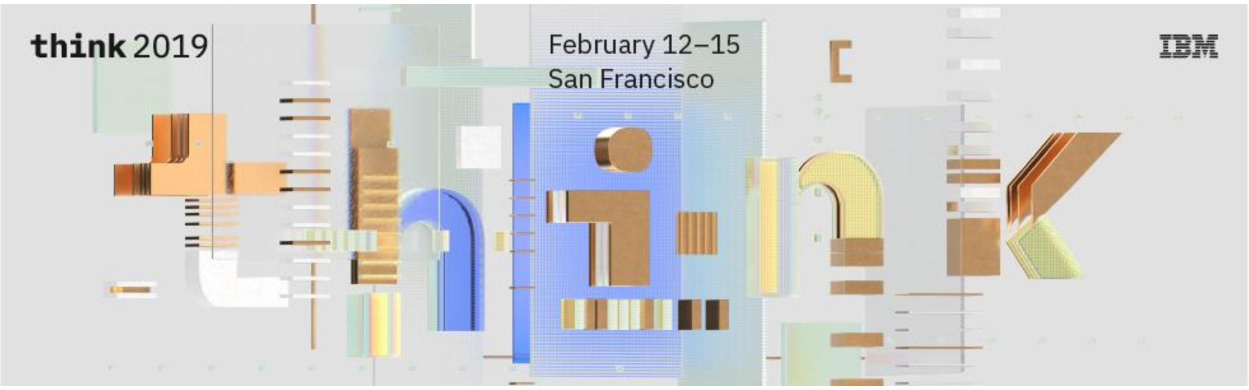


Lab Center – Hands-on Lab

Session 3308

Session Title Hands-On Lab: Deploy Machine Learning Models in Front-End Applications

(Latest PDF will be available in the public github repository : https://github.com/vishkamat/thinlab_3308)

Vishwanath Kamat, IBM, vkamat@us.ibm.com

Qi Jun Wang, IBM, wangqij@us.ibm.com

Table of Contents

Disclaimer

IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM’s sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results like those stated here.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts.
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or**

implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Introduction

In this lab you will learn the lifecycle management of a Machine Learning (ML) model from OLTP applications perspective. Many front-ending applications such as CRM (customer Relation Management), Order fulfillment, Reservation Systems etc could be infused with machine learning model to augment human interaction with customers, systems and business processes. The lab is divided in 3 sections:

Section I – Build and deploy ML Model

- In this section you will learn the typical process used by data scientists to explore and build machine learning model
- Train and test the ML model
- Finally deploy the ML model to be used as web service container

Section II – Work with interactive application

- Review python program that simulates an OLTP application used by a Customer Service Support (CSS) Representative
- Run the application as legacy without ML model
- Modify application to embed ML model
- Visualize the impact of ML model in the application

Section III (optional) – Package application as docker container and deploy

- Setup application to be deployed as docker container
- Build the docker image for application
- Deploy the application docker

Section I

1. Setup Analytics Project

In this section we will:

Build analytical model using SparkML

- Setup Analytics Project in ICP for Data Platform
- Setup data source to be used by Jupyter Notebook
- Build ML model using TelcoChurn notebook and save the model for further testing and evaluation
- Deploy TelcoChurn ML model for online scoring process

Pre-requisites

- IBM Cloud Private for Data instance is already installed and available
- A relational database (such as Db2) already created and available
- Previously created analytics project. Download from https://github.com/vishkamat/thinlab_3308/blob/master/ICP4DTelcoChurn.zip Or download this from Box folder https://ibm.box.com/s/xzzfz0v4k2okfc6tnf9zdskeri8pxu9f
- Rename file after download to your laptop for e.g.: ICP4DTelcoChurnUser7.zip
- Download application python project file: https://ibm.box.com/s/ljtbu2rfktculw46j7udesqvkmtt85q9

Connections and URLs needed for the lab

ICP for Data URL :

your URL will be different, see instructor/sheet provided

https://services-emea.skytap.com:xxxxx/zen/

Login id : <please use sheet provided>

Password : passw0rd

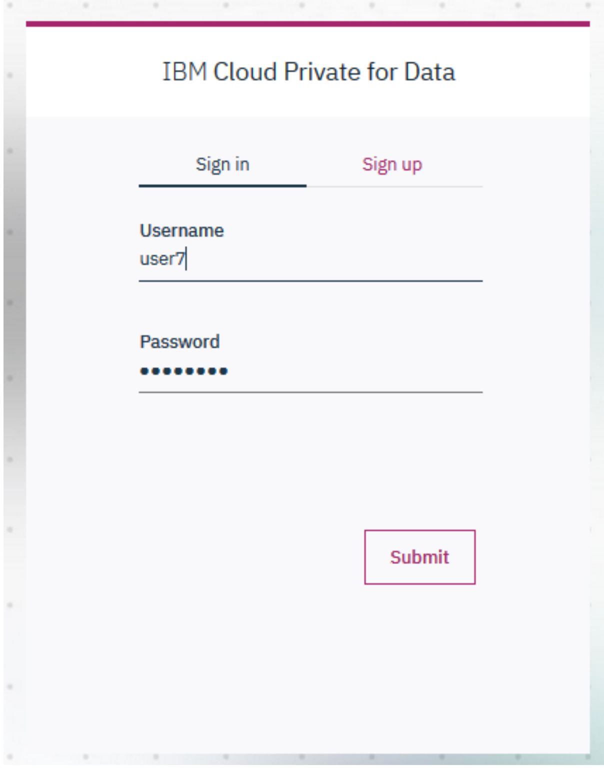
DB2 instance JDBC access parameters

jdbc:db2://dashdb-entry-yp-dal10-01.services.dal.bluemix.net:50000/BLUDB

userid : dash100050

password : oULiV__78Kej

1.1. Access ICP for Data end user web portal using your specific URL provided.
For example : https://services-emea.skytap.com:xxxxx/zen/

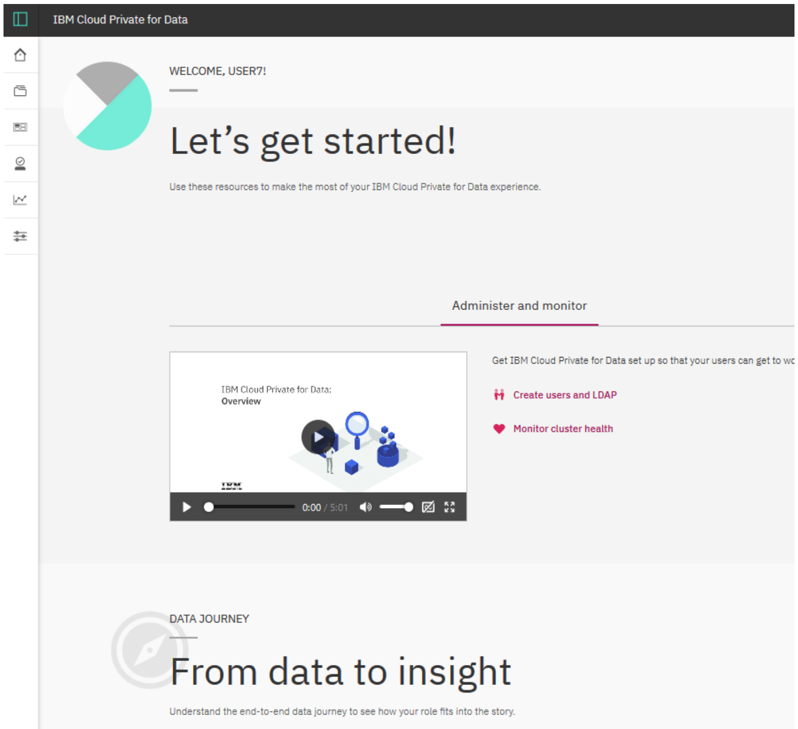


IBM Cloud Private for Data

Username
user7

Password
•••••••••

Use your assigned userid and password to login.
On successful login, you will see following landing page.



WELCOME, USER7!

Let's get started!

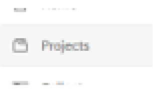
Use these resources to make the most of your IBM Cloud Private for Data experience.

Get IBM Cloud Private for Data set up so that your users can get to wo

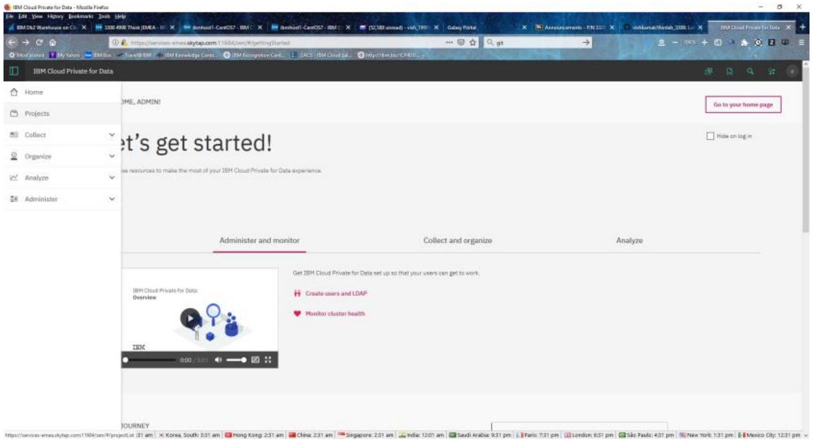
DATA JOURNEY

From data to insight

Understand the end-to-end data journey to see how your role fits into the story.



1.2. Click on "Folder icon" for **Projects** on left-hand side vertical menu

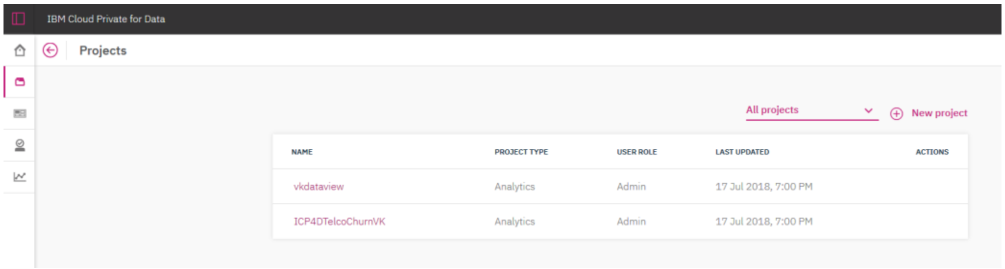


et's get started!

se resources to make the most of your IBM Cloud Private for Data experience.

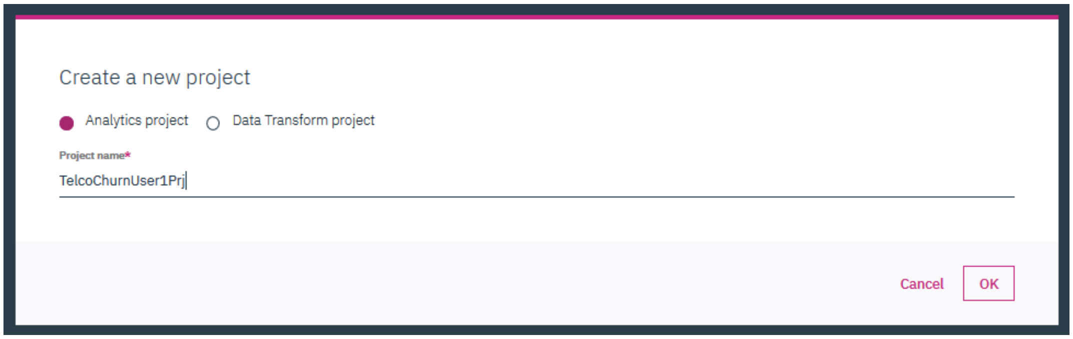
Get IBM Cloud Private for Data set up so that your users can get to work.

1.3. Click on **New Project**



NAME	PROJECT TYPE	USER ROLE	LAST UPDATED	ACTIONS
vkdataview	Analytics	Admin	17 Jul 2018, 7:00 PM	
ICP4DTelcoChurnVK	Analytics	Admin	17 Jul 2018, 7:00 PM	

1.4. This will bring option to create a Analytics project or Data Transform project



Create a new project

● Analytics project ○ Data Transform project

Project name*

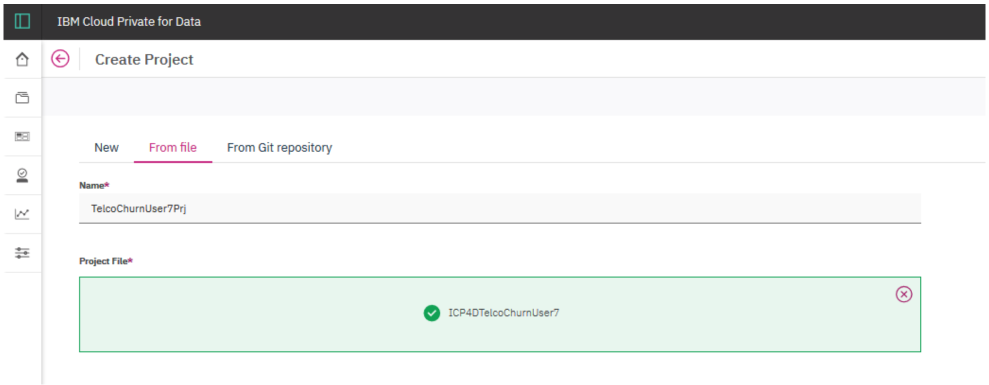
TelcoChurnUser1Prj

1.5. Type TelcoChurn**<your userid>**Prj and click OK (replace with your own userid provided). This will be name of you project.

1.6. This will bring "Create Project" screen. If you have not downloaded the Project file already, please see Pre-requisite section above for the link to download. After download rename downloaded file to (ICP4DTelcoChurn**<your userid>**.zip

1.7. To **import** project select **'From file'**. drag and drop or browse for the Zip file

you downloaded earlier (ICP4DTelcoChurn<**Your UserID>**.zip).



Name*

TelcoChurnUser7Prj

Project File*

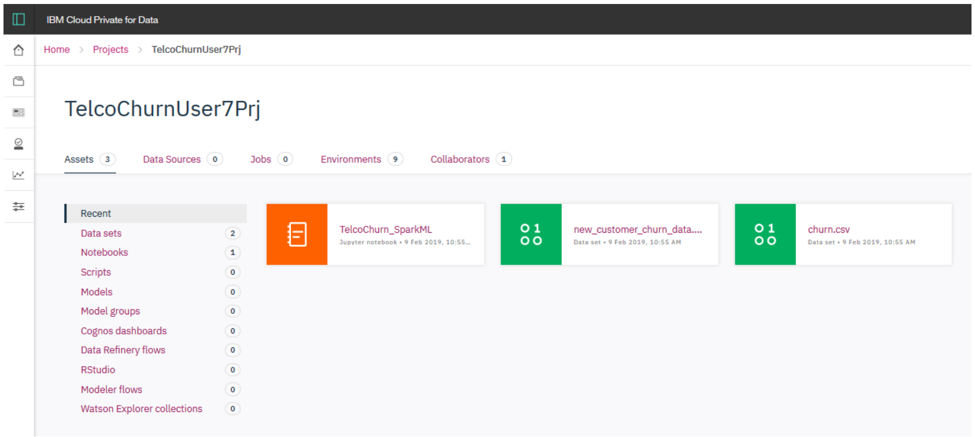
ICP4DTelcoChurnUser7

1.8. Click '**Create**', then the project is created. You may have to click in in Name field and tab to next field in order to see 'Create' button.

The imported project has several "Assets" such as data sets, notebooks etc that will be available for this project.

2. Setup Data Source for Analytics Project

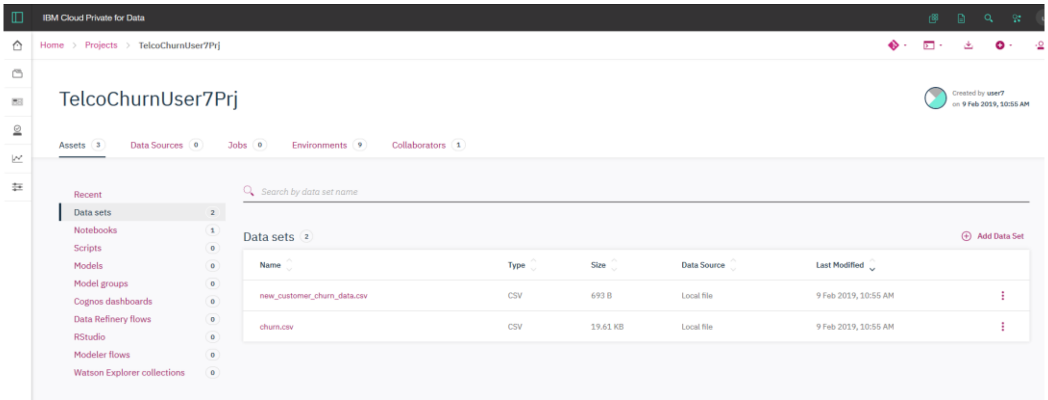
2.1. Click ‘**Assets**’ to view all available assets that are in this project. (may have to switch to different tab to refresh Assets page)



We will be using a data set that has been enriched and made available in a db2 table for this lab.

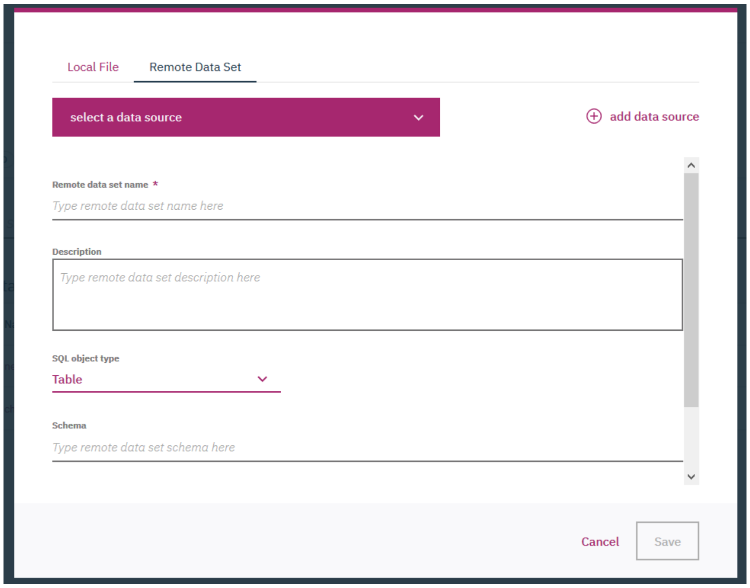
The first part is to create a data source connection for our db2WH in cloud:

From the project folder navigate to the ‘**data sets**’ section and select ‘**add data set**’.

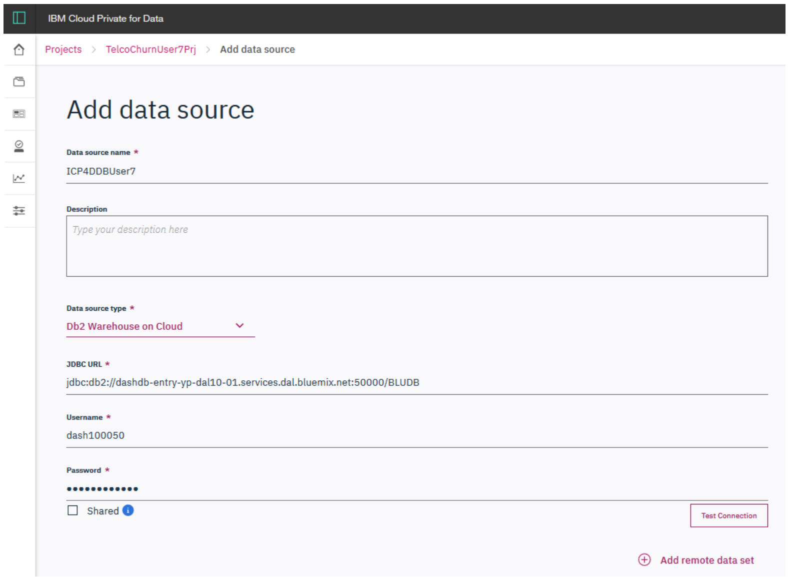


Name	Type	Size	Data Source	Last Modified
new_customer_churn_data.csv	CSV	693 B	Local file	9 Feb 2019, 10:55 AM
churn.csv	CSV	19.61 KB	Local file	9 Feb 2019, 10:55 AM

2.2. Select the ‘**Remote Data Set**’ tab and click ‘**add data source**’ to define the data source connected.



2.3. Fill in the blanks with the corresponding data source information provided in "Pre-requisite" section above .



Add data source

Data source name *
ICP4DDBUser7

Data source type *
Db2 Warehouse on Cloud

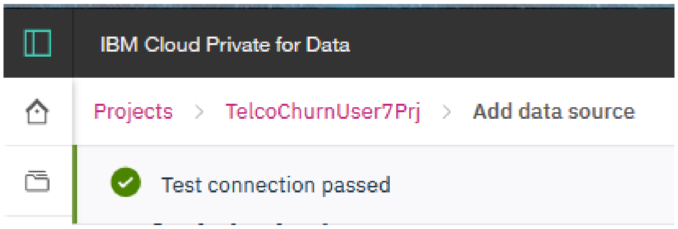
JDBC URL *
jdbc:db2://dashdb-entry-yp-dal10-01.services.dal.bluemix.net:50000/BLUDB

Username *
dash100050

Password *
••••••••••••

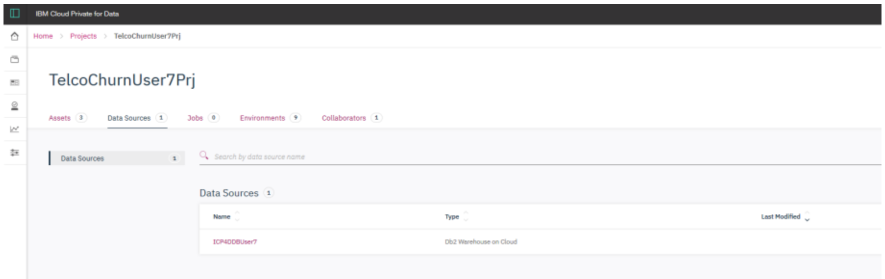
Name: ICP4DDBUser7 (add your userid)
Data source type : Db2 Warehouse on Cloud
JDBC URL : <refer Pre-requisite Section for database connection above>
Username: <refer Pre-requisite Section for database connection above>
Password: <refer Pre-requisite Section for database connection above>

2.4. Click ‘**Test Connection**. A message will be displayed on top to indicate successful connection.


IBM Cloud Private for Data
Projects > TelcoChurnUser7Prj > Add data source

Test connection passed

2.5. Click ‘**Create**’. The connection is created. To verify, Click on **Data Sources.** Newly created source will be displayed.



Data Sources 1

Name	Type	Last Modified
ICP4DDBUser7	Db2 Warehouse on Cloud	

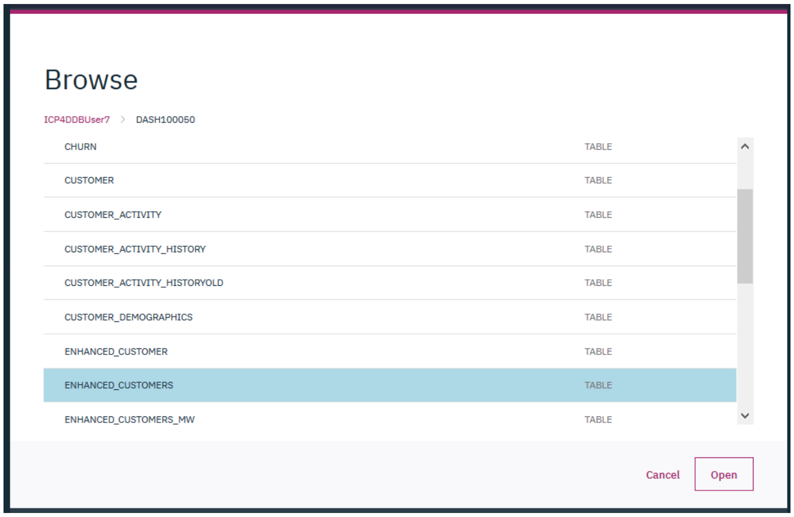
2.6. Let’s create the data set that we need to access in sub-sequent notebook.

2.7. Navigate back to the ‘**Assets**’ and then to ‘**Remote data set**’ tab and choose the data source you just added from the Pull down menu.

2.8. Click on the “add data set” to the right of Remote data set.

2.9. Enter **CUSTOMER** in the “Remote data set name: field

2.10. Scorll-down and select “Browse” button to select specific table that we need to access. Select schema name “**DASH100050”** and then select **“ENHANCED_CUSTOMERS” table.** Click on **“Open”**

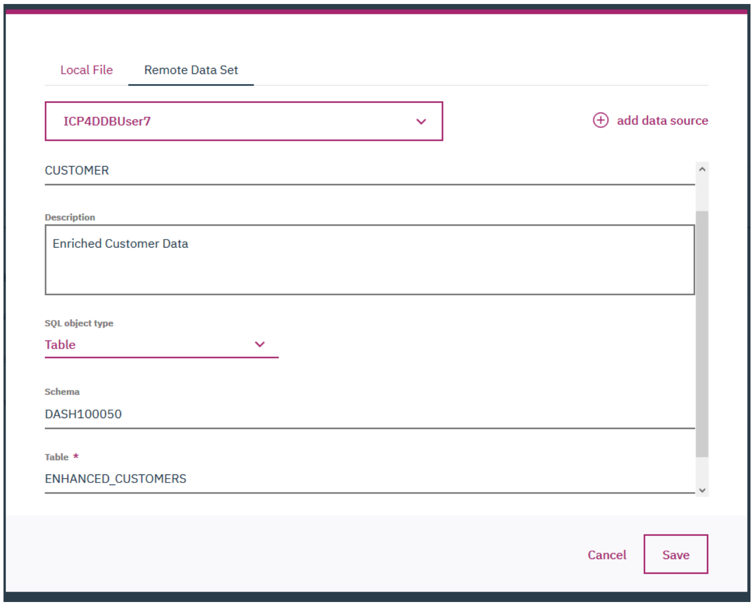


Browse

ICP4DDBUser7 > DASH100050

CHURN	TABLE
CUSTOMER	TABLE
CUSTOMER_ACTIVITY	TABLE
CUSTOMER_ACTIVITY_HISTORY	TABLE
CUSTOMER_ACTIVITY_HISTORYOLD	TABLE
CUSTOMER_DEMOGRAPHICS	TABLE
ENHANCED_CUSTOMER	TABLE
ENHANCED_CUSTOMERS	TABLE
ENHANCED_CUSTOMERS_MW	TABLE

2.11. The screen will be filled with required fields like :



ICP4DDBUser7

CUSTOMER

Description

Enriched Customer Data

SQL object type

Table

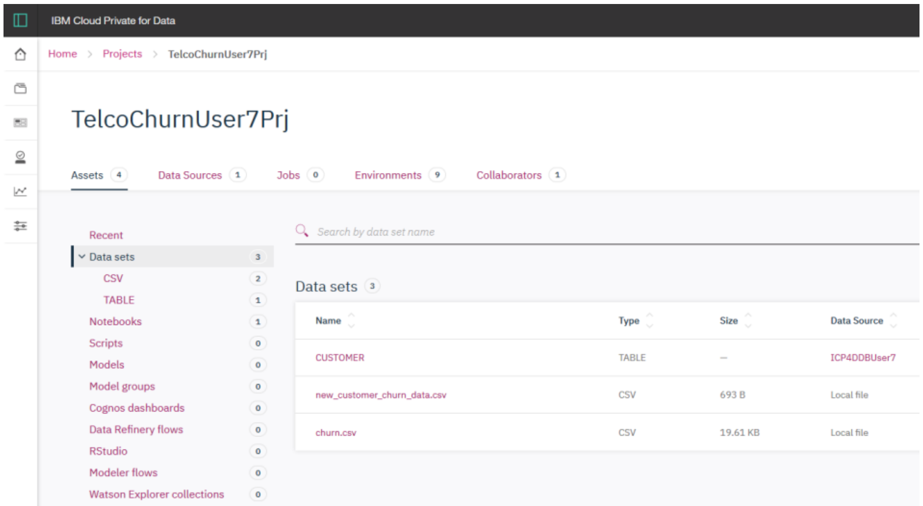
Schema

DASH100050

Table *

ENHANCED_CUSTOMERS

2.12. Click **'Save'**. The connected remote data is now listed in the 'data set' section on your projects **Asset** page



TelcoChurnUser7Prj

Data sets 3

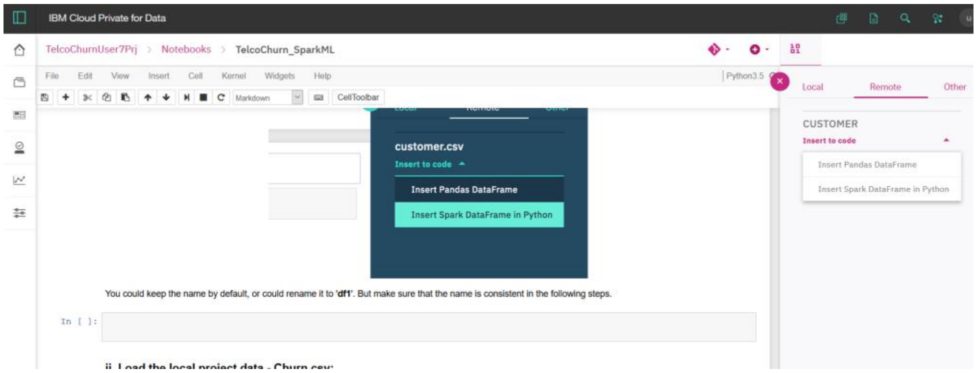
Name	Type	Size	Data Source
CUSTOMER	TABLE	–	ICP4DDBUser7
new_customer_churn_data.csv	CSV	693 B	Local file
churn.csv	CSV	19.61 KB	Local file

3. Build ML Model using Jupyter Notebook

3.1. Jupyter Notebook is an interactive exploration and development tool used by data analysts, data engineers and data scientists. We will use the notebook **TelcoChurn_SparkML** that has been imported as part of our project. The notebook contains comments which will guide you through its use. As you review each cell click in it with your mouse and use the run button on the toolbar to execute it (short-cut is to use **Shift-Enter** key to execute current selected cell.

3.2. Navigate to the ‘Notebooks’ tab and Click on ‘TelcoChurn_SparkML’ notebook. Follow the notebook instructions and execute all cells as directed.

3.3. You will first load the remote **CUSTOMER** data you created in the earlier step. Scorll down to first empty cell as shown below and select the cell :



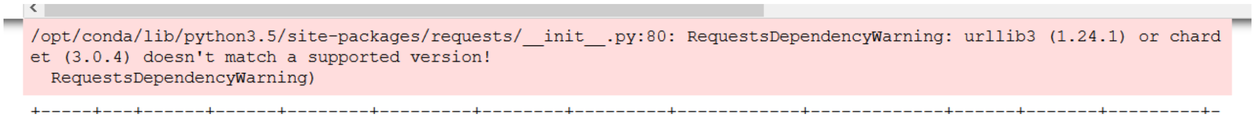
You could keep the name by default, or could rename it to '**df1**'. But make sure that the name is consistent in the following steps.

```
In [ ]:
```

ii. Load the local project data - Churn.csv:

NOTE : Make sure to select Spark DataFrame option and also modify variable “**df**” (generated code use df2,df3 etc based on number of attempts). This dataframe variable is accessed throughout as **df1**.

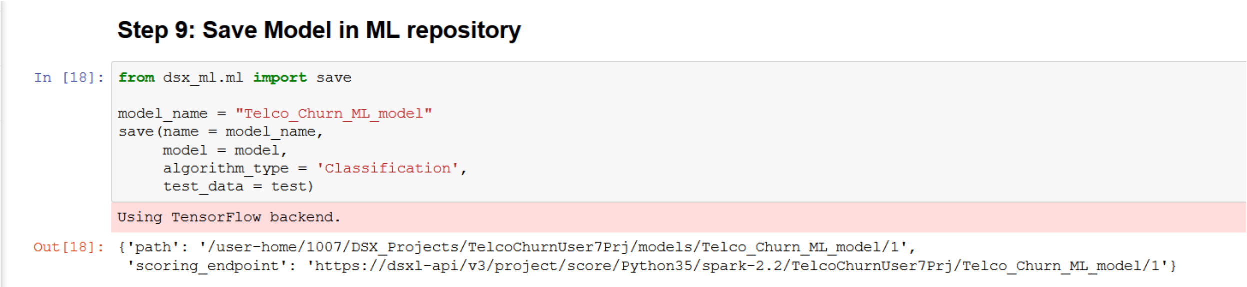
You can ignore this warning:



```
/opt/conda/lib/python3.5/site-packages/requests/__init__.py:80: RequestsDependencyWarning: urllib3 (1.24.1) or chard
et (3.0.4) doesn't match a supported version!
  RequestsDependencyWarning)
+------+---+-------+-------+---------+---------+---------+---------+------------+-------------+-------+--------+----------+-
```

3.4. Follow directions in the Notebook to execute every cell.

3.5. In Step 9 you will save the model into the project repository for further use.



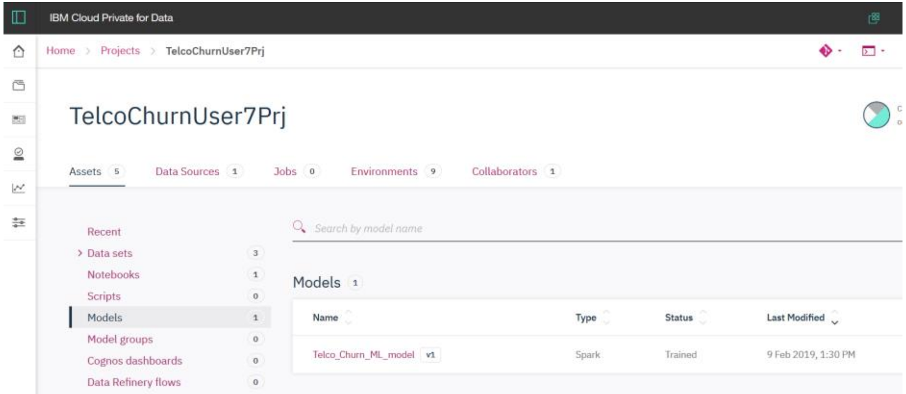
Step 9: Save Model in ML repository

```
In [18]: from dsx_ml.ml import save

         model_name = "Telco_Churn_ML_model"
         save(name = model_name,
              model = model,
              algorithm_type = 'Classification',
              test_data = test)

         Using TensorFlow backend.

Out[18]: {'path': '/user-home/1007/DSX_Projects/TelcoChurnUser7Prj/models/Telco_Churn_ML_model/1',
          'scoring_endpoint': 'https://dsxl-api/v3/project/score/Python35/spark-2.2/TelcoChurnUser7Prj/Telco_Churn_ML_model/1'}
```

NOTE: The scoring_endpoint as the result of above step. For example

```
{'path': '/user-home/1007/DSX_Projects/TelcoChurnUser7Prj/models/Telco_Churn_ML_model/1',
 'scoring_endpoint': 'https://dsxl-api/v3/project/score/Python35/spark-2.2/TelcoChurnUser7Prj/Telco_Churn_ML_model/1'}
```

3.6. Verify that the model is saved under the model section of the project by navigating to the **Assets** view. Look under the ‘**Models**’ tab to make sure that the model is shown. Your model may have a different name and version.



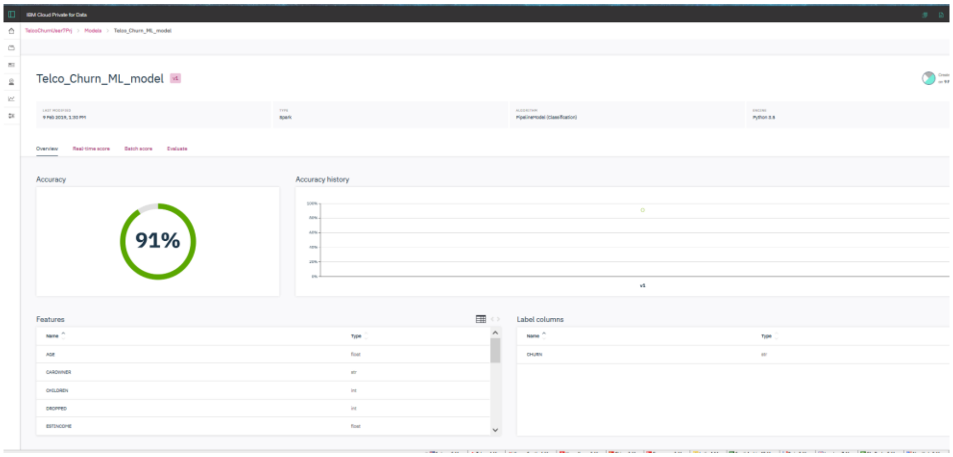
Name	Type	Status	Last Modified
Telco_Churn_ML_model v1	Spark	Trained	9 Feb 2019, 1:30 PM

4. Test and Validate ML Model

4.1. Let’s start by testing the model by creating a sample single case scoring scenario

4.2. Navigate to the Models tab in the **Assets** view of your project.

4.3. Click the the model name ‘**Telco_Churn_ML_model**’ . This will show an Overview of the model asset.



Telco_Churn_ML_model v1

LAST MODIFIED 9 Feb 2019, 1:30 PM

TYPE Spark

ALGORITHM PipelineModel (Classification)

ENGINE Python 3.5

Features

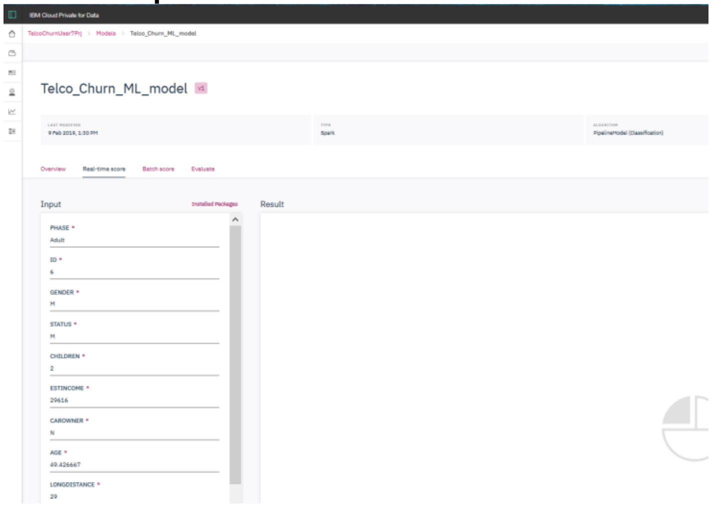
Name	Type
AGE	float
CAROWNER	str
CHILDREN	int
DROPPED	int
ESTINCOME	float

Label columns

Name	Type
CHURN	str

4.4. In the next view select the ‘**Real-time score** tab.

4.5. There will be some data automatically generated for each of the fields in the model. They are displayed in the ‘**Input’** section.



IBM Cloud Private for Data
TelcoChurnUser7Prj > Models > Telco_Churn_ML_model
Overview
Real-time score
Batch score
Evaluate
Installed Packages
Result

Telco_Churn_ML_model v1

LAST MODIFIED 9 Feb 2019, 1:30 PM

TYPE Spark

ALGORITHM PipelineModel (Classification)

Input

PHASE * Adult

ID * 6

GENDER * M

STATUS * M

CHILDREN * 2

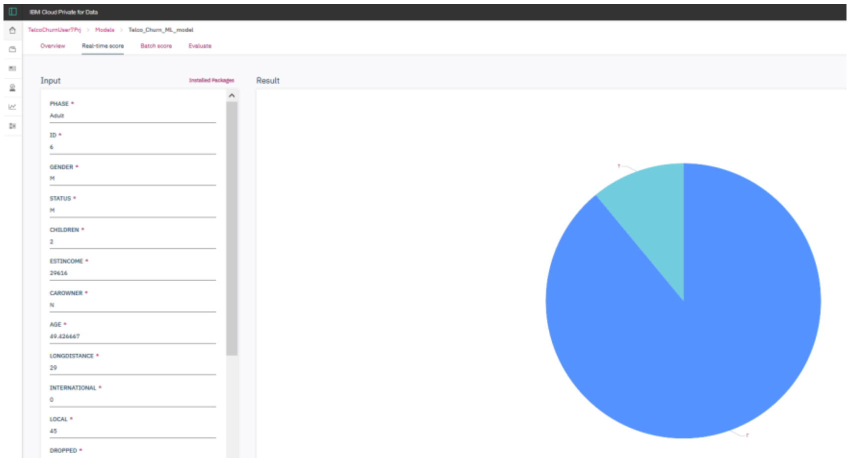
ESTINCOME * 29616

CAROWNER * N

AGE * 49.426667

LONGDISTANCE * 29

4.6. Click ‘submit’ and the online score will show on the right side ‘Result’ section. You can change some of the input values to see how it changes the prediction.


IBM Cloud Private for Data
TelcoChurnUser7Prj > Models > Telco_Churn_ML_model
Overview
Real-time score
Batch score
Evaluate
Input
Installed Packages
Result
T
F

PHASE *
Adult

ID *
6

GENDER *
M

STATUS *
M

CHILDREN *
2

ESTINCOME *
29616

CAROWNER *
N

AGE *
49.426667

LONGDISTANCE *
29

INTERNATIONAL *
0

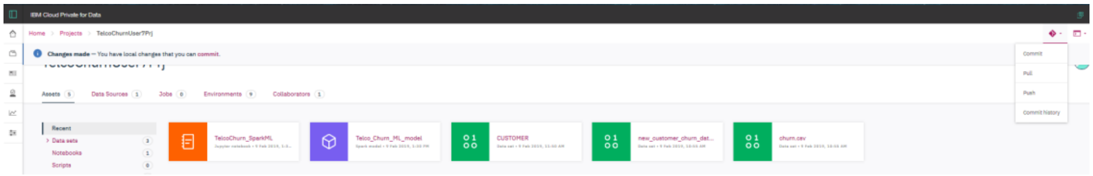
LOCAL *
45

DROPPED *

5. Commit Changes to Project

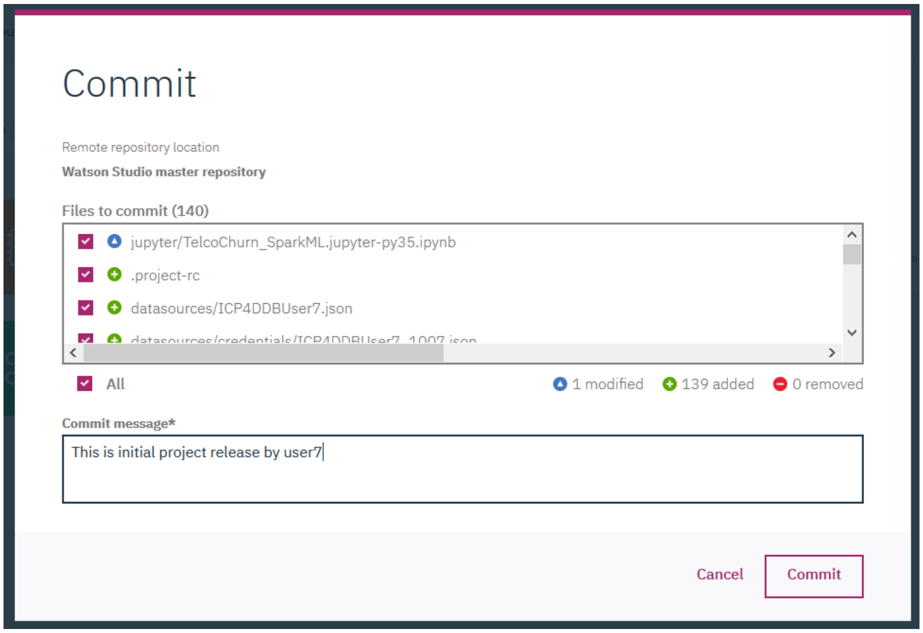
5.1. Next we will prepare the model artifacts for deployment in production setup so that we can score new records. The data scientists may tag a model once satisfied with the result as “published” for production consumption. In ICP for Data platform this step is called creating “**Project release**”.

5.2. Navigate to the project homepage, you will see the message that ask if you want to **commit** the changes to this project from the model development enviornment. This will push all of the assets to the Model Management & Deployment (MMD) environment Click ‘**Commit**’. (In case this message does not showup Click on **“Git actions” and click “commit”** from top right menu.)



Changes made — You have local changes that you can commit.

5.3. The “commit” screen will show list of the assets that are created in this project. Enter suitable comment for e.g. “This is initial project release by user7” .



Commit

Remote repository location

Watson Studio master repository

Files to commit (140)

☑ jupyter/TelcoChurn_SparkML.jupyter-py35.ipynb

☑ .project-rc

☑ datasources/ICP4DDBUser7.json

☑ datasources/credentials/ICP4DDBUser7_1007.json

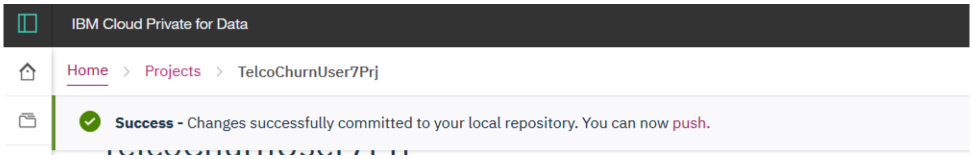
☑ All — 1 modified, 139 added, 0 removed

Commit message*

This is initial project release by user7

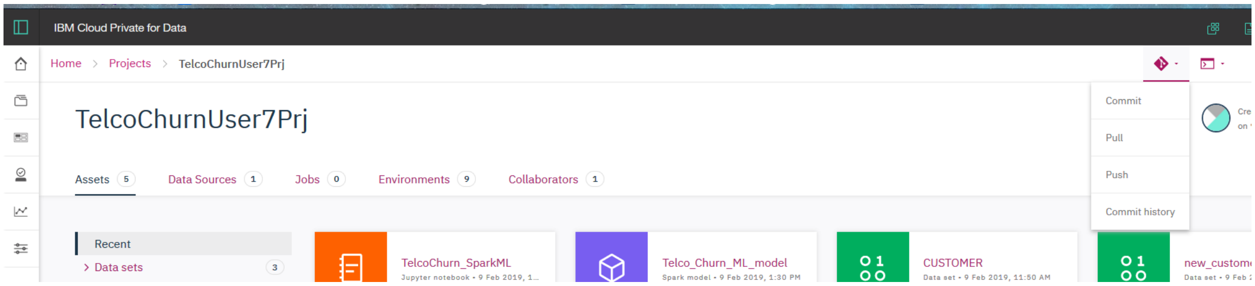
Click ‘**Commit** ‘

You will a message window confirming the changes were commited and now available to “push” for deployment.

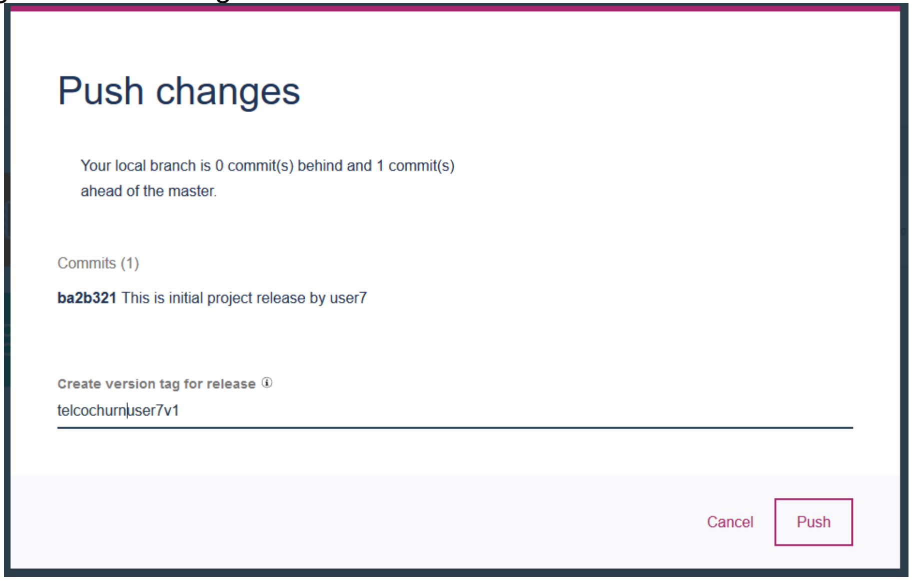


Success - Changes successfully committed to your local repository. You can now push.

5.4. Click on “**push**” (or if that window does not show up, click on “Git actions” and select “push”



This will bring the “Push changes” window like :



Push changes

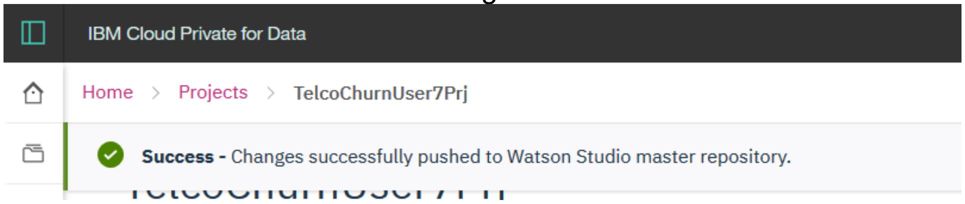
Your local branch is 0 commit(s) behind and 1 commit(s) ahead of the master.

Commits (1)

ba2b321 This is initial project release by user7

Create version tag for release: telcochurnuser7v1

We need to create a “tag” for our release, so that a version of this model can be idenfied for deployment. For e.g. enter tag as “telcochurnuser7v1”
Click “**Push**”. This will show confirmation message window like:



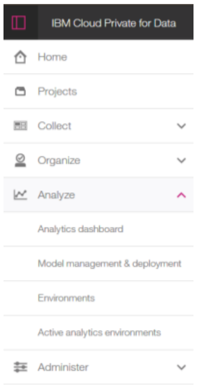
Success - Changes successfully pushed to Watson Studio master repository.

NOTE : Some messages may show “Watson Studio” reference. This is due to similar services being utilized within ICP for Data.

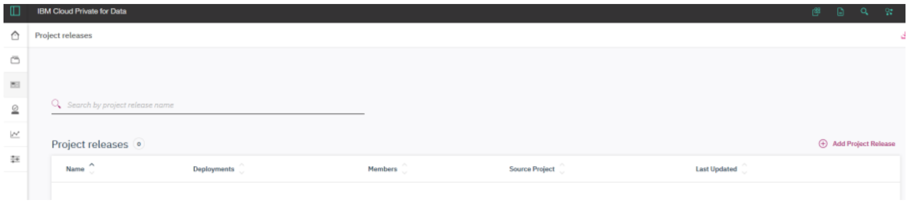
6. Deploy ML Model using Project Release

6.1. To expose a checked-in assets to outside users, an administrator can create a project release and deploy the assets within it. A project release represents a project tag that can be launched as a production environment. An administrator can monitor releases and deployments from the Project releases page.

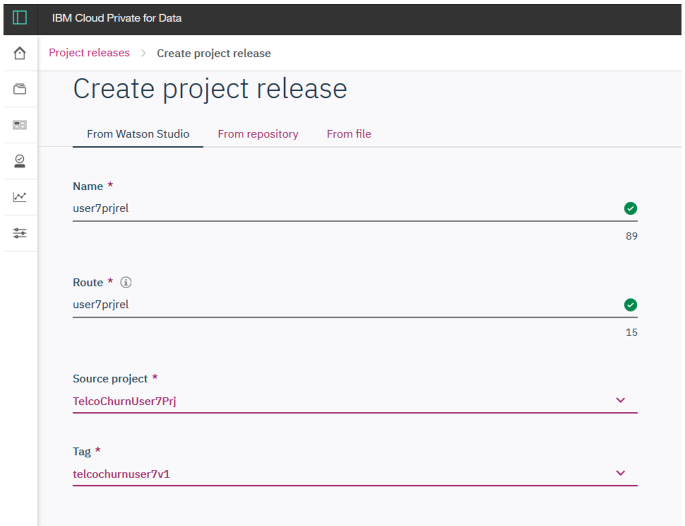
6.2. To open the Project releases page, Navigate to the home menu and select ‘**Analyze’** and then ‘**Model management & deployment’**.


IBM Cloud Private for Data
Home
Projects
Collect
Organize
Analyze
Analytics dashboard
Model management & deployment
Environments
Active analytics environments
Administer

6.3. To add a new project release Click “**Add Project Release** “ from right handside option, as show in following screenshot.



6.4. Add in the details of your Project Release under the ‘**From Watson Studio’** tab:


IBM Cloud Private for Data
Project releases > Create project release

Create project release

From Watson Studio
From repository
From file

Name * user7prjrel 89

Route * user7prjrel 15

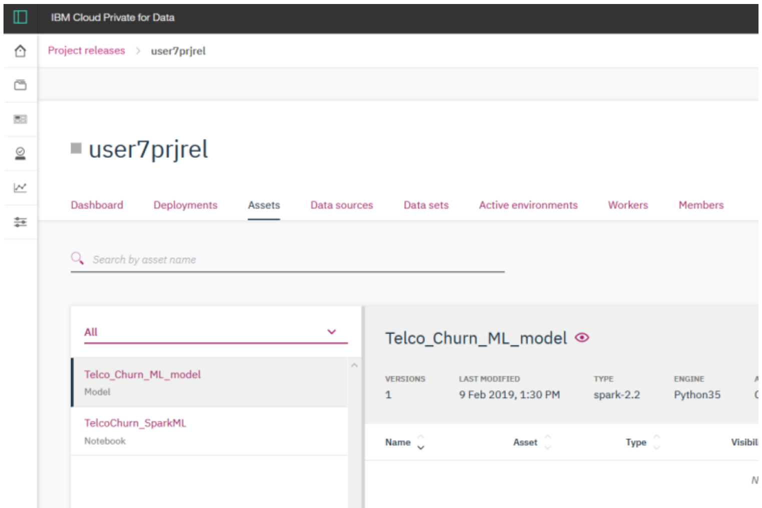
Source project * TelcoChurnUser7Prj

Tag * telcochurnuser7v1

For e.g.

Name : user7prjrel
Route : user7prjrel
Source Project : TelcoChurnUser7Prj
Tag : telcochurnuser7v1
NOTE : The name can contain hyphens but not special characters such as a period (.). The route is the unique ID for the project release, and is used within the deployments' REST paths and URLs. This is a unique part of the url that will be assigned to all of the assets that are created related with this project. Name of the “route” must be lowercase.

6.5. Click ‘**Create’ .** This creates an offline release.



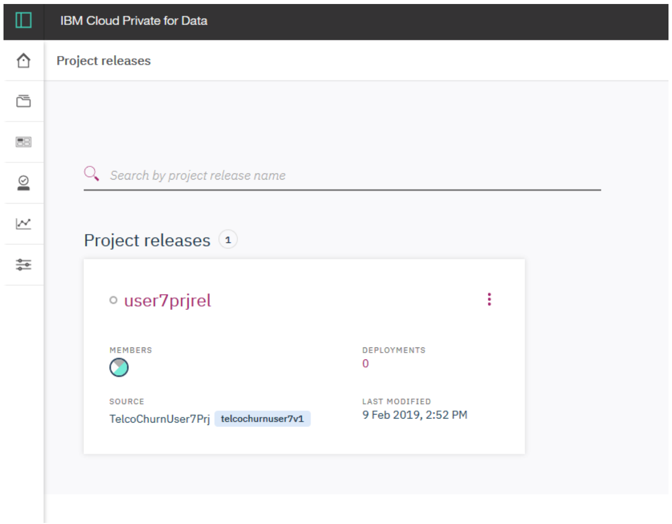
Telco_Churn_ML_model
Model

TelcoChurn_SparkML
Notebook

Telco_Churn_ML_model

VERSIONS 1
LAST MODIFIED 9 Feb 2019, 1:30 PM
TYPE spark-2.2
ENGINE Python35

6.6. Click on “Project releases” from top left. This will show currently available releases. Note that your view shows 0 under Deployments. We will be deploying a model asset within this project release.



Project releases 1

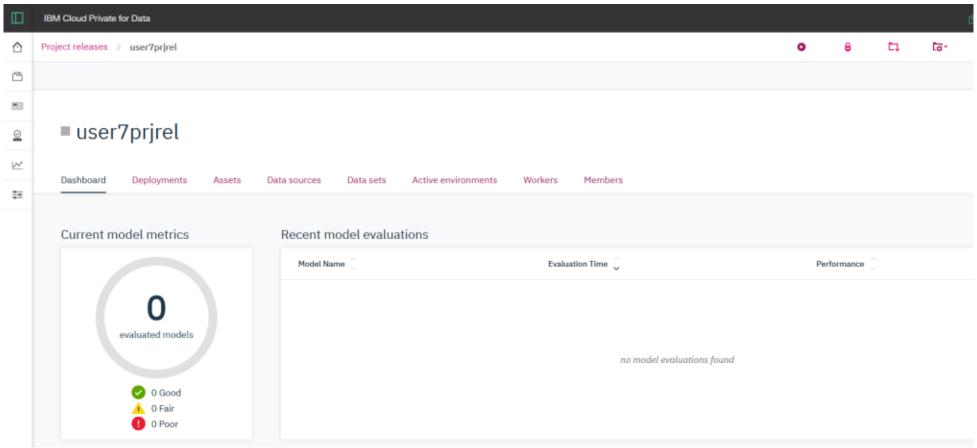
user7prjrel

MEMBERS	DEPLOYMENTS
	0

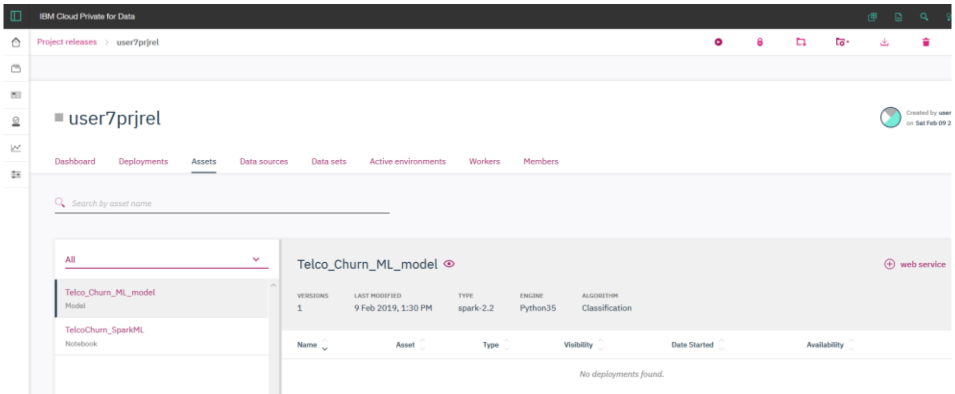
SOURCE	LAST MODIFIED
TelcoChurnUser7Prj telcochurnuser7v1	9 Feb 2019, 2:52 PM

6.7. To view details of the deployed project, from Project release page, click on deployment name that was just created for e.g. “**user7prjrel**”

6.8. This will bring up **Dashboard** for the deployment with several detailed tabs for more information about the Project release.



6.9. Click on **Asset** tab to see all deployed assets related to this Project release.



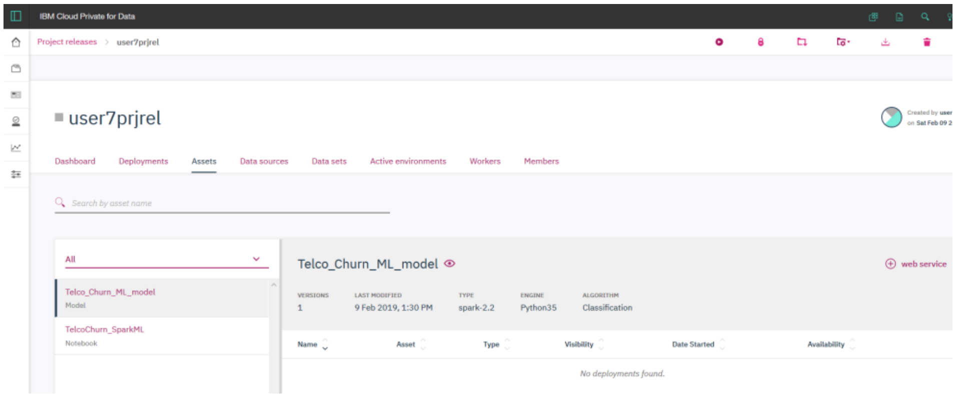
VERSIONS	LAST MODIFIED	TYPE	ENGINE	ALGORITHM
1	9 Feb 2019, 1:30 PM	spark-2.2	Python35	Classification

Name	Asset	Type	Visibility	Date Started	Availability

No deployments found.

6.10. Next we will create a “Realtime online deployment” of the model. The ICP for Data platform will package the required model artifacts and package it as a docker container along with any runtime execution environment. This container is isolated image from other “development” services that are running. An administrator can provison multiple replicas of this service in order to full-fill concurrent access to model scoring in production.

6.11. In the **Assets** tab, you can see all of the analytics assets. There are notebooks, models, and scripts that we created previously.

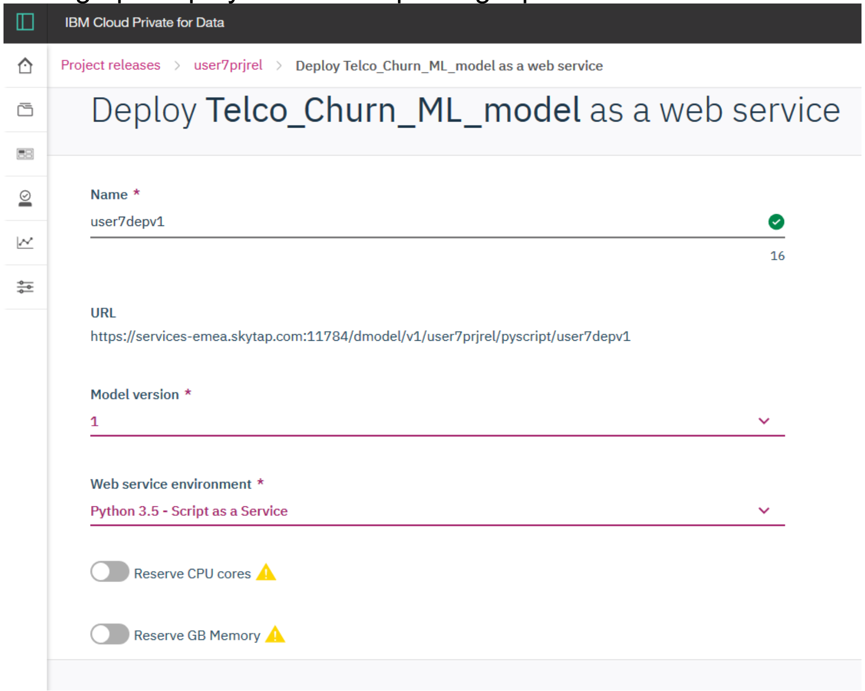


Telco_Churn_ML_model

VERSIONS	LAST MODIFIED	TYPE	ENGINE	ALGORITHM
1	9 Feb 2019, 1:30 PM	spark-2.2	Python35	Classification

No deployments found.

6.12. Selct the **‘TelcoChurn_SparkML_model’** on the left and Click “**web service**”. This will bring up “Deploy” window expecting input as shown below :


IBM Cloud Private for Data
Project releases > user7prjrel > Deploy Telco_Churn_ML_model as a web service
Deploy Telco_Churn_ML_model as a web service
Reserve CPU cores
Reserve GB Memory

Name *
user7depv1
16

URL
https://services-emea.skytap.com:11784/dmodel/v1/user7prjrel/pyscript/user7depv1

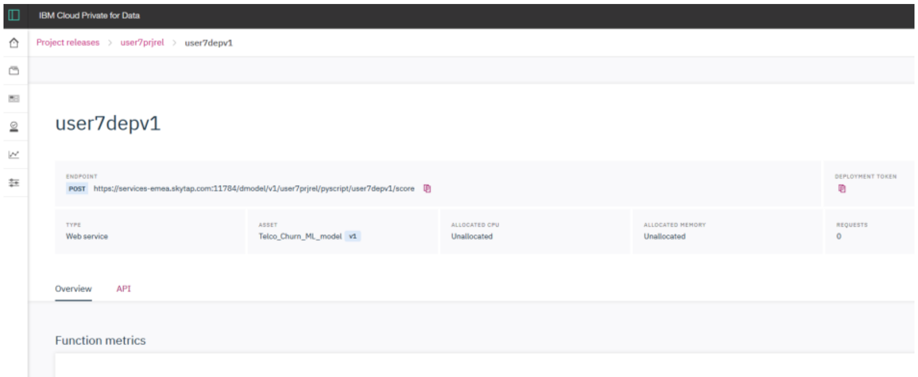
Model version *
1

Web service environment *
Python 3.5 - Script as a Service

Fill in the deployment details. For example :
Name : user7depv1
Model version : 1
Web service environment : Python 3.5 – Script as Service

Click “**Create**”

6.14. This will bring deployment Overview screen with API tab as shown below :



IBM Cloud Private for Data
Project releases > user7prjrel > user7depv1

user7depv1

ENDPOINT
POST https://services-emea.skytap.com:11784/dmodel/v1/user7prjrel/pyscript/user7depv1/score

DEPLOYMENT TOKEN

TYPE	ASSET	ALLOCATED CPU	ALLOCATED MEMORY	REQUESTS
Web service	Telco_Churn_ML_model v1	Unallocated	Unallocated	0

Function metrics

Note your Web service ENDPOINT URL and access token :

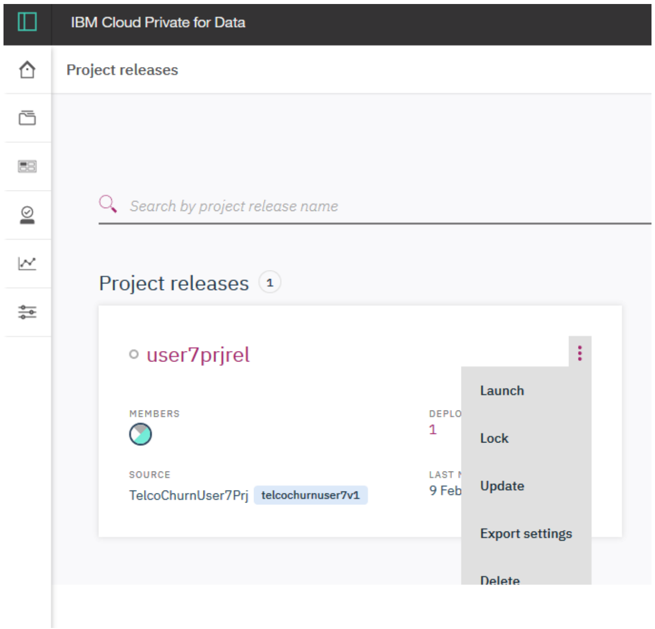
https://services-emea.skytap.com:11784/dmodel/v1/user7prjrel/pyscript/user7depv1/score

Deployment Token :
Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InVzZXI3IiwicGFja2FnZ
U5hbWUiOiJ1c2VyN3ByanJlbCIsInBhY2thZ2VSb3V0ZSI6InVzZXI3cHJqcmVsIiwiaWF
0IjoxNTQ5NzQ1NTMwfQ.0ZmPZKjmPopqAXeHgdaG80MCp_jLzx_5yPaUmTDPGg4w
pTKGMAmT2fe1Pyq7KEzXJJdXbQ_UcvPR_wG44VVo9Hb1Xyrmm_CY3uOj8zLbJ-
EUykw4F7v6ld0lTNd7qUeZCI02TSvQB0PEvz0ROUloJMtsUfFMl4z_RfLbZhoc0U9U0Ef
2EDYAK38MunJa6qiyR2KfuJ8naZ2oxQFAGFbNyswcHltVfrEo2pdjSMeN4kF09O0tx27-
a7HtJJG1NjibNDWVizZHFUG2gKu9aWNaRbKU0_RJmyrjMZk3MGFvesvzyjnOdRcslz
Khfrh_4eGMfZvR1STJoIkaWQVb0gK2xw

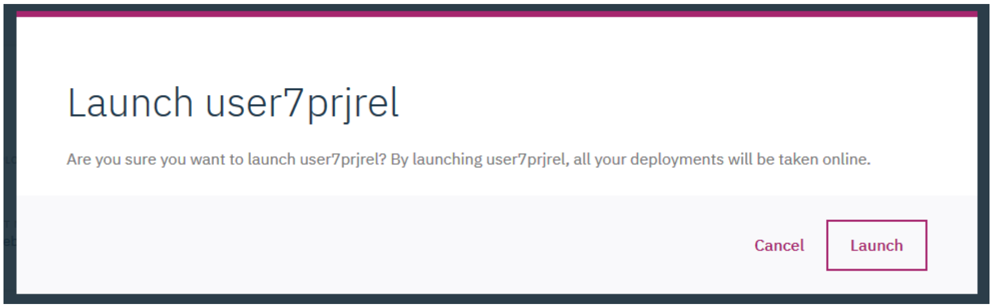
Note: At this time, the online deployment is created. You could also find the REST API and deployment token under ‘API’ tab. Simply click, the token is copied to your clipboard.

6.15. Note that the deployment is not active as yet. We will need to “Launch” the deployment in order instantiate required execution environments.

6.16. Click on Project releases tab from top left side tab. Click on 3 dots right of project release name. You will see screen options as shown below :



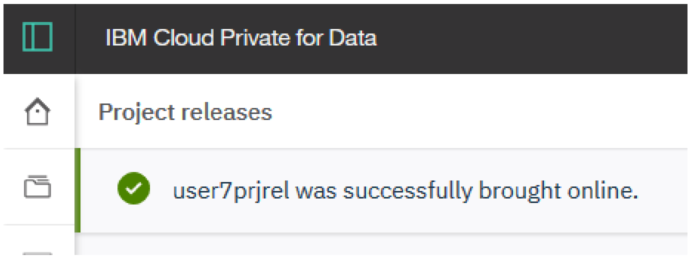
6.17. Click “**Launch**” . This will prompt showing all deployments will be taken online for end user access.



Launch user7prjrel

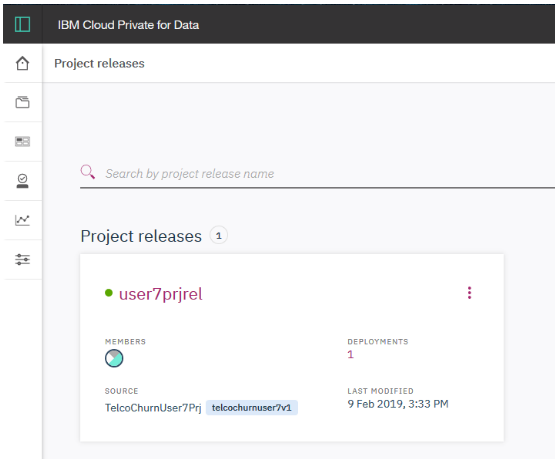
Are you sure you want to launch user7prjrel? By launching user7prjrel, all your deployments will be taken online.

Click “**Launch**” . This will show confirmation message that deployment is successfully brought online :



user7prjrel was successfully brought online.

Note : Actual instantiantion of environments and setup could take 1-2 minutes befoe APIs can be accessible. Under project releases , “Deployments” will show count as 1.



Project releases

Search by project release name

Project releases 1

● user7prjrel

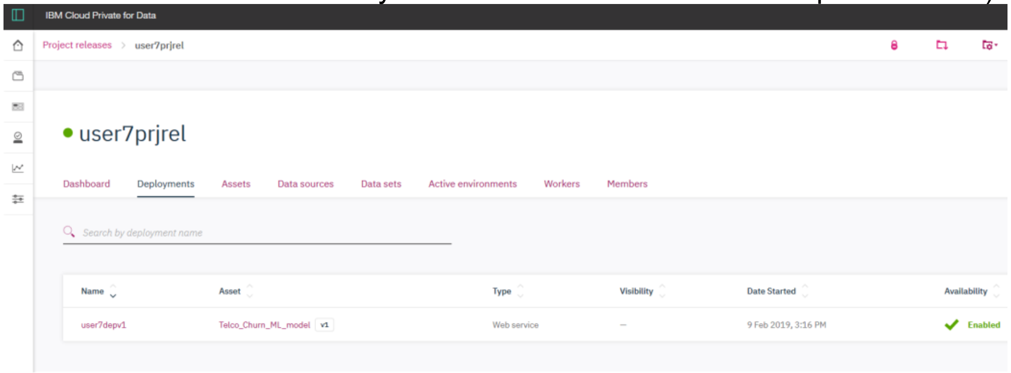
MEMBERS

DEPLOYMENTS
1

SOURCE
TelcoChurnUser7Prj telcochurnuser7v1

LAST MODIFIED
9 Feb 2019, 3:33 PM

6.18. Click on Project release name. This will bring back Dashboard. Click on Deploments tab, now this will show 1 model with “**Avaliablity**” as “**Enabled**” . (The enabled status may not change for 1-2 minutes after “**Launch**” step. Allow some time, it it is still in “disable” state. May need to click to different tab to update status.)

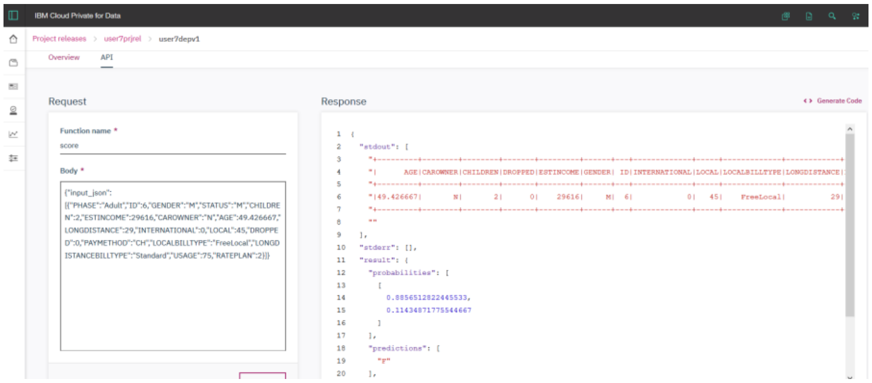


● user7prjrel

Search by deployment name

Name	Asset	Type	Visibility	Date Started	Availability
user7depv1	Telco_Churn_ML_model v1	Web service	–	9 Feb 2019, 3:16 PM	✓ Enabled

6.19. Click on deployment name (for e.g. ***user7depv1***). This will bring up “Overview” and API tab for this deployment. To test the endpoint is functioning, we can run a sample record to score. Click on API tab and Click “**Submit**” under “Body” window.



IBM Cloud Private for Data
Project releases > user7prjrel > user7depv1
Overview
API

Request

Function name *
score

Body *

```
{"input_json":
[{"PHASE":"Adult","ID":6,"GENDER":"M","STATUS":"M","CHILDREN":2,"ESTINCOME":29616,"CAROWNER":"N","AGE":49.426667,"LONGDISTANCE":29,"INTERNATIONAL":0,"LOCAL":45,"DROPPED":0,"PAYMETHOD":"CH","LOCALBILLTYPE":"FreeLocal","LONGDISTANCEBILLTYPE":"Standard","USAGE":75,"RATEPLAN":2}]}
```

Response

```
{
  "stdout": [
    "+---------+--------+--------+-------+---------+------+---+-------------+-----+-------------+------------+
    "|      AGE|CAROWNER|CHILDREN|DROPPED|ESTINCOME|GENDER| ID|INTERNATIONAL|LOCAL|LOCALBILLTYPE|LONGDISTANCE|
    "+---------+--------+--------+-------+---------+------+---+-------------+-----+-------------+------------+
    "|49.426667|       N|       2|      0|    29616|     M|  6|            0|   45|    FreeLocal|          29|
    "+---------+--------+--------+-------+---------+------+---+-------------+-----+-------------+------------+
    ""
  ],
  "stderr": [],
  "result": {
    "probabilities": [
      [
        0.8856512822445533,
        0.11434871775544667
      ]
    ],
    "predictions": [
      "F"
    ],
```

This will show scoring of 1 record using real-time deployment of the model. The Response is shown as JSON record.

Note: Generate Cide option on top right shows how to invoke REST API call using cURL command.

This completes SECTION I of the lab. Your ML model asset was developed, trained, tested and deployed for production scoring usage.

(If you are not able to finish all the steps in this section, instructors have deployed a model that can be used for next section of the lab.)

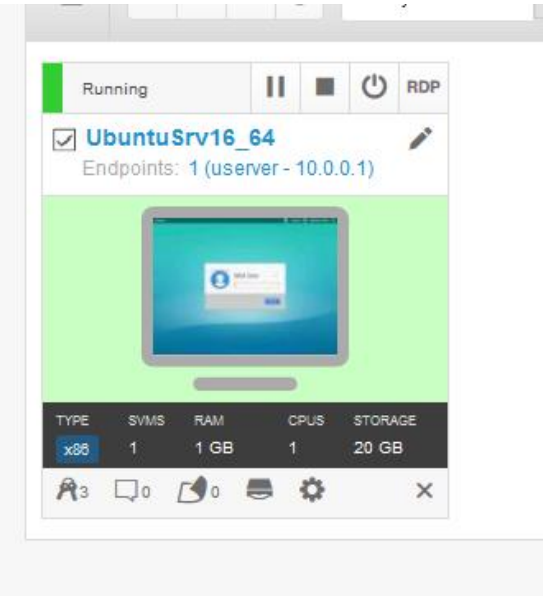
Section II

7. Customer Support Interaction Application

In this section we will create an online application using Python Flask framework. Each attendee will have access to their own linux virtual machine with base python packages installed.

Set up your linux environment

7.1. Login to your linux VM using browser on your laptop. URL and credentials will be provided through automated email msg. (or provided in the class).



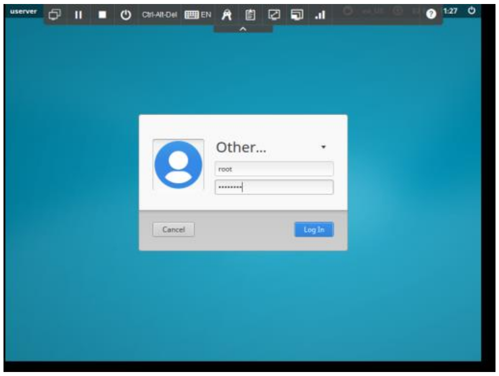
Running

UbuntuSrv16_64

Endpoints: 1 (userver - 10.0.0.1)

TYPE	SVMS	RAM	CPUS	STORAGE
x86	1	1 GB	1	20 GB

7.2. Click on the terminal image. This will prompt for user name ,
userid: root password : thinkibm



Other...

root

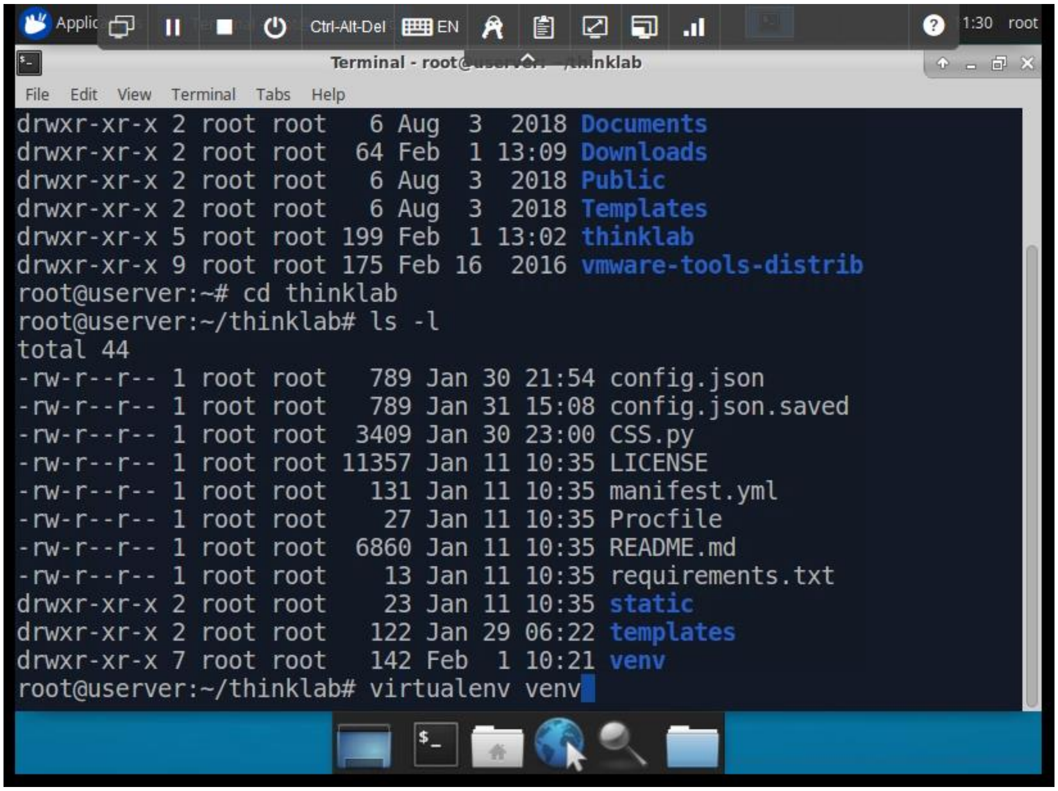
Note: You may get a pop-up window with "Error found when loading /root/profile" . You can ignore this message.

7.3. We will setup the environment that has been pre-packed with all required libraries and packages. Set python virtual environment using following commands

7.3.1. cd thinklab

7.3.2. virtualenv venv

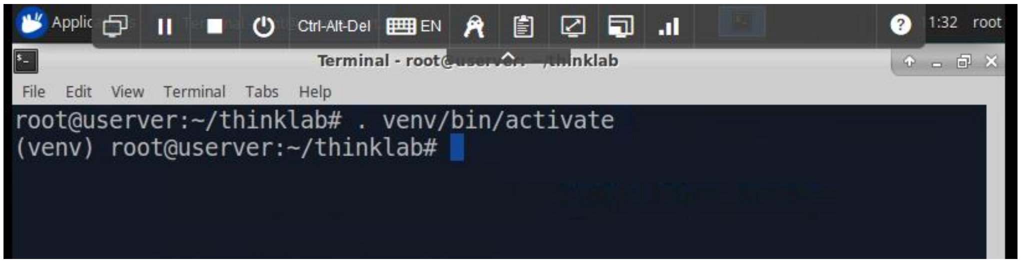
7.3.3. Above command may result in warning/error as the env is already setup. You can ignore the message and go to next step.



```
drwxr-xr-x 2 root root   6 Aug  3  2018 Documents
drwxr-xr-x 2 root root  64 Feb  1 13:09 Downloads
drwxr-xr-x 2 root root   6 Aug  3  2018 Public
drwxr-xr-x 2 root root   6 Aug  3  2018 Templates
drwxr-xr-x 5 root root 199 Feb  1 13:02 thinklab
drwxr-xr-x 9 root root 175 Feb 16  2016 vmware-tools-distrib
root@userver:~# cd thinklab
root@userver:~/thinklab# ls -l
total 44
-rw-r--r-- 1 root root   789 Jan 30 21:54 config.json
-rw-r--r-- 1 root root   789 Jan 31 15:08 config.json.saved
-rw-r--r-- 1 root root  3409 Jan 30 23:00 CSS.py
-rw-r--r-- 1 root root 11357 Jan 11 10:35 LICENSE
-rw-r--r-- 1 root root   131 Jan 11 10:35 manifest.yml
-rw-r--r-- 1 root root    27 Jan 11 10:35 Procfile
-rw-r--r-- 1 root root  6860 Jan 11 10:35 README.md
-rw-r--r-- 1 root root    13 Jan 11 10:35 requirements.txt
drwxr-xr-x 2 root root    23 Jan 11 10:35 static
drwxr-xr-x 2 root root   122 Jan 29 06:22 templates
drwxr-xr-x 7 root root   142 Feb  1 10:21 venv
root@userver:~/thinklab# virtualenv venv
```

7.3.4. source venv/bin/activate

7.3.5. Above command will change the command prompt to indicate virtual environment is set. As shown below:



```
root@userver:~/thinklab# . venv/bin/activate
(venv) root@userver:~/thinklab# 
```

After running above commands you are ready to setup your interactive application.

7.4. Review the base code for the application available under : /root/thinklab folder

Python Flask framework uses folder structure to setup the application artifacts. You will see following folders and files under /root/thinklab

```
(venv) root@userver:~/thinklab# ls -l
total 40
-rw-r--r-- 1 root root  790 Feb  8 15:05 config.json
-rw-r--r-- 1 root root 2850 Feb  7 20:13 CSS_legacy.py
-rw-r--r-- 1 root root 3745 Feb 10 18:05 CSS_ML.py
-rw-r--r-- 1 root root   13 Jan 11 10:35 requirements.txt
drwxr-xr-x 2 root root   23 Feb  6 21:45 static
drwxr-xr-x 2 root root   90 Feb 10 17:31 templates
drwxr-xr-x 7 root root  142 Feb  6 22:19 venv
```

CSS_ML.py – Main python program that contains database read and calling of Webservice for model scoring code.
CSS_legacy.py – Main python program that mimics application with no ML model webservice
config.json – This file is used as input to the program that includes database connection parameters and Webservice end point and access token
For e.g.

```
(venv) root@userver:~/thinklab# cat config.json
{
"db2infor": "DATABASE=BLUDB;HOSTNAME=dashdb-entry-yp-dal10-
01.services.dal.bluemix.net;PORT=50000;PROTOCOL=TCPIP;UID=dash100050;PWD
=oULiV__78Kej;",
"scoring_endpoint": "https://services-
emea.skytap.com:11784/dmodel/v1/telcochurnv1/pyscript/churnmodel/score",
"token": "Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwicGFja2Fn
ZU5hbWUiOiJUZWxjb1ByalJlbDEiLCJwYWNrYWdlUm91dGUiOiJ0ZWxjb2NodXJudjEiL
CJpYXQiOjE1NDkxMzQ3OTF9.1n9Um3qSFF-
k2ut1xEWjRH6DVSjqIvpYYLS3G5yRluZ97s-2-
Zd7FqwvUlFbwN6WJ0Gsv3f_CTYo_68T7SrdGMesHdsWDfMkHet2ifHGUw11v8eJShQ
qnXpWN8VKUy5z2mpdcS5iUkrGlbG-
KHz5tBw8ZTg4zyQcZhmDTZiwhsIWCkFPGkhqrD_geY3H3hC6BR4FYmC7MAC2kfJAN
oA-
SAWM1NaOa07LOz7beT_u2427jk36LW5ioNo_kFTgGRoaK6lZzZYn38k3xta1xtXNJf1g
qQcGWWc99foztjDAsz03r-zsaEB0lfNZ2WnJXcIAvW3wlHdeB_isB5ruAYToKw"
}
```

Modify config.json to your values

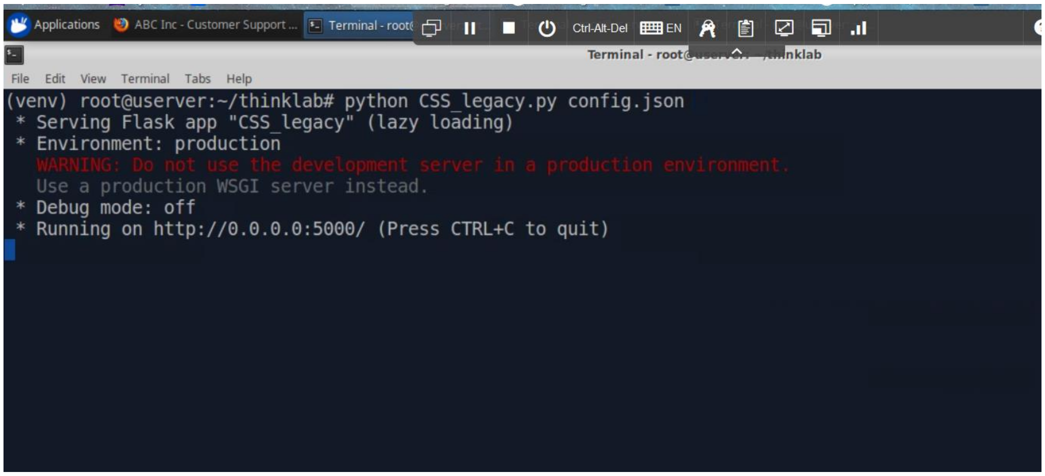
7.5. Modify config.json using “vi” editor from command line and copy-paste your saved webservice end point and token information from Section I of the lab.

Run Customer Support System App in Legacy mode (No ML)

7.6. Try running the application using following command

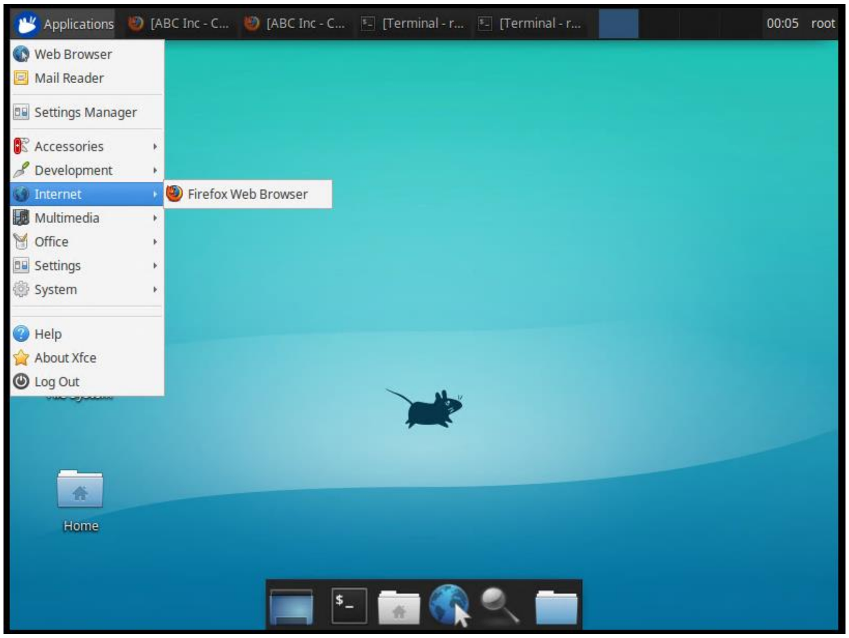
python CSS_legacy.py config.json

Note : On a successful execution, you should see following output :


Applications
ABC Inc - Customer Support ...
Terminal - root
Ctrl-Alt-Del
EN
Terminal - root@userver: ~/thinklab
File Edit View Terminal Tabs Help

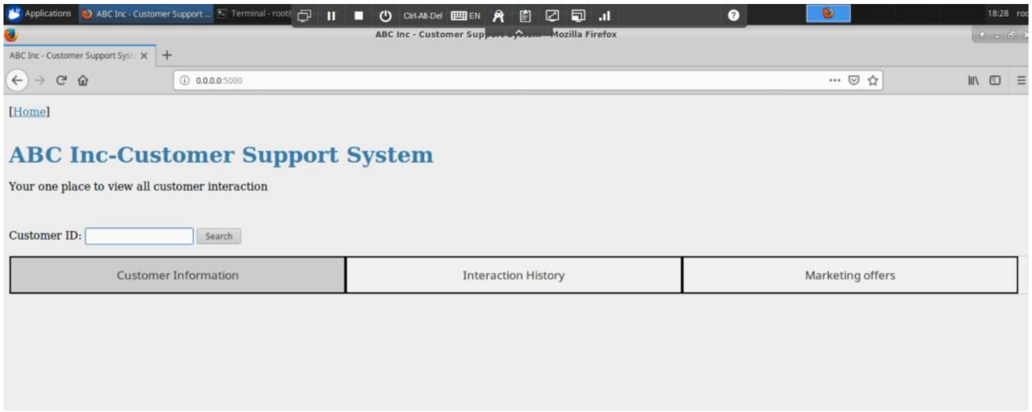
```
(venv) root@userver:~/thinklab# python CSS_legacy.py config.json
 * Serving Flask app "CSS_legacy" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

7.7. To test application, go to **Applications->Internet->Firefox Web Browser**



7.8. Enter your application URL : http://0.0.0.0:5000/ in the browser window

This will bring up the application interface for e.g.



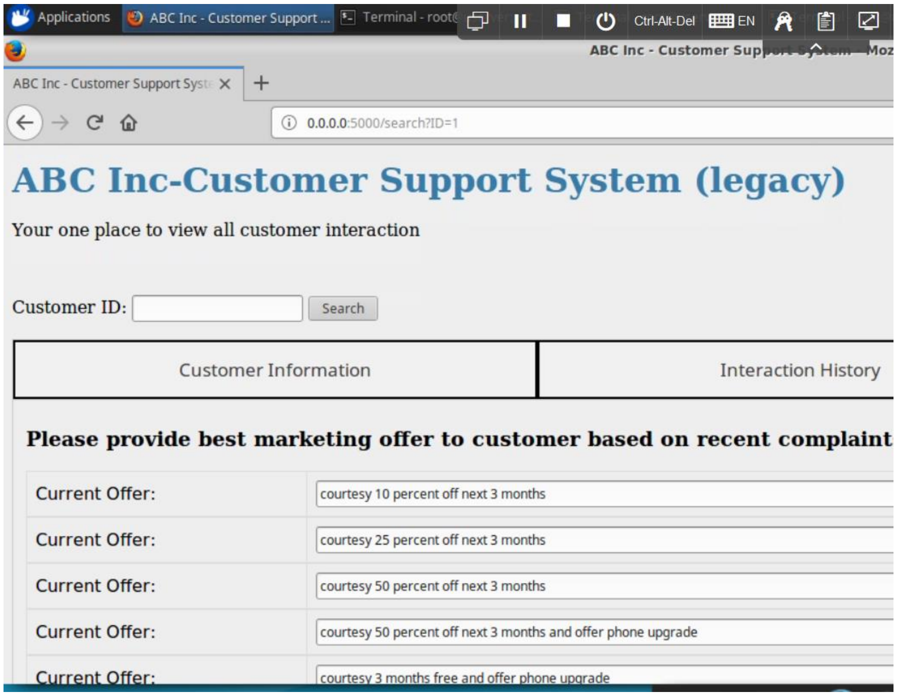
ABC Inc-Customer Support System

Your one place to view all customer interaction

Customer ID:

7.9. We are simulating a customer services representative’s application that captures customer interaction. Try entering 2381 in Customer ID field and click on “search” button. This will populate **Customer Information** tab with general attributes. **Interaction History** tab shows history of previous interaction for this customer. Finally, **Marketing Offers** tab shows current offers created by Marketing/retention department of the ABC Company.

Note : On Marketing Offers tab, there are instructions for Customer Service Rep to make a judgment call on type of offer to be presented based on recent complaints. In absence of ML model, it will be difficult to get most out of our marketing budget with potentially losing precious customers.


Applications
ABC Inc - Customer Support ...
Terminal - root@
Ctrl-Alt-Del
EN
ABC Inc - Customer Support System - Moz
ABC Inc - Customer Support Syst
0.0.0.0:5000/search?ID=1

ABC Inc-Customer Support System (legacy)

Your one place to view all customer interaction

Customer ID: Search

Customer Information	Interaction History

Please provide best marketing offer to customer based on recent complaint

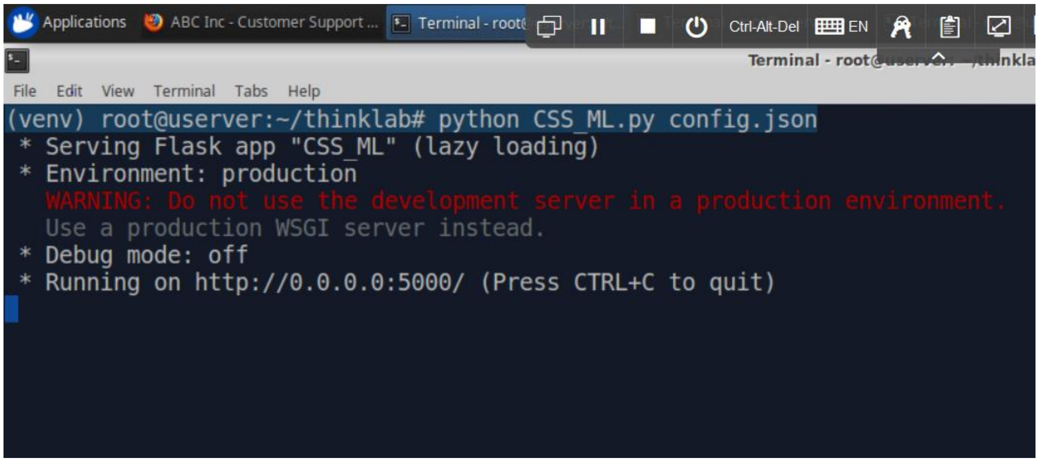
Current Offer:	courtesy 10 percent off next 3 months
Current Offer:	courtesy 25 percent off next 3 months
Current Offer:	courtesy 50 percent off next 3 months
Current Offer:	courtesy 50 percent off next 3 months and offer phone upgrade
Current Offer:	courtesy 3 months free and offer phone upgrade

Run Customer Support System App with ML webservice call

7.10. Lets use the model that was deployed by you in section 1.

7.11. Run the application using CSS_ML.py program as shown below:

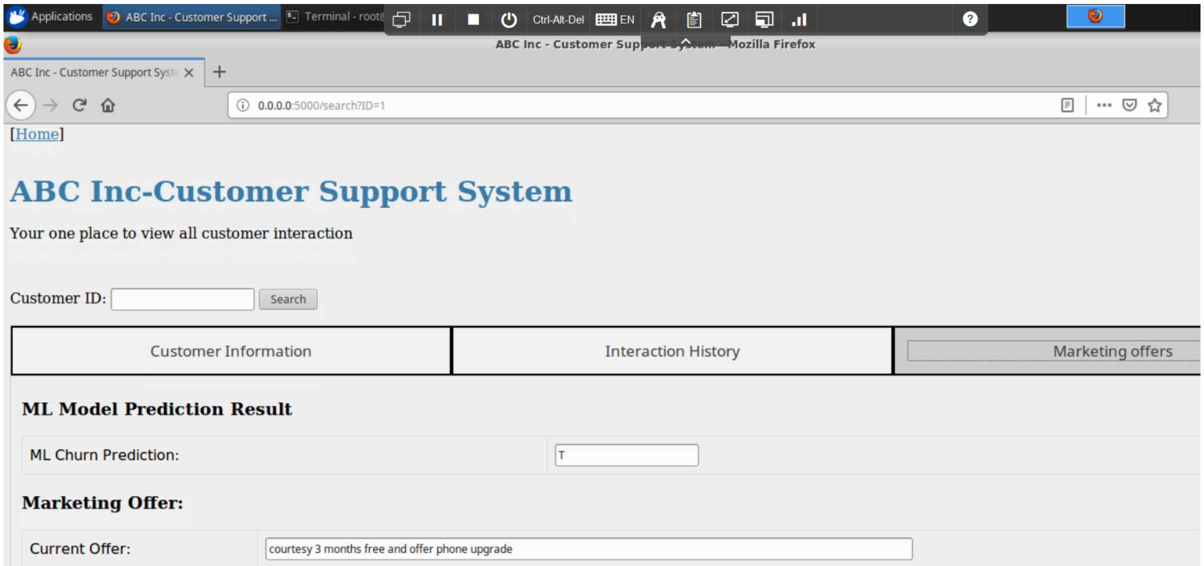
python CSS_ML.py config.json


Applications
ABC Inc - Customer Support ...
Terminal - root@
Ctrl-Alt-Del
EN
Terminal - root@userver: ~/thinkla
File Edit View Terminal Tabs Help

```
(venv) root@userver:~/thinklab# python CSS_ML.py config.json
 * Serving Flask app "CSS_ML" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

7.12. Go back to firebox browser and reenter URL : http://0.0.0.0:5000/

7.13. Try enter Customer ID = 1 . This customer is expected to be **Churned.** Click on Marketing Offers tab and see result :


Applications
ABC Inc - Customer Support ...
Terminal - root@
Ctrl-Alt-Del
EN
ABC Inc - Customer Support System — Mozilla Firefox
ABC Inc - Customer Support Syst
0.0.0.0:5000/search?ID=1

[Home]

ABC Inc-Customer Support System

Your one place to view all customer interaction

Customer ID: Search

Customer Information | Interaction History | Marketing offers

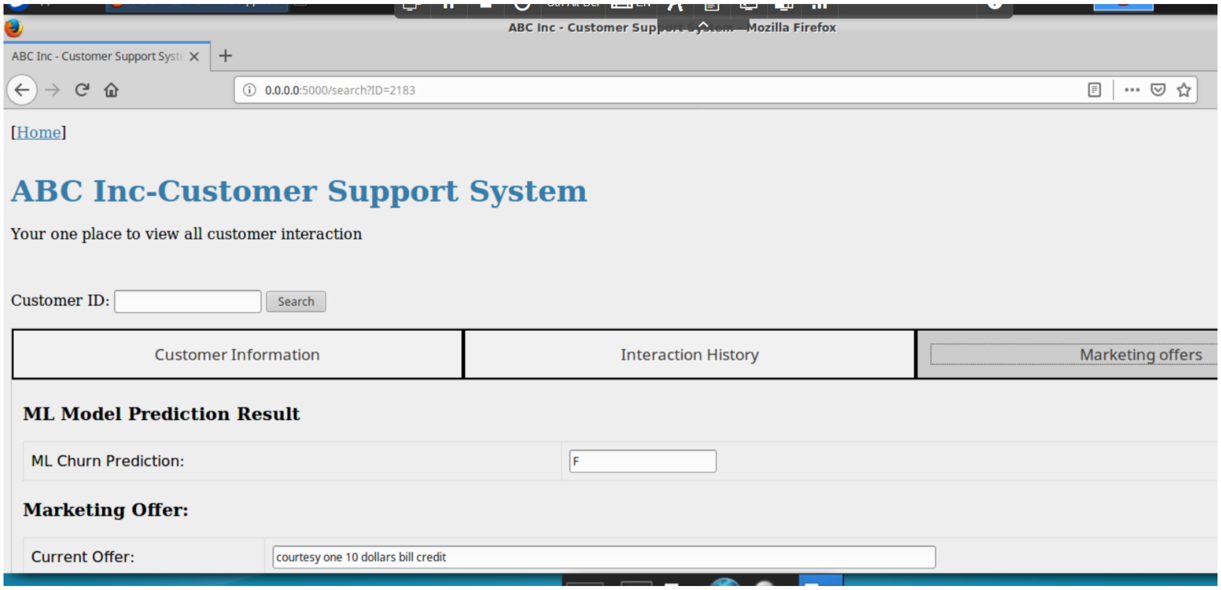
ML Model Prediction Result

ML Churn Prediction: T

Marketing Offer:

Current Offer: courtesy 3 months free and offer phone upgrade

7.14. Try entering Customer ID = 2183 . This customer is not expected to **Churn**. See the difference in Marketing Offers tab.


ABC Inc - Customer Support System — Mozilla Firefox
ABC Inc - Customer Support Syst
0.0.0.0:5000/search?ID=2183

[Home]

ABC Inc-Customer Support System

Your one place to view all customer interaction

Customer ID: Search

Customer Information | Interaction History | Marketing offers

ML Model Prediction Result

ML Churn Prediction: F

Marketing Offer:

Current Offer: courtesy one 10 dollars bill credit

Note that ML model was able to help the customer service ep to choose the Marketing Offers appropriate based on potential of the customer being churn or not.
The ML models enable various possibilities of enriching the applications in a way to augment process and make it more efficient.

7.15. Try to re-run the notebook from Section I and change the model. As long the name of model remains same, you can re-deploy the model and application will transparently start using newer version of the model. The ease of management and deployment is significant as many enterprise customers may spend significant amount of time and effort for that step.

This concludes the Section II of the lab.

Section III

8. Containerize application and deploy in ICP

In this section you will take the python flask application code and package it eb a docker container. The containers are most convenient way maximize server resource utilization. The ICP foundation where ICP for Data runs, provides ability to run such application containers and management of it.

8.1. Package your app as a Docker image

1 On your workstation, ensure that Docker is running:

```
# sudo docker run hello-world
```

2 Create a file named Dockerfile in the sample-project directory:

```
  # Specify the base image
FROM python:2.7

# Copy the requirements.txt file to the root directory of the Docker image
COPY requirements.txt ./

# Install the packages from the requirements.txt file
RUN pip install -r requirements.txt

# Copy the remaining source code into the Docker image
COPY . ./

# Specify the port on which the app will listen
EXPOSE 5001

# Configure a container that will run as an executable.
# When you start the container, it starts a bash shell and runs the Python app within that shell.
ENTRYPOINT [ "/bin/bash", "-c", "python CSS.py config.json"]
```

3 Build the Docker image by running the following command:

```
# docker build -t icp4d-app .
```

4 Confirm that the image was created successfully by running the following command:

```
# docker images  | grep icp4d-app
```

5 Run the image to confirm that it works successfully:

```
# docker run -p 5000:5000 icp4d-app
```

6 Confirm that the image works:

From your web browser, go to http://<host-ip>:5000

7 Save the image:

```
# docker save icp4d-app -o icp4d-app.tar.gz
```

8.2. Creating YAML configuration files

1 icp4d-app-deployment.yaml:

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: icp4d-app
  labels:
    app: icp4d-app
spec:
  template:
    metadata:
      labels:
        app: icp4d-app
    spec:
      containers:
      - name: icp4d-app
        image: 127.0.0.1:8500/apps/icp4d-app:v1
        ports:
        - containerPort: 5001
      imagePullSecrets:
      - name: regcred
```

2 icp4d-app-service.yaml:

```
apiVersion: v1
kind: Service
metadata:
  name: icp4d-app
  labels:
    run: icp4d-app
spec:
  type: NodePort
  ports:
    - port: 32456
      targetPort: 5001
      protocol: TCP
      name: http
    - port: 443
    protocol: TCP
    name: https
  selector:
   app: icp4d-app
```

8.3. Deploying your image to the cluster

1 On the master node (master-1) of the cluster, create a directory called /app1

2 Copy following three files to /app1 folder:

```
# cp icp4d-app.tar.gz *.yaml /app1
```

3 SSH to the master node (master-1) of the cluster as root.

4 Load the image:

```
# docker load -i icp4d-app.tar.gz
```

5 Log in to Docker Registry:

```
# docker login mycluster.icp:8500 -u username
```

Enter your password when prompted. The default user name is admin and the default password is admin.

6 Obtain the ID of your image:

```
# docker images | grep icp4d-app
```

7 Tag your image with the Docker Registry information and with version information:

```
# docker tag <image id> mycluster.icp:8500/apps/icp4d-app:v1
```

8 Create a namespace for your custom apps:

```
# kubectl create namespace app1
```

9 Push the image to Docker Registry:

```
# docker push mycluster.icp:8500/app1/icp4d-app:v1
```

10 Create a Secret in the cluster that holds your authorization token

```
# kubectl create secret docker-registry regcred --docker-server=mycluster.icp:8500 --docker-username=admin --docker-password=admin --docker-email=<your email id>
```

11 Edit the icp4d-app-deployment.yaml file on the master node to update the name of the image.

Change image: icp4d-app to image: mycluster.icp:8500/app1/icp4d-app:v1 and add the created secret to the files as well.

12 Create the deployment:

```
# kubectl create -f icp4d-app-deployment.yaml
```

13 Check the new pod has been created and status is running (ready):

```
# kubectl get po --all-namespaces | grep icp4d-app
```

14 Create the service:

```
# kubectl create -f icp4d-app-service.yaml
```

15 Determine the port where you can communicate with the app:

```
# kubectl describe service icp4d-app
```

Result should be like this:

```
Name:                     icp4d-app
Namespace:                default
Labels:                   run=icp4d-app
Annotations:              <none>
Selector:                 app=icp4d-app
Type:                     NodePort
IP:                       10.0.164.31
Port:                     http  32456/TCP
TargetPort:               5001/TCP
NodePort:                 http  32142/TCP
Endpoints:                172.30.254.92:5001
Port:                     https  443/TCP
TargetPort:               443/TCP
NodePort:                 https  32242/TCP
Endpoints:                172.30.254.92:443
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

16 Verify that the app is running on the cluster. From your web browser, go to

```
http://MASTER_1_IP:32142
```

We Value Your Feedback!

- Don’t forget to submit your Think 2019 session and speaker feedback! Your feedback is very important to us – we use it to continually improve the conference.
- Access the Think 2019 agenda tool to quickly submit your surveys from your smartphone, laptop or conference kiosk.