

Cloudera AI Top Tasks (on premises) Map

Date published: 2023-05-24

Date modified: 2024-12-10

CLOUdera

Legal Notice

© Cloudera Inc. 2025. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Creating and deploying a model.....	4
Registering a model using the AI Registry user interface.....	7
Deploying a model from the AI Registry page.....	8

Creating and deploying a model

Using Cloudera AI, you can create any function within a script and deploy it to a REST API. In a Cloudera AI project, this is typically a predict function that accepts an input and returns a prediction based on the model's parameters.

As an example, we create a function that adds two numbers, and deploy it as a model that returns the sum of the numbers. This function will accept two numbers in JSON format as input and return the sum.



Note: In case of PBJ (Powered by Jupyter) Runtime, a specific decorator, the `cml_model` decorator is needed to create a model. This decorator allows a function to work as a model in a PBJ Runtime. The decorator can also be used to enable gathering of model metrics. For more details, see [Example models with PBJ Runtimes](#).

For Cloudera AI UI

1. Create a new project. Note that models are always created within the context of a project.
2. Click New Session and launch a new Python 3 session.
3. Create a new file within the project called `add_numbers.py`. This is the file where we define the function that will be called when the model is run. For example:

`add_numbers.py`

```
import cml.models_v1 as models

@models.cml_model
def add(args):
    result = args["a"] + args["b"]
    return result
```



Note: In practice, do not assume that users calling the model will provide input in the correct format or enter good values. Always perform input validation.

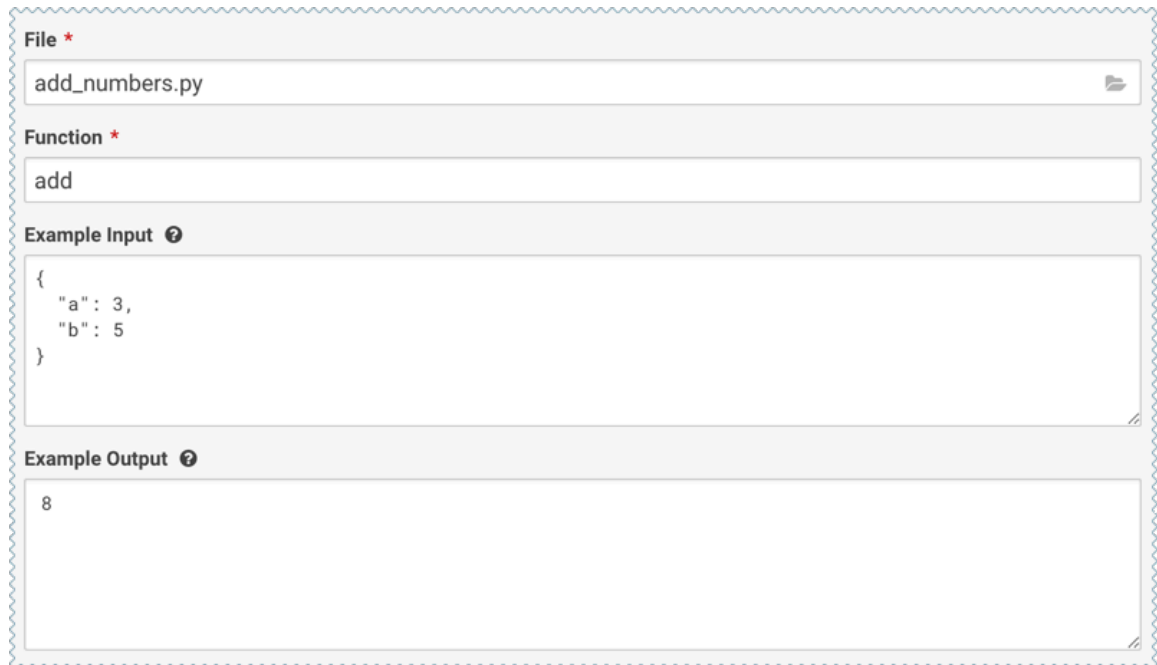
4. Before deploying the model, test it by running the `add_numbers.py` script, and then by calling the `add` function directly from the interactive workbench session. For example:

```
add({"a": 3, "b": 5})
```

The screenshot displays the Cloudera AI interface. On the left, a code editor shows the `add` function defined in `add_numbers.py`. On the right, a Jupyter session titled 'Untitled Session' is running. The session output shows two successful calls to the `add` function: `> add({'a': 3, 'b': 5})` resulting in `8`, and `> add({'a': 4, 'b': 7})` resulting in `11`. The session status is 'Running'.

5. Deploy the add function to a REST endpoint.

- a. Go to the project Overview page.
- b. Click Model Deployments New Model .
- c. Give the model a Name and Description.
- d. In the **General** section of the page, under the Deploy Model as option, you can select the Service Account option, if the model is to be deployed in a service account, and choose the account from the dropdown menu.
- e. Enter the details about the model that you want to build, for example:
 - File: add_numbers.py
 - Function: add
 - Example Input: {"a": 3, "b": 5}
 - Example Output: 8



The screenshot shows the 'New Model' form in the Google Cloud AI Platform console, specifically the 'General' section. The form is titled 'File *' and contains the following fields:

- File ***: A text input field containing 'add_numbers.py'.
- Function ***: A text input field containing 'add'.
- Example Input ?**: A text area containing a JSON object:

```
{  "a": 3,  "b": 5}
```
- Example Output ?**: A text area containing the number '8'.

- f. Select the resources needed to run this model, including any replicas for load balancing. To specify the maximum number of replicas in a model deployment, go to Site Administration Settings Models . The default is 9 replicas, and up to 199 can be set.
- g. Click Deploy Model.

6. Select the model to go on its Overview page. Click Builds to track realtime progress as the model is built and deployed. This process essentially creates a Docker container where the model will live and serve requests.

Add Two Numbers Building Stop Deploy New Build

Overview Deployments **Builds** Monitoring Settings

Build	Status	File	Function	Kernel	Engine Image	Created By	Created At	Comment	Actions
1	Building	add_numbers.py	add	python3	Base Image v	ambreen	Jun 5, 2018, 5:44 PM	Initial revision.	Delete

```

Sending build context to Docker daemon 15.05 MB

Step 1/16 : FROM docker.repository.cloudera.com/cdsw/engine:
----> f8955770daa1
Step 2/16 : ENTRYPOINT node /app/model-runtime/model-server.js
----> Running in 58038f1e58d5

```

7. Once the model has been deployed, go back to the model Overview page and use the Test Model widget to make sure the model works as expected.

If you entered example input when creating the model, the Input field will be pre-populated with those values. Click Test. The result returned includes the output response from the model, as well as the ID of the replica that served the request.

Model response times depend largely on your model code. That is, how long it takes the model function to perform the computation needed to return a prediction. Note that model replicas can only process one request at a time. Concurrent requests will be queued until the model can process them.

For Cloudera AI APIv2

Create and deploy a model using the Models API as instructed in this example:

This example demonstrates the use of the Models API. Run this example:

1. Create a project with the Python template and a legacy engine.
2. Start a session.
3. Run `!pip3 install sklearn`.
4. Run `fit.py`.

The example script first obtains the project ID, then creates and deploys a model.

```

projects = client.list_projects(search_filter=json.dumps({"name": "<your
project name>"}))
project = projects.projects[0] # assuming only one project is returned by
the above query
model_body = cmlapi.CreateModelRequest(project_id=project.id, name="Demo
Model", description="A simple model")
model = client.create_model(model_body, project.id)
model_build_body = cmlapi.CreateModelBuildRequest(project_id=project.id,
model_id=model.id, file_path="predict.py", function_name="predict", ker
nel="python3")
model_build = client.create_model_build(model_build_body, project.id, mod
el.id)
while model_build.status not in ["built", "build failed"]:
    print("waiting for model to build...")
    time.sleep(10)
    model_build = client.get_model_build(project.id, model.id, model_build
.id)
if model_build.status == "build failed":

```

```

print("model build failed, see UI for more information")
sys.exit(1)
print("model built successfully!")
model_deployment_body = cmlapi.CreateModelDeploymentRequest(project_id=p
project.id, model_id=model.id, build_id=model_build.id)
model_deployment = client.create_model_deployment(model_deployment_body,
project.id, model.id, build.id)
while model_deployment.status not in ["stopped", "failed", "deployed"]:
    print("waiting for model to deploy...")
    time.sleep(10)
model_deployment = client.get_model_deployment(project.id, model.id, m
odel_build.id, model_deployment.id)
if model_deployment.status != "deployed":
    print("model deployment failed, see UI for more information")
    sys.exit(1)
print("model deployed successfully!")

```

Registering a model using the AI Registry user interface

You can register a model using the **AI Registry** user interface or the MLFlow SDK.

Using the AI Registry user interface to register a model

Registering a model enables you to track your model and upload and share the model. Registering a model stores the model archives in the Cloudera AI Registry with a version tag. The first time you register a model, **AI Registry** automatically creates a model repository with the first version of the model.

Before you begin

You must have permission to access a project in which the model is created before you can register the model.

Procedure

1. Click Projects in the left navigation pane to display the Projects page.
2. Select the project that contains the model that you want to register.
AI Registry displays all of the models under the specific project along with their source, deployment status, replicas, memory and a drop-down function for actions that can be made pertaining to that model for deployment.
3. Click the Experiments tab in the left navigation pane and select the experiment that contains the model you want to register.
4. Select the model you want to register.
 Cloudera AI displays the Experiment Run Details page.

admin / test1 / Experiments / registermodeltest / Run

Project quick find

Metrics

Name	Value
rmse	0.7931640229276851
r2	0.10862644997792614
mae	0.6271946374319586

Parameters

Name	Value
alpha	0.5
l1_ratio	0.5

Tags

Name	Value	Actions
No Data		

Add Tags

Enter Key Enter Value [Add](#)

Artifacts

- model
 - requirements.txt
 - conda.yaml
 - model.pkl
 - python_env.yaml
 - MLmodel

Make Predictions [Register Model](#)

Predict on a Spark DataFrame:

```
import mlflow

logged_model = '/home/cdsw/.experiments/rkms-xkz8-144a-hv5o/6j6o-k1bf-77ct-6fs4/artifacts/model'

# Load model as a Spark UDF.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model)

# Predict on a Spark DataFrame.
df.withColumn('predictions', loaded_model(*column_names)).collect()
```

[Copy Code](#)

Predict on a Pandas DataFrame:

```
import mlflow
```

[Copy Code](#)

5. Select the run that contains the model you want to register.
6. Select Register Model to begin the registration process.
AI Registry displays the Registry Model dialog box.
7. Enter the name of your registered model.
 You can also enter optional information for the description, version notes, and version tags.
8. Click OK to complete the registration.

Deploying a model from the AI Registry page

You can deploy a model once or more times to create different versions of the model. You can also deploy a model you created in one workbench to a different workbench.

Procedure

1. Select **AI Registry** from the left navigation pane.
2. Select the model you want to deploy.
AI Registry displays the Model Version List page.
3. Select the model version you want to deploy.
AI Registry displays a side window that lists the version information. Dismiss this window to proceed.
4. Under the Actions menu, click Deploy.
5. Select the project you want to deploy to in the dialog box and click Go.
You can select either the project the model is located in or another project to deploy the model to.
AI Registry displays the Deploy a Model page with the detailed model information auto populated.

a / t-3 / Model Deployments / Deploy Model Project quick find admin

Deploy a Model

Deployment Template

☐ Deploy model from code
☒ Deploy registered model

General

* Registered Model
ElasticnetWineModel

* Model Version
Version 1

☒ Enable Authentication
Enforces model API requests to be authenticated with an API key.

Build

Example Input

```
{ "param": "value" }
```

Example Output

```
{ "result": "value" }
```

Runtime

Editor	Kernel	Edition	Version
Workbench	Python 3.7	Standard	2022.04

Configure additional runtime options in [Project Settings](#).

Workspace: [modelregistrytest2](#)

Cloud Provider: (Openshift)

6. If you enable authentication and you have deployed the model to a shared project, you will need to enter an API key to be able to access and use the model.
7. Click OK.