Lab: Deployment in WSL

June 19, 2018

Author: Elena Lowery elowery@us.ibm.com

Table of contents

Contents

Overview

In this lab you will learn how to deploy analytics assets in IBM Watson Studio Local (WSL)

Required software, access, and files

- To complete this lab, you will need access to a Watson Studio Local cluster.

Deployment in WSL

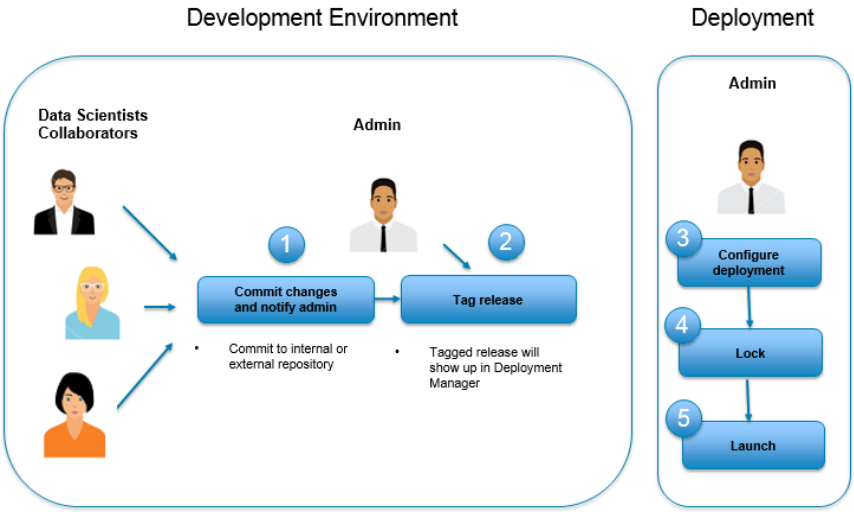
Deployment in WSL is accessed through IBM Watson Machine Learning, which is available in the WSL UI for users with *deployment admin* privilege.

Deployment provides the following capabilities:

- Deployment of models and scripts for online scoring
- Deployment of notebooks and scripts for batch scoring
- Deployment of Shiny applications
- Scheduling of model evaluation.

Some of the deployment tasks can be done in the development environment, but from the licensing perspective, these tasks should only be used for testing.

Deployment Manager provides important separation of the development and deployment tasks. Here's how the deployment workflow is implemented in WSL.



1. Data scientists collaborate on a project, and when they're done with development and testing, they notify the admin that the project is ready to be deployed in production.
2. The admin tags the project release and uses **Deployment Manager** UI to configure deployments.
3. When the project is launched, the REST APIs for deployed assets become available and all scheduled jobs will run as configured.

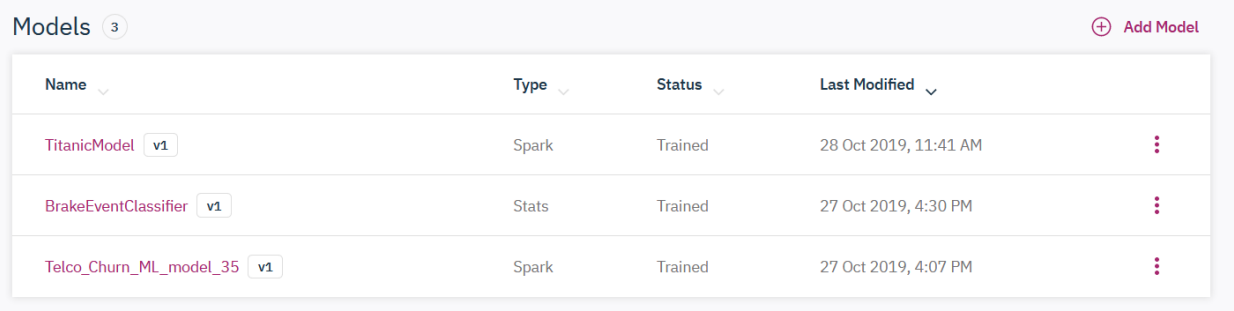
Part 1: Test Assets that need to be deployed

Note: testing is done in the development environment.

Test Online Scoring

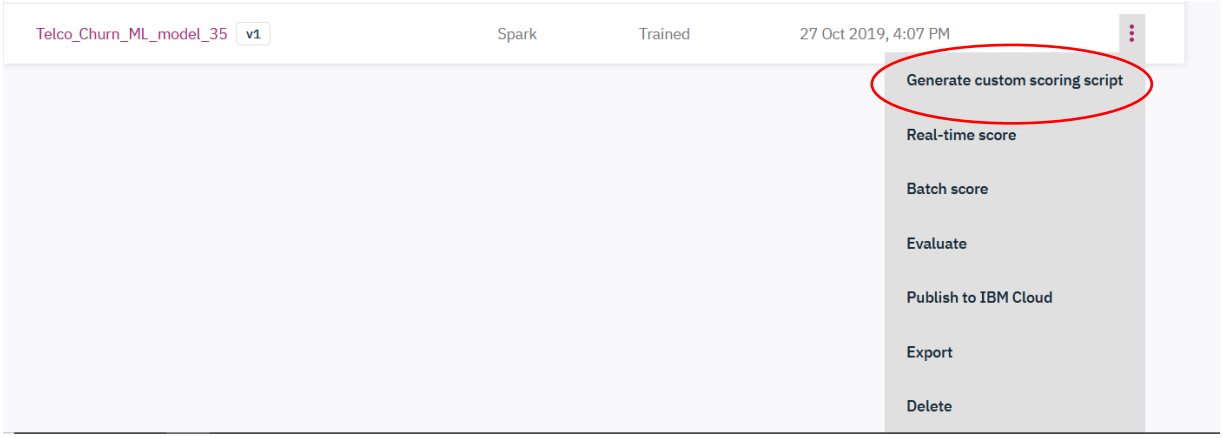
1. If you haven't already created a project, follow instructions for *Lab 1* in this repository
2. If you haven't run through the *TelcoChurn_SparkML_35a* notebook, run through it so that you generate a model. The easiest way to do this is to open the notebook, then in the menu select **Cell -> Run all**

Navigate to the **Assets** view and make sure that the model has been created. Your model may have a different name and version.

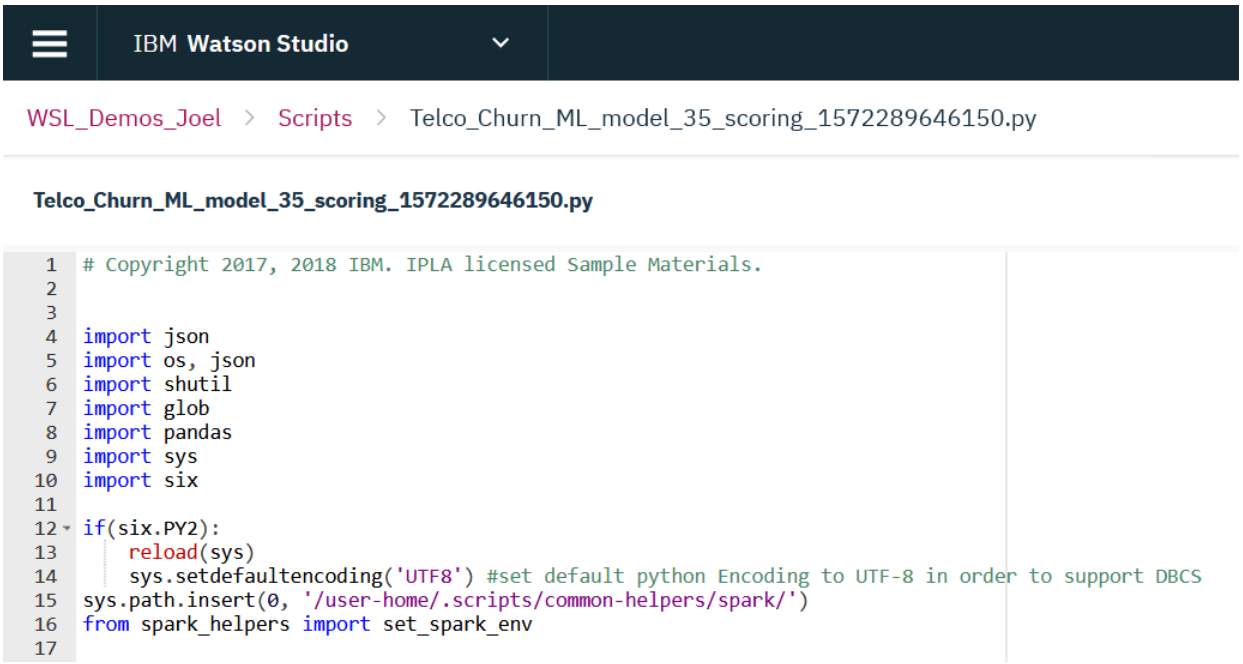


3. Click on the vertical ellipses next to the model.

   Notice that you have **Generate Script** option for the model. If you generate a script for online model deployment, you can add addiitonal operations prior to invoking scoring. Generating a script is an ***optional*** task for online scoring. If you don't generate a script, a default script will be used during deployment.

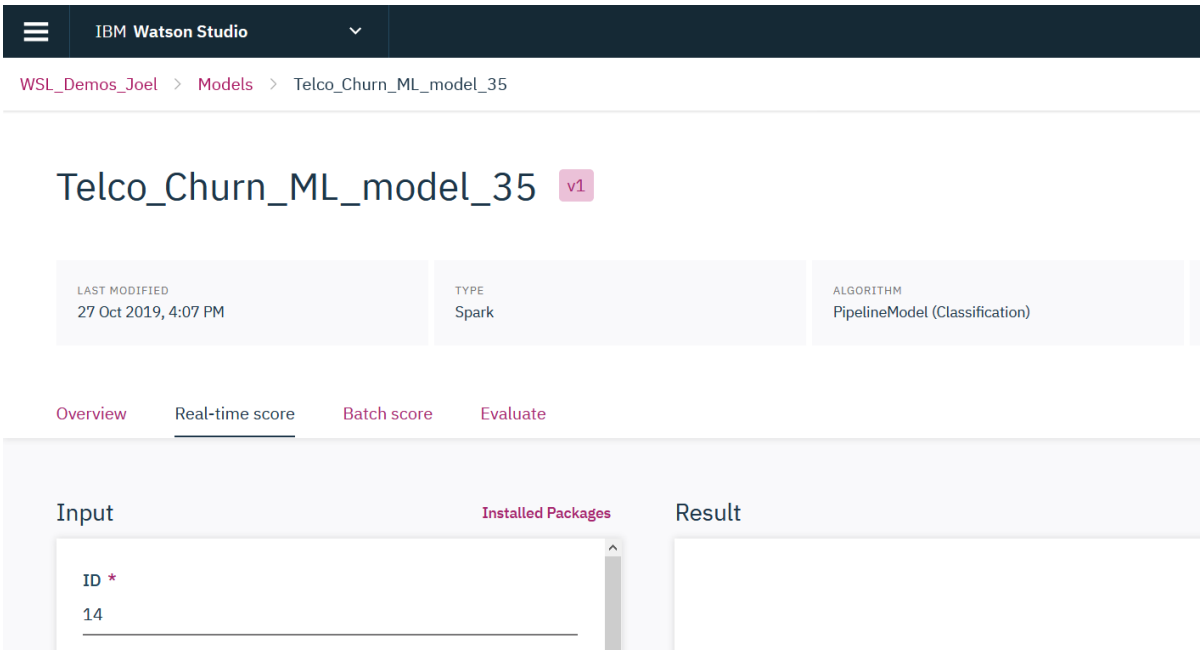


   If you would like to take a look at the default script, click the **Generate Script** button. Take a note of the script name – later you can use it for depoyment.



4. Click on the vertical ellipses next to the model and select **Real-time score**.

   The data for testing is prefilled because we save the training data with the model. You can test with the default data or change some values.

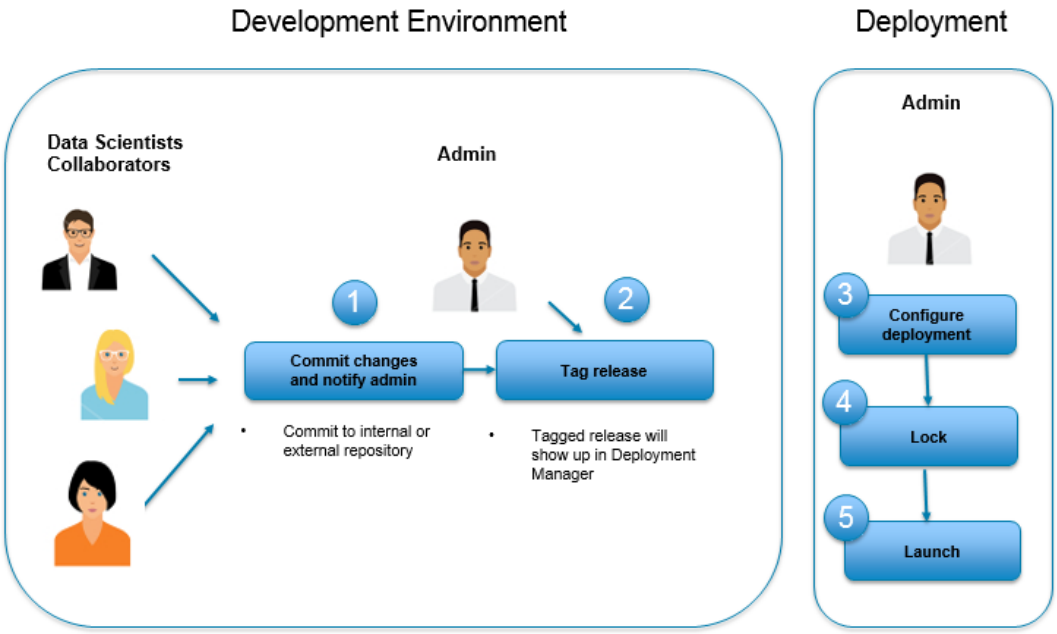


   It's also possible to test with the "internal REST API", as shown in the *TelcoChurn_SparkML* notebook. The external REST API will be available when we deploy the model in the **Deployment Manager** (later in this lab).

Part 2: Deploy project to production

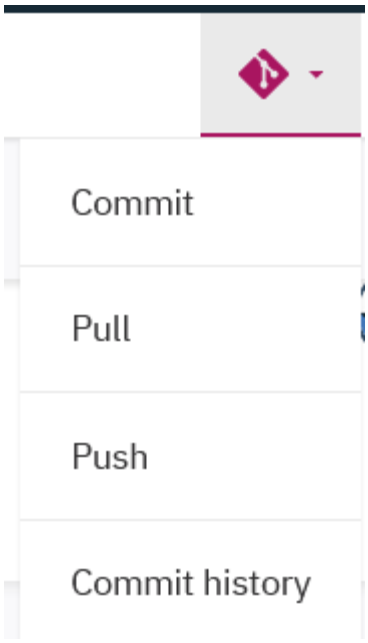
Deployment in WSL can be done by any user who has the deployment admin role for both WSL or the project. Many companies have a designated person who's responsible for deployment of analytics.

As a reminder, we will be following the following steps to deploy analytic assets to production.

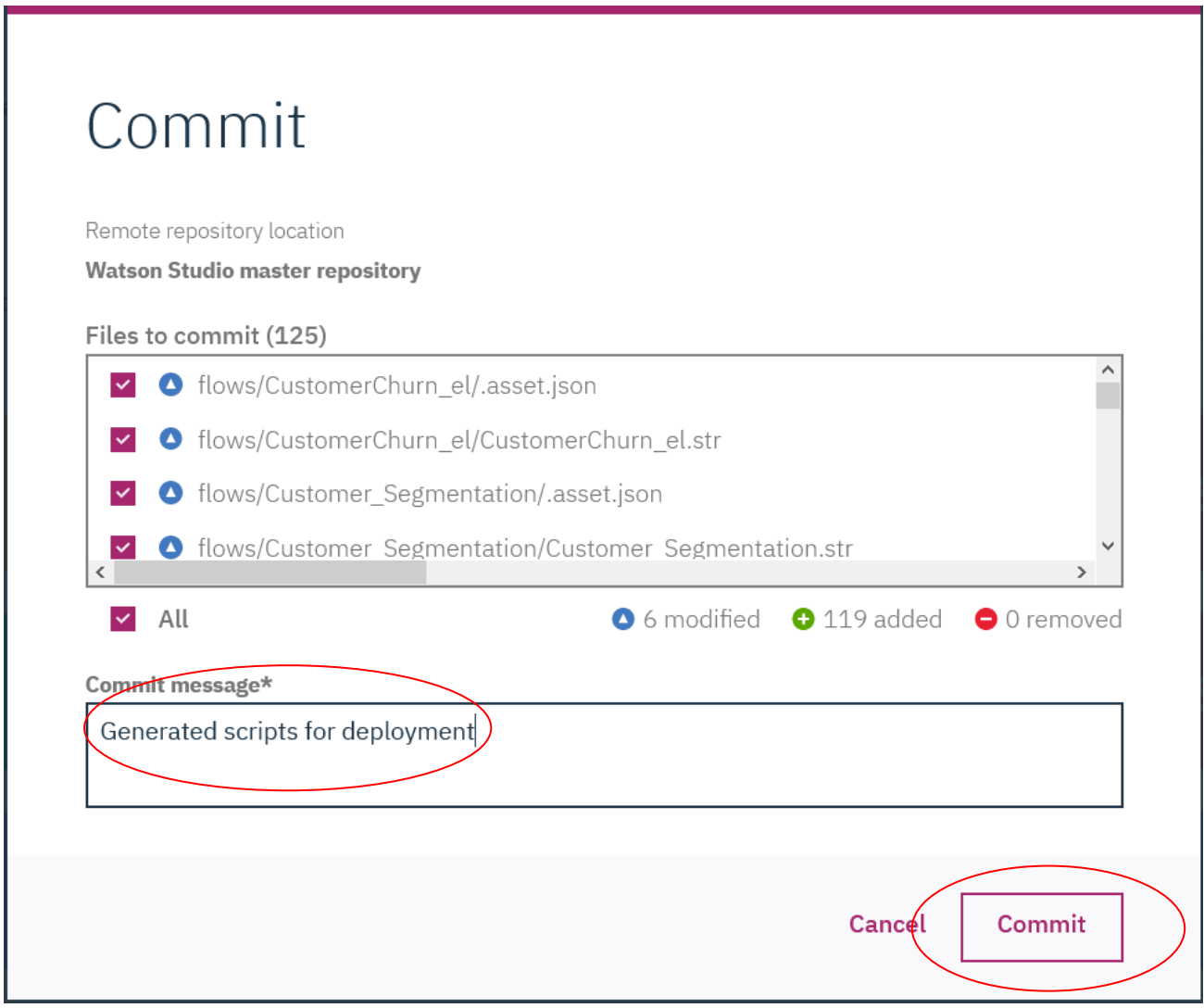


1. Data scientists need to commit changes to the project. You already may have seen the "*You have local changes you can commit*" message close to the menu bar. You can invoke Commit by clicking on the **Git actions** icon in the top right corner. Select **Commit**.

   Note: You have to be in the Project details view to see the icon.


Commit
Pull
Push
Commit history

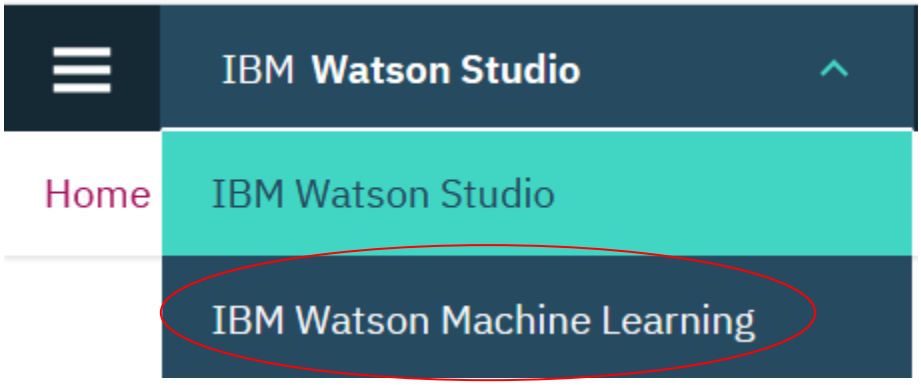
2. Provide the *Commit message* about the changes. Click **Commit**.



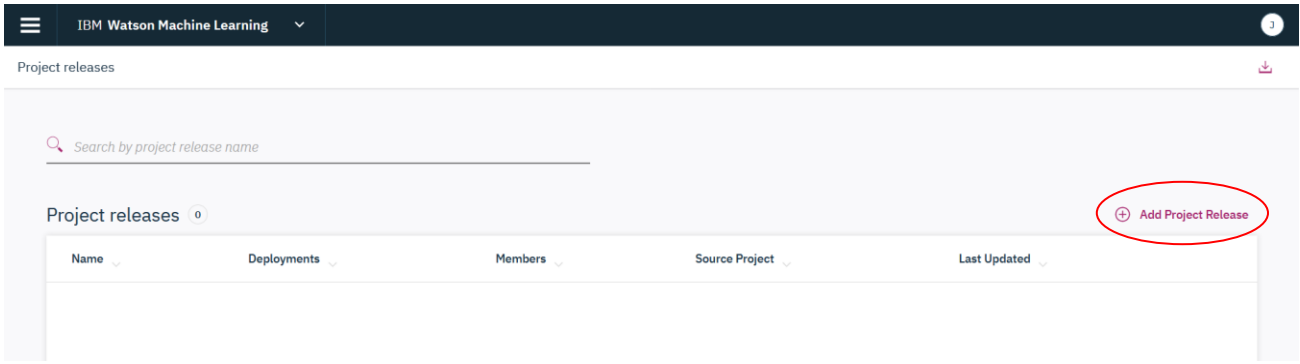
3. Now we are assuming that a different person, a *deployment admin*, is taking over deployment. Click on the **Git actions** icon and select **Push**. Notice that we can add a *tag* to the project.

   A *tag* is used to identify a specific version of the project. There may be many versions of the assets in the project, but only specific versions should be used in production.

4. Navigate to the **Deployment Manager** view by selecting **IBM Watson Machine Learning** from the main menu.

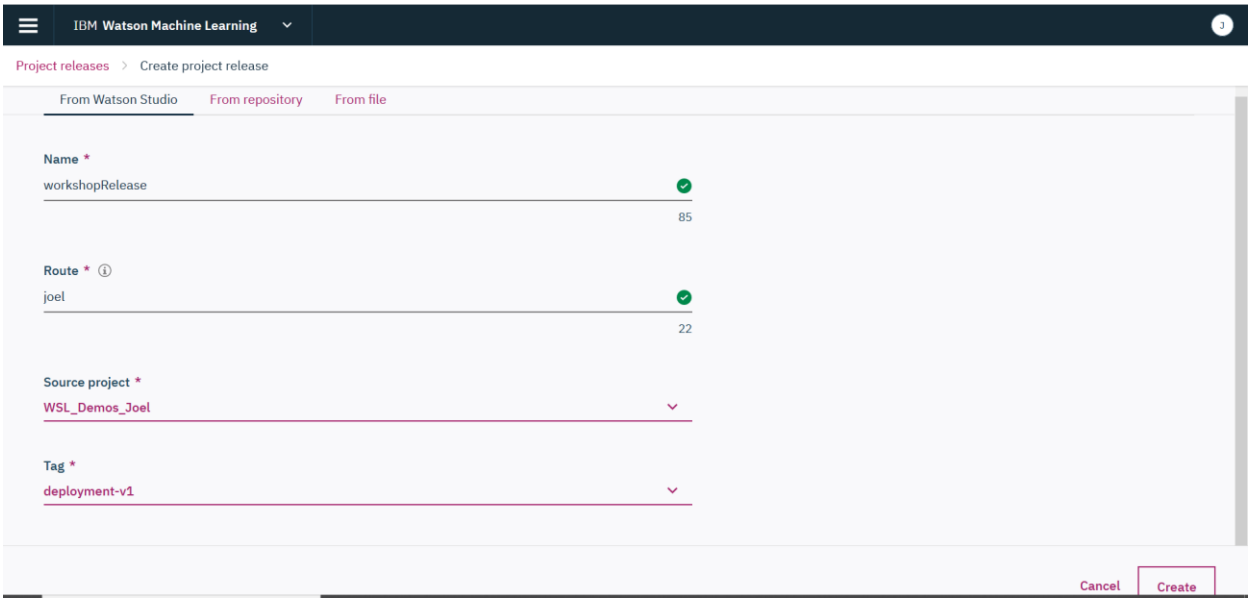


5. Click the **+Add Project Release** icon to create a project release.



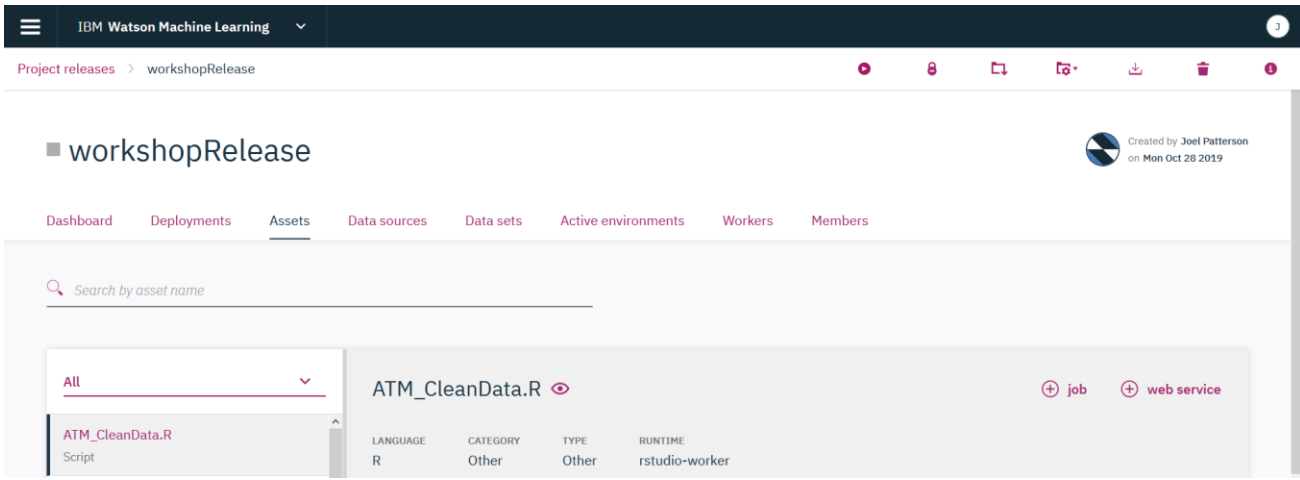
6. Enter the required fields

   - **Name**: *WorkshopRelease1*
   - **Route**: *release1* (this string will be used in all URLs that are generated by deployment, that's why it can't have spaces, upper case characters, and special characters). *Important: if you're sharing a cluster, make the route unique, for example, add initials to release1.*
   - **Source project:** the project that you want to deploy
   - **Tag**: tag that you specified in the earlier steps.

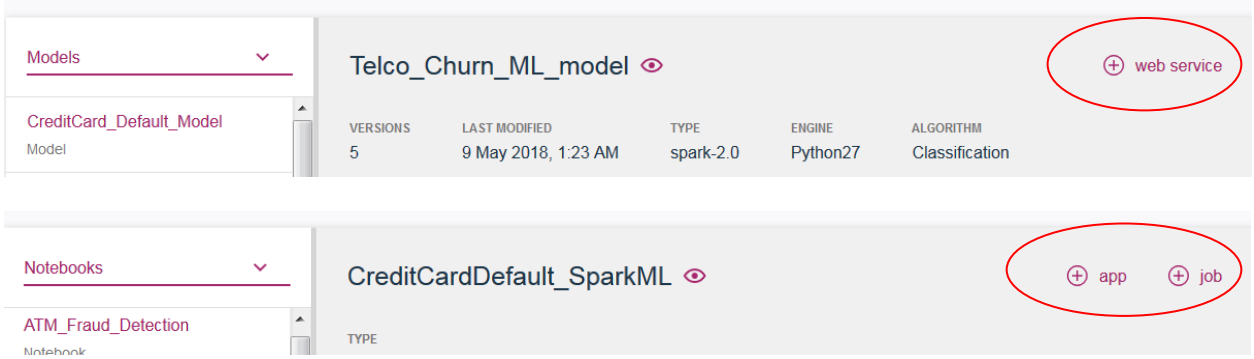


   Click **Create**. This will take a few minutes because WSL is making a copy of all assets in the project.

7. The default view shows all assets that are a part of the project. Notice that you can filter them by type if you select the drop down.



8. Select different asset types (models, notebooks, scripts, etc.), and notice how deployment options (icons in the right corner) change.



WSL automatically determines applicable deployment type for each asset.

- **Models** can be deployed as Web services for **online scoring** (invoked with a REST API). To deploy a model for batch scoring, the "script" deployment option is used. The script is generated on the development side, as we have done in *Part 1* of the lab.

- **Scripts** can be deployed for **online** or **batch scoring**. WSL automatically determines if the script can be used for online scoring (a script generated by WSL or a script that follows the specific format for online scoring). WSL supports deployment of any R or Python script. Documentation explains the required script format for online scoring: https://content-dsxlocal.mybluemix.net/docs/content/local-dev/dsxl-scripts-as-web-services.html

- **Model evaluation** is a type of **batch script deployment**.

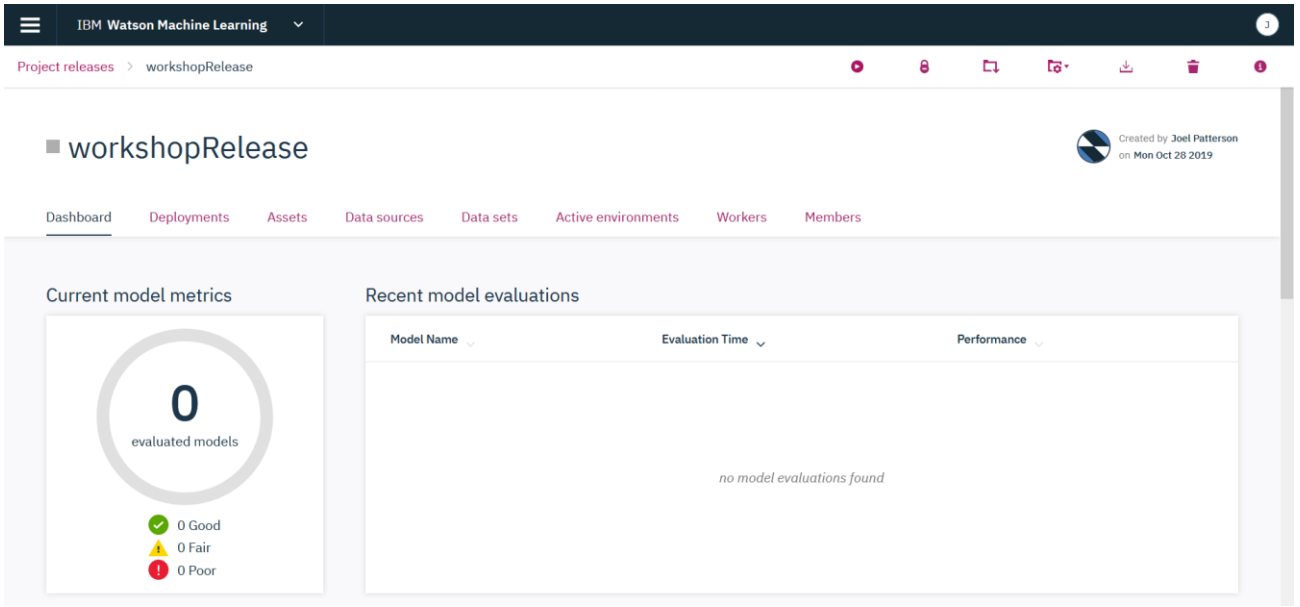
- **Notebooks** can be deployed for **batch scoring** or as **an application**. Batch scoring of notebooks can be used to perform many functions: scoring, model refresh, data preparation, etc. Publishing a notebook as "an application" makes it available as an HTML page that can be accessed with specified level of security.

- **Shiny applications** can be deployed as **an application**. Publishing Shiny as "an application" makes it available as an HTML page that can be accessed with specified level of security.

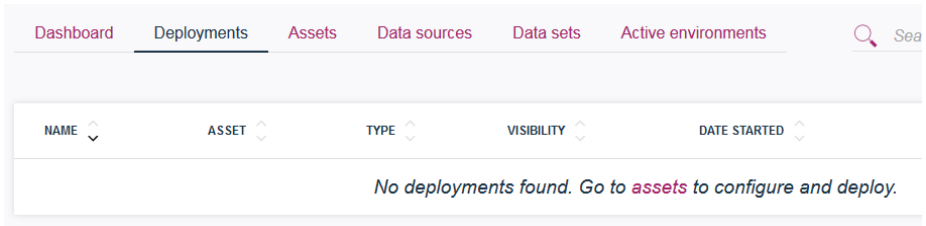
- **SPSS flows** can be deployed for batch scoring.

9. Before we configure deployment, let's review properties of the *release*.

   - On the **Dashboard** tab you will see results of evaluations (none exist now because only just deployed this project). The evaluations shown here are both evaluations done in the development environment and configured in deployment. We will configure an evaluation later in this lab.



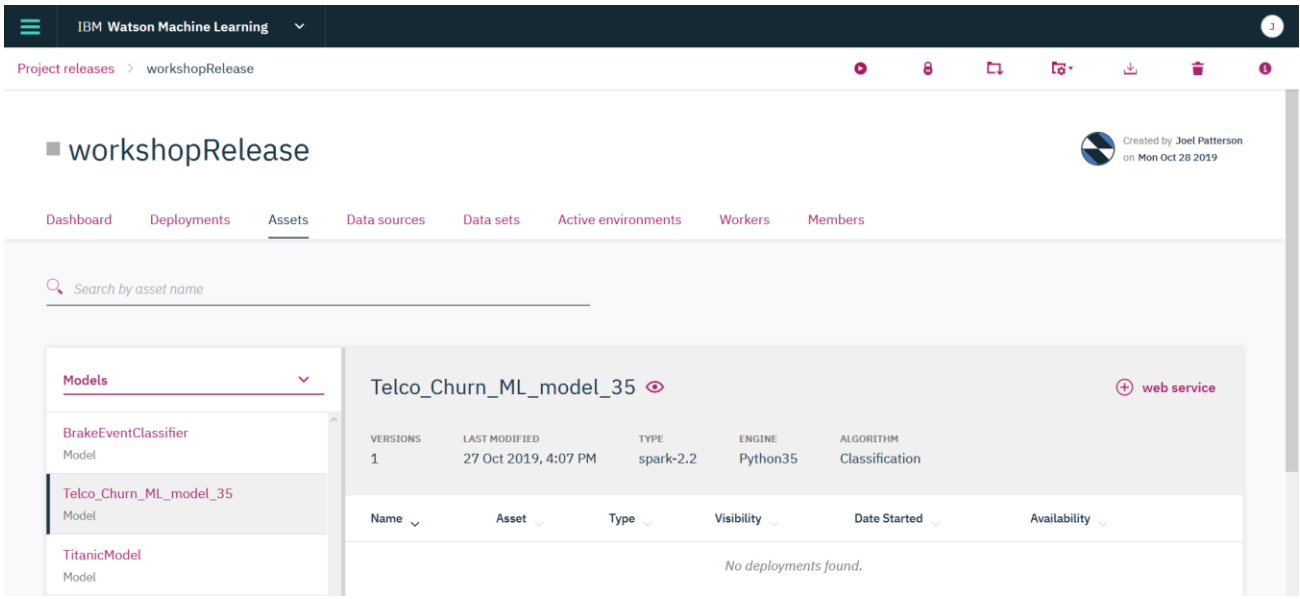
   - The **Deployments** tab is empty at this time because we haven't configured any deployments yet.



- The **Assets** and **Data Sources** tabs show the same assets and data sources as the development side.

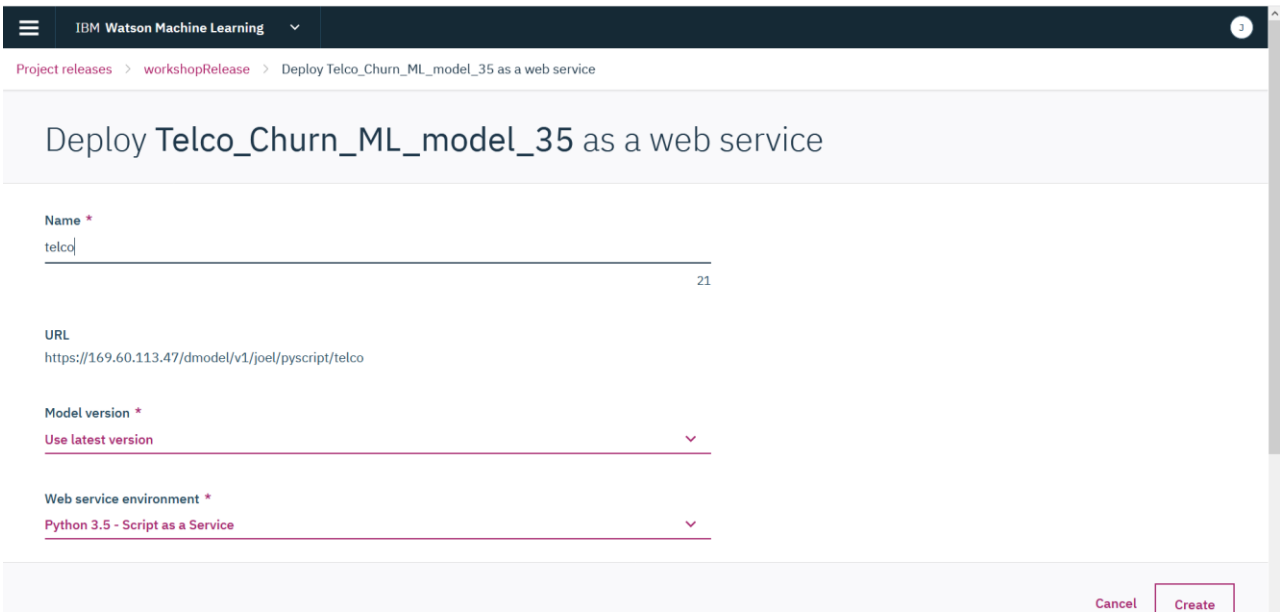
- **Active environments** is also empty because we haven't configured and launched the project yet.

10. We will start with configuring online deployment. On the **Assets** tab, select **Models** in the dropdown, then click on *Telco_Churn_ML_model_35*.



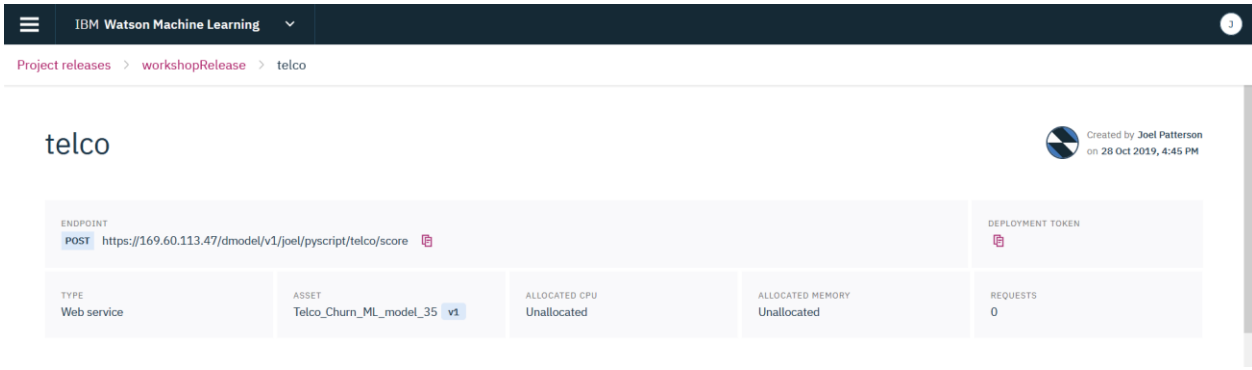
11. Click the **web service** button. Fill out the required fields.

    - **Name**: *telcochurn1*. Notice that the name gets appended to the URL (REST endpoint), that's why it has to be lowercase with no special characters.
    - **Model version**: you can select different versions of the model
    - **Web service environment**: should be the same as the environment in which model was built (selected by default)
    - **Reserve resources**: checking this option will provide dedicated CPUs and memory to online deployment
    - **Replicas**: number of environments that host the service - select at least 2 for high availability (don't do this in the lab!).



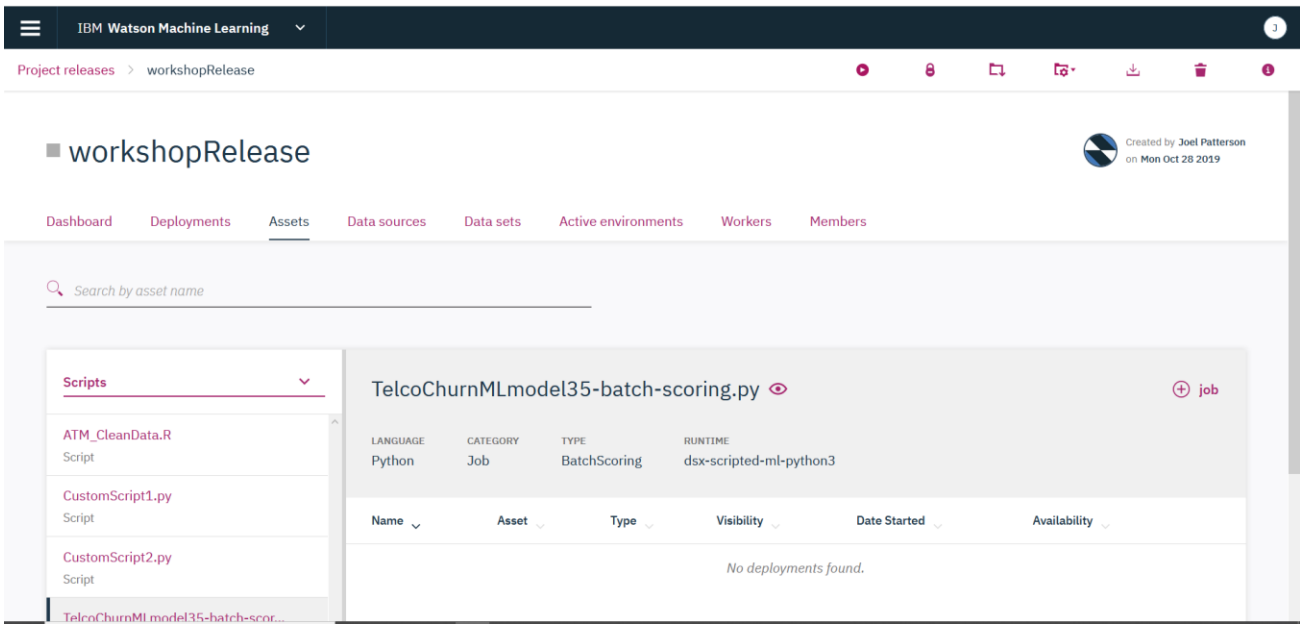
Click **Create**.

The **REST endpoint** is displayed in the model details. This endpoint won't be live until we launch the project, which we will do after we configure all other deployments. There is also a token which will be needed by any application which wish to invoke the model.



12. Next, we will configure batch scoring and model evaluation.

On the **Assets** tab select the **Scripts** from the dropdown. Find the two scripts that we created in the development environment for batch scoring and model evaluation.



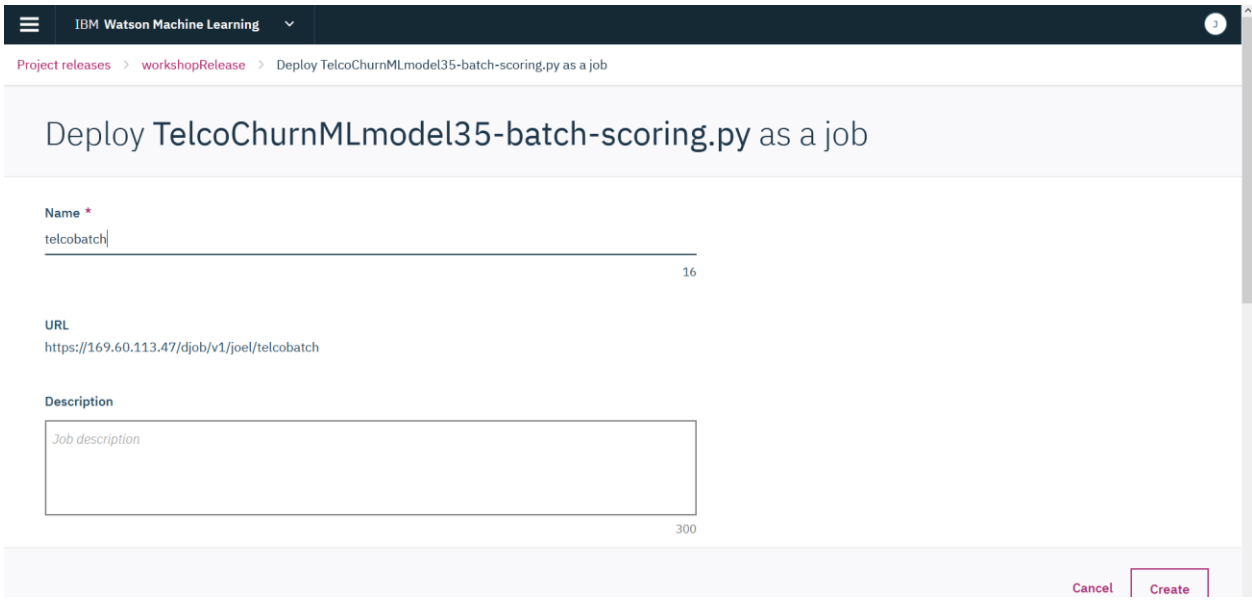
13. Select your batch scoring script and click on **+job** button. Fill out the required fields.

    - **Name**: *telcobatch*. The name gets added to the REST endoint. Batch jobs can be invoked with a REST API. Unlike online scoring, data is not passed in – the data sources that are defined in the script are used.

    - **Type**: batch scoring

    - **Worker**: *Jupyter with Python 3.5, Scala 2.11, R 3.4.3*. The worker should be the same runtime environment as was used for running the script in development.

      Scroll down and review the rest of the fields (default values can be used).

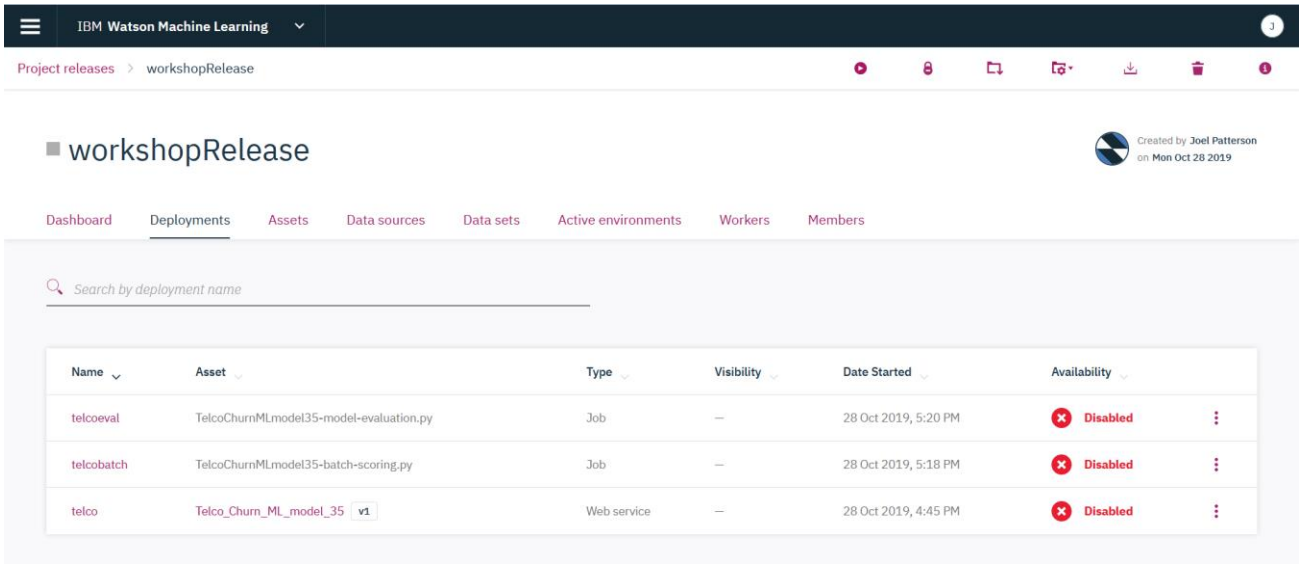


If you specified a schedule for invoking the job, it will take effect when the project is launched. You will also be able to invoke it "on demand" by selecting it in the **Deployments** view. We will complete this step later in the lab.

14. Repeat the same steps to create a model evaluation job. The only difference is the type – *Model Evaluation*.

We don't have an option to modify data source (in our example .csv files) when we deploy assets for batch scoring and evaluation. If we used a remote data source (for example, a database table or a file in HDFS), then we would be able to modify input/output by modifying the data source definition.

If you click on the **Deployments** tab now, you'll see that every deployment is in *Disabled* status because we haven't launched the project.

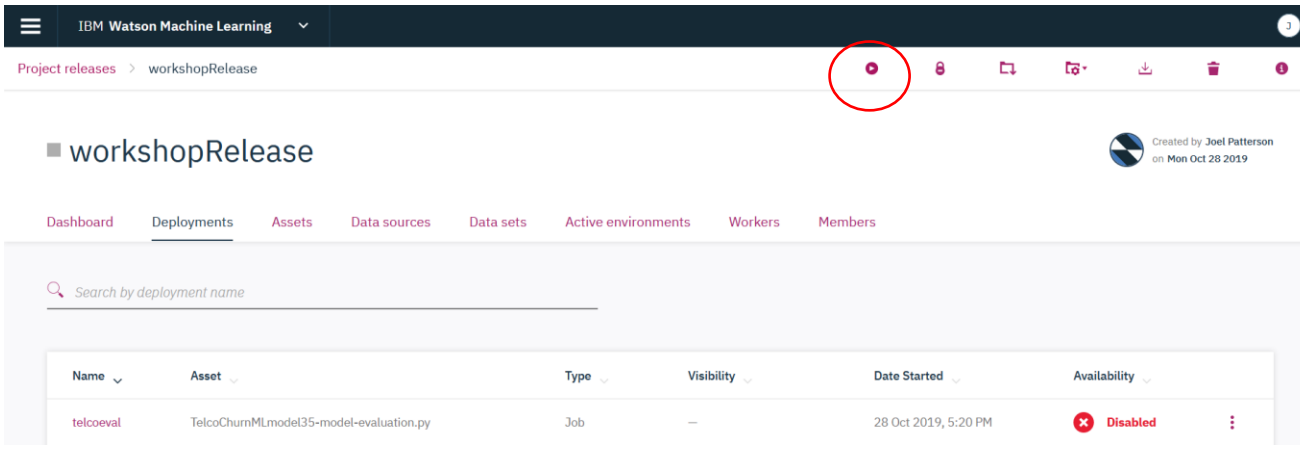


15. In this step we will deploy a notebook as an application.

    In the **Assets** tab select **Notebooks**. Click on any notebook and click **+app**. Provide name and select the desired security setting.

The URL for the notebook will be "live" (accessible) after we launch the release.

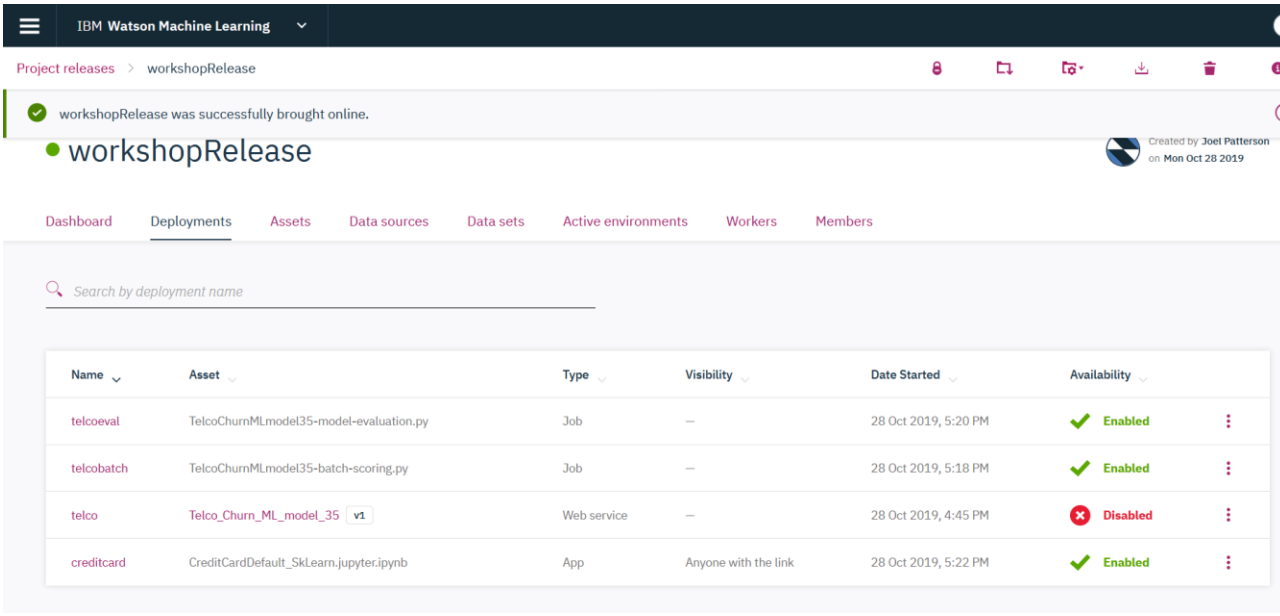
16. Now that we created a few deployments, we can lauch the release by clicking the **Launch** button in the top right corner.



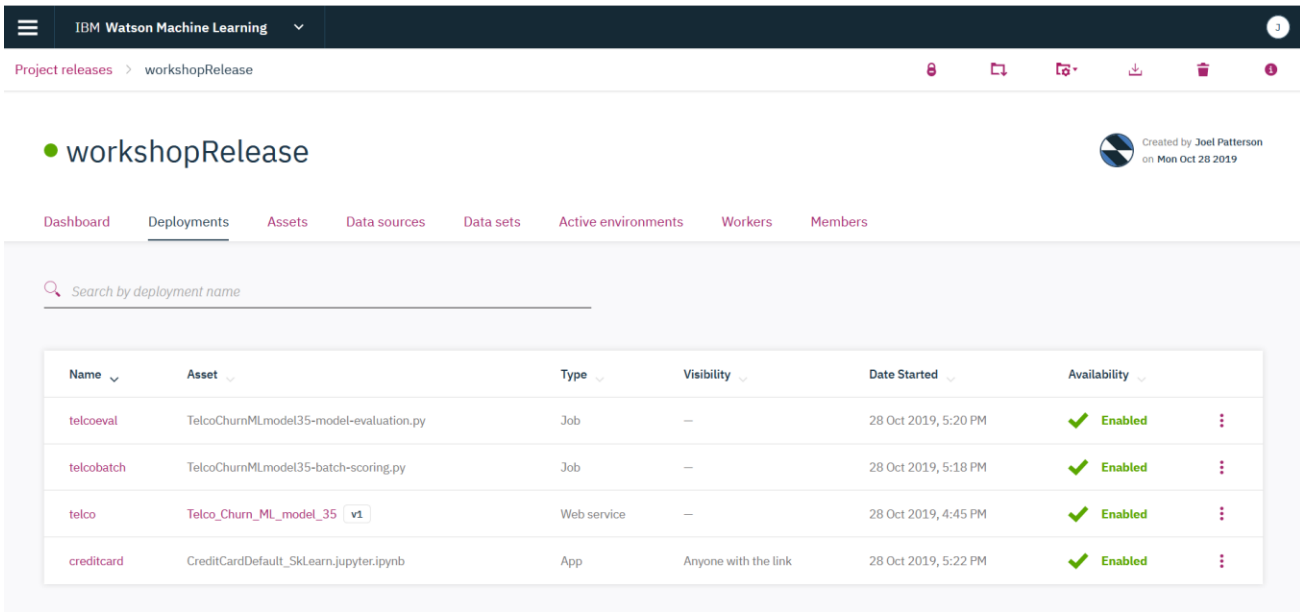
Launching the release will:

- Start all environments that will be used for deployment
- Enable the REST endpoints
- Enable URLs for "applications" (notebooks and Shiny)
- Enable schedules (if they are configured)
- Enable on-demand invocation of jobs.

Notice that when you click Launch, everything with the exception of online (Web service) deployment is immediately enabled.

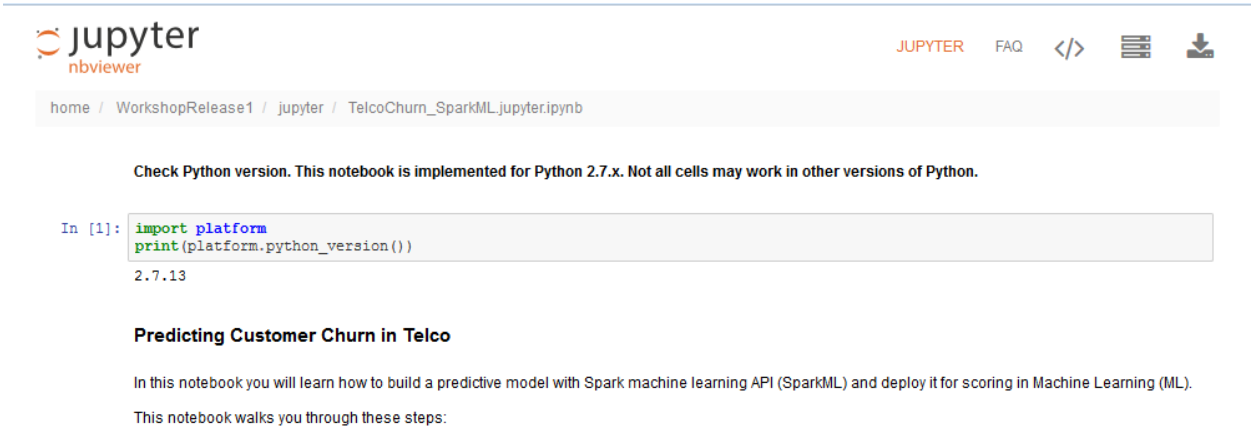


Online deployment environment takes some time to start. When it's started, the status will change to *Enabled*.

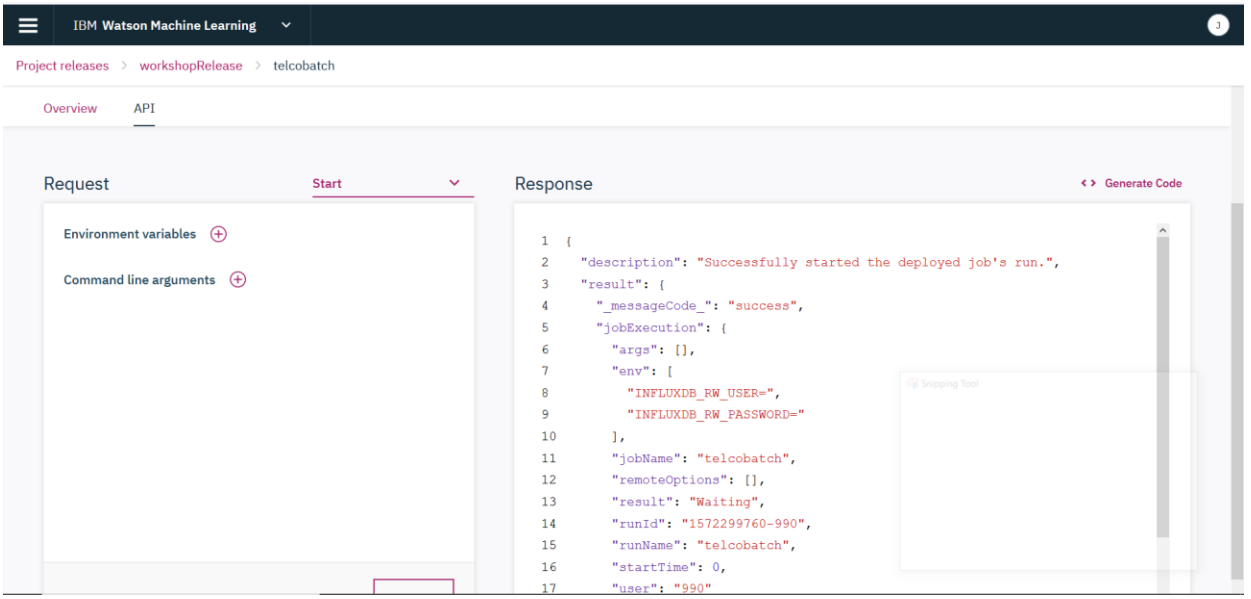


17. Now that all URLs and endpoints are active, we can test them.

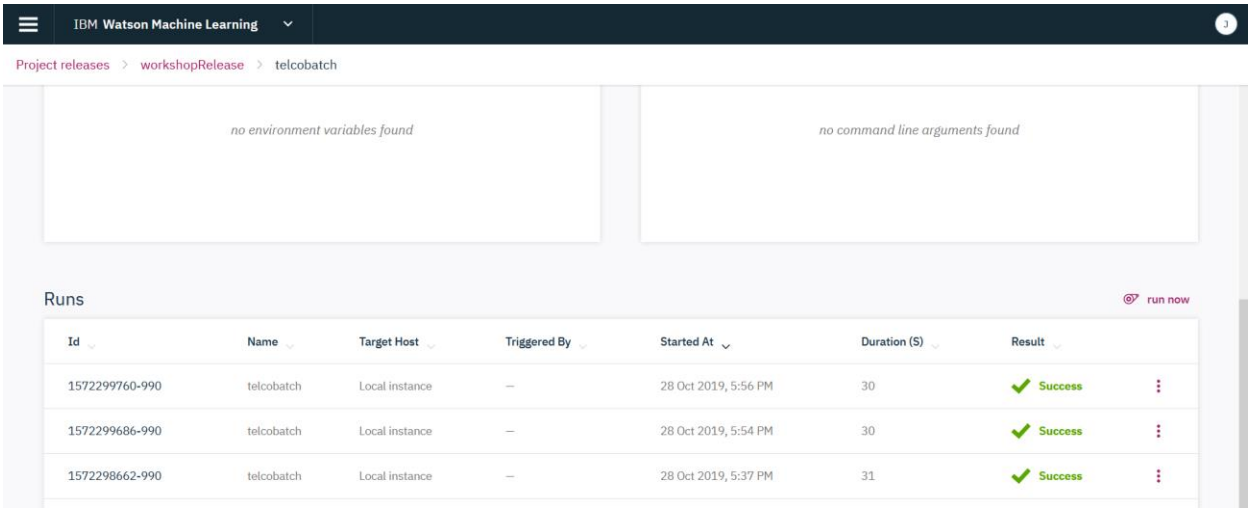
- To test the Notebook URL, click on the ellipses next to notebook deployment, select **Share Endpoint**, and copy and paste it to a new browser window.



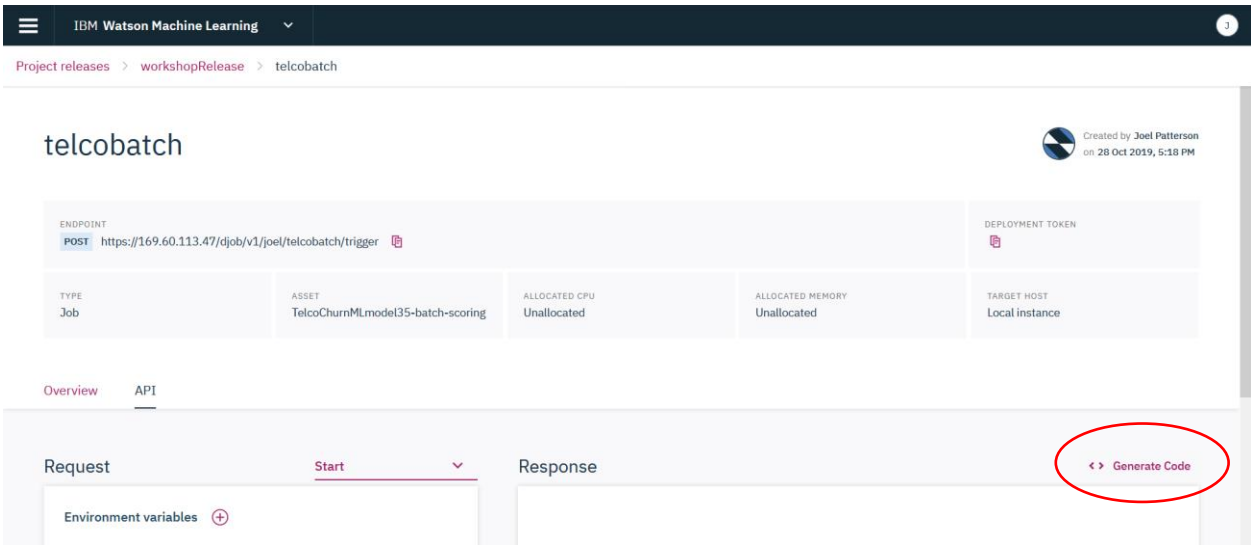
- To test the batch job, click on the deployment (row in the table), which will bring you to deployment details. Select **API** and **Start** from the dropdown menu, then click **Submit**. This issues the REST request to invoke the batch job. Verify that it was successul.



If you click on the **Overview** tab of the deployment details, you will see the job runs that were invoked during testing.



Optionally, you can click the **Generate Code** button, which shows the curl command for invoking the REST endpoint, and run it from a shell.



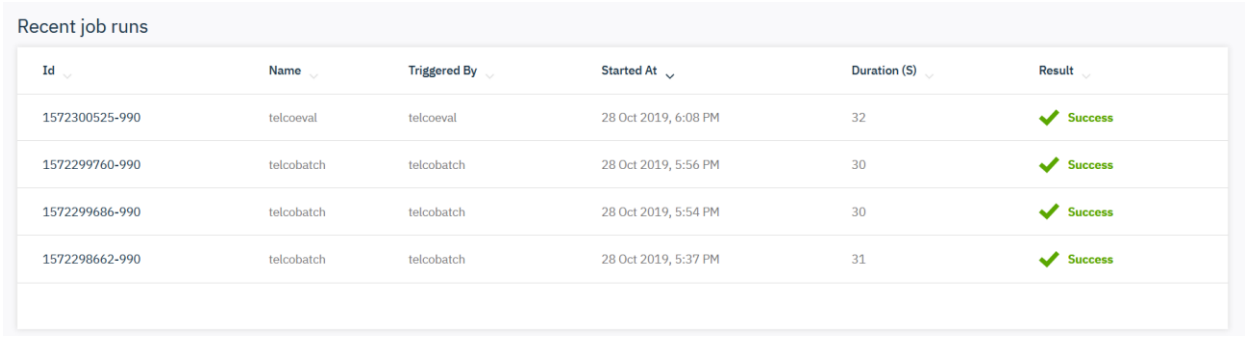

```
sh-4.2# curl -k -X POST \
>   https://169.55.181.211/djob/v1/release1/telcobatch/trigger \
>   -H 'Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZ
SI6ImFkbWluIiwicGFja2FnZU5hbWUiOiJXb3Jrc2hvcFJlbGVhc2UxIiwicGFja2FnZVJvdXRlIjoic
mVsZWFzZTEiLCJpYXQiOjE1MjkOMTc3Mzl9.T_jWzJ1tcw6MNbn52fZ-bCotCiWJ5MIyskSWznncJw3u
B8nCWePgJS4DzomEfkBULjiFBUNnIrEw93-HK-lKBchRxgSF5TRQVPQid-GJ6F3KFU7U0d6_nFN5qcZr
48qw2D1DQfyik0nkdtFYsIsfo0na-rD1iFIFylU3IzWW5x-8J8rflfX9Zj47KKW_vZs9xYWJEptfaqwz
eNkAU4vCbDkFIz8p6FkuD3qSrxIzWbsUXbJ4pOMRyP9jF55KMZO9eaUbYqWSE-FUPHBD4y1qinr2aa-c
DCbLln1x7CXtnrSboEtOxk-RCTbjs9praWCfy-qplkiKJ8Wre7Es_8zYaA' \
>   -H 'Cache-Control: no-cache' \
>   -H 'Content-Type: application/json' \
>   -d ' {"env":[],"args":[]} '
{"description":"Successfully started the deployed job's run.","result":{"_messag
eCode_":"success","jobExecution":{"args":[],"env":[],"jobName":"telcobatch","rem
oteOptions":[],"result":"Waiting","runId":"1529449672-990","runName":"telcobatch
","startTime":0,"user":"990"},"message":"success"}}sh-4.2#
```

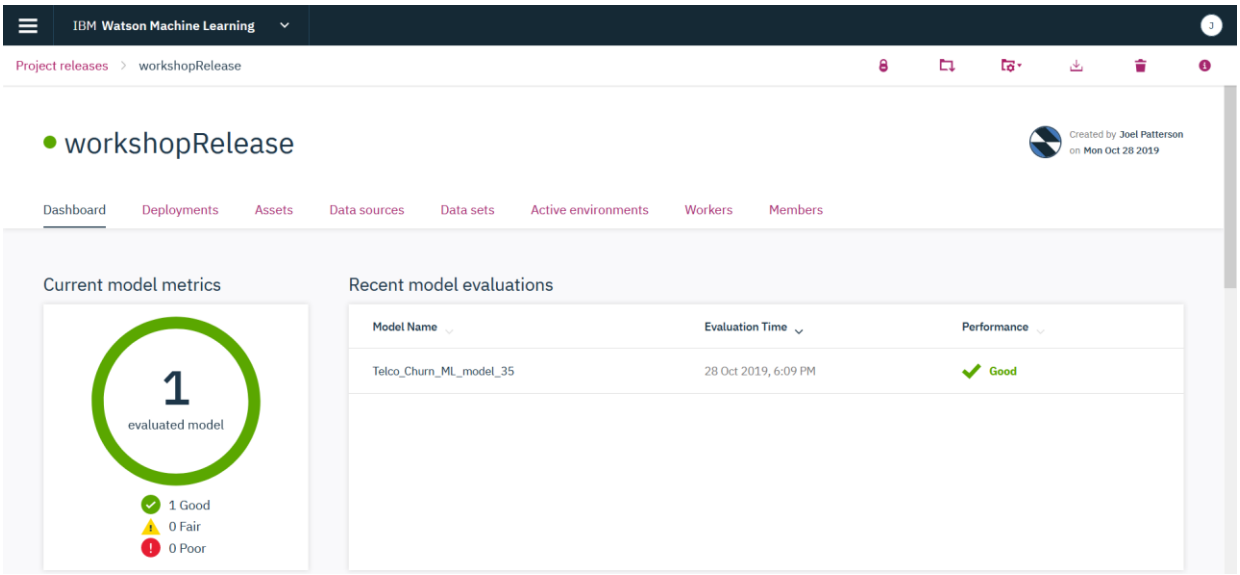
- In this section we will test evaluation. Click on the **Dashboard** tab and review evaluation details (for example, timestamps for recent evaluations). This list will change after we run evaluation.

  Switch to **Deployments** tab and click on the evaluation batch job.

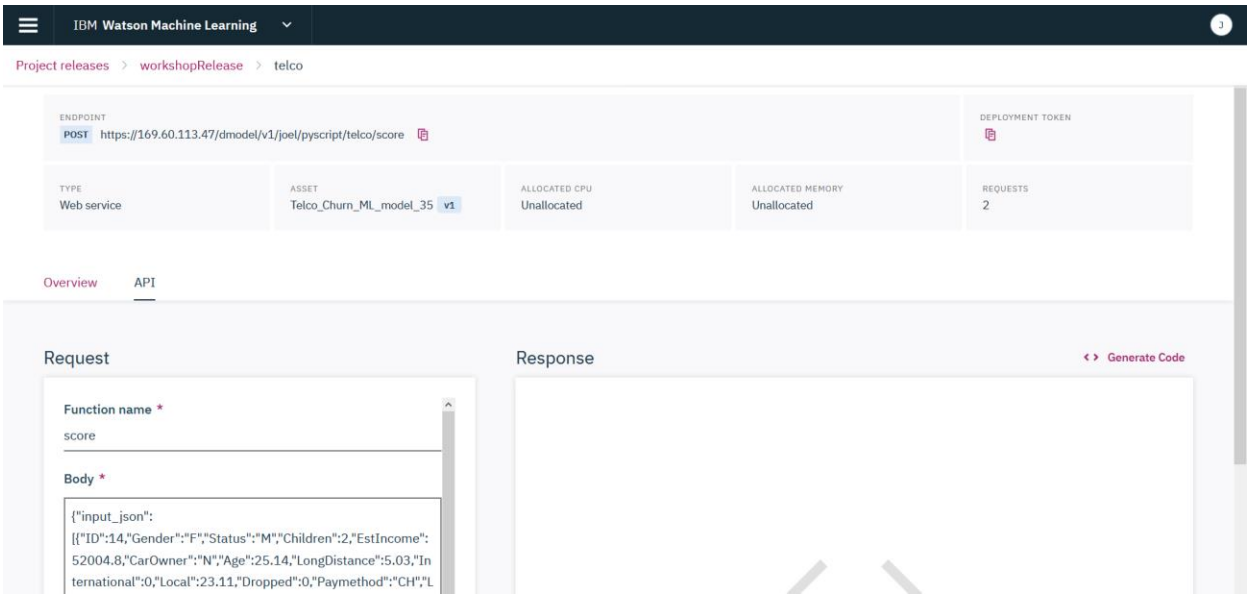
  Similar to batch job testing, select **Start** from the dropdown, then select **Submit**.

  Navigate back to the **Dashboard** tab and scroll down to **Recent job runs** table. The evaluation job will run for 1-2 minutes. When it's done, the **Recent model evaluations** table will be updated.

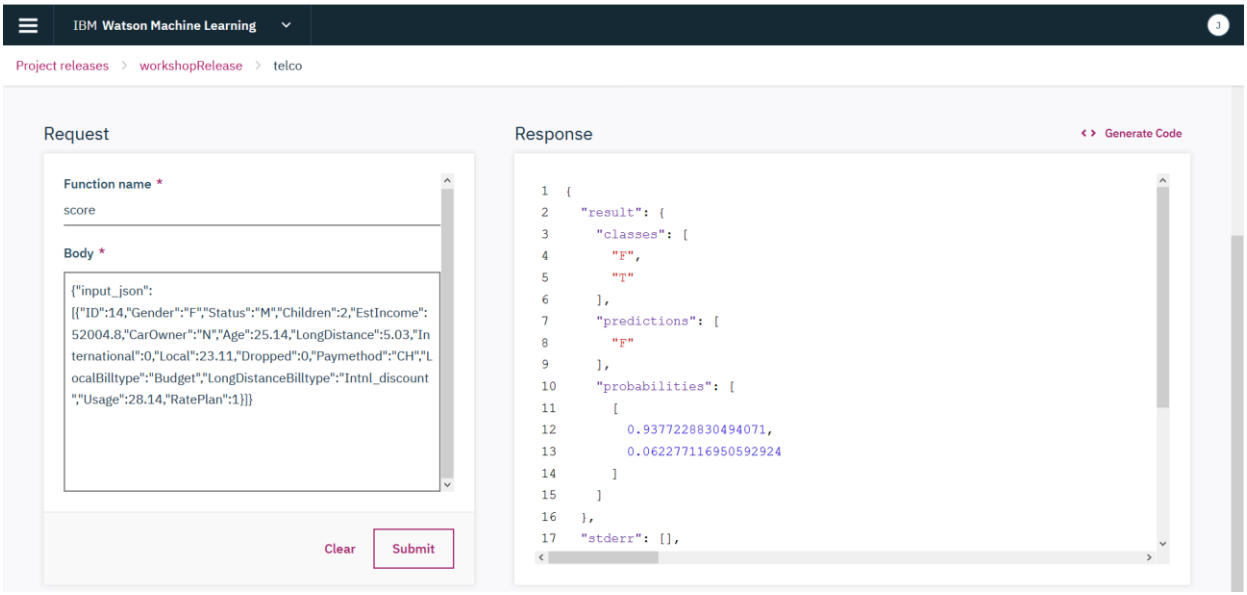




- In this section we will test online scoring. Click on the Telco Churn web service deployment and switch to the **API** tab.

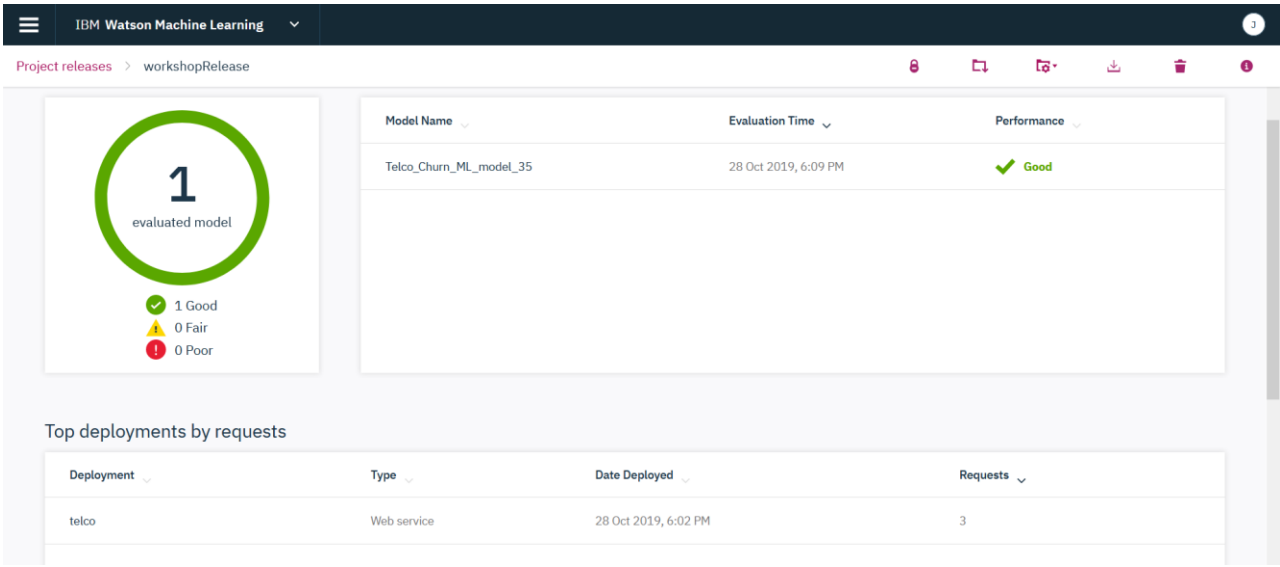


Select the **Submit** button



Optionally, you can click the **Generate Code** button, which shows the curl command for invoking the REST endpoint, and run it from sh.

If you navigate back to deployment Overivew, you'll notice that the invocation metrics have changed



You have finished the **Deployment in WSL** lab.

In this lab you completed the following steps:

- Tested assets in development environment
- Created deployment definitions
- Launched and tested deployment.
