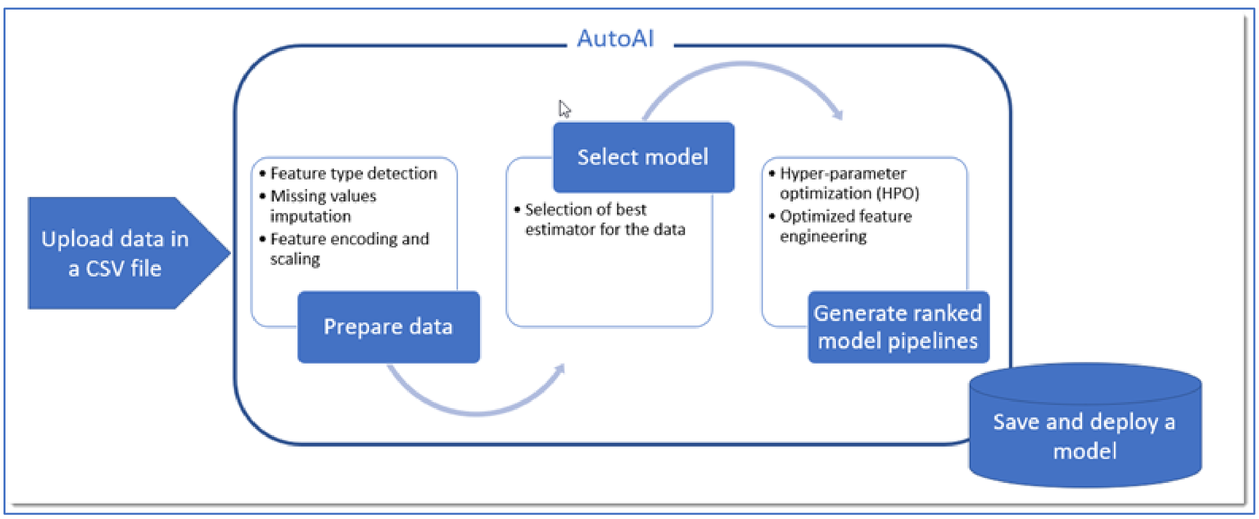
AutoAI + DevOps

Introduction

This lab consists of two parts. The first part will demonstrate the new and exciting AutoAI capability to build and deploy an optimized model based on the Female Human Trafficking datasets. The second part will deploy an application using the IBM Cloud DevOps toolchain that will invoke the deployed model to predict the trafficking risk.

AutoAI in Watson Studio automatically analyzes your data and generates candidate model pipelines customized for your predictive modeling problem. AutoAI algorithms analyze your dataset to discover data transformations, estimator algorithms, and parameter settings that work best for your problem setting. Results are displayed on a leaderboard, showing the automatically generated model pipelines ranked according to your problem optimization objective.

Using AutoAI, you can build and deploy a machine learning model with sophisticated training features and no coding. The tool does most of the work for you.



The AutoAI process follows this sequence to build candidate pipelines:

- [Data pre-processing]
- [Automated model selection]
- [Automated feature engineering]
- [Hyperparameter optimization]

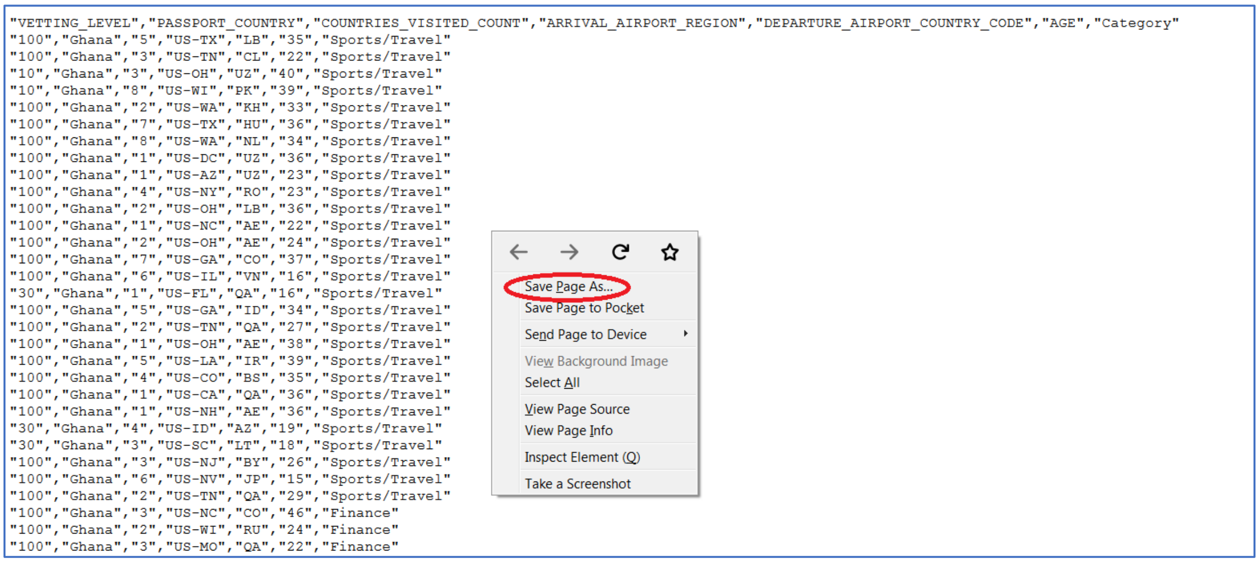
We will perform the following steps in this lab:

1. Download a female human trafficking cleansed dataset from the github repo
2. Add an Auto AI Experiment to create a model to predict the trafficking risk
3. Save and Deploy the model

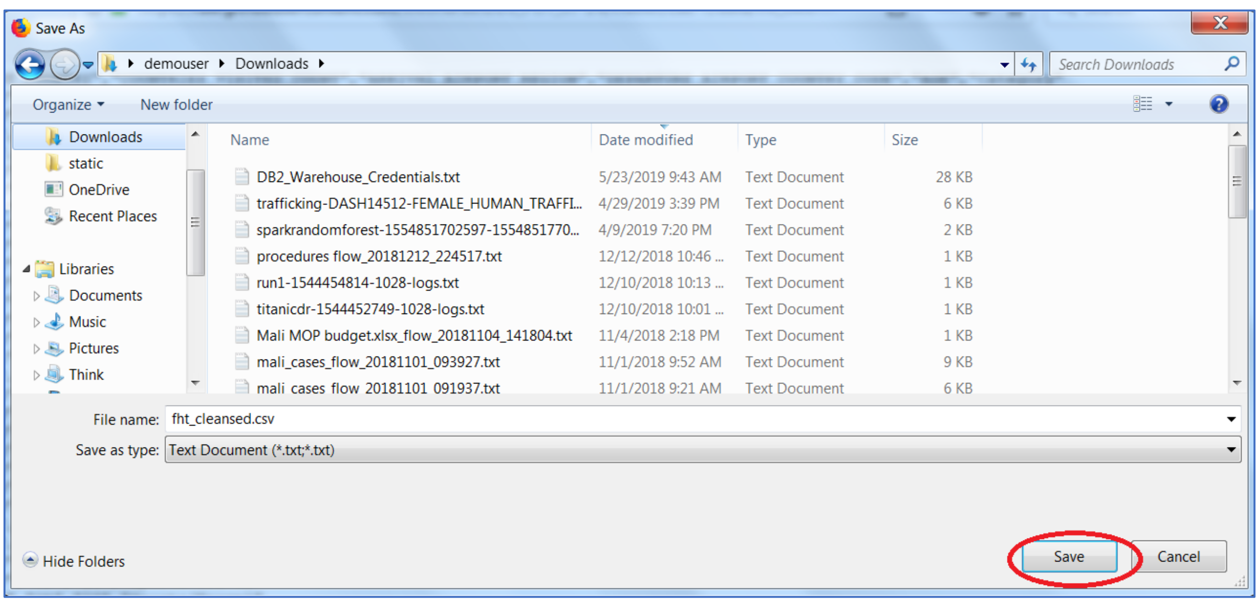
4. Test the model
5. Deploy a simple web front-end application and connecting it to the deployed model using an IBM Cloud DevOps toolchain.

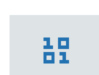
Step 1: Download a female human trafficking cleansed dataset

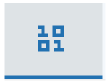
1. Download the fht_cleansed.csv data file from the following location by clicking [here]
2. Right-click on the window, and click **Save Page As…**

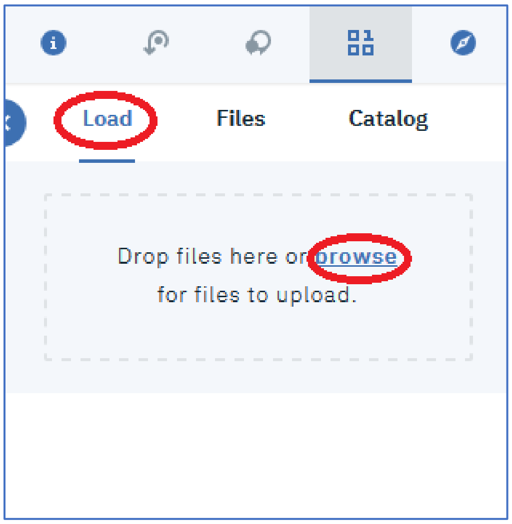


3. Click on **Save**.

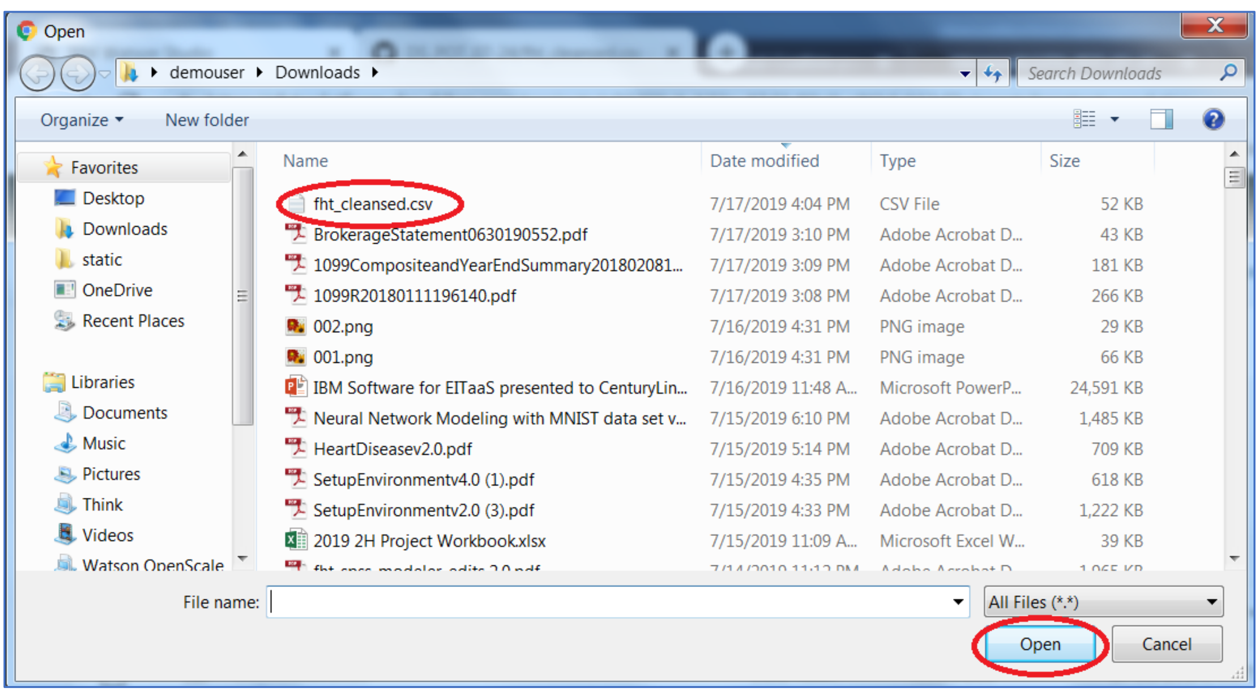


4. Go back to your Watson Studio Labs project. Click on the  icon.

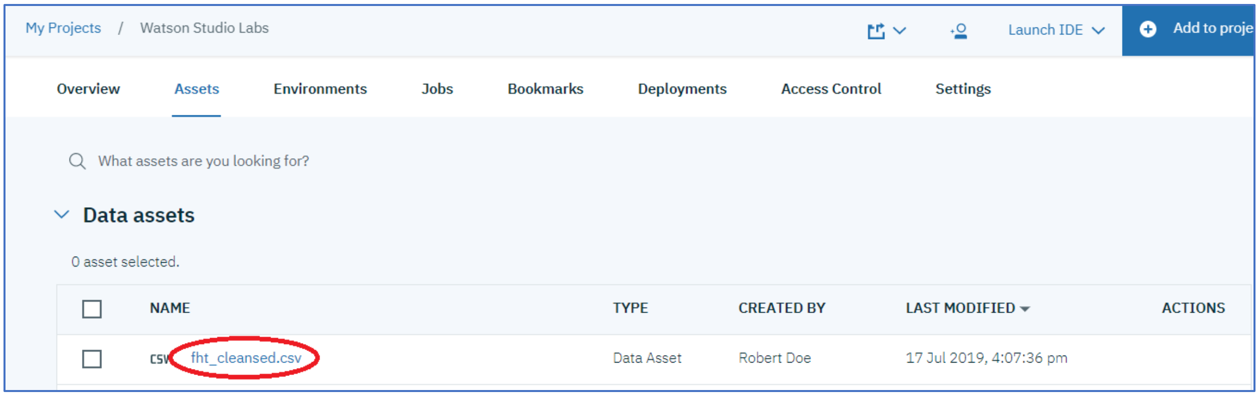
5. Click on the **Load** tab and then click on **browse**. If you don't see the **Load** tab, click on the  icon again.



6. Go to the folder where the fht_cleansed.csv file is stored. Select the fht_cleansed.csv file and then click **Open**.

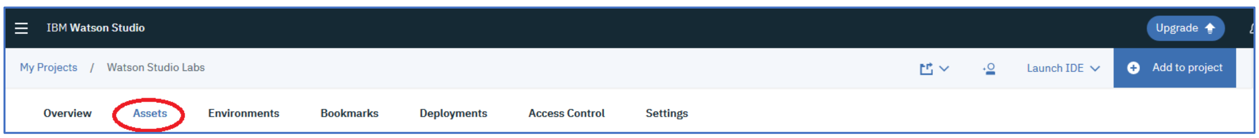


7. The fht_cleansed.csv file is now added as a Data Asset.

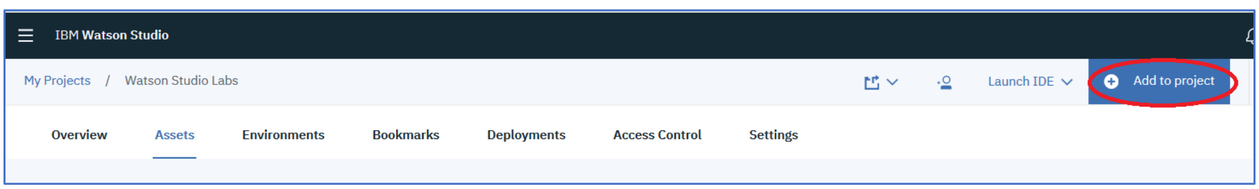


Step 2: Add an AutoAI Experiment.

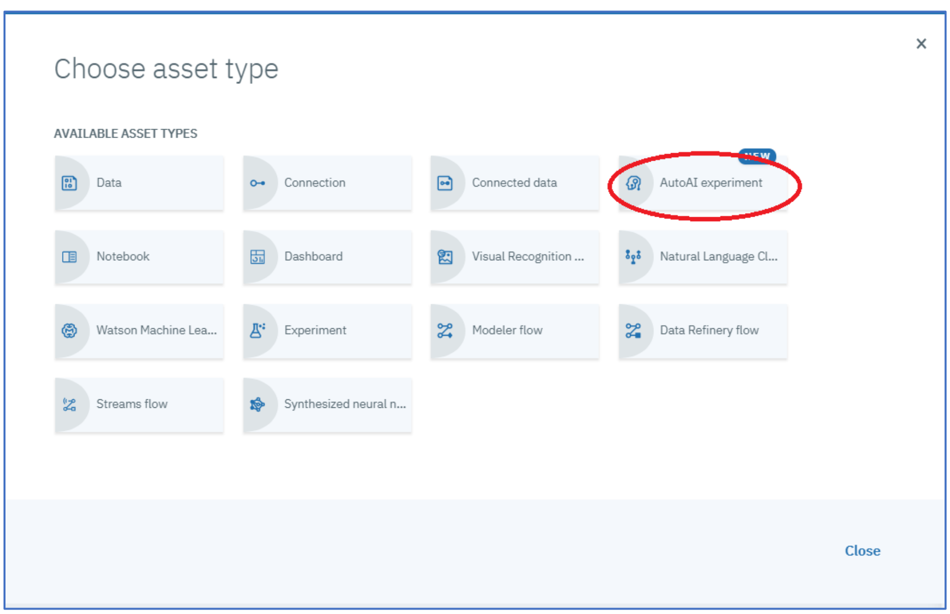
1. If not on the **Assets** page, click on the **Assets** Tab



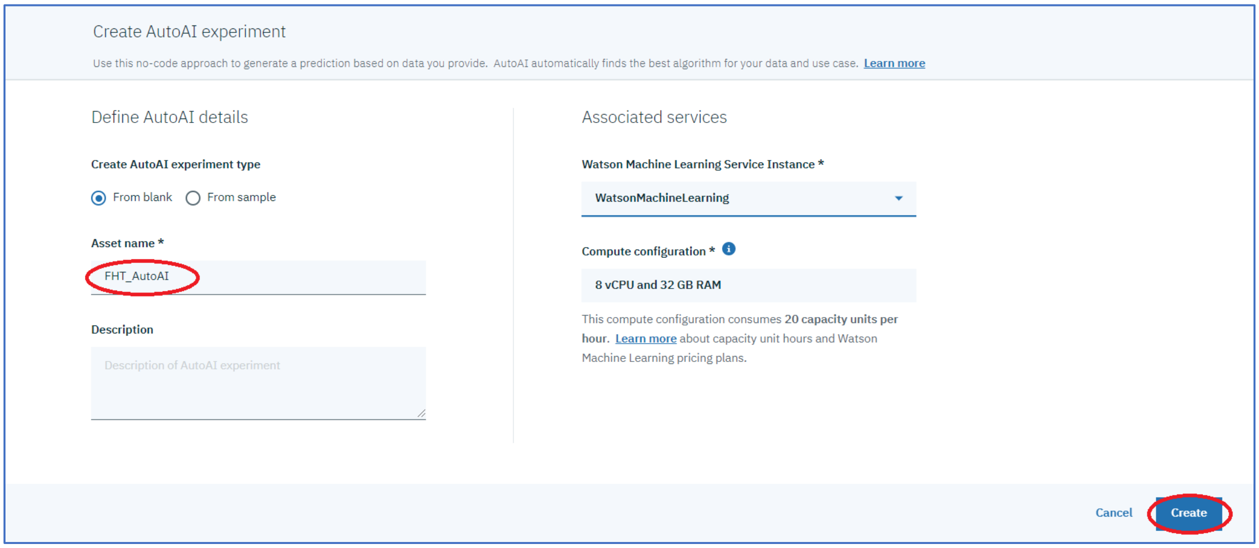
2. Click on **Add to project**.



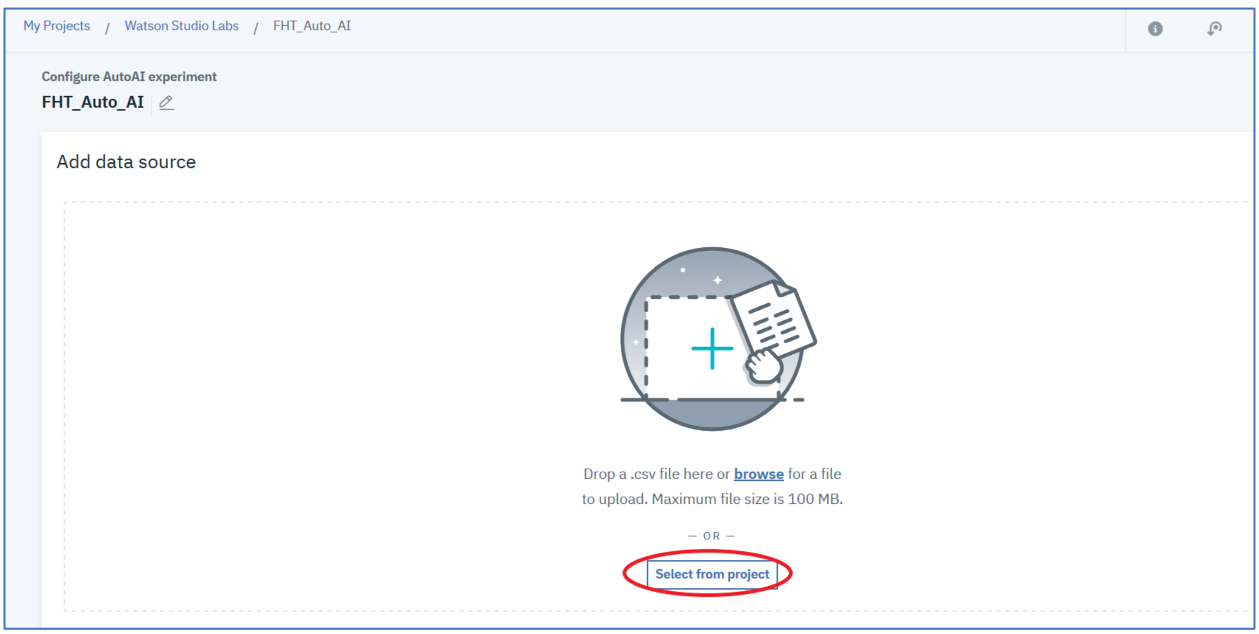
3. Click on **AutoAI Experiment.**



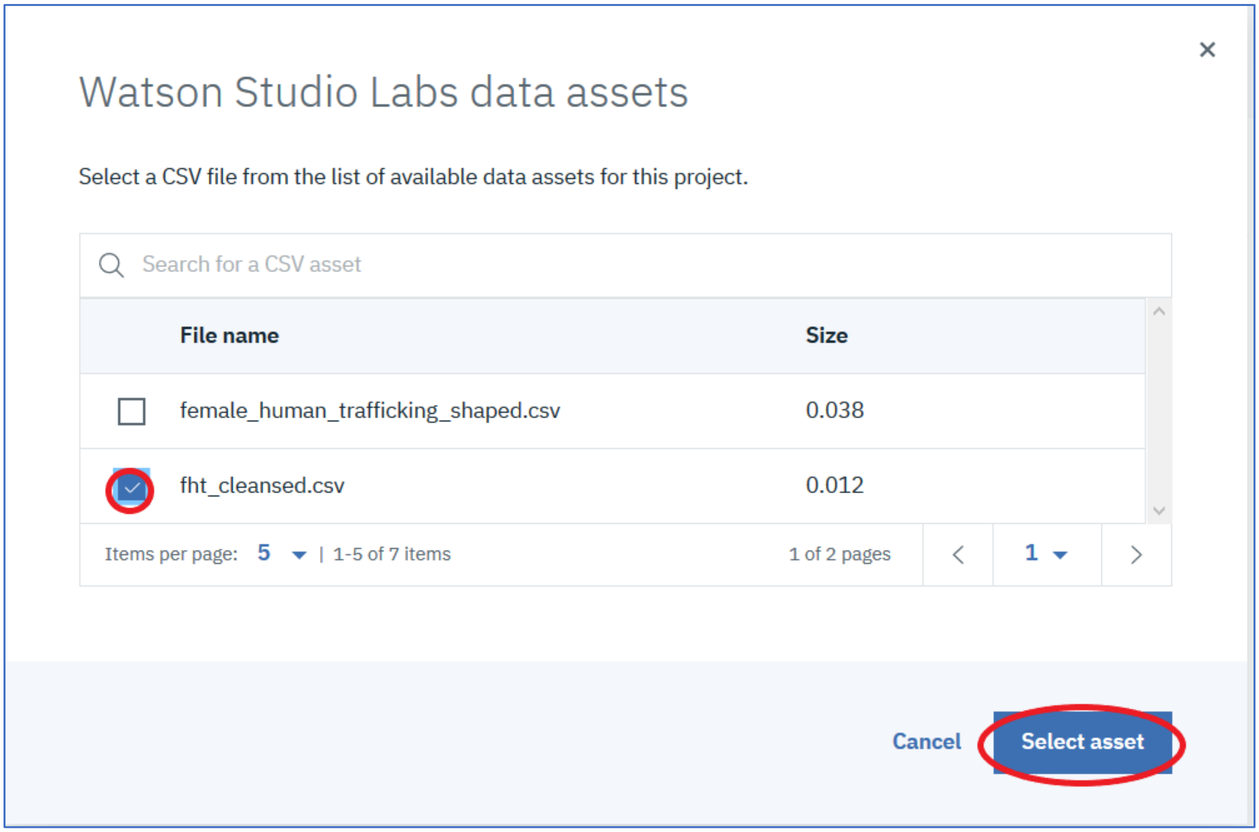
4. Enter an **Asset name**, leave the defaults for the **Watson Machine Learning** and **Compute configuration** and click on **Create**.



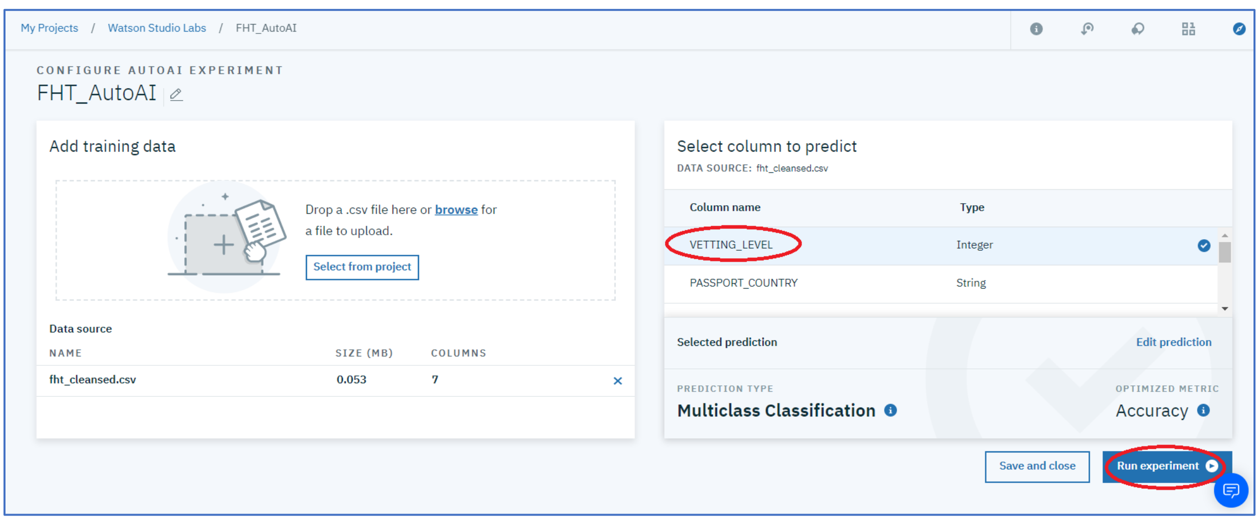
5. Click on **Select from project**.



6. Select the **fht_cleansed.csv** dataset and click on **Select asset**.

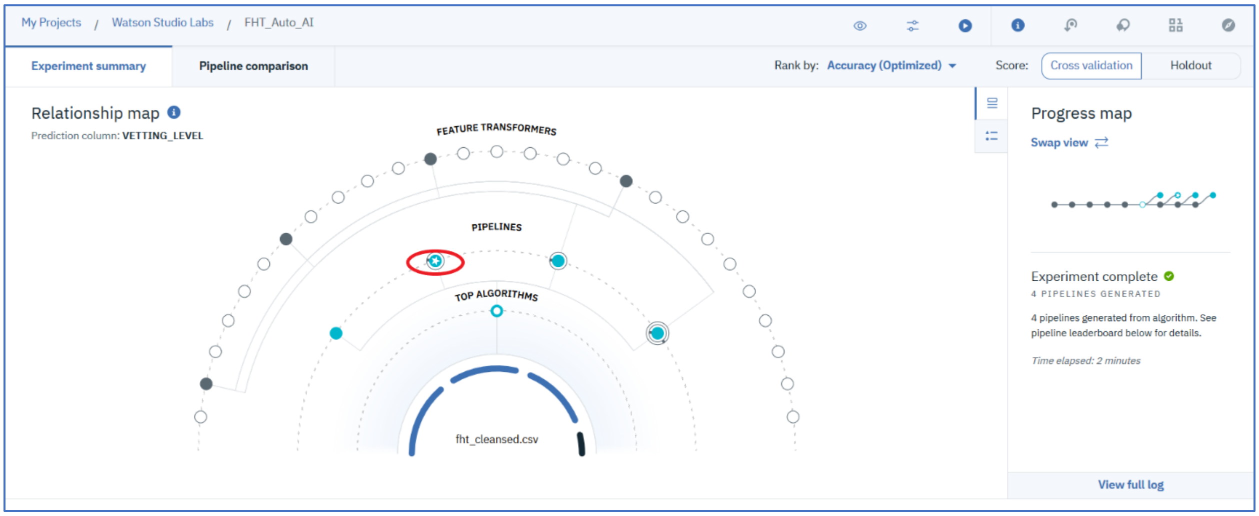


7. Select **VETTING_LEVEL** as the column to predict, leave the default for **OPTIMIZED METRIC** and click on **Run Experiment**. Note the system scanned the VETTING_LEVEL values to determine that a **Multiclass Classification** was the **PREDICTION TYPE**.

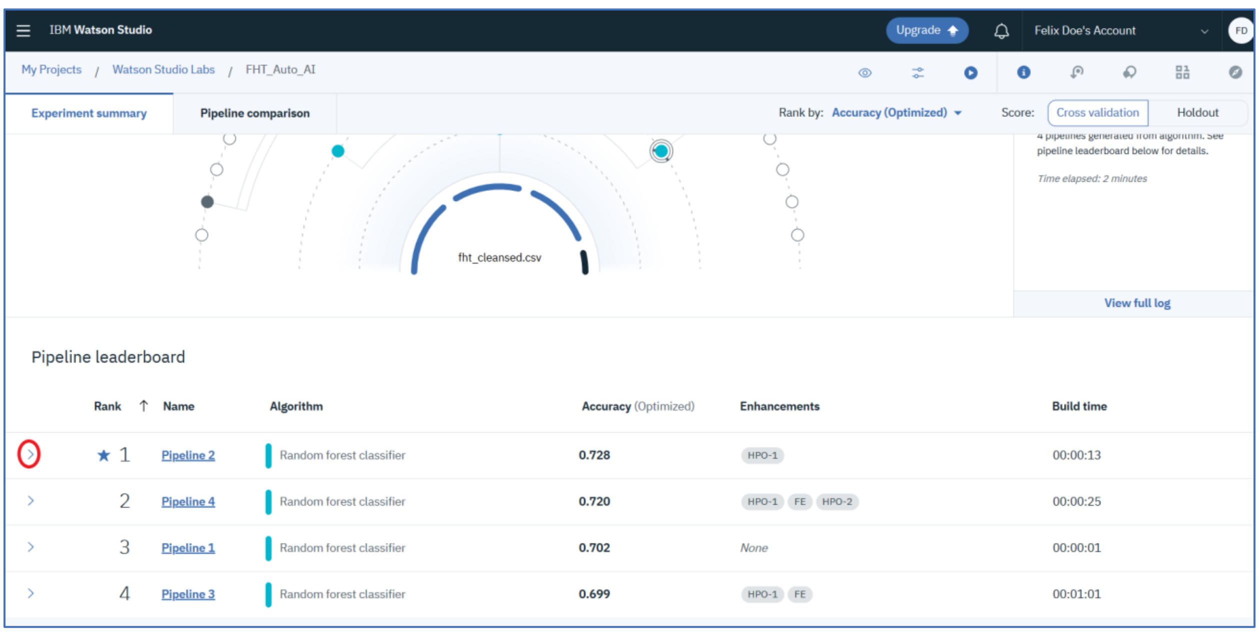


8. Each pipeline is displayed as the processing is completed for the pipeline. Four pipelines will be created. The first pipeline picks the best algorithm. In this case, it is Random Forest. The second pipeline performs a hyperparameter optimization to see if tuning the algorithm parameters will improve the performance metric. The third pipeline will derive

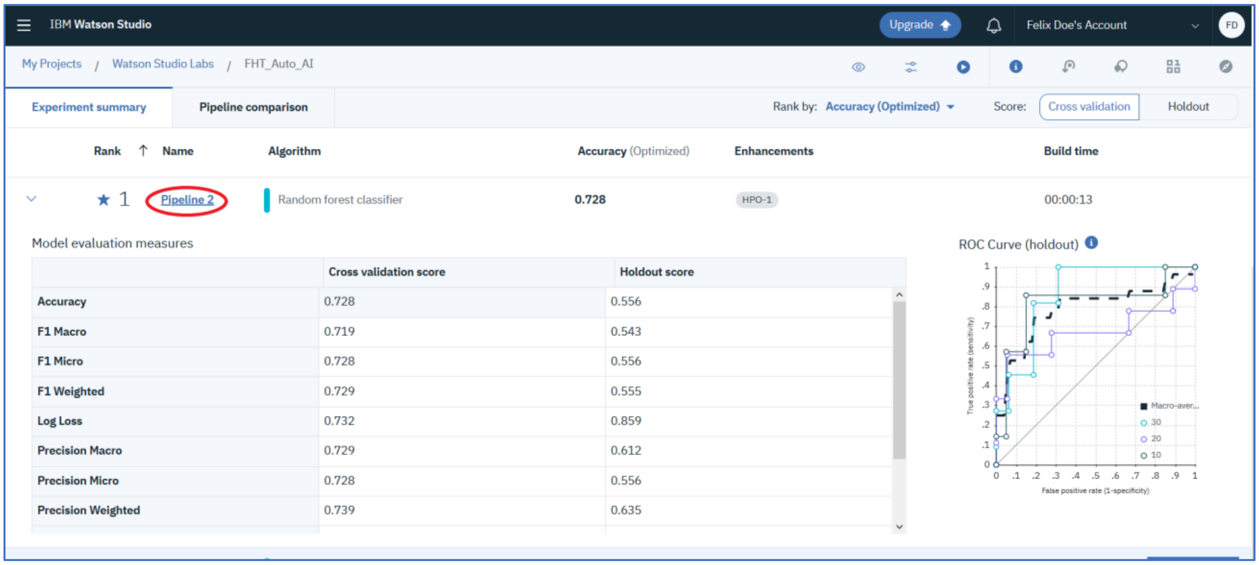
new features (i.e. feature engineering) to improve the performance metric. Finally, the fourth pipeline will do another hyperparameter optimization including the newly derived features. The results are shown visually below. The second pipeline performs the best based on the cross-validation accuracy.



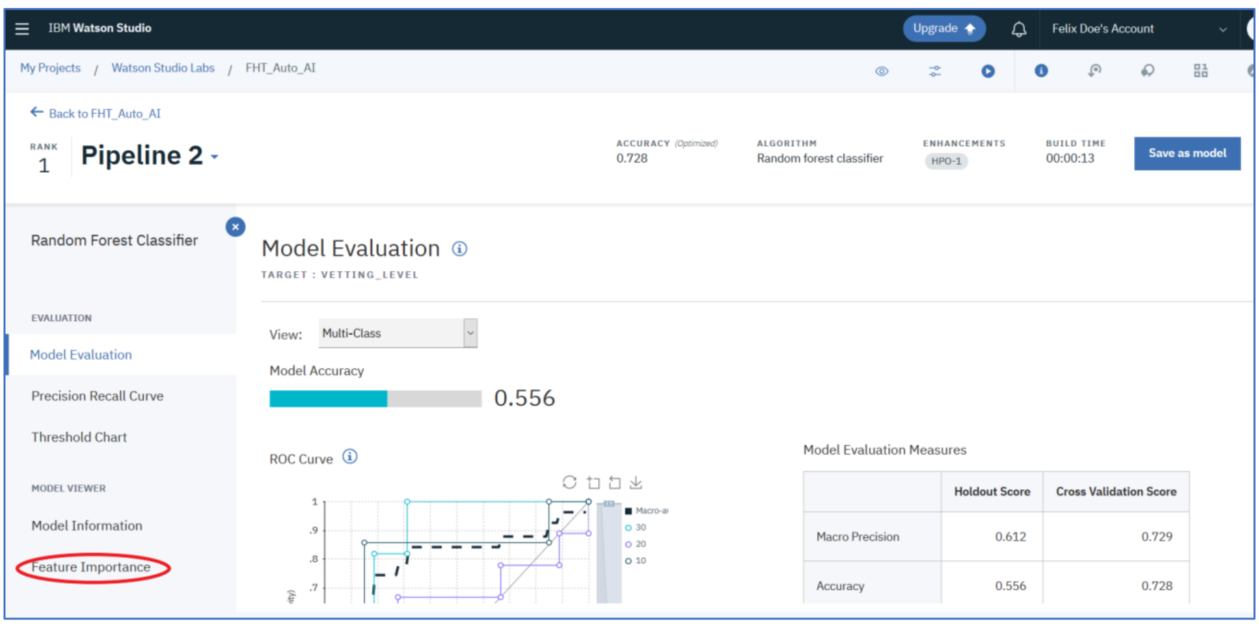
9. Scroll down to view the Pipeline leaderboard. Click on the right caret next to Pipeline 2.



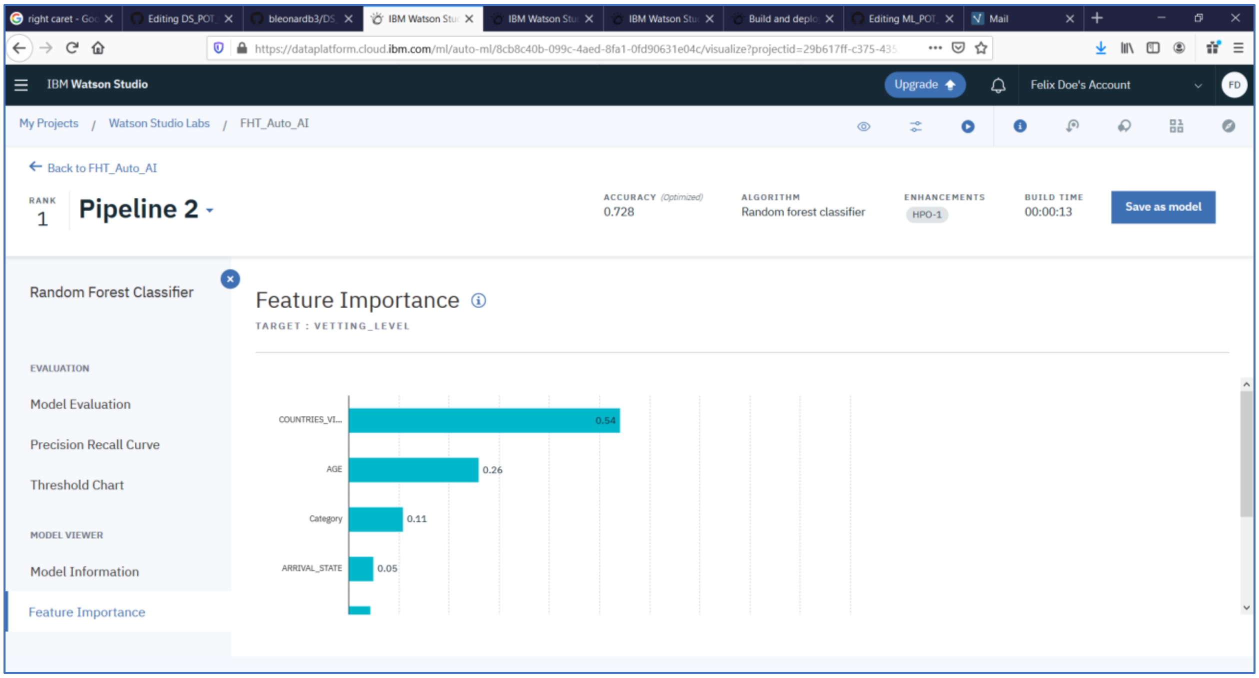
10. Scores are displayed for different metrics for both the training sample and the holdout sample. The holdout sample is 10%. Click on **Pipeline 2.**



11. The model evaluation metric for the holdout sample is displayed. On the left are options for additional information. Click on **Feature Importance**.

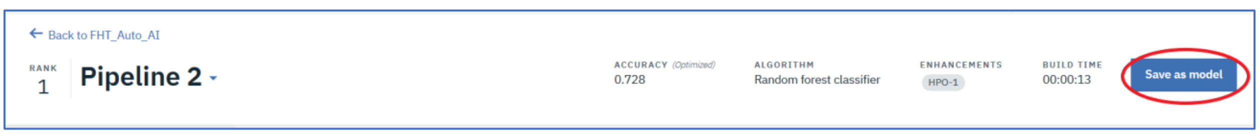


12. According to the **Feature Importance**, the **COUNTRIES_VISITED_COUNT** feature is the most important, followed by the **Age** feature and the **Category** feature.

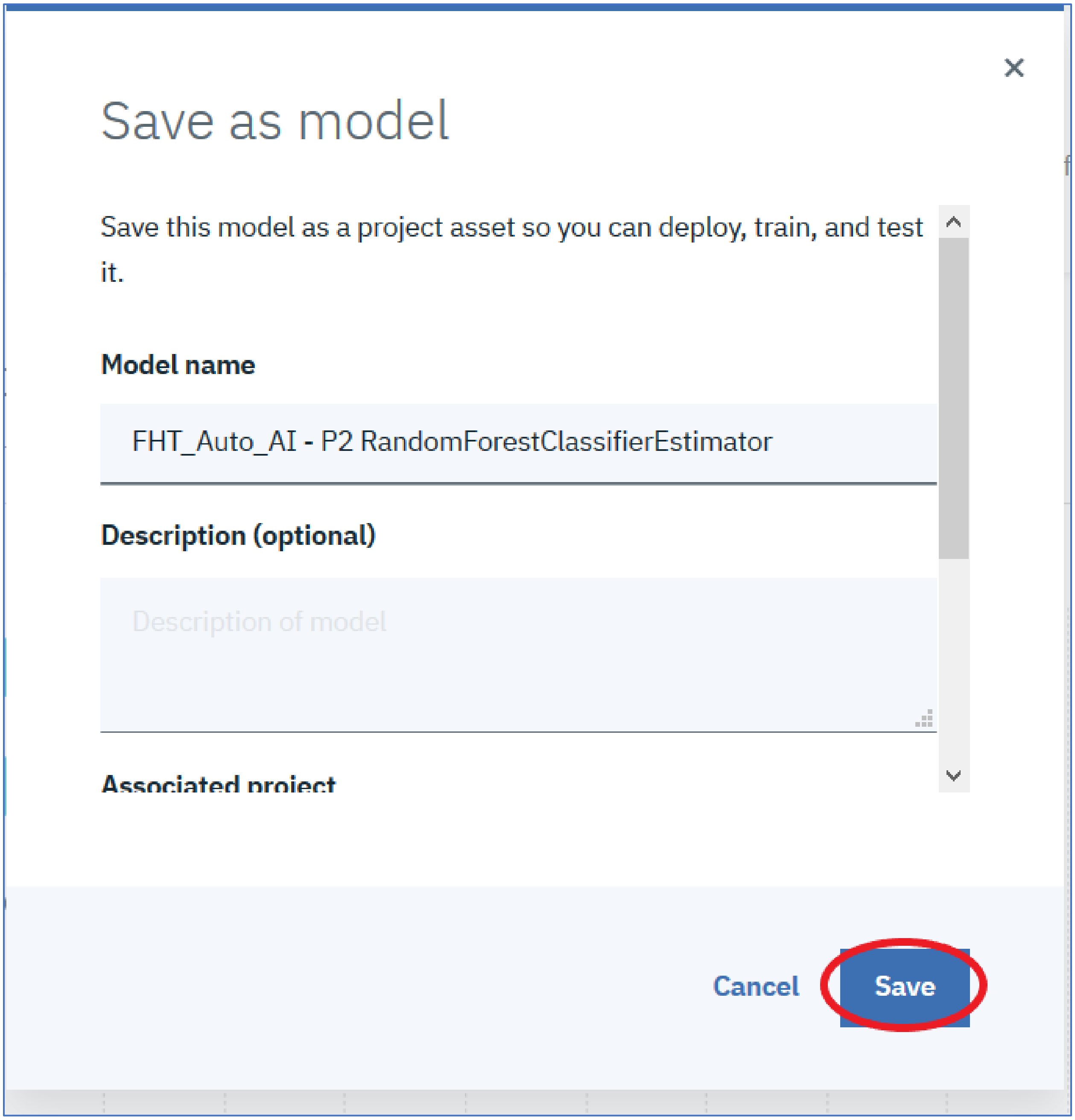


Step 3: Save and Deploy the Model

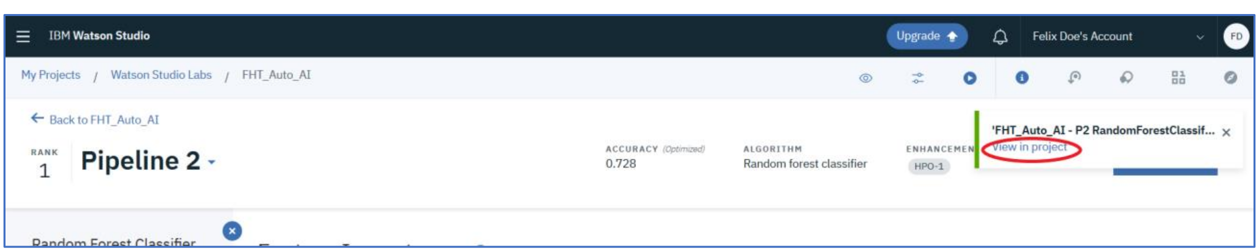
1. Click on **Save as model**



2. Optionally change the default name and click **Save**.



3. A message is displayed showing the model was successfully saved. Click on **View in project.**

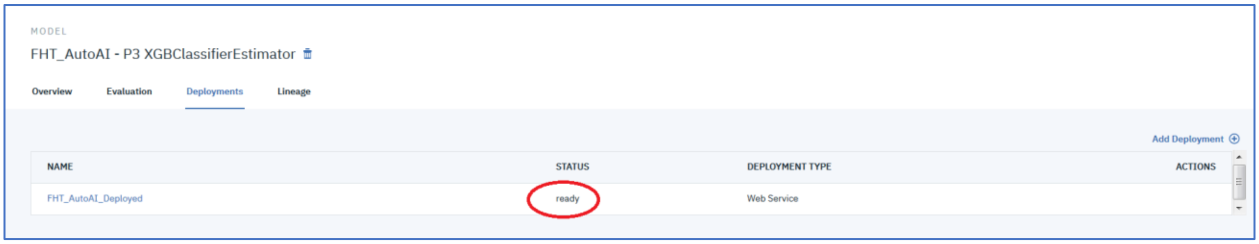


4. Click on **Deployments**.

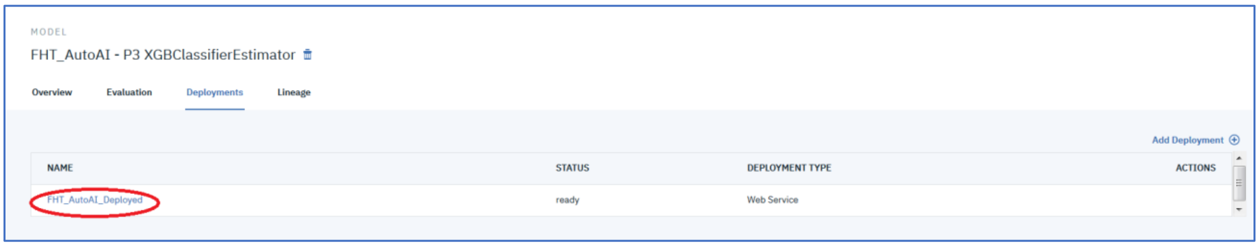
5. Click on **Add Deployment**

6. Enter a **Name**, optionally a **Description** and click **Save**.

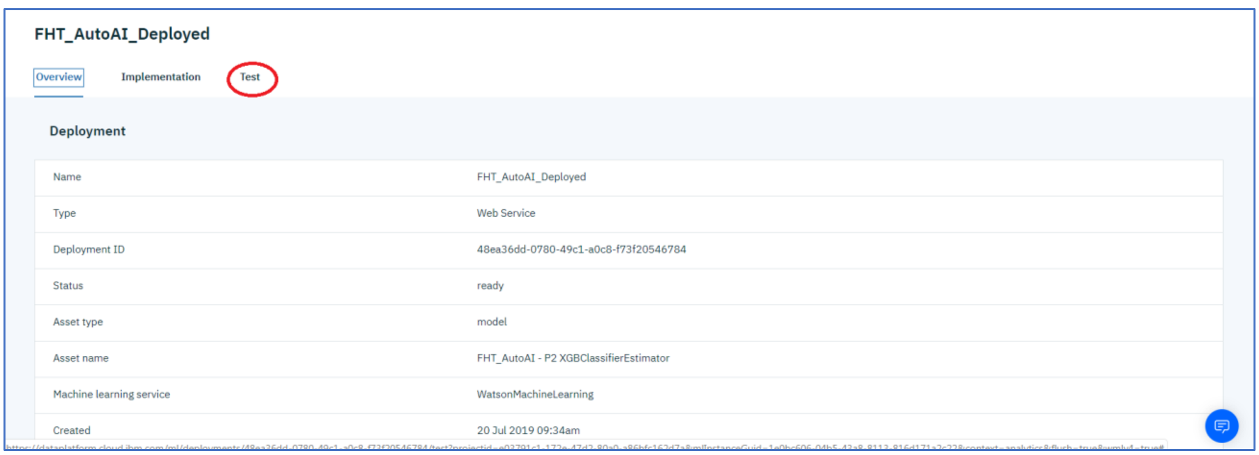
7. The model is successfully deployed in the IBM Cloud.



8. Click on **FHT_AutoAI_Deployed**.



9. Click on **Test.**



10. Enter the following values:

PASSPORT_COUNTRY **– Ghana**

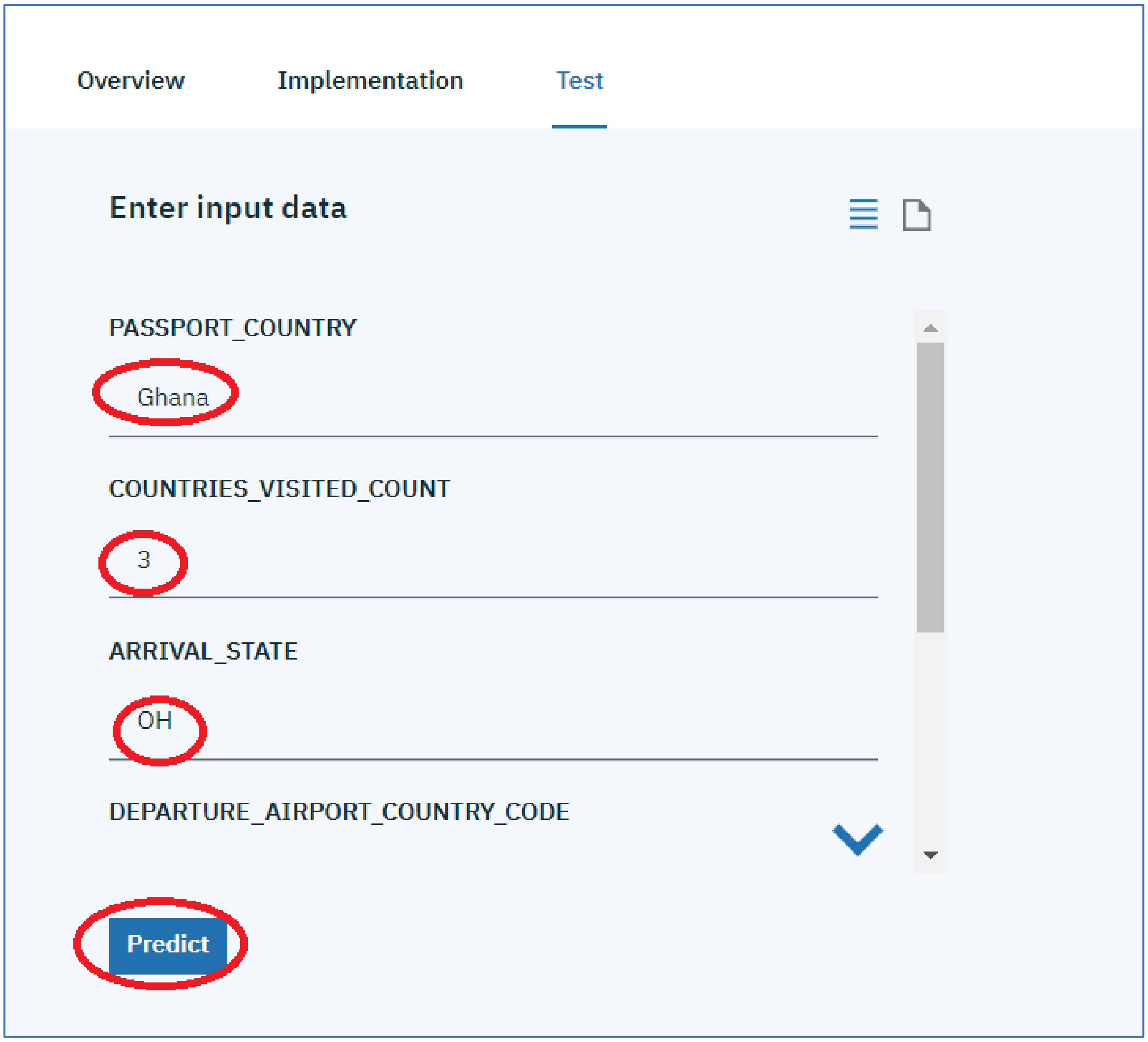
COUNTRIES_VISITED_COUNT – **3**

ARRIVAL_STATE – **OH**

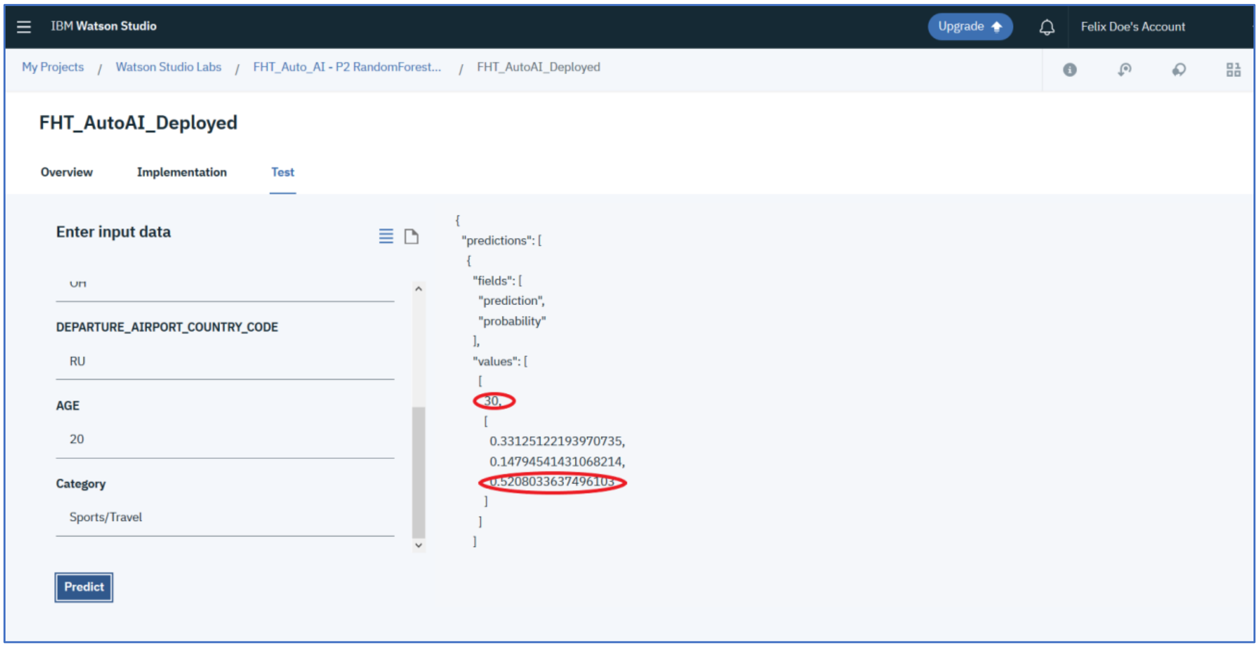
DEPARTURE_AIRPORT_COUNTRY_CODE – **RU**

AGE – **20**

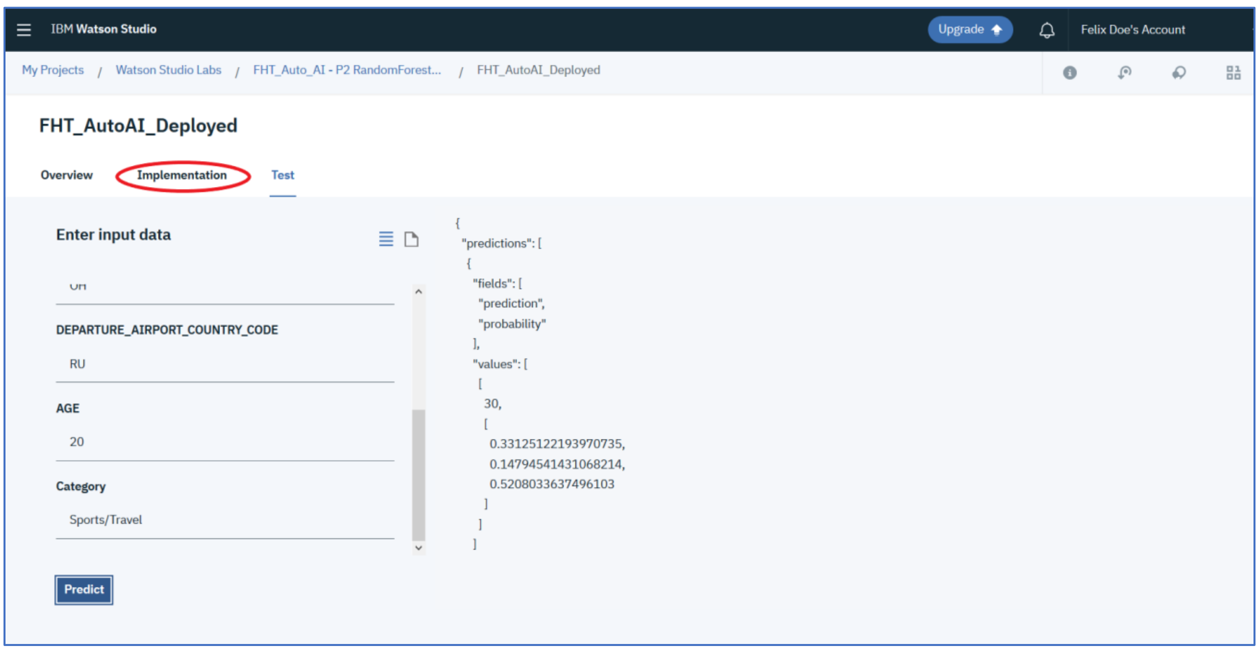
Category – Sports/Travel



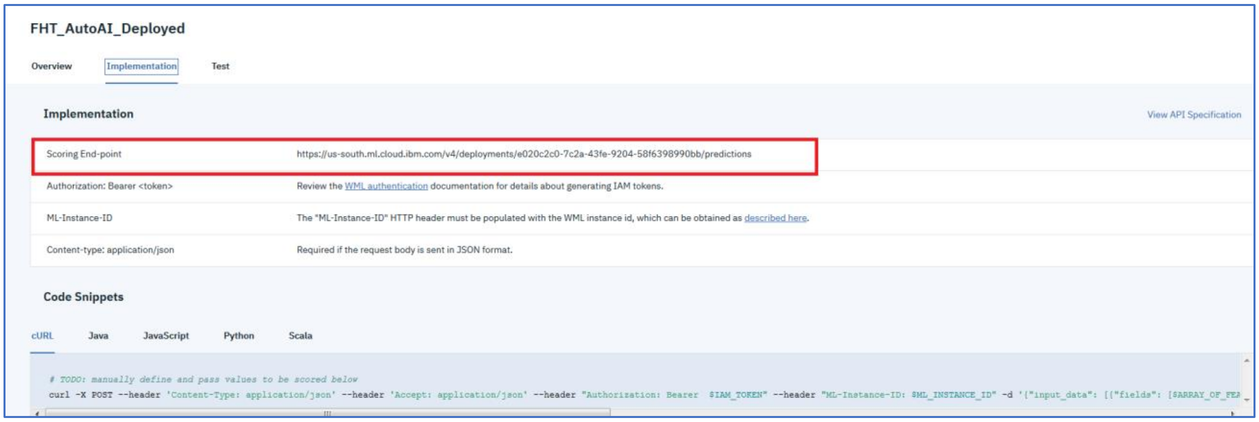
11. The prediction is **30** (low risk) of trafficking with 91% confidence.



12. Click on Implementation



13. 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, or just leave this panel available to cut and paste the scoring endpoint. We will need the scoring endpoint in the next section.



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

This section provides an example of a simple Python Flask web front-end application that invokes the Female Human Trafficking risk prediction deployed model, 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.

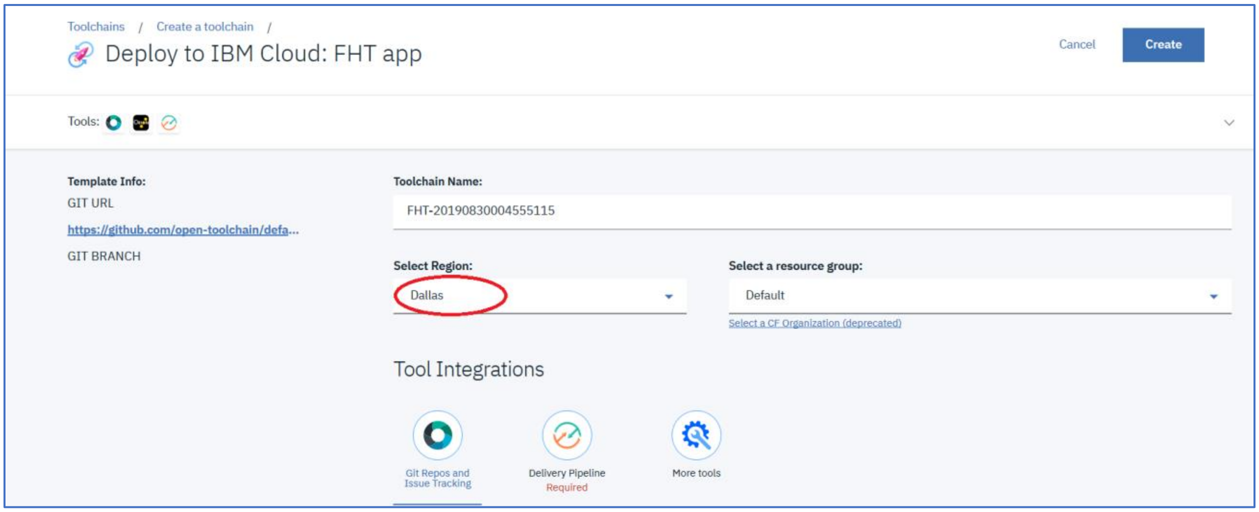
The toolchain uses tools that are part of the Continuous Delivery service. If an instance of that service isn't already in your account, when you click **Deploy**, it is automatically added with the free Lite plan selected.

The steps below guide you in configuring the application to connect to your Watson Machine Learning service, and to update the application with the deployed model's scoring endpoint.

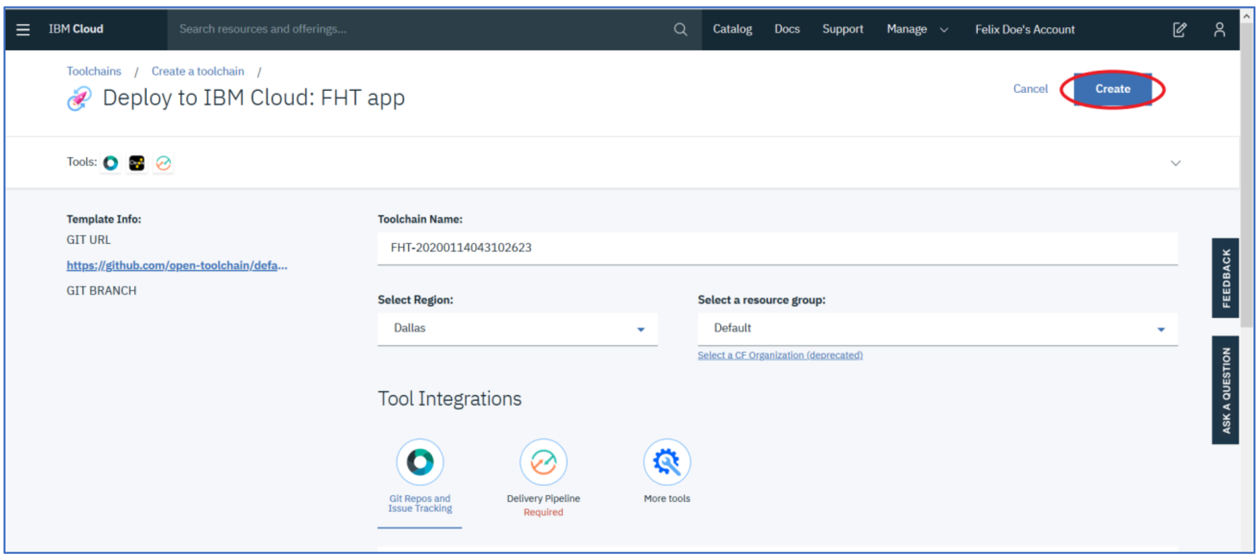
1. Click on the **Deploy to IBM Cloud** link below to deploy a sample Python Flash web application into your IBM Cloud account. Note you may get this message – "*An IBM Cloud account is required. To get started, click Log In or Sign Up at the top of this page*". If you get this message, click on **Log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

   Deploy to IBM Cloud

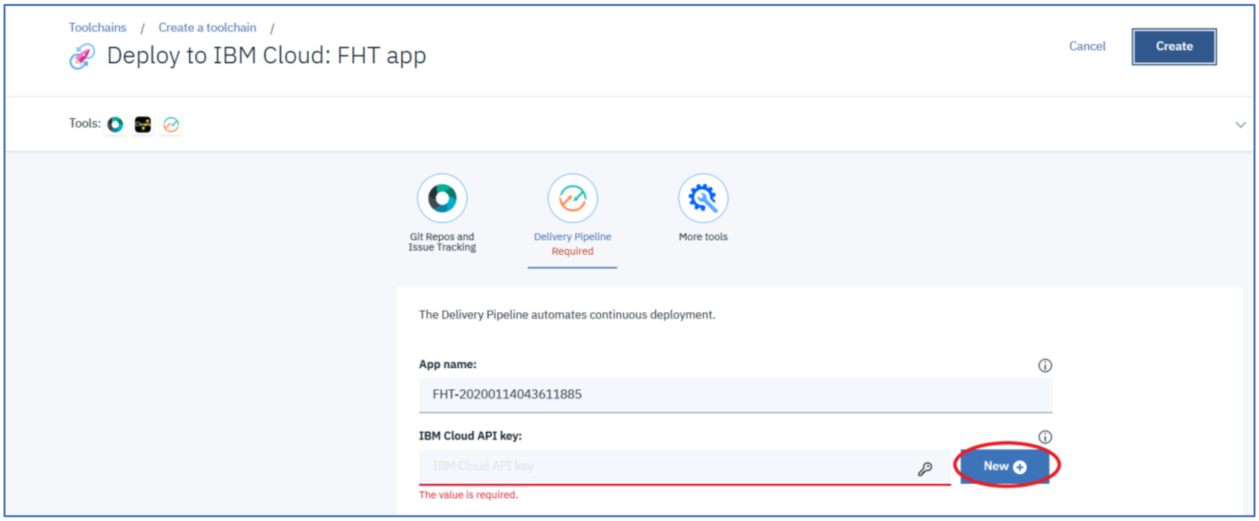
2. In the **Select Region**, make sure that the region is Dallas. If not, change it to Dallas, and wait for the screen to refresh.



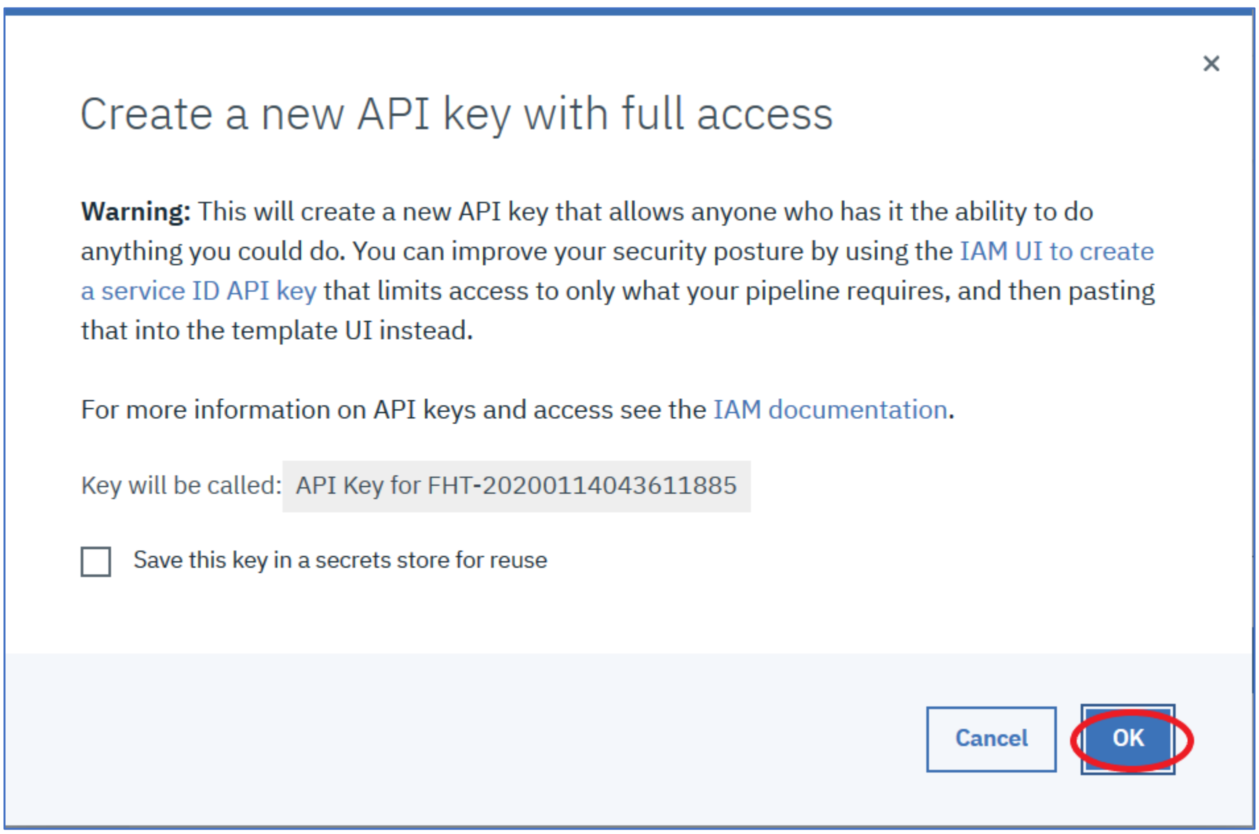
3. Click **Create**.



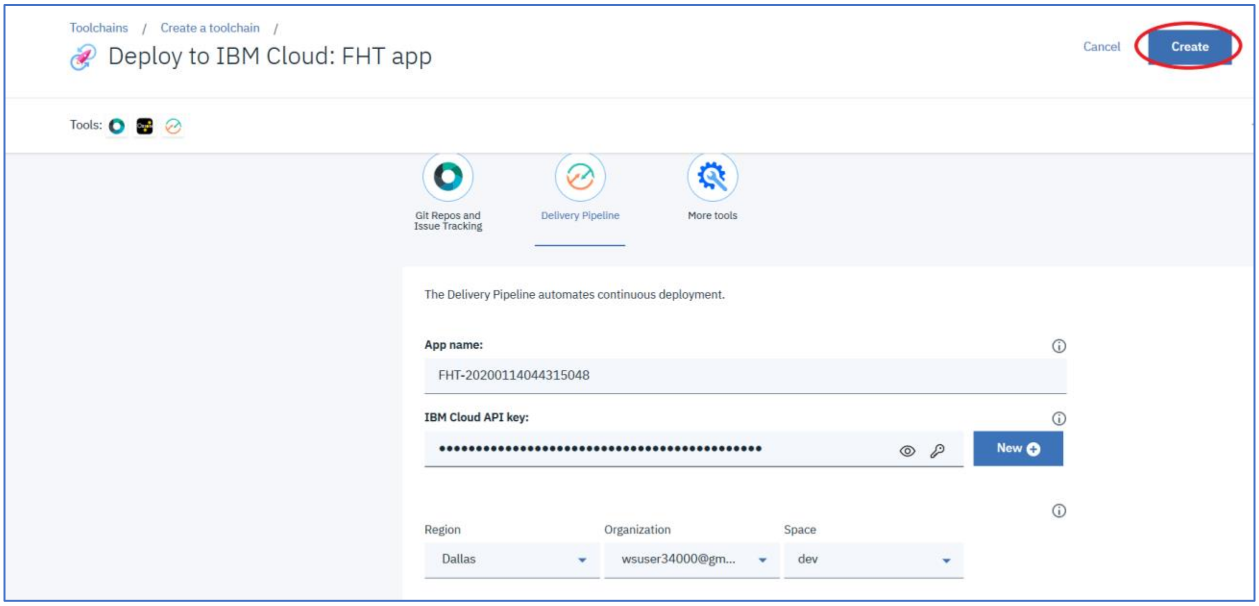
4. Scroll down and click **New+** to create an API key.



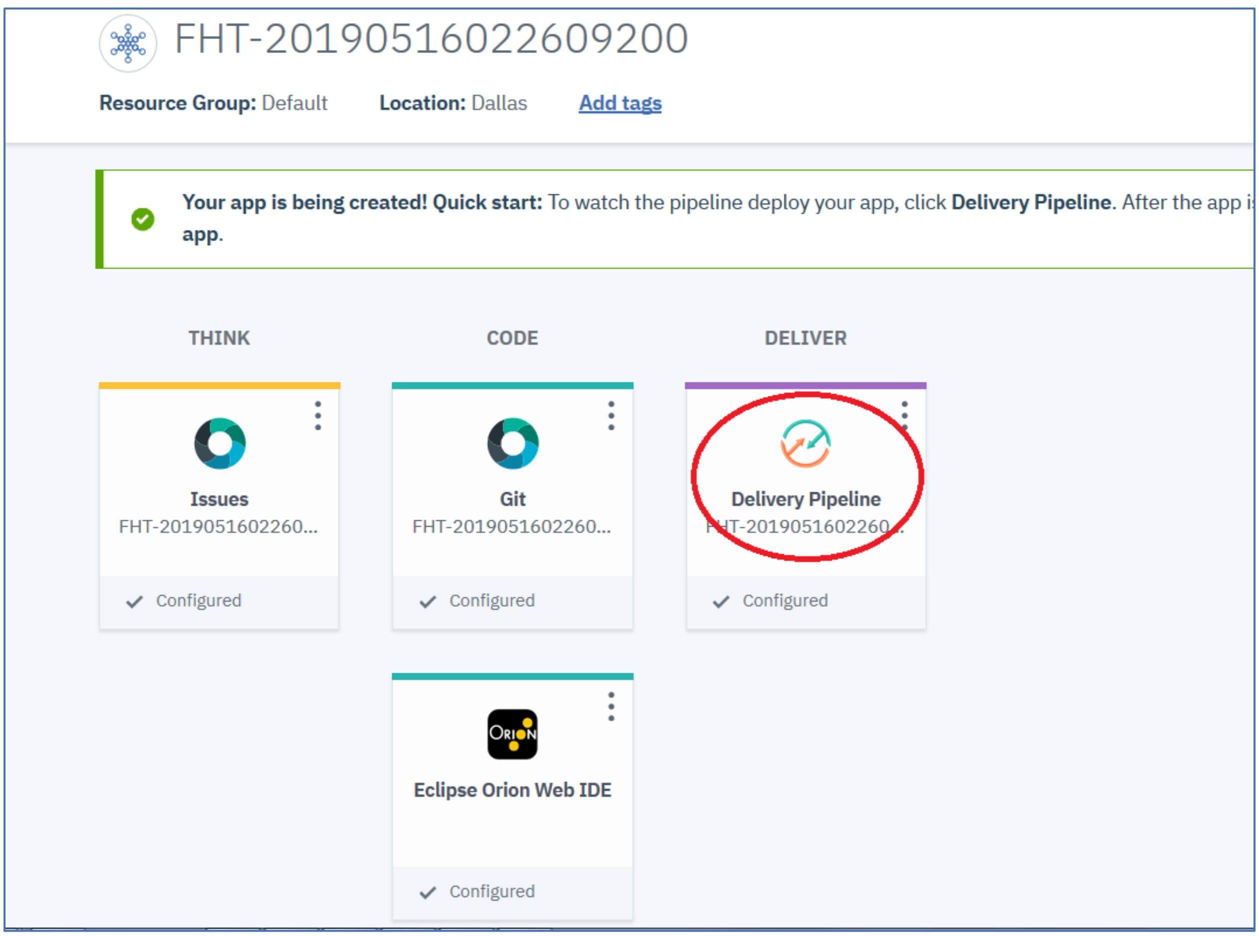
5. Click **OK**.



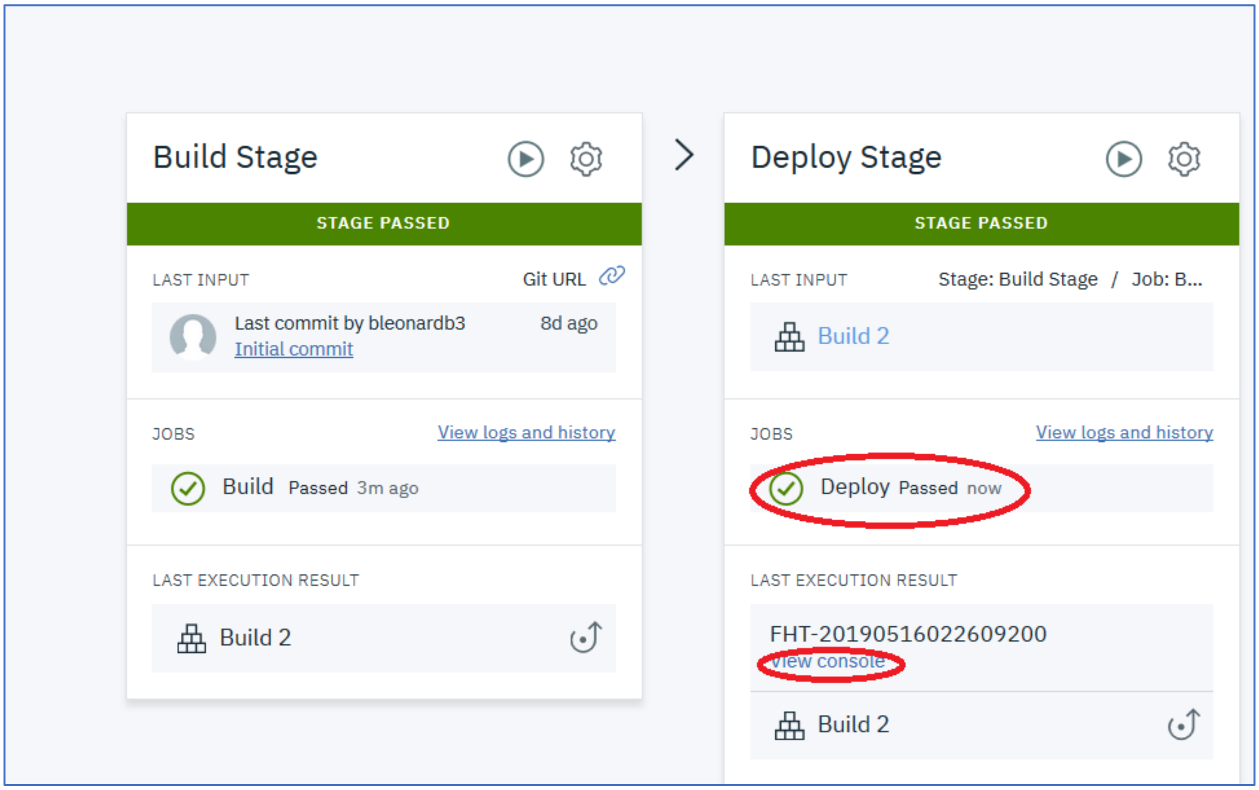
6. Please wait until the Region, Organization, and Space are filled in. **Note that these must match the corresponding Region (Dallas), Organization, and Space where the FHT_WML_Model_Deployed is deployed.** If this is not the case, try a different **Region**. Click on **Create.**



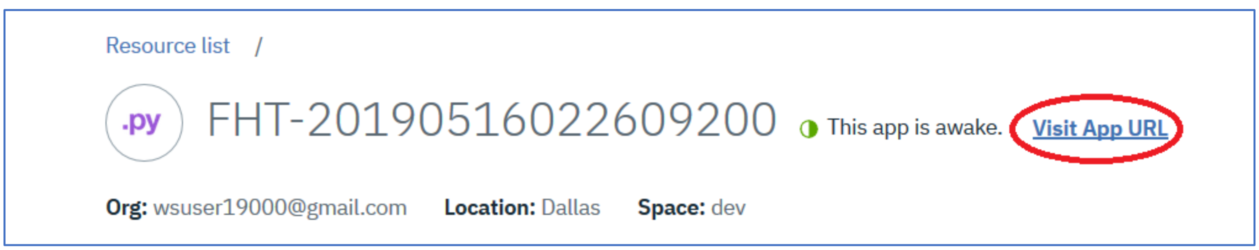
7. Your app is being created! To watch the pipeline deploy your app, click **Delivery Pipeline**.



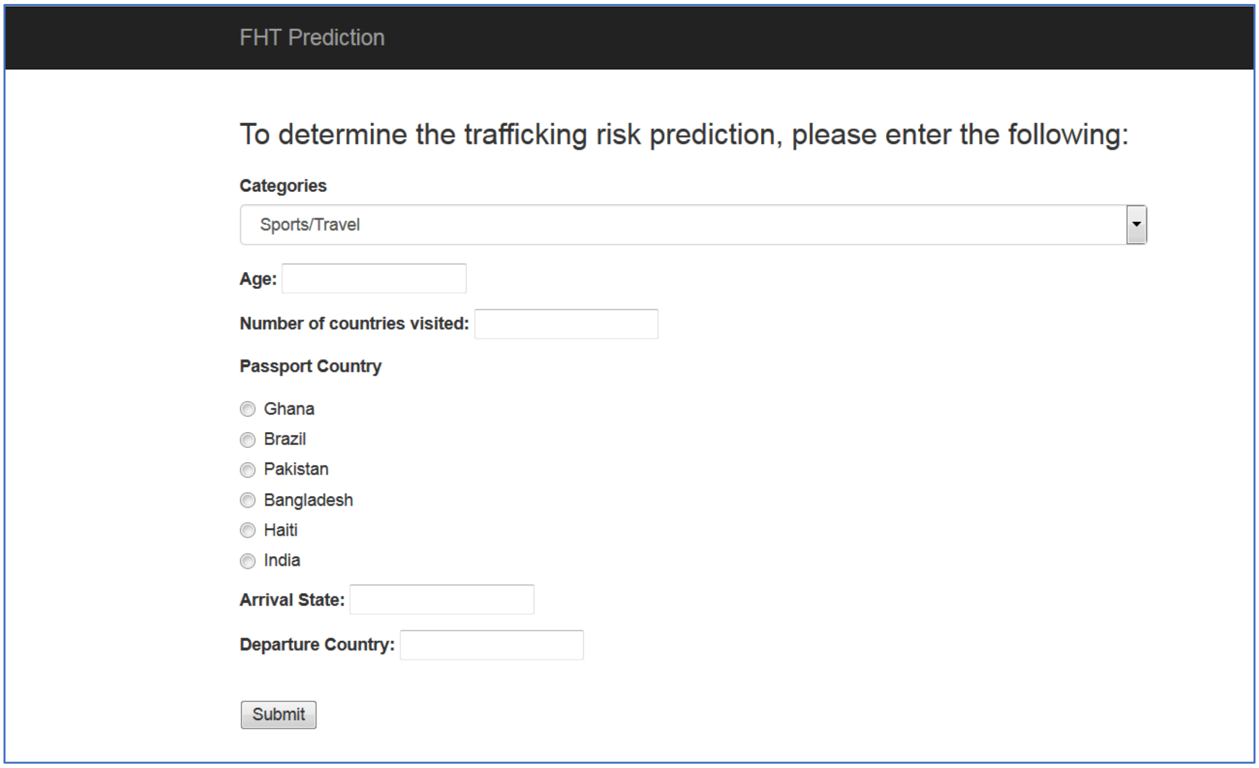
8. After the app is deployed successfully (should say Deploy Passed in the Deploy stage- may take about 2 minutes), view the running app by clicking on **View Console**



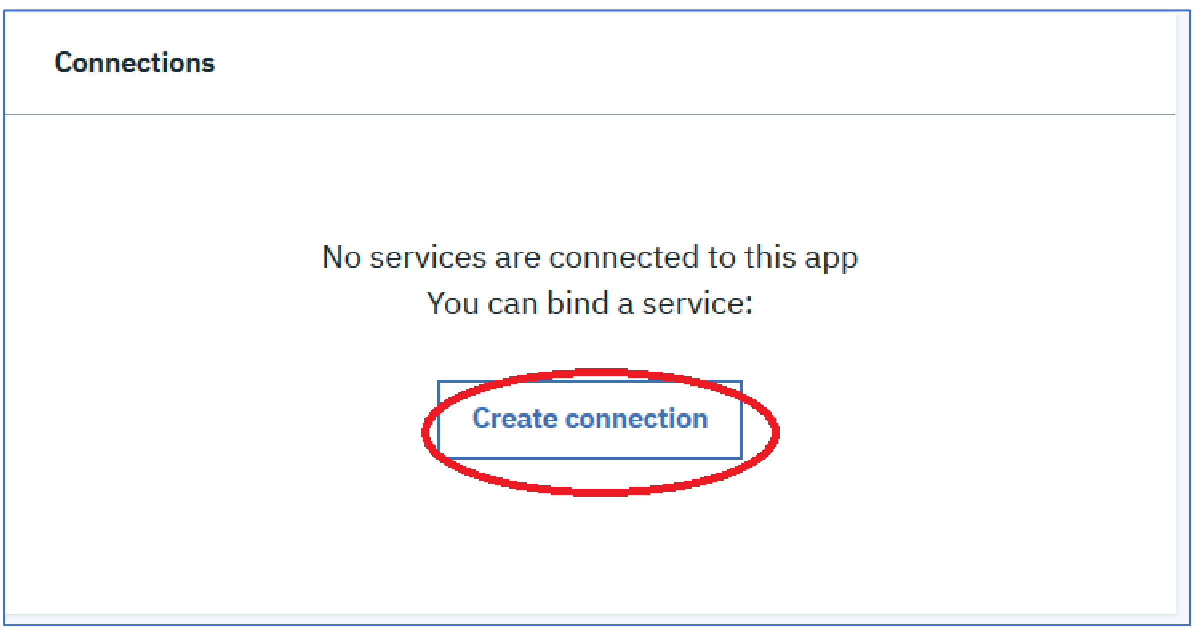
9. Click on **Visit App URL**



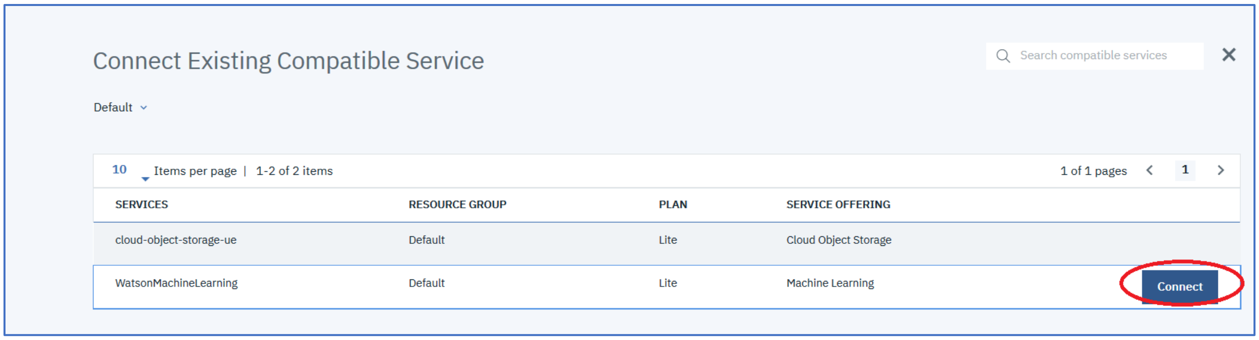
10. The web form collecting the FHT data should appear. Note that the application is not functional until we connect it to the Watson Machine Learning service so if you Submit you will get an error! Close the FHT Prediction browser tab.



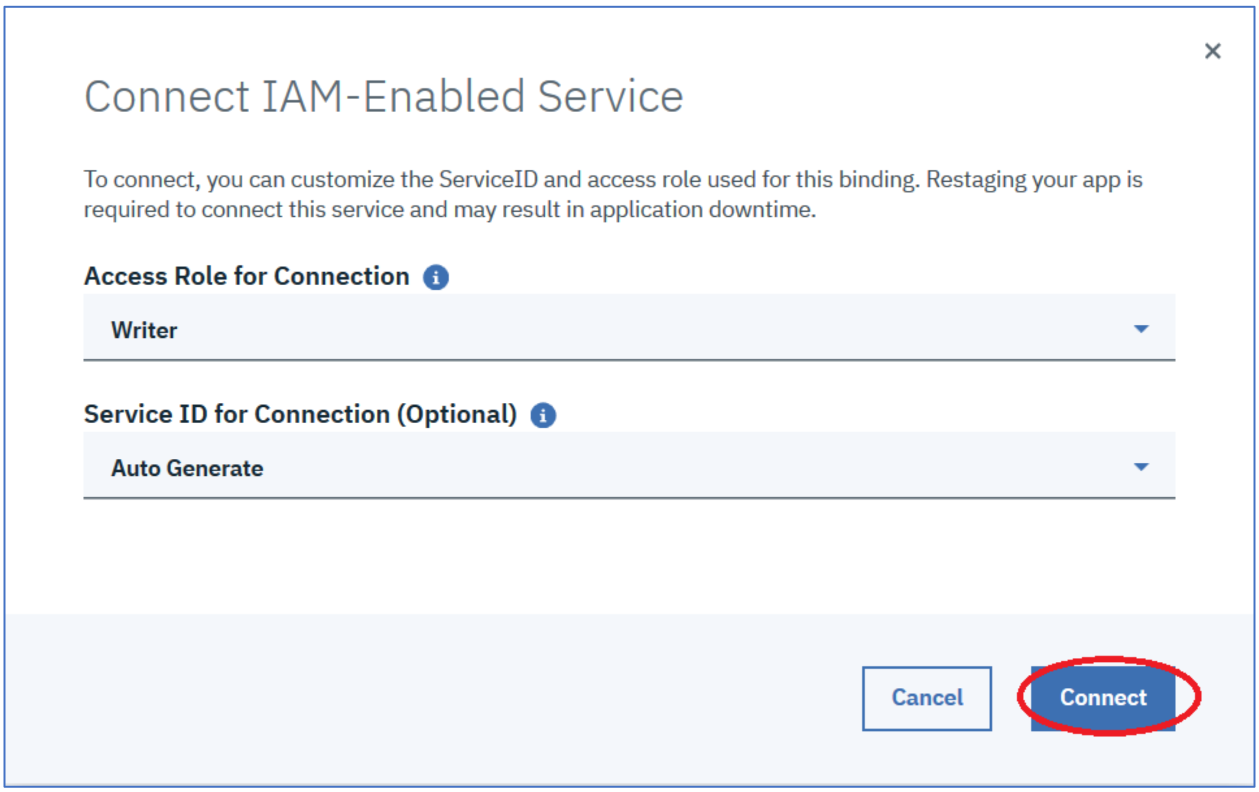
11. We are now going to connect the application to the Watson Machine Learning service that was created earlier. Scroll down until you see the Connections panel. Click on **Create Connection**.



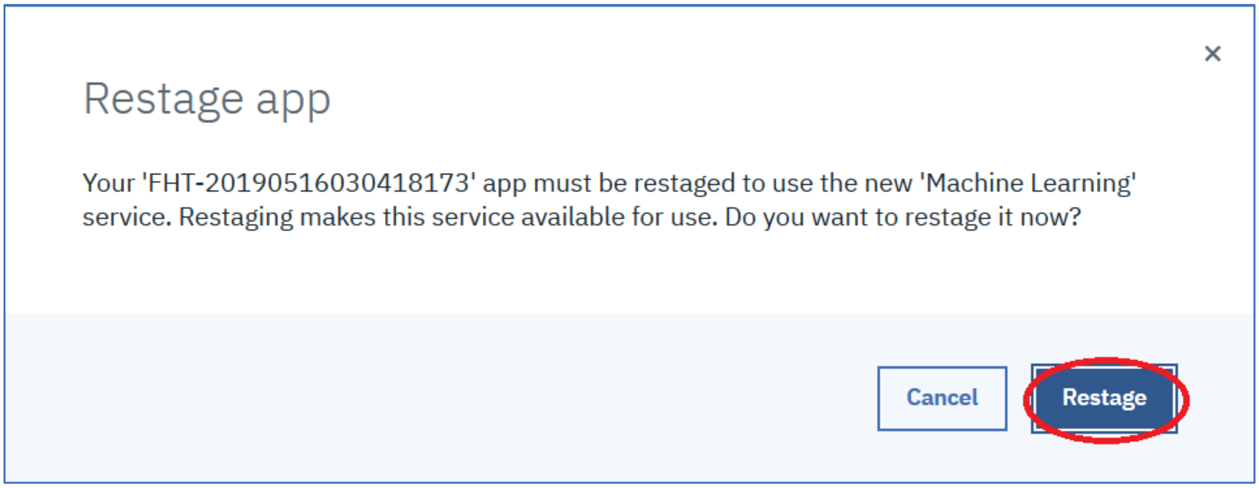
12. You should see at least 2 services listed, a Cloud Object Storage service, and a Watson Machine Learning service. Point the cursor on the **Machine Learning** service for your application, and then click on **Connect**.



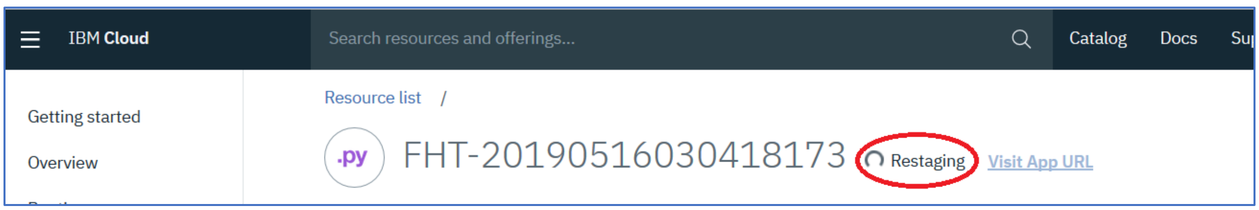
13. A **Connect IAM-enabled service** pop up will appear. Click **Connect**.

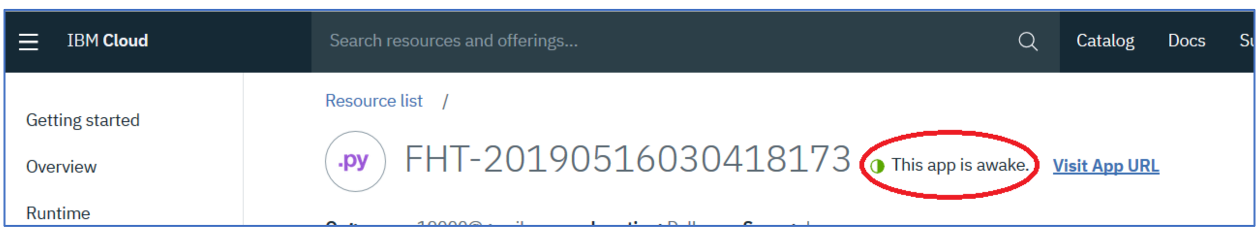


14. A **Restage app** pop up will appear. Click on **Restage**.

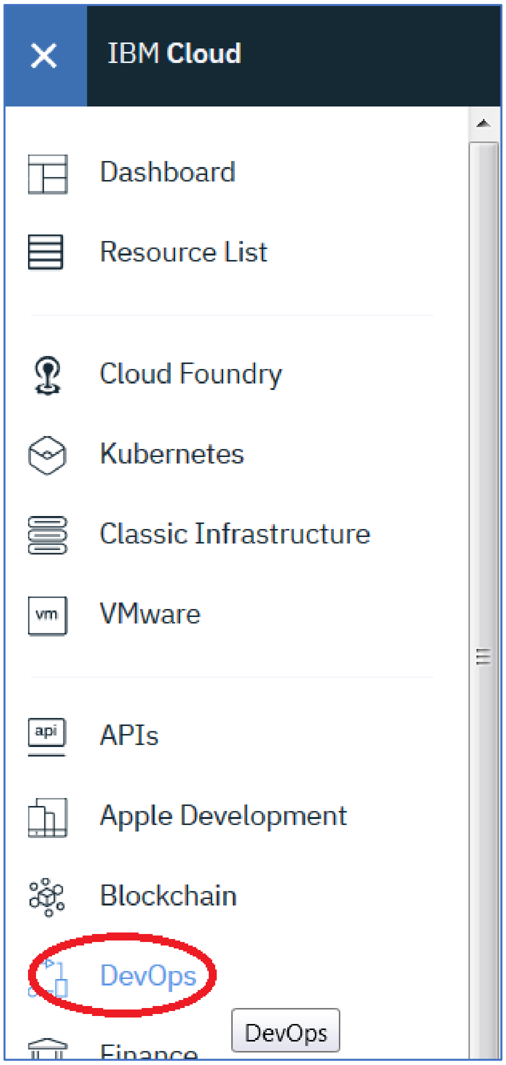


15. Wait for the application status to change from **Restaging** to This app is awake , or something similar.

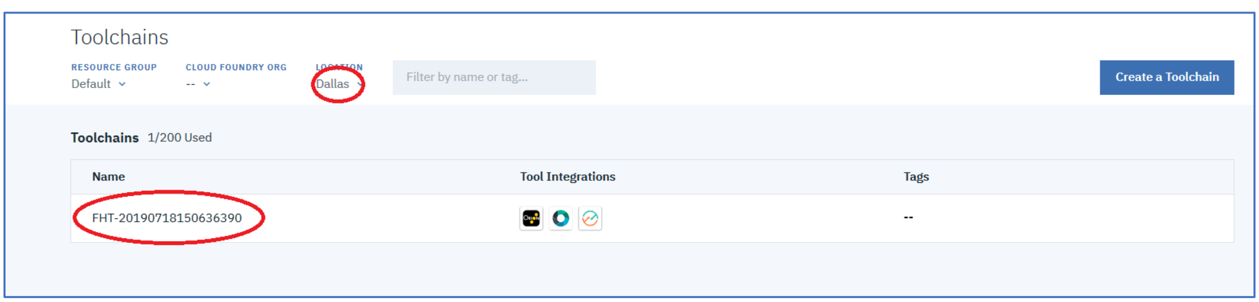




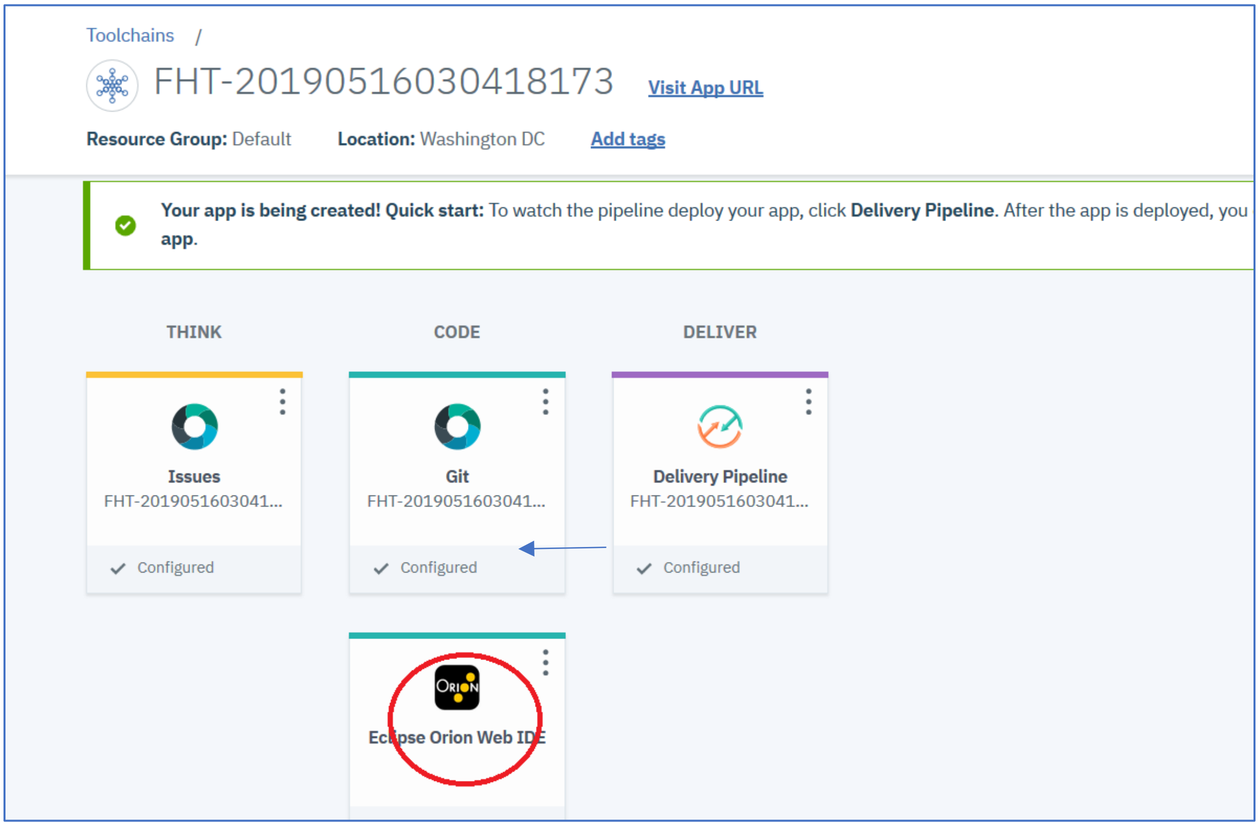
16. We now have tied the web application to the Watson Machine Learning service. Note that the Watson Machine Learning service could have more than one deployed model available to select and then embed in the web application. We now need to copy the scoring endpoint, which we previously copied and pasted 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.



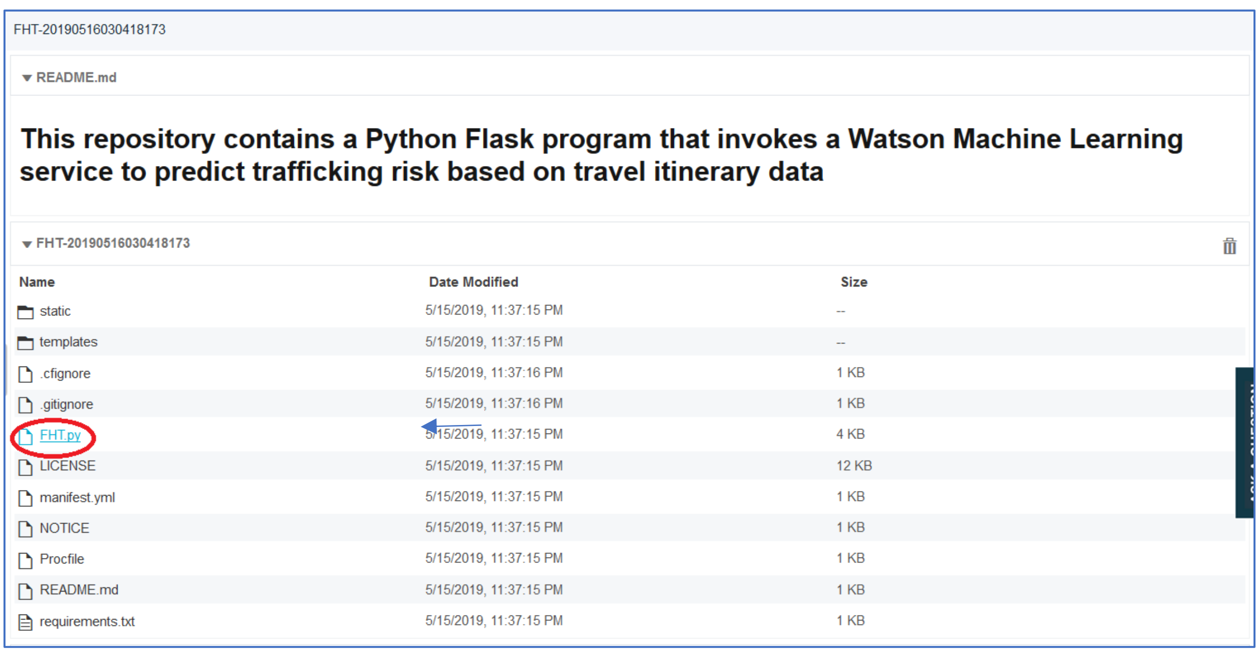
17. We are now going to paste the scoring endpoint into the application code. Click on the Toolchain (FHT-2019xxxxx below). If no toolchain appears, switch the location to Dallas.



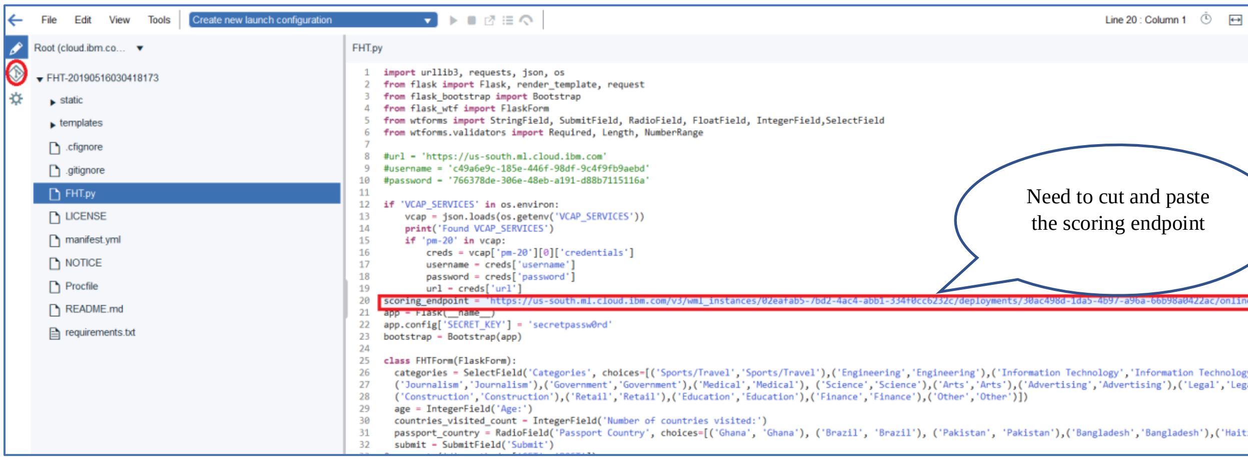
18. Click on the Eclipse Orion Web IDE. The IDE will enable the editing of the source code to update the scoring endpoint url.



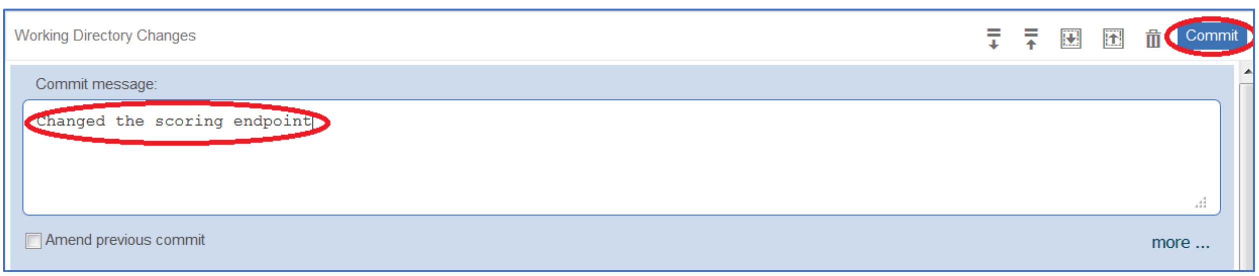
19. Click on the FHT.py file. This is a python source file.



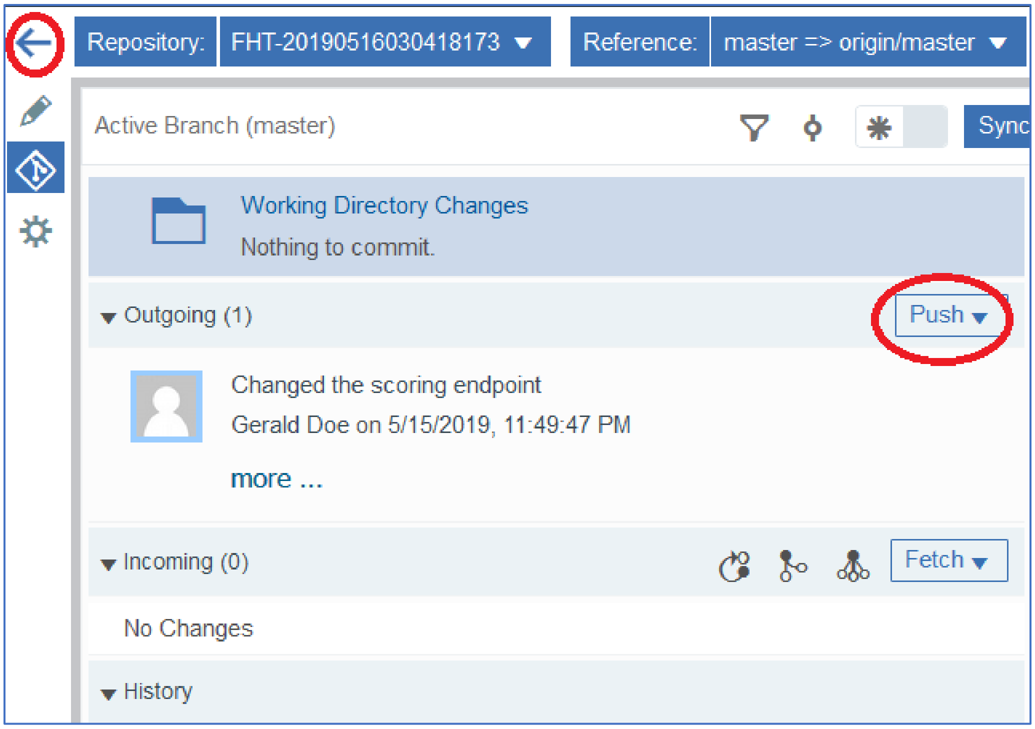
20. Go back to the Notepad file and copy the scoring endpoint to the clipboard. Look around line 20 in the FHT.py file for the "scoring endpoint =". Select the scoring endpoint value in line 20 (starting with https:// may want to use Shift-End to get to the end of the line, and then back up one space 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.



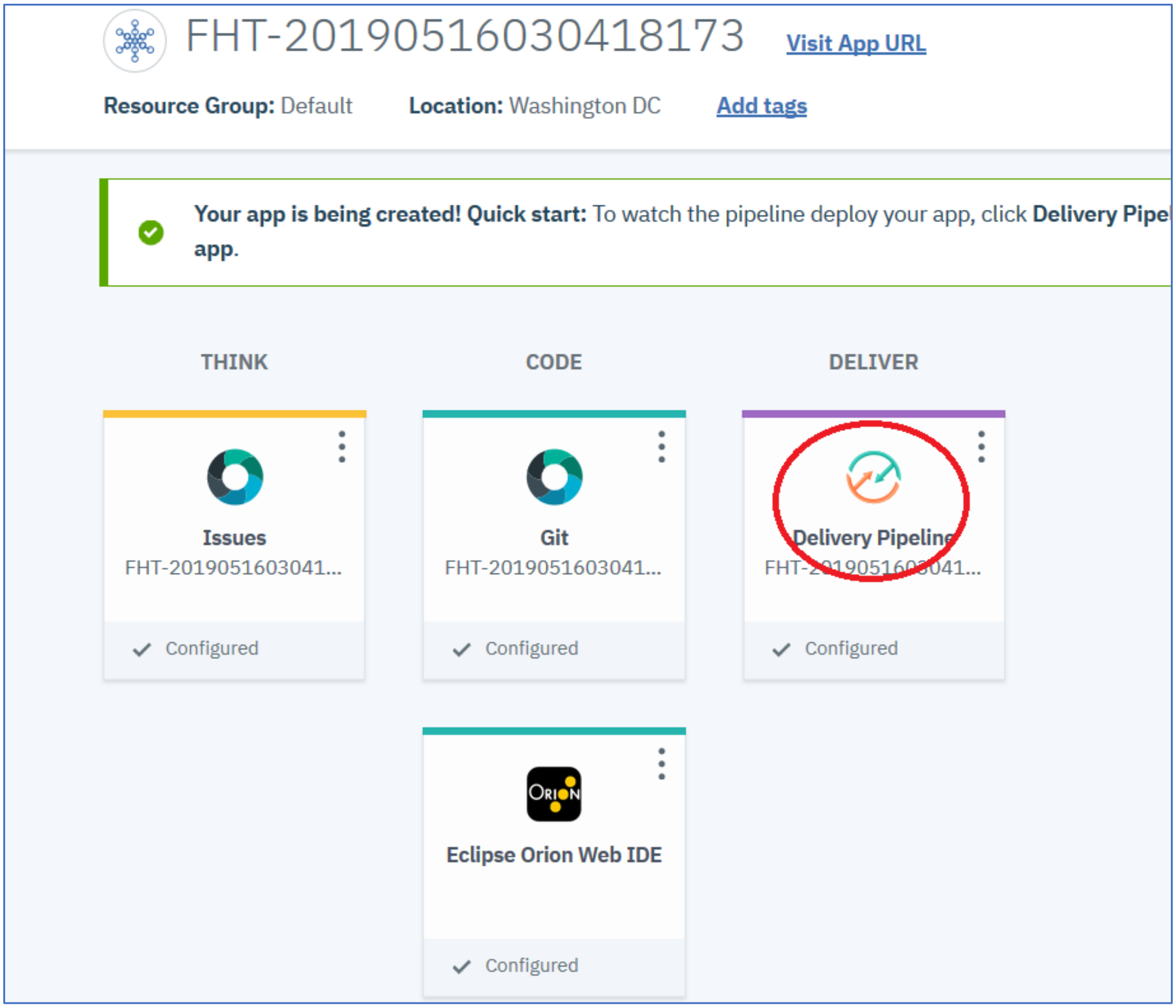
21. The next step is to commit the change to the git repository. Enter "Changed the Scoring Endpoint" in the Enter Commit Message field, and then click on **Commit**.



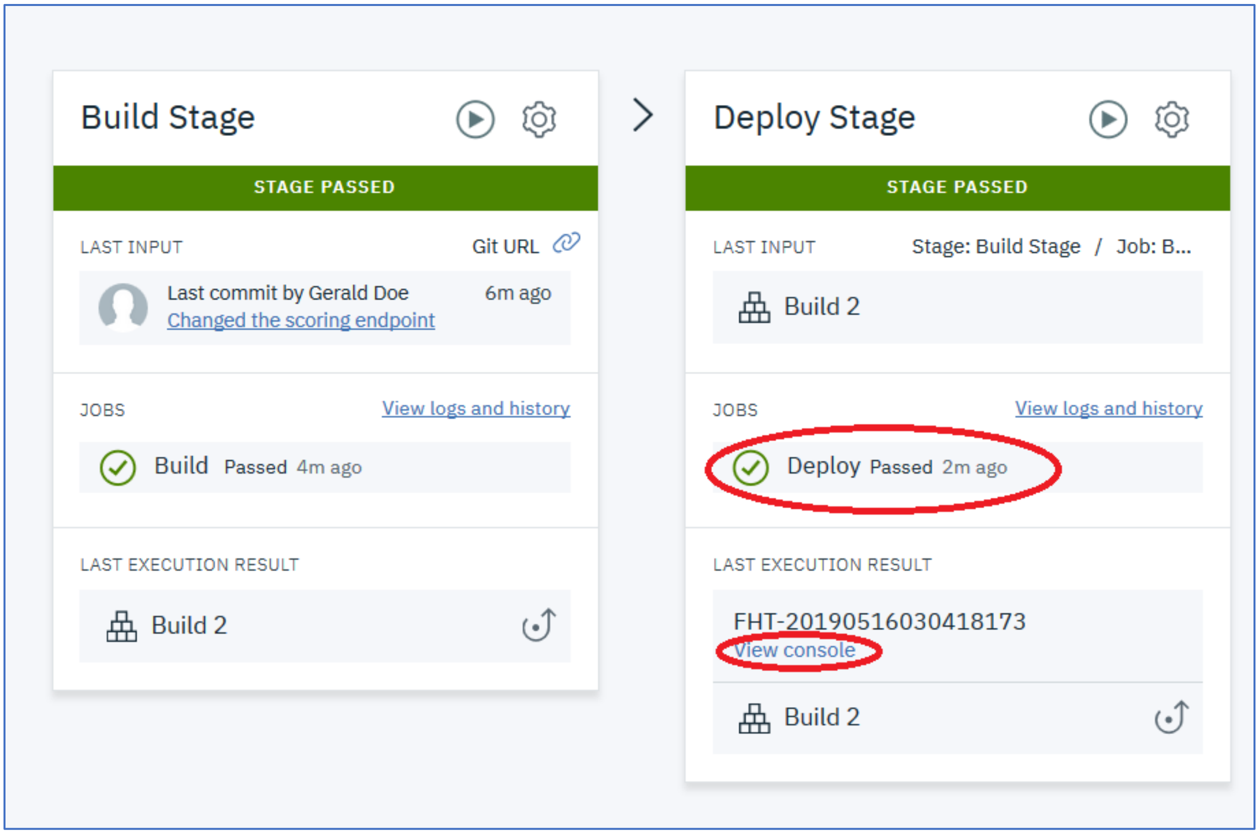
22. Then click on **Push** to push the changes to the central Git repo which will start the build and deploy of the application. Click on the left arrow to return to the Toolchain.



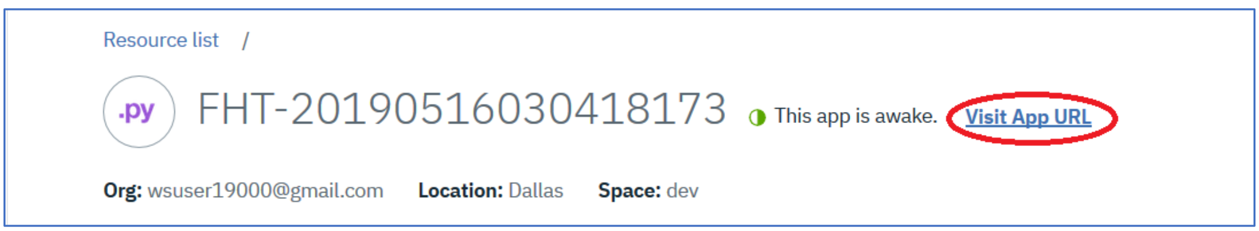
23. Click on the **Delivery Pipeline** to view status of the deployment as before. Refresh the screen if the stage status doesn't change.



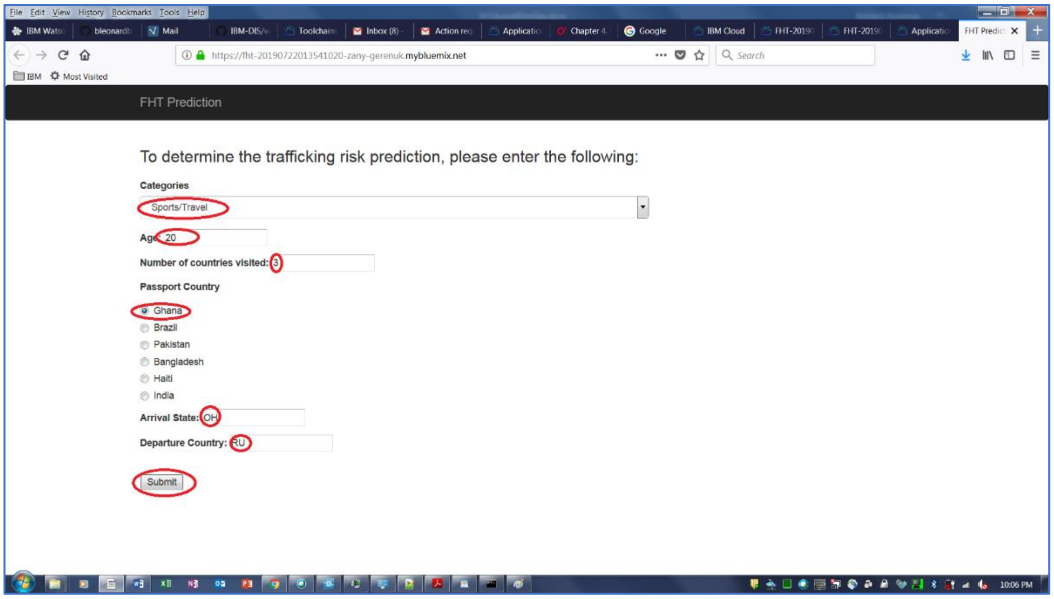
24. Once the Deployment status shows **Deploy passed now** it shouldn't take longer than 2 minutes (reload the browser in case the UI didn't update after 2 minutes). Click on **View Console.**



25. Click on **Visit App URL**.



26. The web form should appear. Enter data in all the fields and click on the **Submit** button.



27. 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.