

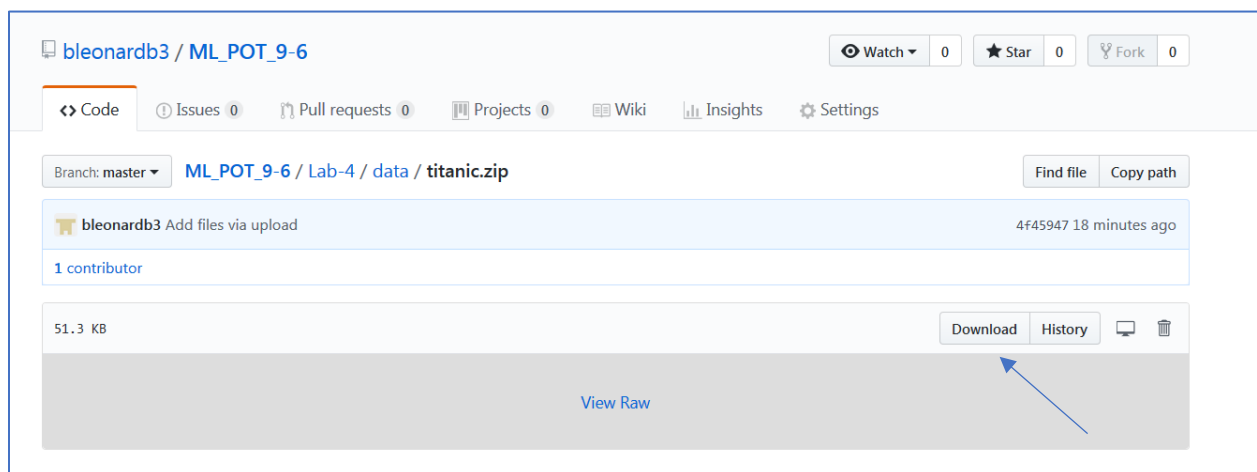
Watson Machine Learning Overview

This lab will introduce the Watson Machine Learning capability using the Titanic dataset. The lab will consist of the following steps:

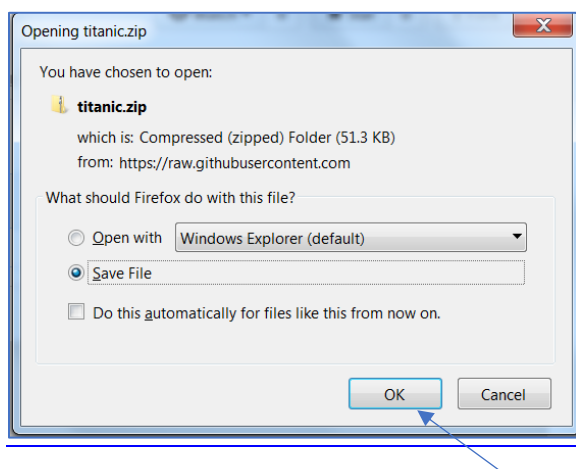
1. Adding a data asset to the Watson Studio Labs project
2. Creating a Model to predict whether a passenger would survive
3. Deploying and Testing the Model
4. Deploying a simple web front-end and connecting it to the Titanic deployed model.


Step 1: Adding a Data Asset to the project

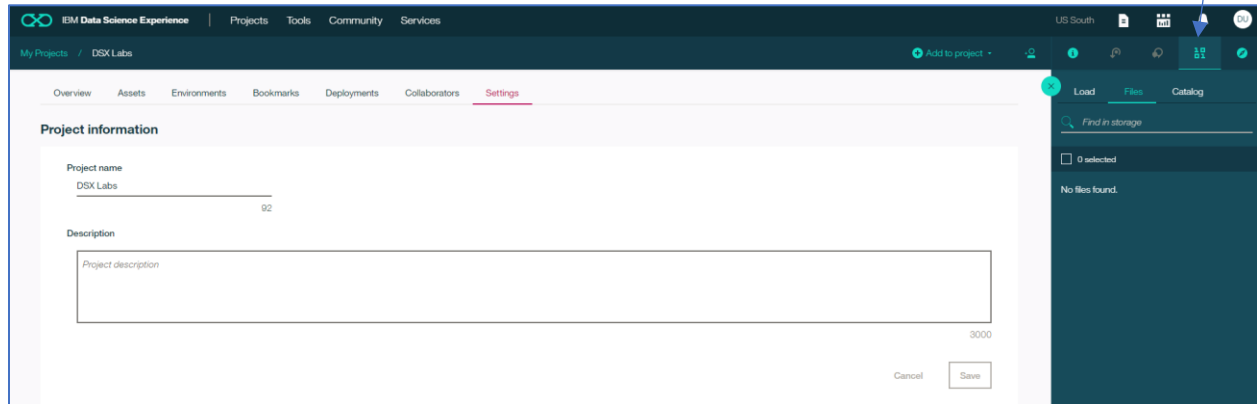
1. Download the Titanic data file from the following location by clicking on the link [Titanic Data](#) and then clicking on the **Download** button.



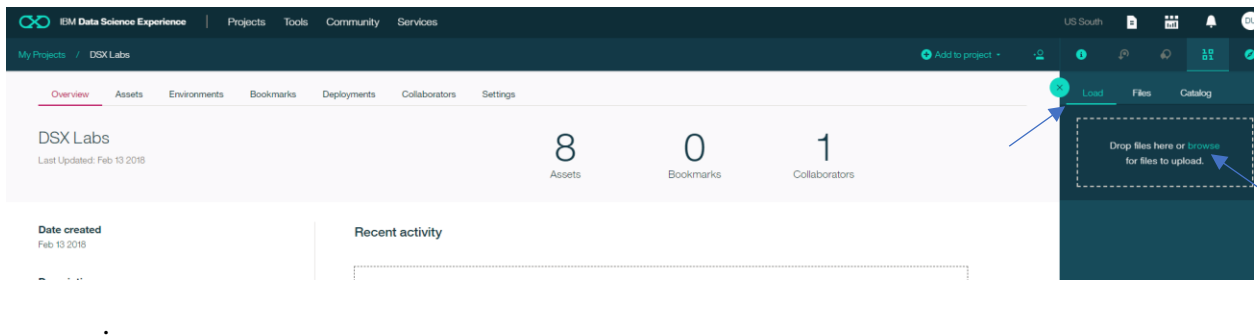
2. Click on the **OK** button in the pop-up dialog.



3. Navigate to the directory where the file has been downloaded. Unzip the titanic.zip file. There should be two files (1) titanic_cleansed.csv and (2) titanic.csv. You will use the **titanic_cleansed.csv** for this lab.
4. Go back to your Watson Studio Labs project. Click on the  icon.

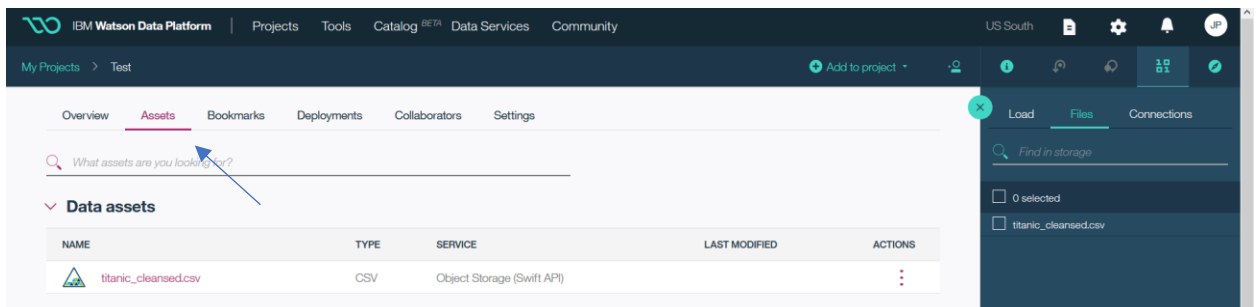


3. Click on **Load** and then **browse** and then go to the folder where the titanic_cleansed.csv is stored. Select titanic_cleansed.csv and then click Open.

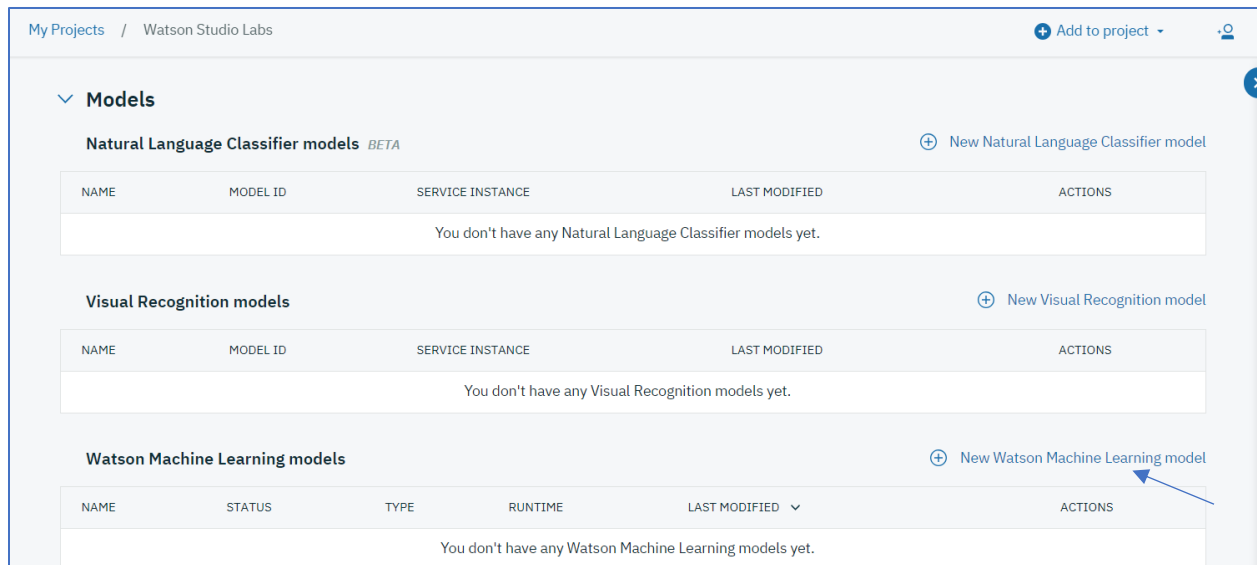


Step 2: Create a Model to predict survival

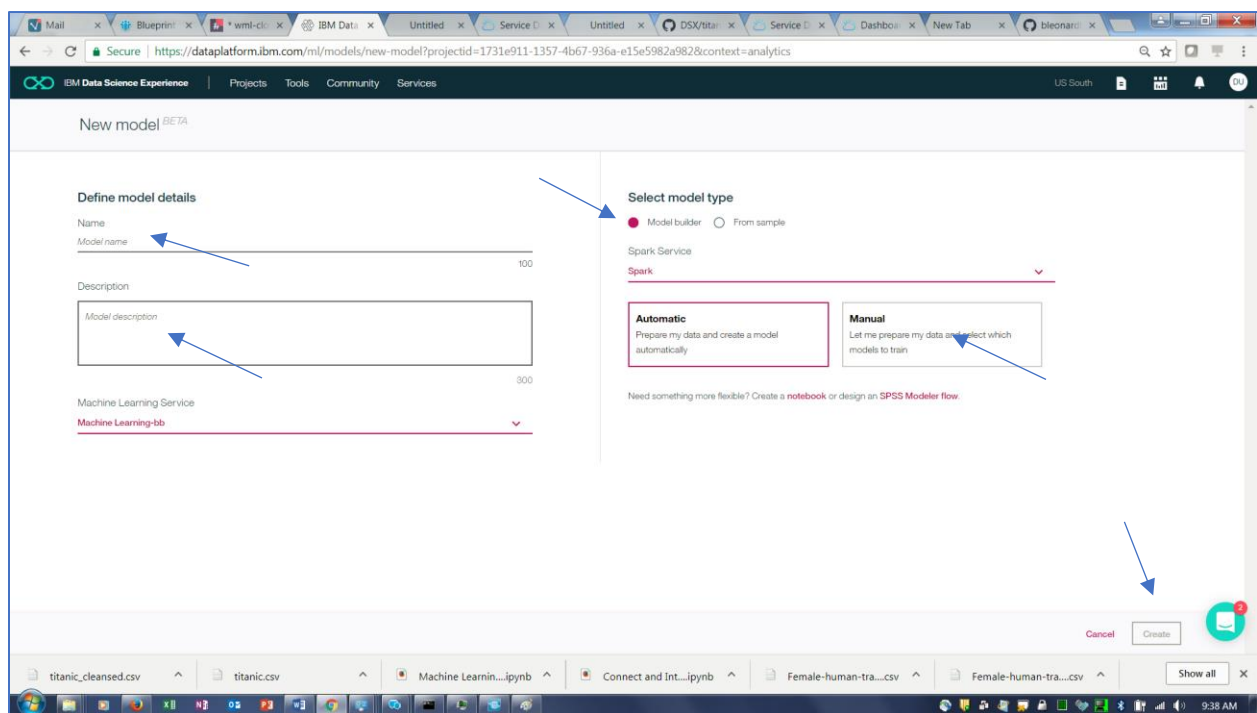
1. Click on the **Assets** Tab



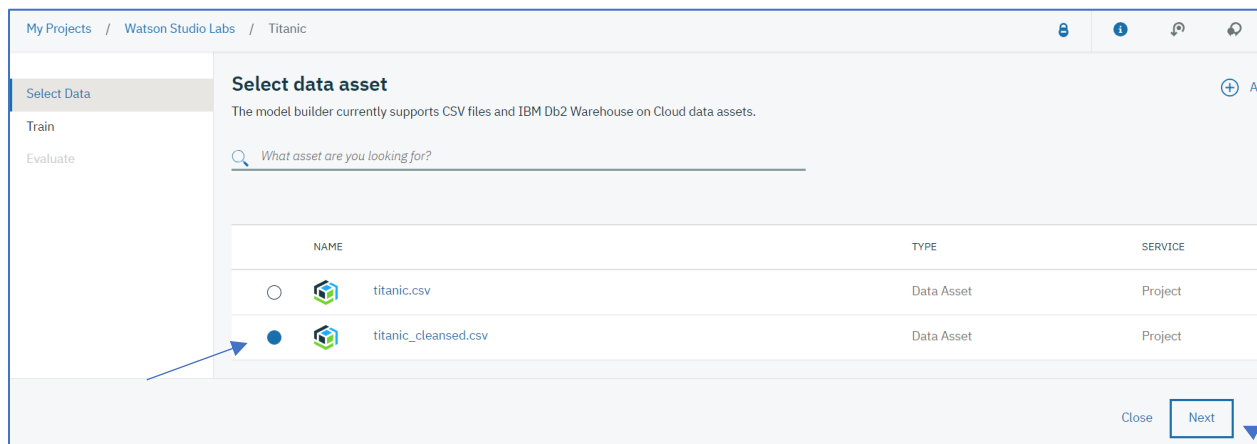
2. Click on **New Watson Machine Learning model**.



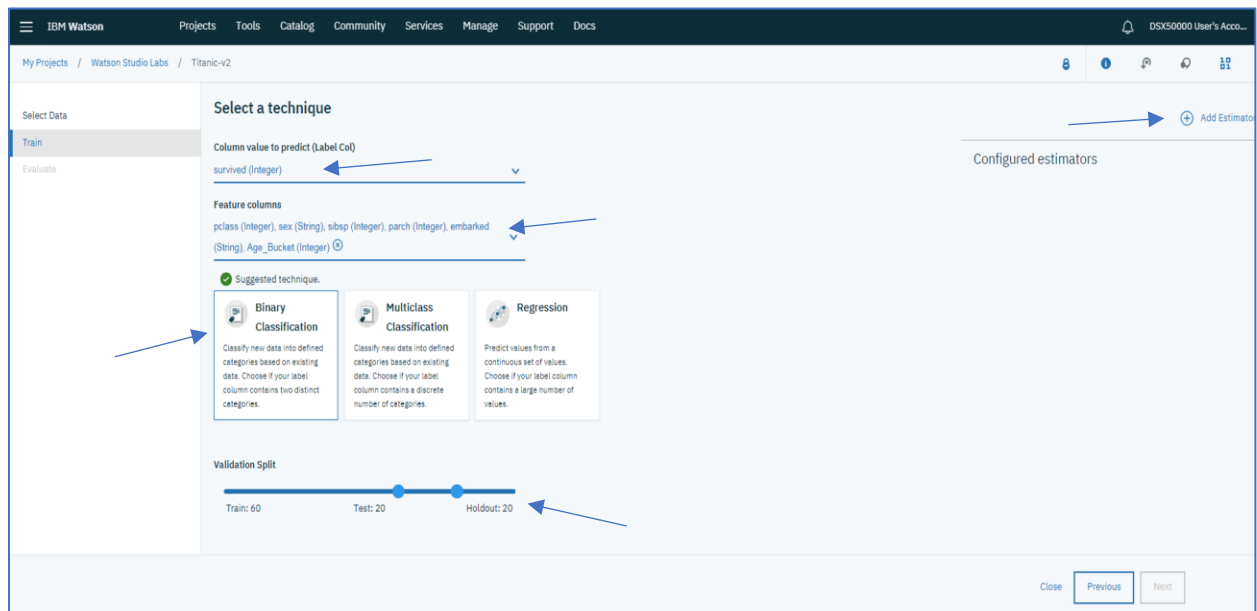
3. Enter a **Model Name** (eg Titanic), optionally a **Description**, leave **Model Builder** selected, select **Manual**, and click on **Create**.



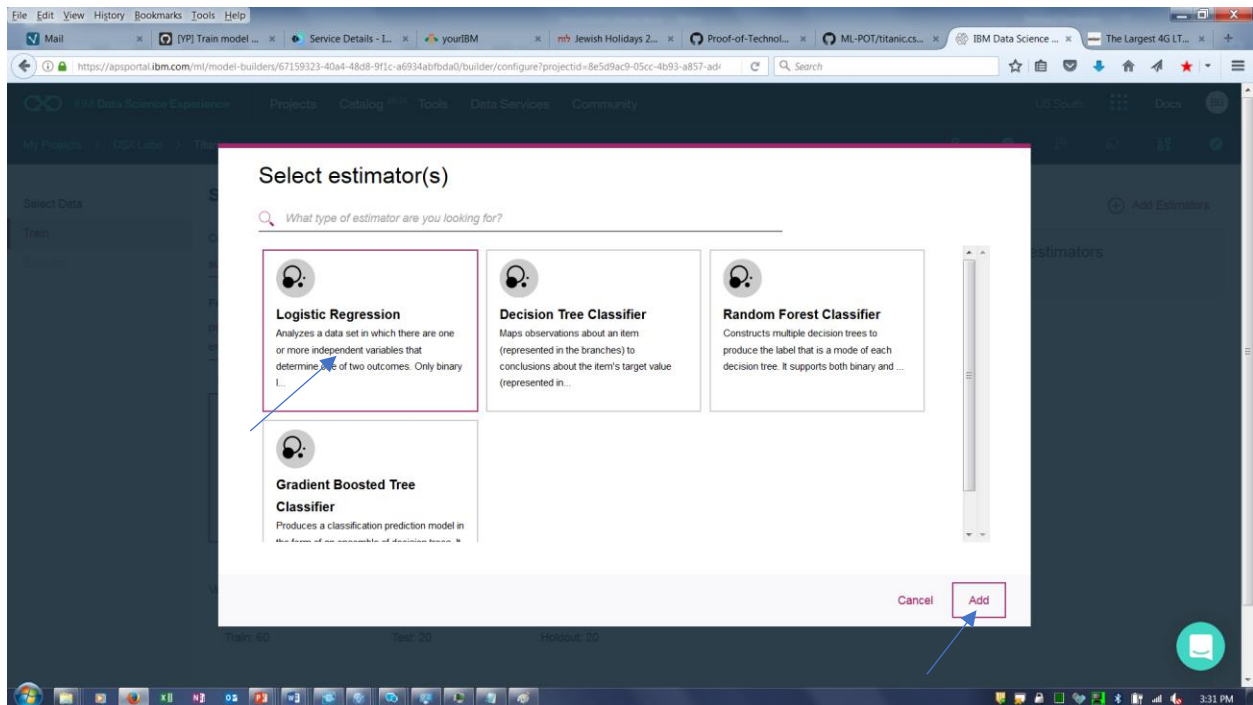
4. Click on the **titanic_cleansed.csv** and click on **Next**



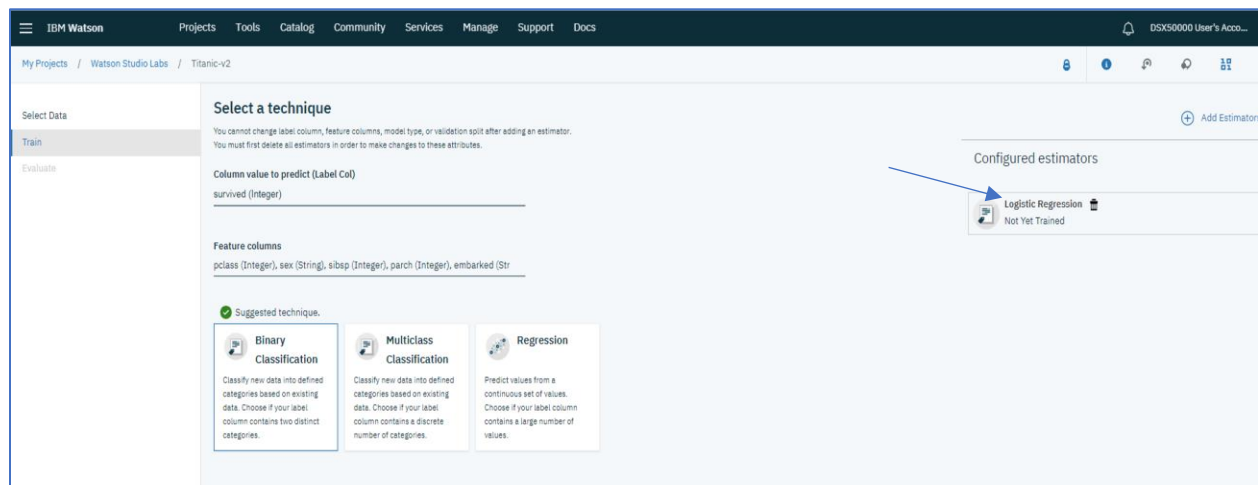
- For **Column value to predict (Label Col)** select **survived**. For **Feature columns** select the following features (**pclass, sex, sibsp, parch, embarked, and Age_Bucket**) . Click on the **Binary Classification** Box (which is suggested by the service). Adjust the **Validation Split** as desired. Click on **Add Estimators** to add the specific models to use.



- Select **Logistic Regression**. You can select more if you wish to see the results of multiple models. Select **Add**.




7. Note you can adjust the algorithm's hyper-parameters by clicking on Logistic Regression.



8. For now we will leave the hyper-parameters unchanged, so click **Cancel**.

Configure Logistic Regression

Weight column
[Select](#) 

Elastic net parameter (double)
0


Fit an intercept
☐ `fitIntercept`


Maximum iterations for convergence (integer)
100

Regularization parameter (double)
0

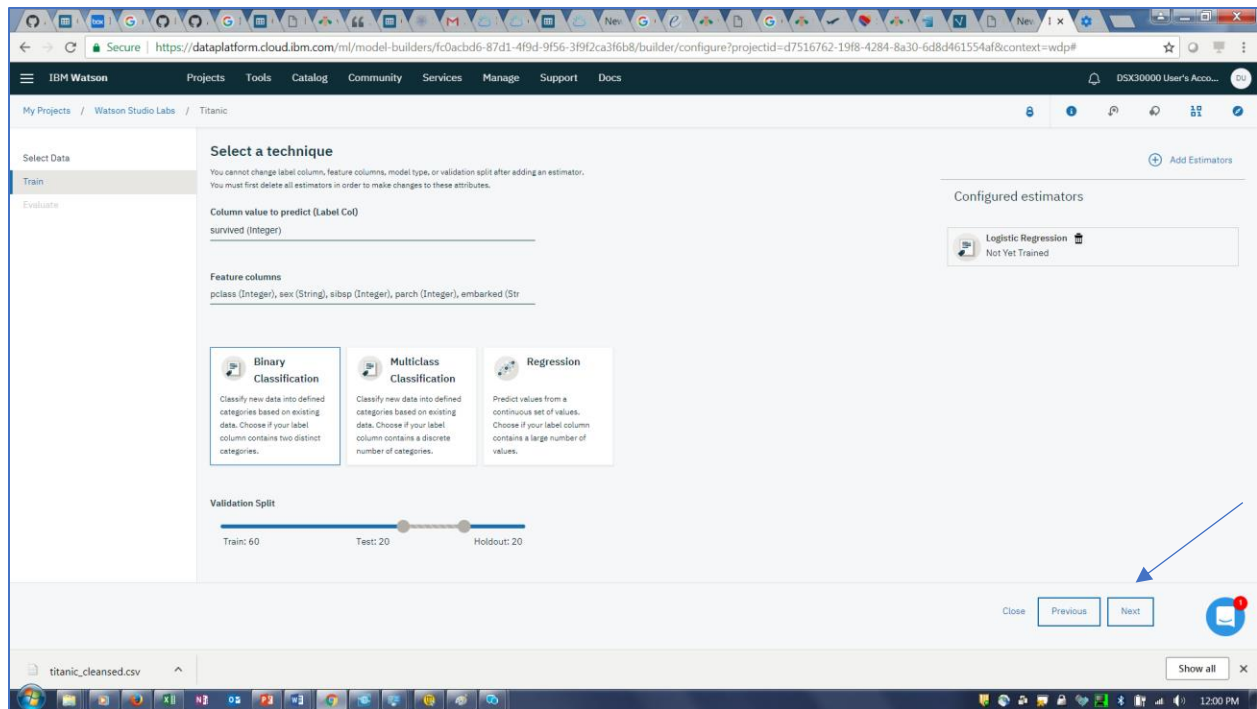
Standardize the training features before fitting the model
☐ `standardization`

Threshold (double)
0.5



 [Cancel](#) [Save](#)

9. Select the **Next** button.

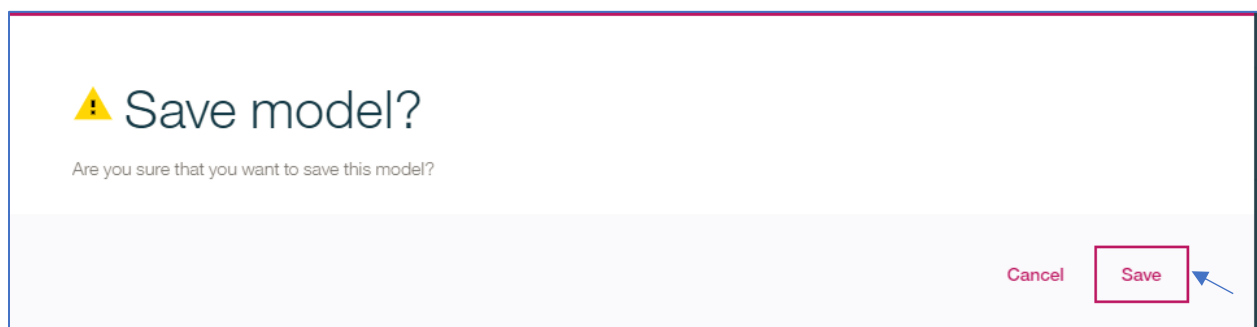


10. The system trains and evaluates each model. If more than one model was selected, the models would be listed in descending order of quality with the best result at the top. Note: if a model fails to run (rare, but happens), select Previous, delete the estimator, and re-add it. Then run again. Click on **Logistic Regression** (if it is the best) and then click **Save**.

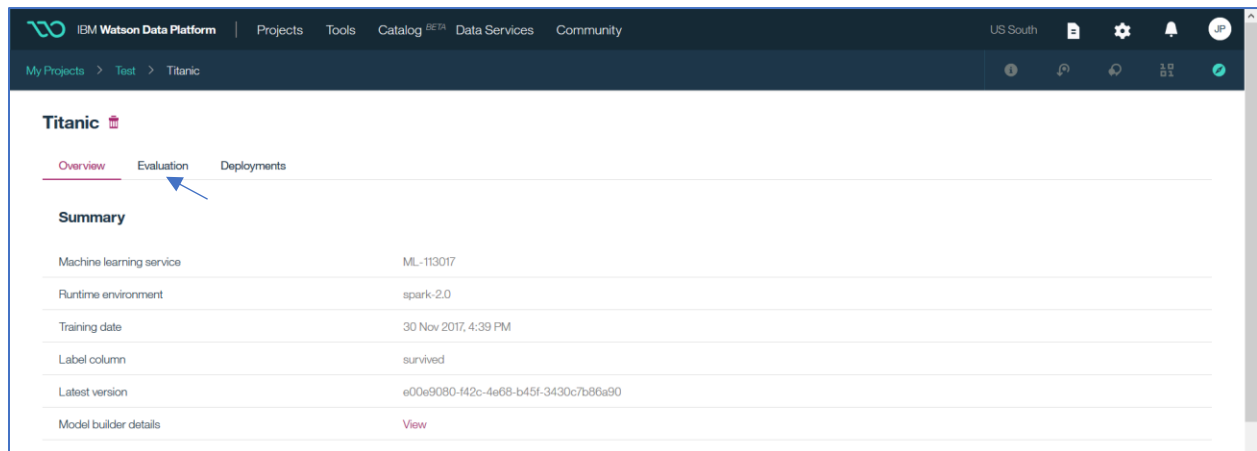
The screenshot shows the 'Select model' screen with a table of model performance metrics. A blue arrow points to the 'LogisticRegression' row, and another blue arrow points to the 'Save' button.

ESTIMATOR TYPE	STATUS	PERFORMANCE	AREA UNDER ROC CURVE	AREA UNDER PR CURVE	LAST EVALUATION
LogisticRegression	Trained & Evaluated	Good	0.88379	0.87048	22 Jul 2018, 12:08 PM

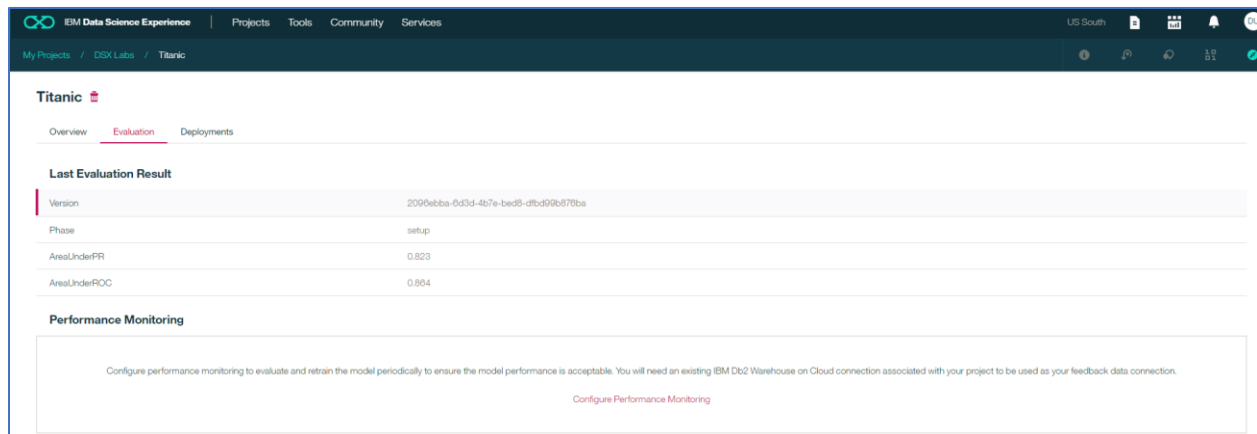
11. Click **Save** again on the next screen.



12. The system displays the model training summary. Click on Evaluation.



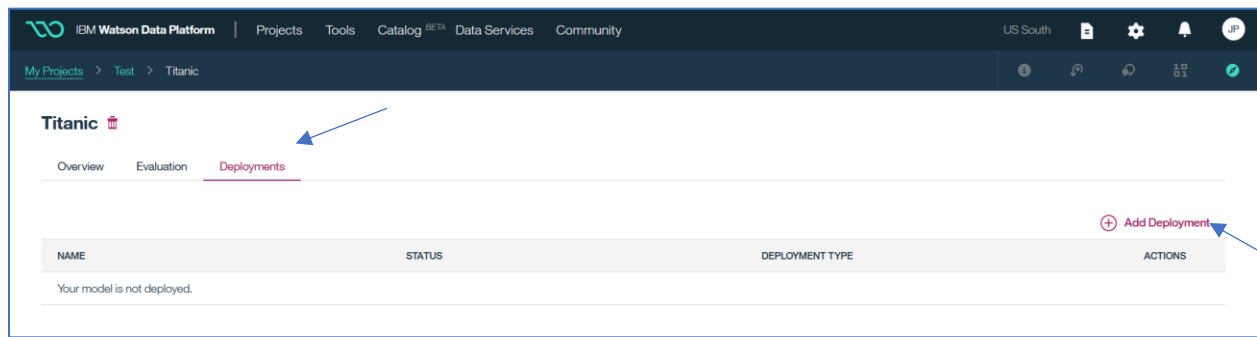
13. The system displays the recorded evaluation statistics for the run. You can also set up Continuous Learning (Performance Monitoring) on this screen. We will not do this now.



Step 3: Deploying a Model

We can deploy the model to enable applications to invoke it via an API call. This is called a Web Service deployment or Online deployment.

1. Select the **Deployments** Tab
2. Click on **Add Deployment**



- Three options for deployment are available. For this lab, we are going to embed the predictive model scoring in a web application. Therefore, select the **Web Service** tab, enter **Titanic_Deployment** for **Name**, and click on **Save**.

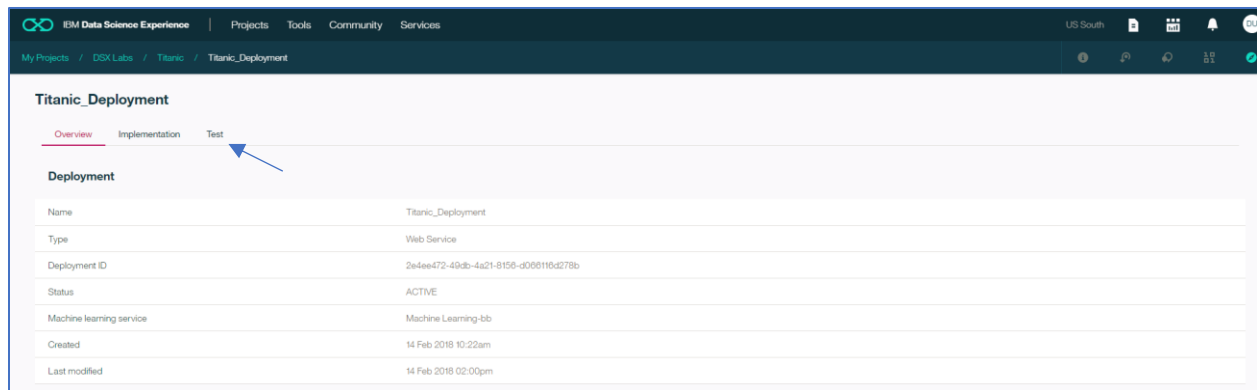
The screenshot shows the 'Create Deployment' form. The 'Name' field is filled with 'Titanic_Deployment', the 'Description' field is empty, and the 'Web service' deployment type is selected. The 'Save' button is highlighted with a blue arrow.

- The system responds with an acknowledgement that the model was successfully deployed. Click on **Titanic_Deployment** to test the deployed API.

The screenshot shows the IBM Watson Data Platform interface for the 'Titanic' project. The 'Deployments' tab is selected, and the 'Titanic_Deployment' entry is highlighted with a blue arrow. The table below shows the current deployment status.

NAME	STATUS	DEPLOYMENT TYPE	ACTIONS
Titanic_Deployment	DEPLOY_SUCCESS	Web Service	

- The system displays information about the deployed service. Click on **Test** to test out the API.



6. Enter values for the following fields:

pclass - 1

sex – female

sibsp (number of siblings or spouses) – 0

Scroll down continue entering values for:

parch (number of parents or children) – 0

embarked – S

Age_Bucket - 1

and then click on **Predict**. Note that any values inputted for any of the fields not included in the model parameters (e.g. name) will not affect the prediction.

Titanic_Deployment

Overview Implementation Test

Enter input data

☰

📄

pclass

1

name

sex

female

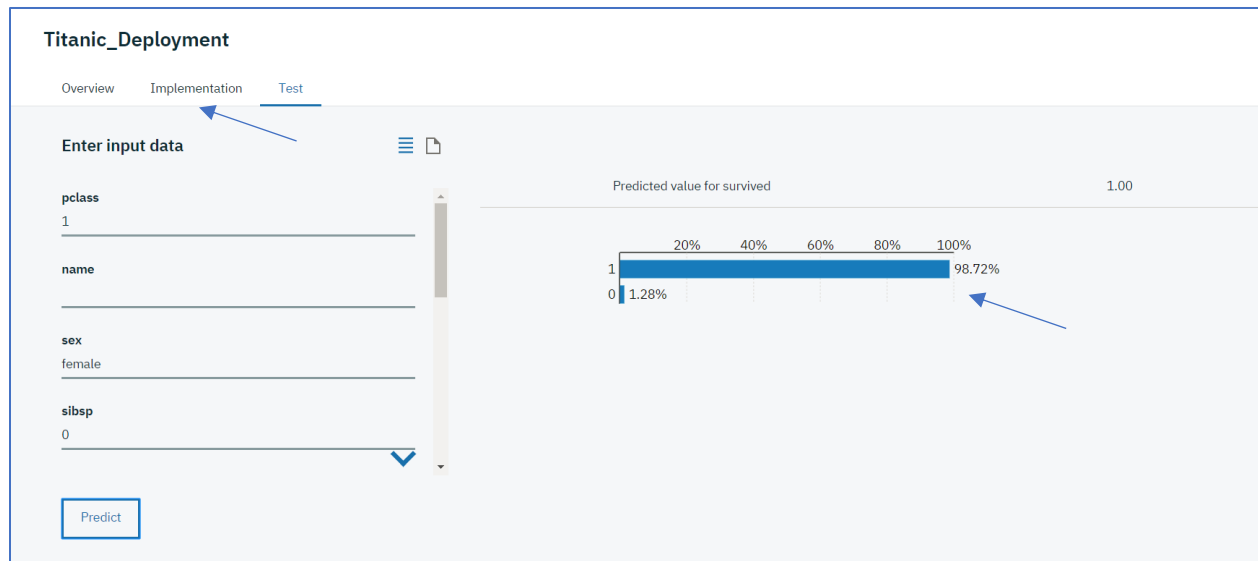
sibsp

0

✓

Predict

7. The predicted result should be returned which indicates that the model has been successfully deployed. Now click on the **Implementation** tab.



8. The Implementation panel provides information for the application developers to invoke the deployed model. It includes sample code in various programming languages and the scoring endpoint to be used when invoking the web service. Open Windows Notepad to copy and paste the scoring endpoint. We will be using this endpoint in Step 4.

Titanic_Deployment

Overview **Implementation** Test

Implementation

View API Specification

Scoring End-point: https://bm-watson-ml.mybluemix.net/v3/wml_instances/c4862b9a-5cb3-4d8e-9884-5255e55484e5/published_models/1485e1f-818a-4239-8db5-37312d8d32a7/deployments/2e4ee472-48db-4a21-8150-d06116d27bb/online

Authorization: Bearer <token> See code snippets below for information on how to retrieve the WML Authorization Token to be passed with scoring requests.

Content-type: application/json Required if the request body is sent in JSON format.

Code Snippets

cURL Java JavaScript Python Scala

```
# retrieve your $WML_SERVICE_CREDENTIALS_USERNAME, $WML_SERVICE_CREDENTIALS_PASSWORD, and $WML_SERVICE_CREDENTIALS_URL from the
# Service credentials associated with your IBM Cloud Watson Machine Learning Service Instance

curl --basic --user $WML_SERVICE_CREDENTIALS_USERNAME:$WML_SERVICE_CREDENTIALS_PASSWORD $WML_SERVICE_CREDENTIALS_URL/v3/identity/token

# the above CURL request will return an auth token that you will use as $WML_AUTH_TOKEN in the scoring request below
# TODO: manually define and pass values to be scored below
curl -X POST --header 'Content-type: application/json' --header 'Accept: application/json' --header 'Authorization: Bearer $WML_AUTH_TOKEN' -d '{"fields": ["pclass", "name", "sex", "sibsp", "parch", "ticket", "fare", "embarked", "Age"]}'
```

Step 4: Deploy a simple web front-end to invoke the Watson Machine Learning service

This section will provide an example of a simple Python Flask web front-end application that invokes the Titanic scoring API demonstrating embedding machine learning in a web app. You

will click on a link below that will deploy the sample Python web application into your IBM Cloud account. A toolchain will be set up for continuous delivery of the application. The application code will be cloned from a public Git repository into a private Git repo in your account that will be set up as part of the toolchain. Each time you commit changes to the repo, the app will be built and deployed.

The toolchain uses tools that are part of the Continuous Delivery service. If an instance of that service isn't already in your organization, when you click **Deploy**, it is automatically added with the free [Lite](#) plan selected.

You will need to customize the application to provide the credentials for your Watson Machine Learning service, and to provide the scoring endpoint.

1. Click on the **Deploy to Bluemix** link below to deploy a sample Python Flask web application into your IBM Cloud account. Note you may get this message – “*An IBM Cloud account is required. To get started, click Log In or Sign Up at the top of this page*”. If you get this message, click on **Log In**.

[Deploy to Bluemix](#)

2. Scroll down to the bottom. Click on the **Create+** button to create an IBM Cloud API key.

The Delivery Pipeline automates continuous deployment.

App name: ⓘ

IBM Cloud API Key: ⓘ

*The value is required.

Region **Organization** **Space**

*The value is required. *The value is required. *The value is required.

Create +

3. Click on the **Create** button.

×

Create API key

The following API key will be created to deploy your application from the Delivery Pipeline.

API Key for Titanic-20180722183628794

API keys can be managed from **Manage > Security > Platform API keys**.

Cancel

Create

- Please wait until the Region, Organization, and Space are filled in. **Note that these must match the corresponding Region, Organization, and Space where the Titanic model was deployed.** Unless you changed the default space (dev) on creating your Watson Studio account, they should match. Otherwise update the Space value, if needed (if you have multiple spaces defined you will see them listed when clicking on the down arrow in the Space field). Click on the **Deploy** button

The Delivery Pipeline automates continuous deployment.

App name: ⓘ

Titanic-20180722183628794

IBM Cloud API Key: ⓘ

..... 👁

Create +

Region

Organization

Space

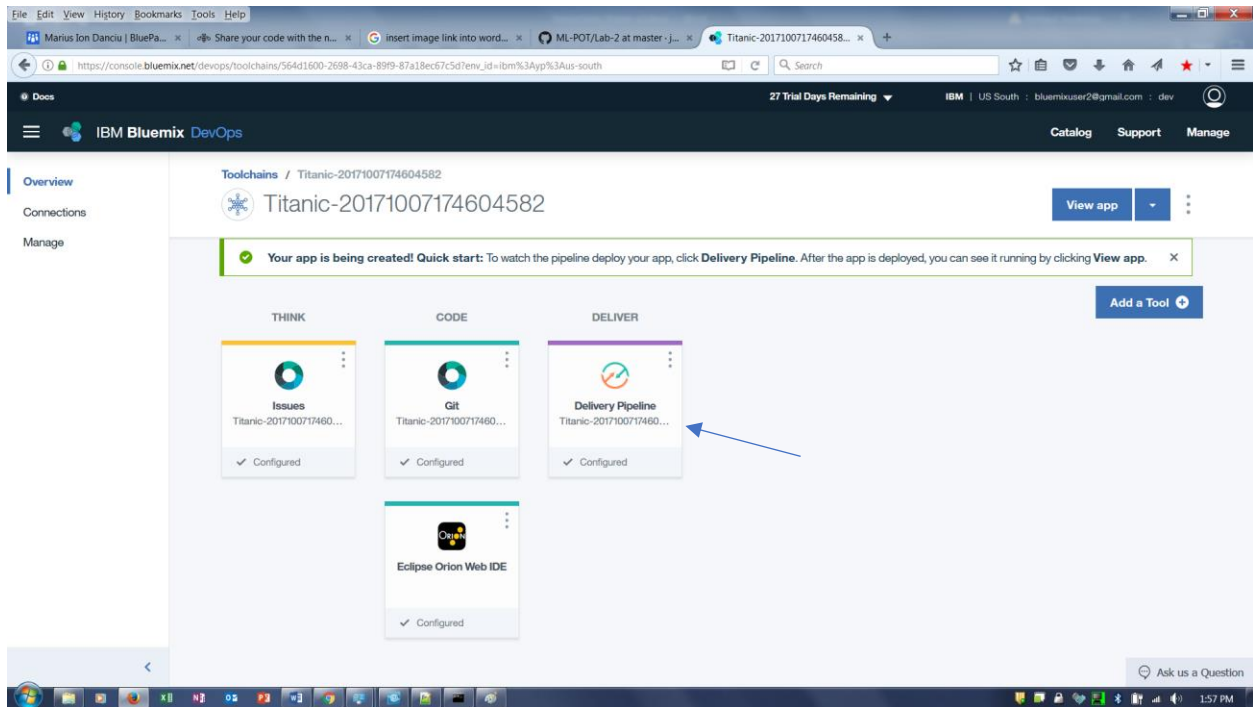
US South (Production) ▼


DSXUser30000@gm... ▼

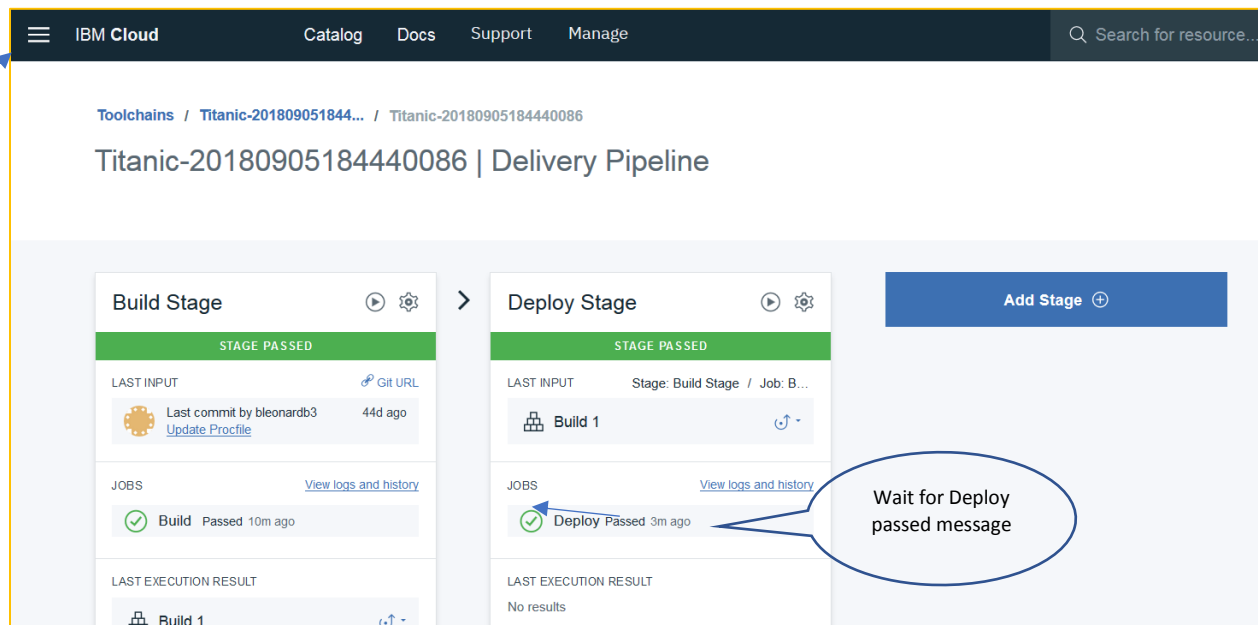
dev ▼

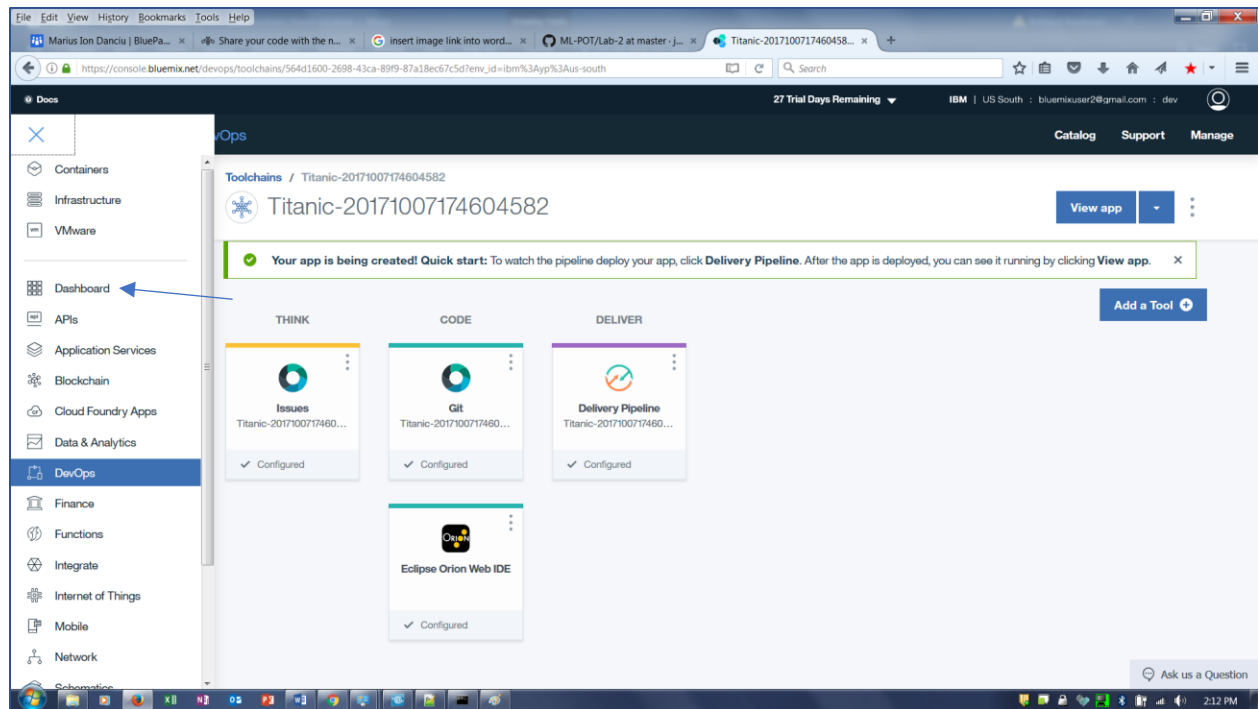
Deploy

5. Your app is being created! To watch the pipeline deploy your app, click **Delivery Pipeline**.

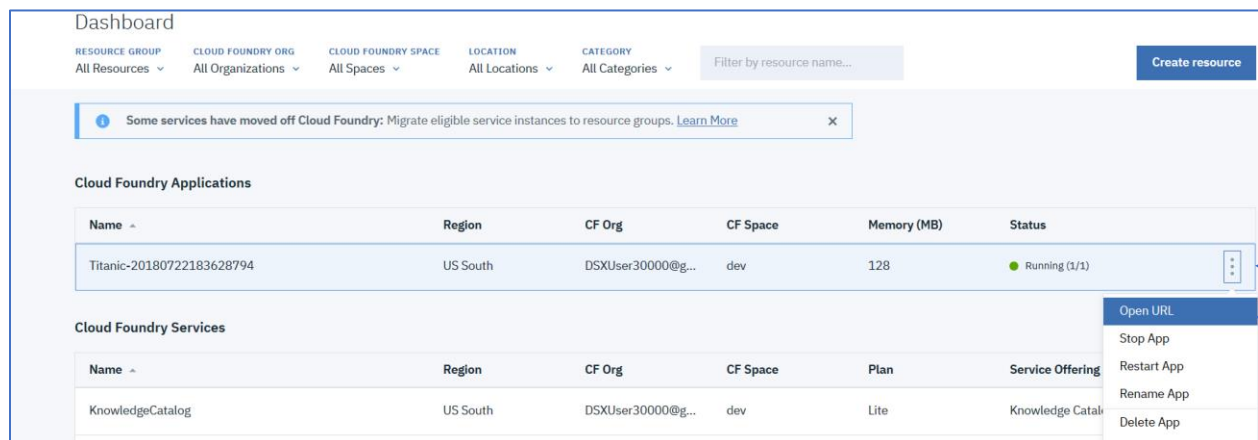


6. After the app is deployed successfully (should say Deploy Passed in the Deploy stage- may take about 2 minutes), view the running app by clicking the hamburger  icon in the left-hand corner and **Dashboard** in the pulldown to navigate to the **Dashboard** where the running application should be listed.





7. Click on the vertical ellipsis on the right-side of the Titanic application listing and then click on **Open URL** to view the web application. If the **Open URL** option does not appear (sometimes it is delayed), you can click on the Titanic application name, and on the next screen, click on Visit App URL



8. The web form collecting the Titanic passenger data should appear. Note that the application is not functional until we connect it to the Watson Machine Learning service so if you Submit you will get an error! Close the Titanic Prediction tab.
9. We are now going to connect the application to the Watson Machine Learning service that was created earlier. Click on the application name.

The screenshot shows the IBM Bluemix Dashboard. At the top, there's a navigation bar with 'Catalog', 'Support', and 'Manage' links. Below the navigation bar is a search bar and a 'Create' button. The main content area is titled 'Dashboard' and shows 'Cloud Foundry Apps (1)' with a table of 1 item. The table has columns: NAME, ROUTE, MEMORY (MB), INSTANCES, RUNNING, STATE, and ACTIONS. The first row shows an app named 'Titanic-20171007174604582' with a route 'titanic-20171007174604582-jackleg-pirogue.myblu', 128 MB memory, 1 instance, 1 running, and a 'Running' state. Below the apps table is a 'Services (5)' section showing a table of 5 services. The services are: Continuous Delivery, Data Science Experience-gz, DSX-ObjectStorage, DSX-Spark, and Watson Machine Learning. A blue arrow points to the 'Titanic-20171007174604582' app in the table.

NAME	ROUTE	MEMORY (MB)	INSTANCES	RUNNING	STATE	ACTIONS
Titanic-20171007174604582	titanic-20171007174604582-jackleg-pirogue.myblu	128	1	1	Running	

NAME	SERVICE OFFERING	PLAN	ACTIONS
Continuous Delivery	Continuous Delivery	Free (Deprecated)	
Data Science Experience-gz	Data Science Experience	Lite	
DSX-ObjectStorage	Object Storage	Lite	
DSX-Spark	Apache Spark	Lite	
Watson Machine Learning	Machine Learning	Free	

10. Scroll down until you see the Connections panel. Click on **Create Connection**.

The screenshot shows the IBM Cloud dashboard for a specific app. The left sidebar has a 'Connections' link. The main content area shows the app details: 'Titanic-20171130220547025' with a 'Running' status. Below this, there are four circular gauges: 'BUILDPACK' (Python), 'INSTANCES' (1), 'MB MEMORY PER INSTANCE' (128), and 'TOTAL MB ALLOCATION' (128). Below the gauges is a 'Connections' panel with the text 'No services are connected to this app. You can bind a service:' and a 'Create connection' button. A blue arrow points to the 'Create connection' button. To the right of the 'Connections' panel is a 'Runtime cost' section showing 'Current charges for billing period' as \$0.00 and 'Estimated total for billing period' as \$0.00.

Cloud Foundry apps /

Titanic-20171130220547025 Running [Visit App URL](#)

Org: jpatter@us.ibm.com Location: US South Space: dev

BUILDPACK
Python

INSTANCES
All instances are running
Health is 100%

MB MEMORY PER INSTANCE
128

TOTAL MB ALLOCATION
128
7,875 GB still available

Connections

No services are connected to this app
You can bind a service:

[Create connection](#)

Runtime cost

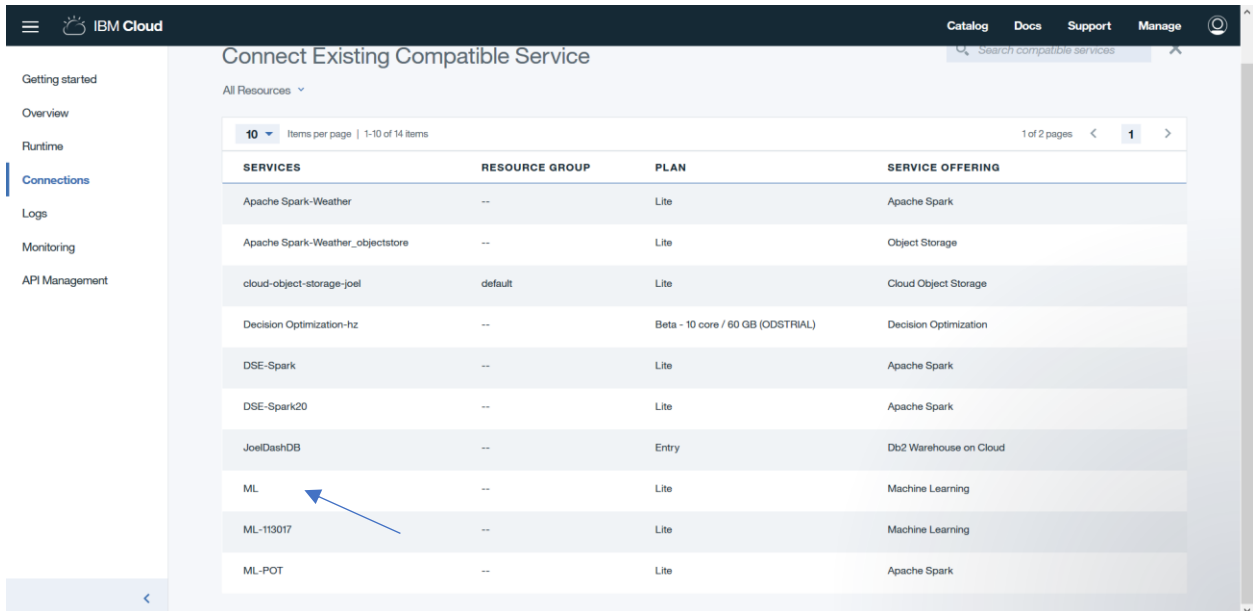
\$0.00
Current charges for billing period

\$0.00
Estimated total for billing period
(Nov 1, 2017 - Nov 30, 2017)

Current and estimated cost excludes [connected services](#).

[View full usage details](#)

11. You should see at least 3 services listed, a Cloud Object Storage service, a Spark service, and a Watson Machine Learning service. Click on the **Machine Learning** service for your application, and then click on **Connect**.



IBM Cloud

Catalog Docs Support Manage

Search compatible services

Getting started

Overview

Runtime

Connections

Logs

Monitoring

API Management

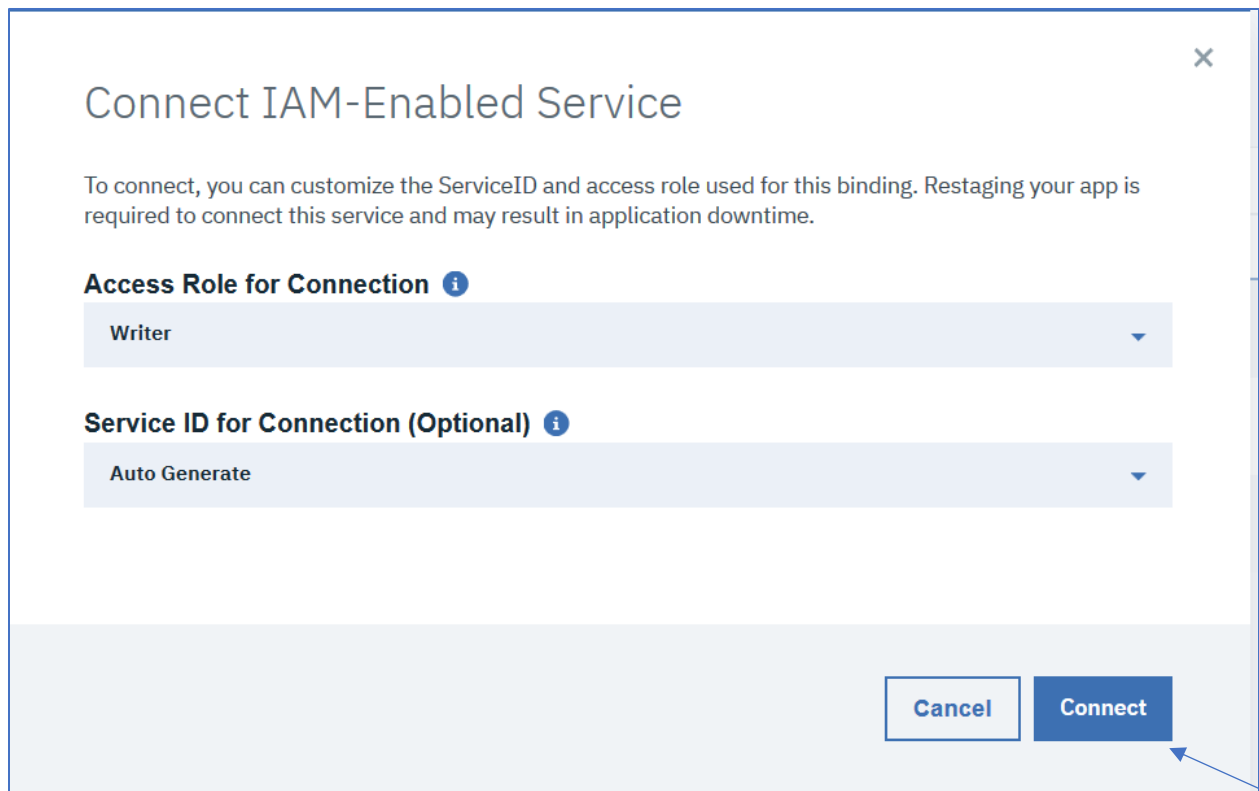
Connect Existing Compatible Service

All Resources

10 Items per page | 1-10 of 14 items 1 of 2 pages < 1 >

SERVICES	RESOURCE GROUP	PLAN	SERVICE OFFERING
Apache Spark-Weather	--	Lite	Apache Spark
Apache Spark-Weather_objectstore	--	Lite	Object Storage
cloud-object-storage-joel	default	Lite	Cloud Object Storage
Decision Optimization-hz	--	Beta - 10 core / 60 GB (ODSTRIAL)	Decision Optimization
DSE-Spark	--	Lite	Apache Spark
DSE-Spark20	--	Lite	Apache Spark
JoelDashDB	--	Entry	Db2 Warehouse on Cloud
ML	--	Lite	Machine Learning
ML-113017	--	Lite	Machine Learning
ML-POT	--	Lite	Apache Spark

12. A **Connect IAM-enabled service** pop up will appear. Just click **Connect**.



×

Connect IAM-Enabled Service

To connect, you can customize the ServiceID and access role used for this binding. Restaging your app is required to connect this service and may result in application downtime.

Access Role for Connection ⓘ

Writer

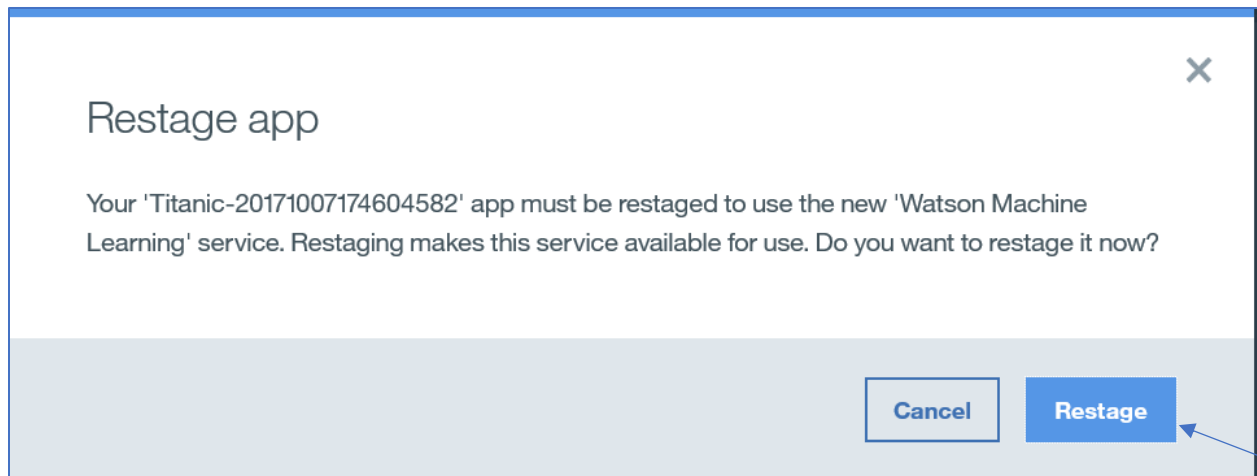
Service ID for Connection (Optional) ⓘ



Auto Generate

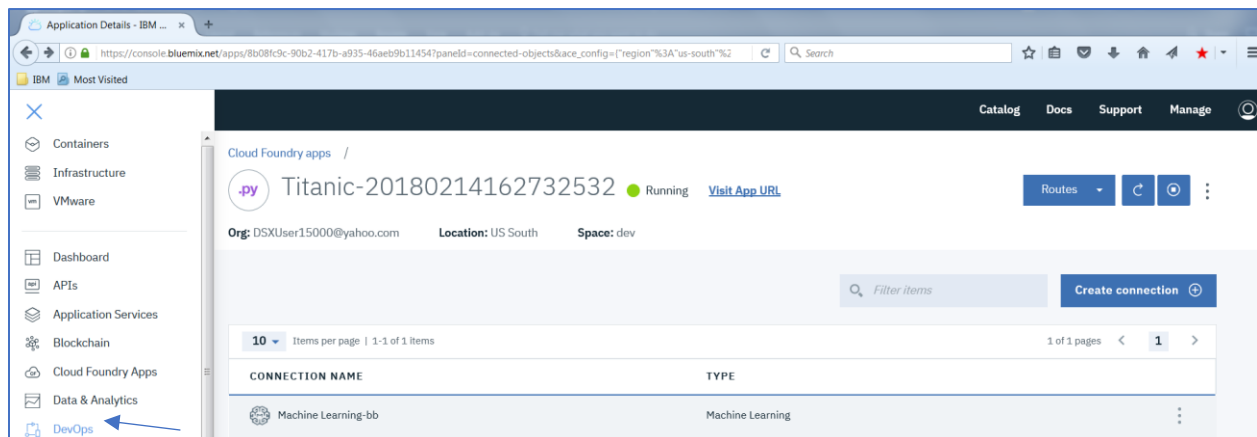
Cancel

Connect

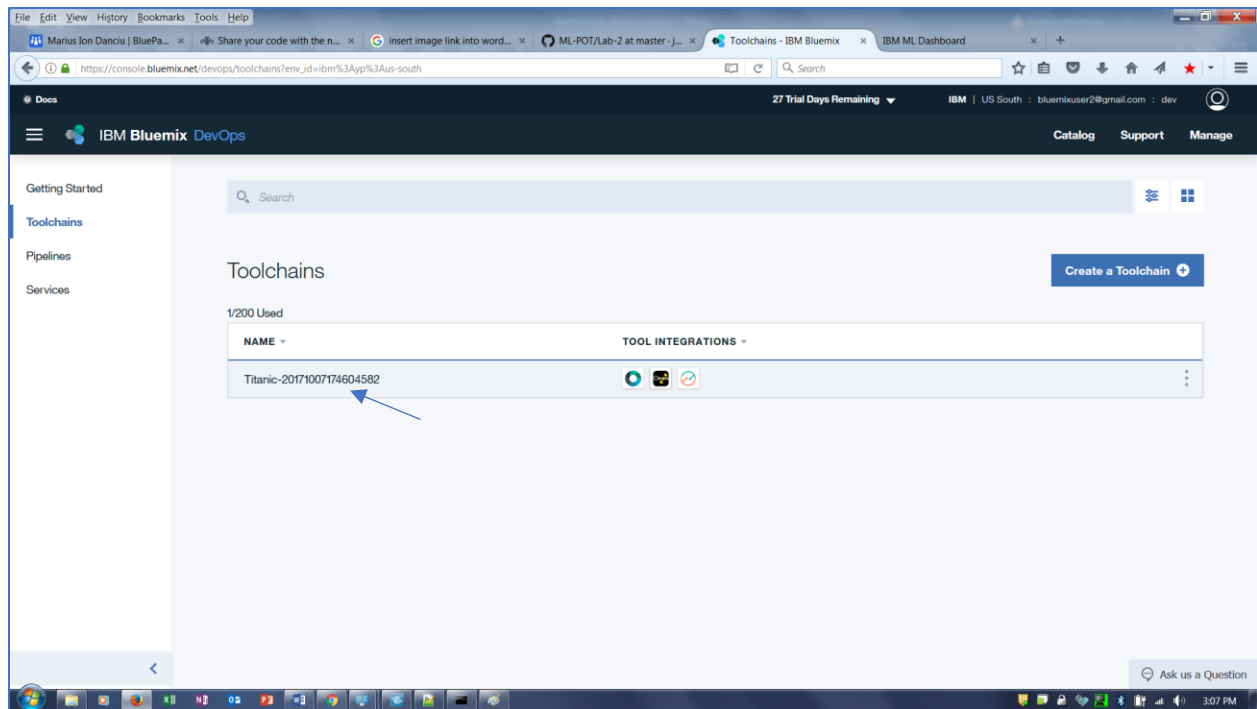
13. A **Restage app** pop up will appear. Click on **Restage**.



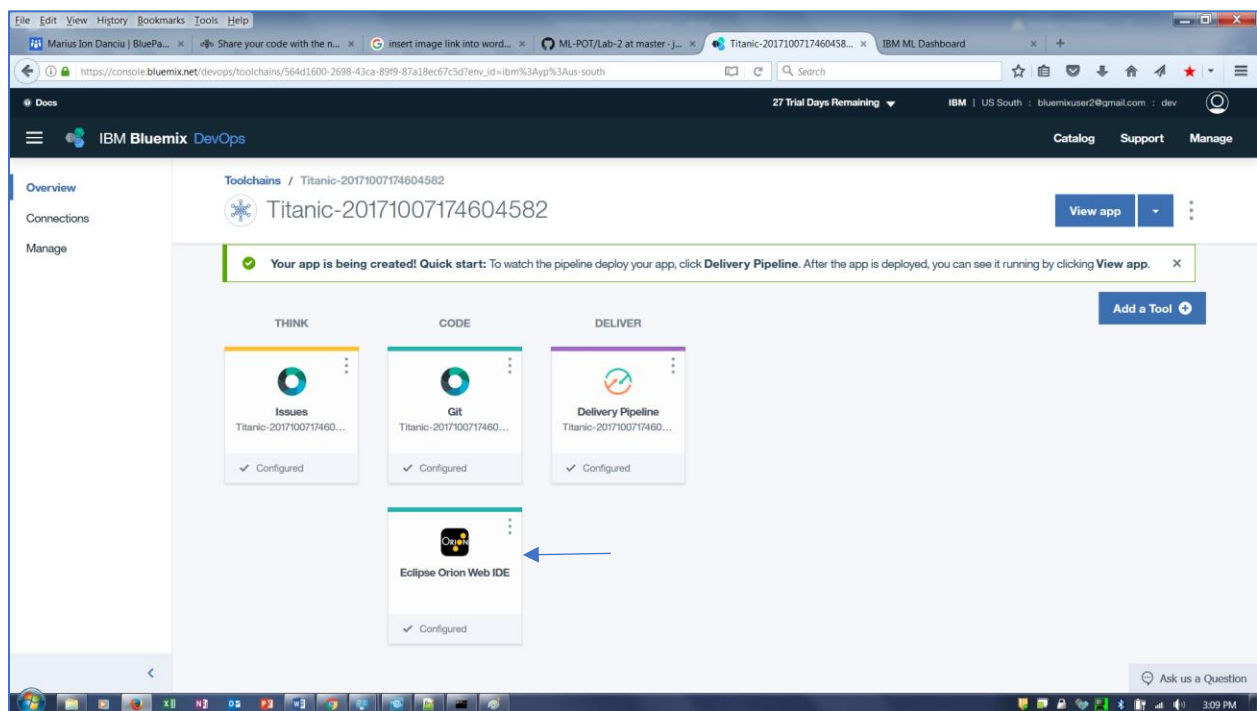
14. Wait for the application status to change to  **This app is awake**, or something similar.
15. We now have tied the web application to the Watson Machine Learning service. Note that the Watson Machine Learning service could have more than one deployed model available to select and then embed in the web application. In our case, we have only one deployed model. We now need to copy the scoring endpoint of that deployed model (previously copy and pasted into Notepad) and paste it in the web application code. Click on the  icon and click on DevOps in the pulldown to navigate to the Toolchain.



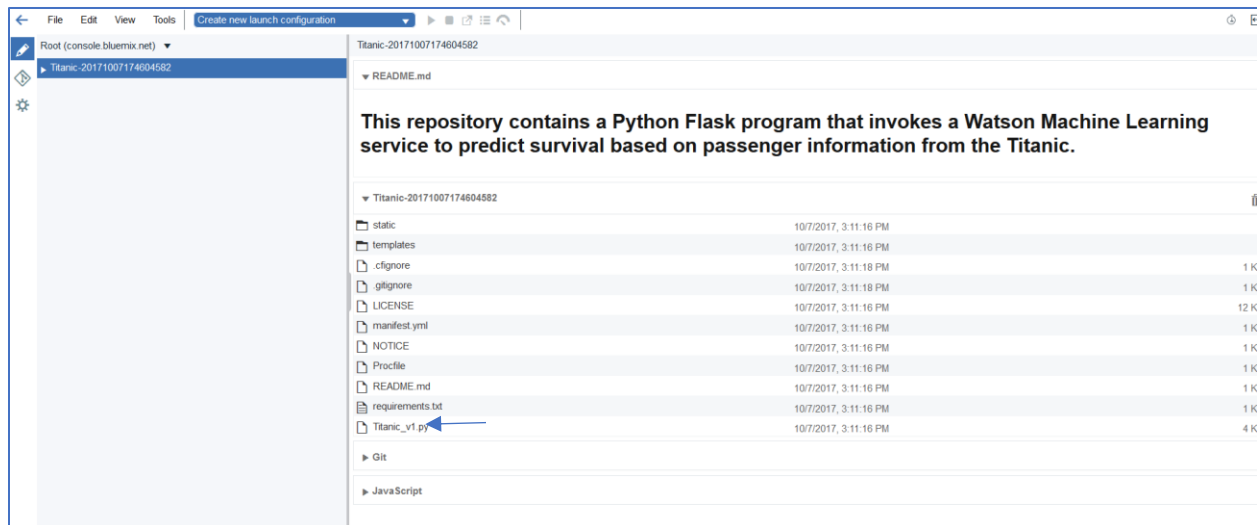
16. We are now going to paste the scoring endpoint into the application code. Click on the Toolchain (Titanic-2018xxxxx below).




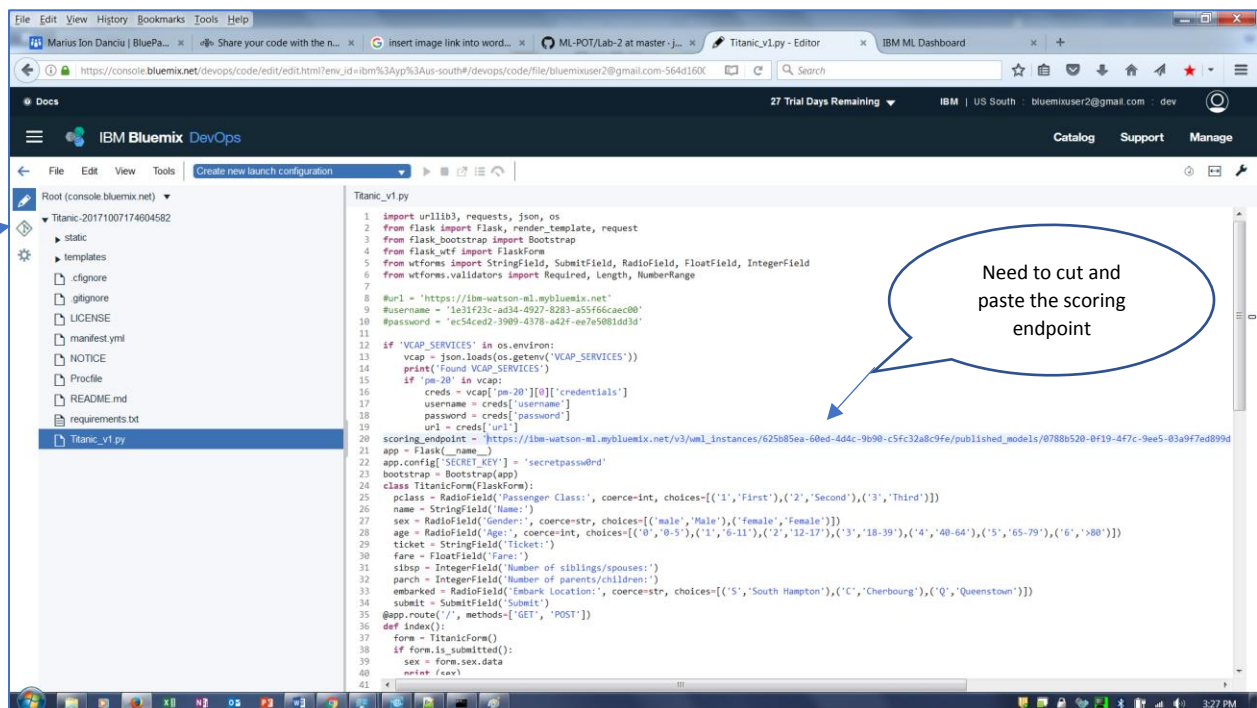
17. Click on the Eclipse Orion Web IDE.



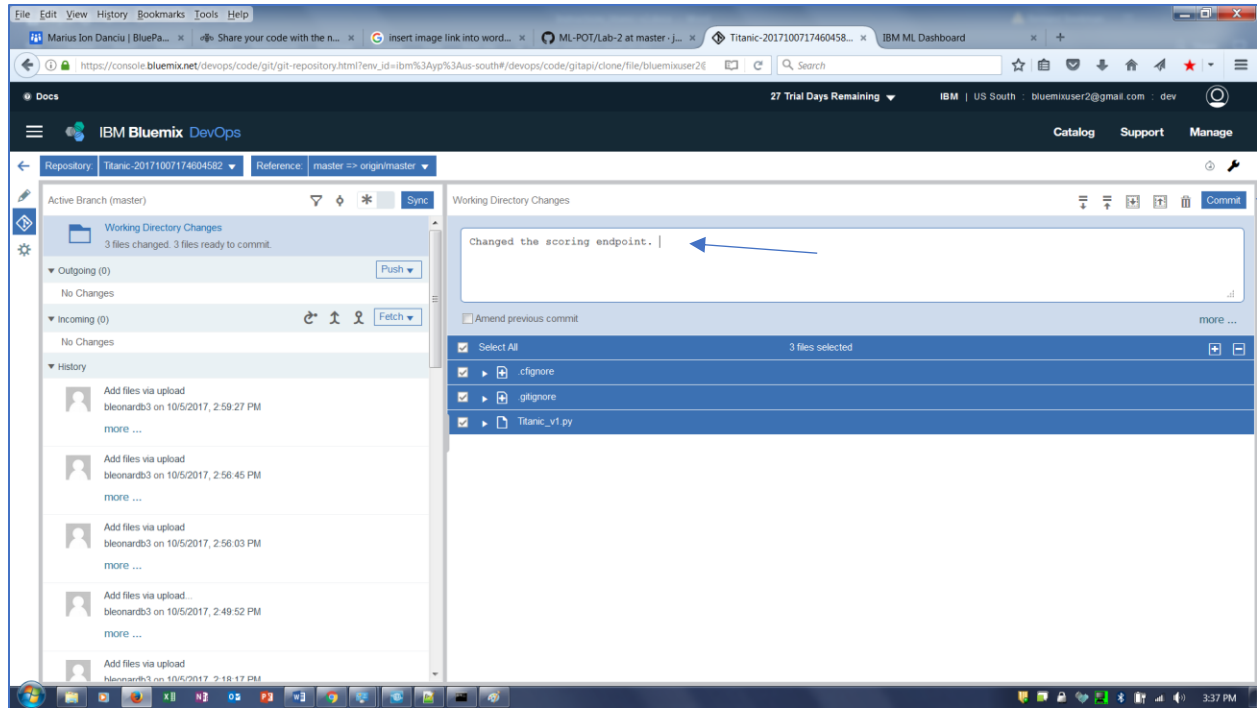
18. Click on the Titanic_v1.py file. This is a python source file.



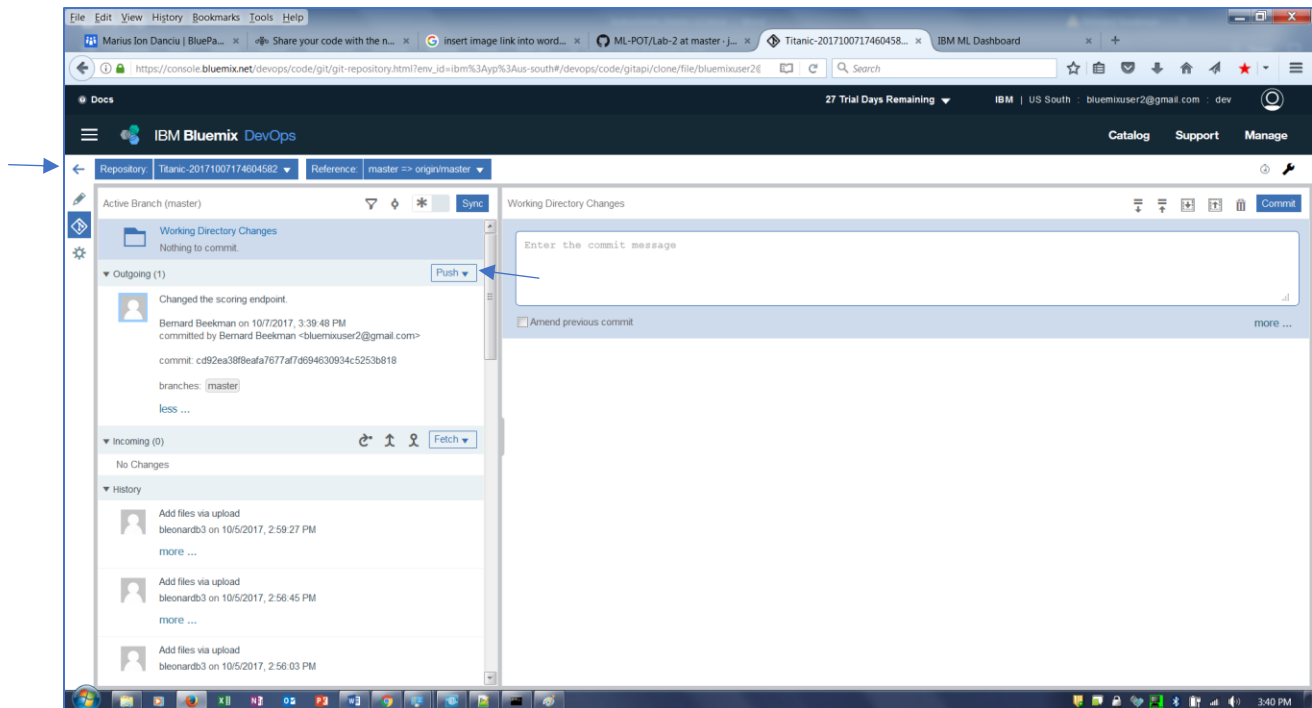
19. Go back to the Notepad file, and copy the scoring endpoint to the clipboard. Look around line 20 in the Titanic_v1.py file for the “scoring endpoint =”. Select the scoring endpoint value in line 20 (starting with https:// may want to use Shift-End to get to the end of the line, and then back up one space to not select the endpoint quote – if you do just make sure to put it back in). Enter Ctrl-V to paste the new scoring endpoint from your Notepad file. Enter Ctrl-S or File > Save to save the file. Then click on the  icon on the top left.




20. The next step is to commit the change to the git repository. Enter “Changed the Scoring Endpoint” in the Enter Commit Message field, and then click on **Commit**.




21. Then click on **Push** to push the changes to the central Git repo which will start the build and deploy of the application. Click on the left arrow to return to the Toolchain.



22. Click on the **Delivery Pipeline** to view status of the deployment as before. Once the Deployment status shows **Deploy passed now** (note it shouldn't take long so reload the

browser in case the UI didn't update). You can see the running app by clicking  icon in the left-hand corner and Dashboard in the pulldown to navigate to the **Dashboard**

where the running application should be listed. Click on the vertical ellipse  on the right-side of the Titanic application listing and then click on **Open URL** to view the web application (See steps 7 and 8 above).

23. The web form should appear. Enter data in all the fields and click on the **Submit** button. (the submit button is located at the bottom of the web form – you may need to scroll).

To determine the survival prediction, please enter the following:

Passenger Class:

☒ First
☐ Second
☐ Third

Name:

Gender:

☒ Male
☐ Female

Number of siblings/spouses:

Number of parents/children:

Ticket:

Fare:

Embark Location:

☒ South Hampton
☐ Cherbourg
☐ Queenstown

Age:

☒ 0-5
☐ 6-11
☐ 12-17

24. You should see something similar to the following depending on the values of the input fields that you entered. Click on the **Try Again!**, if you want to experiment with different inputs.

Titanic Prediction

prediction: survived
probability: 0.827966430684

[Try Again!](#)