep target	Existing resources are not affected by the promotion.

Select **Override** and click **Analyze**.



Promote
Development to Production (Parameters will be kept as in target)

Configuration 1
Review 2

Scope*
Workspace

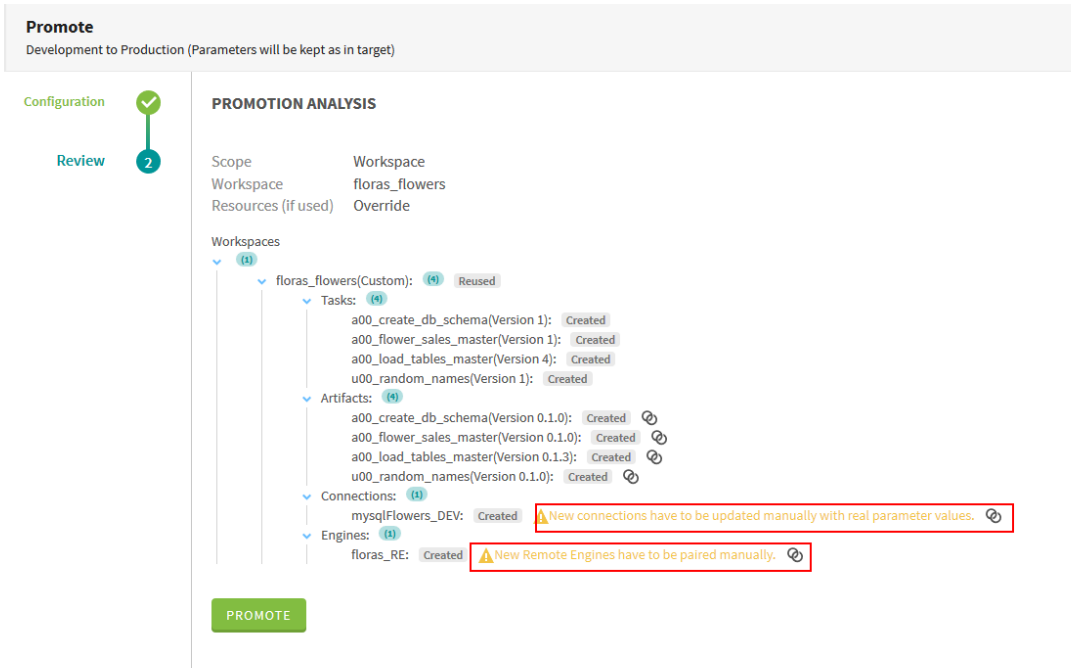
Workspace*
floras_flowers

Resources (if used)*
Override

Analyzing the promotion

The first step in a promotion is to analyze the work to be done. Clicking **Analyze** on a configured promotion displays a page that lists what Talend Cloud will do after the user is ready to commit and execute the promotion. In the example below, notice that all development Tasks, Artifacts, and Connections will be promoted. As there is no Remote Engine configured for the production environment, one will be created in Talend Cloud. You still need to install and pair the Remote Engine on your Linux server.

Talend Cloud also warns you that your connection parameters will be created in the production environment, but you will have to supply production values.



Promote
Development to Production (Parameters will be kept as in target)

Configuration
Review 2

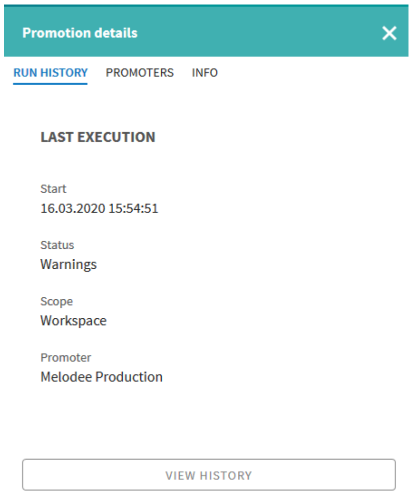
PROMOTION ANALYSIS

Scope	Workspace
Workspace	floras_flowers
Resources (if used)	Override

Workspaces (1)
- floras_flowers(Custom): (4) Reused
  - Tasks: (4)
    - a00_create_db_schema(Version 1): Created
    - a00_flower_sales_master(Version 1): Created
    - a00_load_tables_master(Version 4): Created
    - u00_random_names(Version 1): Created
  - Artifacts: (4)
    - a00_create_db_schema(Version 0.1.0): Created
    - a00_flower_sales_master(Version 0.1.0): Created
    - a00_load_tables_master(Version 0.1.3): Created
    - u00_random_names(Version 0.1.0): Created
  - Connections: (1)
    - mysqlFlowers_DEV: Created New connections have to be updated manually with real parameter values.
  - Engines: (1)
    - floras_RE: Created New Remote Engines have to be paired manually.

Click **Promote**.

Now in the Promotion Details, there is run history.



LAST EXECUTION

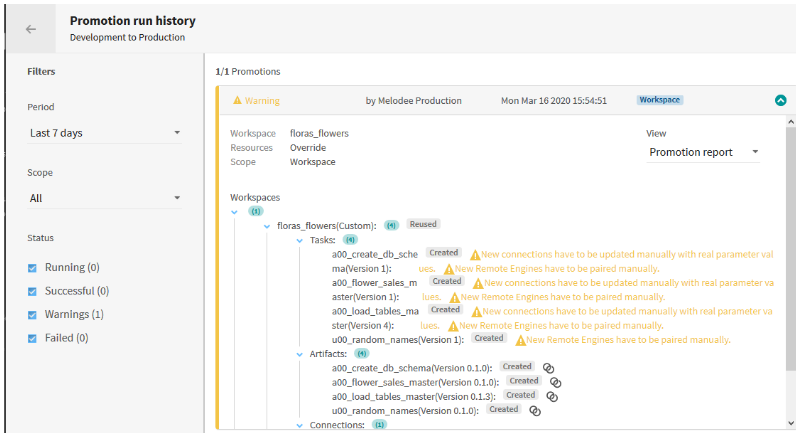
Start
16.03.2020 15:54:51

Status
Warnings

Scope
Workspace

Promoter
Melodee Production

Click **View History** to see the results of your development to production promotion.



1/1 Promotions

Warning by Melodee Production Mon Mar 16 2020 15:54:51 Workspace

Workspace	floras_flowers
Resources	Override
Scope	Workspace

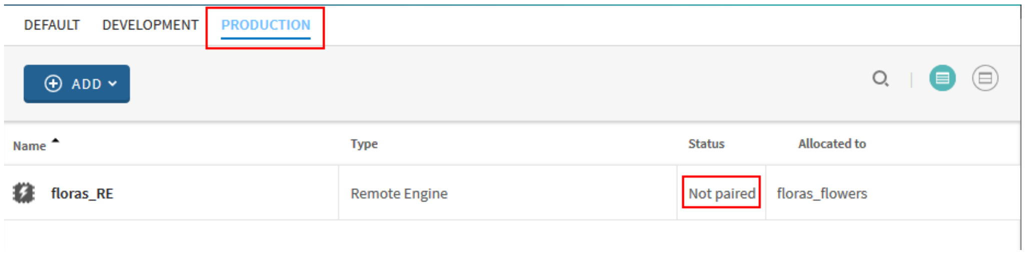
View: Promotion report

```
Workspaces
(1)
  floras_flowers(Custom): (4) Reused
    Tasks: (4)
      a00_create_db_schema(Version 1): Created  New connections have to be updated manually with real parameter values.  New Remote Engines have to be paired manually.
      a00_flower_sales_master(Version 1): Created  New connections have to be updated manually with real parameter values.  New Remote Engines have to be paired manually.
      a00_load_tables_master(Version 4): Created  New connections have to be updated manually with real parameter values.  New Remote Engines have to be paired manually.
      u00_random_names(Version 1): Created  New Remote Engines have to be paired manually.
    Artifacts: (4)
      a00_create_db_schema(Version 0.1.0): Created
      a00_flower_sales_master(Version 0.1.0): Created
      a00_load_tables_master(Version 0.1.3): Created
      u00_random_names(Version 0.1.0): Created
    Connections: (1)
```

As you see, there are warnings thrown. There are two warnings: the first is a reminder you need to install and pair a Remote Engine. The second reminds you to supply values for your connection parameters.

Pairing the production Remote Engine

As the production user, and from the Management Console, access the **Engines** page. There is a new Remote Engine in the Production environment with status **Not paired**.

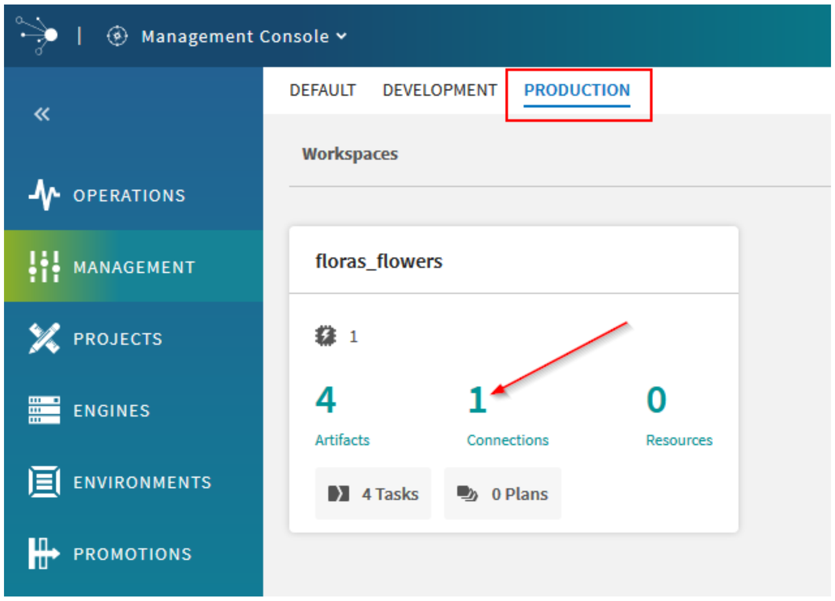


Name	Type	Status	Allocated to
floras_RE	Remote Engine	Not paired	floras_flowers

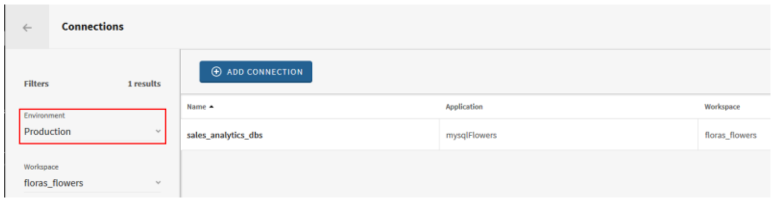
Review the section on Remote Engine Pairing and start your production Remote Engine. Access the pairing URL and pass the Production Remote Engines pairing key to the Remote Engine and wait until the status on this page changes to **Paired**.

Configuring the production Connections parameters

Access the **Management** menu from the **Management Console**. Notice you now have 4 artifacts, 4 tasks, and 1 connections in the **floras_flowers** workspace in the Production environment. Click the **1** Connections.



Notice it is showing as listing connections for the Production environment.


Connections

Filters 1 results

Environment: Production

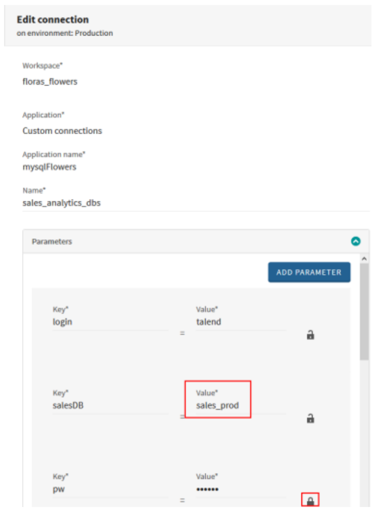
Workspace: floras_flowers

ADD CONNECTION

Name	Application	Workspace
sales_analytics_dbs	mysqlFlowers	floras_flowers

Click the name of the connection and then add production values to the connection parameters. For this use case, they are the same as for the development connection except for the database names:

- **sales_prod**
- **analytics_prod**


Edit connection
on environment: Production

Workspace*: floras_flowers

Application*: Custom connections

Application name*: mysqlFlowers

Name*: sales_analytics_dbs

Parameters

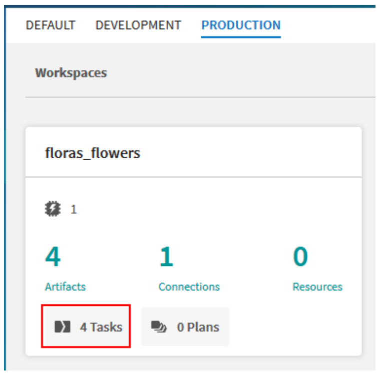
ADD PARAMETER

Key*		Value*
login	=	talend
salesDB	=	sales_prod
pw	=	••••••

Click the **lock** icon to hide the value of the password.

Running production Tasks

Return to the **Management** menu and select the **Tasks** link.



Starting with the create DB schema master Job, notice that connection parameters are present and correct. For Jobs that have an Advanced Parameter, add the connect value for production (file: **/prod_context_values.csv**). Run each Job in order and check the logs. Use your MySQL workbench to verify that database artifacts were created in the production schemas.

All Jobs have now run with production parameters on a production Remote Engine.