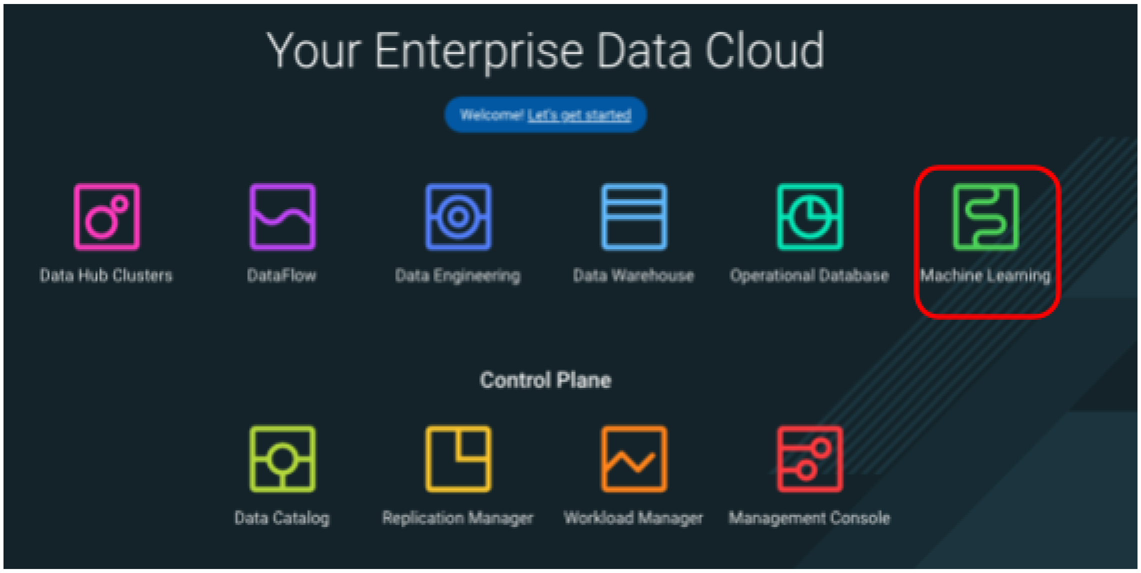
Cloudera Machine Learning Workshop

1) Open CDP, using the “admin” user within the Test Drive link.

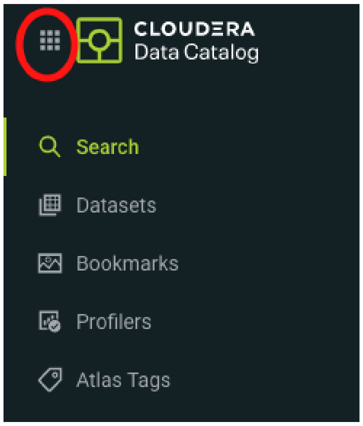
Your link should look something like (remember click the link in your email not the link below)
http://login.trycdp.com/auth/realms/trycdp-trialxx/protocol/saml/clients/samlclient?tn=trialxx_admin@trycdp.com&p=X *xx represents the trial user #
*X represents the password

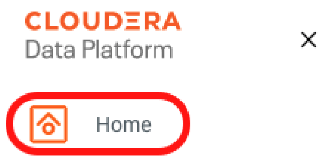
2) Click the “Machine Learning” tile within the CDP Home Screen



How do you get to the CDP Home Screen?

- From any experience such as “Data Catalog”, click the 9 square at the top left and then click “Home”

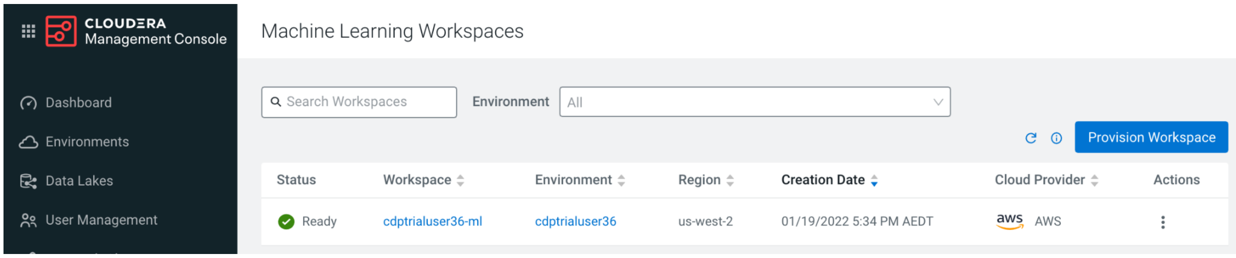




Part 1 - REQUIRED IN ORDER FOR LAB TO WORK PROPERLY: Create a CML Project

(A Workspace should already be provisioned for you)

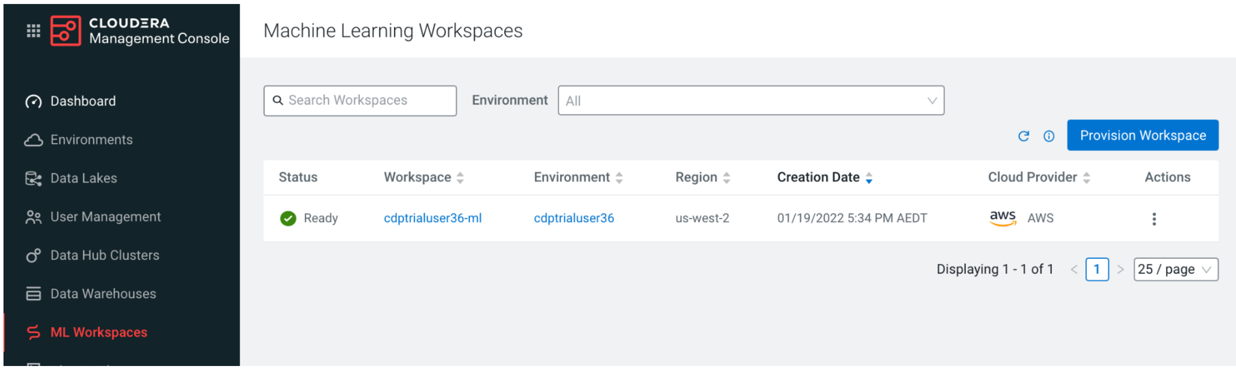
Workspaces are the heart of the Cloudera Machine Learning (CML)



Machine Learning Workspaces

Status	Workspace	Environment	Region	Creation Date	Cloud Provider	Actions
Ready	cdptrialuser36-ml	cdptrialuser36	us-west-2	01/19/2022 5:34 PM AEDT	AWS	⋮

A Workspace is a small cluster that runs on a kubernetes service to provide teams of data scientists to develop, test, train, and ultimately deploy machine learning models. **Click into the Workspace by clicking the Workspace name.**

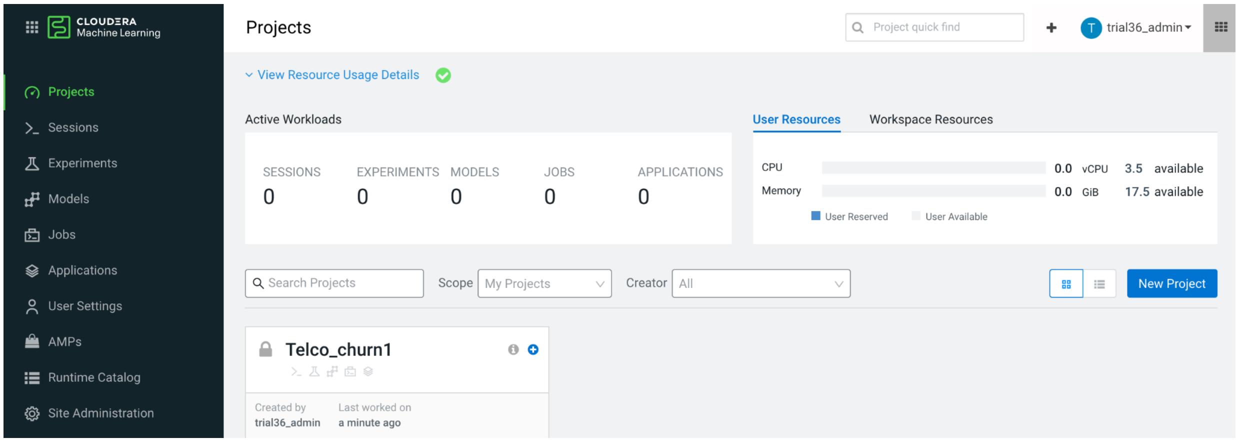


Machine Learning Workspaces

Status	Workspace	Environment	Region	Creation Date	Cloud Provider	Actions
Ready	cdptrialuser36-ml	cdptrialuser36	us-west-2	01/19/2022 5:34 PM AEDT	AWS	⋮

Displaying 1 - 1 of 1

You can visualize all of the Projects and Resources that are part of the Projects page. Next we will create a Project where we will develop and deploy models along with other CML features. **Click on "New Project"**



Active Workloads

SESSIONS	EXPERIMENTS	MODELS	JOBS	APPLICATIONS
0	0	0	0	0

User Resources | Workspace Resources

CPU	0.0	vCPU	3.5	available
Memory	0.0	GiB	17.5	available

■ User Reserved ■ User Available

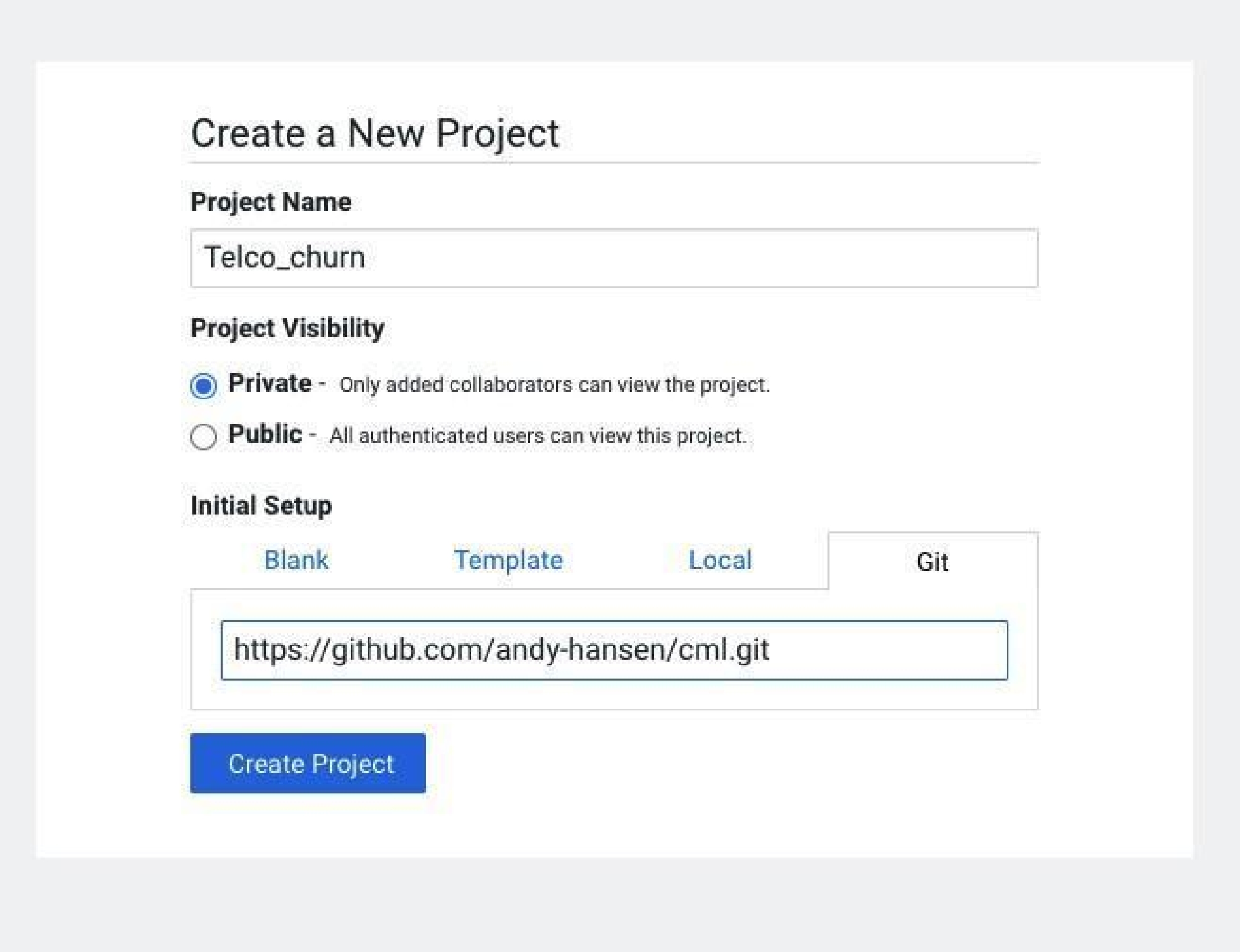
Telco_churn1

Created by trial36_admin

Last worked on a minute ago

When creating a new project give a Name, Visibility, and initial configuration.

Project Name: Telco_churn Visibility: Private Initial Setup: Git -> https://github.com/andy-hansen/cml.git



Create a New Project

Project Name

Telco_churn

Project Visibility

● Private - Only added collaborators can view the project.

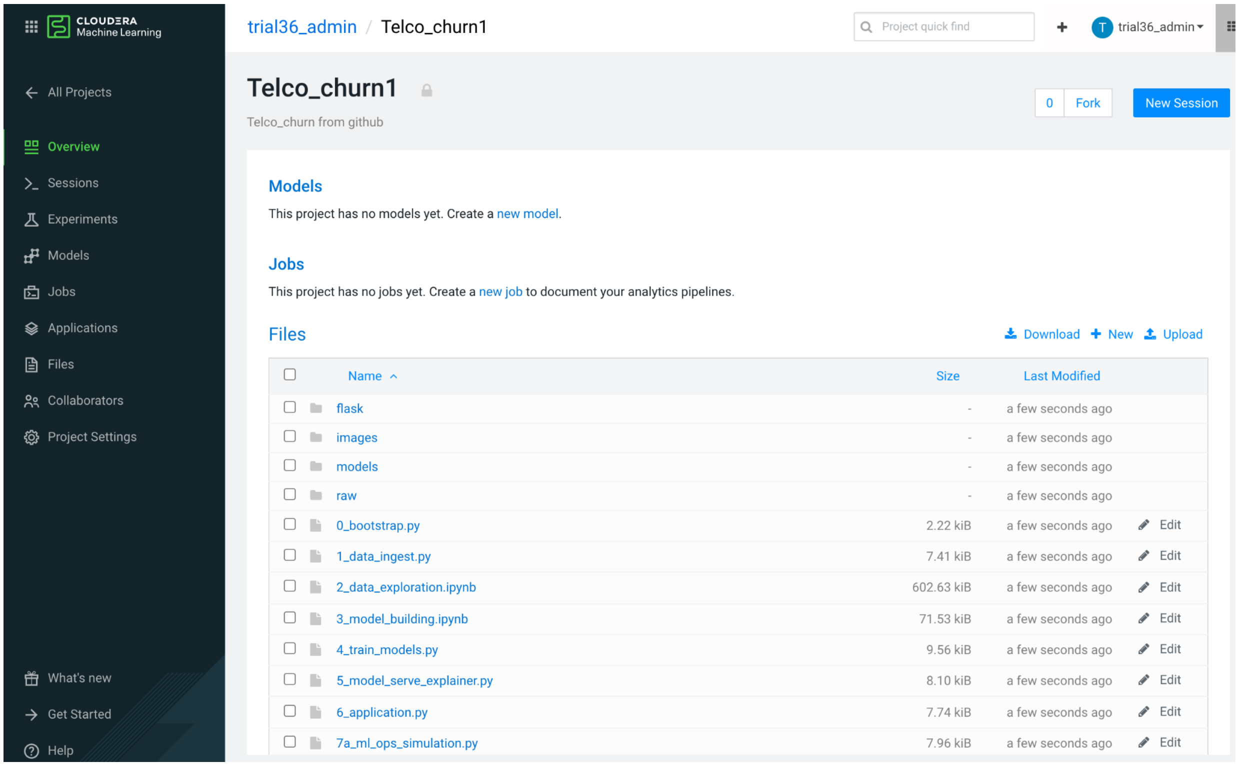
○ Public - All authenticated users can view this project.

Initial Setup

https://github.com/andy-hansen/cml.git

Part 2: CML Project Overview

Overview gives you access to all the features of a CML project. We will only have files copied from the Github repo currently. Initially it is good to start on the management components of a project.



Telco_churn1

Telco_churn from github

Models

This project has no models yet. Create a new model.

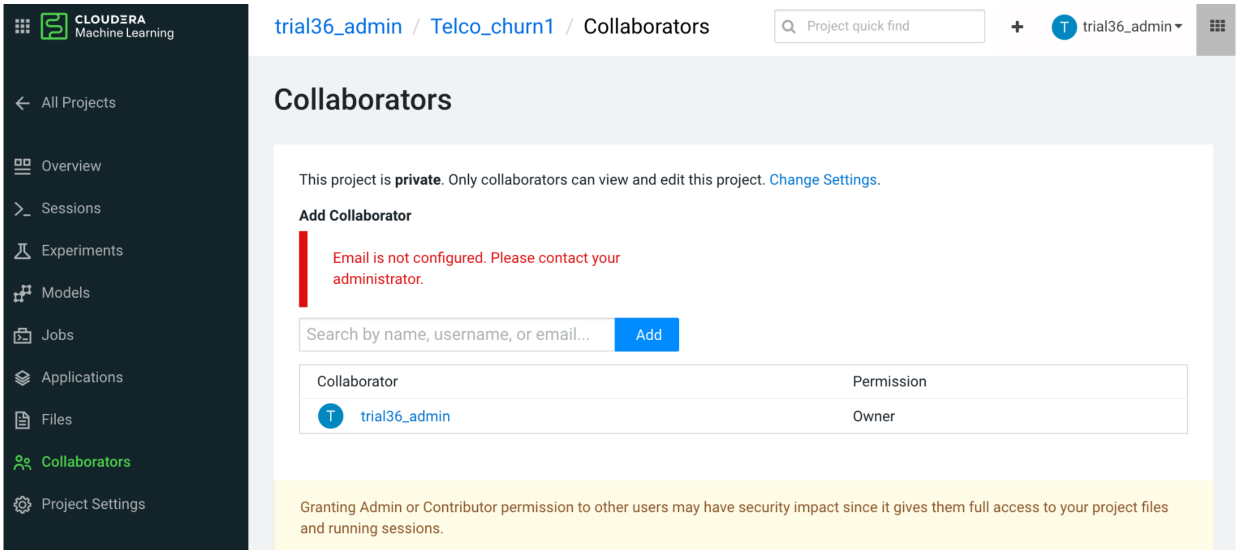
Jobs

This project has no jobs yet. Create a new job to document your analytics pipelines.

Files

□	Name	Size	Last Modified	
□	flask	-	a few seconds ago	
□	images	-	a few seconds ago	
□	models	-	a few seconds ago	
□	raw	-	a few seconds ago	
□	0_bootstrap.py	2.22 kiB	a few seconds ago	Edit
□	1_data_ingest.py	7.41 kiB	a few seconds ago	Edit
□	2_data_exploration.ipynb	602.63 kiB	a few seconds ago	Edit
□	3_model_building.ipynb	71.53 kiB	a few seconds ago	Edit
□	4_train_models.py	9.56 kiB	a few seconds ago	Edit
□	5_model_serve_explainer.py	8.10 kiB	a few seconds ago	Edit
□	6_application.py	7.74 kiB	a few seconds ago	Edit
□	7a_ml_ops_simulation.py	7.96 kiB	a few seconds ago	Edit

Collaborators:
For our demo we aren't adding additional collaborators.
You can give access to other users with certain permissions for the encompassing project so teams of users can collaborate together. You can set up Admins, Contributor, Operator, and Viewer permissions.



Collaborators

This project is **private**. Only collaborators can view and edit this project. Change Settings.

Add Collaborator

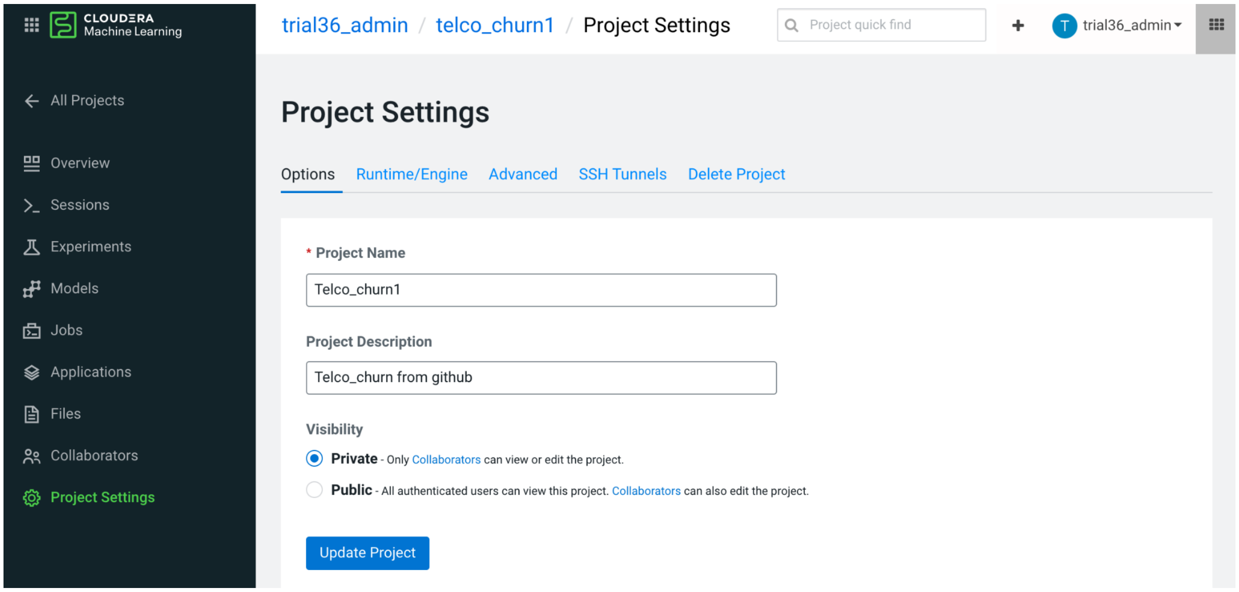
Email is not configured. Please contact your administrator.

Collaborator	Permission
T trial36_admin	Owner

Granting Admin or Contributor permission to other users may have security impact since it gives them full access to your project files and running sessions.

Project Settings:

Taking a look at Project Settings, this is where you can define several options for the current project. You have the ability to define different engines where your code in CML will run. There are project variables that can be defined and used throughout your code. SSH tunnels can also be configured to connect to other services as needed. More details can be found in our docs [here](#).



Project Settings

* Project Name

Telco_churn1

Project Description

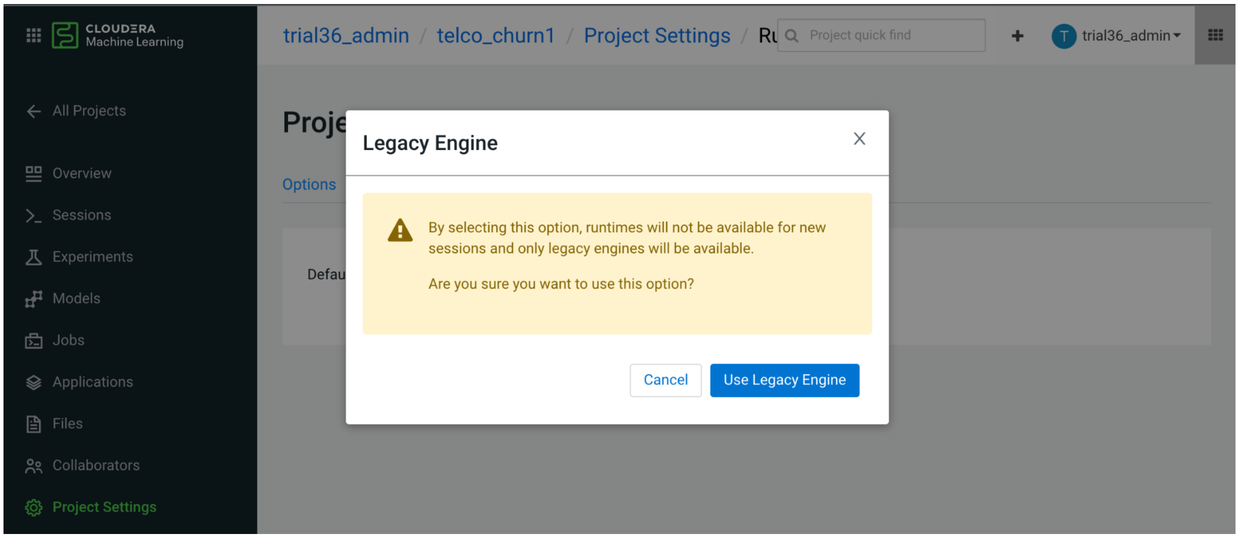
Telco_churn from github

Visibility

● **Private** - Only Collaborators can view or edit the project.

○ **Public** - All authenticated users can view this project. Collaborators can also edit the project.

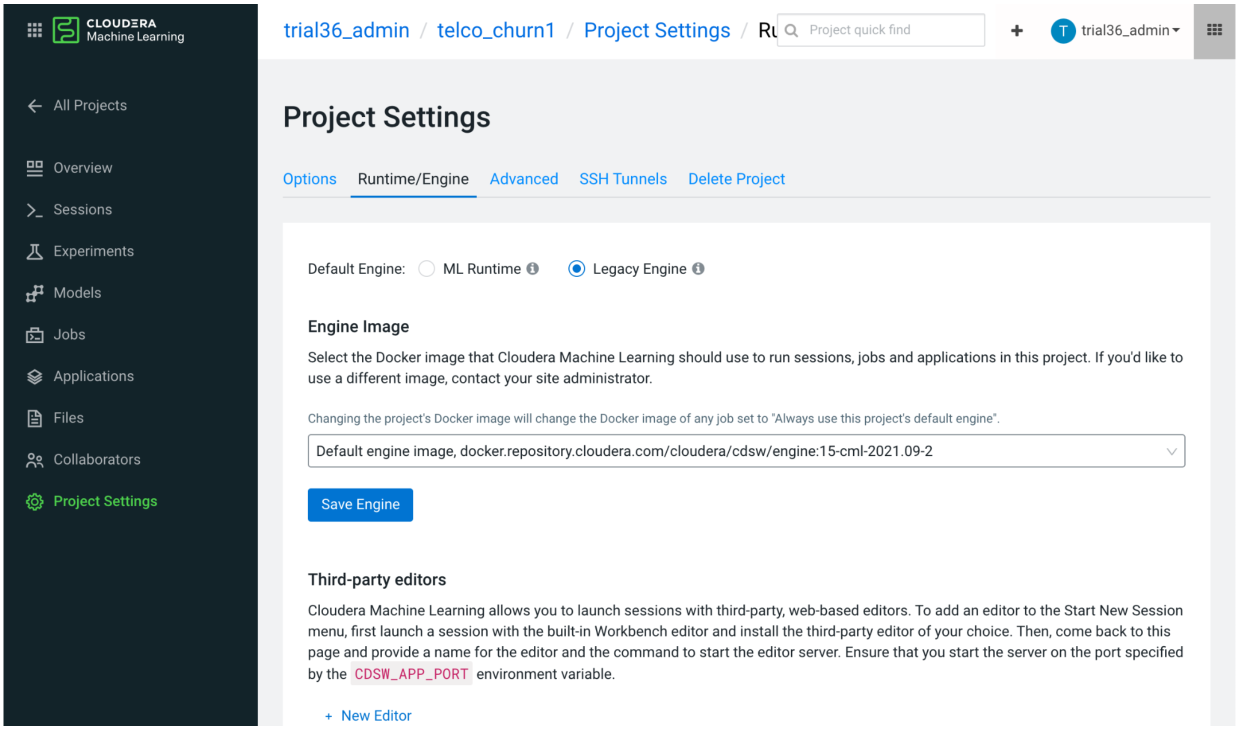
-> Change the Runtime/Engine to Legacy Engine -> Click Save Engine



Legacy Engine

By selecting this option, runtimes will not be available for new sessions and only legacy engines will be available.

Are you sure you want to use this option?



Project Settings

Default Engine: ○ ML Runtime ● Legacy Engine

Engine Image

Select the Docker image that Cloudera Machine Learning should use to run sessions, jobs and applications in this project. If you'd like to use a different image, contact your site administrator.

Changing the project's Docker image will change the Docker image of any job set to "Always use this project's default engine".

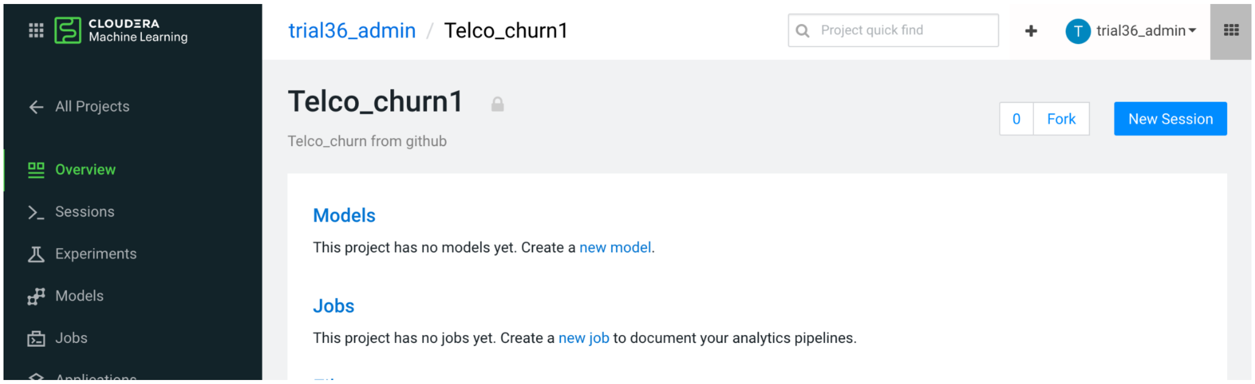
Default engine image, docker.repository.cloudera.com/cloudera/cdsw/engine:15-cml-2021.09-2

Third-party editors

Cloudera Machine Learning allows you to launch sessions with third-party, web-based editors. To add an editor to the Start New Session menu, first launch a session with the built-in Workbench editor and install the third-party editor of your choice. Then, come back to this page and provide a name for the editor and the command to start the editor server. Ensure that you start the server on the port specified by the `CDSW_APP_PORT` environment variable.

Part 3: CML Sessions and Workbench

Sessions allow you to perform actions such as run R or Python code. They also provide access to an interactive command prompt and terminal. Sessions will be built on a specified Engine Image, which is a docker container that is deployed onto the Workspace. In addition you can specify how many resources are used per session. **From the Overview page click on New Session**



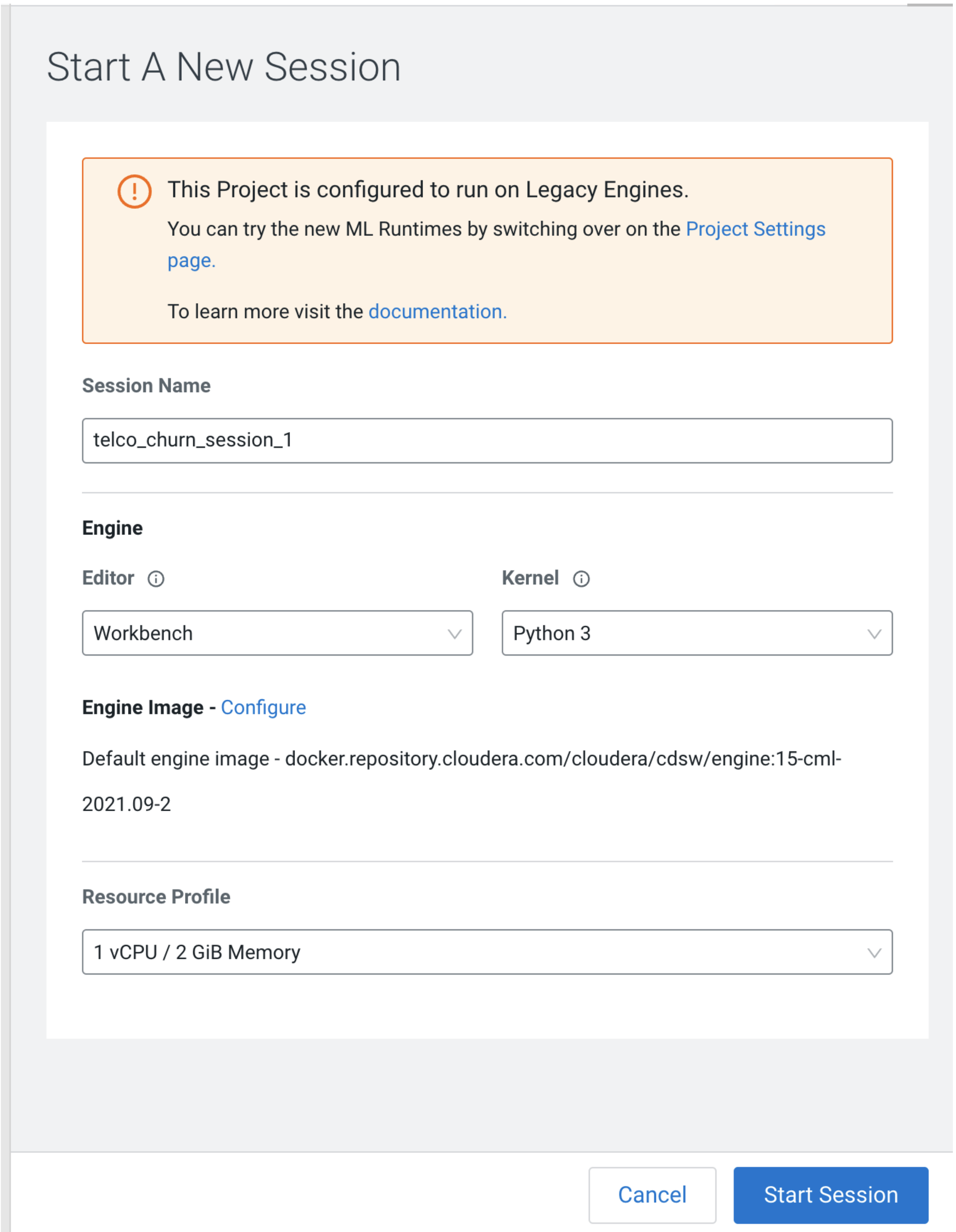
Telco_churn from github

This project has no models yet. Create a new model.

This project has no jobs yet. Create a new job to document your analytics pipelines.

Session Name: telco_churn_session_1 Editor: Workbench Kernel: Python 3 Engine Image: Default Resource Profile: 1vCPU/2 GiB Memory

Then select Start Session



Start A New Session

This Project is configured to run on Legacy Engines.

You can try the new ML Runtimes by switching over on the Project Settings page.

To learn more visit the documentation.

Session Name

telco_churn_session_1

Engine

Editor: Workbench

Kernel: Python 3

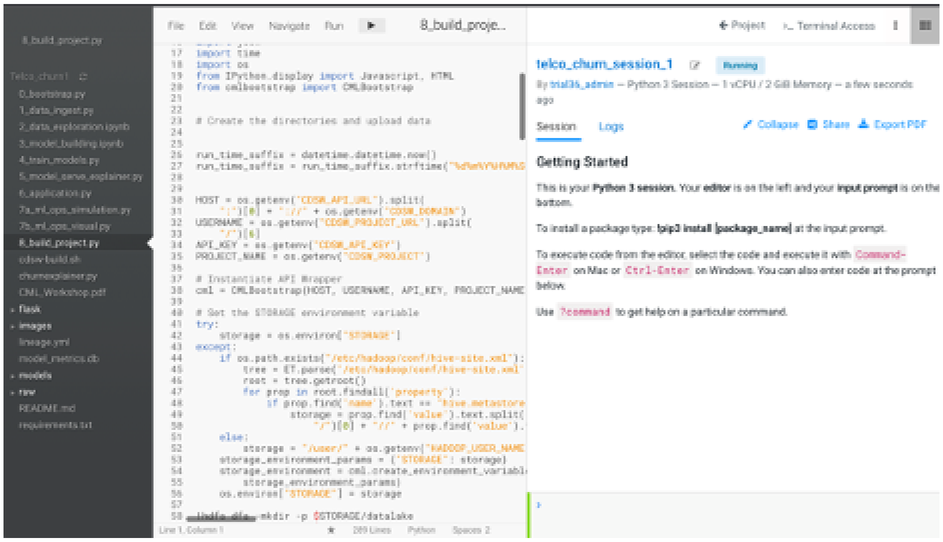
Engine Image - Configure

Default engine image - docker.repository.cloudera.com/cloudera/cdsw/engine:15-cml-2021.09-2

Resource Profile

1 vCPU / 2 GiB Memory

The Workbench is now starting up and deploying a container onto the workspace at this point. Going from left to right you will see the project files, editor pane, and session pane. **Once you see the flashing red line on the bottom of the session pane turn steady green the container has been successfully started.**



```
import time
import os
from IPython.display import Javascript, HTML
from cmlbootstrap import CMLBootstrap


# Create the directories and upload data


run_time_suffix = datetime.datetime.now()
run_time_suffix = run_time_suffix.strftime("%d%m%Y%H%M%S


HOST = os.getenv("CDSW_API_URL").split(
    ":")[0] + "://" + os.getenv("CDSW_DOMAIN")
USERNAME = os.getenv("CDSW_PROJECT_URL").split(
    "/")[6]
API_KEY = os.getenv("CDSW_API_KEY")
PROJECT_NAME = os.getenv("CDSW_PROJECT")

# Instantiate API Wrapper
cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME

# Set the STORAGE environment variable
try:
    storage = os.environ["STORAGE"]
except:
    if os.path.exists("/etc/hadoop/conf/hive-site.xml"):
        tree = ET.parse("/etc/hadoop/conf/hive-site.xml"
        root = tree.getroot()
        for prop in root.findall('property'):
            if prop.find('name').text == "hive.metastore
                storage = prop.find('value').text.split(
                    "/")[0] + "//" + prop.find('value').
    else:
        storage = "/user/" + os.getenv("HADOOP_USER_NAME
    storage_environment_params = {"STORAGE": storage}
    storage_environment = cml.create_environment_variabl
        storage_environment_params)
    os.environ["STORAGE"] = storage

!hdfs dfs -mkdir -p $STORAGE/datalake
```

telco_churn_session_1

By trial36_admin — Python 3 Session — 1 vCPU / 2 GiB Memory — a few seconds ago

Getting Started

This is your **Python 3 session**. Your **editor** is on the left and your **input prompt** is on the bottom.

To install a package type: **!pip3 install [package_name]** at the input prompt.

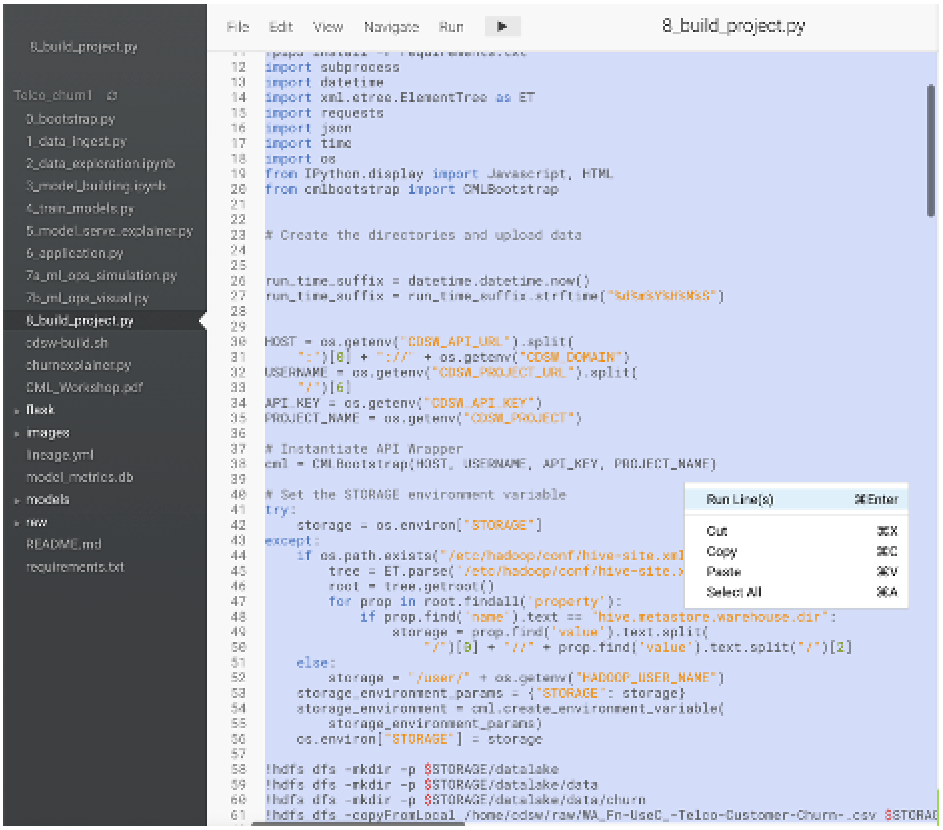
To execute code from the editor, select the code and execute it with `Command-Enter` on Mac or `Ctrl-Enter` on Windows. You can also enter code at the prompt below.

Use `?command` to get help on a particular command.

Open up the script 8_build_project.py from the left pane. In the editor pane we are going to select and run the script in several parts. Throughout the script you will see breaks in the code defined by Part 1, Part 2, Part 3, etc.

Part 1: Will install any required packages to be used. As an example, flask is installed as part of the project. A variable is also set as part of the project. Last but not least we are loading a file from the project as a test dataset and moving that to a s3 location.

Select and highlight the contents of part 1, then Right click -> Run Line(s)



```
!pip3 install -r requirements.txt
import subprocess
import datetime
import xml.etree.ElementTree as ET
import requests
import json
import time
import os
from IPython.display import Javascript, HTML
from cmlbootstrap import CMLBootstrap


# Create the directories and upload data


run_time_suffix = datetime.datetime.now()
run_time_suffix = run_time_suffix.strftime("%d%m%Y%H%M%S")


HOST = os.getenv("CDSW_API_URL").split(
    ":")[0] + "://" + os.getenv("CDSW_DOMAIN")
USERNAME = os.getenv("CDSW_PROJECT_URL").split(
    "/")[6]
API_KEY = os.getenv("CDSW_API_KEY")
PROJECT_NAME = os.getenv("CDSW_PROJECT")

# Instantiate API Wrapper
cml = CMLBootstrap(HOST, USERNAME, API_KEY, PROJECT_NAME)

# Set the STORAGE environment variable
try:
    storage = os.environ["STORAGE"]
except:
    if os.path.exists("/etc/hadoop/conf/hive-site.xml
        tree = ET.parse('/etc/hadoop/conf/hive-site.x
        root = tree.getroot()
        for prop in root.findall('property'):
            if prop.find('name').text == "hive.metastore.warehouse.dir":
                storage = prop.find('value').text.split(
                    "/")[0] + "//" + prop.find('value').text.split("/")[2]
    else:
        storage = "/user/" + os.getenv("HADOOP_USER_NAME")
    storage_environment_params = {"STORAGE": storage}
    storage_environment = cml.create_environment_variable(
        storage_environment_params)
    os.environ["STORAGE"] = storage

!hdfs dfs -mkdir -p $STORAGE/datalake
!hdfs dfs -mkdir -p $STORAGE/datalake/data
!hdfs dfs -mkdir -p $STORAGE/datalake/data/churn
!hdfs dfs -copyFromLocal /home/cdsw/raw/WA_Fn-UseC_-Telco-Customer-Churn-.csv $STORAG
```

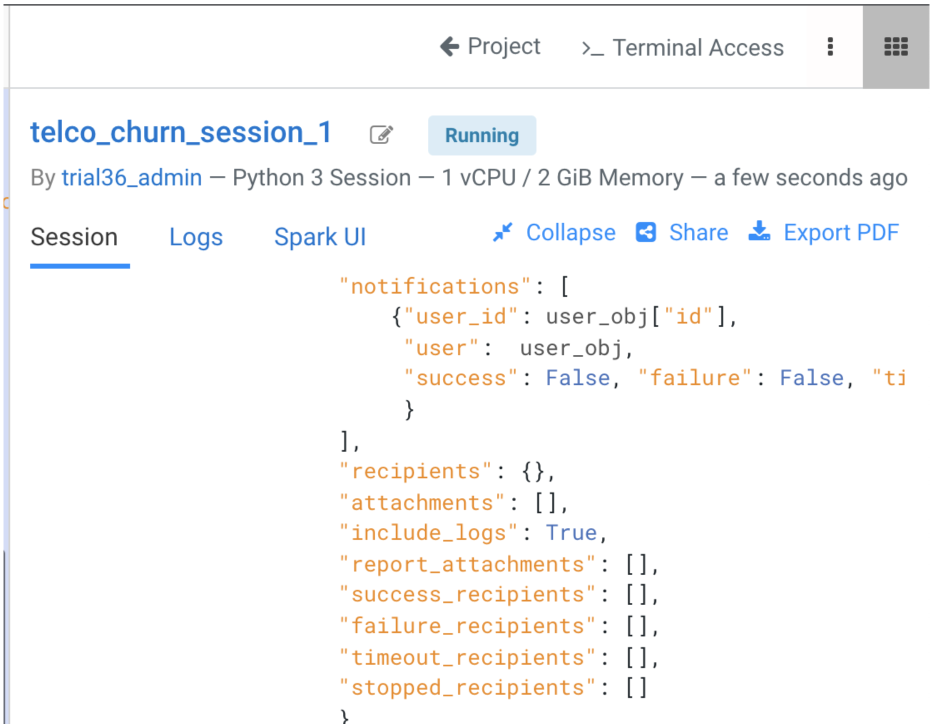
Part 2: The telco churn dataset is ingested from s3 and a hive table is created using Spark.

Select and highlight the contents of part 2, then Right click -> Run Line(s)

Part 3: Create a CML Job and start the job. A job automates the action of launching an engine, running a script, and tracking the results, all in one batch process. Jobs are created within the purview of a single project and can be configured to run on a recurring schedule. You can customize the engine environment for a job, set up email alerts for successful or failed job runs, and email the output of the job to yourself or a colleague.

Select and highlight the contents of part 3, then Right click -> Run Line(s)

Once the Job is started we will look at what was created. **Click the Project button** (top right corner of screen).

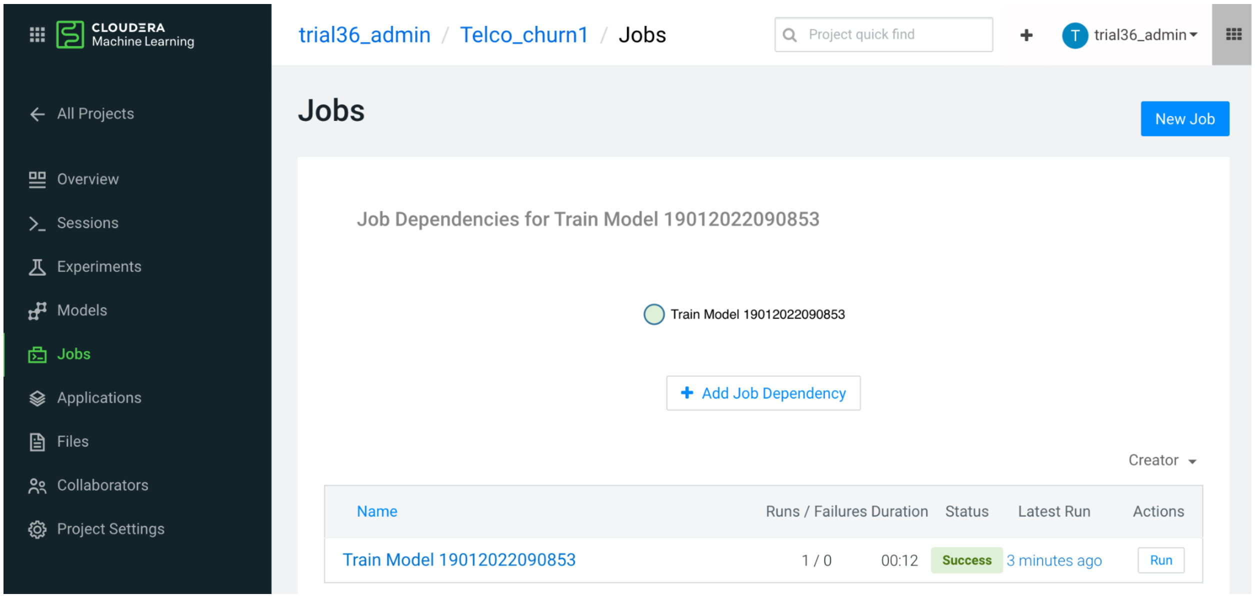


telco_churn_session_1 Running

By trial36_admin — Python 3 Session — 1 vCPU / 2 GiB Memory — a few seconds ago

```
                        "notifications": [
                            {"user_id": user_obj["id"],
                             "user":  user_obj,
                             "success": False, "failure": False, "ti
                             }
                        ],
                        "recipients": {},
                        "attachments": [],
                        "include_logs": True,
                        "report_attachments": [],
                        "success_recipients": [],
                        "failure_recipients": [],
                        "timeout_recipients": [],
                        "stopped_recipients": []
                        }
```

Click on Jobs to explore the job that was created and details can be found by **Clicking on the Job Name.**



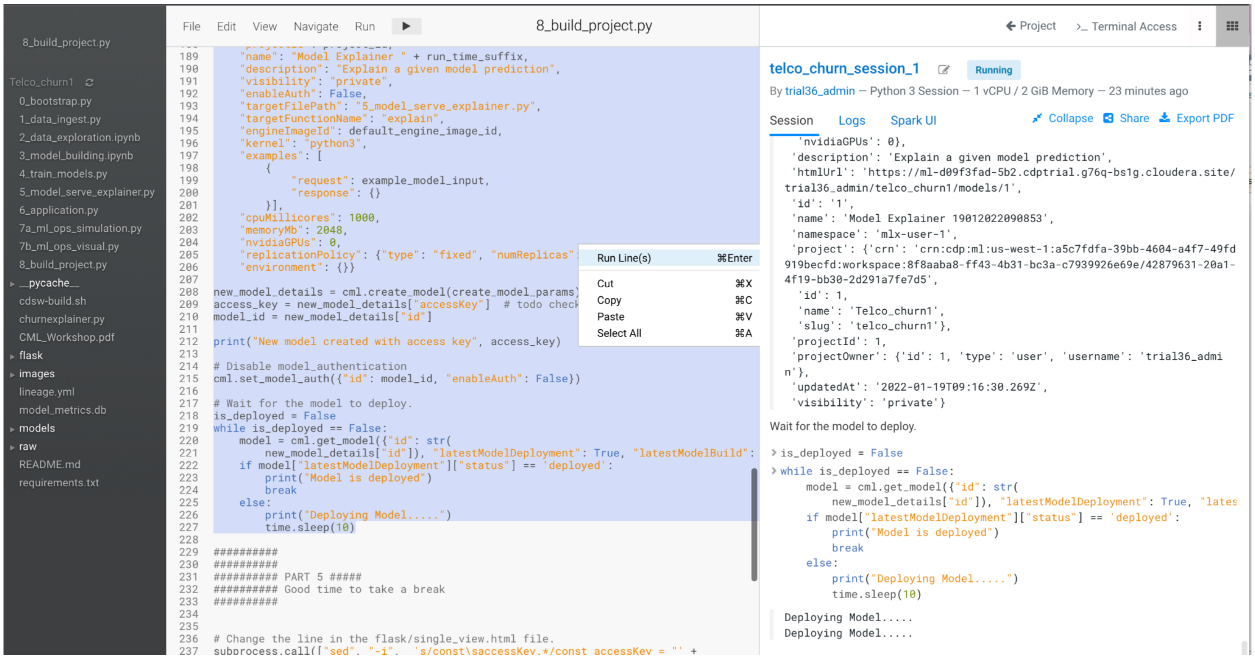
Jobs

Job Dependencies for Train Model 19012022090853

Name	Runs / Failures	Duration	Status	Latest Run	Actions
Train Model 19012022090853	1 / 0	00:12	Success	3 minutes ago	Run

Part 4: Using CML, you can create any function within a script and deploy it to a REST API. In a machine learning project, this will typically be a predict function that will accept an input and return a prediction based on the model's parameters.

Select and highlight the contents of part 4, then Right click -> Run Line(s)
When getting to deploying the Model this can take a little time, and is a good spot to stretch the legs and refill your coffee.



```
    "name": "Model Explainer " + run_time_suffix,
    "description": "Explain a given model prediction",
    "visibility": "private",
    "enableAuth": False,
    "targetFilePath": "5_model_serve_explainer.py",
    "targetFunctionName": "explain",
    "engineImageId": default_engine_image_id,
    "kernel": "python3",
    "examples": [
        {
            "request": example_model_input,
            "response": {}
        }],
    "cpuMillicores": 1000,
    "memoryMb": 2048,
    "nvidiaGPUs": 0,
    "replicationPolicy": {"type": "fixed", "numReplicas"
    "environment": {}}

new_model_details = cml.create_model(create_model_params)
access_key = new_model_details["accessKey"]  # todo check
model_id = new_model_details["id"]

print("New model created with access key", access_key)

# Disable model_authentication
cml.set_model_auth({"id": model_id, "enableAuth": False})

# Wait for the model to deploy.
is_deployed = False
while is_deployed == False:
    model = cml.get_model({"id": str(
        new_model_details["id"]), "latestModelDeployment": True, "latestModelBuild":
    if model["latestModelDeployment"]["status"] == 'deployed':
        print("Model is deployed")
        break
    else:
        print("Deploying Model.....")
        time.sleep(10)

##########
##########
########## PART 5 #####
########## Good time to take a break
##########


# Change the line in the flask/single_view.html file.
subprocess.call(["sed", "-i",  's/const\saccessKey.*/const accessKey = "' +
```

```
   'nvidiaGPUs': 0},
  'description': 'Explain a given model prediction',
  'htmlUrl': 'https://ml-d09f3fad-5b2.cdptrial.g76q-bs1g.cloudera.site/
trial36_admin/telco_churn1/models/1',
  'id': '1',
  'name': 'Model Explainer 19012022090853',
  'namespace': 'mlx-user-1',
  'project': {'crn': 'crn:cdp:ml:us-west-1:a5c7fdfa-39bb-4604-a4f7-49fd
919becfd:workspace:8f8aaba8-ff43-4b31-bc3a-c7939926e69e/42879631-20a1-
4f19-bb30-2d291a7fe7d5',
   'id': 1,
   'name': 'Telco_churn1',
   'slug': 'telco_churn1'},
  'projectId': 1,
  'projectOwner': {'id': 1, 'type': 'user', 'username': 'trial36_admi
n'},
  'updatedAt': '2022-01-19T09:16:30.269Z',
  'visibility': 'private'}
```

Wait for the model to deploy.

```
> is_deployed = False
> while is_deployed == False:
      model = cml.get_model({"id": str(
          new_model_details["id"]), "latestModelDeployment": True, "lates
      if model["latestModelDeployment"]["status"] == 'deployed':
          print("Model is deployed")
          break
      else:
          print("Deploying Model.....")
          time.sleep(10)
```

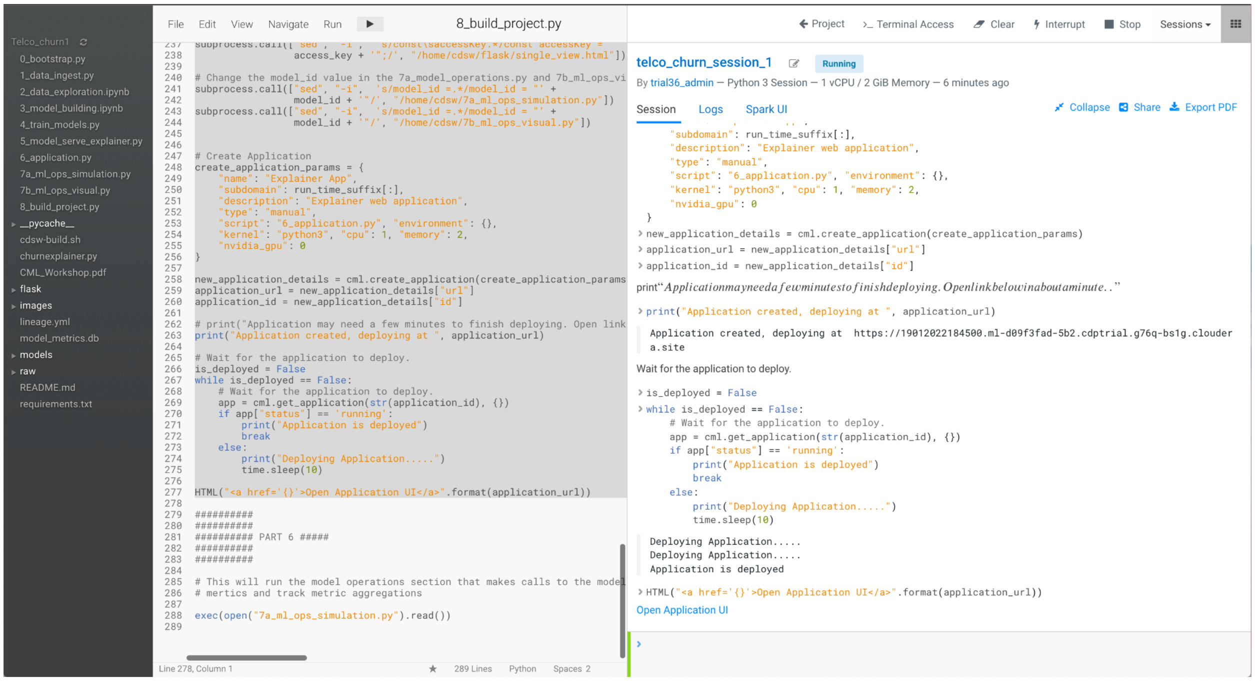
```
Deploying Model.....
Deploying Model.....
```

Part 5: Applications give data scientists a way to create ML web applications/dashboards and easily share them with other business stakeholders. Applications can range from single visualizations embedded in reports, to rich dashboard solutions such as Tableau. They can be interactive or non-interactive.

Applications stand alongside other existing forms of workloads in CML (sessions, jobs, experiments, models). Like all other workloads, applications must be created within the scope of a project. Each application is launched within its own isolated engine. Additionally, like models, engines launched for applications do not time out automatically. They will run as long as the web application needs to be accessible by any users and must be stopped manually when needed.

Select and highlight the contents of part 5, then Right click -> Run Line(s)

When deploying the CML Flask Application you will be provided with a URL to follow at the end of your session pane output. **Click to Open Application UI**



```
subprocess.call(["sed", "-i", 's/const\saccessKey.*/const accessKey = ' +
                 access_key + '";/', "/home/cdsw/flask/single_view.html"])

# Change the model_id value in the 7a_model_operations.py and 7b_ml_ops_vi
subprocess.call(["sed", "-i",  's/model_id =.*/model_id = "' +
                 model_id + '"/', "/home/cdsw/7a_ml_ops_simulation.py"])
subprocess.call(["sed", "-i",  's/model_id =.*/model_id = "' +
                 model_id + '"/', "/home/cdsw/7b_ml_ops_visual.py"])


# Create Application
create_application_params = {
    "name": "Explainer App",
    "subdomain": run_time_suffix[:],
    "description": "Explainer web application",
    "type": "manual",
    "script": "6_application.py", "environment": {},
    "kernel": "python3", "cpu": 1, "memory": 2,
    "nvidia_gpu": 0
}

new_application_details = cml.create_application(create_application_params
application_url = new_application_details["url"]
application_id = new_application_details["id"]

# print("Application may need a few minutes to finish deploying. Open link
print("Application created, deploying at ", application_url)

# Wait for the application to deploy.
is_deployed = False
while is_deployed == False:
    # Wait for the application to deploy.
    app = cml.get_application(str(application_id), {})
    if app["status"] == 'running':
        print("Application is deployed")
        break
    else:
        print("Deploying Application.....")
        time.sleep(10)

HTML("<a href='{}'>Open Application UI</a>".format(application_url))

##########
##########
########## PART 6 #####
##########
##########

# This will run the model operations section that makes calls to the model
# mertics and track metric aggregations

exec(open("7a_ml_ops_simulation.py").read())
```

telco_churn_session_1 Running

By trial36_admin — Python 3 Session — 1 vCPU / 2 GiB Memory — 6 minutes ago

```
    "subdomain": run_time_suffix[:],
    "description": "Explainer web application",
    "type": "manual",
    "script": "6_application.py", "environment": {},
    "kernel": "python3", "cpu": 1, "memory": 2,
    "nvidia_gpu": 0
 }
> new_application_details = cml.create_application(create_application_params)
> application_url = new_application_details["url"]
> application_id = new_application_details["id"]
```

print“*Applicationmayneedafewminutestofinishdeploying.Openlinkbelowinaboutaminute..*”

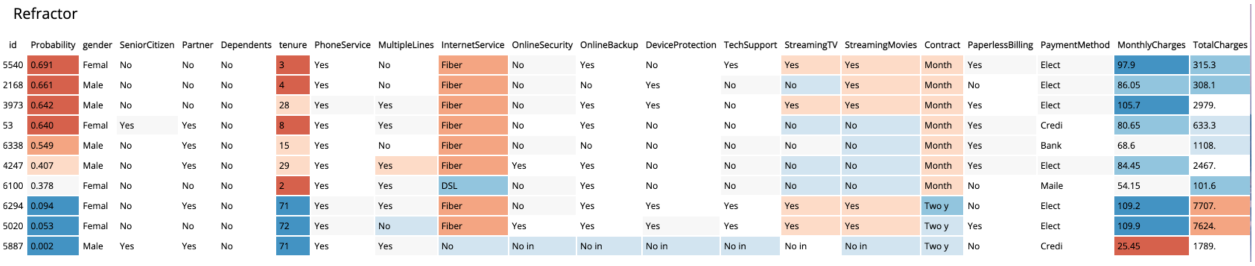
```
> print("Application created, deploying at ", application_url)

  Application created, deploying at  https://19012022184500.ml-d09f3fad-5b2.cdptrial.g76q-bs1g.clouder
  a.site
```

Wait for the application to deploy.

```
> is_deployed = False
> while is_deployed == False:
      # Wait for the application to deploy.
      app = cml.get_application(str(application_id), {})
      if app["status"] == 'running':
          print("Application is deployed")
          break
      else:
          print("Deploying Application.....")
          time.sleep(10)

  Deploying Application.....
  Deploying Application.....
  Application is deployed

> HTML("<a href='{}'>Open Application UI</a>".format(application_url))
```

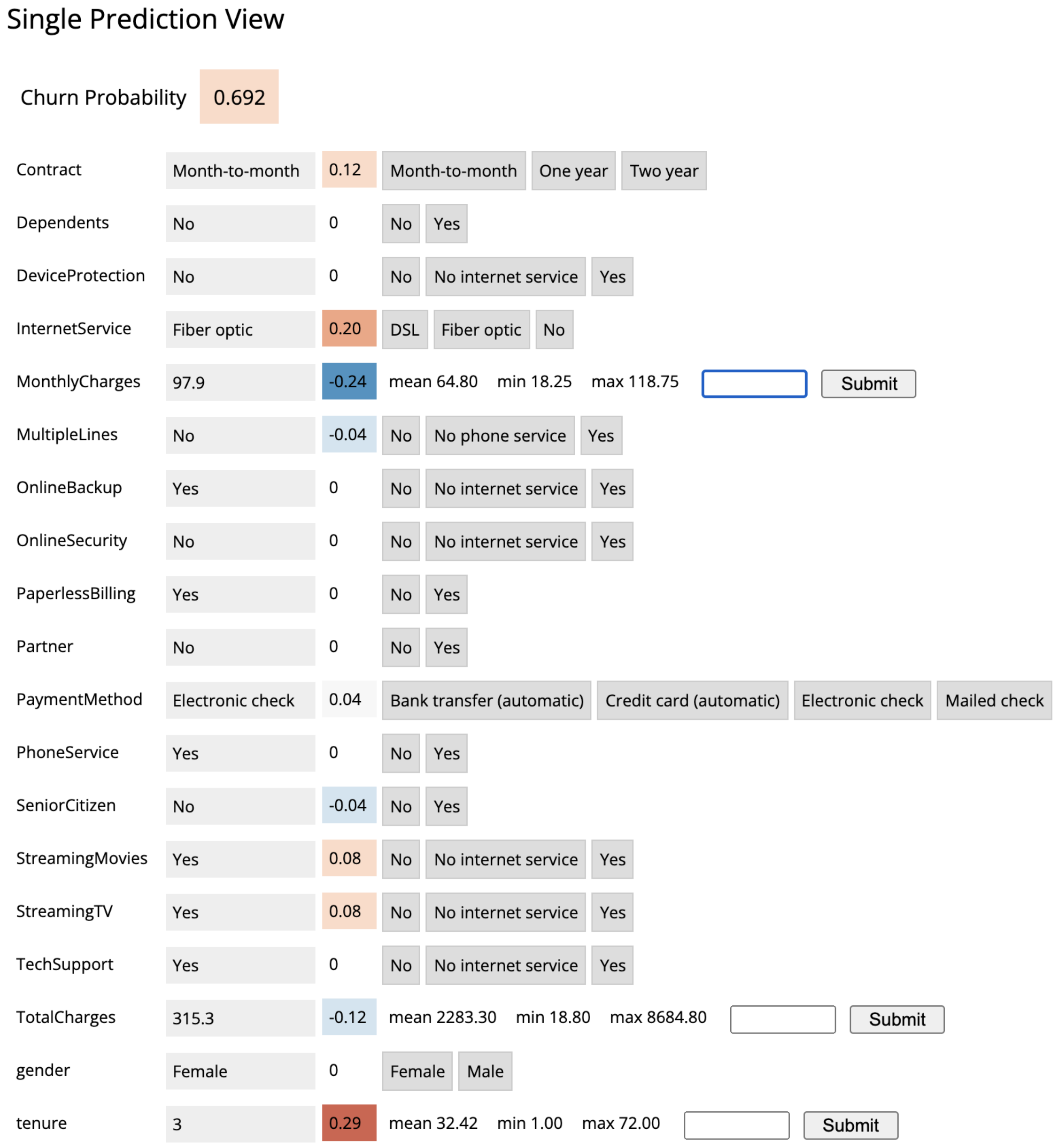
Open Application UI

Within the Flask Application UI you can experiment with some of the parameters that will run against the model to predict how likely a customer is to churn:



Refractor

id	Probability	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges
5540	0.691	Femal	No	No	No	3	Yes	No	Fiber	No	Yes	No	Yes	Yes	Yes	Month	Yes	Elect	97.9	315.3
2168	0.661	Male	No	No	No	4	Yes	No	Fiber	No	No	Yes	No	No	Yes	Month	No	Elect	86.05	308.1
3973	0.642	Male	No	No	No	28	Yes	Yes	Fiber	No	Yes	Yes	No	Yes	Yes	Month	Yes	Elect	105.7	2979.
53	0.640	Femal	Yes	Yes	No	8	Yes	Yes	Fiber	No	Yes	No	No	No	No	Month	Yes	Credi	80.65	633.3
6338	0.549	Male	No	Yes	No	15	Yes	No	Fiber	No	No	No	No	No	No	Month	Yes	Bank	68.6	1108.
4247	0.407	Male	No	Yes	No	29	Yes	Yes	Fiber	Yes	Yes	No	No	No	No	Month	Yes	Elect	84.45	2467.
6100	0.378	Femal	No	No	No	2	Yes	Yes	DSL	No	Yes	No	No	No	No	Month	No	Maile	54.15	101.6
6294	0.094	Femal	No	Yes	No	71	Yes	Yes	Fiber	No	Yes	Yes	Yes	Yes	Yes	Two y	No	Elect	109.2	7707.
5020	0.053	Femal	No	No	No	72	Yes	No	Fiber	Yes	Yes	Yes	Yes	Yes	Yes	Two y	Yes	Elect	109.9	7624.
5887	0.002	Male	Yes	Yes	No	71	Yes	Yes	No	No in	No in	No in	No in	No in	No in	Two y	No	Credi	25.45	1789.

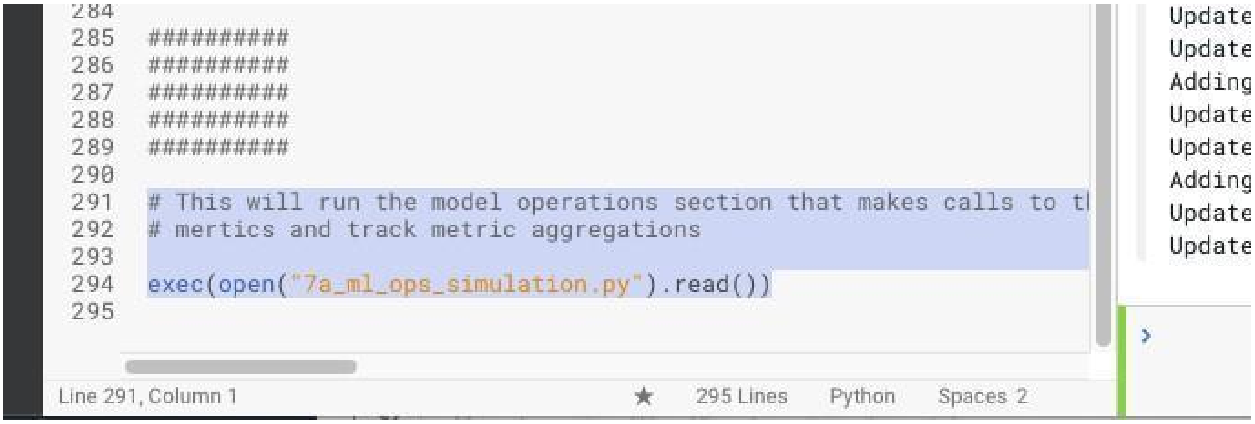


Single Prediction View

Churn Probability 0.692

Contract	Month-to-month	0.12	Month-to-month	One year	Two year		
Dependents	No	0	No	Yes			
DeviceProtection	No	0	No	No internet service	Yes		
InternetService	Fiber optic	0.20	DSL	Fiber optic	No		
MonthlyCharges	97.9	-0.24	mean 64.80	min 18.25	max 118.75		Submit
MultipleLines	No	-0.04	No	No phone service	Yes		
OnlineBackup	Yes	0	No	No internet service	Yes		
OnlineSecurity	No	0	No	No internet service	Yes		
PaperlessBilling	Yes	0	No	Yes			
Partner	No	0	No	Yes			
PaymentMethod	Electronic check	0.04	Bank transfer (automatic)	Credit card (automatic)	Electronic check	Mailed check	
PhoneService	Yes	0	No	Yes			
SeniorCitizen	No	-0.04	No	Yes			
StreamingMovies	Yes	0.08	No	No internet service	Yes		
StreamingTV	Yes	0.08	No	No internet service	Yes		
TechSupport	Yes	0	No	No internet service	Yes		
TotalCharges	315.3	-0.12	mean 2283.30	min 18.80	max 8684.80		Submit
gender	Female	0	Female	Male			
tenure	3	0.29	mean 32.42	min 1.00	max 72.00		Submit

Part 6: Run the last code snippet in the workbench to complete the project build script (8_build_project.py). This goes through a process of simulating a model that drifts over 1000 calls to the model. The file contains comments with details of how this is done. **Select and highlight the contents of part 6, then Right click -> Run Line(s)**

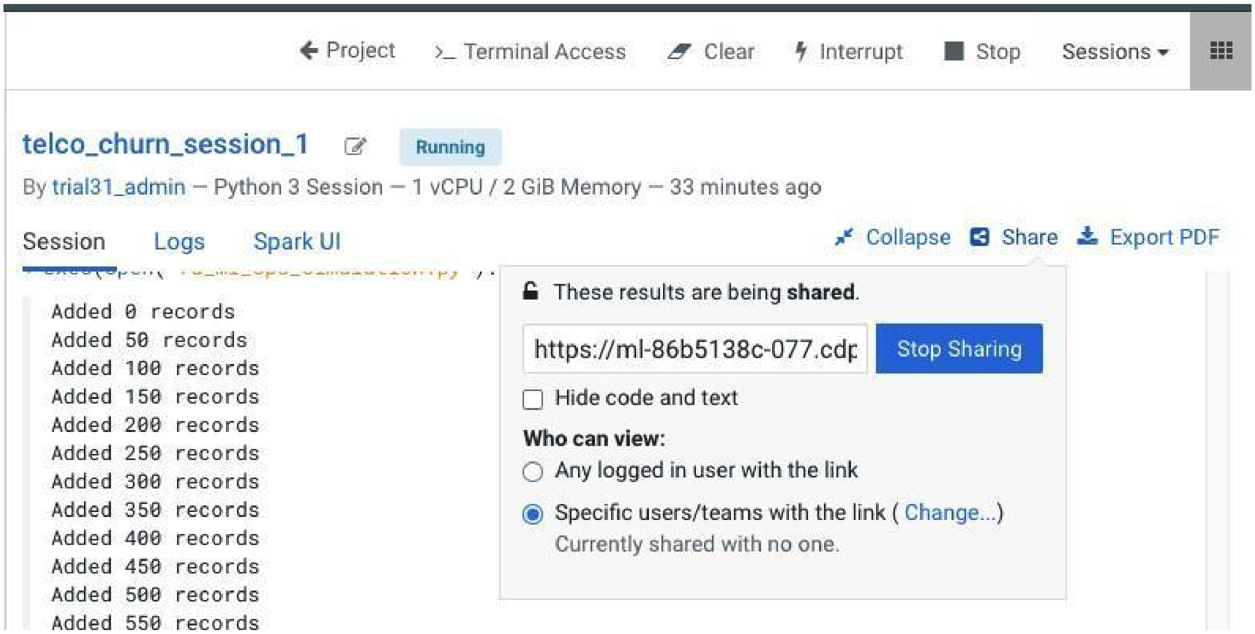


```
##########
##########
##########
##########
##########

# This will run the model operations section that makes calls to t
# mertics and track metric aggregations

exec(open("7a_ml_ops_simulation.py").read())

```

```
Update
Update
Adding
Update
Update
Adding
Update
Update
```

You can also share the workbench session with other users if they would like to view results of the code. A URL can be shared out if users outside of CML would like to view code/results of work.

Click on share at the top of the Session pane



telco_churn_session_1 Running

By trial31_admin — Python 3 Session — 1 vCPU / 2 GiB Memory — 33 minutes ago

```
Added 0 records
Added 50 records
Added 100 records
Added 150 records
Added 200 records
Added 250 records
Added 300 records
Added 350 records
Added 400 records
Added 450 records
Added 500 records
Added 550 records
```

These results are being **shared**.

https://ml-86b5138c-077.cdp Stop Sharing

☐ Hide code and text

Who can view:

○ Any logged in user with the link

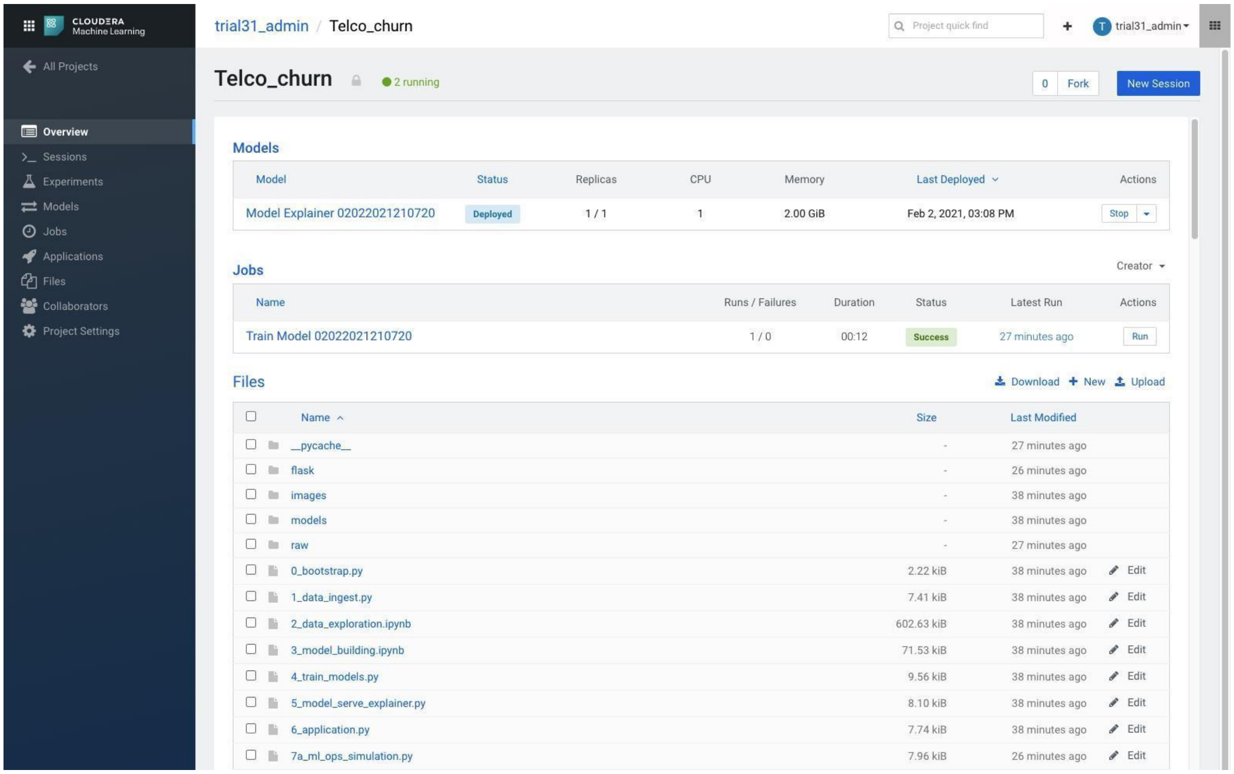
● Specific users/teams with the link (Change...)
Currently shared with no one.

A full CML project should now be running with a Job, Model, and Application deployed! Go back to the project and take a look at the Model and Applications page.

Going to the **Model** page will show you the deployed model. Clicking on the Model will give various tabs of monitoring and statistics in addition to previous deployments of the same model.

On the **Applications** page we can see our running flask app we looked at previously.

Models and Applications will continue to run even if the workbench session is stopped or timeout occurs. These will run on engines as part of the CML workspace



trial31_admin / Telco_churn

Telco_churn 2 running

Models

Model	Status	Replicas	CPU	Memory	Last Deployed	Actions
Model Explainer 02022021210720	Deployed	1 / 1	1	2.00 GiB	Feb 2, 2021, 03:08 PM	Stop

Jobs

Name	Runs / Failures	Duration	Status	Latest Run	Actions
Train Model 02022021210720	1 / 0	00:12	Success	27 minutes ago	Run

Files

☐	Name	Size	Last Modified	
☐	__pycache__	-	27 minutes ago	
☐	flask	-	26 minutes ago	
☐	images	-	38 minutes ago	
☐	models	-	38 minutes ago	
☐	raw	-	27 minutes ago	
☐	0_bootstrap.py	2.22 kiB	38 minutes ago	Edit
☐	1_data_ingest.py	7.41 kiB	38 minutes ago	Edit
☐	2_data_exploration.ipynb	602.63 kiB	38 minutes ago	Edit
☐	3_model_building.ipynb	71.53 kiB	38 minutes ago	Edit
☐	4_train_models.py	9.56 kiB	38 minutes ago	Edit
☐	5_model_serve_explainer.py	8.10 kiB	38 minutes ago	Edit
☐	6_application.py	7.74 kiB	38 minutes ago	Edit
☐	7a_ml_ops_simulation.py	7.96 kiB	26 minutes ago	Edit