**EmailProcessing – Connection POC Demo**

Steven Reeves, Distribution Data Scientist, Text Analytics Specialist

[sdreeves@us.ibm.com](mailto:name@ibm.com)

A close up of a logo

Description automatically generated

**Table of contents**

Contents

[Overview 1](#_Toc60759620)

[Required software, access, and files 1](#_Toc60759621)

[Required skills 4](#_Toc60759622)

[ModelOps Overview 5](#_Toc60759623)

[Feature Parity and Roadmap 6](#_Toc60759624)

[Deployment Options 7](#_Toc60759625)

[IBM Cloud Security and WML API 8](#_Toc60759626)

[API Key 8](#_Toc60759627)

[Cloud Location URL 9](#_Toc60759628)

[Watson Machine Learning Client API 10](#_Toc60759629)

[Data access – Cloud Object Storage 14](#_Toc60759630)

[Summary 15](#_Toc60759631)

[Manual Deployment to Deployment Spaces 16](#_Toc60759632)

[Programmatic Deployment to Deployment Spaces 28](#_Toc60759633)

[Optional: Additional practice 30](#_Toc60759634)

[Manual Deployment 30](#_Toc60759635)

[Deployment with API 30](#_Toc60759636)

[Deployment of Notebooks as a Job 31](#_Toc60759637)

[Deployment of Notebooks as a URL 35](#_Toc60759638)

[Conclusion and Next Steps 39](#_Toc60759639)

[Appendix A – Change Data Access 40](#_Toc60759640)

# Overview

In this Demo you will learn how to use a Python Script to pre-process a sub-set of the publicly available Enron Email Data Set.

Data:

* Enron.csv
* Prepocessed.csv

Python Script:

* PreprocessEmail

Modeler Text Analytics Stream

* EnronEmailProcessing

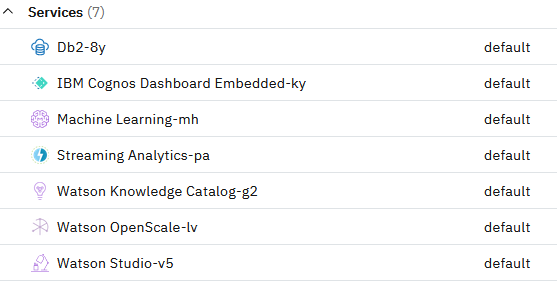
# Required software, access, and files

1. To complete this lab, you will need a **Cloud Pak for Data as a Service** (**CPDaaS**) account: <https://dataplatform.cloud.ibm.com>

* If you don’t have a CPDaaS account, use the same URL to sign up for a free trial. The account will be activated in approximately 5 minutes.

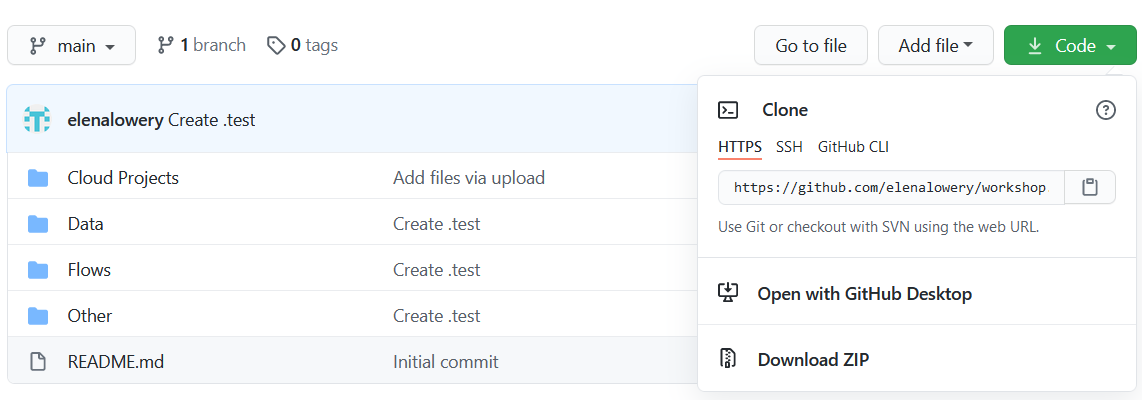
1. If you already have an **IBM Cloud** account, make sure that you provisioned the required services – **Watson Studio** and **Watson Machine Learning .**

* Navigate to your *Services Dashboard* in your **IBM Cloud** dashboard: <https://cloud.ibm.com/login>
* Check if the mentioned services are displayed under **Services**. If not, search for the services in the **Catalog** and add them.

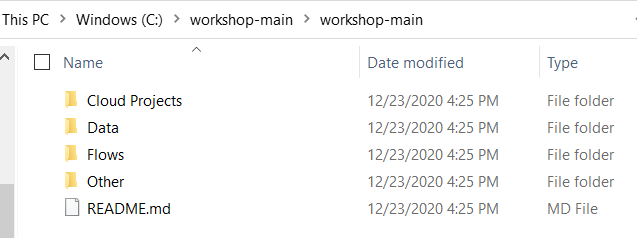


1. You will also need files from this *GitHub* page: <https://github.com/elenalowery/workshop>

* In the Github repo page, click **Code** and select **Download ZIP**.

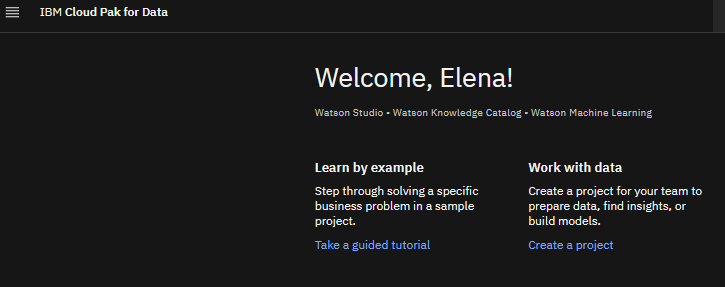


* Unzip the downloaded file until you get to this directory structure:

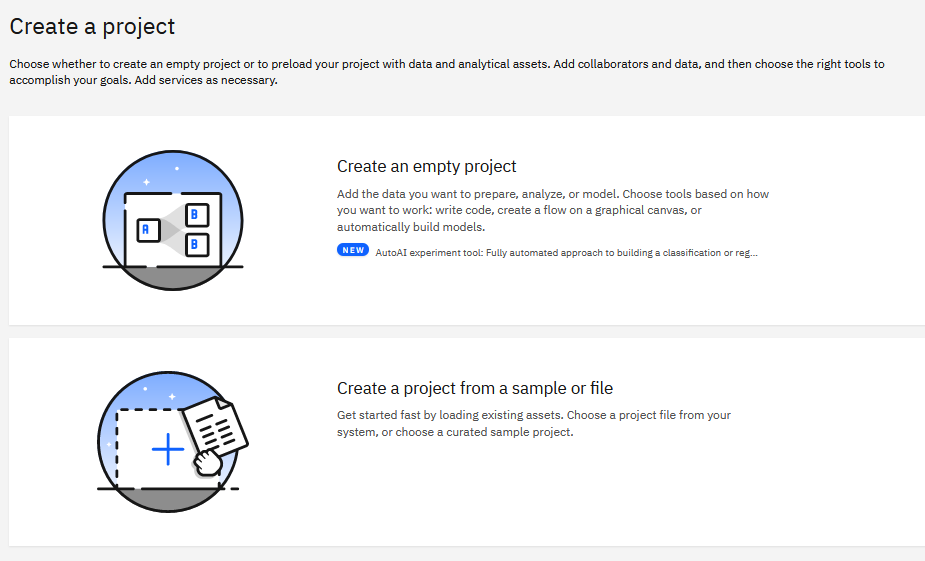


In the lab we will refer to this folder as the git repo folder.

1. Log in to **CPDaaS**, then click **Create a project**: <https://dataplatform.cloud.ibm.com>

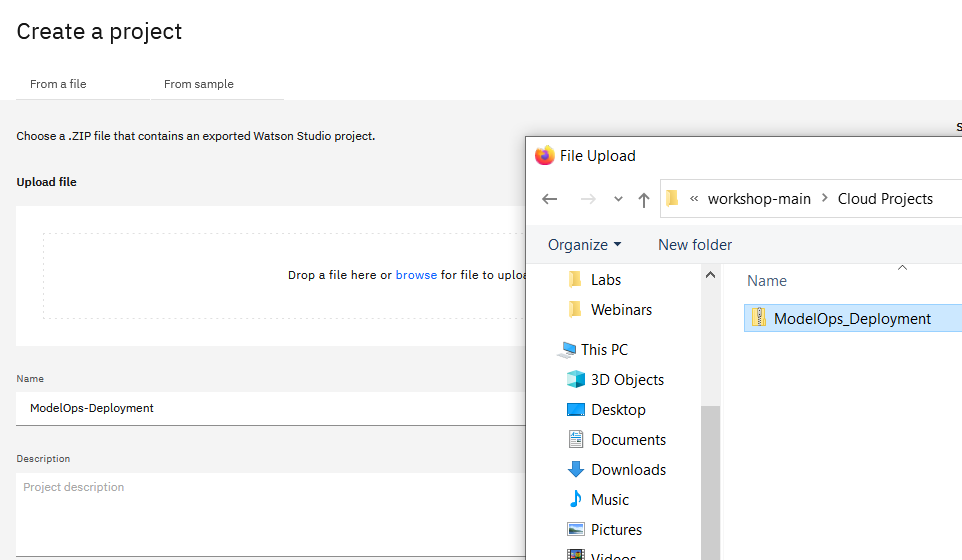


1. Select the project from file option.

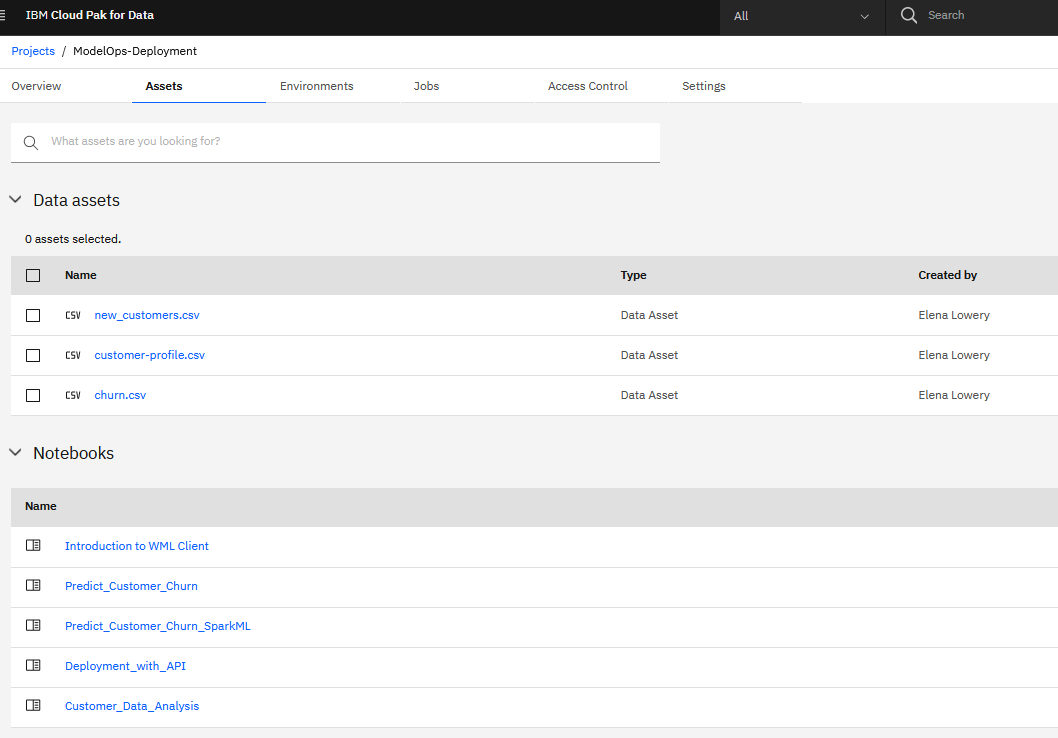


1. Enter project name, for example, ModelOps-Deployment. Navigate to the downloaded git repo/Cloud Projects folder and select ModelOps\_Deployment.zip.

Click **Create**.



1. In your project, navigate to the **Assets** view. You should see the following notebooks and data assets:



# Required skills

We recommend that users who work through this lab:

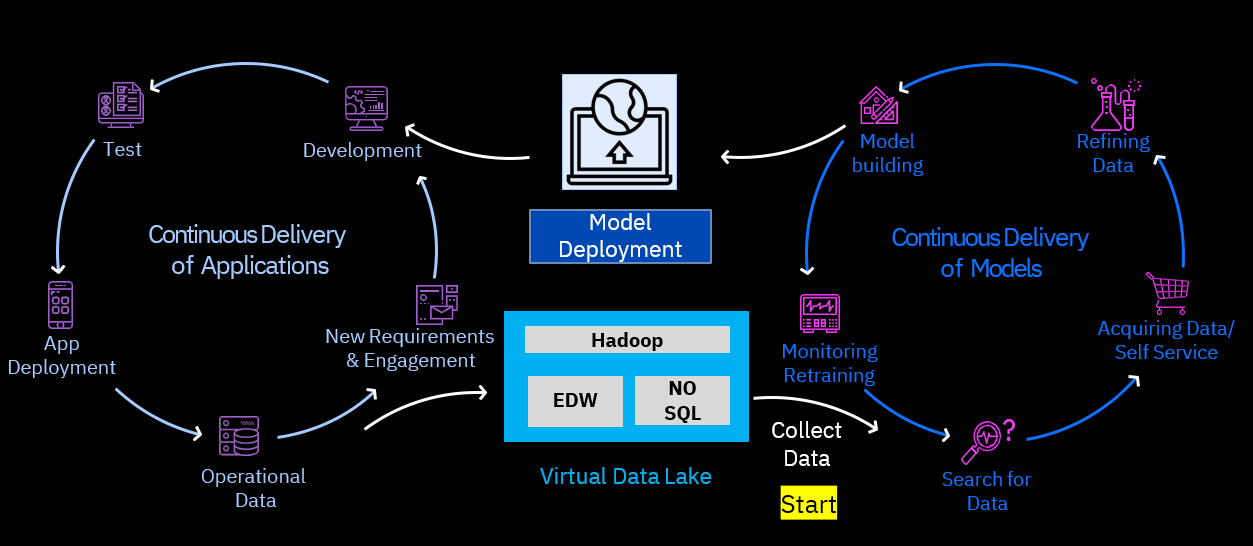
* Understand data science lifecycle
* Have at least beginner knowledge of Jupyter Notebooks
* Have at least beginner knowledge of Python.

# ModelOps Overview

**ModelOps** is a process of developing and deploying data science assets to production. An important focus of ModelOps is automation of deployment, monitoring, and governance.

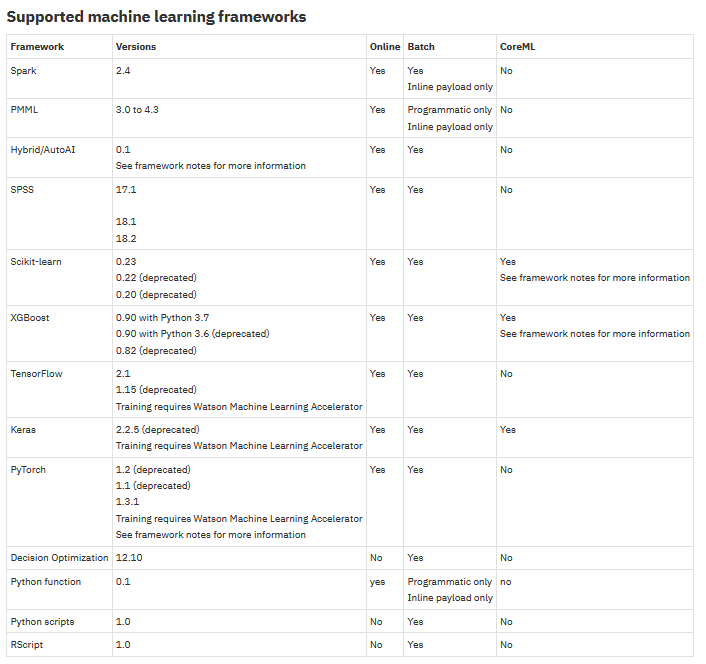
In this lab we will cover the **Model Deployment** phase of ModelOps. We chose to focus on deployment for several reasons:

* **Cloud Pak for Data (CPD)** implements enterprise-level support for deployment of data science assets.
* Model deployment can be challenging to implement when a customer doesn’t understand available features.
* Automation of model deployment is a technical topic that requires understanding of new concepts.



Deployment is the process of configuring an analytic asset for integration with other applications or access by business users. Several types of analytics assets can be deployed in CPD. The most current list of supported deployments can be found in documentation:

* **Cloud Pak for Data as a Service (CPDaaS):** <https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/pm_service_supported_frameworks.html>
* **Cloud Pak for Data:** <https://www.ibm.com/support/knowledgecenter/en/SSQNUZ_3.5.0/wsj/analyze-data/pm_service_supported_frameworks.html>



CPD includes several *services* that are used for ModelOps. The key services are:

* **Watson Studio:** contains tools for building data science assets (Jupyter Notebook, RStudio, Modeler Flows, AutoAI)
* **Watson Machine Learning (WML):** used to deploy data science assets
* **Watson OpenScale:** used to monitor deployed models.

In CPD documentation you will see references to *WML* and *WML API* because it’s the main service that’s used for deployment.

## Feature Parity and Roadmap

Currently **Cloud Pak for Data** supports more deployment options than **CPDaaS**.

Specifically, **Cloud Pak for Data** supports deployment of *Python* and *R scripts*, and *Python functions*. Functions and scripts enable:

* Deployment of pipelines: combine multiple steps that are required for scoring.
* Deployment of any model framework, even if it’s not supported “out of the box”. For example, deployment of *H20* models or a different versions of *TensorFlow*.

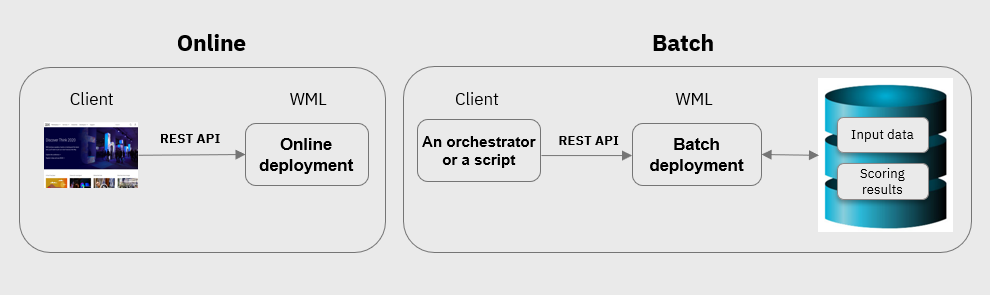
Support for script and function deployment is on the roadmap for **CPDaaS**. IBM is also working on implementing *Kubeflow* for deployment of data science assets.

## Deployment Options

CPD supports three deployment options:

* **Online**: a real time request/response deployment option. When this deployment option is used, models or functions are invoked with a REST API. A single row or multiple rows of data can be passed in with the REST request.
* **Batch**: a deployment option that reads and writes from/to a static data source. A batch deployment can be invoked with a REST API.
* **CoreML**: provides the capability to save a model for deployment in CoreML runtime. CoreML is not included with CPD.

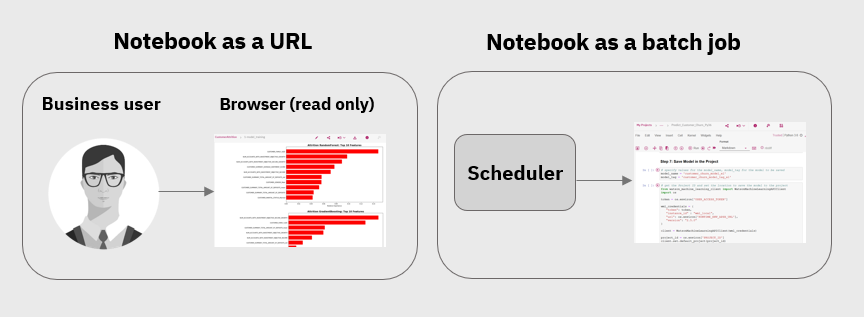
**Figure 1: Model and function deployment**



In addition to models, it’s also possible to deploy

* Notebooks as a read-only HTML page (a URL)
* Notebooks as a scheduled batch job.

**Figure 2: Notebook deployment**



In this lab we will review deployment of a *scikit-learn* model for batch and real time scoring. We will also deploy notebooks.

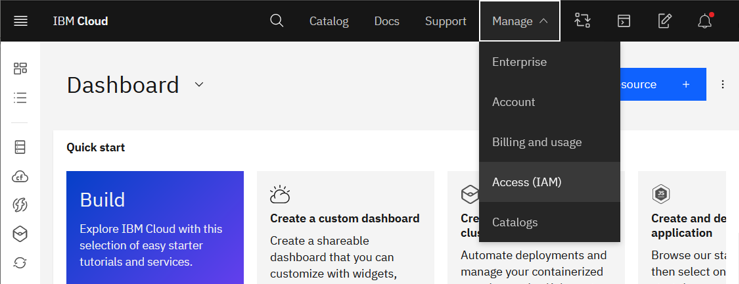
## IBM Cloud Security and WML API

Before we deploy models, we need to review several concepts related to security and architecture of IBM Cloud.

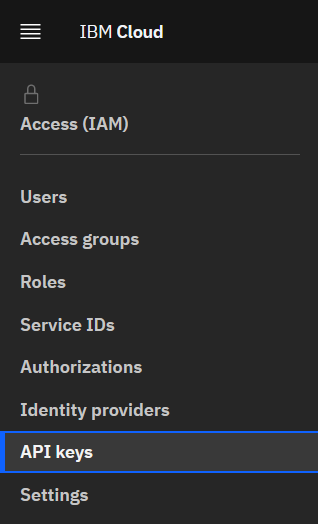
### API Key

In cloud environments access to services is managed by *API keys*. An API key is an encrypted string that identifies an application without any principal (userid). In IBM each user must create their own API key from the **IBM Cloud** console.

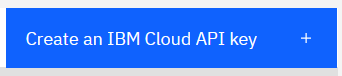
1. Log in to your **IBM Cloud** account and navigate to the **Dashboard** view.
2. Click **Manage -> Access (IAM)** from the top menu



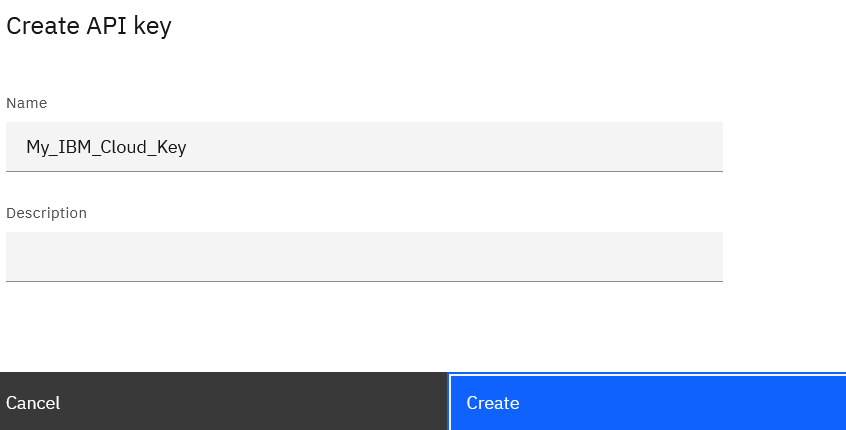
1. Select **API Keys** from the left navigation menu.



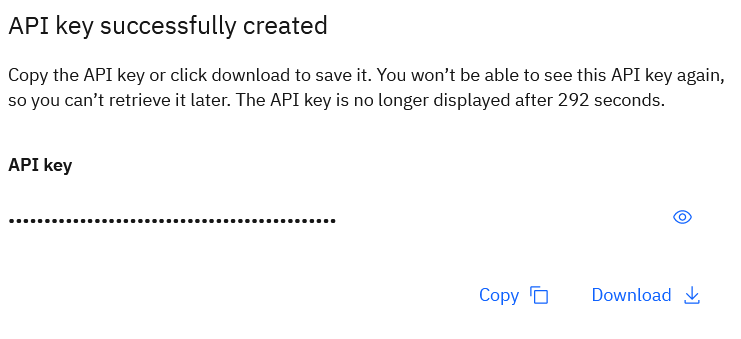
1. Click **Create an IBM Cloud API key**



1. Specify any name for the key and click **Create**.



1. Copy the API key to the notepad and download it. You will need the API key in the next section of the lab.



### Cloud Location URL

Each IBM Cloud hosting site has an assigned URL. This URL is used by various APIs, including the Watson Machine Learning API that we will use in this lab. You can look up the URL for your cloud location on this Webpage: <https://cloud.ibm.com/apidocs/machine-learning#endpoint-url>



### Watson Machine Learning Client API

*Watson Machine Learning (WML) client API* provides functions for deploying data science assets. It’s called a *client* API because it interacts with the WML *service*.

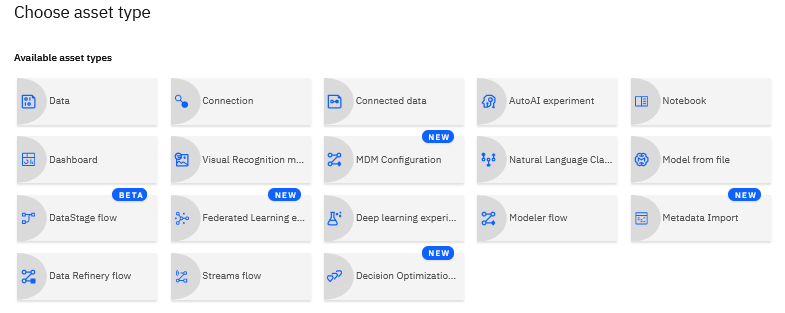
*WML client API* is implemented as a Python package, which is installed into **Cloud Pak for Data** by default. However, it’s a best practice to check for latest updates of WML client API before using it.

Functions in WML API have required parameters. In this section we will review some of the most often used parameters in WML API.

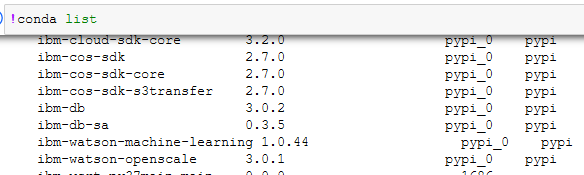
*Note: The implementation of the following steps is provided in the Introduction to WML Client notebook that you can find in the imported project.*

1. In your project create a new notebook by selecting **Add to Project** from the top menu.



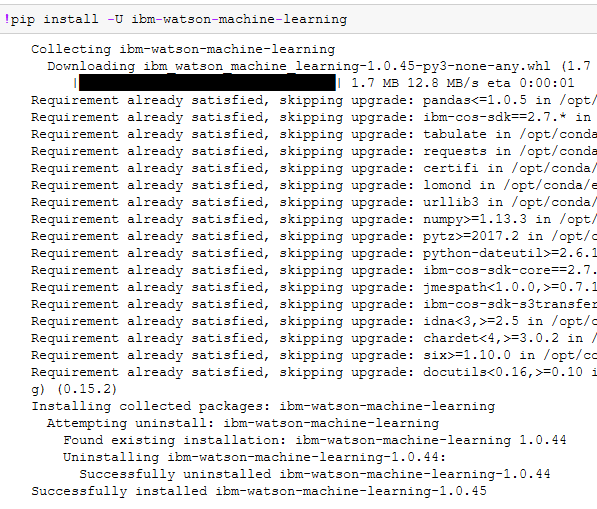


1. In a notebook cell run command: !conda list
2. This command shows all packages installed in the notebook environment. Notice that the *ibm-watson-machine-learning library* version *1.0.44* is installed.



1. Run the following command to check for updates: !pip install -U ibm-watson-machine-learning

Notice that the library was upgraded to version 1.0.45.



1. Next, we will import and instantiate the *APIClient* class.

The *APIClient* is the main class that we will use for deploying assets. To create the API client class ,we need to provide the *Cloud API key* and *URL* of *IBM Cloud hosting site (location)*. We looked up these values in previous sections - you will need to use your own API key in the code below by substituting the ‘XXXX…’ with your own key.

Write and run the following code in a new notebook cell. This code does not provide output.

# Import the APIClient class

from ibm\_watson\_machine\_learning import APIClient

# IMPORTANT

# Replace with your Cloud API key and location

api\_key = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX' #enter your key here – keep the single quotes

location = 'https://us-south.ml.cloud.ibm.com'

wml\_credentials = {

"apikey": api\_key,

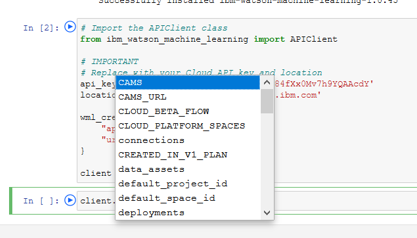
"url": location

}

client = APIClient(wml\_credentials)

1. Now that the *APIClient* object has been instantiated, we can invoke various functions.

In the next cell, type in *client.* (the name of your object followed by a period), hit the tab key, and you will see the list of available properties and functions.



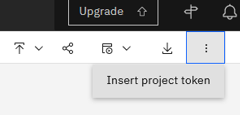
While the best way to learn about the Client API is by reviewing documentation, this option if useful for quick lookup.

Save this link for your reference:

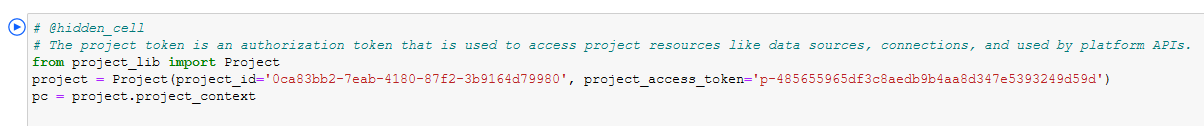
* WML Client API documentation: <https://ibm-wml-api-pyclient.mybluemix.net/>

1. WML Client requires that we set *Project id* before we save models. We look up project id by using a menu in a notebook.

From the top menu bar in the notebook select the **More** icon and click **Insert project token**.



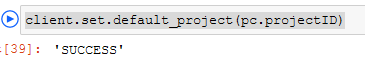
This action always generates a cell on top of the notebook. Scroll up to the top and notice that the code creates *a project context object* – *pc*. This object has a project id property, which will use to set the project id.



1. Ensure that you run the code cell which was automatically created to set the project\_id token at the top of the notebook. Once the automatically generated cell has been run, create a new cell at the bottom of your notebook and write and run the following code

client.set.default\_project(pc.projectID)

The output of running this cells should be *SUCCESS*.

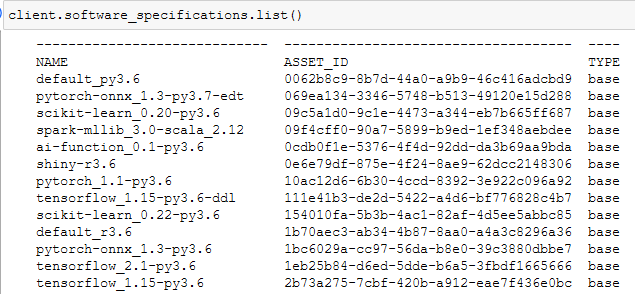


The last concept that we will review in this section is WML property called *software specifications list*. *Software specifications* refer to various frameworks that can be deployed in WML.

1. Create a new cell at the bottom of our notebook, then write and run the following code:

client.software\_specifications.list()

The output shows values that we can select when deploying data science assets.

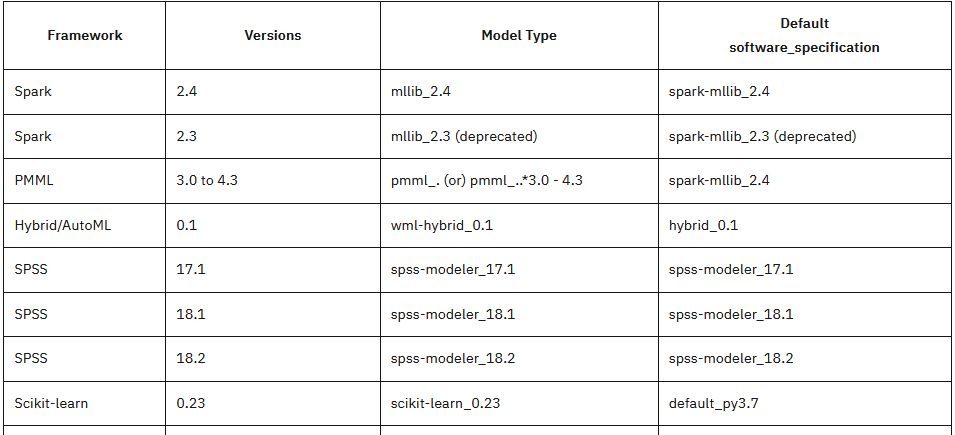


Open WML API documentation page that describes model types and corresponding software specifications: <https://dataplatform.cloud.ibm.com/docs/content/wsj/wmls/wmls-deploy-python-types.html>

Each data science modeling framework has a *Type* and a *Software Specification*. When working with WML API, these values must be used exactly as shown in the table.

For example, when saving a scikit-learn model, we must train with scikit-learn API version *0.23*, and we must use the Type *scikit-learn\_0.23* and the software specification *default\_py3.7* when deploying it with WML Client.

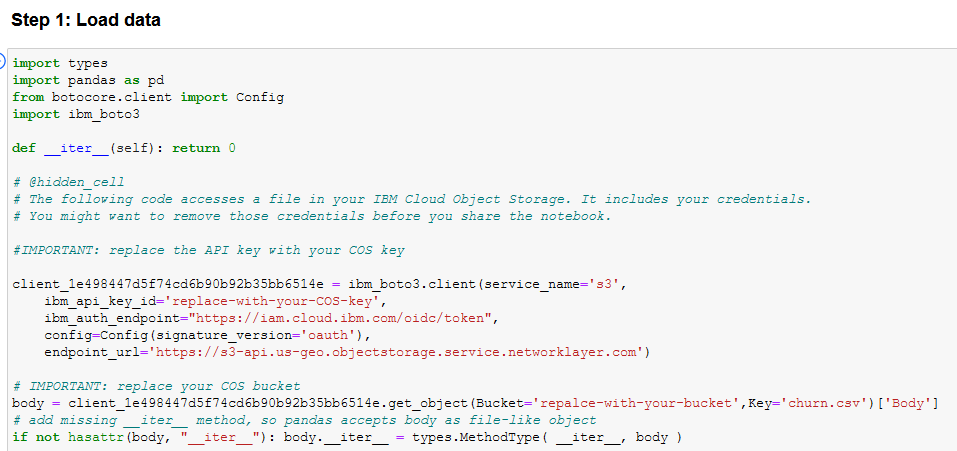
The *client.software\_specifications.list()* function that we used in this step allows us to verify and copy/paste values.



### Data access – Cloud Object Storage

In this lab we will use csv files as data sources. In CPDaaS files are loaded into **Cloud Object Storage (COS)** service instance, which is associated with your CPDaaS account. Like all services in CPDaaS, we need to access it with an API key.

CPDaaS provides code generation for data access. The sample notebooks were created using instructor’s COS, and we will need to replace these values for your COS instance.



The instructions are captured in **Appendix A – Modify Data Access**.

You don’t need to perform this task in this section. Data access changes step will be included in lab steps.

### Summary

We have finished reviewing prerequisite concepts. In this section you learned:

* How to get an *IBM Client API key* and *location* of your IBM Cloud
* How to instantiate *WML Client* object
* How to look up *software specification*, which is needed for deployment of data science assets
* How to change data access code to point to your storage.

In the next section you will use these steps to deploy a scikit-learn model for batch and real time scoring.

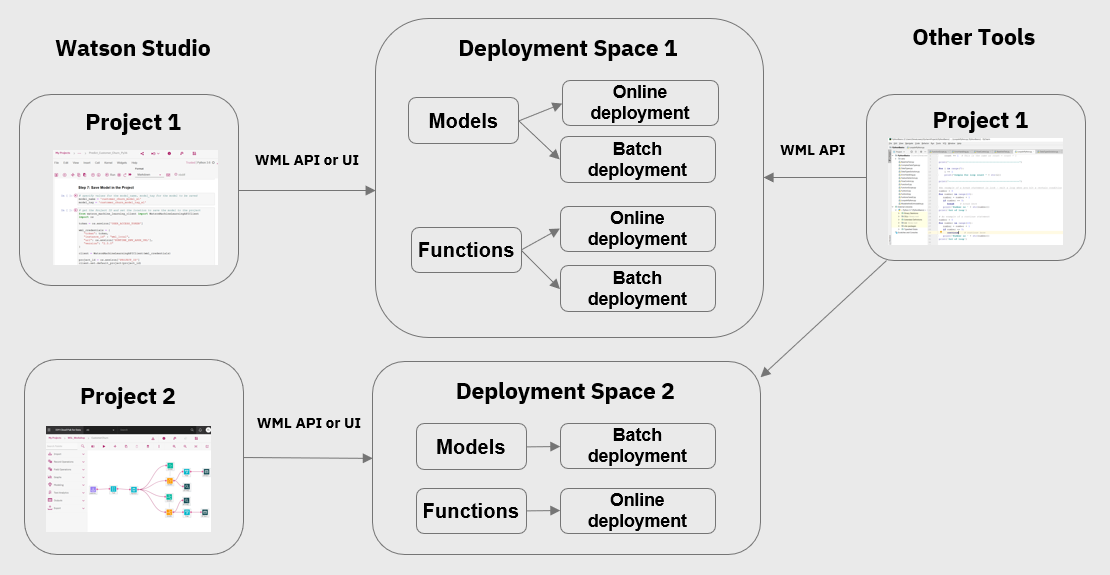
## Manual Deployment to Deployment Spaces

In this section we will deploy a model into a *deployment space*. As the name suggests, a *deployment space* is a container that’s used to organize analytics assets for deployment.

Here are some important facts about deployment spaces:

* You can work with deployment spaces via UI and via API.
* If you’re using an API, many clients can publish to the same deployment space. For example, a notebook from another project or even a tool outside of Watson Studio (for example, *PyCharm*).

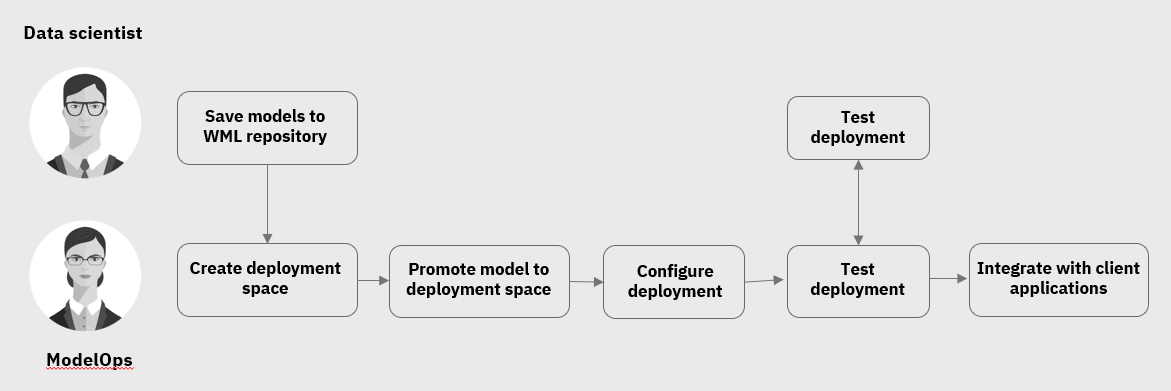
**Figure 1: Deployment Spaces and projects**



The deployment flow for most asset types consists of the following steps:

1. Create a deployment space.
2. If working through the GUI, save the asset (model, PMML, SPSS, etc.) into project repository.
3. Promote the asset to the deployment space either using GUI or APIs
4. Configure deployment (online or batch).
5. Test the deployed asset.
6. Integrate the deployed asset with another application (via REST API).

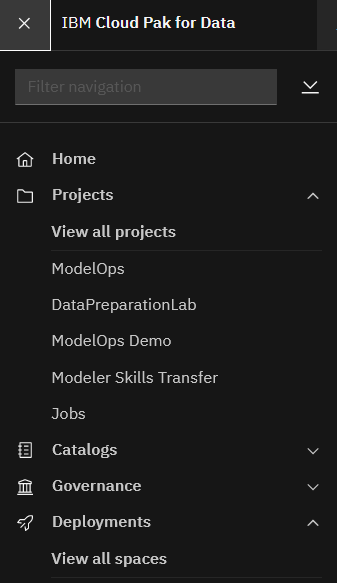
**Figure 2: Deployment workflow in Watson Studio**



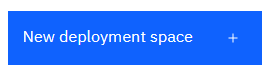
While it’s possible to complete the entire deployment workflow with *WML client API*, we recommend that you start with configuring deployment via UI. Deployment with the API can provide flexibility and automation, but if you don’t understand the implementation details of the deployment workflow, it may lead to poor governance of deployment.

We will start by creating the deployment space.

1. From **CPDaaS** main menu click **View all spaces** under **Deployments**.



1. In **Deployments** view click **New deployment space**.

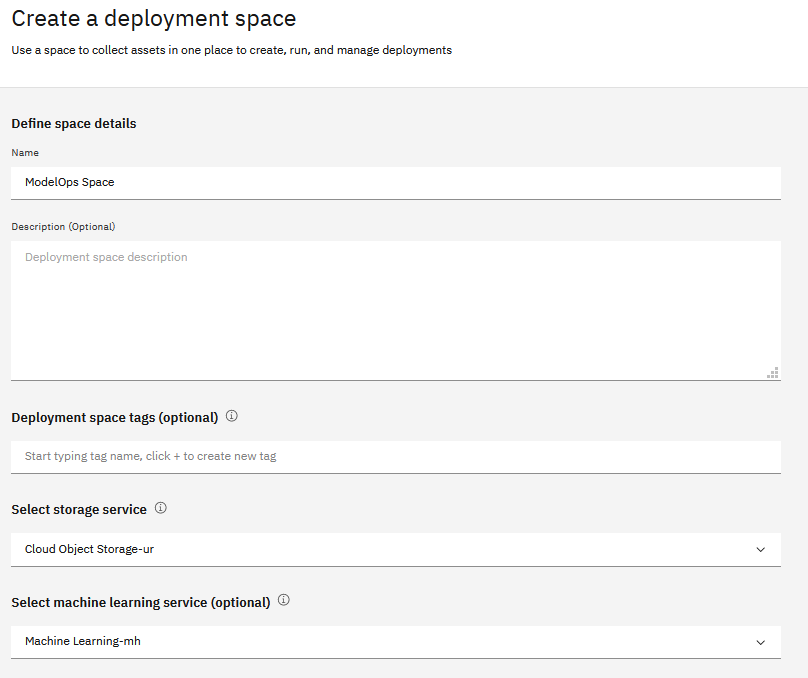


1. Provide a name, for example *ModelOps Space*. Make sure to select **storage** and **machine learning** services from dropdowns.

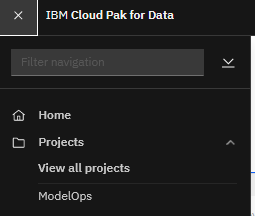
Click **Create**.

*Note: If you do not have options available in dropdowns, provision the Cloud Object Storage and Watson Machine Learning services in* ***IBM Cloud*** *and come back to this step.*

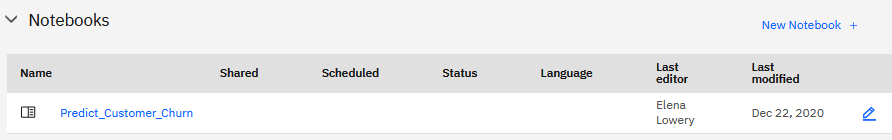
Storage service is used to store input and output files for batch jobs. Machine learning service (WML) provides runtime environments for all deployments.



After the space has been created, navigate back to the Project using the main **CPDaaS** menu.

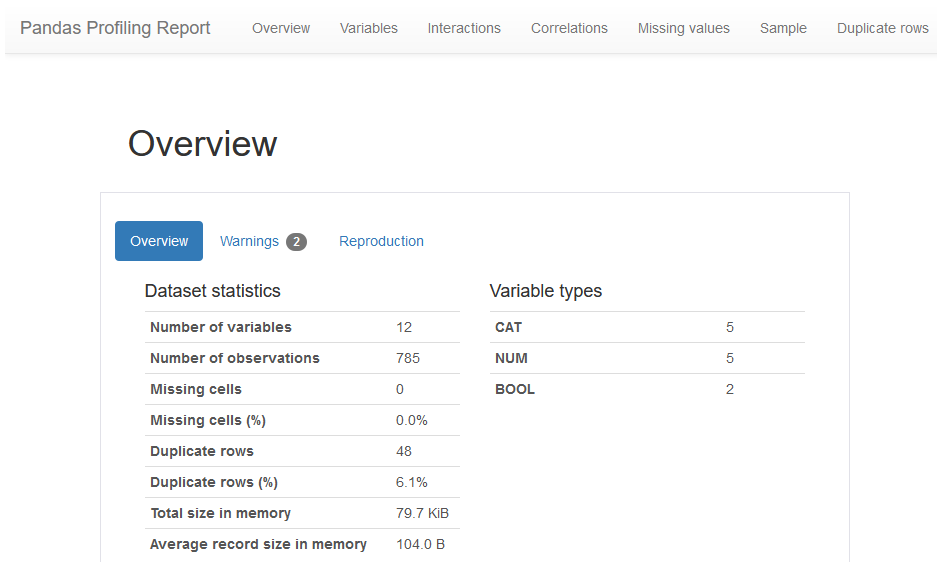


1. From the project view open the *Predict\_Customer\_Churn* notebook in *Edit* mode by clicking on the pencil icon.



1. Review the notebook.

* This notebook performs typical steps in building a classification mode for predicting customer churn: data understanding, data preparation, model building, model tuning, and model evaluation.
* The notebook includes a ***Pandas Profiling Report***, which allows you to interactively explore data.



1. Change data access to read from your COS. See **Appendix A – Modify Data Access** for more details.
2. Insert the project token: from the top menu bar in the notebook select the **More** icon (vertical ellipses) and click **Insert project token**.

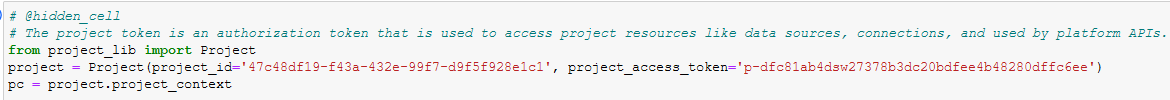
**Explanation:**

* We completed this step to instantiate the project context object (*pc*), which is used to set the project ID – a requirement for WML API.
* The notebook sets project id in **Step 7**, and while we don’t need to change this code, we do need to generate the *pc* object instantiation code each time we import the notebook into a new project because *project context* is different for each project.

This code, which sets the default project id, is the same for each project:

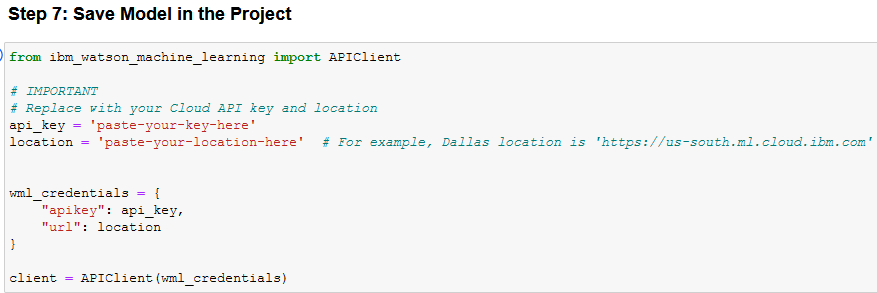


However, the *pc* object is different for each project (*project\_id* and *access token*)



1. Scroll down to **Step 7** and replace

* *Cloud API key*
* *Location (Cloud URL)*



1. Review the next cell.

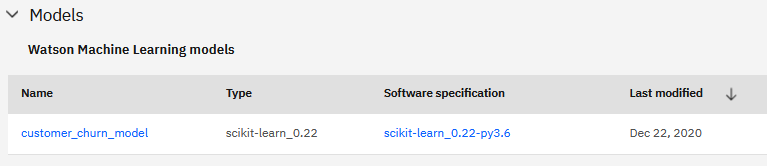
This code saves the model to the project.

* *default\_py3.7* is the *software\_specification* that’s listed in documentation for scikit-learn framework version 0.23
* *scikit-learn\_0.23* specified for the *TYPE* property is the value listed in documentation
* *pipeline* is the model that was trained in this notebook
* *X-train* and *y\_train* are the data that were used to train the model. Only schema of the data (and not the actual data) is saved.



You do not need to make any changes in this cell.

1. Next, run the notebook – either cell by cell or by selecting **Run All** from the menu.
2. Save the notebook and navigate back to the project **Assets** view. The saved model is displayed in the **Models** section.

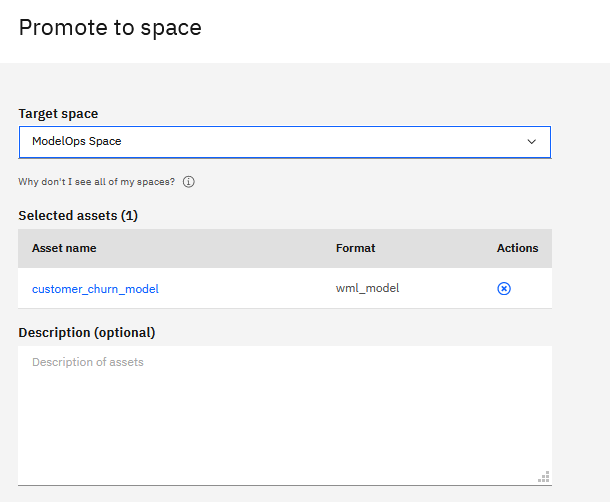


***Important Note:*** *At this time the API does not support model versioning. A different model with the same name will be saved each time you run the code to save a model. This issue will be fixed in a future release.*

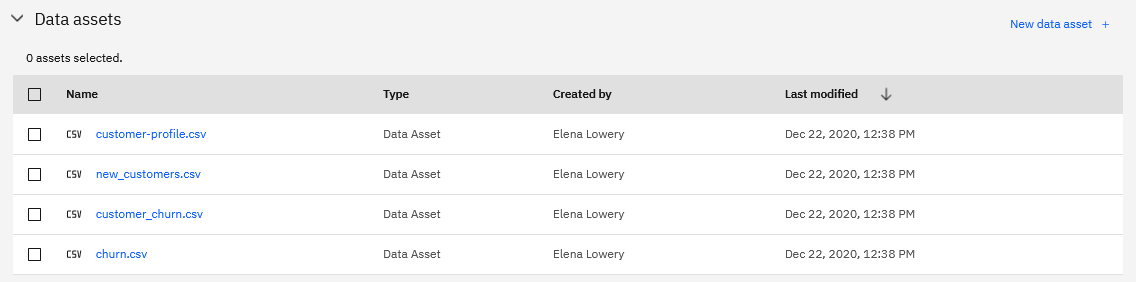
1. Click on the ellipsis next to the model and select **Promote**.



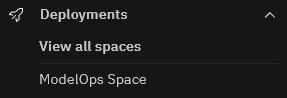
1. Select your deployment space from the ***Target space*** dropdown.



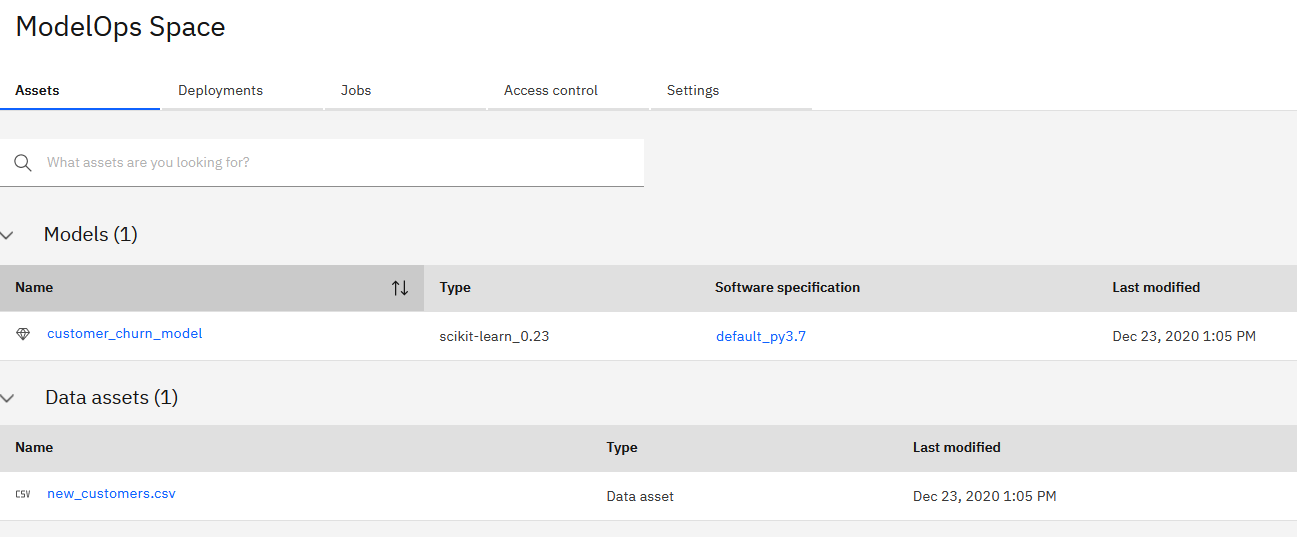
1. From the same **Project Assets** view promote the *new\_customers* data asset. We will use this file as an input dataset for batch scoring.



1. From the **CPDaaS**main menu navigate to your deployment space.

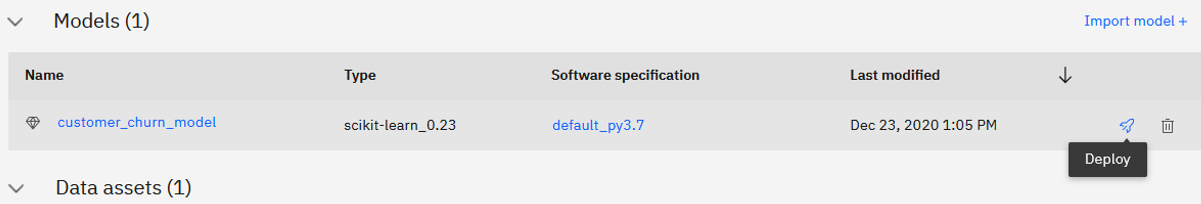


The models and data assets can now be used for configuration of deployment.



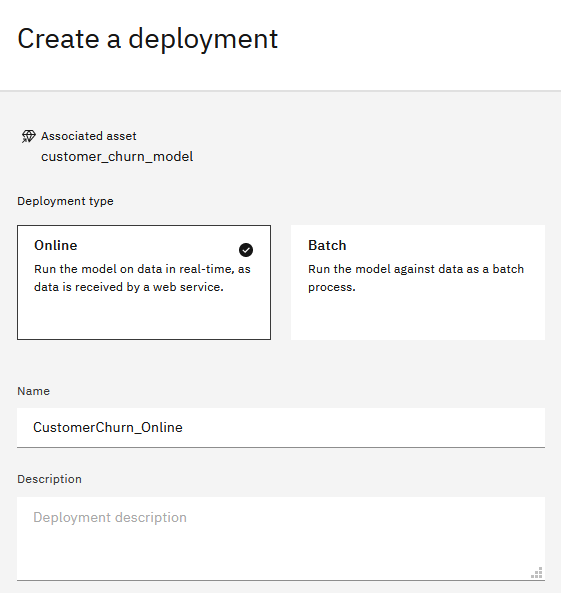
We will configure the churn model for online (real time) and batch scoring.

1. Click on the **Deploy** icon next to the model.



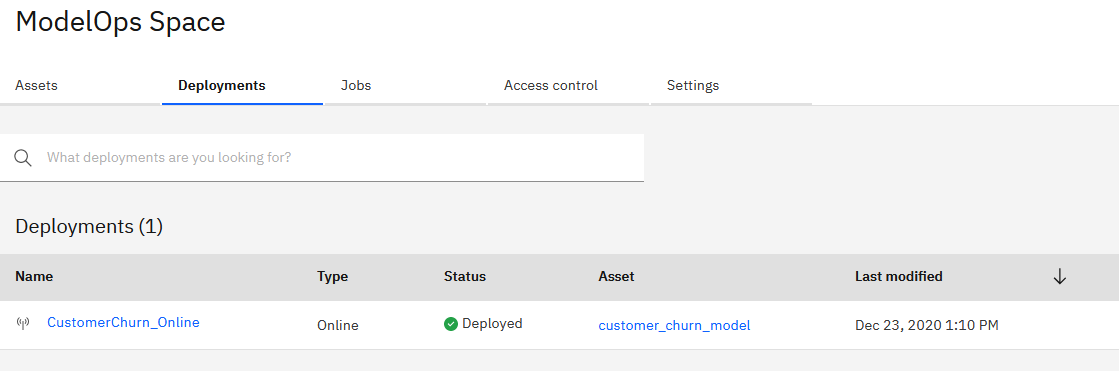
First, let’s configure an *Online Deployment*.

1. Select the **Online** box, provide deployment name and click **Create**.



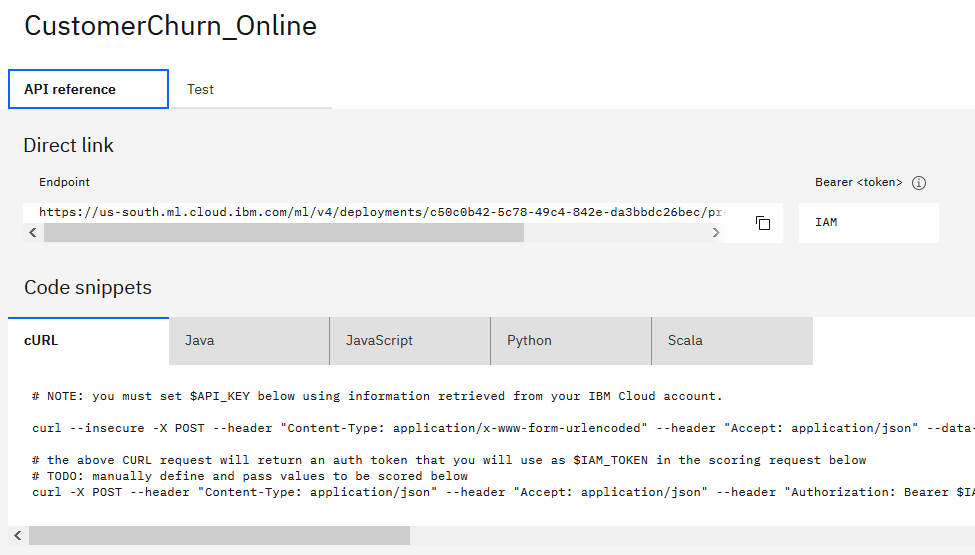
1. In the deployment space main view click on **Deployments**.

Wait till Status changes to *Deployed*.



1. Click on the deployment to display sample invocation code and the test interface.

Code samples provided on this page can be used by software engineers who need to invoke real time scoring from other applications.



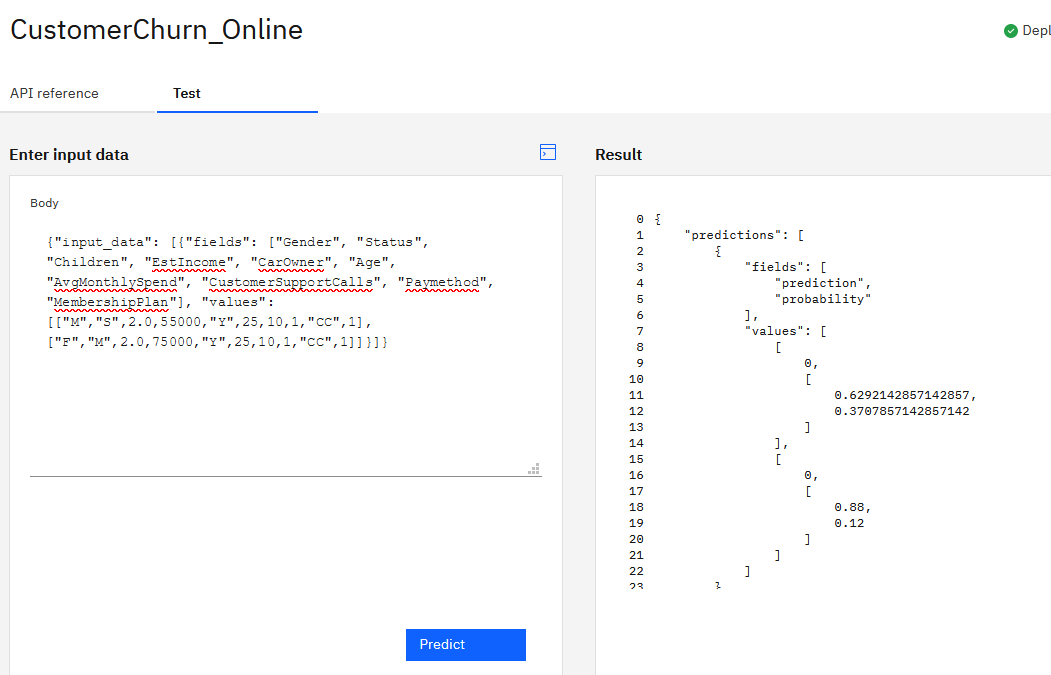
We can also test the deployed model interactively.

1. Click on the **Test** tab and enter the following values for testing:

{"input\_data": [{"fields": ["Gender", "Status", "Children", "EstIncome", "CarOwner", "Age", "AvgMonthlySpend", "CustomerSupportCalls", "Paymethod", "MembershipPlan"], "values": [["M","S",2.0,55000,"Y",25,10,1,"CC",1], ["F","M",2.0,75000,"Y",25,10,1,"CC",1]]}]}

You can copy this string from *data\SampleCustomerRecord.txt* file in of the unzipped git repo folder.

We passed in 2 records for scoring. The prediction for both is 0 (customer will not churn). Confidence in prediction is 62.9 percent for the 1st record and 88 percent for the 2nd record.



**You have finished configuring and testing an online deployment of an open source model.**

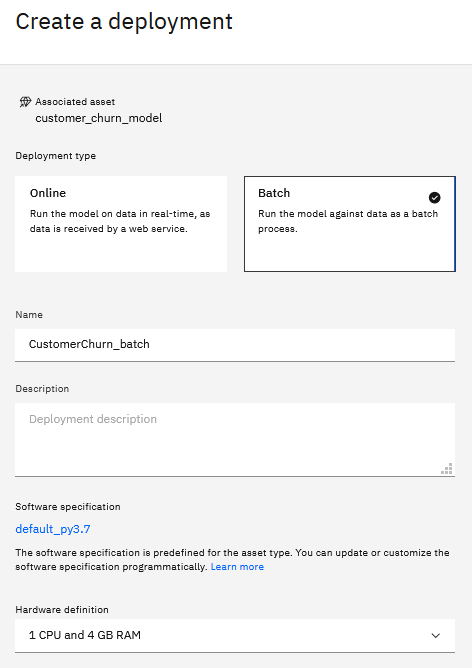
Next, we will configure batch deployment.

When configuring batch deployment input and output data assets must be in the same deployment space as the model - that’s why earlier in the lab we promoted *new\_customers.csv* to our deployment space.

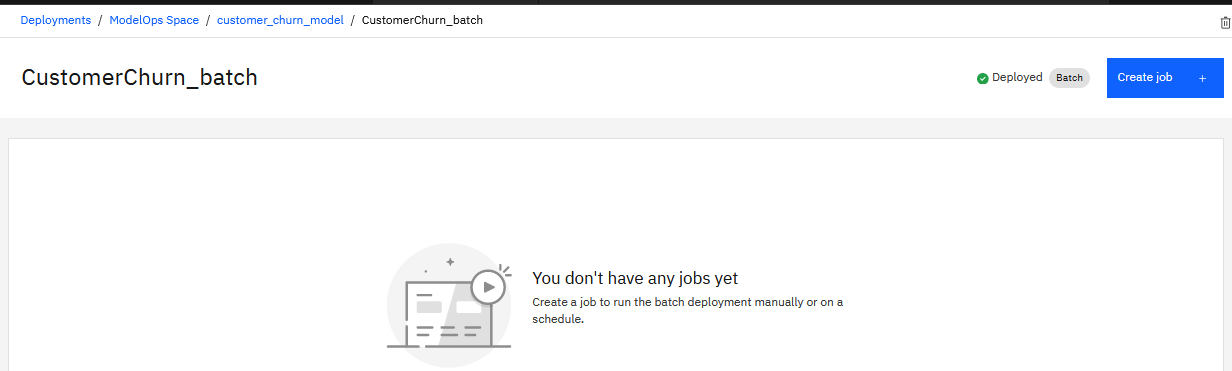
*Note: In our example we are using csv files, but database tables can also be used as input and output data sources. Similar to files, connections will have to be promoted to a deployment space.*

1. Navigate to the **Assets** page of the deployment space and click **Deploy** next to the model.
2. Select **Batch** and provide a *name* for the batch deployment. Select the *1 CPU and 4GB RAM* **Hardware Definition** and click **Create**.

Hardware definition is the hardware configuration that will be used to run this job when it’s invoked. We did not specify the hardware definition for online scoring because all online deployments run in a standard environment configuration. The number of records that need to be processed by batch jobs and required completion time vary, and that’s why it’s important to have a configurable environment for batch scoring.

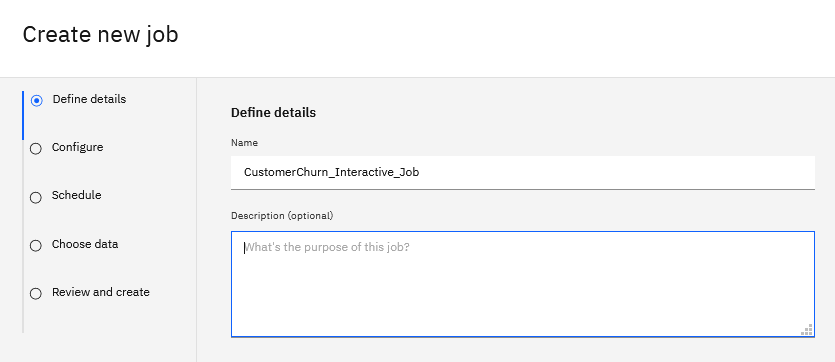


1. Click **Create Job**.



1. Provide a job name (e.g. CustomerChurn\_Interactive\_Job).

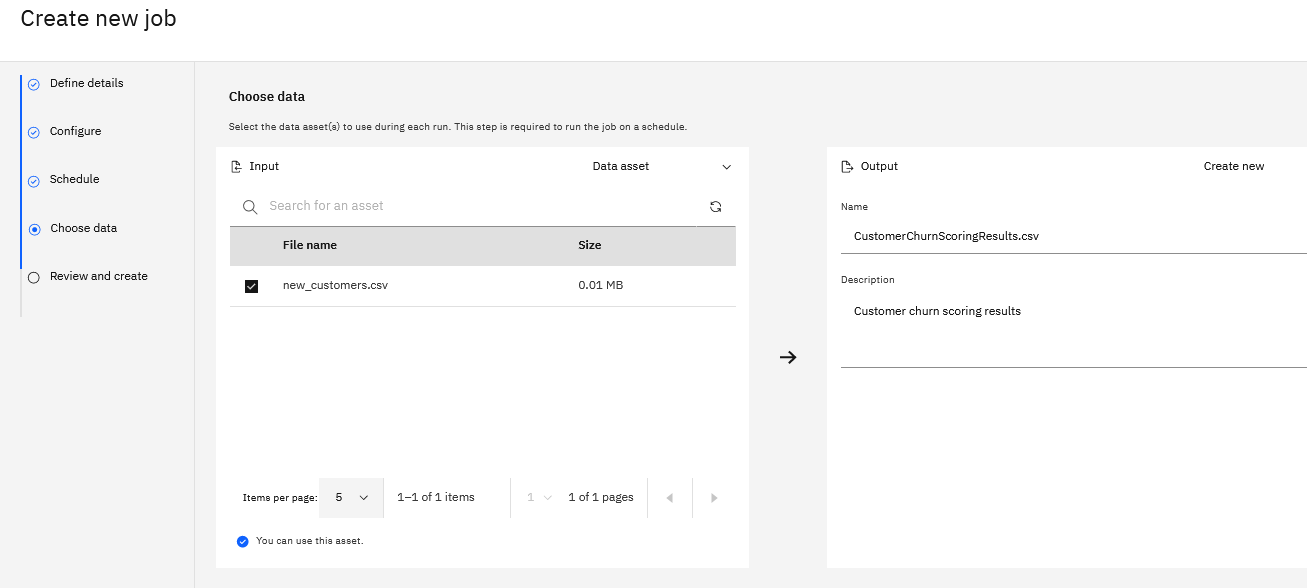
We named this job an *interactive* job because we will invoke it on demand vs. a schedule.



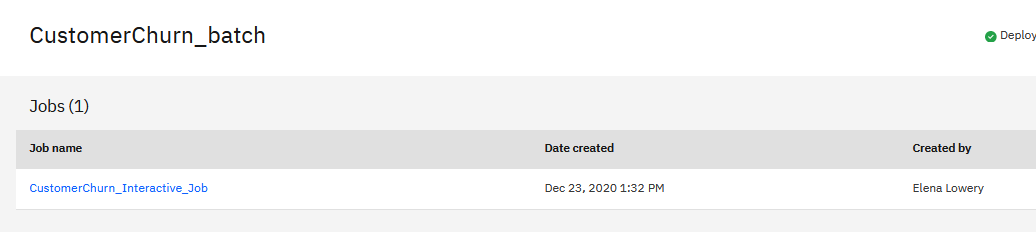
1. Click **Next** until you get to the **Choose data** screen.

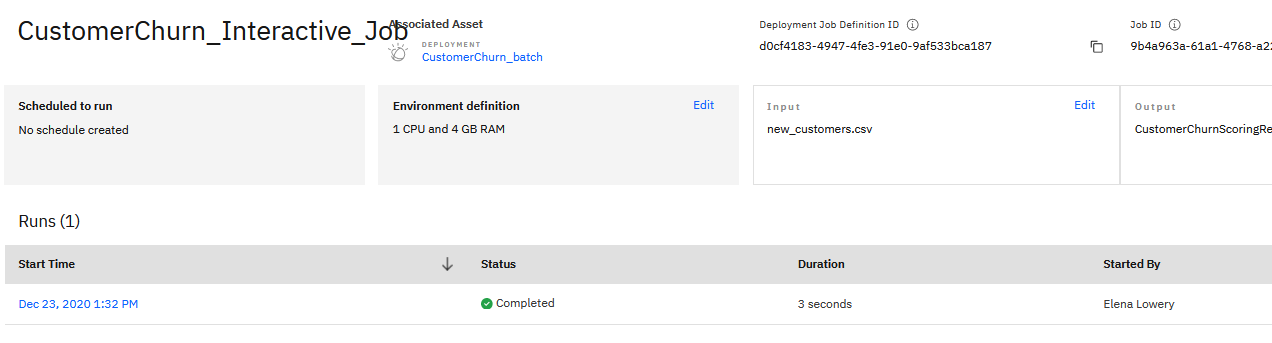
* Select *new\_customers.csv* as the **Input** file
* Provide the **Output** file name, for example, *CutomerChurnScoringResults.csv*

Click **Next**, then click **Create**



1. Click on the newly created job and notice that the run was completed.

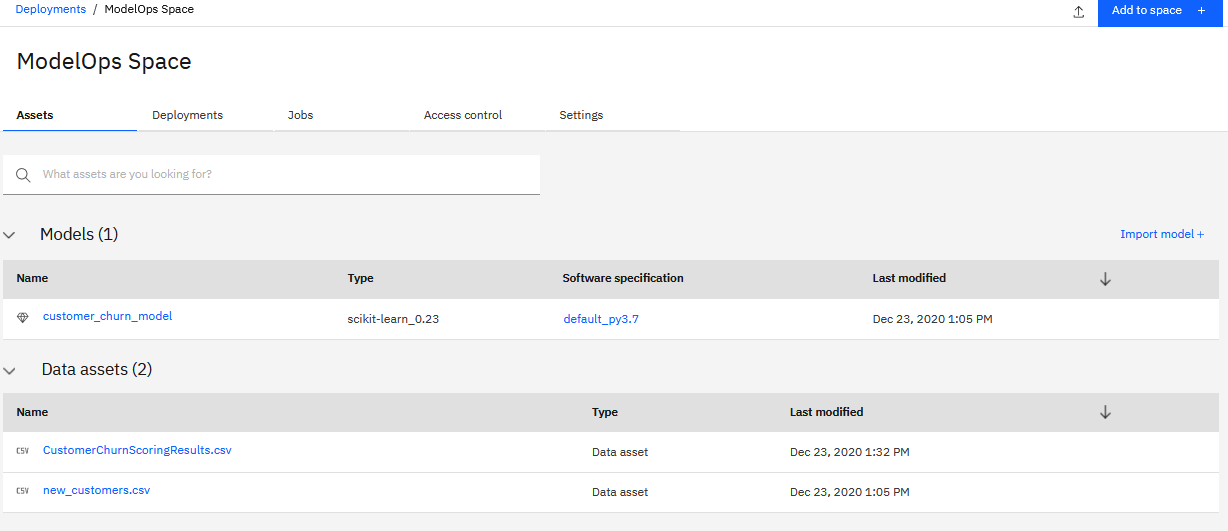




1. Navigate back to the deployment space main view.

Scoring results have been saved under **Data assets**. If you wish, download and review scoring results.

*Note: as we mentioned earlier, in most cases connections (and not files) will be used as input and output data sources for batch scoring, and therefore download of files will not be required. If files are used, it’s also possible to automate copy of files to other locations by using CPDaaS APIs.*



**You have finished configuring an open source model for batch scoring.**

## Programmatic Deployment to Deployment Spaces

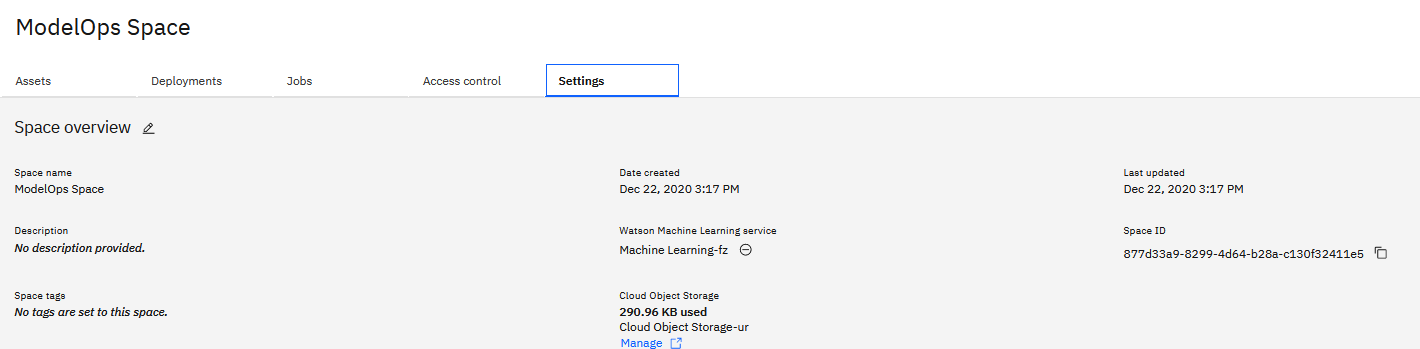
An increasing number of customers want to automate deployment of data science assets. This section shows a simplified example of automatic deployment. With *WML API* it’s also possible to integrate with a CI/CD (Continuous Integration/Continuous Delivery) server such as *Jenkins* or *Travis*.

We will complete the same deployment steps as we did manually in the previous section with *WML Client API*.

First, we need to look up deployment space id.

1. Navigate to the deployment space you created in the previous section and click on **Settings**.

Copy the *space id*.



1. In your project open the *Deployment\_with\_API* notebook in edit mode.

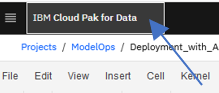
This notebook is similar to the notebook we used in the previous section – it builds a model for predicting customer churn.

1. Change data access to read from your COS. See **Appendix A – Modify Data Access** for more details.
2. Navigate to **Step 7** in the notebook and replace the following values:

* *Space id* that you copied in the previous step
* *Cloud API key*
* *Location (Cloud URL)*

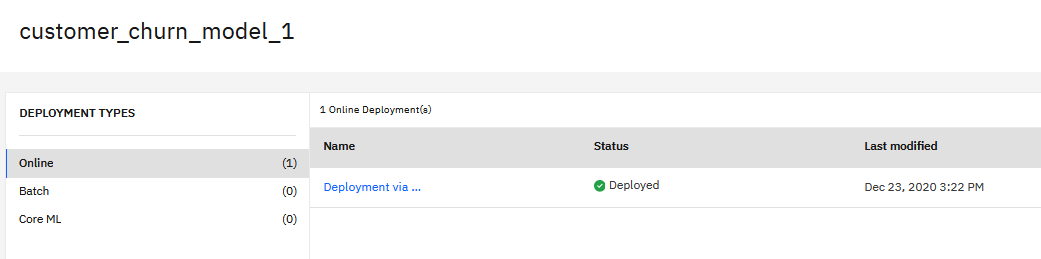


1. Prior to running the notebook, open a 2nd browser tab where you can watch the creation of deployment. You can do that by right clicking on the main Cloud Pak for Data link and selecting the **Open Link in New Tab** from your browser menu.



1. Run all cells in your notebook up to **Step 7**. As you run Steps 7 and 8, refresh your Deployment Spaces browser tab to verify that the code in the notebook published the model to the deployment spaces, and then created an online deployment.

The notebook tests the deployment. But if you wish, you can also test it manually in the UI using the same steps as in manual deployment approach.



**You have completed online model deployment with the WML API.**

## Optional: Additional practice

If you would like to practice deployment without step-by-step instructions, you can use a SparkML model that we build in the *Predict\_Customer\_Churn\_SparkML* notebook as well as examples form the **Gallery**.

### Manual Deployment

High-level instructions:

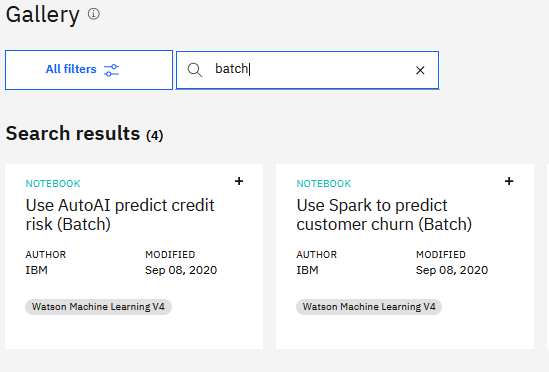
1. Open the *Predict\_Customer\_Churn\_SparkML* notebook.
2. Review the notebook and make the same changes that we’ve made in other notebooks (marked in cells starting from **Step 9**).
3. Complete the manual steps for model deployment. For *SparkML* models only online deployment is supported.

### Deployment with API

1. From the main Cloud Pak for Data menu access the **Gallery.**



1. In the **Gallery** search for *batch*



1. Add the *Use Spark to predict customer churn (Batch)* notebook to your project.
2. Review the code to create a batch deployment with WML API and replicate it in one of the notebooks you used in this lab.

# Deployment of Notebooks as a Job

In the previous section we reviewed the deployment approach that’s recommended for production environments.

In some cases, a data scientist may want to deploy assets for batch execution directly in the project, and not in a deployment space. This capability in Cloud Pak for Data is called *Jobs*.

Here are a few examples of use cases for jobs:

1. A data scientist needs to run a script to prepare data for daily analysis. The script runs at night, and prepared data is available for the data scientist in the morning.
2. A data scientist needs to prepare data for modeling. Since the data preparation task runs for a few hours, he would like to run it as a job.
3. A data scientist wants to train or retrain a model in batch mode.

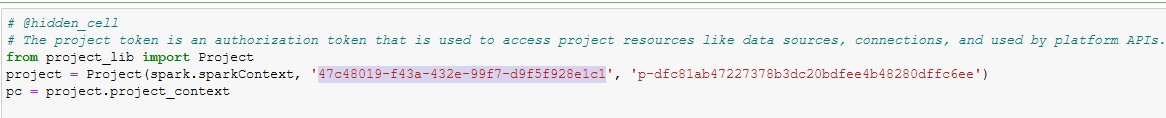
Currently **Cloud Pak for Data** supports more deployment options than **CPDaaS**. Both Cloud Pak for Data and CPDaaS support jobs for notebooks and *Data Refinery*. **Cloud Pak for Data** also supports deployment of Python scripts, R scripts, and Modeler flows as jobs.

You can find more information about jobs in documentation:

* **Cloud Pak for Data:** <https://www.ibm.com/support/knowledgecenter/en/SSQNUZ_3.5.0/wsj/manage-data/jobs.html>
* **CPDaaS**: <https://dataplatform.cloud.ibm.com/docs/content/wsj/manage-data/jobs.html?context=analytics&audience=wdp>

In this lab we will create a job that uses a notebook to train and save a model into the project.

1. Navigate to your project and open the *Predict\_Customer\_Churn\_SparkML* notebook in edit mode.
2. Change data access to read from your COS. See **Appendix A – Modify Data Access** for more details.
3. Insert project token and copy the *project id.*



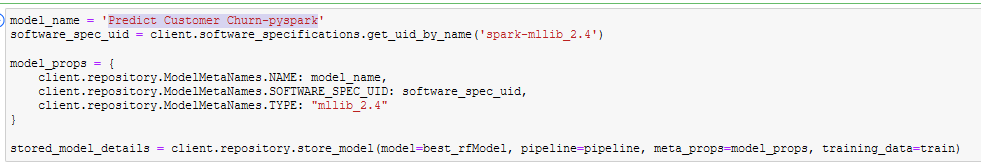
1. Scroll down to **Step 9** and replace

* *Project id*
* *Cloud API key*
* *Location (Cloud URL)*

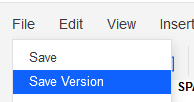


1. Note the name of the model and if you wish, change it (also in **Step 9**).

When we run the notebook in batch mode, this model will be added to the project.



1. Jobs run a specific version of the notebook, therefore save and version the notebook before creating the job. From the **File** menu choose **File -> Save Version**



1. Click on the **Jobs** icon in the top menu bar.

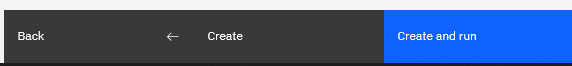


1. Step through the job wizard and provide required information.

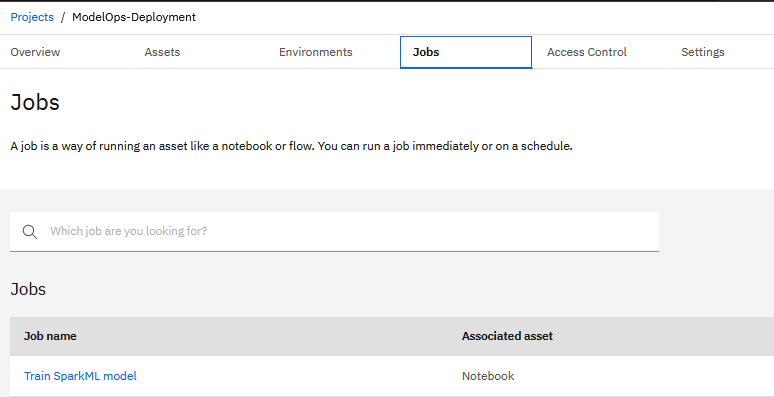
* **Name**: choose any name
* **Configure**: keep the default settings
* **Schedule**: keep the default settings.

|  |  |
| --- | --- |
|  |  |

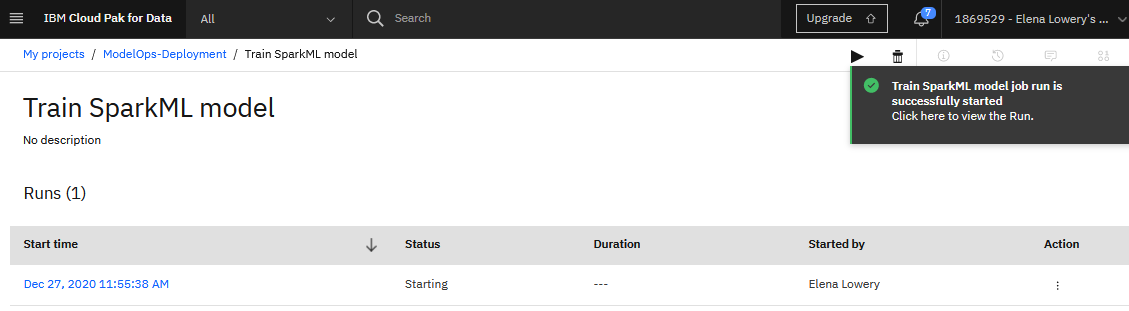
1. Click the **Create** button.



1. Navigate to the project view and notice that you do not yet have the *SparkML* model in the **Assets** view.
2. Click on the **Jobs** tab.

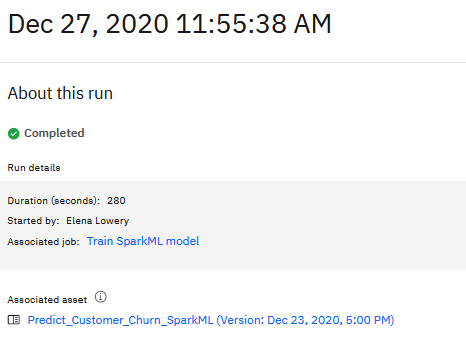


1. Click on the job, then the **Run** icon in the job view.

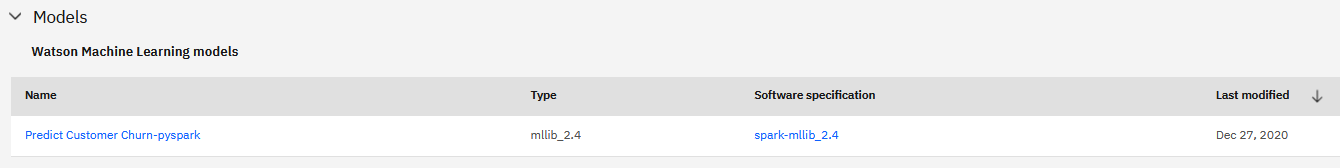


1. Click on the **Run** details to monitor the status.

*Note: this job runs for 4-5 minutes.*



1. Navigate to the project and verify that the model has been created.



Jobs functionality is also available via API. Some tasks that can be completed with the **Jobs API** are:

* Create jobs
* Invoke jobs
* Monitor job status.

These functions provide flexibility in implementing automation of various tasks in the data science lifecycle. For example, we can use the Jobs API to automate the model retraining process after historical data has been refreshed.

The Jobs API is a part of **Watson Data API**. You can find more information about the Jobs API in documentation: <https://cloud.ibm.com/apidocs/watson-data-api-cpd#jobs-list>

**You have completed deploying a notebook as a batch job.**

# Deployment of Notebooks as a URL

Many data science projects focus on data exploration and data understanding, not just on model building and deployment. One of the main advantages of Jupyter notebooks over Python/R scripts is the ability to add formatted text (markdown) and visualizations. Data scientists often need to share notebooks with their colleagues and business users.

While notebooks can be downloaded and shared as files, an easier and more secure way to share notebooks is by publishing it as a *URL*. In Cloud Pak for Data, when a notebook is published as a URL, it’s loaded as a static HTML page. The viewer will be able to look at explanations and visualizations, but they will not be able to run code cells.

This approach is similar to “publishing a report”. If the notebook needs to be used interactively, then it should be shared by adding collaborators to the project.

There are several use cases for publishing a notebook as a URL:

* Present results of data analysis
* Provide transparency in model building: show feature importance, model parameters, model evaluation
* Document results of several model training options for review by a data science team before the final approval.

In this section you will learn how to publish a notebook as a URL.

1. Navigate to your project and open the *Customer\_Data\_Analysis* notebook in edit mode.

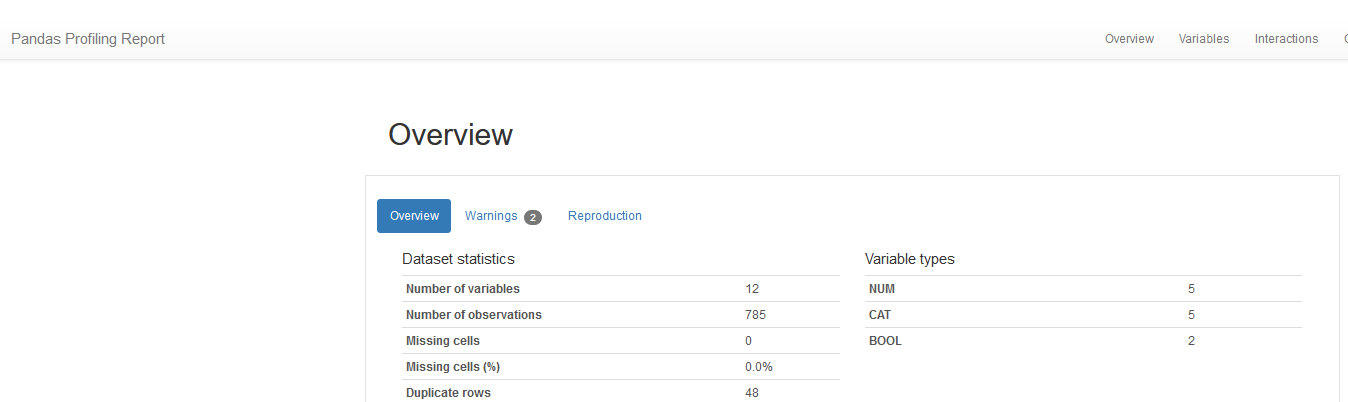
This notebook loads data that’s used for building customer churn model and creates a pandas profiling report that can be used by data scientists to understand the data.

We need to run the notebook before publishing it because we need to generate the pandas profiling report. While we can run all cells and immediately publish the notebook, the report will look nicer if we remove the output of some code cells.

1. Change data access to read from your COS. See **Appendix A – Modify Data Access** for more details.
2. From the notebook menu select **Cell -> Run All**.
3. From the notebook menu select **Cell -> All Output -> Clear**.
4. Run just the last cell again.

**

You should see the pandas profiling report under this cell.



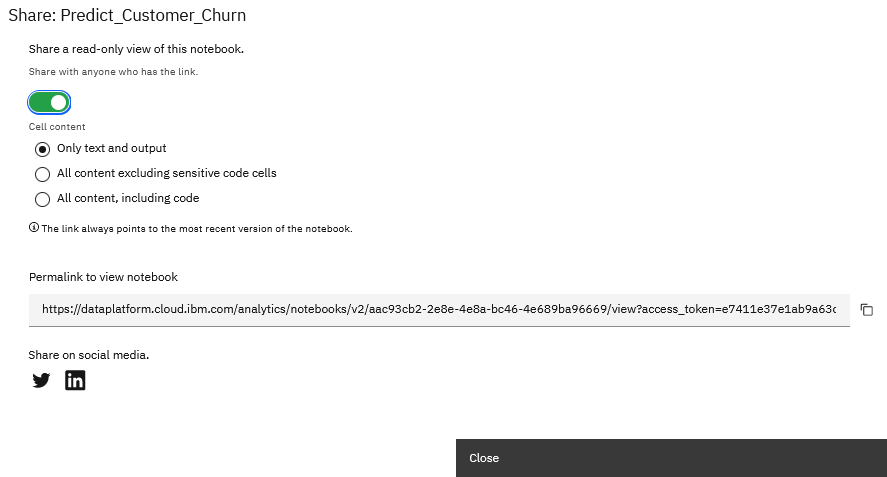
1. From the notebook version select **File -> Save version**.

*Note: it’s important to select Save version, and not just Save.*

1. From the top menu bar click the **Share** icon.

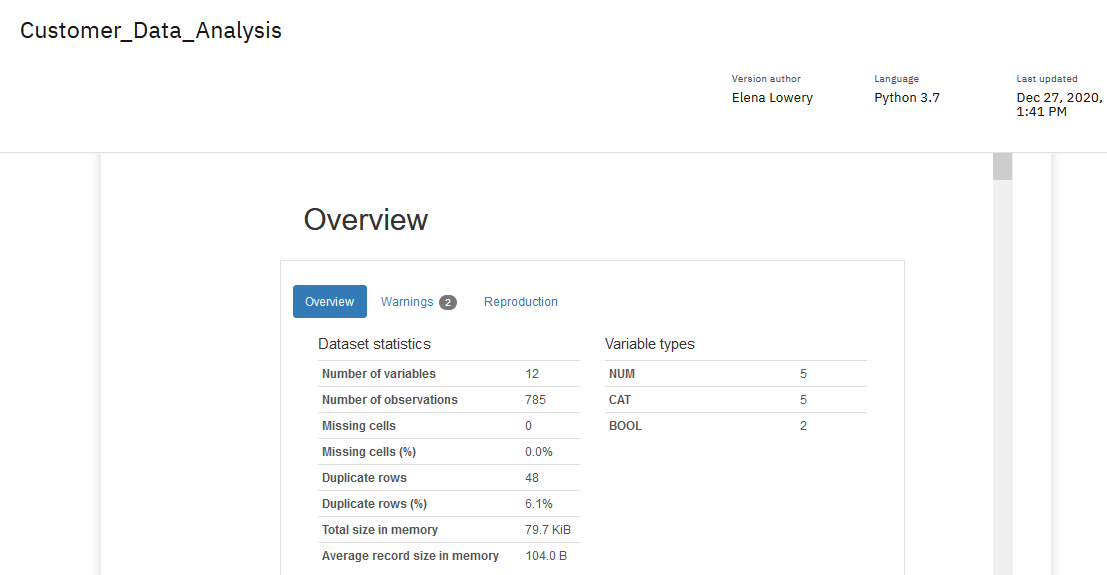


1. Turn on sharing and select the **Only text and output** option.



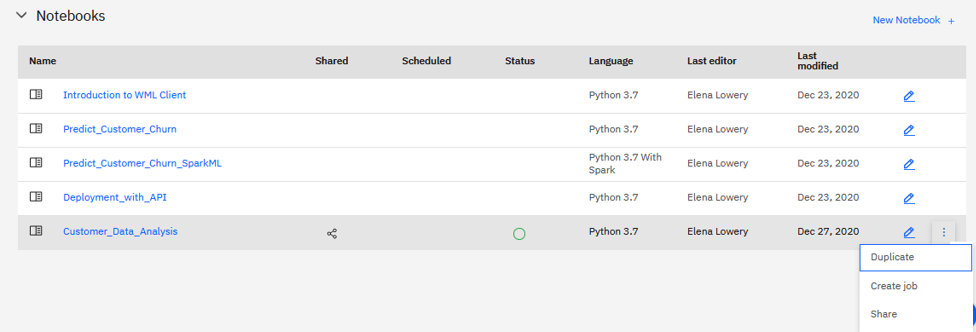
1. Copy the permalink and paste it into another browser window.

Users with this link can now view the pandas profiling report.

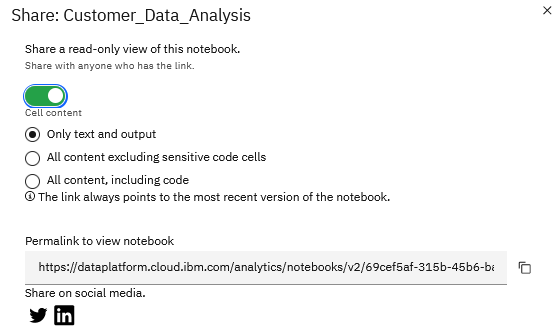


1. Navigate back to the project assets view.

We now have the *share* icon next to the notebook. Clicking on the icon will bring up the configuration screen on which you can disable sharing, change sharing settings, or copy the URL.



Note the comment in the menu – the sharing option always displays the latest version of the notebook. If you’re concerned about unplanned changes to this notebook, make a copy of the notebook or lock it.



**You have finished the task of publishing a notebook as a URL.**

# Conclusion and Next Steps

In this lab you learned how to complete the following ModelOps tasks in Cloud Pak for Data:

* Manual deployment of models
* Automatic deployment of models
* Deployment of notebooks as Jobs
* Deployment of notebooks as a URL.

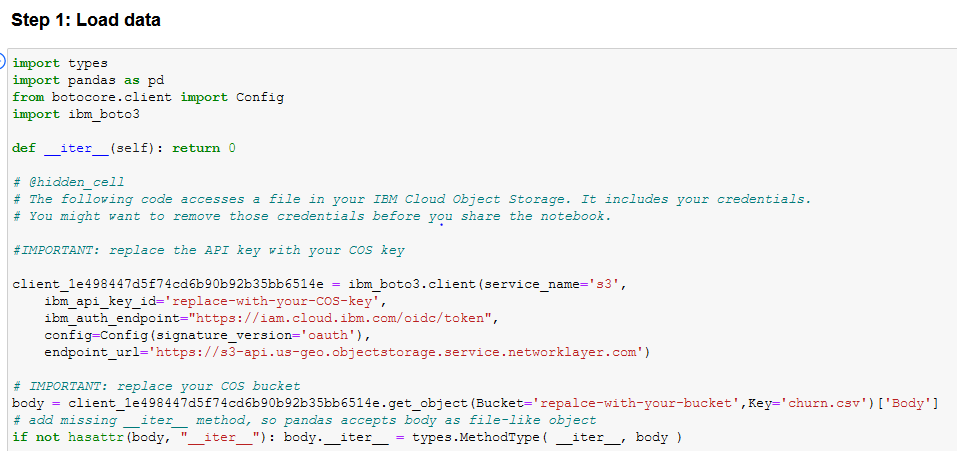
After classification and regression models have been deployed in Cloud Park for Data, they can be monitored by **Watson OpenScale**. You can learn more about these capabilities in the **ModelOps -** **Model Monitoring on Cloud Pak for Data** lab.

# Appendix A – Change Data Access

In this section you will learn how to change data access code to read files from your Cloud Object Storage (COS) service.

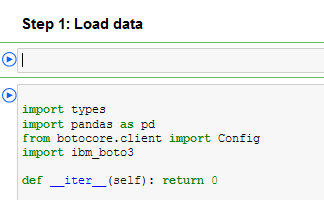
1. Open the notebook in edit mode.
2. Locate the step that loads data, usually **Step 1**.

Cells that load files from COS contain references to an IBM API key, endpoint, and a COS bucket.

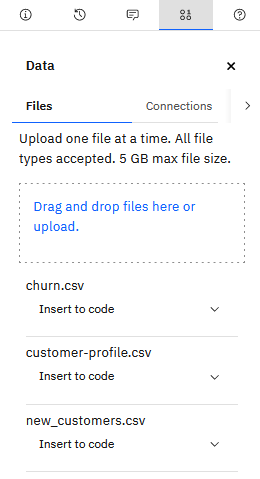


1. Insert a cell above the first “load data” cell and position the cursor in it.

**Important**: if you don’t position the curser in a new cell, it may insert code in the wrong place and break the notebook.

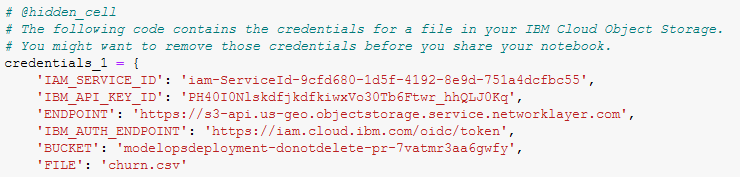


1. In the top menu click on the **Data** icon

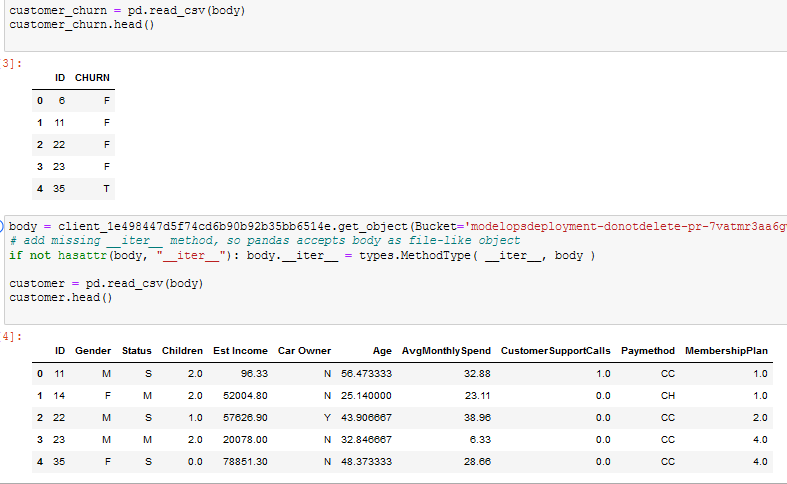


1. Select **Insert to code** -> **Credentials** for any data source, for example, churn.csv.
2. From the generated cell copy the **API key** and the **bucket** values and paste them into **each** cell that reads data. In most sample notebook we read in at least 2 CSV files.

Double check if the **endpoint** is the same. If not, replace it.



1. Run cells from the beginning of the notebook through the cells that read data to make sure it has been read in correctly.



1. Delete the credentials cell that you previously inserted.
2. Save the notebook and continue with the next steps in the lab.
3. If you’re working with SparkML (Predict\_Customer\_Churn\_Spark\_ML) notebook, instead of bucket property, you will need to change the COS URL setting (paste the Bucket value into COS URL) in **each cell** that reads data.

The bucket property is copied from the same generated code: **Insert to code** -> **Credentials** for any data source, for example, churn.csv.

You also need to change the COS API key.

