H2O.ai

DOCUMENTATION

For questions, please contact **support@h2o.ai**

Contents

What is H2O Document AI?

H2O Document AI is an H2O AI Cloud (HAIC) engine that lets you build accurate AI models that:

- Classify documents
- Extract text, tables, and images from documents
- Group, label, and refine extracted information from documents

H2O Document AI supports various documents and use cases to help organizations understand, process, and manage large amounts of unstructured data. Upload your documents to H2O Document AI using the H2O Document AI web interface (in HAIC) or API. H2O Document AI allows you to handle a wide variety of documents, including:

- Image scans (faxes in PDF or other formats, pictures with text, and non-editable forms)
- Documents with embedded text that have text and layout metadata (PDF docs, Word docs, HTML pages)
- Documents with regular text "left to right/top to bottom" (CSVs, emails, editable forms)

H2O Document AI uses a combination of:

- Intelligent Character Recognition (**ICR**), which leverages learning algorithms for generalizable character and word recognition,
- Document layout understanding, and
- Natural Language Processing (**NLP**) to develop highly accurate models rapidly.

Benefits of H2O Document AI

H2O Document AI is designed to help organizations automate document processes and find insights throughout their large volume of documents. As a result, H2O Document AI:

- Frees up teams to do higher-value work activities
- Provides relief to users, analysts, and managers by increasing efficiency while reducing redundancies
- Allows the enterprise to go beyond OCR-based template methods and RPA-based memorization efforts, which are not scalable as the variety and volume of documents changes
- Enables organizations to focus on quicker time to value primary users and secondary consumers by focusing on developing applications and more rapid integrations using highly accurate extracted information and knowledge (as opposed to updating rules, re-automating template management, and moving documents)

Capabilities

Automatically learns and trains models

H2O Document AI combines Intelligent Character Recognition (ICR) with Natural Language Processing (NLP) and layout intelligence to automatically learn and train models.

Generates highly accurate results fast

H2O Document AI generates highly accurate results fast. Designed with the experience of top Kaggle grandmasters and a diverse set of customers, H2O Document AI uses a combination of intelligent character recognition (ICR) and natural language processing (NLP) as well as optical character recognition (OCR).

Integrates with existing applications and workflows

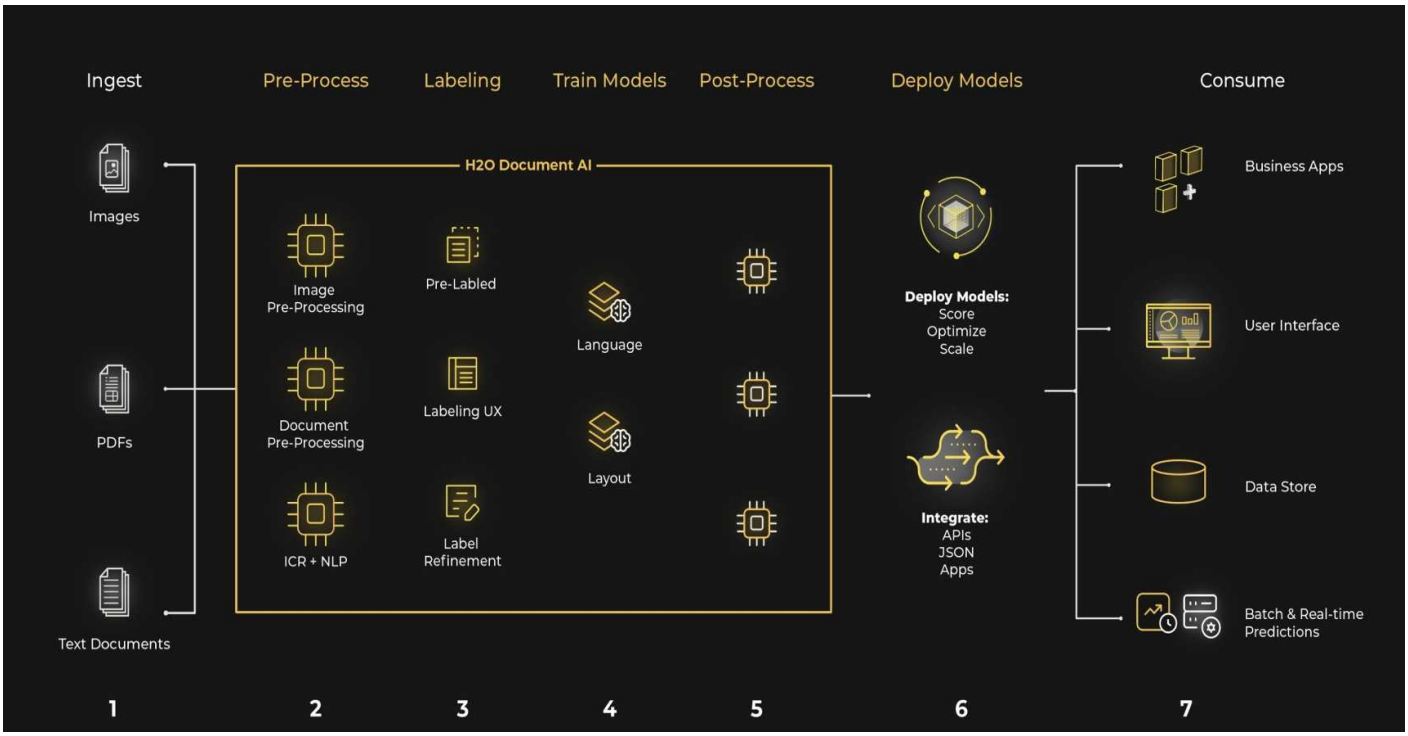
With the H2O Document AI REST API, H2O Document AI easily integrates with existing applications and workflows. In particular, the REST API can process documents and train and score models. Scoring outputs from H2O Document AI are flat JSON files you can access via the REST API.

Integrates with H2O Wave for custom H2O Document AI applications

With the H2O Document AI REST API and H2O Wave, you can develop rapid AI applications to deploy built models in H2O Document AI.

Model flow

The flow of an H2O Document AI model from creation to deployment and consumption can be summarized in the following sequential steps (discussed in the below sections):



Step 1: Ingest

Upload your documents to H2O Document AI using the Document AI web interface or API. H2O Document AI lets you handle a wide variety of documents, including:

- Image scans (faxes in PDF or other formats, pictures with text, and non-editable forms)
- Documents with embedded text which have text and layout metadata (PDF docs, Word docs, HTML pages)
- Documents with regular text “left to right/top to bottom” (CSVs, emails, editable forms)

Step 2: Pre-process

Pre-process documents before training with a set of state-of-the-art computer vision and NLP product features. Pre-processing includes support for:

- Recognizing and handling embedded text
- Recognizing and handling logos
- Page orientation resolution
- Deskewing
- Cropping
- Text formatting optimization
- Color binarization
- Addressing input PDF quality challenges

Step 3: Labeling

Add, improve, and validate document labels:

- Integrates with common label formats
- Provides advanced options for validating labels against scored documents and determining labeling sufficiency

Step 4: Train models

Select the training data set within H2O Document AI, and it will automatically learn the document and create models.

- Language understanding and layout recognition using learning based on deep learning, transformer architectures, and machine learning
- AI-ML engine that uses multiple computer vision and NLP algorithms for diverse AI tasks
  - Entity recognition
  - Document and page classification
  - Form understanding
  - Grouping & set identification

Step 5: Post-process

Post-process to ensure consistency, accuracy, and organization of scored documents. H2O Document AI lets you perform a range of customized post-processing jobs that use AI algorithms vs. rules to ensure high-quality predictions and insights.

- Organizing prediction sets
- Confidence and probability measures
- Datatype standardization – date, times, currency codes, international numerical formats, locations

Step 6: Deploy models

Publish models into your cloud or on-premises environment of choice. Integrate models into existing systems, processes, and applications via APIs or JSON documents.

Step 7: Consume

After deploying your models, you can:

- Consume the model through business apps
- Store data
- Batch score in real-time to obtain predictions

Recommended workflow

The following is the recommended workflow for beginner H2O Document AI users.

Step 1: Ingest

When you begin H2O Document AI, you start with a blank Project repository. Begin by creating a project and importing the files you want to work with. Once the files have been imported, they are accessible from the Document sets page, and a blank annotation set appears on the Annotation sets page.

Step 2: Label

From the Document sets page, run OCR on the document set to retrieve the tokens for the files. The completed OCR file appears on the Annotation sets page.

On the Annotation sets page, begin annotating your blank document set by using edit in page view. You can apply regional attributes (labels) to your created bounding boxes and/or file attributes (classes) to each individual file within your set of documents.

Once you've finished annotating your files, combine the information you derived from OCR (that is, the tokens) and the annotations you've just created by running the apply labels job.

Step 3: Train models

After you've retrieved your fully labeled tokens, you can build your model. Select your labeled annotation set and run the Train Model job. If you want to classify your document pages, run a **Page Classification** model (requires *class* and *text* attributes). If you want the tokens of your images labeled, run a **Token Labeling** model (requires *label* and *text* attributes). Note that this can take some time to build. The finished model appears on the Models page.

Note: You can build a model with or without using a validation set.

Step 4: Deployment and post-processing

You can now use this model to predict on other data or publish a scoring pipeline to feed new documents into.

Use cases

H2O Document AI enables you to solve many use cases around unstructured data that:

- Vary greatly in language and layout
- Come in inconsistent or irregular formats
- Come from multiple internal and external sources
- Are digitally born, physical print, or scan/fax
- Have rigidly defined standards or contain free text

Example use case: University of San Francisco (UCSF) Health

The University of San Francisco (UCSF) Health is one of the top 10 hospitals in the United States and ranked #1 in neurology & neurosurgery.

UCSF Health struggled with numerous document processes, such as medical referrals, that led to wasted time and poor customer experiences.

With H2O Document AI, UCSF Health has fully automated multiple processes for multiple documents. As a result, higher levels of efficiency have been reported while customer experiences have improved.

"When we started this journey, we were hopeful that information extraction from semi-structured documents was possible, but we weren't sure. Some in the industry told us it couldn't be done. Now that the UCSF-H2O.ai collaboration team has delivered, it opens up many possibilities." - Bob Rogers, Expert in Residence for AI, UCSF

Business domains

You can use H2O Document AI in various industries, such as finance, insurance, transportation, healthcare, life sciences, telecom, manufacturing, service providers, legal, government, retail, energy, utilities, etc. For example:

Finance	Insurance	Transportation	Healthcare and life sciences
Bank documents	Customer engagement	Air Waybil (AWB)	Medical Records
Bank confirmations	Policies, claims, & contracts	Contracts	School forms
Municipal bonds	Benefits administration	Regulatory forms	Durable medical equipment (DME) requests
LIBOR regulations	Special handling forms		Progress notes
Marketing documents			Claim forms

Tutorials overview

Here is the collection of available H2O Document AI tutorials.

H2O Document AI - Publisher

Learning path

```
graph LR;
    User[User] --> A1[Tutorial 1A: Introduction to H2O Document AI - Publisher];
    User[User] --> B1[Tutorial 1B: Creating an evaluation model in H2O Document AI - Publisher];
    %% Apply custom color to nodes
    style User fill:#FEC925;
    style A1 fill:#FEC925;
    style B1 fill:#FEC925;
    %% Add links to each node
    click A1 "/h2o-document-ai/tutorials/publisher/basic-publisher/tutorial" "Go to Tutorial 1A"
    click B1 "/h2o-document-ai/tutorials/publisher/eval-model/eval-tutorial" "Go to Tutorial 1B"
```

- Tutorial 1A: Introduction to H2O Document AI - Publisher > This tutorial will walk you through the basics of H2O Document AI - Publisher.
- Tutorial 1B: Creating an evaluation model in H2O Document AI - Publisher > This tutorial will walk you through how to create a model using an evaluation annotation set so that you can access accuracy metrics from a trained model.

H2O Document AI - Viewer

There are no available tutorials for H2O Document AI - Viewer. Refer to the guide for H2O Document AI - Viewer to learn the basic workflow of the application.

Tutorial 1A: Introduction to H2O Document AI - Publisher

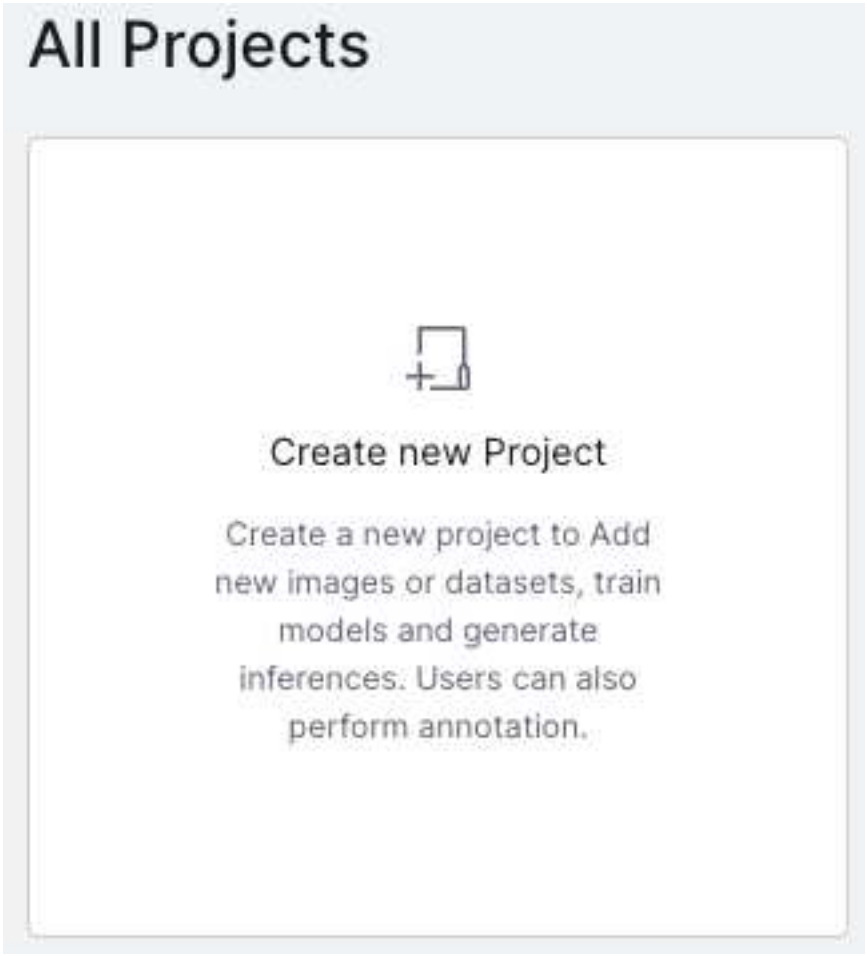
This tutorial will walk you through the basics of H2O Document AI - Publisher and will follow the recommended beginner workflow. For this tutorial, we will be building a Token Labeling model (which focuses on labeling your files' tokens).

Prerequisites

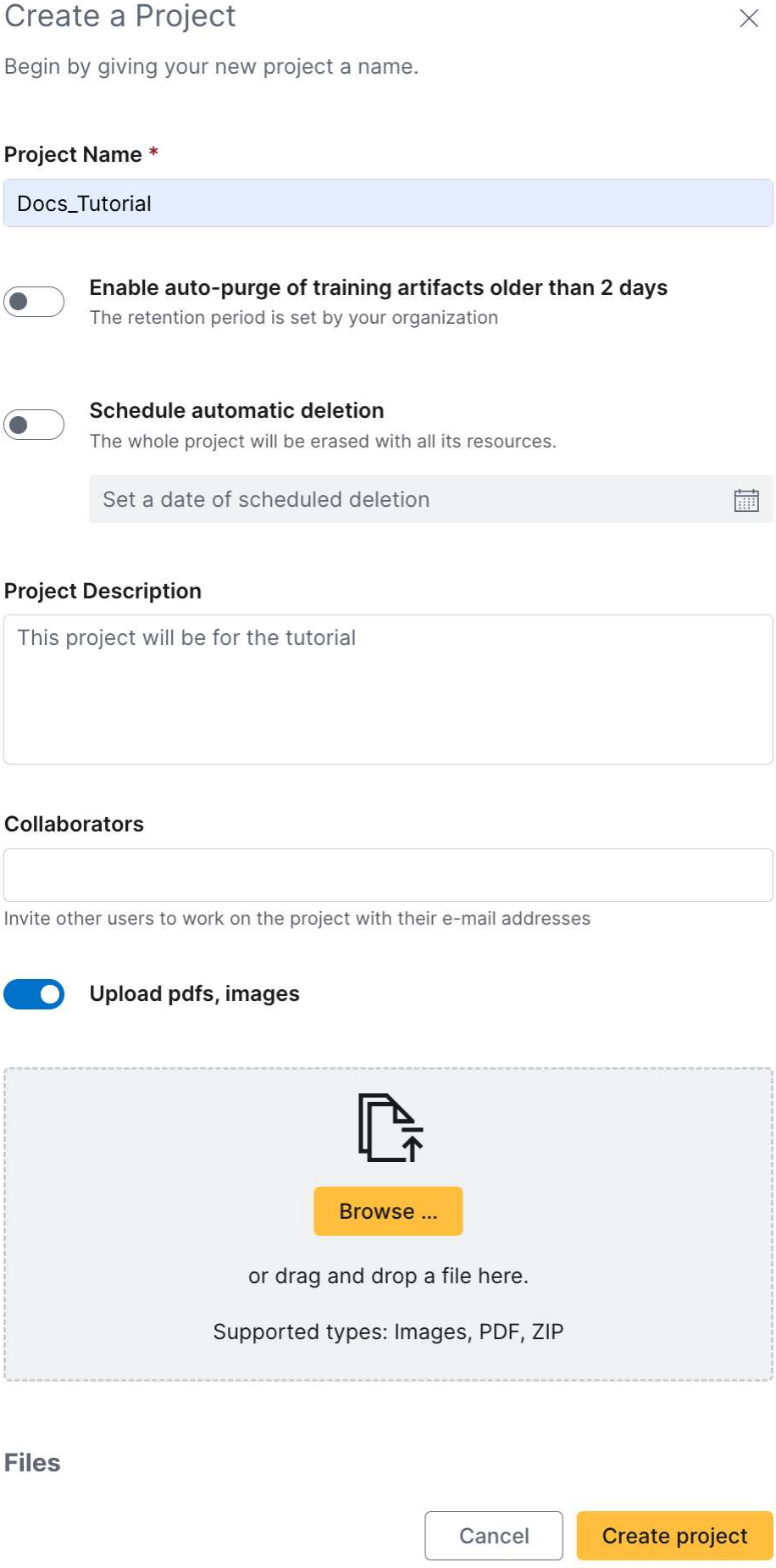
- A copy of this zip file containing medical referral documents that you will be using in this tutorial
- A copy of this medical referral file to submit to your published pipeline
- An understanding of the recommended beginner workflow page

Step 1: Creating the project

Start by creating a new project. Click the **"Create new Project"** button on H2O Document AI - Publisher's homepage.

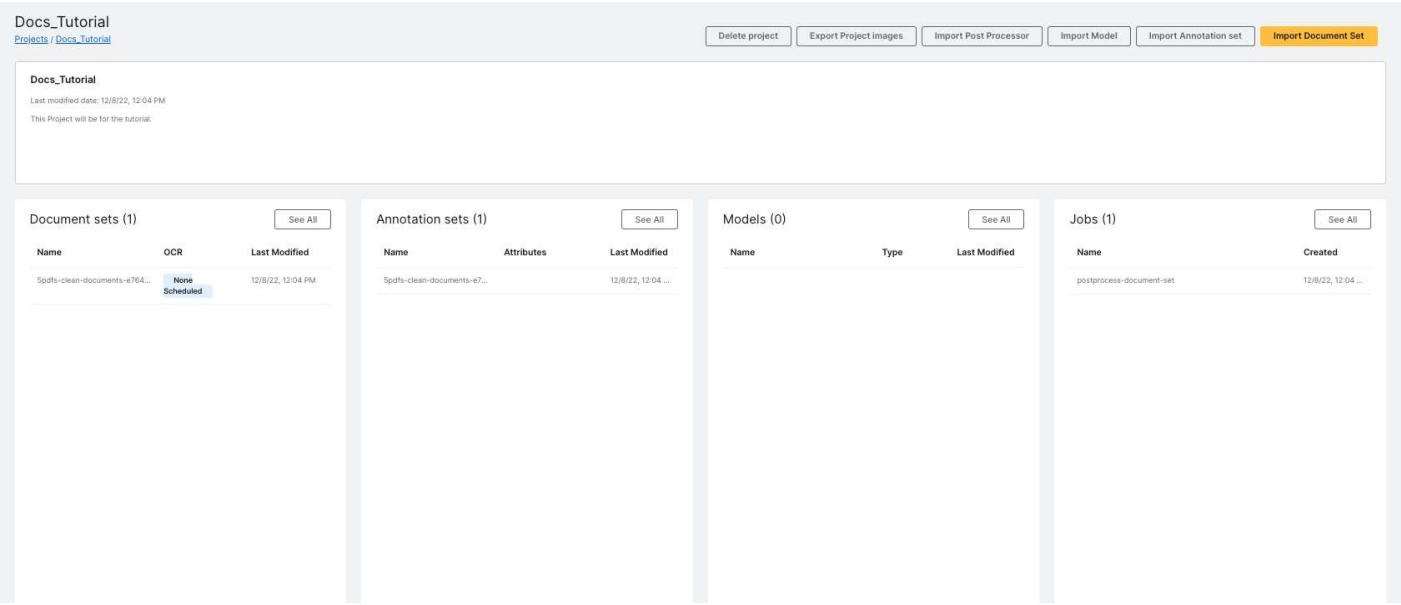


This takes you to the Create a Project panel. 1. Provide `Docs_Tutorial` as your project name 2. (Optional) Provide `This Project will be for the tutorial.` as the description 3. Add the zip file `5pdfs-clean-documents.zip` which contains the set of documents, PDFs, and images you will work with 4. Click **Create project**.



Project main page

After creating your new project, you will be taken to your project's main page. Your project will now also be accessible from the left navigation bar. From your project's main page, you can see each available page: Document sets, Annotation sets, Models, and Jobs. Published Pipelines is only accessible from the left navigation bar.



You can access these pages in two ways:

- clicking the page name on the left navigation bar
- clicking "See All" on the desired panel from the project's main page

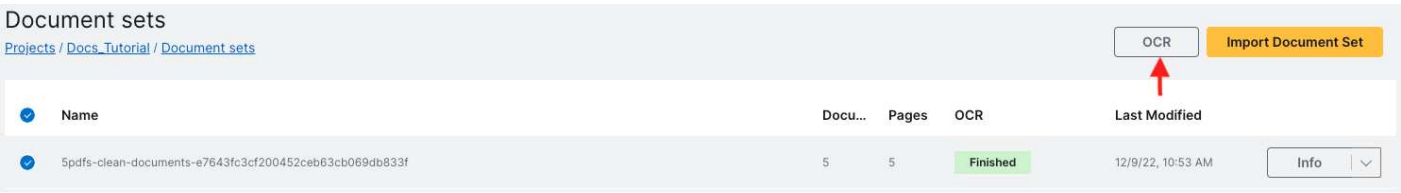
Your uploaded files will appear on the Document sets page and the Annotation sets page. You can see these files from the project's main page or by going directly into the Document sets or Annotation sets page.

Step 2: Running OCR

Navigate to the Document sets page. You will now run optimal character recognition (OCR) on your files. This lets you extract the tokens from your files.

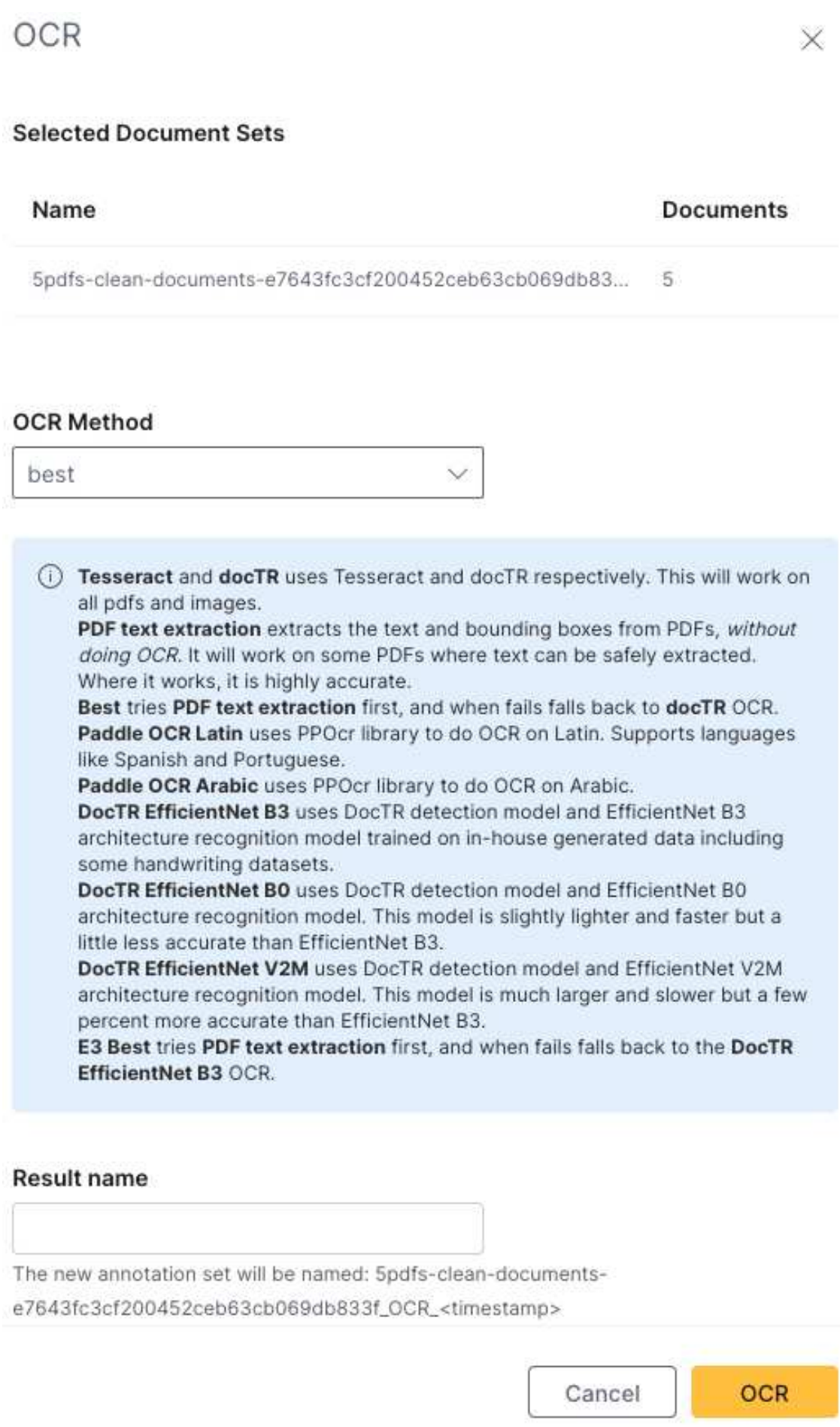
Click on the checkbox next to the files you uploaded. Selecting this checkbox will make the **OCR** button available in the upper navigation bar.

Click **OCR**.



On the OCR launch page:

1. Select `best` for the OCR method
2. Provide `Tutorial_OCR` as the Result name
3. (Optional) Provide `Running OCR on the 5 imported PDFs.` as the description
4. Click **OCR**

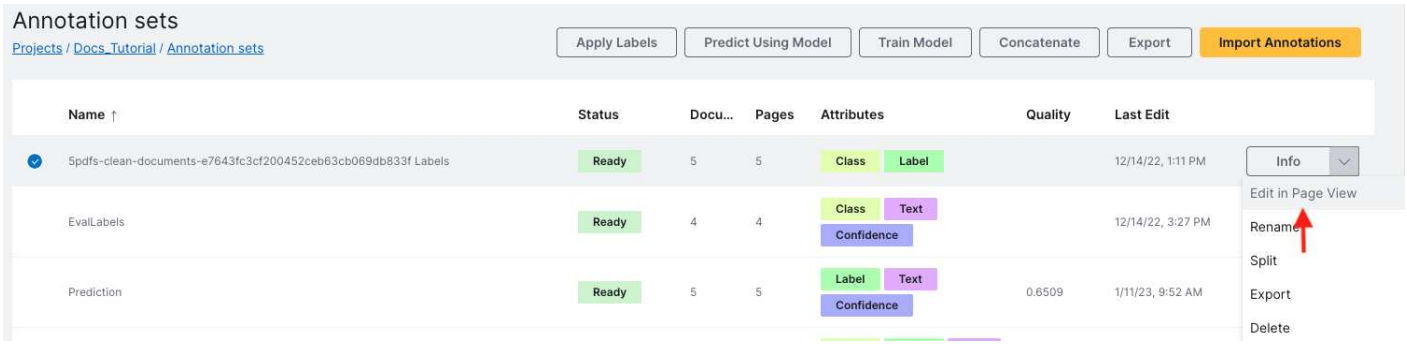


When your OCR job is finished running, you can access your tokens from the Annotation sets folder. It will have the attributes *text* and *confidence* attached to its file.

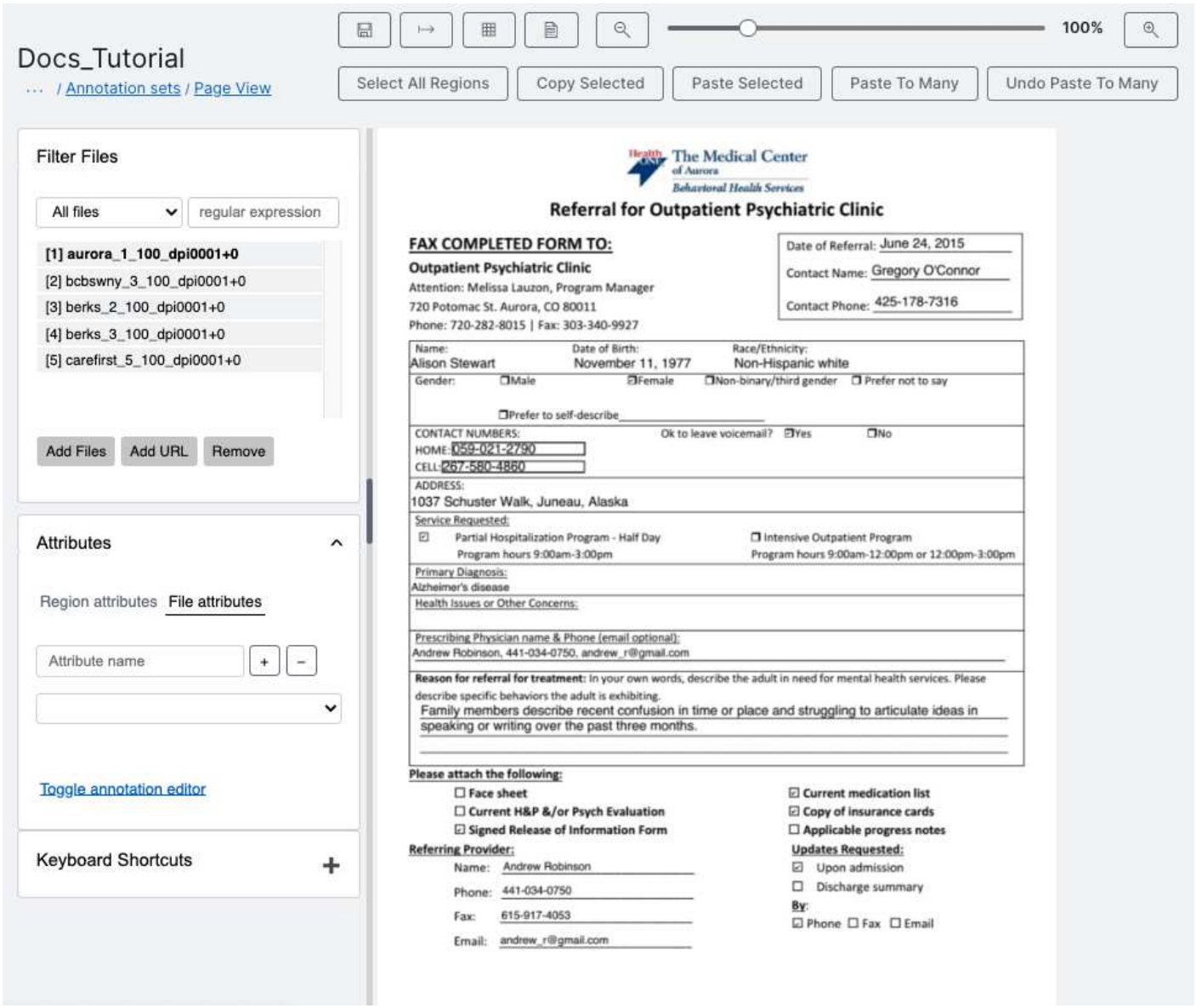
Step 3: Annotating your files

Navigate to the Annotation sets page. Your two annotation sets should be the OCR tokens set you just created and the blank labeling set that got uploaded to this page when you created your project.

Find your blank annotation set. Go to the end of its row and click the drop-down arrow next to the **Info** button. Then, click **Edit in Page View** from the drop-down options.



This will take you to the annotations page. Here, you can provide labeled regions to your files and classify what type of page each file in your set is.

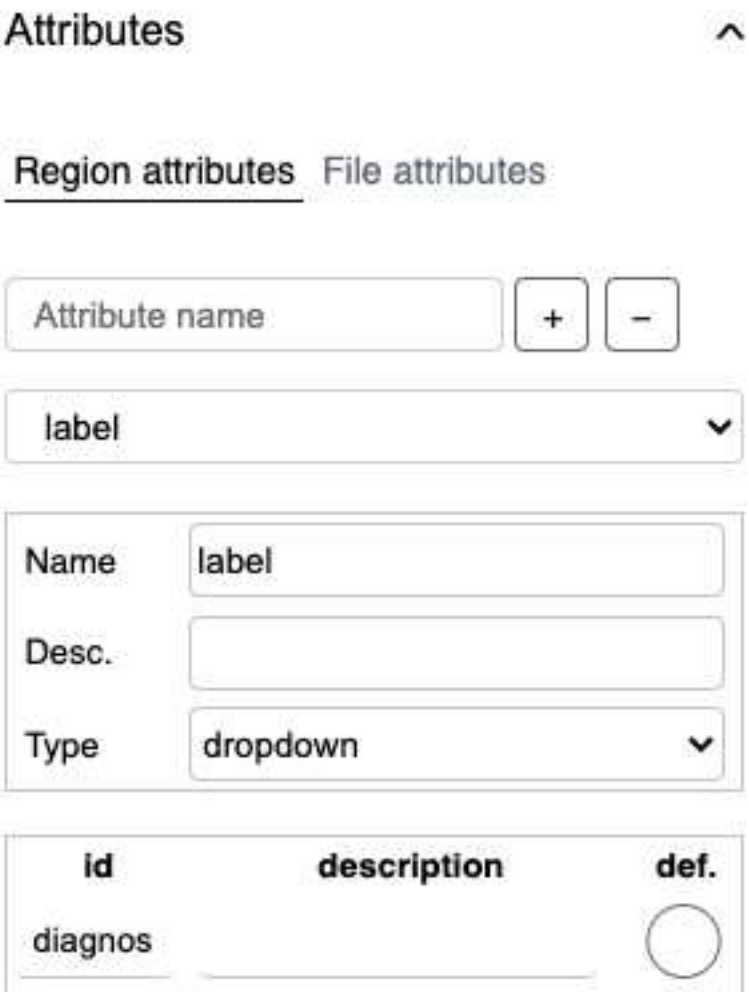


Note: Creating labels for your files **will** be time consuming.

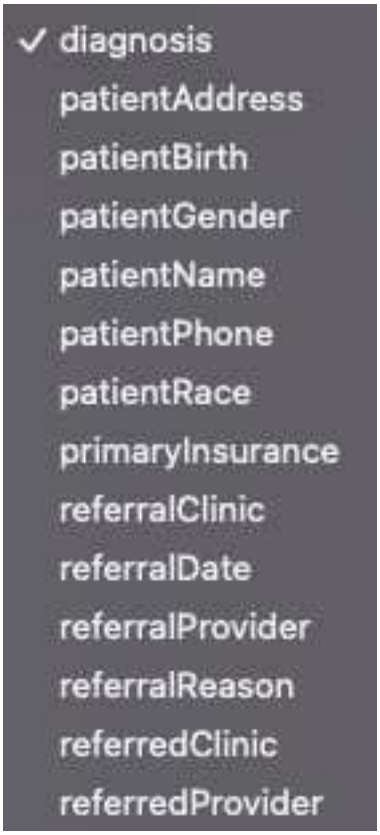
Creating region attributes

The following steps describe how to add a new attribute.

1. In the **Attributes** box, ensure that **Region attributes** is selected.
2. In the **Attribute name** field, enter "label" and click the + button to add the new attribute. (**Note:** For this step, the name of attribute must be set to "label" in order for H2O Document AI to work correctly.)



Create option ids You can now create the **option ids** (that is, the labels that can be assigned to regions when annotating documents) you want to use to label your files. For this tutorial, select *dropdown* as the **Type**, and then enter the option ids you want to use. In this tutorial, the following labels are used:

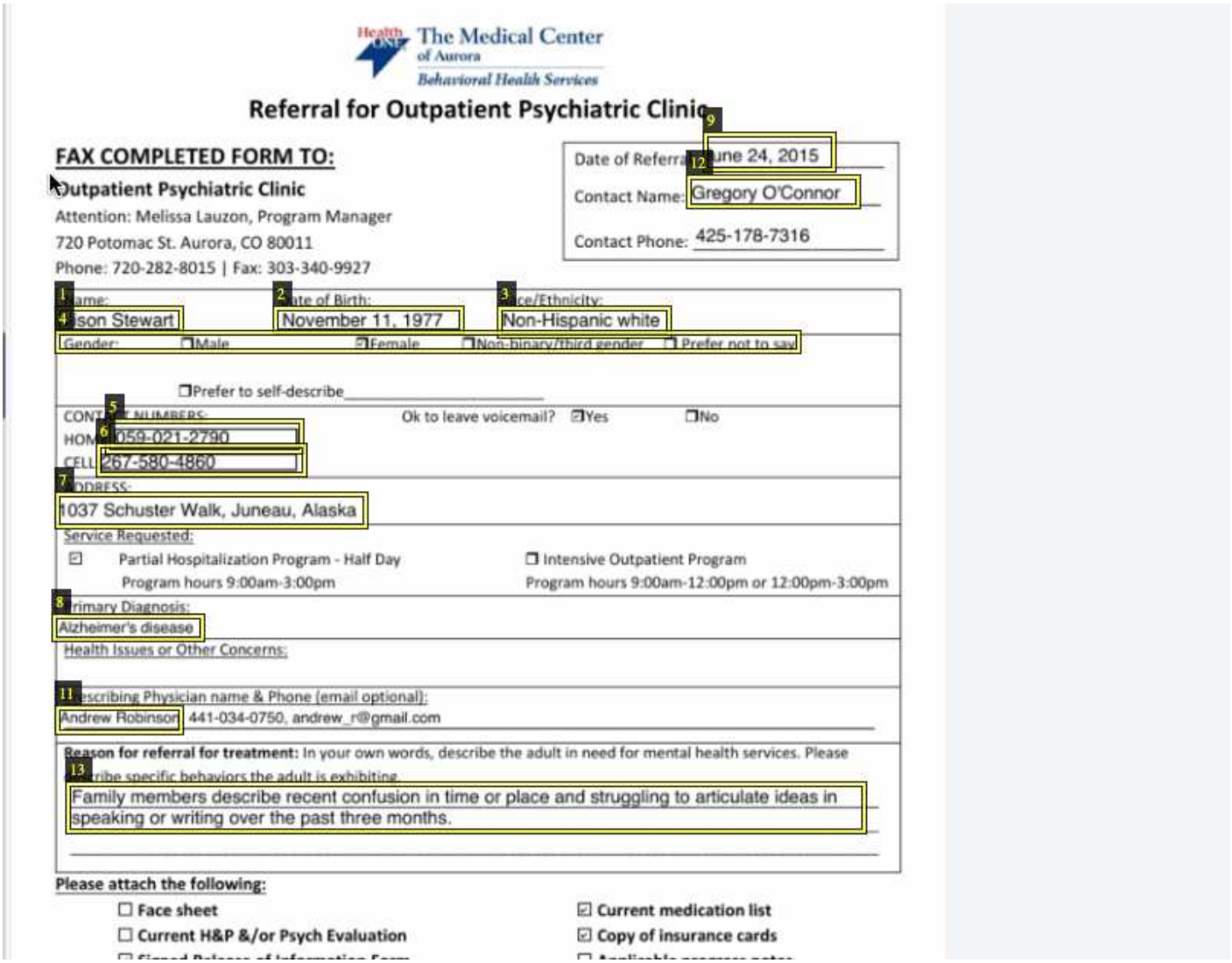


You can set one of these ids as your default value ("def."). This means that when you start assigning these ids to regions, that id will be the first choice provided.

Applying bounding boxes

To create a labeling region on your image, left-click and hold from one corner of the area you want bounded and drag to the opposite corner. Release the left-click to create the bounding box.

While the box is still highlighted, select the region id from the drop down menu on the annotation editor.



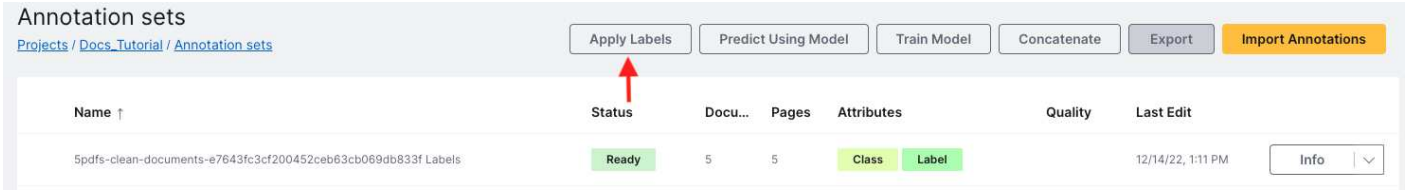
Create regions on all five documents using the option ids as guidance for what needs labeled. After creating each bounding box, select the option id that fits that region. Each page should have a handful of labeled regions.

After you have created all of your regions and labeled them, **save your work** by clicking the save button on the top tool bar.

Step 4: Applying labels

Navigate back to the Annotation sets page. You can see your labeled set has the *label* attribute now. Combining that with your token set (which has the *text* attribute) will give you a file that has both the *label* and *text* attributes. You need both of these attributes to create a `TokenLabeling` model.

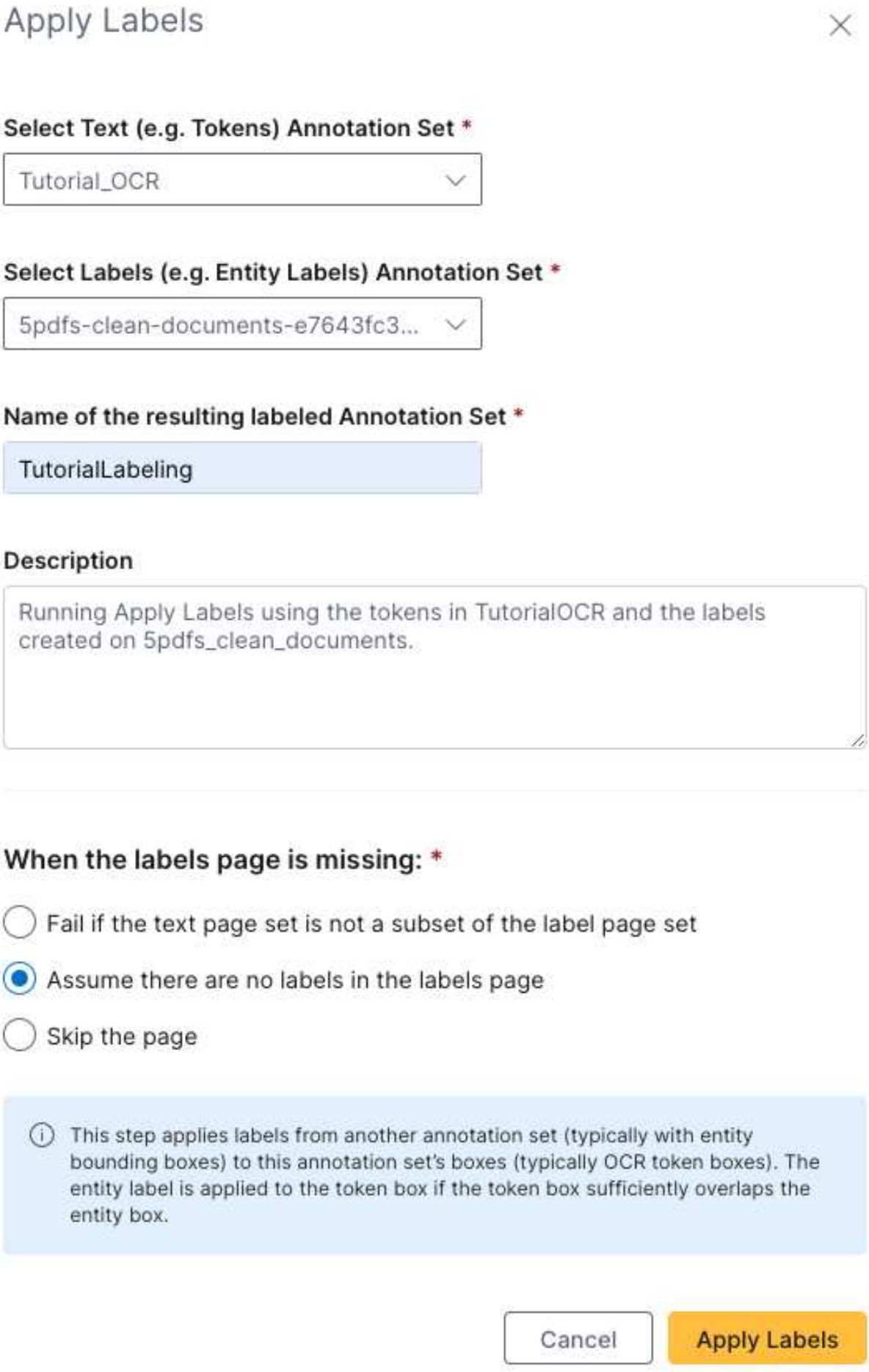
Click **Apply Labels** in the upper navigation bar. We will now combine the information we extracted from **OCR** and created in **Edit in Page View** by applying labels to our tokens.



On the **Apply Labels** launch page:

1. Select `Tutorial_OCR` for your text annotation set
2. Select `5pdfs-clean-documents Labels` for your labeled annotation set

3. Provide `TutorialLabeling` as the name
4. (Optional) Provide `Running Apply Labels using the tokens in Tutorial_OCR and the labels created on 5pdfs_clean_documents` as the description
5. Select *Assume there are no labels in the labels page* for what to do when the labels page is missing

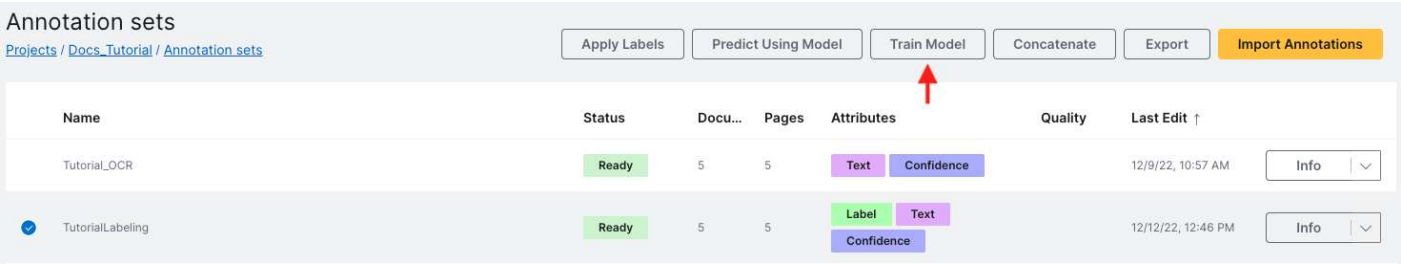


The labeled token set will appear on your Annotation sets page when the job is finished running.

Step 5: Training your Token Labeling model

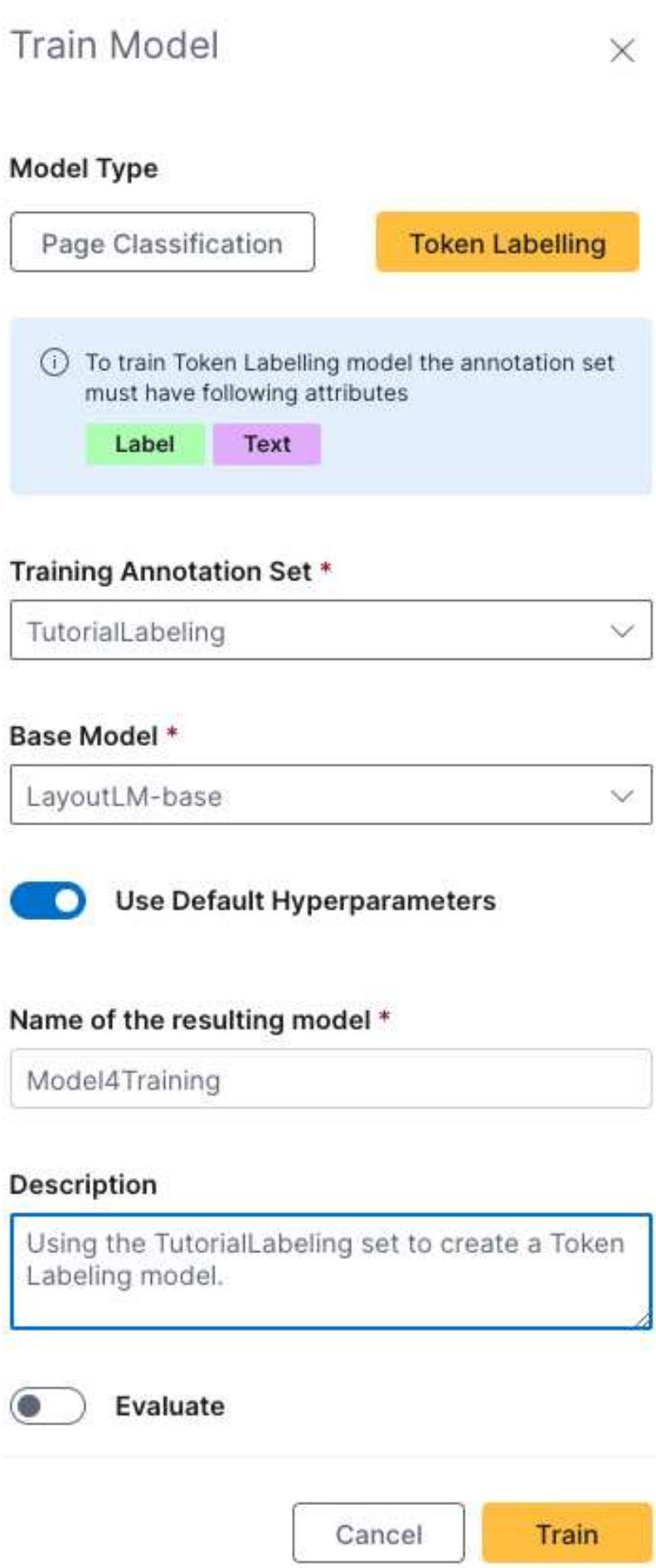
Now that you have a set with both the *text* and *label* attributes, you can build a Token Labeling model!

On the Annotation sets page, select the check box next to your Labeled Tokens set. Click **Train Model**.



On the **Train Model** launch page:

1. Select **Token Labeling** as your model type
2. Ensure that `TutorialLabeling` is set as your training annotation set
3. Provide `Model4Tutorial` as the name
4. (Optional) Provide `Using the TutorialLabeling set to create a Token Labeling model` as te description
5. Toggle **Evaluate** off since we have no validation set
6. Click **Train**

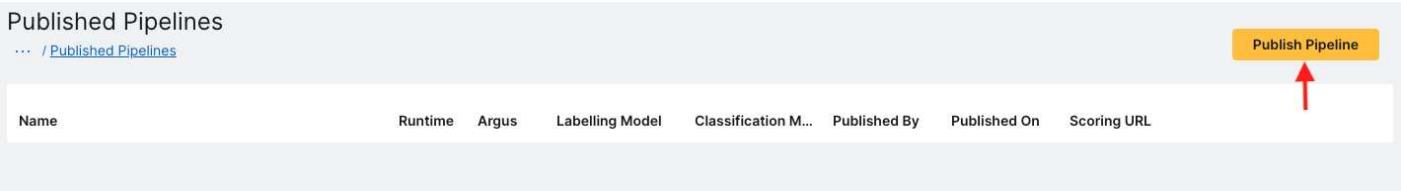


Once your model training job has finished running, your model will be available on the Models page.

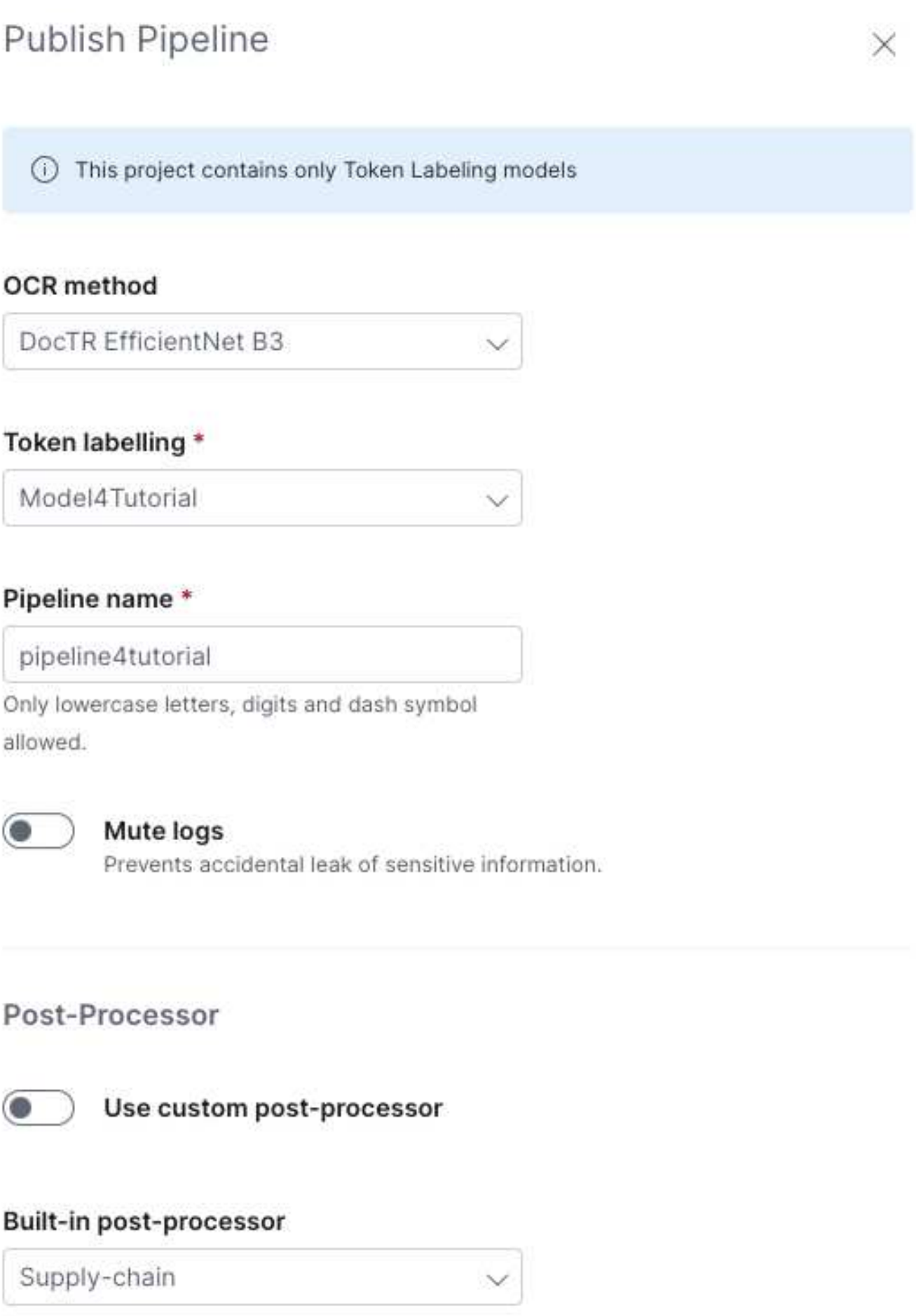
Step 6: Publishing a pipeline with your token labelling model

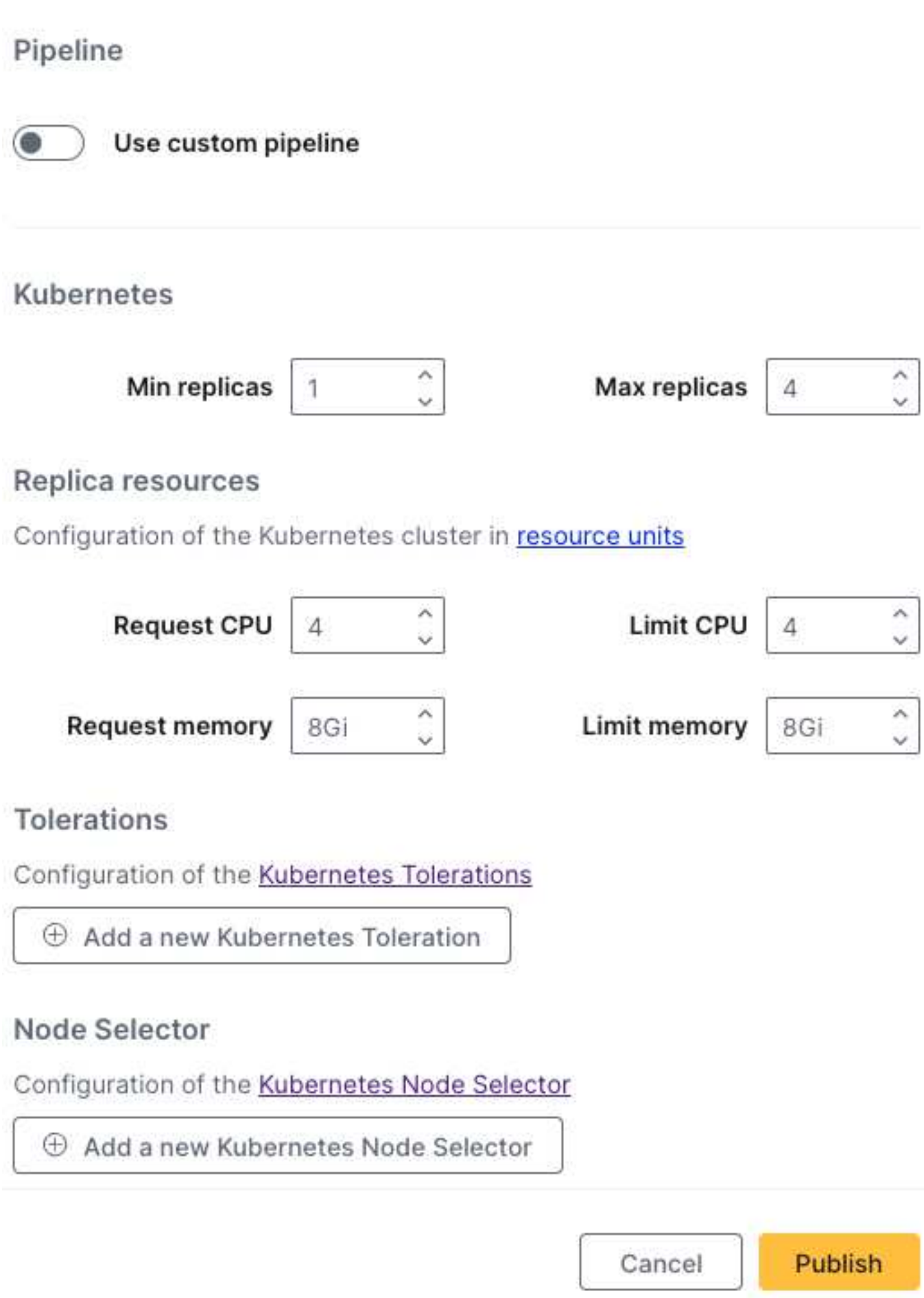
Navigate to the Published Pipelines page. You can now publish a scoring pipeline using the token labeling model you built.

Click **Publish Pipeline** in the upper navigation bar.



On the **Publish Pipeline** launch page: 1. Select *DocTR EfficientNet B3* as the OCR method you want your pipeline to use 2. Select your token labeling model 3. Provide `pipeline4tutorial` as the name for your Pipeline 4. Select *Supply-chain* as the built-in post-processor 5. Keep the rest of the default values 6. Click **Publish**

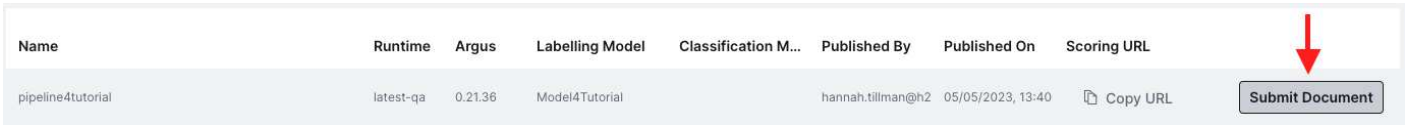




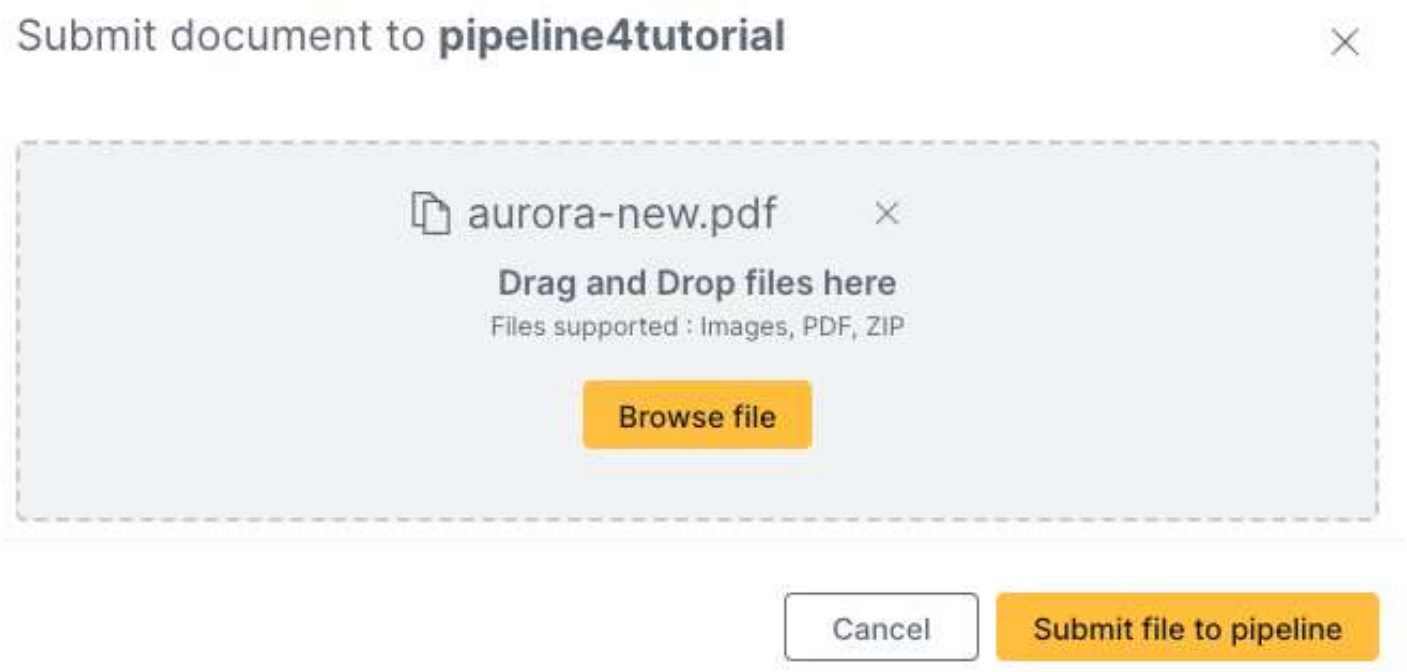
When your pipeline has finished publishing you will be able to access the scoring URL if you want to use the bulk scorer. Otherwise, you can submit new, uniquely named documents directly to the pipeline in the UI by clicking on the **Submit Document** button at the end of the pipeline row.

Submitting a new file to be scored

Try submitting your extra medical referral document to your published pipeline. Click **Submit document** to submit a new document to be scored.



On the submit document panel: 1. Add the `aurora-new.pdf` 2. Click **Submit file to pipeline**



Your file will be scored by your pipeline. To access your results: 1. Click the background processes bell icon on the top banner next to your account name 2. Click **View** next to the Scoring `aurora-new.pdf` finished process

A JSON of your scoring results will be available.

Summary

In this tutorial, you learned how to utilize the basic workflow of H2O Document AI - Publisher. You worked with creating a project, running OCR, annotating your images, applying labels to your tokens, building a token labeling model, and publishing a pipeline.

Next

If you would like to try building another model with validation using the information you have learned from this tutorial, test out Tutorial 1B: Creating an evaluation model in H2O Document AI - Publisher.

Tutorial 1B: Creating an evaluation model in H2O Document AI - Publisher

This tutorial will walk you through how to create a model using an evaluation annotation set so that you can access accuracy metrics from a trained model.

Prerequisites

- A copy of this zip file containing medical referral documents that you will be using in this tutorial
- Completion of the previous tutorial for a basic understanding of H2O Document AI - Publisher

Step 1: Create your project, run OCR, and annotate your files

You will be using the same set-up as the introduction tutorial. There are two ways to begin: 1. Continue using the same project from Tutorial 1A: Introduction to H2O Document AI - Publisher (continue to Step 2) 2. Perform the following three steps if you want to make a new project:

Creating the project

From the landing page, create a new project. 1. Click **Create a new project** 2. Provide `Eval_Tutorial` as the project name 3. Add the zip file `5pdfs_clean_documents.zip` of medical referral documents 4. Click **Create project**

Running OCR

Navigate to your Document sets page. 1. Select the *5pdfs_clean_documents* row 2. Click **OCR** on the upper navigation bar 3. Select the `best` OCR method 4. Provide the name `Tutorial_OCR` 5. Click **OCR**

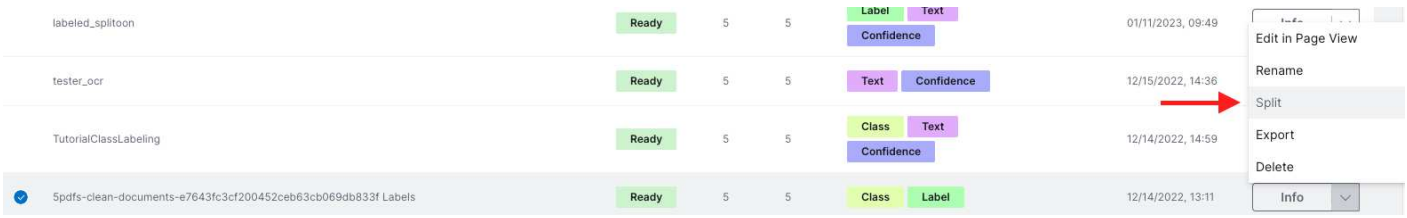
This will provide the *text* label and tokens you need for the apply labels step.

Annotating your files

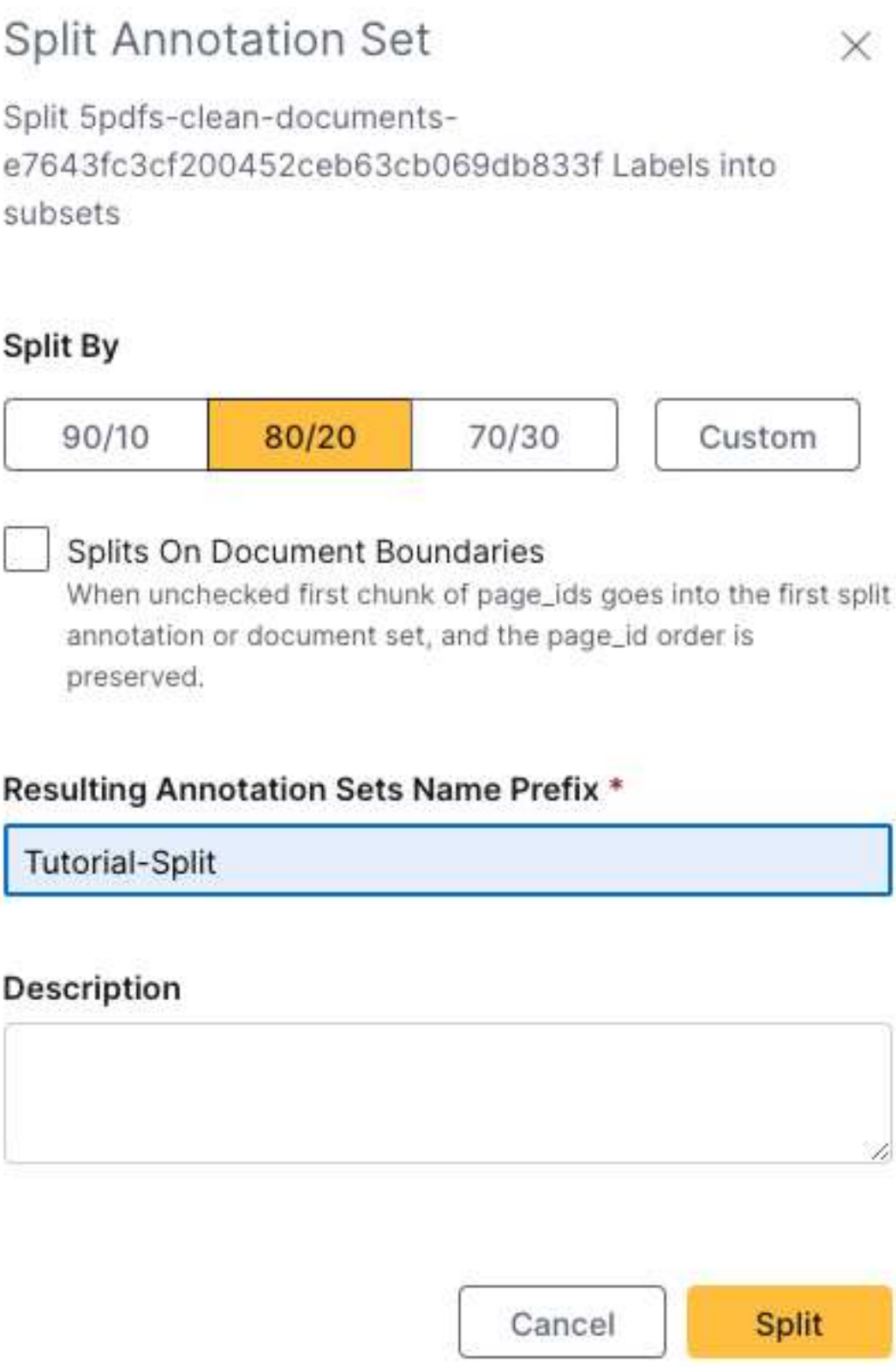
Navigate to your Annotation sets page. You will now annotate your blank annotation set. Refer to the previous tutorial for an in-depth walkthrough of annotation. 1. Open *5pdfs_clean_documents Labels* in **Edit in Page View** 2. Create your regional attributes following the outline of the previous tutorial 3. Apply your bounding boxes on each document and select the appropriate label for each box 4. Save your progress 5. Exit **Page View** and return to the Annotation sets page

Step 2: Splitting your labels annotation set

Since you are training a model with evaluation, you first need to create a training set and an evalution set. You have to split `5pdfs_clean_documents Labels` to do this.



1. Click the drop-down arrow at the end of `5pdfs_clean_documents Labels`
2. Click **Split**
3. Select **80/20** for your split
4. Provide the name prefix `Tutorial-Split`
5. Click **Split**



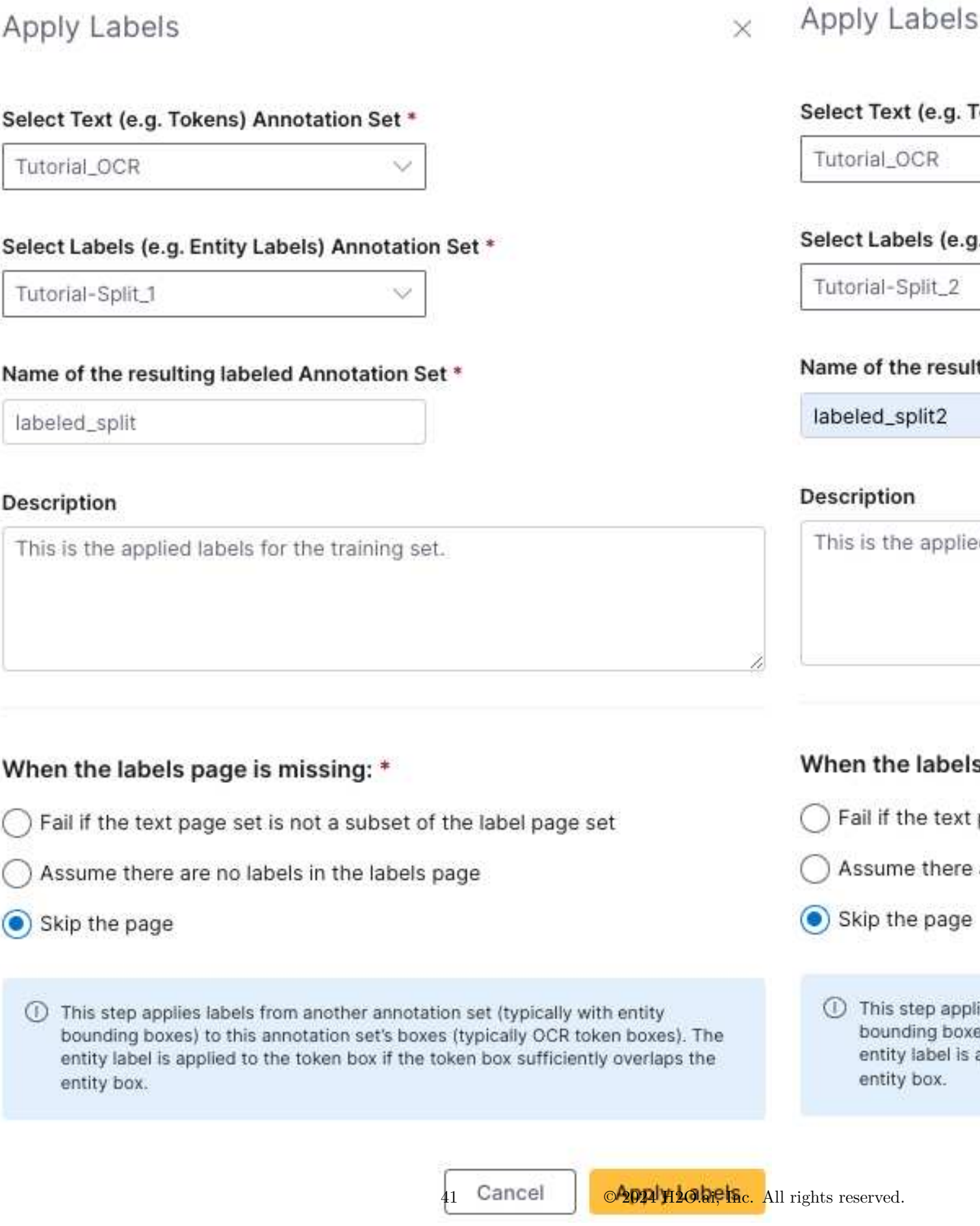
Step 3: Applying labels to your split annotation sets

You will now apply labels to each of your split annotation sets (so to `Tutorial-Split_1` and `Tutorial-Split_2`). 1. Click **Apply Labels** from the upper navigation bar of the Annotation sets page 2. Select `Tutorial_OCR` for your text annotation set 3. Select `Tutorial-Split_1` for your labels annotation set 4. Provide the resulting labeled annotation set the name `labeled-split1` 5. (Optional) Provide the description: `"This is the applied labels for the training set."` 6. Select *Skip the page* for what to do when the labels page is missing 7. Click **Apply Labels** to create your training

applied labels annotation set 8. Repeat 1-7 for `Tutorial-Split_2` to create your evaluation applied labels annotation set

Training labels	Evaluation labels

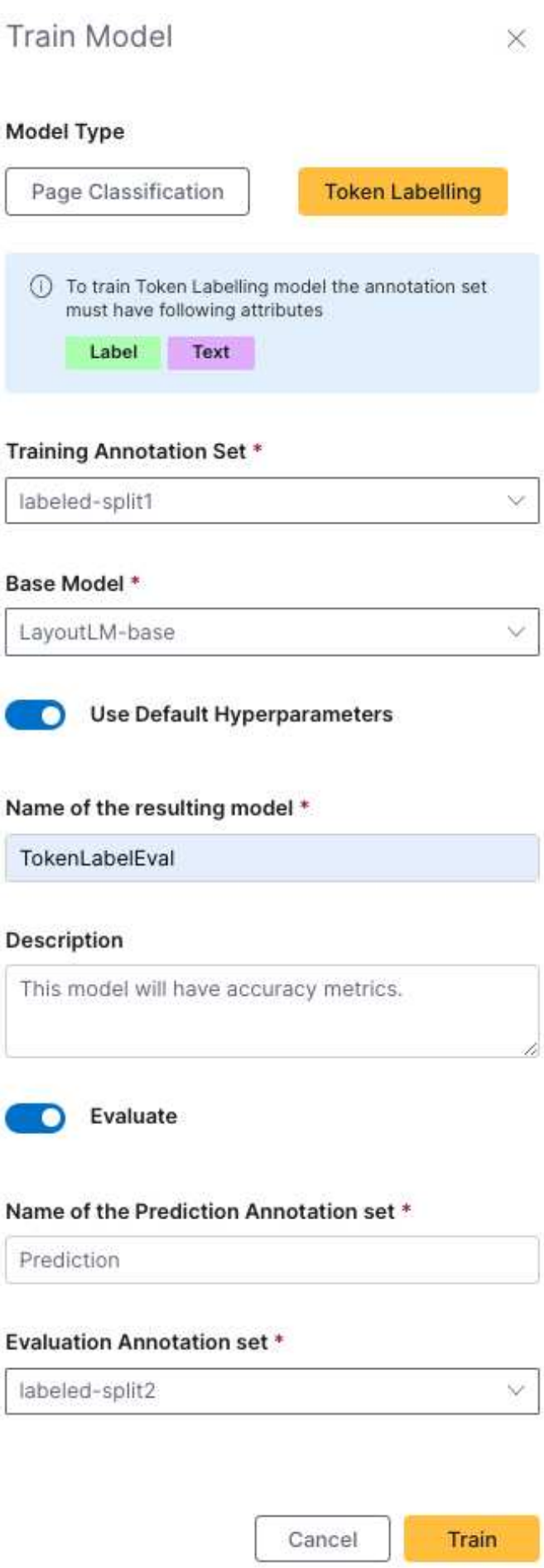
Training labels Evaluation labels



Training labels	Evaluation labels

Step 4: Training your evaluation model

Now that you have your training and evaluation split annotation sets, you can build an evaluation model. From the Annotation sets page, click **Train Model** on the upper navigation bar. 1. Switch the model type to **Token Labelling** 2. Select `labeled-split1` as your training annotation set 3. Name your resulting model `TokenLabelEval` 4. (Optional) Provide `This model will have accuracy metrics.` as the description 5. Name your prediction annotation set `Prediction` 6. Select `labeled-split2` as your evaluation annotation set 7. Click **Train**

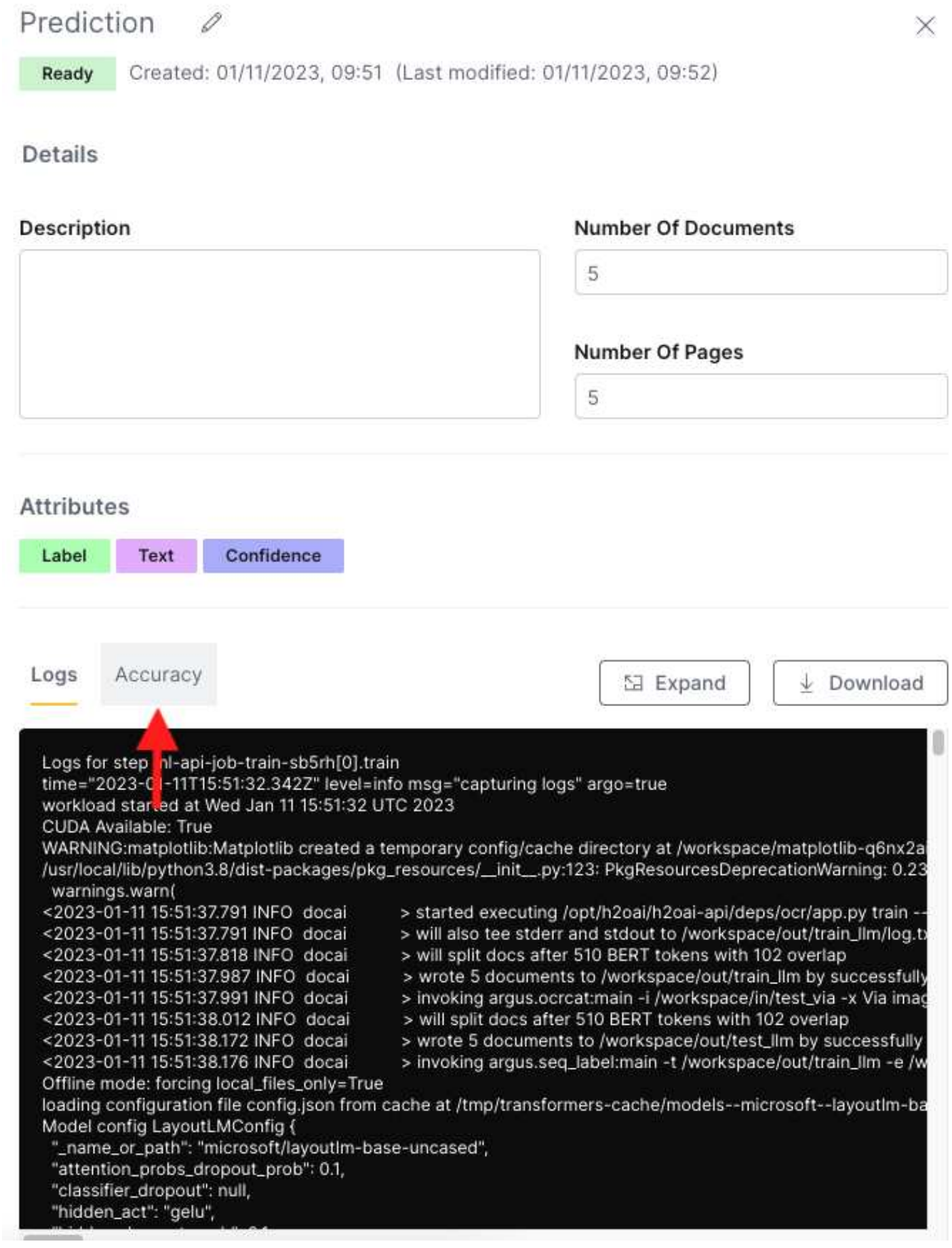


Your trained model can be found on the Models page. You prediction annotation which has your accuracy metrics can be found on the Annotation sets page with a quality score. Your quality score is the f1-score of the model that was applied to the dataset.

Step 5: Accessing your accuracy metrics

Stay on the Annotation sets page. Your `Prediction` evaluation annotation set should be at the top. Click **Info** at the end of the row to bring up the information of `Prediction`.

This brings you to the information panel. Here you can update the name or description of your evaluation set. You can also access the logs and accuracy information. Click **Accuracy** to access the accuracy information.

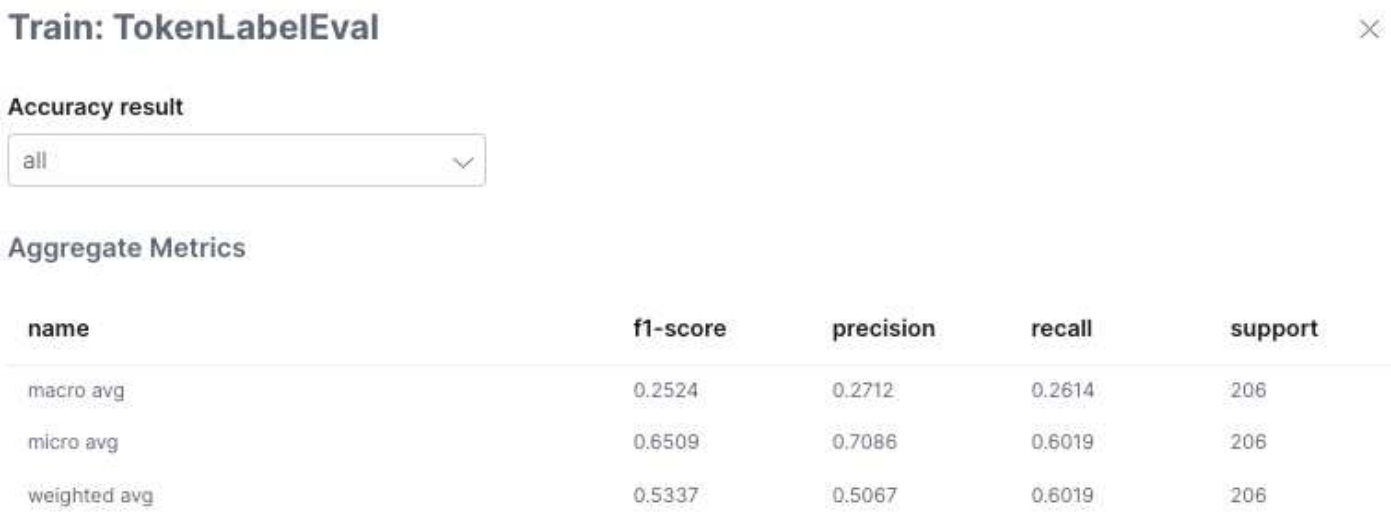


On the accuracy panel, you can scroll through the information on the side bar, you can expand the information into a pop-out screen by clicking **Expand**, or you can download a JSON of your accuracy information by clicking **Download**. For this tutorial, click **Expand** to bring your information up in an easier-to-read pop-out screen.

Your accuracy information is broken down into three tables: aggregate metrics, per class metrics, and a class confusion matrix.

Aggregate metrics

The aggregate metrics measure the accuracy of the entire dataset.



Train: TokenLabelEval

Accuracy result

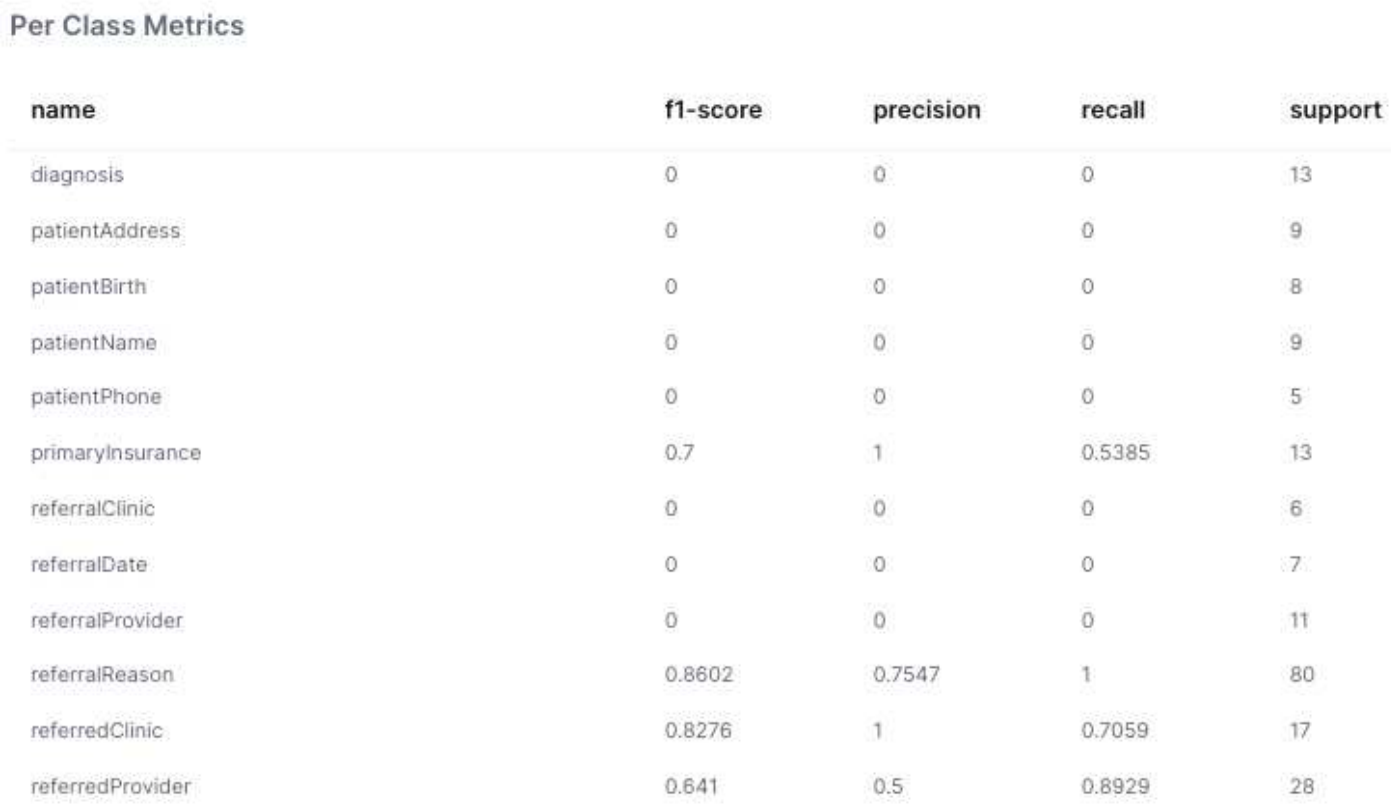
all

Aggregate Metrics

name	f1-score	precision	recall	support
macro avg	0.2524	0.2712	0.2614	206
micro avg	0.6509	0.7086	0.6019	206
weighted avg	0.5337	0.5067	0.6019	206

Per class metrics

The per class metrics measure the accuracy of one class versus the rest of the datapoints in the dataset.

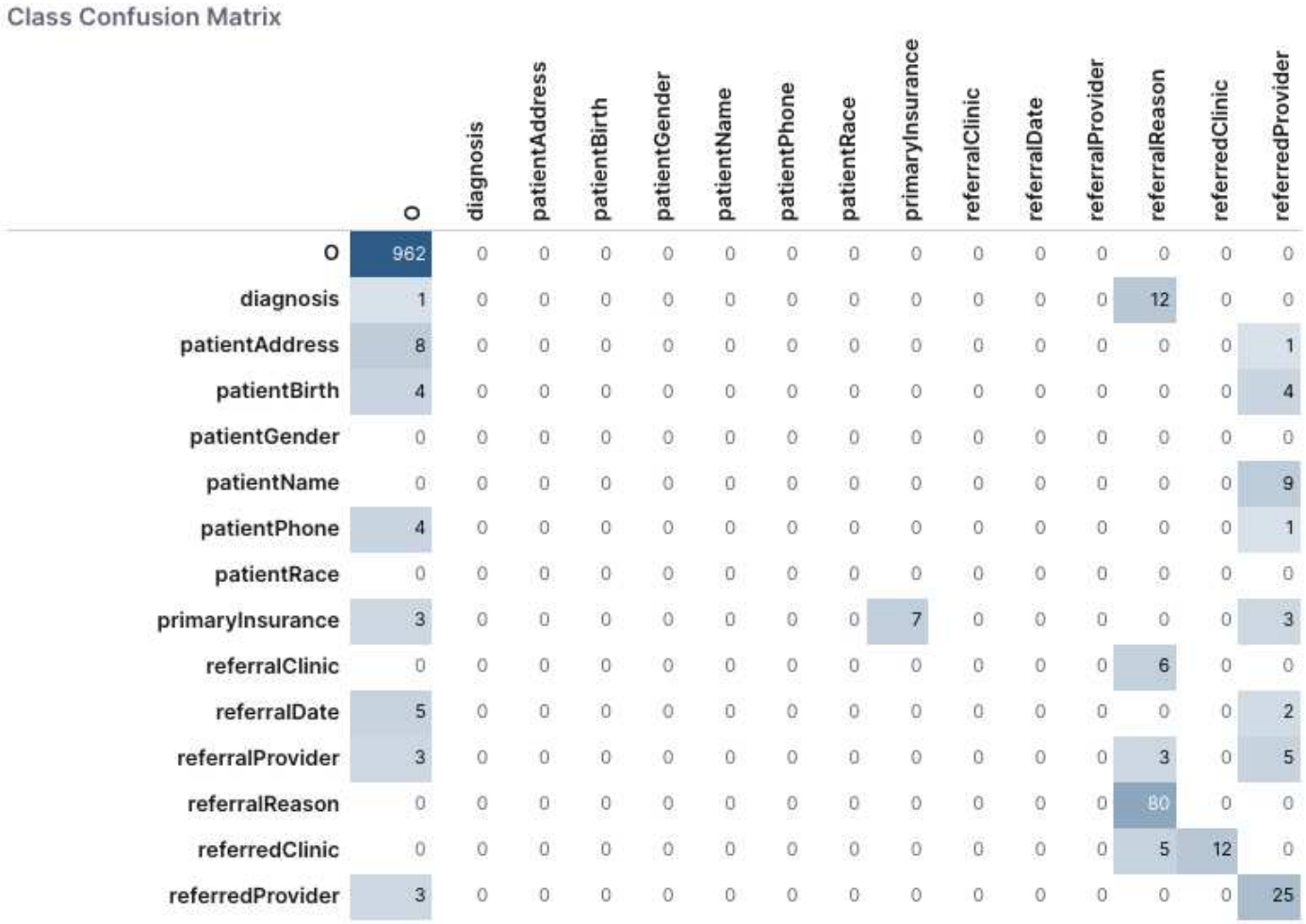


Per Class Metrics

name	f1-score	precision	recall	support
diagnosis	0	0	0	13
patientAddress	0	0	0	9
patientBirth	0	0	0	8
patientName	0	0	0	9
patientPhone	0	0	0	5
primaryInsurance	0.7	1	0.5385	13
referralClinic	0	0	0	6
referralDate	0	0	0	7
referralProvider	0	0	0	11
referralReason	0.8602	0.7547	1	80
referredClinic	0.8276	1	0.7059	17
referredProvider	0.641	0.5	0.8929	28

Class confusion matrix

The class confusion matrix provides a tabular visualization of the model predictions versus the actual values for all classes.

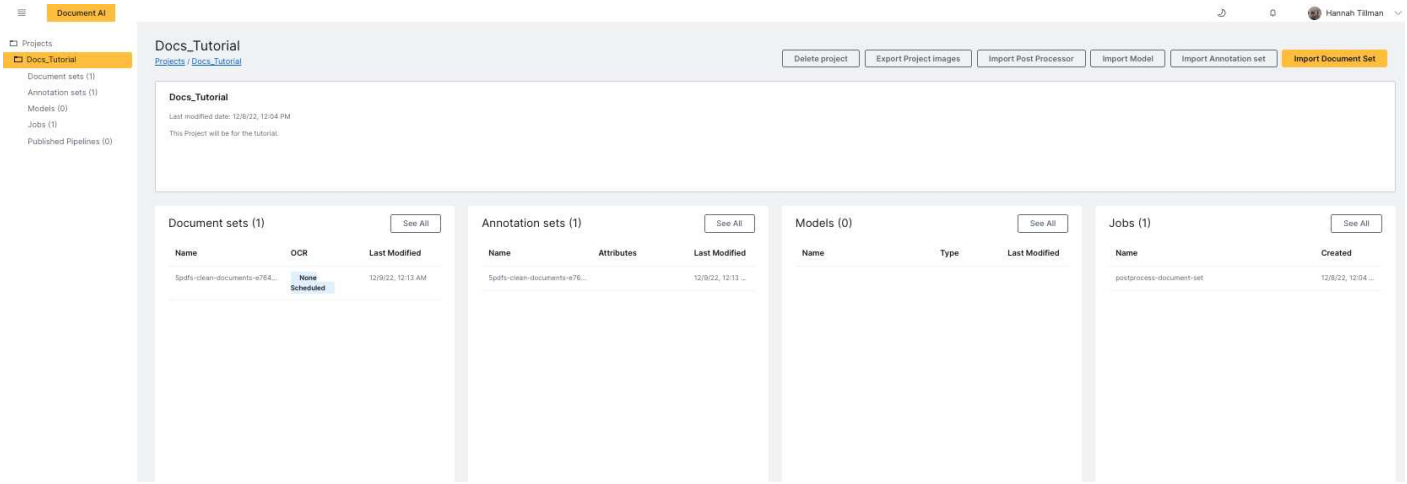


Summary

In this tutorial, you learned how to train a model with evaluation allowing you to access accuracy metrics. You split your labels annotation set and applied labels to each split. Then, you built a model using those two labeled splits. Finally, you learned how to navigate the accuracy panel.

Understanding the user interface

The user interface of H2O Document AI - Publisher is intuitive.

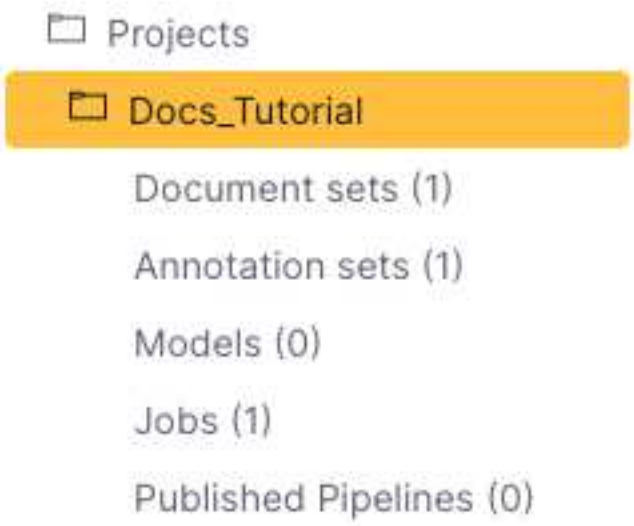


Navigating H2O Document AI - Publisher

There are four ways you can interact with H2O Document AI - Publisher: the left navigation bar, the upper navigation bar, the **Info** button at the end of each set/model/job row, and the drop-down arrow next to the **Info** button.

Left navigation bar

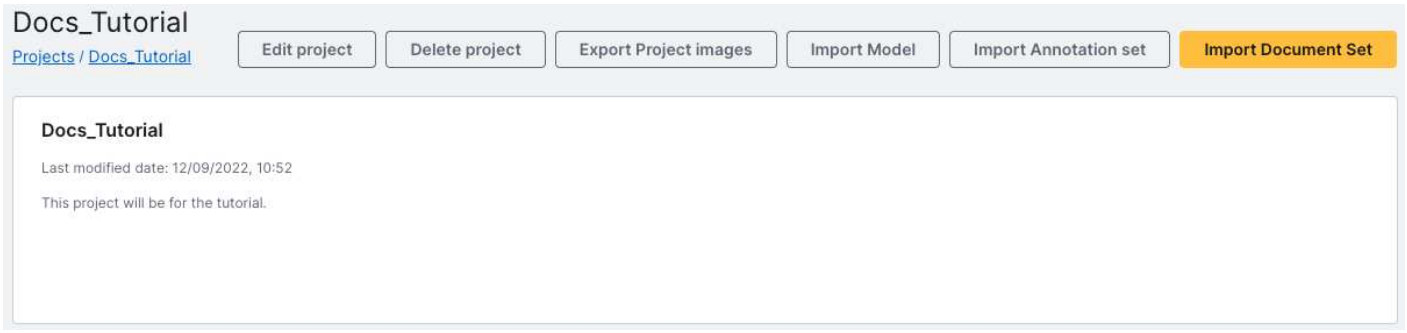
The **left navigation bar** takes you to the other pages within your project: Document sets, Annotation sets, Models, Jobs, and Published Pipelines. The number of files or items on each page is listed next to the page name (e.g. *Document sets (1)*). This navigation bar stays the same regardless of what page you are on.



Upper navigation bar

The **upper navigation bar** shows certain tasks you're able to accomplish on each page. These tasks are options which don't depend on which files you've selected (and are therefore clickable without selecting a file) or are options that depend on selection and support batch operation (and are therefore not clickable until you have selected a file).

This navigation bar changes depending on which page you are on. On the Project page, for example, there is the option to delete the project, export project images, import a post processor, import a model, import an annotation set, or import a document set.



Info button

The **Info** button 

gives you access to your set, model, or job's information and logs. You can update the name and description of the set, model, or job here. You also have access to accuracy information for prediction annotation sets from here.

Drop-down arrow

The drop-down arrow next to the **Info** button gives you primary actions that can be clicked without expanding the context menu. The drop-down arrow on the Annotation sets page, for example, gives you the option to edit your annotation set in Page View, rename your annotation set, split your annotation set, export your annotation set, or delete your annotation set.

Interacting with sets

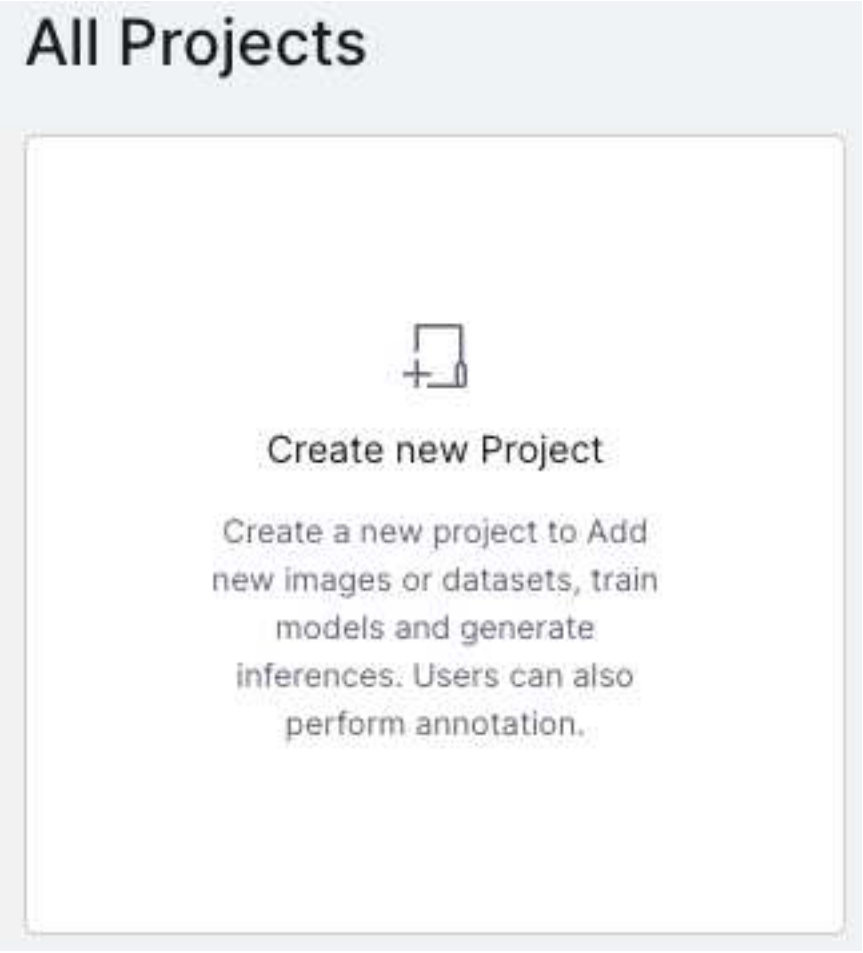
When interacting with document or annotation sets, you can either **click the check** next to the name of the set (this will let you select more than one set at a time) or you can **click the set's row** to select a single set at a time. Based on the set(s) you select, certain options may become available in the upper navigation bar.

Creating and managing a project

A **Project** is a set of data (documents or pages with various annotations used for training and evaluation) and models (a system that you train on the training data to make the predictions on other documents or pages not seen before).

Create a new Project

1. Click the **Create a new Project** panel on the H2O Document AI - Publisher homepage



2. Enter the **name** of your project
3. **Toggle** whether to auto-purge your training artifacts
4. **Toggle** whether to schedule the deletion of your project along with its resources
5. (Optional) Add a **description** for your project
6. Input any collaborators you want to work with on this project (if none, then leave empty)
7. **Upload** the desired documents, images, or compressed files
8. Click **Create project**

You will find your newly created project under **Projects** on the left navigation bar. You will also see the artifact uploaded under the **Documents sets** and **Annotation sets** sections on your Project home page.

Note: Depending upon the size of the uploaded file, it might take some time to populate the **Documents sets** and **Annotation sets** sections with the number of documents and the number of pages. You can check the status of the upload by looking at the **Status** column on the **Document sets**, **Annotation sets**, or **Jobs** pages. A job that is running is *pending* or *processing*. A finished job is *done* or *ready*.

You can also check the status of a project by opening the background processes tab. You access this by clicking the **bell icon** next to your account name.

Auto-purge

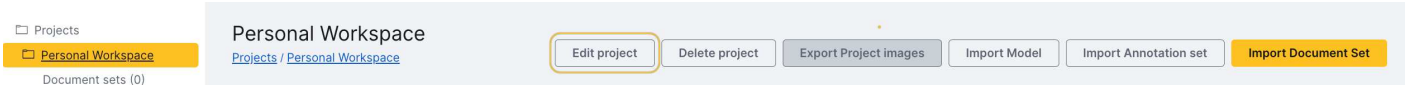
Automatically purge your training artifacts. Set this option when creating or editing your project. This feature is enabled by your system administrator. The retention period is set by your system administrator, too.

Scheduled deletion

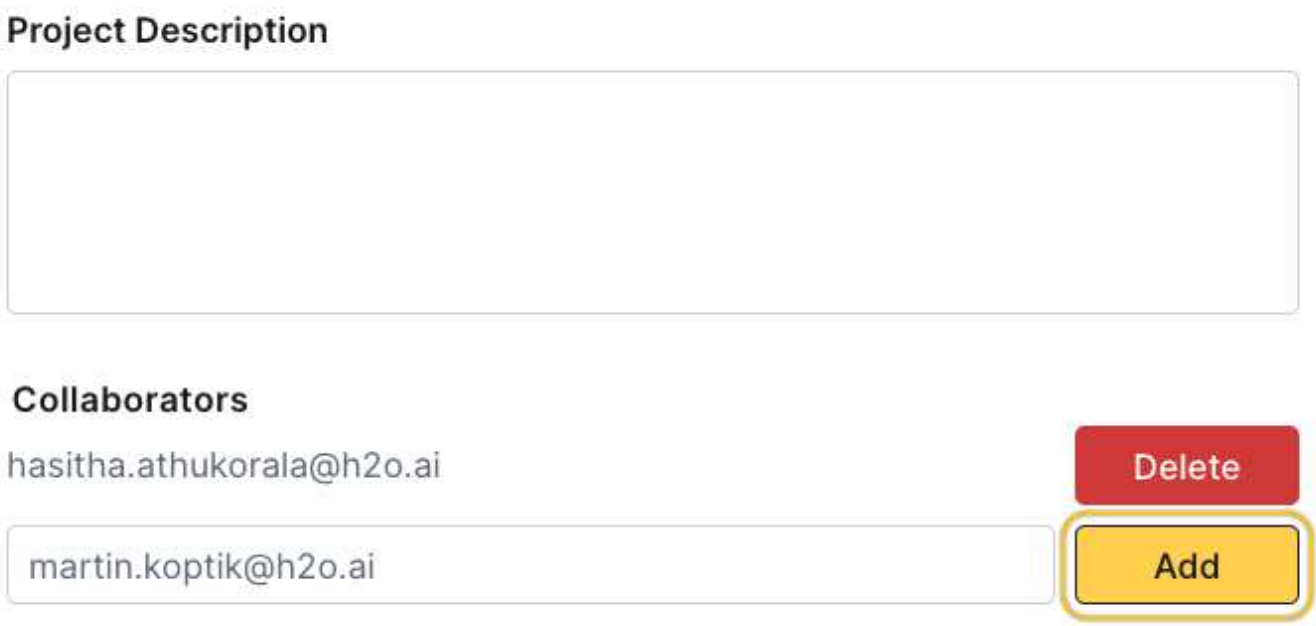
Schedule the deletion of your whole project and all of its resources. By toggling on this feature while creating or editing your project, you can set the date you want to erase your project.

Project collaboration

You can add other users (collaborators) to work on your project by providing their email addresses in the **Edit Project** panel in the upper navigation bar.



The project creator is the owner, and ownership cannot be transferred. Only the owner can add or remove collaborators and open the Edit Project panel. Aside from these owner-specific privileges, collaborators have the same permissions as the owner, including the ability to work with document and annotation sets and publish pipelines.



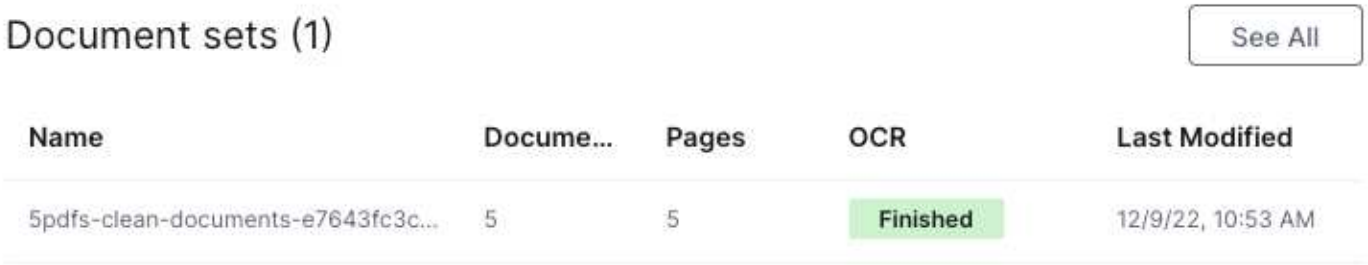
The UI manages collaborators using a picker component, and role bindings are made exclusively through email. The UI verifies that the email exists within your organization, but it does not provide autocomplete. If the email is invalid, an error response will be displayed.

Project panels

The project panels show overviews of each page. You can access each page by clicking **See All** on this panel or by accessing it via the left navigation bar.

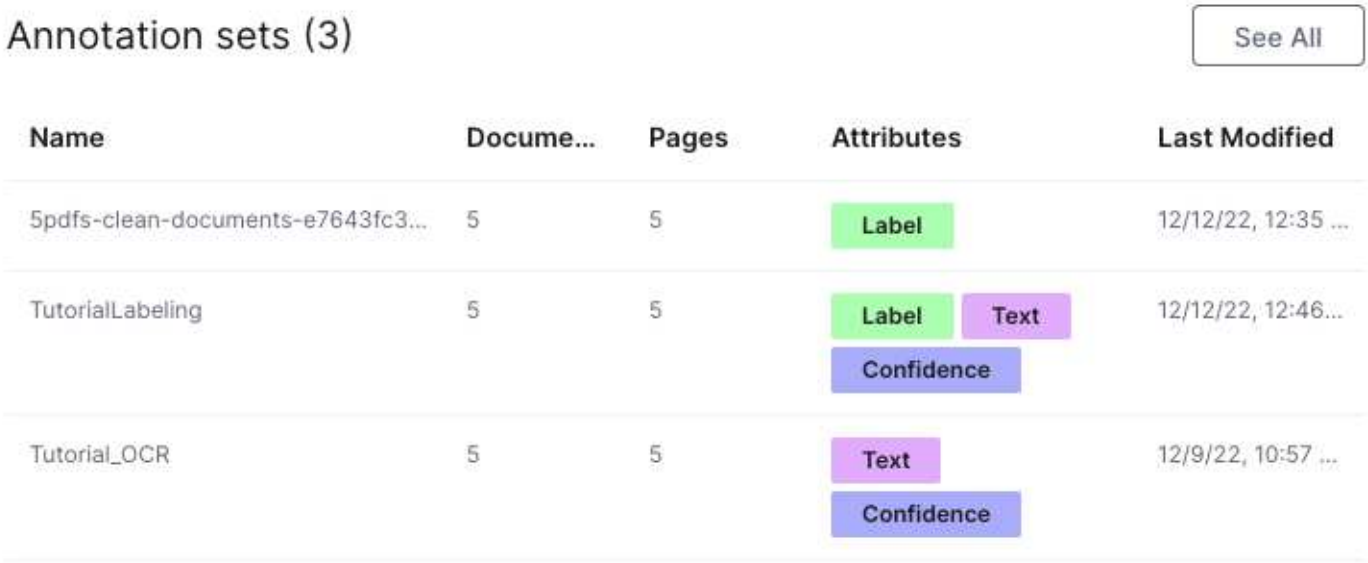
Document sets

See the most recent updates and scroll through older ones. Select "See All" to access the **Document sets** page.



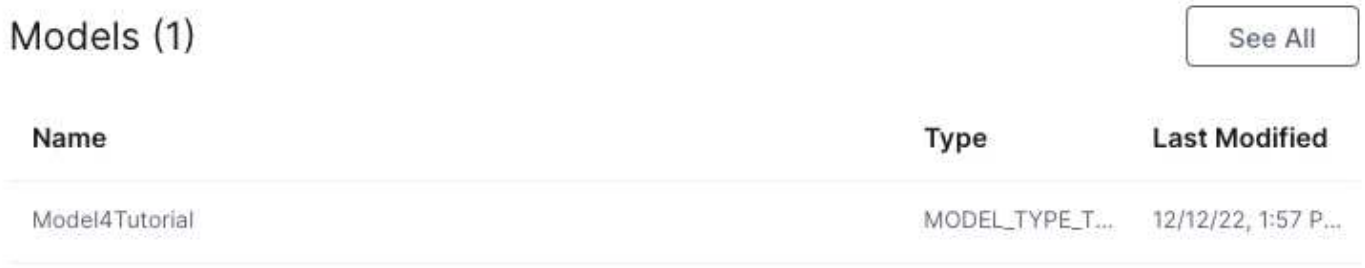
Annotation sets

See the most recent updates and scroll through older ones. Select "See All" to access the **Annotation sets** page.



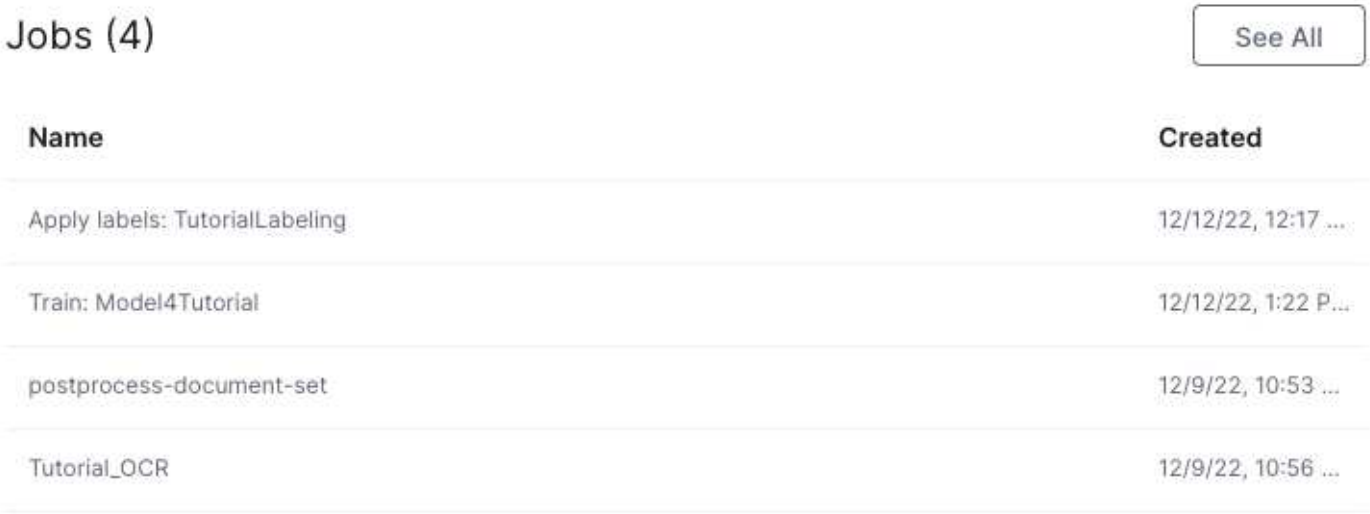
Models

See the most recent updates and scroll through older ones. Select "See All" to access the **Models** page.



Jobs

See the most recent updates and scroll through older ones. Select "See All" to access the **Jobs** page.

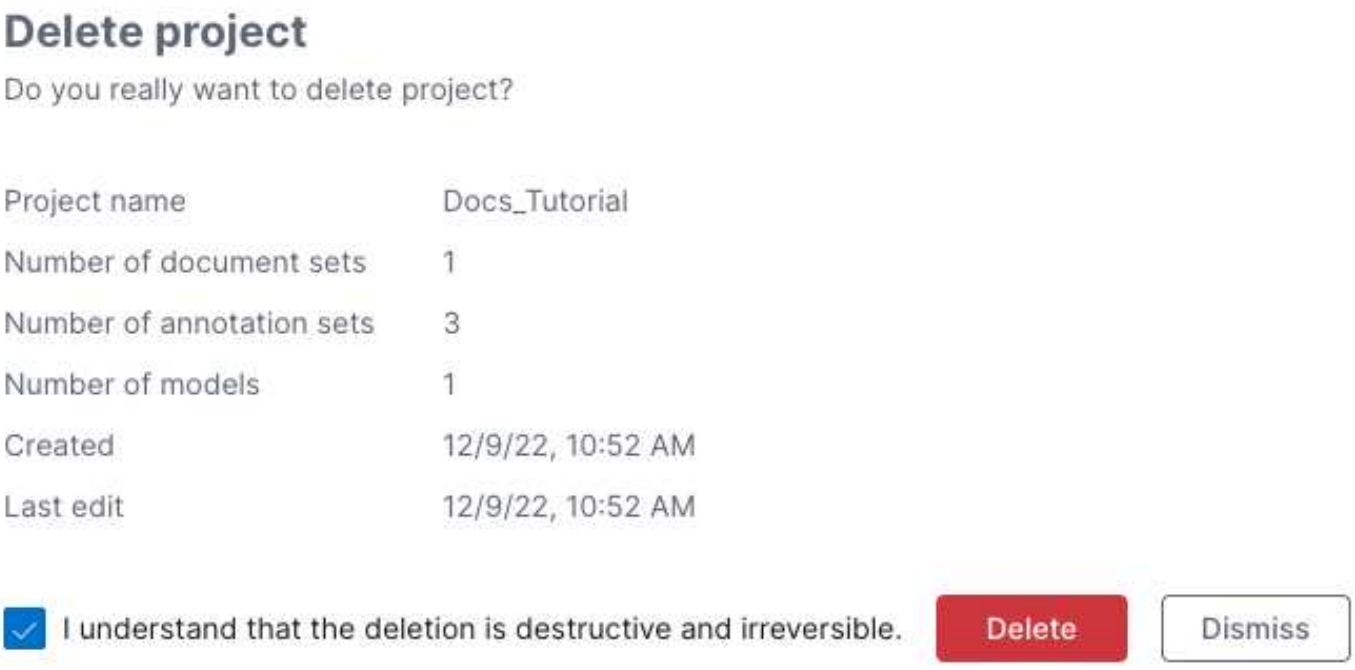


Project jobs

Interact with your project from the upper navigation bar.

Delete project

Delete your project. You will be prompted to acknowledge that the act of deletion is destructive and irreversible.



Export Project images

Export all or selected images from your project. Exporting lets you work with annotation sets offline.

1. Choose which annotation set images to export by clicking the box next to their names
2. Click **Export**

When the job is finished running, the images will be downloaded to your local computer.

Import Post Processor

Import a custom Post Processor.

1. Provide a **name** for your post processor
2. (Optional) Provide a description of your post processor
3. **Drag and drop** the zipped post processor directory or **browse your local files**
4. Click **Import**

Import Model

Import a previously built model that you have saved to your local computer.

1. Provide a **name** for your imported model
2. (Optional) Provide a description for your model
3. **Drag and drop** the file or **browse your local files** for the zipped model file
4. Click **Import**

Import Annotation set

Import a JSON file of a previously annotated annotation set that you have saved to your local computer.

1. Provide a **name** for the imported annotation set
2. (Optional) Provide a description for your annotation set
3. **Drag and drop** the JSON file or **browse your local files** for the file you want to import
4. Click **Import**

Import Document set

This will import a set of documents.

1. Provide a **name** for the document set
2. (Optional) Provide a **description** for your document set
3. Select if you want to **copy the attributes** from an available attribute set
4. **Upload** the desired documents, images, or compressed files. You can either drag and drop the zip file or browse for it

Once the document is imported, it will appear in the Document sets page and an entry will appear in the Annotation sets page.

Using document sets

Work with available document sets or upload a new set.

OCR

The optical character recognition (OCR) method figures out what text is on the page. Start by selecting which document set(s) you want to work with. The **OCR** button is now available. Click **OCR** in the upper navigation bar.

1. Ensure you selected the correct **document set(s)**
2. Select the OCR Method. These are the engines for running OCR. This could be one of:
    - **Tesseract**: works on pdfs and images, less accurate than PDF text extraction
    - **PDF text extraction**: works on some PDFs where text can be safely extracted and will perfectly extract the text
    - **docTR**: works on pdfs and images, less accurate than PDF text extraction
    - **Paddle OCR Latin**: uses PPOcr library to do OCR on Latin (e.g. Portuguese and Spanish)
    - **Paddle OCR Arabic**: uses PPOcr library to do OCR on Arabic
    - **DocTR EfficientNet B3**: uses the DocTR detection model and the EfficientNet B3 architecture recognition model trained on in-house generated data including handwritten datasets
    - **DocTR EfficientNet B0**: slightly lighter and faster, but less accurate than EfficientNet B3
    - **DocTR EfficientNet V2M**: larger and slower, but more accurate than EfficientNet B3
    - **E3 Best**: first runs PDF test extraction, and if it cannot extract the text, runs DocTR EfficientNet B3 OCR
    - **best**: first runs PDF text extraction, and if it cannot extract the text, runs docTR OCR
3. Provide a **Result name**
4. (Optional) Add a description
5. Click **OCR** to run the action

After this action finishes running, your OCR object will appear on the Annotation sets page.

Supported languages

Here is a list of the languages supported by H2O Document AI - Publisher:

Latin

- Afrikaans
- Azerbaijani
- Bosnian
- Czech
- Welsh
- Danish
- Spanish
- Estonian
- French
- Irish
- Croatian
- Hungarian
- Indonesian
- Icelandic
- Italian
- Kurdish
- Lithuanian
- Latvian
- Maori

- Malay
- Maltese
- Dutch
- Norwegian
- Occitan
- Polish
- Portuguese
- Romanian
- Serbian (Latin)
- Slovak
- Slovenian
- Albanian
- Swedish
- Swahili
- Tagalog
- Turkish
- Uzbek
- Vietnamese
- German

Arabic

- Arabic
- Persian
- Uyghur
- Urdu

Import document set

This will import a set of documents. The names of these files must be unique for the import to succeed.

1. Provide a **name** for the document set
2. (Optional) Add a description for the document set
3. Select if you want to **copy the attributes** from an available attribute set
4. **Upload** the desired documents, images, or compressed files. You can either drag and drop the zip file or browse for it

Once the document is imported, it will appear on the Document sets page and an entry will appear on the Annotation sets page.

Supported file types

- Images
- PDFs
- ZIPs (and nested ZIP files)

You can upload multiple files with the same name (they can have the same file sets and everything). You'll just be prompted to select which file should be used. You will only be prompted when importing an annotation set which references an ambiguous file.

Interacting with a document set

Each document set has an **Info** button at the end of the row. Clicking **Info** will give you the details of your document set (e.g. the description or number of pages). You can also find the logs for your document set here. To see the full log, click **Expand**. You can also download the log by clicking **Download**.

The drop-down arrow next to the **Info** button gives you the option to either rename, export, or delete your document set.

Rename

Rename the document set and provide a new description.

Export

Export a zip file of the document set to your local computer.

Delete

Delete the document set. You will be prompted to acknowledge that the act of deletion is irreversible before you can delete your model

Using annotation sets

Work with available annotation sets or create new ones. An annotation set can contain multiple types of annotations, including text (usually generated from the optical character recognition OCR process), token entity annotations of type "label", and page annotations of type "class".

Apply Labels

This action applies labels from the "Labels" annotation set to the boxes of a Text annotation set (typically OCR token boxes). The entity label is applied to the token box if the token box sufficiently overlaps the entity box. This job also combines attribute types from the two sets you are using (e.g. giving the applied label set the attributes *text* and *label*).

1. Select the **Text Annotation Set**
2. Select the **Labels Annotation Set**
3. Provide a **name** for the resulting labeled Annotation Set
4. (Optional) Provide a description
5. Select what this method should do **if the labels page is missing**
6. Click **Apply Labels**

The newly labeled annotation set will appear at the top of your annotation sets list. Open the file in "Edit in Page View" to see the annotated labels.

Predict Using Model

Predict on an annotation set using a successfully built model.

1. Select the **model** to use for predictions
2. Provide a **name** for the prediction set
3. (Optional) Provide a description for the prediction set
4. Select the **Evaluation Annotation set**
5. Click **Predict** to retrieve the predictions

Train Model

Train a model based on your annotation sets. Training models is the H2O Document AI - Publisher job that requires the most time.

1. Select the **model type**. One of:
    - Token Labeling - requires the annotation set to have the *label* and *text* file attributes
    - Page Classification - requires the annotation set to have the *class* and *text* file attributes
2. Select the specific **annotation set** you want to train the model
3. (Optional) Set the batch size and number of epochs
4. Provide a **name** for the resulting model
5. (Optional) Provide a description of the model
6. Click **Train** to build the model

In the Train Model screen, you can also select whether you want to evaluate the model. This lets you train a model using a validation set. Before building the model:

1. Toggle **Evaluate** on
2. Provide a **name** for the Prediction Annotation set
3. Select the **Evaluation Annotation Set**

The model will appear on the Models page when it is finished. If you provided a validation set to build your model, you will also be able to access the **accuracy** of your model. You view the accuracy of your model on the prediction annotation set which is generated after your model is built. The prediction annotation set can be accessed from the Annotation sets page. Learn more about the accuracy panel.

Concatenate

You can concatenate (combine) annotation sets that have the same attributes.

1. Select the annotation sets you want to combine (sets must contain the same attributes)
2. Provide a result name
3. (Optional) Provide a description for your concatenated annotation set
4. Click **Concatenate**

The concatenated annotation set will appear in the Annotation sets page.

Export

Export the selected annotation set(s).

1. Select the file(s) you want to export
2. Click **Export** on the upper navigation bar

Import annotations

Import a JSON file of a previously annotated annotation set that you have saved to your local computer.

1. Provide a **name** for the imported annotation set
2. (Optional) Provide a description
3. **Drag and drop** the JSON file or **browse your local files** for the file you want to import
4. Click **Import**

Interacting with an annotation set

Each annotation set has an **Info** button at the end of the row. Clicking **Info** will give you details of your annotation set (e.g. the description or number of pages). You can also find the logs for your annotation sets here. To see the full log, click **Expand**. You can also download the log by clicking **Download**.

The drop-down arrow next to the **Info** button gives you the option to edit your annotation set in Page View, rename your annotation set, split your annotation set, export your annotation set, or delete your annotation set.

Edit in Page View

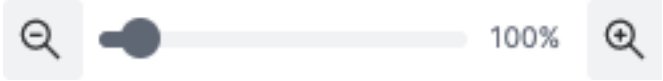
Annotate documents using the VGG Image Annotator (VIA) tool. Fill in the regions on the image and assign region or file attributes.

Tool Bar  : Save all the changes you have made to the annotation set within the project.

 : Export your annotation set as a JSON file to your local computer.

 : Display all files in a grid. From here, you can group images by file or region type and display images in collated groups.

 : Toggle the annotation editor open and closed.



: Zoom in and out of the image you are actively annotating by using either the buttons or the scroller.

: Select all the regions on the image you're actively working on. This is convenient if you're working with multiple of the same image with different information on each image since you can copy over all the regions at once.

Copy Selected : Copy the selected region(s).

Paste Selected : Paste the copied region(s).

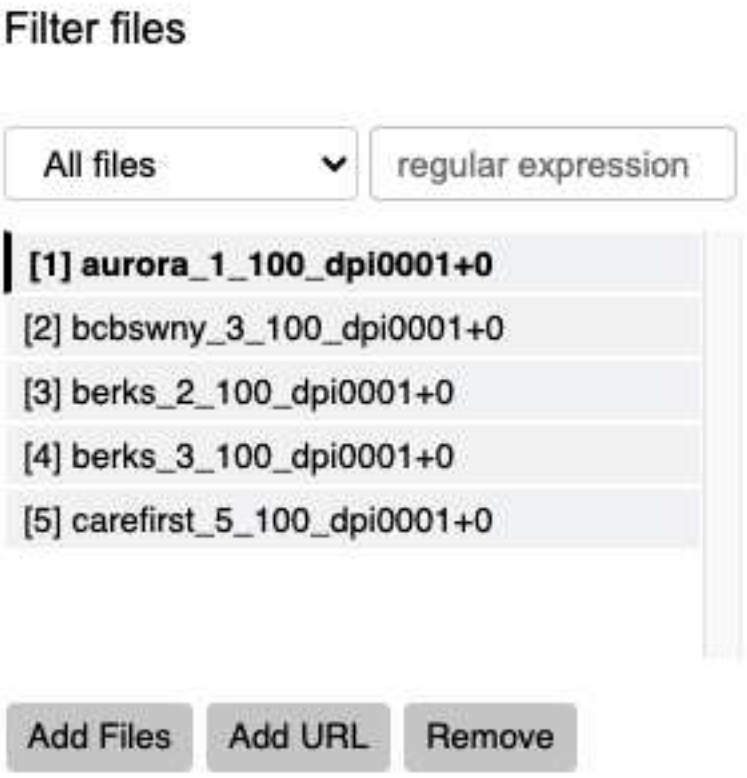
Paste To Many : Paste the copied region(s) to multiple images at once.

Undo Paste To Many : Undo the paste to many command.

Filter files This box lets you filter which files you want to see. One of:

- **All files** - shows all available files
- **Show files without regions** - shows files without any added bounding boxes
- **Show files missing region annotations** - shows files that have bounding boxes but are missing annotations for those boxes
- **Show files missing file annotations** - shows files that are missing file attributes
- **File that could not be loaded** - shows files that could not be loaded
- **Regular Expression** - search for a file or group of files by name

Here, you can import new files by choosing them from your local files or by providing a URL. You can also remove files by providing a file id, file name, and a number of regions.



Current workaround **Tip:** Workaround If you are adding a file that is not an image:

- First, export the PDF as a JPG or PNG.
- Then, add the file in edit in page view.

If the number of documents and pages does not update when you add a new file in edit in page view, refresh the page or click on another menu item (e.g. Document Sets) and click back to Annotation Sets to see the updated number of pages.

Attributes This box contains the ability to add region and file attributes. Toggle which one you are actively working with by selecting from the two options.

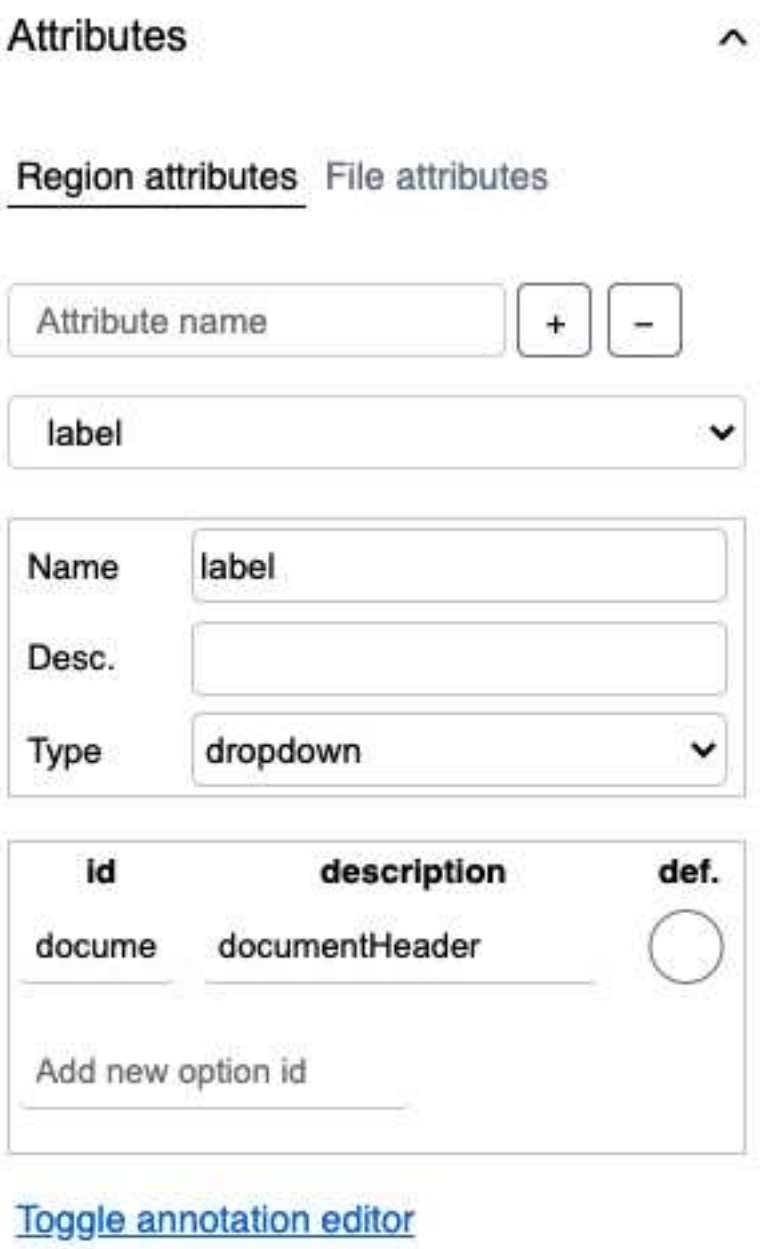
Note: You can edit or remove your annotations by clicking the **Annotations Editor page button** on the tool bar, by clicking the "Toggle annotation editor" note in the Attributes box, or by pressing the **space bar**.

Region attributes Region attributes allot attributes to specific regions on the page. Applying region attributes gives the annotation set the *label* attribute.

1. In the **Attributes** box, ensure that **Region attributes** is selected.
2. In the **Attribute name** field, enter "label". (**Note**: For this step, the name of the attribute must be set to "label" in order for H2O Document AI to work correctly).
3. Click + to add the new attribute.

For the new attribute, you can now create option ids.

1. Select the **attribute type** (one of: text, checkbox, radio, image, or dropdown)
    - **text** - assign an individual id to each region
    - **checkbox** - assign a single or multiple ids to each region from the created option id list
    - **radio** - assign a single id to each region from the created option id list
    - **image** - assign an image (provided via an image URL or b64) to each region from the created option id list
    - **dropdown** - assign a single id to each region from the created option id list
2. (checkbox, radio, image, dropdown only) Begin filling in the **option ids** (e.g. "PatientName" or "SupplierAddress") in the "Add new option id" box
    - (Optional) Provide a description of the option id
    - (Optional) Set an option id as the default value (**def**). This value will be the first to appear when selecting option ids for each region



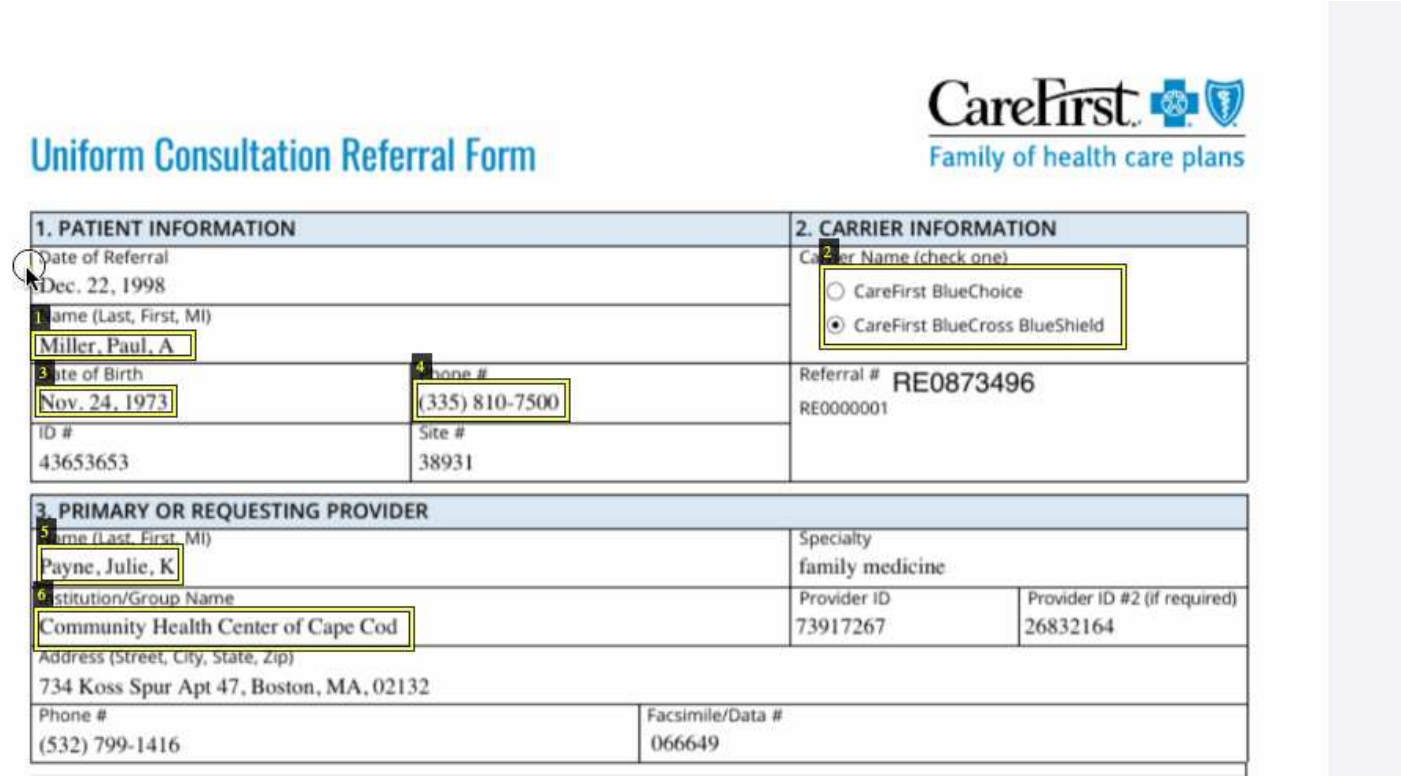
File attributes File attributes allot attributes for the entire page, not just a region drawn on a page (e.g. tagging a page as a medical referral). Applying file attributes gives the annotation set the *class* attribute.

1. Provide a **name** for the attribute
2. (Optional) Provide a description
3. Select the **attribute type** (one of: text, checkbox, radio, image, or dropdown)
4. (checkbox, radio, image, dropdown only) Begin filling in the **option ids** (e.g. "MedReferral" or "BirthCert") in the "Add new option id" box
5. Assign the file an attribute by opening the annotation editor (one way is to press the **space bar**). Go to the File Annotations section. Next to file name (the file you are currently on), choose a class for the file attribute or (text only) type in the class

Using bounding boxes to create regions On your document, find an area you want annotated. Draw regions on the image using bounding boxes:

1. **Press the left mouse button** on the corner of the region you want to bound
2. **Drag the mouse cursor** to encompass the object you are bounding
3. **Release** the mouse button

The region should still be higlighted (colored gray). You can now **select an option id** (checkbox, radio, image, dropdown only) from the available dropdown menu or **type the option id** into the text box (text only).



warning reminder **Save your annotations** before leaving Page View. This ensures your changes are saved and available to be loaded later for continued annotation.

Rename

Rename the annotation set and provide a new description.

Split

Split the annotation set.

1. Select the split ratio or input a custom split (either by regular expression or manually input the split)
2. Toggle whether you want to split on a document's boundaries
3. Provide a name prefix for the resulting annotation sets
4. (Optional) Provide a description

Export

Export the annotation set to your local computer.

Delete

Delete the annotation set. You will be prompted to acknowledge that the act of deletion is irreversible before you can delete your set

Using models

Interact with or create new models.

Train

Train a model based on your annotation sets. Training models is the Document AI job that consumes the most time.

1. Select the **model type**. One of:
   - Page Classification - requires the annotation set to have the *class* and *text* file attributes
   - Token Labeling - requires the annotation set to have the *label* and *text* file attributes
2. Select the **annotation set** you want to train the model
3. Select your base model. One of:
   - **layoutLM-base**: (default) takes document layout information into account for modeling. Useful for documents that have rich visual features (e.g. invoices, supply chain docs, ID cards)
   - **deberta-V3-base**: current SOTA model in English NLP field. Useful for plain documents (e.g. legal documents)
   - **multilingual-deberta-V3-base**: multilingual version of the deberta-V3-base model
4. Toggle whether you want to define the hyperparameters
5. Provide a **name** for the resulting model
6. (Optional) Provide a description of the model
7. Toggle whether you want to **Evaluate** the model
   - Provide a **name** for the Prediction Annotation set
   - Select the **Evaluation Annotation set**
8. Click **Train** to build the model

The model will appear on the Models page when it is finished. If you provided a validation set to build your model, you will also be able to access the **accuracy** of your model. You view the accuracy of your model on the prediction annotation set which is generated after your model is built. The prediction annotation set can be accessed from the Annotation sets page.

Setting your hyperparameters

You can use the default hyperparameters provided by the Document AI - Publisher UI or you can set your own. To set your own hyperparameters, toggle **Use Default Hyperparameters** off on the train model panel. These hyperparameters are used to fine tune the internal model on user-provided documents.

Batch size The batch size is a hyperparameter that defines the number of samples to work through before updating the model parameters. The batch size, or global training batch size, defines the number of documents (with their bounding box coordinates) that are passed over to the processing device (with CPUs or GPUs) at a time. If more than one GPU is available, then `per_device_batch_size` is determined by dividing the batch size by the number of GPUs.

Tip: Smaller batch sizes take longer to run. It takes a long time to reach the optimal solution, but it may converge faster to a good enough solution. The model starts learning with a small number of documents which prevents overfitting on the training dataset because the model only sees a small subset of data at a time. This will generalize well on the test set. For smaller training sizes (tens or hundreds of documents), we recommend using a lower batch size.

Large batch sizes reduce the training time of the model but risk the result of poor generalization or slow model convergence. These can also result in Out of Memory (OOM) errors. For larger training sizes (thousands of documents), we recommend using a slightly larger batch size to cut down on training time.

The default value for batch size is set to 4.

Epochs An epoch is the hyperparameter that defines the number of times that the learning algorithm works through the entire training dataset. An epoch is defined as one pass over all the training documents.

Tip: Increasing the number of epochs results in the model going from underfitting, to optimal fitting, to overfitting on the training data. A good number of epochs helps in reaching the optimal fit.

The default value for epochs is set to 25.

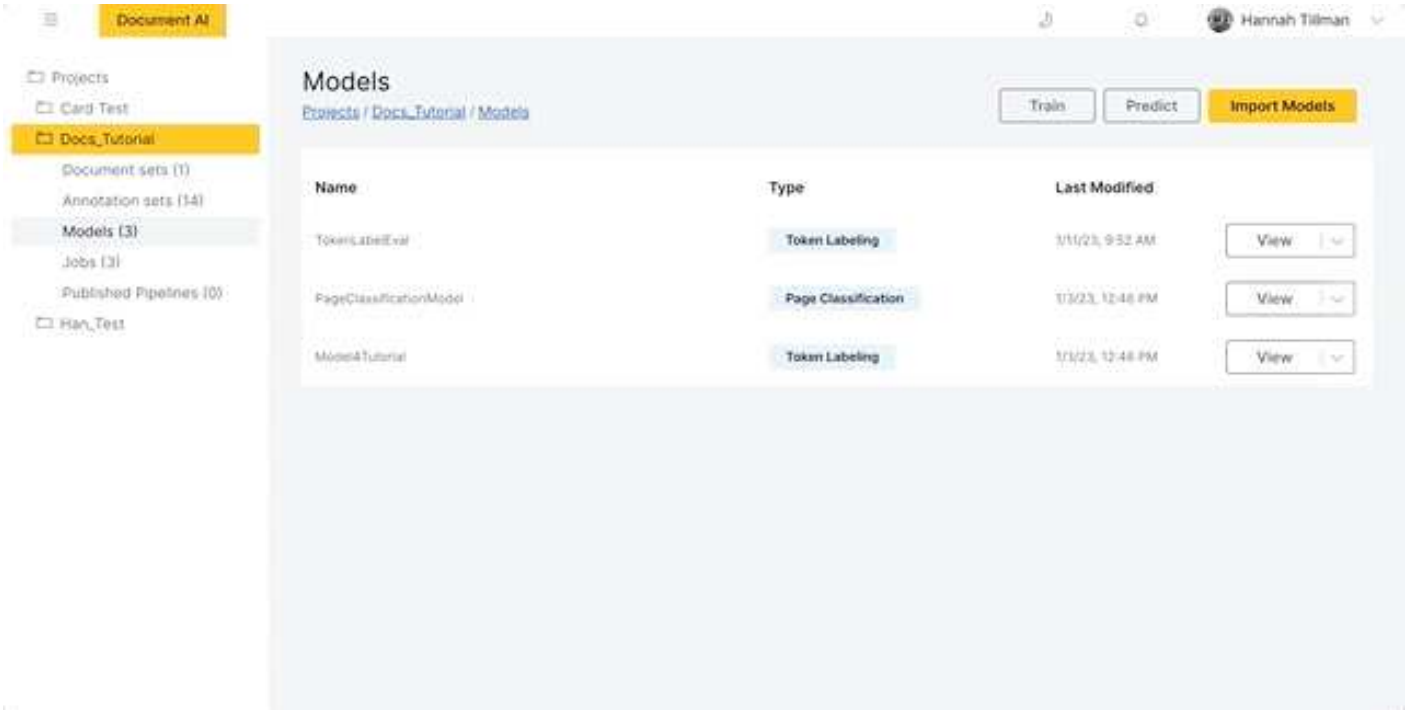
Learning rate The learning rate hyperparameter sets the learning rate for the AdamW optimizer.

Tip: The default value of `5e-5` works well for most models. Deberta models for page classification tasks tend to achieve better results using a lower rate (for example, `2e-5`).

Understanding accuracy

You receive accuracy information when you build a model using an evaluation set. To view the accuracy panel, navigate to the Annotation sets page. You can see the new prediction annotation set on the top row. It will have a quality score, which is the f1-score of the model that was applied to the dataset. Click **Info** on the prediction annotation set that was built with your model.

Click **Accuracy** located below the description box. You can either scroll through the accuracy results or click **Expand** for a larger pop-out of the accuracy panel. You can also download the results of the accuracy panel in a JSON file by clicking **Download** on the accuracy panel.



Accessing top N accuracy The accuracy panel by default shows you the results for all of the data. To view the top N accuracy results, click the **Accuracy result** dropdown and select the top N accuracy you want.

Accuracy metrics Three types of metrics are displayed on the accuracy panel: aggregate metrics, per class metrics, and the class confusion matrix.

- Aggregate metrics measure the accuracy of the entire dataset using the f1-score, precision, recall, and support measures.
- Per class metrics measure the accuracy of one class versus the rest of the datapoints in the dataset using the f1-score, precision, recall, and support measures.
- The class confusion matrix provides a tabular visualization of the model predictions (columns) versus the actual values (rows) for all the classes.

The accuracy panel differs between page classification models and token labeling models. For page classification models, the accuracy panel shows the metrics over different file attributes, and includes macro avg and weighted avg values for the aggregate metrics. For token labeling models, the accuracy panel shows the metrics over different region attributes, and includes macro avg, micro avg, and weighted avg values for the aggregate metrics.

Predict

Predict on an annotation set using a successfully built model.

1. Select the **model** to use for predictions

2. Provide a **name** for the prediction set
3. (Optional) Provide a description for the prediction set
4. Select the **Evaluation Annotation set**
5. Click **Predict** to retrieve the predictions

Your predictions will be available on the Annotation sets page when the job is finished running.

Import Models

Import a previously built model that you have saved to your local computer.

1. Provide a **name** for the imported model
2. (Optional) Add a description
3. **Drag and drop** the file or **browse your local files** for the zipped model file
4. Click **Import**

Interacting with a model

Each trained model has an **Info** button at the end of the row. Clicking **Info** will give you the details of your model (e.g. the description or base model). You can also find the logs for your model here. To see the full log, click **Expand**. You can also download the log by clicking **Download**. If you trained your model with a validation set, you can also access the accuracy of that model.

The drop-down arrow next to the **Info** button gives you the option to either rename, export, or delete your model.

Rename

Rename the model and provide a new description.

Export

Export a zip file of the model to your local computer.

Delete

Delete the model. You will be prompted to acknowledge that the act of deletion is irreversible before you can delete your model

Using jobs

The Jobs page displays currently active and completed Jobs.

From this page you can see the:

- Job ID
- Job type
- Status
- Created date & time
- Duration for which the job ran

Status

There are four status states on the Jobs page: *Pending*, *Done*, *Canceling*, and *Canceled*.

A *Pending* job is currently running. This job can be cancelled.

A *Done* job has finished running. It has either successfully run or failed. Check either the logs or the page the finished job printed to see whether the job was successful or not. You can now interact with the new product.

A *Cancelling* job is in the process of being canceled.

A job is marked *Canceled* when it is done being stopped.

Interacting with a job

Each job has an **Info** button at the end of the row. Clicking **Info** will provide you with the log for your job. To view the full log in a pop-out window, click **Expand**. You can also download a text file of your log by clicking **Download**.

Cancelling a job

You can cancel a *Pending* job from this page. Jobs with the *Pending* status are the only time a Job will have a drop-down arrow next to the **Info** button.

1. Click the drop-down arrow next to the **Info** button on the *Pending* job
2. Click **Cancel**

Using published pipelines

This page describes how to publish a scoring pipeline, how to use the pipeline to score new documents, and how to make curl requests to the published pipeline.

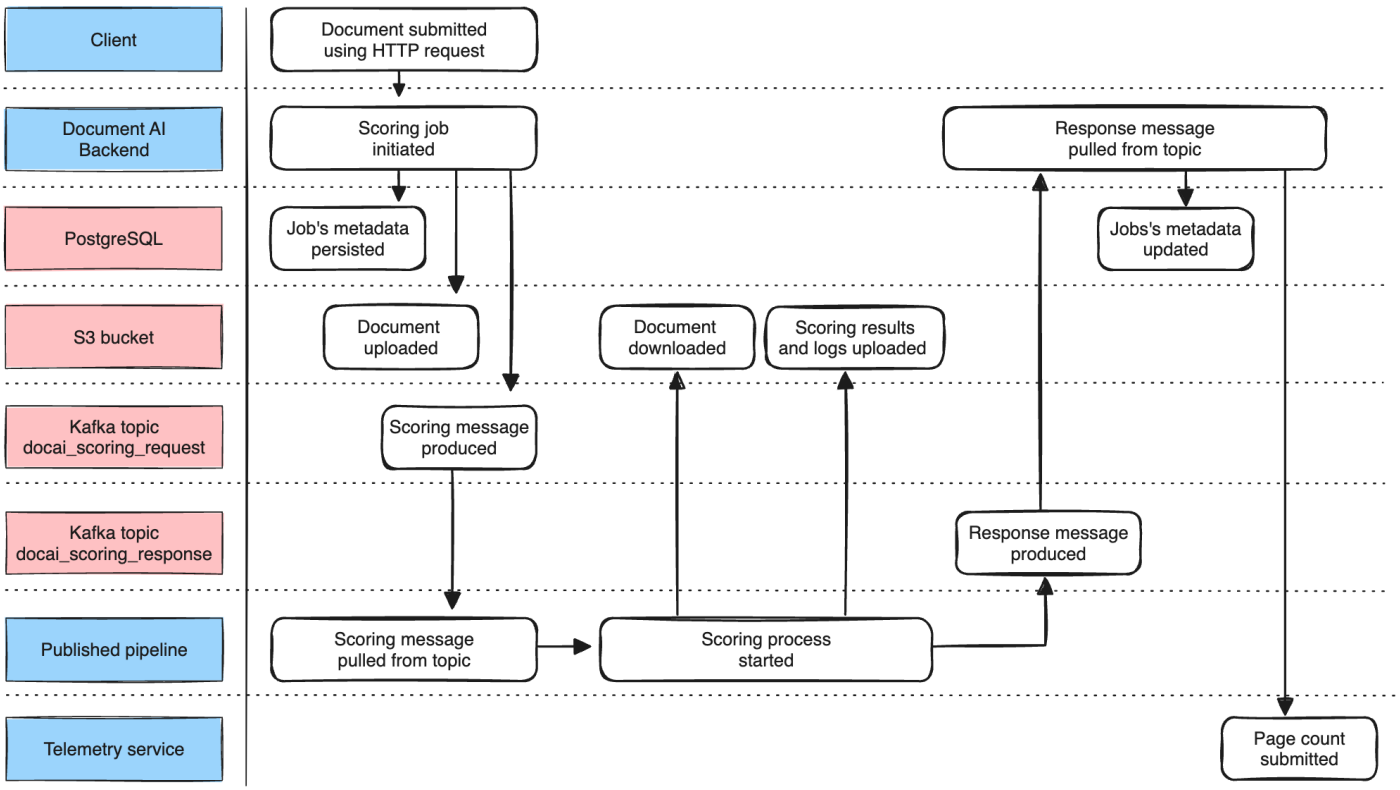
Process of document scoring

The following section outlines how documents are scored.

1. You submit a document (input) to the docai-backend using a multipart upload HTTP request. This request is handled by a single replica of the backend.
2. The backend creates corresponding job records in a PostgresSQL database and stores the uploaded document in an S3 bucket.
3. The backend publishes a scoring request message to a single partition of `docai_scoring_request` Kafka topic. This message contains only scoring job metadata like location of the input document, the name of the pipeline, etc.
4. A corresponding replica of each published pipeline (scorer) pulls the message from the connected Kafka partition. It checks whether the message is addressed to it by comparing the target pipeline name. The addressed scorer will start processing that document. Non-addressed scorers discard the message.
5. Before scoring a document, the scorer downloads the input document from the S3 bucket. At the end of a successful scoring, the scorer uploads the result and logs back to the S3 bucket and produces a message to `docai_scoring_response` Kafka topic.
6. The backend pulls that message and updates the job metadata in the PostgresSQL database. The backend also sends the number of processed pages to the Telemetry service.

Tip:

- A pipeline must be published in order to score a document.
- All published pipelines share one Kafka topic. Therefore, there must be enough partitions for the pipeline with the highest number of replicas. If the number of replicas is higher than the number of partitions, replicas that exceed this will not receive scoring messages. Keep in mind that the number of partitions can't be decreased, only increased.

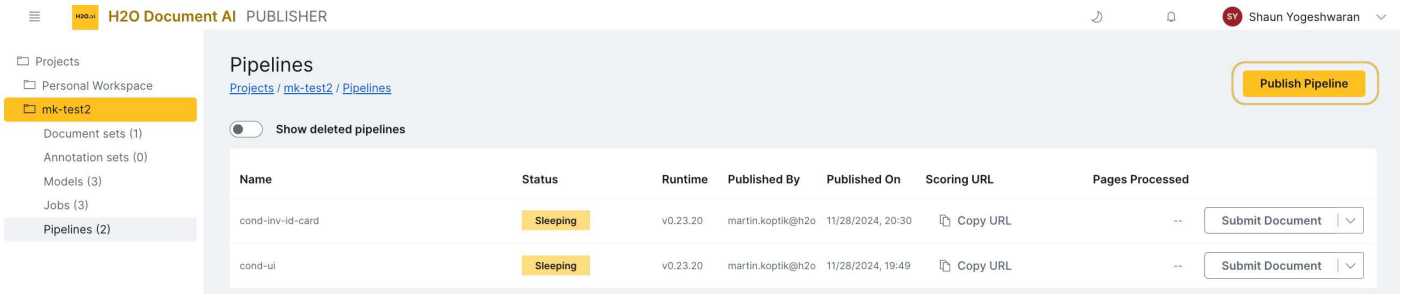


Publishing a scoring pipeline

The following steps describe how to publish a scoring pipeline.

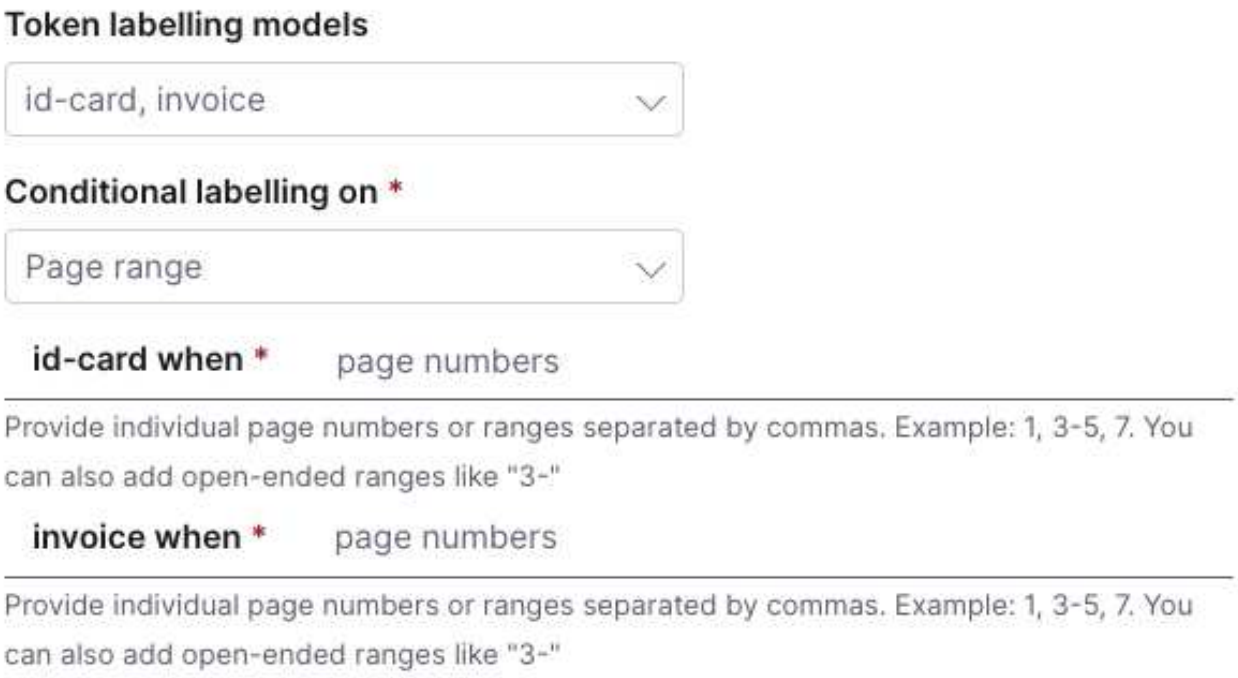
1. Navigate to the **Pipelines** page in the left navigation bar after training your model.

2. Click **Publish Pipeline**.



3. Provide a name for your pipeline. The name has to be unique and can only support lowercase letters, digits, and the dash ("-") symbol. 4. (Optional) Toggle to **mute your logs** if you need to protect sensitive information. 5. Next, under the **Pipeline** section, you can either use the basic configuration or use a custom configuration. **Note:** When using a custom configuration, changes will be reflected in the **YAML** file found under the **Custom Config** menu.

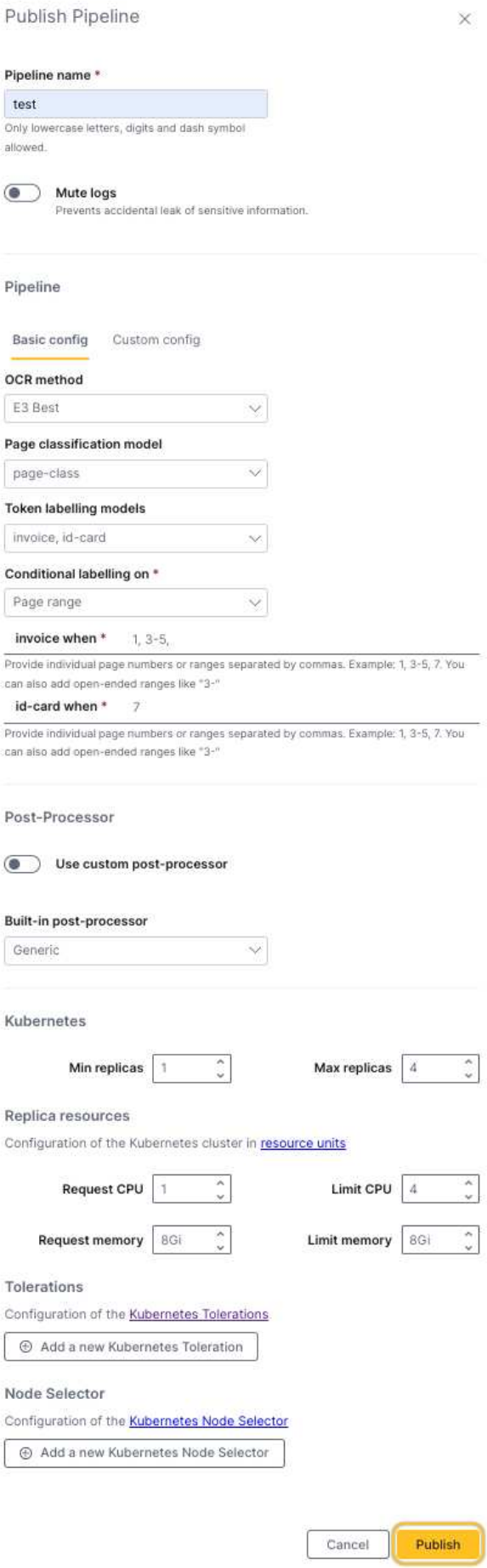
6. Select which optimal character recognition (OCR) method you want your Pipeline to use for scoring from the menu. Use a different method for scoring than you used for training.
7. Choose a **Page classification model** from the drop-down menu if required. This model categorizes pages based on their content type, improving processing accuracy.
8. Choose a **Token labeling model(s)** from the drop-down menu. **Note:** If you are choosing multiple labeling models, select a condition from the dropdown menu and specify the criteria for each labeling model.



9. Select a post-processor:
   - Generic
   - Supply-chain
   - Create a custom post-processor
10. Configure how your scoring pipeline is scheduled:
    - Kubernetes:
      - **Min replicas**: Set the minimum number of replicas required
      - **Max replicas**: Set the maximum number of replicas to handle traffic spikes
    - Replica resources:
      - CPU:
        - **Request CPU**: Allows pipeline to be provisioned so long as the requested amount is available

    - **Limit CPU**: If the limit is reached, the pipeline is throttled down to the limit
  - Memory:
    - **Request memory**: Amount of memory required to schedule the pipeline
    - **Limit memory**: If the limit is exceeded, the pod will be killed and the pipeline will be restarted
- Tolerations: (Optional) Nodes you want to schedule your scoring pipeline to
- Node selector: (Optional) Select nodes that have specific labels on them

11. Click **Publish**.



Pipeline information A published pipeline provides the

following information:

- Name: The name of the pipeline.
- Status: The curent state the pipeline is in.
- Runtime: The version of the pipeline.
- Published By: The user who published the pipeline.
- Published On: The date and time the pipeline was published.
- Scoring URL: The scoring URL you need for the bulk scorer.
- Pages Processed: The number of pages this pipeline has processed.

The following are the available states for the pipelines:

- Starting: Pipeline is attempting to start.
- Pending: Pipeline is unable to attempt starting (usually due to insufficient resources or incorrect Kubernetes settings). Try editing to restart it.
- Running: Pipeline successfully started and is ready to score documents.
- Failing: Pipeline that is failing to start (perhaps due to wrong YAML configuration or wrong post-processor).
- Deleted: Pipeline that has been unpublished and can no longer be edited.

Edit your pipelines You can edit everything about your pipeline except the name. You can interact with pipelines in any state except for a pipeline that you have unpublished. If you want to change the name of your pipeline you have to make a new pipeline.

Tip: When editing your pipeline, there is additional information about it at the top of the editing panel in a blue box. This can be helpful to figure out why your pipeline won't start.

To edit your pipeline, do the following:

1. Click the **dropdown arrow** at the end of your pipeline.
2. Select **Edit**.
3. Make your desired edits.
4. Click **Edit** to save your changes.

Pipeline logs

You can access logs for your pipeline. Pipeline logs are seen next to the failing status, and you can access them by clicking on the log. This can help you debug the reason your pipeline failed.

Upgrade your pipelines When upgrading Document AI to a new version (e.g. $0.7 > 0.8$) Document AI will automatically upgrade all existing pipelines to the new version as well. The pipelines using the older Document AI version also use the old runtime, but the new version installation will attempt to update them. If the pipeline does not start once it has been updated to the new version, then the pipeline will be rolled back to the previous version.

Revert to a previous version of your pipeline You can revert your pipeline to a previous version of that pipeline. You can see all the previous versions of a pipeline (similar to how you can see prior edits) which will let you rollback your pipeline to a previous version if your upgraded pipeline is failing.

Deleted pipelines Toggle on **Show deleted pipelines** to see your pipelines in the deleted state. Deleted pipelines cannot be interacted with. You can, however, make a new pipeline using the same name as the deleted pipeline. If you use the same name as a deleted pipeline, you won't reuse anything else from that deleted pipeline. That deleted pipeline will just be overwritten.

To delete your pipeline, do the following:

1. Click the **dropdown arrow** at the end of your pipeline.
2. Click **Unpublish**.
3. Select the check that your acknowledge that this act is destructive and irreversible.
4. Click **Unpublish**.

Put your pipeline to sleep You can provide a setting to put your pipeline to sleep if a document has not been submitted in a certain amount of time. Sleeping pipelines don't use resources because they're scaled down to 0 replicas.

Wake Up sleeping pipelines Wake your pipeline back up by submitting a document to it. The submitted document will stay in a queue before the pipeline wakes back up. Waking up pipelines go from *starting* to *running*.

Configuring the sleeping pipelines feature You can configure the sleeping pipelines feature globally via Helm during the Document AI installation. This feature applies the same settings to all pipelines.

1. **Enable the feature**: By default, this feature is turned off. To use it, enable it in the Helm configuration.
2. **Set Idle Time**: Define the idle time after which the controller scales the pipeline down to 0 replicas.

UI behavior

- Pipelines scaled down to 0 remain visible in the UI with the status **Sleeping**.
- Users can submit documents to sleeping pipelines. The system queues these documents and wakes up the pipeline.

Document submission and wake-up process

- Once a sleeping pipeline is woken up, it retrieves the queued document and begins processing.
- This process works the same whether a document is submitted via the UI or API.
- The wake-up time for a pipeline is nearly identical to the time required to publish a new pipeline.

Post-processors

When publishing a pipeline, you need to add a post-processor. You can either use one of the built-in post-processors (i.e. Generic or Supply-chain) or you can write your own custom post-processor. Both built-in post-processors merge individual tokens that were both predicted as the same label. For example, if token `John` and token `Smith` are both predicted as label `customer_name` they will be merged to the single prediction `John Smith`.

To access additional post-processor recipes, see the H2O Document AI Recipe Repository.

Generic The Generic post-processor includes Top N and the ability to produce an image snippet. Top N delivers a second view of the data that shows what the top predicted class will be as well as the second most likely and third most likely. Image snippet returns a cropped image of each prediction rectangle in a byte string. The Generic post-processor produces huge byte strings which can make it difficult to find the real predictions.

Supply-chain The Supply-chain post-processor includes line ID groupings. Line ID groupings are used to group predictions together (e.g. item ID, price, and quantity).

Table support H2O Document AI - Publisher offers table support for your documents. To use the table support feature, you need to use a slightly modified version of the supply-chain post-processor. This addition adds the *tableID* element which is needed for table support to work.

This code takes the *class* type to label each interesting column of the table. The model then predicts all entities with the extra medatata: *tableId* and *lineID*. If the entity belongs to the table, the *tableID* and *lineID* metadata are non-empty.

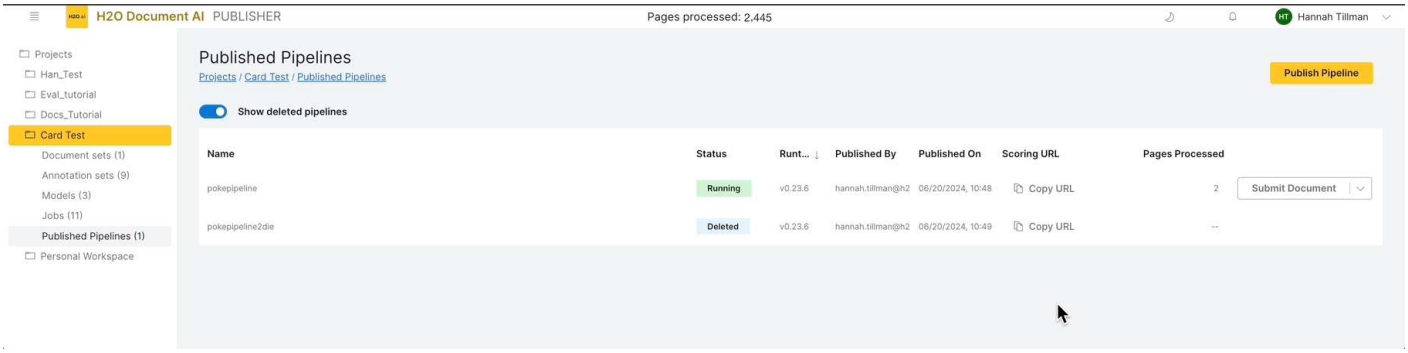
Note:

- *tableID* denotes the table
- *lineID* denotes the row
- *class* denotes the column
- *entity* is the text predict for the cell

Submitting new documents to the pipeline

After your pipeline has been published and is Running, you can use it to score new documents. Track the progress of your submitted file under background processes. You access background processes by clicking the bell icon next to your account name.

When the document you submitted to the pipeline has finished processing, you can access the result (a JSON response) from background processes by clicking the "View" button. This will bring you to the Scoring Results page. You can access the logs from the scoring results page, too.



Universal scoring pipeline

You can construct your own custom pipeline in the UI. This is known as the universal scoring pipeline (USP). The USP grants you more flexibility than is available in the UI alone. All available functionalities exist as processors, and the pipeline module orchestrates them by following the configurations you set via YAML. This lets you chain together many processors to perfrom a single scoring task, thus creating a single pipeline.

The USP affords several interesting functionalities that traditional pipelines do not, such as:

- conditional processing: run a page classifier first and send documents in different classes to separate token classifiers,
- non-text object detection with regular text OCR methods,
- OCR-only pipelines,
- and many more possibilities!

Pipeline flow

The pipeline is broken down into individual tasks with a single processor per task. The tasks are carried out sequentially. The `Intake` processor ingests the input documents and returns an annotation set wth the documents to be processed further downstream. Then, each subsequent processor ingests a list of annotation sets, performs its process, then outputs a list of annotation sets for the next processor to process. At the end of the pipeline, a post processor is needed to translate the annotation sets into a JSON file for consumption.

There are processors that are highly common amongst most pipelines such as `Intake` and `PdfExtract`, but the beauty of the USP is its highly customizable nature.

Accessing the universal scoring pipeline

To use the USP: 1. Navigate to the Publish Pipelines panel. 2. Click **Publish Pipeline**. 3. Toggle on **Use custom pipeline**.

This gives you access to the config YAML file that allows you to construct your custom pipeline. When you publish your pipeline, it will read the YAML file when it runs.

If you set options in the UI panel, the YAML file will read them. For example, if you select the `Tesseract` option for OCR, it will have that preset as the OCR option for your Pipeline in the custom pipeline. To update the USP with the changes you select in the Publish Pipelines panel, toggle the **Use custom pipeline** off and back on. The changes you select will now be reflected in the YAML file.

Of course, you do not have to use any of the options in the Publish Pipelines panel. You can set all of your processors directly in the YAML file.

USP and custom post processors

The USP can utilize your custom post processors as well as the ones that come pre-baked into the UI. To use a custom post processor, toggle on **Use custom post-processor**. Any custom post processor you write or upload will be directly linked to the YAML file as soon as you toggle on **Use custom pipeline**. Your post processor will be the last task executed.

H2O provided pipelines and processors H2O provides several premade pipelines and post processors for you to use.

Hybrid OCR Hybrid OCR reads the embedded text directly and passes only images into the OCR engine. This is useful in instances where most of the content of the document is embedded text but there are some embedded images with text (like a logo) which is not part of the embedded text content. If you were to just use a PDF text extractor, you would miss out on the text present in the images. It could also be very time consuming to run OCR for both the embedded text and the text inside the image. The hybrid OCR processor is an in-between ground where you can utilize the efficiency of PDF text extraction without missing out on the text in logos or other text present in images.

```
# Hybrid OCR processor:
argus.pipeline -t
argus.processors.ocr_processors.Intake
root_docs_path=/Users/raji/Downloads/test-hybrid-ocr
follow_symlinks=true
-t
argus.processors.ocr_processors.PdfTextExtract
include_embedded_images=True
-t
argus.processors.ocr_processors.NormalizeImages
resample_to_dpi=300
normalize_image_format=.png
-t
argus.processors.ocr_processors.GenericOcr
ocr_style=BestTextExtract
-f
my_via
type=Via
-o
try_ocr
```

Note: You must set `include_embedded_images=True` in the PdfTextExtract step in order to perform OCR on the images. The OCR engine is the one mentioned in the following step:

```
argus.processors.ocr_processors.GenericOcr
ocr_style=BestTextExtract
```

Manipulating artifacts

When you open a new Publish Pipelines panel and toggle on **Use custom pipeline** without selecting a model option, the artifacts will be empty in the pipeline. Selecting a model to use for the pipeline will fill the artifacts with the source URL of the model and the unique name of that model. This artifact information will also populate the the Predict task which is the processor task that has your model run predictions. You can fill in this information yourself if you have a model you want to use instead of having the UI prefill this information for you.

Universal scoring pipeline example

Let's look at an example of a custom pipeline and walk through how it works. The following pipeline is USP-exclusive feature: an OCR-only pipeline.

```
spec: # A general pipeline example that only does OCR on a set of data

  pipeline:
    steps:

    - tasks:
       - name: "Intake" # name of task
         type: PipelineTask # if absent, defaults to PipelineTask.  Can also be PipelineReorderInputs and Inp
         class: argus.processors.ocr_processors.Intake # fqn_of_Processor class
         parameters:
           root_docs_path: /input/path
           follow_symlinks: true

    - tasks:
```

```
    - name: "PdfExtract"
      type: PipelineTask
      class: argus.processors.ocr_processors.PdfTextExtract

- tasks:
    - name: "ImageNormalize"
      class: argus.processors.ocr_processors.NormalizeImages
      parameters:
        resample_to_dpi: 300
        normalize_image_format: .jpg

- tasks:
    - name: "OCR"
      class: argus.processors.ocr_processors.GenericOcr
      parameters:
        ocr_style: DocTROcr

- tasks:
    - name: "PostProcess"
      class: argus.processors.post_processors.ocr_only_post_processor.PostProcessor
      parameters:
        output_format: 'json'
```

The first task is `Intake` which lists all the documents that need to be processed by processors further downstream. The second task is `PdfExtract` which attempts to extact the text on a page, and if it can't, leaves the page untouched. The third task is `ImageNormalize` which normalizes the format and DPI and validates the image. The fourth task is `OCR` which will, in this case, run the docTR OCR method. The final task is the post processor: you need a post processor to translate the data from an annotation set to a JSON file for use.

The `MiniProgram` processor

The `MiniProgram` processor is a special type of processor that lets you write small "in-between" steps without implementing a full-fledged processor. It runs once for every page and modifies all aspects of the annotation set (both the current document and the current page) by:

1. Defining a local variable for each page which allows it access to the current page/document/annotation set data structure;
2. Running the `MiniProgram` processor (which possibly modifies the local variables);
3. Updating the output annotation set from the local variables after `MiniProgram` processor finishes.

The `when` parameter All processors take a `when` parameter (in addition to their processor-specific parameters). The `when` parameter's value is a slightly extended mini program that allows you to:

- run processors conditionally on some pages or documents depending on the information contained in the input, or to drop them from the output annotation sets.
- manipulate an annotation set in the same way as a regular mini program.

Accessing a scoring pipeline via curl

The following sections describe how to access published pipelines by sending API requests using curl.

Authentication

The following is a sample curl command to retrieve an access token from an identity provider.

Note:

- The access token has a short lifetime of approximately five minutes and may expire while a document is being processed. If this occurs, rerun the curl command to retrieve a new access token. The processing of the document is not affected by the token expiry.
- The access token that is returned by this command must be included in all requests to the proxy.

```
ACCESS_TOKEN=$(curl -X POST 'http://keycloak.34.211.115.161.nip.io/auth/realms/wave/protocol/openid-
connect/token' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-d 'password=REPLACE_ME' \
-d 'username=REPLACE_ME' \
-d 'grant_type=password' \
-d 'response_type=token' \
-d 'client_id=admin-cli' | jq -r .access_token)
```

Listing published pipelines

To retrieve a list of published pipelines, run the following curl command:

```
curl https://document-ai-scorer.cloud-qa.h2o.ai/pipeline -H "Authorization: Bearer ${ACCESS_TOKEN}"
```

The following is the expected response to the preceding command:

```
[{"name":"ak-pipeline-test","scoringUrl":"https://document-ai-scorer.cloud-qa.h2o.ai/async/model/ak-
pipeline-test/score"}]
```

Submitting documents to a scoring pipeline

To retrieve a list of published pipelines, run the following curl command:

```
curl -v https://document-ai-scorer.cloud-qa.h2o.ai/async/model/bd-trial/score \
-F documentGuid=cdbd0e44-2672-4d63-94dd-9afb110547ec \
-F document=@./test_api_svcs.pdf \
-H "Authorization: Bearer ${ACCESS_TOKEN}"
```

The following is the expected response to the preceding command:

```
{"jobUri":"https://document-ai-scorer.cloud-qa.h2o.ai/job/adde851e-b9b4-11ec-9ceb-
fe652bf3bfb4","jobId":"adde851e-b9b4-11ec-9ceb-fe652bf3bfb4"}
```

Checking pipeline status

```
curl https://document-ai-scorer.cloud-qa.h2o.ai/job/adde851e-b9b4-11ec-9ceb-fe652bf3bfb4 -
H "Authorization: Bearer ${ACCESS_TOKEN}"
```

The following is the expected response to the preceding command:

```
{"status":"succeeded","resultUri":"https://document-ai-scorer.cloud-qa.h2o.ai/job/adde851e-b9b4-
11ec-9ceb-fe652bf3bfb4/result"}
```

Retrieving a prediction response

To retrieve a prediction response from a published scoring pipeline, run the following curl command after the status of a pipeline has changed to `succeeded`:

```
curl https://document-ai-scorer.cloud-qa.h2o.ai/job/adde851e-b9b4-11ec-9ceb-fe652bf3bfb4/result -
H "Authorization: Bearer ${ACCESS_TOKEN}"
```

The following is the expected response to the preceding command:

```
{"documentGuid":"some-id","entityConfidences":[],"modelMetadata":{"version":"370f1c"},"pageFailures":[],"page
```

Template method

The template method lets you easily access the target text in a document by knowing its coordinates ahead of time. The template method is best used for cases where you have many documents from the same vendor (or of the same format) and expect many more in the future. Due to the repetitive nature of these documents, the location of the target text is predictable, so you don't need a model to predict its location. Instead, you can have the template look at the text in the assigned coordinates. This allows the template to be more accurate than a model.

To use the template method, contact the H2O Document AI team to help you set up your template.

Using the template method

After your template has been created, you need to navigate to the Published Pipelines page to utilize it.

1. Click **Publish Pipeline**.
2. Select an OCR method that uses PDF text extraction (for example, E3 Best).
3. Add a model (either a finetuned or a dummy model). **Tip:** If you encounter a document that was not templated, a finetuned model will provide you with reasonable output. A dummy model will provide you with junk output.
4. Provide a pipeline name.
5. Toggle on **Use custom post-processor**. Paste your template code into the exposed post-processor.
6. Click **Publish**.

This will create a pipeline that utilizes your template. You can submit documents to your published pipeline by clicking the **Submit Document** button at the end of the row, or you can use the bulk scorer to submit many documents at once.

Using the H2O Document AI - Bulk Scorer

Scoring documents in the UI can be time consuming. To score large amounts of documents, use the H2O Document AI - Bulk Scorer, which is run from the command line using either a Docker or Python environment. This is because the H2O Document AI - Bulk Scorer is currently separate from the main H2O Document AI UI.

Note: When using the H2O Document AI - Bulk Scorer, you can run the same job multiple times. Running the same job creates a new output file (that is, no files are overwritten).

Download the H2O Document AI - Bulk Scorer

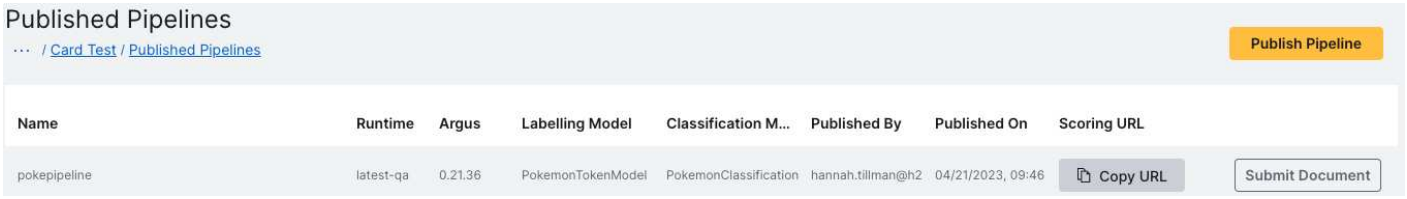
Use either of the following download links to gain access to the H2O Document AI - Bulk Scorer:

- Docker Image
- Python Wheel

You can also contact the H2O Document AI team.

Retrieve your scoring URL

Access your scoring URL from the H2O Document AI - Publisher Published Pipelines panel.



Start the H2O Document AI - Bulk Scorer

Docker

```
# Using H2O Document AI - Bulk Scorer with Docker
# Step 1: Download the Docker image archive
wget https://s3.amazonaws.com/artifacts.h2o.ai/releases/ai/h2o/document-ai-bulk-scorer/rel-
v0.2.5/docker/docai-scorer_0.2.5_docker.tar.gz

# Step 2: Load the Docker image from the archive
docker load
**Python**
```

Using H2O Document AI - Bulk Scorer with Python

Step 1: Download the Python wheel

wget https://s3.amazonaws.com/artifacts.h2o.ai/releases/ai/h2o/document-ai-bulk-scorer/rel-v0.2.5/python/py3.8/docai_scorer-0.2.5-py3-none-any.whl

Step 2: Ensure Python 3.8 is available

python -V

Step 3: Create a virtual environment

python -m venv .venv

Step 4: Activate your virtual environment

source .venv/bin/activate

Step 5: Upgrade pip if needed

pip install –upgrade pip

Step 6: Install the Python wheel**

pip install docai_scorer-0.2.5-py3-none-any.whl

Step 7: Test your installation

docai –help

```
### Retrieve a list of all available commands
If you don't know where to begin after starting the H2O Document AI - Bulk Scorer, the help command retrieves
```

–help

```
## Configuration
The H2O Document AI - Bulk Scorer can be configured using a YAML file. The file location defaults to `./confi
config` command.

Configuration options provided in the `config.yaml` can be overridden using environmental variables or comman
the-location-of-the-files) option is provided in multiple places, the option set in the command line takes th

The [H2O Document AI - Bulk Scorer example section](#h2o-document-ai---bulk-scorer-
example) contains an example `config.yaml` file.

### Authentication
Authentication can be done in two ways:
1. Through username/password authentication using `docai_user` and `docai_password` + Keycloak.
2. Through Managed Cloud authentication for SSO support. These values can be obtained from the "Accessing H2O

If both options are provided, then the second option will be used.

### Authentication with username/password
You can specify your authentication credentials in the config.yaml file, in the environmental variables, or f

### DocAI user

**Docker**
```

setting from the config.yaml file:

docai_user: “<>”

setting from environmental variables:

-e DOCAI_USER=<>

setting from the command line:

–docai_user <>

Python

setting from the config.yaml file:

docai_user: “<>”

setting from environmental variables:

run before the docai command:

export DOCAI_USER=<>

run in front of docai command:

DOCAI_USER=<> docai ...

setting from the command line:

–docai_user <>

DocAI password

Docker

setting from the config.yaml file:

docai_password: “<>”

setting from environmental variables:

-e DOCAI_PASSWORD=<>

setting from the command line:

–docai-password <>

Python

setting from the config.yaml file:

docai_password: “<>”

setting from environmental variables:

run before the docai command:

export DOCAI_PASSWORD=<>

run in front of `docai` command:

DOCAI_PASSWORD=<> docai …

setting from the command line:

–docai-password <>

Auth base URL

Docker

setting from the config.yaml file:

auth_base_url: “<>”

setting from environmental variables:

-e DOCAI_AUTH_BASE_URL=<>

setting from the command line:

–auth-base-url <>

Python

setting from the config.yaml file:

auth_base_url: “<>”

setting from environmental variables:

run before `docai` command:

export DOCAI_AUTH_BASE_URL=<>

run in front of `docai` command:

DOCAI_AUTH_BASE_URL=<> docai …

setting from the command line:

–auth-base-url <>

Keycloak client ID

Docker

setting from the config.yaml file:

keycloak_client_id: “<>”

setting from environmental variables:

-e DOCAI_KEYCLOAK_CLIENT_ID=<>

setting from the command line:

–keycloak-client-id <>

Python

setting from the config.yaml file:

keycloak_client_id: “<>”

setting from environmental variables:

run before `docai` command:

export DOCAI_KEYCLOAK_CLIENT_ID=<>

run in front of `docai` command:

DOCAI_KEYCLOAK_CLIENT_ID=<> docai ...

setting from the command line:

–keycloak-client-id <>

Keycloak realm

Docker

setting from the config.yaml file:

keycloak_realm: “<>”

setting from environmental variables:

-e DOCAI_KEYCLOAK_REALM=<>

setting from the command line:

–keycloak-realm <>

Python

setting from the config.yaml file:

keycloak_realm: “<>”

setting from environmental variables:

run before the docai command:

export DOCAI_KEYCLOAK_REALM=<>

run in front of docai command:

DOCAI_KEYCLOAK_REALM=<> docai …

setting from the command line:

–keycloak-realm <>

Authentication for Managed Cloud
The following authentication commands provide SSO support for the Managed Cloud. These commands can be specif

Note:
If you provide username/password authentication as well as authentication for the Managed Cloud, then the aut

Platform token

Docker

setting from the config.yaml file:

platform_token: “<>”

setting from environmental variables:

-e DOCAI_PLATFORM_TOKEN=<>

setting from the command line:

–platform_token <>

Python

setting from the config.yaml file:

platform_token: “<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_PLATFORM_TOKEN=<>

run in front of `docai` command:

DOCAI_PLATFORM_TOKEN=<> docai ...

setting from the command line:

–platform_token <>

Token endpoint URL

Docker

setting from the config.yaml file:

token_endpoint_url: “<>”

setting from environmental variables:

-e DOCAI_TOKEN_ENDPOINT_URL=<>

setting from the command line:

–token_endpoint_url <>

Python

setting from the config.yaml file:

token_endpoint_url: “<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_TOKEN_ENDPOINT_URL=<>

run in front of `docai` command:

DOCAI_TOKEN_ENDPOINT_URL=<> docai ...

setting from the command line:

–token_endpoint_url <>

Platform client ID

Docker

setting from the config.yaml file:

platform_client_id: “<>”

setting from environmental variables:

-e DOCAI_PLATFORM_CLIENT_ID=<>

setting from the command line:

–platform_client_id <>

Python

setting from the config.yaml file:

platform_client_id: “<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_PLATFORM_CLIENT_ID=<>

run in front of `docai` command:

DOCAI_PLATFORM_CLIENT_ID=<> docai …

setting from the command line:

–platform_client_id <>

Input and output

The following input/output commands can be specified in the `config.yaml` file, in the environmental variable

Inputting the location of the file(s)

This command queues a path to an image file, a directory with images, or a zip file with images for the H2O D
i <image_file> -i <image_directory> -i <image_file>`).

Docker

setting in the config.yaml file:

images: <>

setting from environmental variables:

-e DOCAI_IMAGES=<>

setting from the command line:

–images <> -i <>

Python

setting from the config.yaml file:

images: <>

setting from environmental variables:

run before the `docai` command:

export DOCAI_IMAGES=<>

run in front of `docai` command:

DOCAI_IMAGES=<> docai ...

setting from the command line:

–images <> -i <>

Set the allowed file types
By default, the H2O Document AI - Bulk Scorer can read the following file types:

- PDF
- PNG
- JPEG
- JPG
- BMP
- TIFF
- GIF

However, if you want the H2O Document AI - Bulk Scorer to only score certain file types (for example, JPEG an

Docker

setting from the config.yaml file:

valid_image_file_extensions:

- “<>”

setting from environmental variables:

-e DOCAI_VALID_IMAGE_FILE_EXTENSIONS=<>

setting from the command line:

–valid-image-file-extensions <>

Python

setting from the config.yaml file:

valid_image_file_extensions:

- “<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_VALID_IMAGE_FILE_EXTENSIONS=<>

run in front of `docai` command:

DOCAI_VALID_IMAGE_FILE_EXTENSIONS=<> docai …

setting from the command line:

–valid-image-file-extensions <>

Input an encrypted zip file
This command lets you input an encrypted zip file.

Docker

setting from the config.yaml file:

zip_password: “<>”

setting from environmental variables:

-e DOCAI_ZIP_PASSWORD=<>

setting from the command line:

–zip-password <>

Python

setting from the config.yaml file:

zip_password: “<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_ZIP_PASSWORD=<>

run in front of `docai` command:

DOCAI_ZIP_PASSWORD=<> docai …

setting from the command line:

–zip-password <>

Input multiple encrypted zip files
This command lets you input multiple encrypted zip files.

Docker

setting from the config.yaml file:

zip_password_file: “<>”

setting from environmental variables:

-e DOCAI_ZIP_PASSWORD_FILE=<>

setting from the command line:

–zip-password-file <>

Python

setting from the config.yaml file:

zip_password_file: “<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_ZIP_PASSWORD_FILE=<>

run in front of `docai` command:

DOCAI_ZIP_PASSWORD_FILE=<> docai …

setting from the command line:

–zip-password-file <>

```
### Filter input image list
This command can filter the inputted image list. For example, if you set `--
regex ".*1[.](jpeg|pdf|png)$"` in the command line, then the H2O Document AI - Bulk Scorer only selects image
```

```
**Docker**
```

setting from the config.yaml file:

regex: “<>”

setting from environmental variables:

-e DOCAI_REGEX=<>

setting from the command line:

–regex “<>”

```
**Python**
```

setting from the config.yaml file:

regex: “<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_REGEX=<>

run in front of `docai` command:

DOCAI_REGEX=<> docai …

setting from the command line:

–regex “<>”

Provide an output directory
This command provides a directory to save the results of the H2O Document AI - Bulk Scorer job.

Docker

setting from the config.yaml file:

out_dir: “<>”

setting from environmental variables:

-e DOCAI_OUTPUT_DIR=<>

setting from the command line:

–output-dir <> -o <>

Python

setting from the config.yaml file:

out_dir: “<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_OUTPUT_DIR=<>

run in front of `docai` command:

DOCAI_OUTPUT_DIR=<> docai ...

setting from the command line:

–output-dir <> -o <>

Provide a temporary image directory
This command provides a temporary image directory for unzipped files.

Docker

setting from the config.yaml file:

temp_image_dir: “<>”

setting from environmental variables:

-e DOCAI_TEMP_IMAGE_DIR=<>

setting from the command line:

–temp-image-dir <>

Python

setting from the config.yaml file:

temp_image_dir: “<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_TEMP_IMAGE_DIR=<>

run in front of `docai` command:

DOCAI_TEMP_IMAGE_DIR=<> docai ...

setting from the command line:

–temp-image-dir <>

Pipeline

The following pipeline commands can be specified in the `config.yaml` file, in the environmental variables, c

Scorer base URL

In order to access a pipeline, you need to provide the base URL for the scorer. You can [access the base scor
your-scoring-url) from the H2O Document AI - Publisher UI on the Published Pipelines page.

Docker

setting from the config.yaml file:

scorer_base_url: “<>”

setting from environmental variables:

-e DOCAI_SCORER_BASE_URL=<>

setting from the command line:

–scorer-base-url <>

Python

setting from the config.yaml file:

scorer_base_url: “<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_SCORER_BASE_URL=<>

run in front of `docai` command:

DOCAI_SCORER_BASE_URL=<> docai …

setting from the command line:

–scorer-base-url <>

Provide a pipeline to use for scoring
This command provides the pipeline to use for scoring the new documents.

Docker

setting from the config.yaml file:

pipeline=“<>”

setting from environmental variables:

-e DOCAI_PIPELINE=<>

setting from the command line:

–pipeline <> -p <>

Python

setting from the config.yaml file:

pipeline=“<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_PIPELINE=<>

run in front of `docai` command:

DOCAI_PIPELINE=<> docai …

setting from the command line:

–pipeline <> -p <>

```
### Number of replicas
This command provides the number of replicas. This value should match what you set when you published your pi
```

```
**Docker**
```

setting from the config.yaml file:

num_replicas: <>

setting from environmental variables:

-e DOCAI_NUM_REPLICAS=<>

setting from the command line:

–num-replicas <>

```
**Python**
```

setting from the config.yaml file:

num_replicas: <>

setting from environmental variables:

run before the `docai` command:

export DOCAI_NUM_REPLICAS=<>

run in front of `docai` command:

DOCAI_NUM_REPLICAS=<> docai …

setting from the command line:

–num-replicas <>

```
## Options for the run
The following options can be specified in the `config.yaml` file, in the environmental variables, or from the

### Providing a name for the run
This command provides a name for the scoring job. You can use the name to outline what is run in the job. For
5page-10rps-1pod"` to reflect the settings you established.


**Docker**
```

setting from the config.yaml file:

name: <>

setting from environmental variables:

-e DOCAI_NAME=<>

setting from the command line:

–name <> -n <>

```
**Python**
```

setting from the config.yaml file:

name: <>

setting from environmental variables:

run before the `docai` command:

export DOCAI_NAME=<>

run in front of `docai` command:

DOCAI_NAME=<> docai ...

setting from the command line:

–name <> -n <>

```
### Verbosity
```

```
This command prints the entire configuration (from the `config.yaml` file, the environmental variables, and t
```

```
**Docker**
```

setting from the config.yaml file:

verbose: true/false

setting from environmental variables:

-e DOCAI_VERBOSE=TRUE/FALSE

setting from the command line:

–verbose / –no-verbose

```
**Python**
```

setting from the config.yaml file:

verbose: true/false

setting from environmental variables:

run before the `docai` command:

export DOCAI_VERBOSE=TRUE/FALSE

run in front of `docai` command:

DOCAI_VERBOSE=TRUE/FALSE docai …

setting from the command line:

–verbose / –no-verbose

```
### Request a dry run of the information
This command performs a dry run of the provided commands. No requests will be made to the H2O Document AI - B
```

```
**Docker**
```

setting from the config.yaml file:

dry_run: true/false

setting from environmental variables:

-e DOCAI_DRY_RUN=TRUE/FALSE

setting from the command line:

–dry-run // –no-dry-run

Python

setting from the config.yaml file:

dry_run: true/false

setting from environmental variables:

run before the `docai` command:

export DOCAI_DRY_RUN=TRUE/FALSE

run in front of `docai` command:

DOCAI_DRY_RUN=TRUE/FALSE docai ...

setting from the command line:

–dry-run // –no-dry-run

Requesting a list of the given images
This command returns a list of all the inputted images without running the scorer. This command is useful whe

Docker

setting from the config.yaml file:

list_images: true/false

setting from environmental variables:

-e DOCAI_LIST_IMAGES=TRUE/FALSE

setting from the command line:

–list-images // –no-list-images

Python

setting from the config.yaml file:

list_images: true/false

setting from environmental variables:

run before the `docai` command:

export DOCAI_LIST_IMAGES=TRUE/FALSE

run in front of `docai` command:

DOCAI_LIST_IMAGES=TRUE/FALSE docai …

setting from the command line:

–list-images // –no-list-images

```
### Request logs for the scorer
This command requests logs for the run.


**Docker**
```

setting from the config.yaml file:

scorer_logs: true/false

setting from environmental variables:

-e DOCAI_SCORER_LOGS=TRUE/FALSE

setting from the command line:

–scorer-logs // –no-scorer-logs

```
**Python**
```

setting from the config.yaml file:

scorer_logs: true/false

setting from environmental variables:

run before the `docai` command:

export DOCAI_SCORER_LOGS=TRUE/FALSE

run in front of `docai` command:

DOCAI_SCORER_LOGS=TRUE/FALSE docai …

setting from the command line:

–scorer-logs // –no-scorer-logs

Log level
This command sets the log type for the run (one of: "INFO" or "DEBUG").

Docker

setting from the config.yaml file:

log_level: “<>”

setting from environmental variables:

-e DOCAI_LOG_LEVEL=<>

setting from the command line:

–log-level <>

Python

setting from the config.yaml file:

log_level: “<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_LOG_LEVEL=<>

run in front of `docai` command:

DOCAI_LOG_LEVEL=<> docai ...

setting from the command line:

–log-level <>

Dynamic page subsetting
This command is a corresponding HTTP server handler that passes an extra field to the underlying scorer. It l

Docker

setting from the config.yaml file:

extra: “<>”

setting from environmental variables:

-e DOCAI_EXTRA=<>

setting from the command line:

–extra <>

Python

setting from the config.yaml file:

extra: “<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_EXTRA=<>

run in front of `docai` command:

DOCAI_EXTRA=<> docai ...

setting from the command line:

–extra <>

Benchmark a run
The following benchmark commands can be specified in the `config.yaml` file, in the environmental variables,

Benchmark a job
This command will specify whether to set run this job as a benchmark.

Docker

setting from the config.yaml file:

benchmark: true/false

setting from environmental variables:

-e DOCAI_BENCHMARK=TRUE/FALSE

setting from the command line:

–benchmark \ –no-benchmark

Python

setting from the config.yaml file:

benchmark: true/false

setting from environmental variables:

run before the `docai` command:

export DOCAI_BENCHMARK=TRUE/FALSE

run in front of `docai` command:

DOCAI_BENCHMARK=TRUE/FALSE docai ...

setting from the command line:

–benchmark \ –no-benchmark

Provide a path for the benchmark results
This command provides a path for the benchmark results.

Docker

setting from the config.yaml file:

benchmark_results: “<>”

setting from environmental variables:

-e DOCAI_BENCHMARK_RESULTS=<>

setting from the command line:

–benchmark-results <>

Python

setting from the config.yaml file:

benchmark_results: “<>”

setting from environmental variables:

run before the `docai` command:

export DOCAI_BENCHMARK_RESULTS=<>

run in front of `docai` command:

DOCAI_BENCHMARK_RESULTS=<> docai …

setting from the command line:

–benchmark-results <>

```
## H2O Document AI - Bulk Scorer example
The following is an example of how to use the H2O Document AI - Bulk Scorer.

First, create a `config.yaml` file with the following information:
```

config.yaml

Configuration example file for H2O Document AI - Bulk Scorer

version: 0.2.5

Authentication can be done in two ways. If both are specified,

then the SSO support authentication will be used.

1. Authentication using docai_user, docai_password + Keycloak

docai_user: “your_username” docai_password: “your_password” auth_base_url: “https://auth.1234567.h2o.ai/auth” keycloak_client_id: “kc-id” keycloak_realm: “kc-realm”

Or

2. Authentication for SSO support using patform_token on Managed Cloud.

The following values can be obtained through the “Accessing H2O AI Cloud APIs” section

platform_token: “your1platform2token3here4…” token_endpoint_url: “https://auth.1234567.h2o.ai/token/endpoint/url” platform_client_id: “your-client-id”

Input & Output

images: null out_dir: “./results”

Pipeline

scorer_base_url: “https://document-ai-scorer.tester.h2o.ai” pipeline: “tester-pipeline” num_replicas: 4

Options for this run

name: nem-2x-5page-1rps-1pod verbose: true dry_run: false list_images: false log_level: “INFO”

Benchmark

benchmark: false num_requests: 1 benchmark_results: null

valid_image_file_extensions:

- “.pdf”
- “.jpeg”

temp_image_dir: null

```
**Docker**

Next, you create a new docker image and load the H2O Document AI - Bulk Scorer. Then, you provide your user i
u`).
The `-v` lines set up the location of the:

- files to be ingested into the H2O Document AI - Bulk Scorer (`local/test-nem-
5p` points to `/home/appuser/app/test-nem-5p`),
- the location of the output directory (`local/results` points to `home/appuser/app/results`), and
- the location of the configuration file (`config.yaml` points to `/home/appuser/app/.env`) respectively.

The last line is the final part of the configuration that is passed to the H2O Document AI - Bulk Scorer and

The following information is added to the command line that is not provided in the `config.yaml` file:

- The location of the input files (`-i`) is "test-nem-5p"; this file location is set in the first `-
v` command
- This example is set as a benchmark test for this run only (`--benchmark`)
- The benchmark results (`--benchmark-results`) print to "results/nem-2x-5page-1rps-1pod.csv"
```

$ docker load **Python**

Next, you create a new Python environment and load the H2O Document AI - Bulk Scorer.

The following information is added to the command line that is not provided in the `config.yaml` file:

- The location of the `config.yaml` file is exposed through the `--config` command
- The location of the input files (`-i`) is “test-nem-5p”
- This example is set as a benchmark test for this run only (`--benchmark`)
- The benchmark results (`--benchmark-results`) print to “results/nem-2x-5page-1rps-1pod.csv”

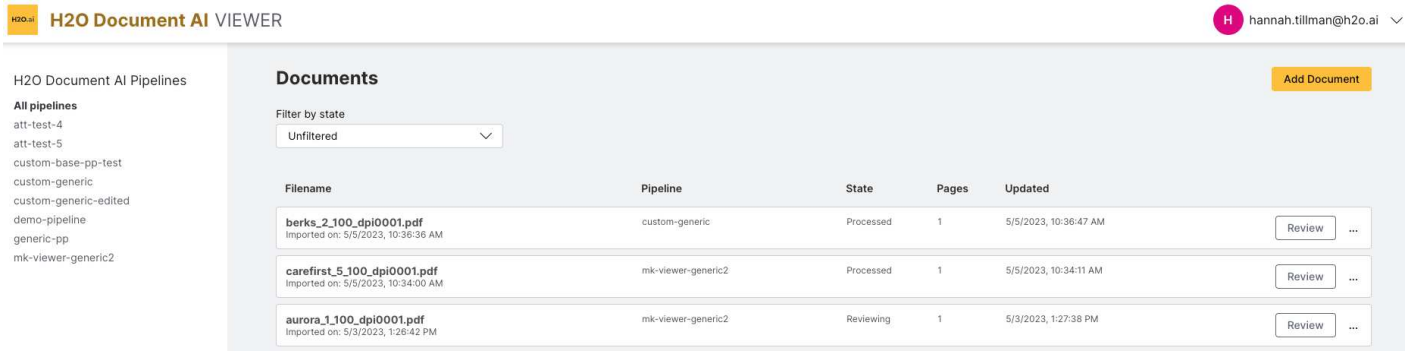
```
python -m venv .venv
source .venv/bin/activate
pip install docai_scorer-0.2.5-py3-none-any.whl
docai --config local/config.yaml -i local/test-nem-5p --benchmark --benchmark-results results/nem-
2x-5page-1rps-1pod.csv
```


Using H2O Document AI - Viewer

H2O Document AI - Viewer is an H2O AI Cloud (HAIC) application that lets you process documents. Your documents are processed using pipelines that are built and published in H2O Document AI - Publisher. Your H2O Document AI - Publisher and H2O Document AI - Viewer applications are intrinsically linked together.

Understanding the dashboard

The H2O Document AI - Viewer dashboard shows your available pipelines as well as your processed documents and their information. From this dashboard you can add more documents to be scored by an available pipeline.



Pipelines

The dashboard shows the scored documents of all pipelines by default. You can select to see the results of individual pipelines by clicking the pipeline name on the H2O Document AI Pipelines column.

Documents

The documents you have already processed are available in the order they were added to H2O Document AI - Viewer. You can see your added document after it has finished processing.

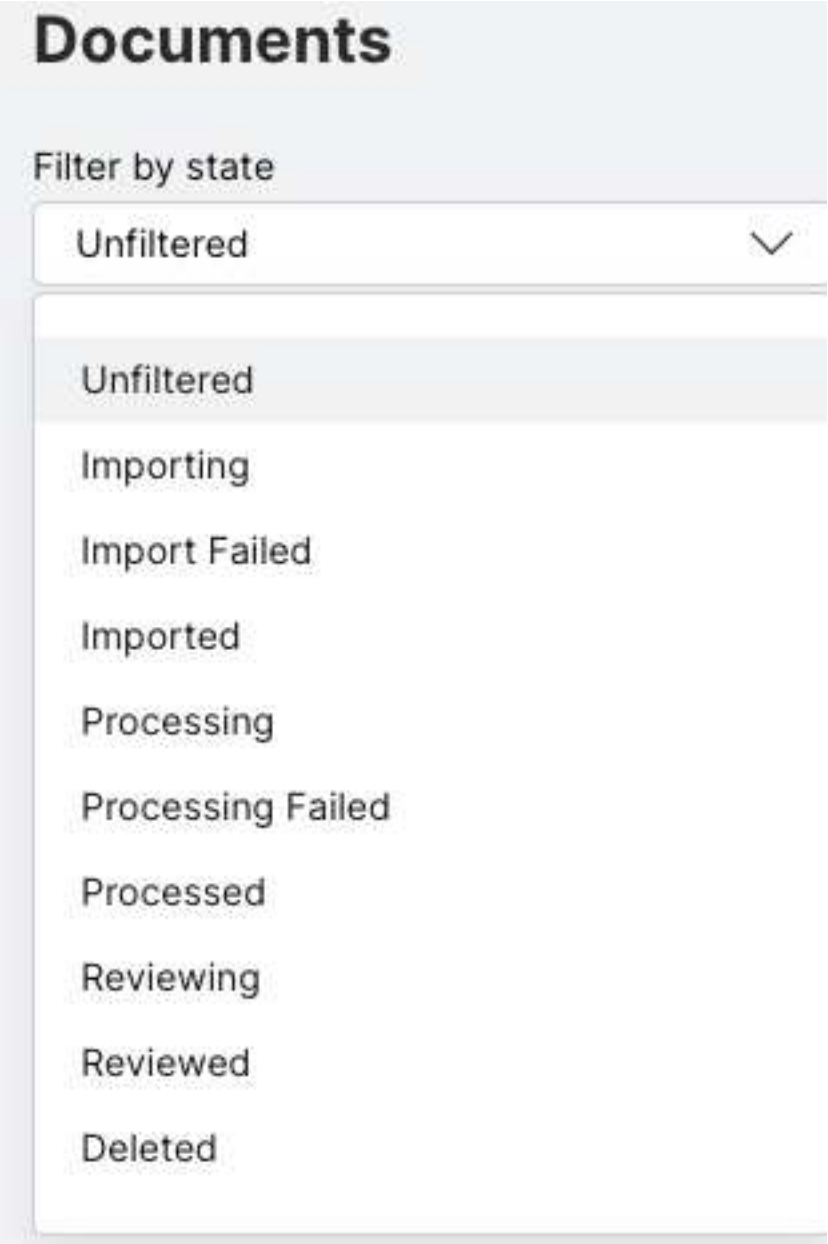
Each document row has the following information:

- **Filename**: The name of the document you added (multiple documents with the same filename can be added)
- **Pipeline**: The pipeline you used to process the document
- **State**: The current status of your document; one of:
  - *Importing* : This document is currently importing
  - *Import Failed* : This document failed to import and can only be deleted
  - *Imported* : This document has been imported but not processed
  - *Processing* : This document is currently processing
  - *Process Failed* : This document failed to process and can only be deleted
  - *Processed* : This document has successfully been processed and is ready to be reviewed

  - *Reviewing* : This document is in the process of being reviewed
  - *Reviewed* : This document has been reviewed
  - *Deleted* : This document has been deleted and cannot be interacted with any longer
- **Pages**: The number of pages the document has
- **Updated**: The last time your document was updated

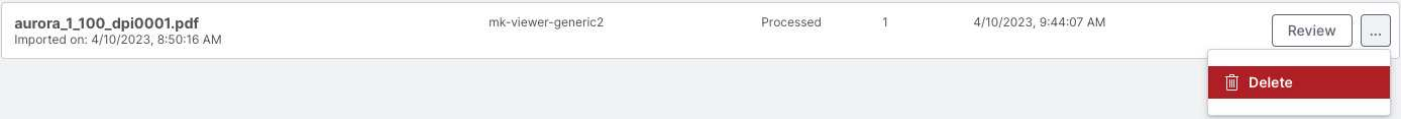
You can review your document any time after it has successfully processed. Reviewing your document allows you to access your Labels and values.

Filtering by state You can filter through your documents by their state.



Deleting a document You can delete a document from the dashboard.

1. Click the meatball menu at the end of your document row.
2. Click **Delete**.

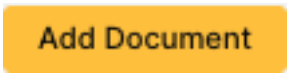


Workflow: Using H2O Document AI - Viewer

The following steps describe the workflow for H2O Document AI - Viewer.

Step 1: Add a document to be processed by an available pipeline

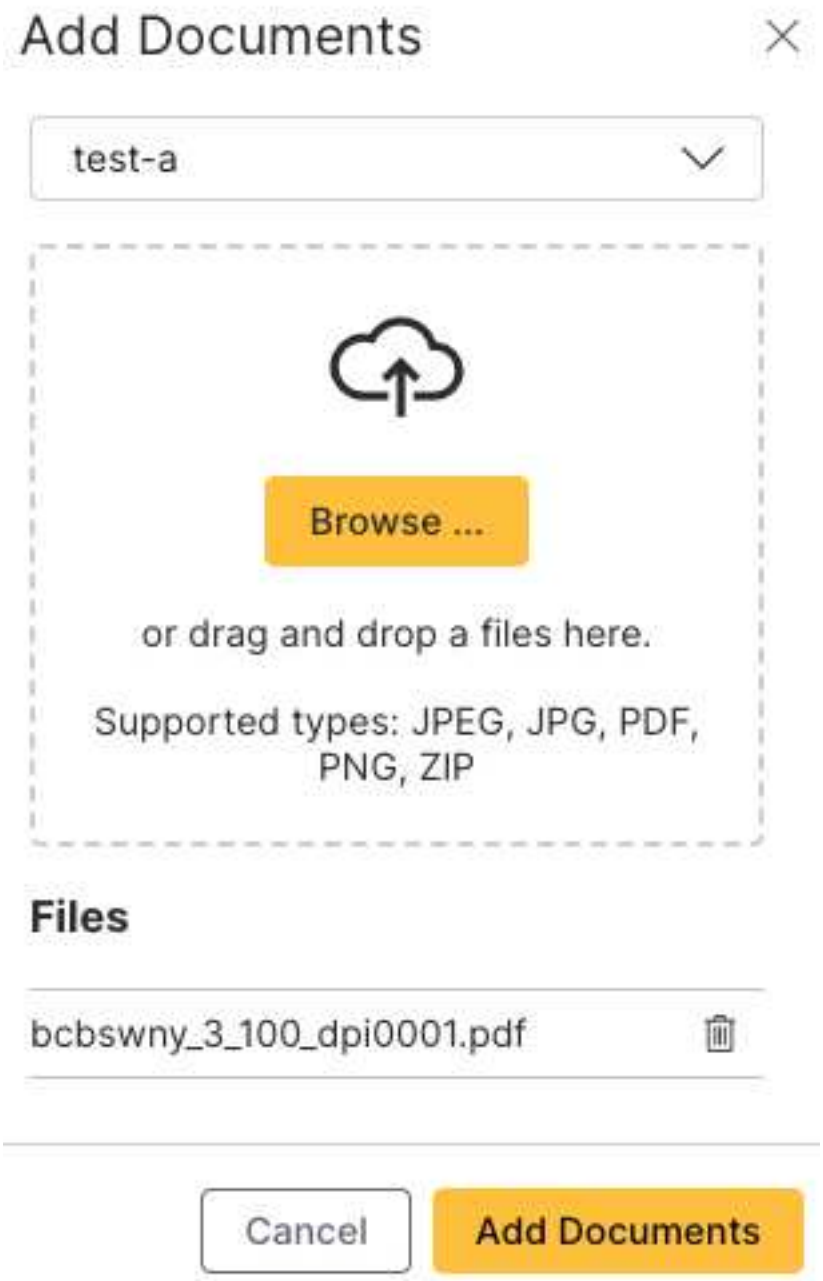
Start by adding a new document to be scored by H2O Document AI - Viewer.

1. Click **Add document** 

to add a new document.
2. Select which **pipeline** you want to use to score your document.
3. Select the **file** you want to process (you can only add one document at a time, but your document can have multiple pages).
4. Click **Add Document** to process the document.

Note: H2O Document AI - Viewer only accepts the following file types:

- A PDF file
- A JPEG file
- A JPG file
- A PNG file
- A ZIP file



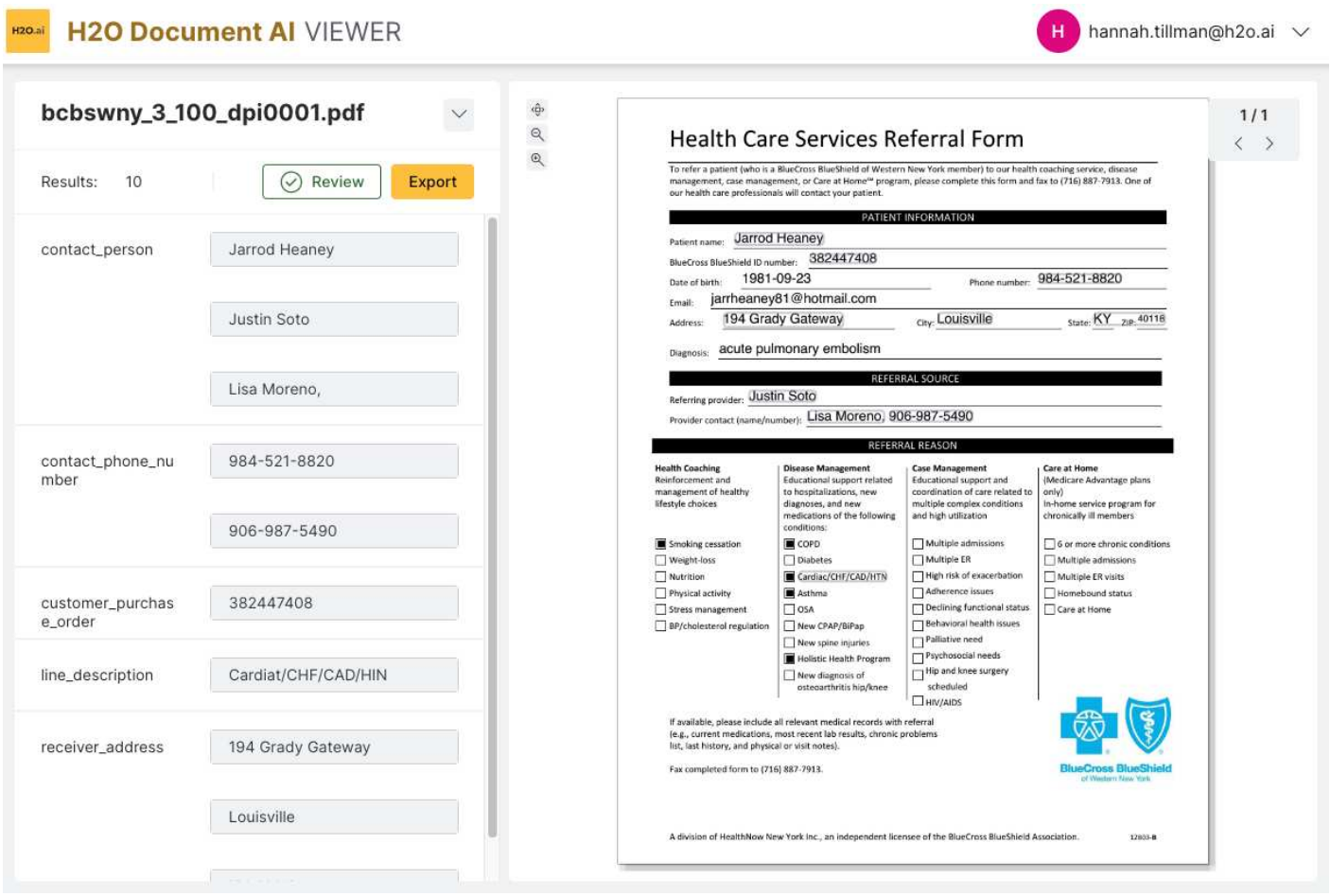
Your chosen pipeline then processes your file. A processing bar will show that the file is actively processing. You will be able to interact with your document after it has finished processing.

Tip: Processing a new file will take at least a few seconds (possibly longer) depending on the number of pages in the document and the number of predicted values.

When your document has finished processing you will be able to click **Review**  at the end of the document row to access the document results page.

Step 2: Access the processed document and review it for inconsistencies and inaccuracies

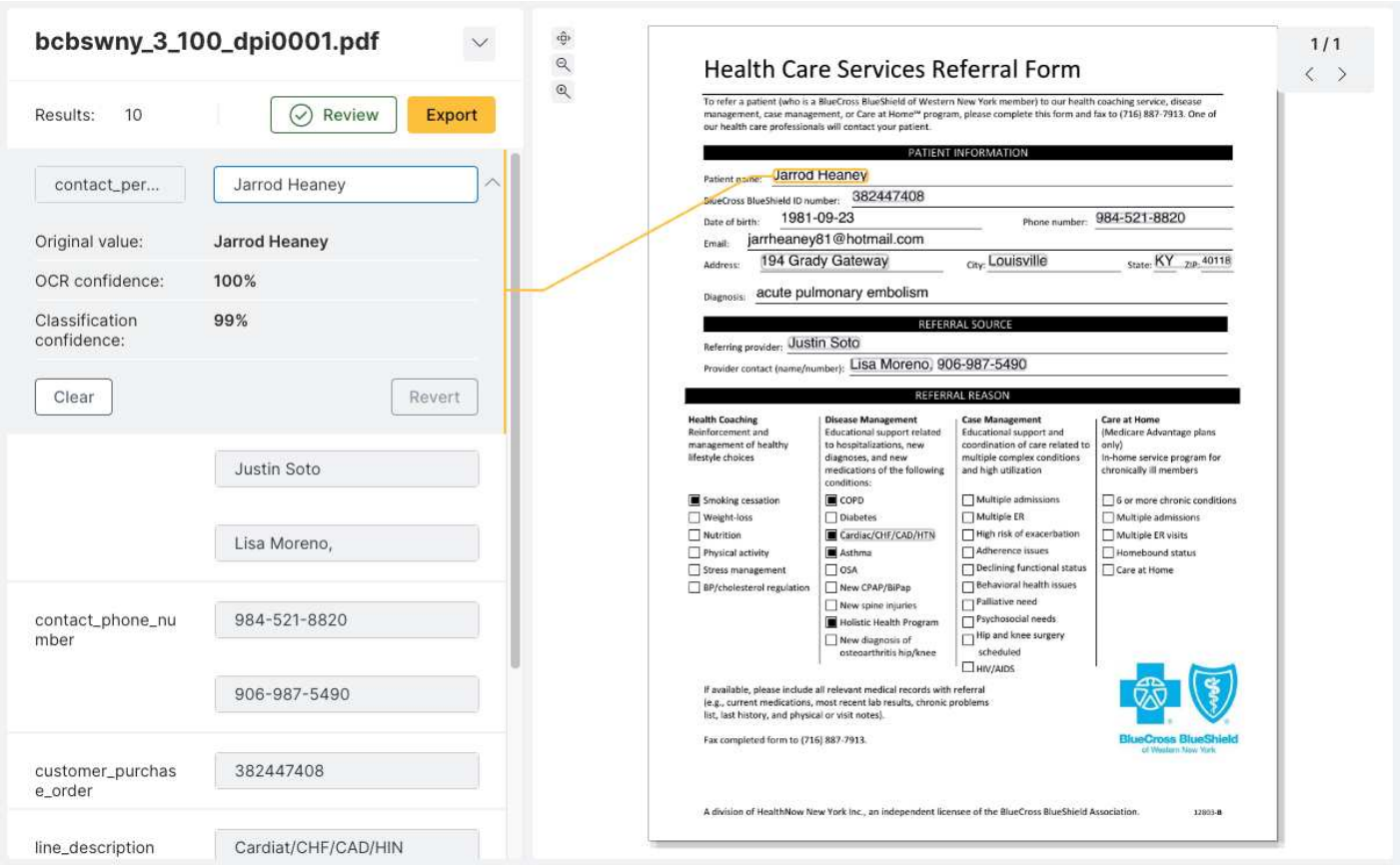
You review your document from the document results page. The document results page is split into the information panel on the left-side of the screen and the marked document on the right.



Information panel The top of the information panel displays: 1. The name of your document 2. The number of values (e.g. *Results: 10*) 3. The **Review** button 4. The **Export** button

If you click the drop-down arrow next to your document's name, you can access additional information: 1. The pipeline used to process your document 2. The state of your document (either *Reviewing* or *Reviewed*) 3. When your document was processed 4. The most recent export time

The information panel also shows the breakdown of your Labels and values. You can interact with the values predicted by your selected pipeline. When you select a value, it will point to that value's location on the marked document.



The information about that value will also be displayed when you click the value. The original value predicted by your pipeline is displayed along with the OCR confidence percentage and token classification percentage.

Interacting with values You can review your predicted values. Go through each predicted value and check that the pipeline correctly processed the information from the document.

For any incorrect predictions, change the content of the value: 1. Click the pencil icon  that appears when you hover over your value. This lets you edit your value. 2. Correct the predicted value by typing your correction in the value box. Your changes will automatically save.

After you have updated the value, your value box will be marked with a gold dot to show that you have changed the value.

If you want to change your value back to the original value, you can revert your changes by clicking **Revert**. You can also clear out the contents of the value box by clicking **Clear**.

After you have finished reviewing all of your document's values, Click the **Review** button 

to mark your document as Reviewed.

Current workaround **Tip:** Workaround Using a published pipeline that was built using a model that was trained using a single document in H2O Document AI - Viewer might result in that document having zero results. This is highly dependent on the quality of the model that was trained. If you try to score with the same file used for training, then it will probably, though not guaranteed, find at least one entity. Additionally, it is dependent on the quality and size of the bounding box created during edit in page view in H2O Document AI - Publisher.

Labels and values *Labels* are created in H2O Document AI Publisher and assign regions of a document with different meanings. For example, you can label a region on a document as `contact_name`. When that region has tokens in it, those tokens are assigned the label `contact_name`. Each label can have multiple values.

Values are the tokens within labeled regions. They are the text predicted and post-processed by the pipeline. For example, if the tokens `JOHN` and `SMITH` are in the labeled region for `contact_name`, then the predicted value for the labeled region `contact_name` is `JOHN SMITH`.

Step 3: Export the values of the document to your local computer

You can export your values to your local computer. 1. Click **Export** 

on the document results page. This will export a JSON file of your correct values. 2. Access your JSON file from your Downloads folder on your local computer.

Returning to the dashboard After you have finished reviewing your document and exporting your values, you can return to the H2O Document AI - Viewer's dashboard by clicking the product name in the upper left corner.

Key terms

This documentation has several unique terms used for H2O Document AI - Publisher and H2O Document AI - Viewer. The explanations for all such terms can be found here.

Annotation

Before data can be extracted from a document, it must first be annotated. Annotation refers to the process of labeling and organizing documents in a manner that makes them suitable for further analysis. This process can, for example, involve marking images or texts with bounding boxes that have labels attributed to each box.

Annotation set

An annotation set refers to the collection of different types of annotations. For example:

- Text annotations (usually from the OCR process)
- Token entity annotations (labels)
- Page annotations (classes)

Attribute

An attribute is a type of annotation. There are two types of attributes: region (which classify certain regions on the document/file) and file (which classify a whole document/file). Within an annotation set, multiple attributes can be created, with each storing different types of attributes about each document. For example, you may want to create an attribute for the main entity recognition annotations, and another for grouping line items together.

In H2O Document AI - Publisher, the attributes can be created in Page View below the file list. After attributes are created and the annotation set is saved, the set of attributes is shown in the annotation set list. This is a good way of quickly distinguishing between different annotation sets. In addition, the Apply Labels and Train Model actions require specific attributes, and the choice lists are filtered by the corresponding attribute types required.

AutoML

AutoML or Automated Machine Learning is the process of automating algorithm selection, feature generation, hyperparameter tuning, iterative modeling, and model assessment. AutoML tools such as H2O Driverless AI makes it easy to train and evaluate machine learning models. Automating the repetitive tasks around Machine Learning development allows individuals to focus on the data and the business problems they solve.

Batch size

The batch size defines the number of documents (with their bounding box coordinates) that are passed over to the processing device (with CPUs or GPUs) at a time. It is a hyperparameter that defines the number of samples to work through before updating the LayoutLM model parameters. If more than one GPU is available, then `per_device_batch_size` is determined by dividing the batch size by the number of GPUs.

Bounding box

In H2O Document AI - Publisher, a bounding box highlights and takes a spatial location in an image or document while a label is attributed to the bounding box.

Class

A type of information (such as “customer address” or “customer phone number”). In reference to an entire document (i.e. assigning a document a *class*), this is the type of document it is (such as “medical card” or “driver’s license”).

Concatenated annotation sets

The process of combining two or more annotation sets into a single larger set that includes all documents. Before annotation sets can be combined, they must have the same attributes.

As new documents become available, they can be annotated individually and combined with existing annotation sets to make a larger training set using the concatenate function.

Document sets

A set of documents for H2O Document AI - Publisher that can include PDFs, images (PNG, JPG, GIF), or a zip file containing a collection of the preceding filetypes, including a mix of multiple different types of documents.

Embedded text

Embedded text refers to metadata stored inside a PDF that conveys a precise definition of the text in the page, including the location of the text. When available, this can be used directly in order to more efficiently and accurately obtain the data needed for Document AI models.

Embedded text is usually available in documents created by software systems such as Microsoft Word, an order processing system, or a web browser. Embedded text is often unavailable in images from scanners, phones, or faxes, even if those images are stored as PDFs. When embedded text is not available, OCR using computer vision is used to obtain the text and location data needed for Document AI models.

Embedded text can be added to a PDF by any OCR process, so when Document AI encounters embedded text, it uses an algorithm to detect whether the embedded text is authentic and uses only authentic embedded text.

Entity

Entity refers to a set of related bounding boxes or is the instance of a class ("54th Avenue NYC, NY" for a customer address or "608-806-1234" for a customer phone number). If a model scores each of three contiguous tokens as address, it is common to group these together as a single multi-token entity. This step occurs in the post-processing stage.

Epochs

An epoch is defined as one pass over all the training documents. The number of epochs is a hyperparameter that defines the number of times that the learning algorithm works through the entire training dataset.

Ingest

The process of uploading documents to H2O Document AI - Publisher using the web interface or API.

Intelligent Character Recognition

Intelligent Character Recognition (ICR) is an advanced Optical Character Recognition (OCR) that recognizes characters beyond font libraries in a generalized manner.

Jobs

An action taken by the H2O Document AI - Publisher system. Some examples include importing documents, annotation set operations such as saving, executing models such as OCR or token classification.

Label

In H2O Document AI, the term label is used specifically for annotating token entities. When a document set is initially uploaded to H2O Document AI - Publisher, an annotation set is created with the region attribute "label". When a document is added to and processed by H2O Document AI - Viewer, the labels are available on the document results page.

Labeling

In H2O Document AI - Publisher, this is the task of detecting and tagging data with labels in images, videos, audio, and text. Labeling data is an important step in data preparation and preprocessing for building AI.

LayoutLM

LayoutLM is a multi-modal AI modeling architecture that is designed specifically for document understanding tasks, incorporating features of the text and also the locations of the text.

Models

An artifact that has been trained to perform H2O Document AI - Publisher tasks.

Natural language processing (NLP)

NLP is a subfield of linguistics, computer science, and artificial intelligence that is concerned with the interactions between computers and human language. In particular, NLP knows how to program computers to process and analyze large amounts of natural language data.

Optical character recognition (OCR)

Optical character recognition (OCR) recognizes characters in documents or images and provides the text and text location.

Page classification

An H2O Document AI - Publisher model type that learns what type of document a page is by using the text within the page. For example, you can train a page classification model to differentiate between invoices, receipts, and pay stubs.

Post-processing

Modeling stages that occur after the primary AI model(s). In H2O Document AI - Publisher, a common post-processing step is to aggregate contiguous tokens together to create a single entity. Another example is to standardize date text into a standard date format.

Pre-processing

Modeling stages that occur before the primary AI model(s). In H2O Document AI - Publisher, image processing tasks are handled as pre-processing.

Predict

Predict refers to the process of using a model to create annotations against an annotation set. This is typically done using the Predict Using Model option from the annotation sets page. However, this can also be done while training the model by using the evaluation section of the train interface. Each creates a new annotation set with the attributes being predicted from the model. Predicting is often referred to as scoring and running inference.

Project

A set of data and models related to a particular data type. You must create a project before you can upload any data. Projects store all document sets, annotation set, models, and published pipelines.

Publish

The term Publish refers to the process of creating a pipeline of multiple actions that collectively process a document into a result set. In most cases, this describes the end goal of H2O Document AI - Publisher, where the OCR stage, one or more trained models, and post processors are combined into a single process that is optimized for Document AI MLOps. Whereas use of the H2O Document AI - Publisher user interface works in single batch jobs to create elements of a pipeline, processing documents from end-to-end with a Rest API occurs in H2O Document AI MLOps, and Publish refers to the action of creating the pipeline.

Quality

You are given a quality score on a prediction annotation set when you train a model using evaluation. The quality is the f1-score of the model that was applied to the dataset.

Result sets

Result sets show the final stage of the data after applying one or more post-processing actions to an annotation set. Converting individual token predictions into multi-token entities is an example that would transform an annotation set into a result set.

Split annotation set (SAS)

The process of dividing a single annotation set into smaller pieces. This is commonly done to set up an AI task into training and validation sets. When training AI models, it is common to use a portion of the labeled data to train the model where the model sees the document and answers. The other portion of the labeled data is then used to judge the accuracy where the model predicts answers it was not shown. The errors are calculated and analyzed. This helps to ensure that the model works against documents it has never seen.

Tagging

The process of making unstructured data more structured by manually or automatically adding tags or annotations to various components of the unstructured data.

Token labeling

The process of adding annotations to tokens or sets of tokens. In H2O Document AI - Publisher, this usually refers to adding entity annotations, or region attributes, of the class “label”.

Train models

The process of training an H2O Document AI - Publisher model with an annotation set. This involves using an annotation set with “text” and “label” to train a token labeling model, or an annotation set with “text” and “class” to train a page classification model. Training models is the H2O Document AI - Publisher task that requires the most time.

Value

In H2O Document AI - Viewer, a value is the predicted token within a labeled region. Values are located on the document results page.

Release notes

v0.12.2 (August 29, 2025)

Improvements

- Upgraded Go to version `1.24.6`.
- Upgraded Argo Workflows to `v3.7.1` to resolve known security vulnerabilities.
- Enabled offline mode for transformers in the scorer runtime, allowing execution in air-gapped environments.

v0.12.1 (August 4, 2025)

Improvements

- Upgraded Argo Workflows to `v3.7.0` to resolve known security vulnerabilities.
- Made pipeline sidecar's `securityContext` configurable via Helm values.

Bug fixes

- Fixed a bug where training failed for some base models (`deberta-v3-base`, `multilingual-deberta-v3-base`) due to protobuf implementation conflicts.
- Fixed an issue where importing models failed during post-processing because test PDFs were not packaged correctly.
- Updated internal Kafka client to the latest version.
- Fixed a crash when publishing a pipeline with a conditional multi-labeling model and no condition values selected.

v0.12.0 (July 21, 2025)

New features

- Added support for custom GPU scheduler integration with Run.ai. When enabled, Document AI can use a custom scheduler for dynamic GPU workloads.
- Reduced disk space requirements and simplified deployment by merging the OCR and scorer runtime images into a single, smaller image.

Improvements

- Patched security vulnerabilities.
- Upgraded Argo Workflows to `v3.6.10`.

Bug fixes

- Fixed an issue where the `Apply labels` operation failed if a document set contained a UUID in its name.

v0.11.3 (May 13, 2025)

Bug fix

- Fixed an issue where private CA certificates were not properly merged with the system CA bundle. This caused SSL validation errors when connecting to external services, such as Azure Blob Storage or Azure PostgreSQL. The fix ensures that required newline characters are preserved during certificate merging.

v0.11.2 (May 8, 2025)

Improvements

- Patched security vulnerabilities.

Bug fixes

- Fixed an issue where the Document AI scorer deployment did not merge the system CA bundle with user-provided certificates during installation.
- Fixed an RBAC issue that allowed users without the appropriate cluster role to publish pipelines.
- Added warnings for users who lack the required cluster roles when attempting to unpublish pipelines.

- Fixed an RBAC issue where users with score-only permissions received no warning when attempting to roll back a pipeline.

v0.11.1 (April 14, 2025)

Improvements

- Patched security vulnerabilities.
- Upgraded Go to version `1.24.2`.

v0.11.0 (April 4, 2025)

New features

- Added Azure Blob Storage support as an alternative to MinIO in Azure environments.
  - Configurable through specific integration points:
    - `azureBlob` configuration in **document-ai** Helm chart values
    - `datastore` and `azureBlob` sections in **ml-api** Helm chart values
    - `artifactRepository` configuration for Argo Workflows **Caution:**
      Configuration requirements:
- Only one storage option's `enabled` field can be set to `true` (either S3 or Azure). Setting both to `true` or both to `false` will cause the application to return a fatal error.
- S3-compatible storage remains the default if not explicitly configured.

Improvements

- Upgraded Argo Workflows to `v3.6.5` to resolve several vulnerabilities.
- Upgraded Go to version `1.24.1`.
- Upgraded Go dependencies.

Bug fixes

- Fixed an issue where the list of pipelines showed user ID instead of preferred username.
- Fixed an issue where runtime version was displayed as empty in the pipeline list.
- Fixed an issue where custom post-processor code was not prefilled in the pipeline publishing modal.

v0.10.0 (Febrary 28, 2025)

New features

- Document AI can now be configured to schedule dynamic workloads into Kubernetes namespaces that are predefined in the corresponding authorization workspace.
- The logo can be customized in the UI.

Improvements

- Project deletion logic has been improved to delete any remnant data if the project is deleted while any pending job is running.
- Upgraded Argo Workflows to `v3.6.4` for improved stability and performance.
- The auto-deletion warning now explicitly states that published pipelines will also be deleted when a project is removed.

Bug fixes

- Fixed an issue that prevented project deletion when pending jobs existed.
- Fixed an issue where `.zip` files couldn't be uploaded into Viewer.
- Fixed an issue where button text was cut off in the `STATUS` field.
- Fixed a misprint in the warning message when adding a collaborator.
- Fixed an issue where results were incorrectly displayed when processing documents without tables.
- Fixed inconsistent use of PostgreSQL library – Fully integrated `github.com/h2oai/go-pkg/database/postgres` for all database connections, ensuring support for passwordless connection strings and IAM authentication for AWS RDS.

v0.9.2 and v0.9.1 (December 3, 2024)

Improvements

- Annotated all pods (system and user workloads) for telemetry and cost allocation, ensuring better tracking and reporting capabilities.
- Updated the discovery service response to include `services/document-ai-backend`, improving the visibility and integration of these services within H2O AI Cloud.
- Improved the UX for adding collaborators to a project by replacing the single textbox with a tabular list, allowing easier management of email addresses and edits.
- Improved the UX of the **Publish Pipeline** panel by reflecting changes in **YAML** after enabling the custom pipeline, allowing dynamic model selection, and supporting multiple models in the pipeline with a dynamic table instead of static dropdowns.
- Upgraded Argo Workflows to `v3.6.0`.
- Improved the description field by adding a slider to prevent long descriptions from pushing other menu tiles to the bottom.
- Kafka now distributes messages using a Round Robin strategy, improving load balancing and scalability between pipeline replicas.

Fixes

- Fixed issue where pipelines remain visible in the Viewer after the associated project is deleted in the Publisher.
- Fixed issue where unsupported file extensions could be dragged and dropped without a clear error message, now displaying a warning when such files are added.
- Fixed issue where uploading a JPEG and then downloading it resulted in the file extension being changed to PDF, now retaining the original file extension and mime type.
- Fixed issue where `ProjectID` was not persisted with the pipeline, now ensuring `ProjectID` is attached to the **K8s Job Extractor** `cloud.h2o.ai/workspace-id` label via **multipart upload service**.
- Fixed an issue where the **Helm package** command used the `--app-version` switch instead of `--version`, causing a mismatch between the Helm chart version and the release version.
- Fixed issue where, when Auto-deletion was enabled for a project, the artifacts were deleted but the empty project remained in the list, even though Auto-Purge was not enabled and the retention period was set to two days.
- Fixed issue where the collaborator's email address input was not processed when the user blurred the field via mouse, causing users to believe the input was filled but not submitted or validated.
- Fixed issue where pipelines couldn't be listed due to the backend reading `NULL` into a string, causing a `Scan error`.
- Fixed issue where uploading a nested zipped file failed due to postprocessing being stuck in a pending state, preventing successful file upload and processing.
- Fixed issue where the namespace was not fully visible in the Revisions pop-up by making the columns responsive to size.
- Fixed an issue where the UI failed to fetch the `openid-configuration` from Keycloak, causing the browser to freeze due to immediate retry attempts in a continuous loop.
- Fixed an issue where users were incorrectly notified about unsaved changes after saving their changes to annotation sets while attempting to leave the page.
- Fixed an issue where failed models appeared in the list to be selected for publishing in the pipeline.
- Fixed an issue where a failed model was present in the list of models to be selected for publishing in the **Token labeling** section.
- Fixed an issue where it was possible to publish a pipeline with spaces instead of valid page class values, which should require logical values in the page class fields.
- Fixed issue where the **Publish Pipeline** button was not enabled even when no model was uploaded.
- Fixed bug where users had to click the **Save** button twice when adding several collaborators at once.
- Fixed an issue that would not let users to run Document AI on OpenShift clusters.
- Fixed an issue where the Helm chart did not allow configuring the `securityContext` of the scorer's sidecar container, now enabling this configuration in `v0.9.2`.

v0.9.0 (November 1, 2024)

New Features

- Added ability to downgrade your pipeline version to the previous version. This lets you recover pipelines from failure after auto-upgrade.
- Added visible logs you can access if your pipeline fails to start so you can debug it easier.

- Added ability to put pipelines to sleep to keep them from consuming resources when not actively being used.
- Added a validation rule to the pipeline that checks that the requested CPU or memory is not higher than the upper limit.

Improvements

- Added ability to see pipelines in all states (including deleted pipelines).
- Added ability to upgrade pipelines concurrently by bringing in a configurable pool of workers.
- Added a readiness probe to the scorer/pipeline to check connectivity with Kafka since Kafka is required for the scorer/pipeline to work correctly.
- Improved setting transparency for pipelines.
- Updated the consistency of the naming in Viewer from "status" to "state".

Fixes

- Fixed pipeline publication process starting but not completing in UI after several hours despite finishing in backend.
- Fixed new runtime version not reflecting in UI after the upgrade.
- Fixed long file names overflowing the text boxes in the UI by causing the text to wrap when it is too long.
- Fixed labels not saving from one annotation set when concatenating it to another annotation set with the same type of label IDs.
- Fixed concatenation of two separate document sets with OCR resulting in the wrong number of documents.
- Fixed inability to concatenate an OCR-ed annotation set with a non-OCR-ed annotation set / annotation set without labels / annotation set with the same type of labels / with the same label.
- Fixed auto-deletion's "less than 2 weeks ahead" time counter not reading properly, causing the banner to always be red.
- Fixed the cleared token value reverting back after re-login.
- Fixed "Edit Project" reverting your changes on the Edit Project panel if you stay there long enough without doing anything.
- Fixed pipelines that were put to sleep due to them failing not being rewoken to fail further by feeding them new documents.
- Fixed error in failing to fetch and score documents of large sizes or zipped documents of large sizes in Publisher.
- Fixed previous document name being shown in Viewer when reviewing further documents though the labels were from the new document.
- Fixed the inability to edit a published pipeline so that it contained the OCR-method only.
- Fixed annotation set importation prompting you to reconcile duplicates when there are, in fact, no duplicates.
- Fixed pipelines published without models (i.e. OCR-only pipelines) not being visible in the Project.
- Fixed publishing a pipeline with no models that has the same name as a previously un-published pipeline that has models displaying those models from the un-published pipeline with the same name until the page is refreshed.
- Fixed pipeline auto-upgrader getting stuck in the `pending-upgrade` state by performing a rollback to the last state if necessary.
- Fixed OCR methods failing except for Tesseract.
- Fixed inability to split an OCR-ed annotation set.
- Fixed copied attributed in imported documents not being present in the annotation detail table.
- Fixed misspelling on the add documents panel in the UI.
- Fixed misprint in the error message about adding the names of attributes from deleting the names of the attributes.
- Fixed inability to import a file larger than 5M.
- Ensured that the rollback to previous pipeline version button overlays properly.
- Addressed multiple security concerns.

v0.8.1 (July, 2024)

Improvements

- Implemented new pipeline backward compatibilities: Upgrading Document AI to a new verion (0.7 > 0.8) will cause all existing pipelines to automatically upgrade to the new version.
- Updated Argus to v0.22.3.
- Updated bulk scorer to v0.2.5.
- Added a separate namespace for user workloads in Kubernetes.
- Added SSL/SASL support for communication with Kafka.
- Added `gocloud.dev` as the drop-in replacement for `*sql.DB` to handle DB connections from the API server.

- Introduced pod disruption budgets for every deployment.
- Added compatibility with pipelines published in previous versions of Document AI (v0.7.x).
- Introduced the `extra` parameter to bulk scorer.

Fixes

- Fixed pipelines with long names only being partially deployed.
- Extended support exclusively to `networking.k8s.io/v1` and dropped support to `networking.k8s.io/v1beta1` for ingresses.
- Removed ZIP files as a listed option when submitting documents to a pipeline to score in Publisher UI since ZIP is not supported for scoring.
- Fixed pipelines with PaddleOCR-Latin failing to score documents due to file read permission issue.
- Fixed pipelines failing to score large files with a high number of pages.

v0.8.0 (July, 2024)

New features

- Added the ability to configure `nodeSelector` and `tolerations` for doc-proxy-scorer.
- Implemented table support.
- Implemented project collaboration between users using email invite.
- Introduced the hybrid OCR processor to read PDF characters directly from documents and to extract text contained in images.

Improvements

- Introduced new scoring infrastructure.
- Migrated pipeline deployment to Viewer backend. Pipeline publishing still uses Helm charts, but now creates a new record in a postgres table which is necessary for the endpoint listing the published pipelines (required for Viewer backend scalability).
- Implemented a get Job endpoint that allows polling the request/job state.
- Added capability for Viewer to parse various JSON outputs.

Fixes

- Fixed copied attributes from an annotation set not being reflected in the annotation detail table.
- Fixed the inability to import a document set with copied attributes set.
- Fixed the inability to delete an annotation set if it was exported before.
- Fixed renaming an annotation set producing an empty string instead of the changed name.
- Fixed not being able to publish a pipeline with the same name as an unpublished pipeline.
- Fixed Helm release of a published pipeline having the wrong appVersion (appVersion is indicated as `---version`).
- Removed misleading log messages.
- Resolved critical vulnerabilities in the image `h2oai-mlapi-worker-ocr`.
- Fixed auto deletion issue where a project was being deleted on `[date -1]`.
- Fixed the name of a document set disappearing after being edited.
- Fixed the inability to delete documents when the corresponding annotation set was edited.
- Fixed the pipeline controller ticker (which syncs pipelines) stopping working when it errored in the middle. It will now restart.
- Fixed pipelines published in v0.7.x not showing correct page counts.
- Fixed documents not being scored when pipeline is re-scaled.
- Fixed the inability to unpublish pipelines.
- Fixed pipeline autoscaling not working.
- Fixed error when obtaining logs for scoring documents in Publisher.
- Fixed failure to unzip ingested ZIP files.

v0.7.2 (Mar 14, 2024)

Fixes

- Fixed a memory leak to stop out-of-memory pod failure after a certain amount of scored documents.
- Fixed issue with page population.

v0.7.1 (Feb 12, 2023)

New features

- Implemented RBAC for publishing pipelines.

Improvements

- Added compatibility with pipelines published in previous versions of Document AI (v0.6 and v0.5).
- Made Viewer work with pipelines that have custom V4 post-processor.

Fixes

- Fixed archive extracting.
- Fixed deletion of DocumentSet when corresponding AnnotationSet was edited.

v0.7.0 (Nov 5, 2023)

New features

- Introduced the universal scoring pipeline.
- Introduced ability to automatically purge training artifacts.
- Introduced ability to schedule the deletion of your whole project and all of its resources.
- Introduced new base models for training a model in Publisher.
- Introduced learning rate for model training in Publisher.
- Implemented role-based access control (RBAC) to Document AI on HAIC.

Improvements

- Expanded file support for Viewer document import to include JPG and ZIP files.
- Created a custom pipeline repository which includes fine-tuned BERT models and OCR-only pipelines.
- Added the ability to skip pages from scoring.

Fixes

- Removed support for Kubernetes v.< 1.23.
- Deleting now works on Project Level even if there are Job failures; this also deletes underlying Pipelines.

v0.6.2 (Aug 23, 2023)

New features

- Added the ability to update or add new templates.

Fixes

- Fixed an issue where the logic used to pull the feature store web proxy image did not work if the image registry is empty.
- Made an improvement to include `input_dir` content in the request going to the custom post-processor deployment.

Known issues

- After publishing a pipeline using a model trained on one file on H2O Document AI Viewer, the document will show zero results.

v0.6.1 (Jul 28, 2023)

Improvements

- Added support for Kubernetes 1.26.
- Updated telemetry implementation to make scored documents more efficiently retrieved.

Known issues

- Adding a file in edit in page view only supports image files.
- Adding a file in edit in page view does not update page and document number on the annotation sets page.
- After adding a file in edit in page view, leaving edit in page view, then returning to edit in page view, you cannot view said added file.

v0.6 (May 21, 2023)

New features

- Introduced H2O Document AI - Viewer for business users to score documents on built pipelines. [**MVP**]
- Introduced initial telemetry integration.
- Added the ability to score PDFs with page ranges.

Improvements

- Renamed original H2O Document AI to H2O Document AI - Publisher.
- Added a button to Published Pipelines to retrieve the scoring URL.
- Sped up CPU for EfficientNet OCR models by running with optimized OpenVINO.
- Sped up PDF scoring with JPG instead of PNG.

Fixes

- Fixed an issue where long names could not be used when publishing pipelines.
- Fixed an issue where zip files could not be uploaded from Windows machines.
- Fixed an issue where job status was still reported for cancelled jobs.

v0.5 (Apr 13, 2023)

New features

- Added optical character recognition (OCR) language support for:
  - Latin (e.g. Spanish)
  - Arabic (e.g. Persian)
- Added Document Text Recognition (DocTR) EfficientNet models to better recognize handwritten documents.
- Added ability to set batch size and number of epochs for model training.
- Added command-line bulk scorer to score a large number of documents (ships separate from main product).

Improvements

- Upgraded the ML API to v0.4.0.
- Refactored and improved the training user interface for better usability.
- Added the ability to gate access to H2O Document AI based on a user's role.

FAQs

H2O Document AI is an H2O AI Cloud (HAIC) engine that lets you build accurate AI models that:

- Classify documents
- Extract text, tables, and images from documents
- Group, label, and refine extracted information from documents

H2O Document AI supports various documents and use cases to help organizations understand, process, and manage large amounts of unstructured data. Upload your documents to H2O Document AI using the H2O Document AI web interface (in HAIC) or API. H2O Document AI lets you handle a wide variety of documents, including:

- Image scans (faxes in PDF or other formats, pictures with text, and non-editable forms)
- Documents with embedded text that have text and layout metadata (PDF docs, Word docs, HTML pages)
- Documents with regular text "left to right/top to bottom" (CSVs, emails, editable forms)

H2O Document AI uses a combination of:

- Intelligent Character Recognition (**ICR**), which leverages learning algorithms for generalizable character and word recognition,
- Document layout understanding, and
- Natural Language Processing (**NLP**) to develop highly accurate models rapidly. *** The following sections provide answers to frequently asked questions. If you have additional questions, please send them to cloud-feedback@h2o.ai.

Models

These questions involve model training and model functions.

What is the format of an exported model?

Models are exported as zip files with the artifacts necessary to execute the LayoutLM model only.

What is the requirement to run an exported model?

You need H2O Document AI's specific pipeline to run the exported model. It will not run in H2O MLOps or in any other customer environment (unless they handle all the parts in the same way).

You could execute the model and use Microsoft's LayoutLM code, however, this is reasonably complex.

Is exporting a model similar to creating a scoring pipeline?

Model export is not the same thing as publishing a scoring pipeline.

If you export a model, that can only be done in H2O Document AI's UI for the LayoutLM models. You can run that model open source, but you still need to know how with tokenization, location embeddings, and other elements in place. It is essentially a transformer's architecture.

The pipeline that is deployed when you publish a pipeline contains:

- the way you ingest documents
- your chosen method of OCR (which can include checking embedding quality, using embedded text, rotation, detecting and recognizing)
- the ability to execute a page classification, token labeling model, or both
- the ability to execute post-processing against the raw predictions of the above models

Pipelines

The questions involve pipeline publishing and scoring documents.

How do the replica values work when running the bulk scorer?

The number of replicas should not exceed the maximum number of replicas you set when you published the pipeline. If the number of replicas you are using exceeds the maximum number of replicas available, it will take time to allocate more replicas because they will need to be freed up first.

