

Table of Contents

Table of Contents	0
Setting the stage	1
Situation	1
Getting connected	3
Environment information	3
Creating your project and pipeline server	5
Create a project	5
Creating a workbench	8
Git-Clone the common repo	13
Normalizing Content	14
Notebook 1-normalize content	15
Metadata extraction & chunking	16
Notebook 2-metadata extraction & chunking	17
Why is data preprocessing for LLMs so challenging?	22

Setting the stage

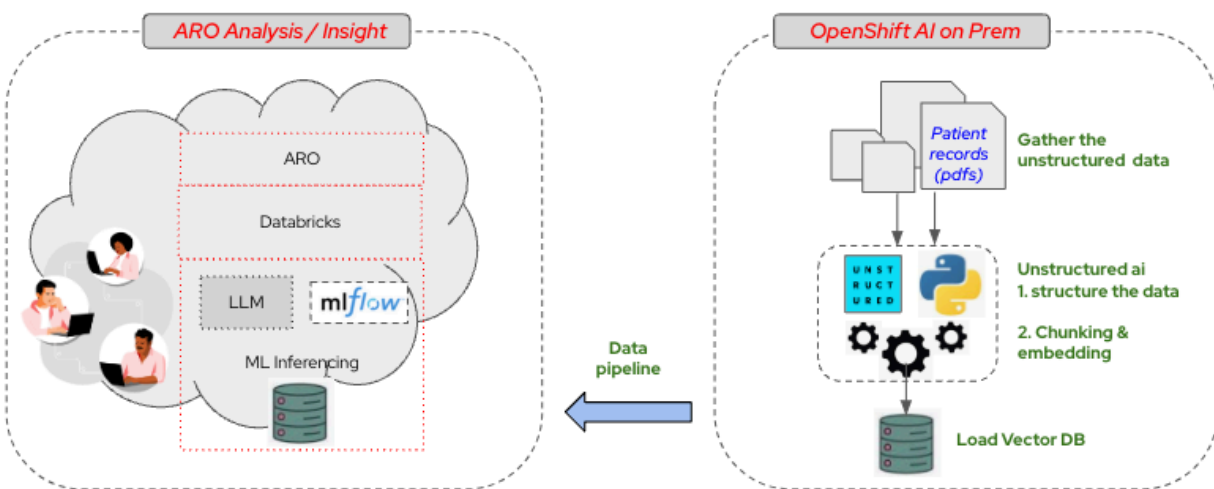
In order to make this lab seem more realistic, we will be describing this imaginary scenario.

Situation

- We all work for a healthcare company that is interested in structuring their 'unstructured' patient chart data for future LLM usage.
 - Creating a Chatbot that can answer questions about medical procedures has been discussed.
- A small team, consisting of Data Scientists and MLOPs engineers, were asked to create a POC to demonstrate the above idea.
- The data, for the POC, consists of 'anonymous' medical patient data which can be in any of the following formats. The team has been asked to devise a method to structure the following 'unstructured' data so that it could be used within an LLM or Chatbot:
 - PDF
 - HTML
 - Power point
 - Handwritten Note
 - Word document
- The team currently has all of their data 'on-prem', but would like to build/train/deploy their future LLM model in the cloud.
- The team would like to preprocess the data onprem. This enables them to only send the data to the cloud that is needed. This also enables them to avoid storage costs for any unneeded data.
- The team looked at a number of vendor solutions and decided to use the Unstructured SDK from Unstructured.io. The SDK is a no-code tool that ensures unstructured data can be 'structured' and continuously flow to an LLM.
- **Steps in the 'unstructured' process include:**
 - Normalizing and Structuring PDF data

- Perform Metadata Extraction and Chunking
- Using Weaviate to store their vectorized data
- To quickly deliver a POC to senior management, the team has decided to restrict data types, only pre-processing medical chart pdfs on prem, chunking and embedding, then loading the documents into a vector database (weaviate).
- Eventually the team will extract the chunks for inclusion in a prompt template when they interact with their LLM in the cloud.

Gather & Structure pdf data & insert into Vector DB



The following lab exercise will help you understand the 'unstructured' process, used by this team, in their first PoC. The next 60 minutes is your official training session, for you to ramp up and gain an understanding of the various technologies involved during the POC phase.

Getting connected

For the purposes of this training session, we have provisioned a single OpenShift cluster, with OpenShift AI deployed on it.

Each person attending this lab will have a unique user account in which to do their work.

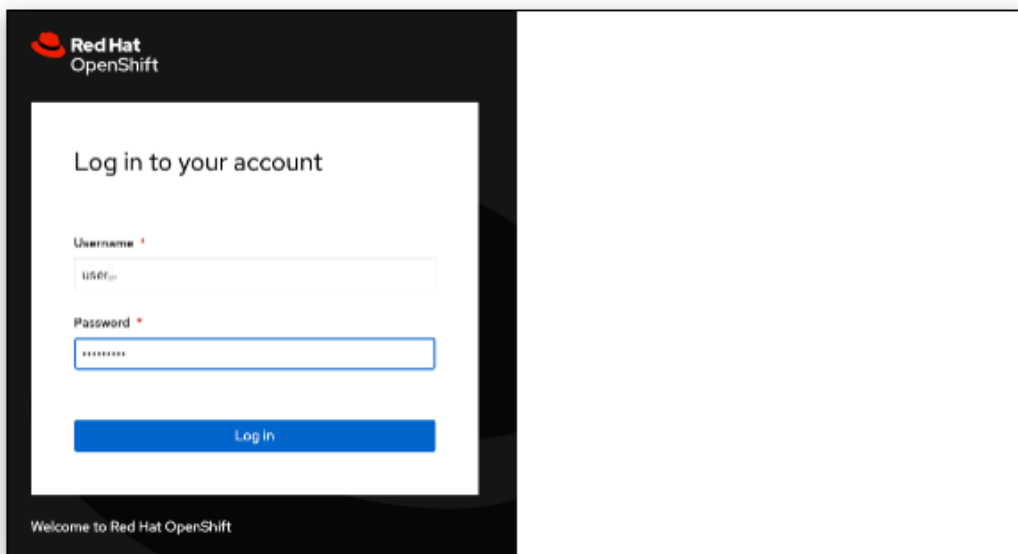
Environment information

If you are using the customized version of the instructions, the information below will render properly. If not, you will see placeholder values instead.

- Your account id: userX
- Your password: openshiftX

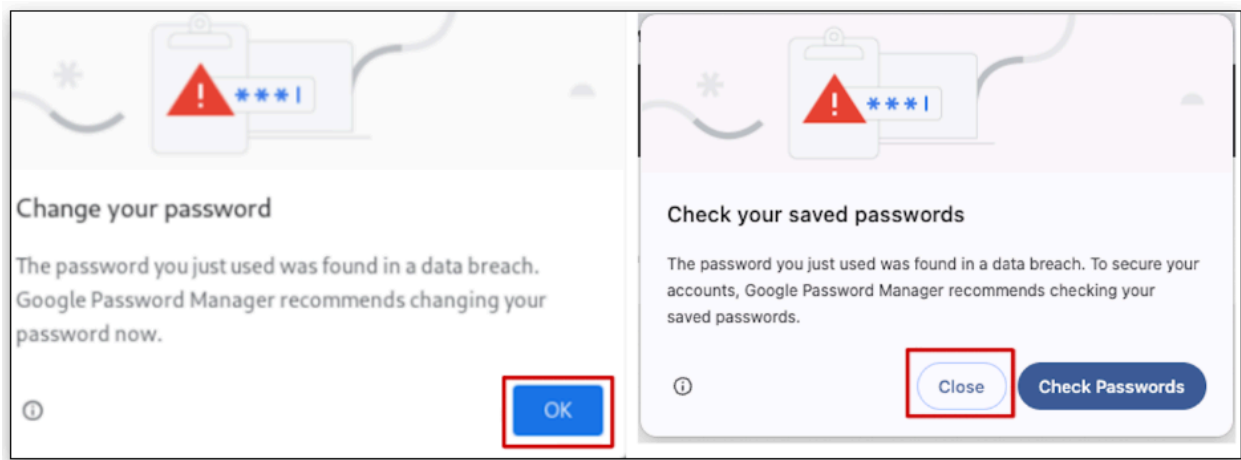
In a new window or tab, open the following URL and log in:

- The Red Hat OpenShift AI Dashboard URL for our shared environment:
 - <https://rhods-dashboard-redhat-ods-applications.apps.cluster-vrcbt.sandbox2323.opentlc.com/>
- Enter your credentials (as detailed above)
- The result should look like:

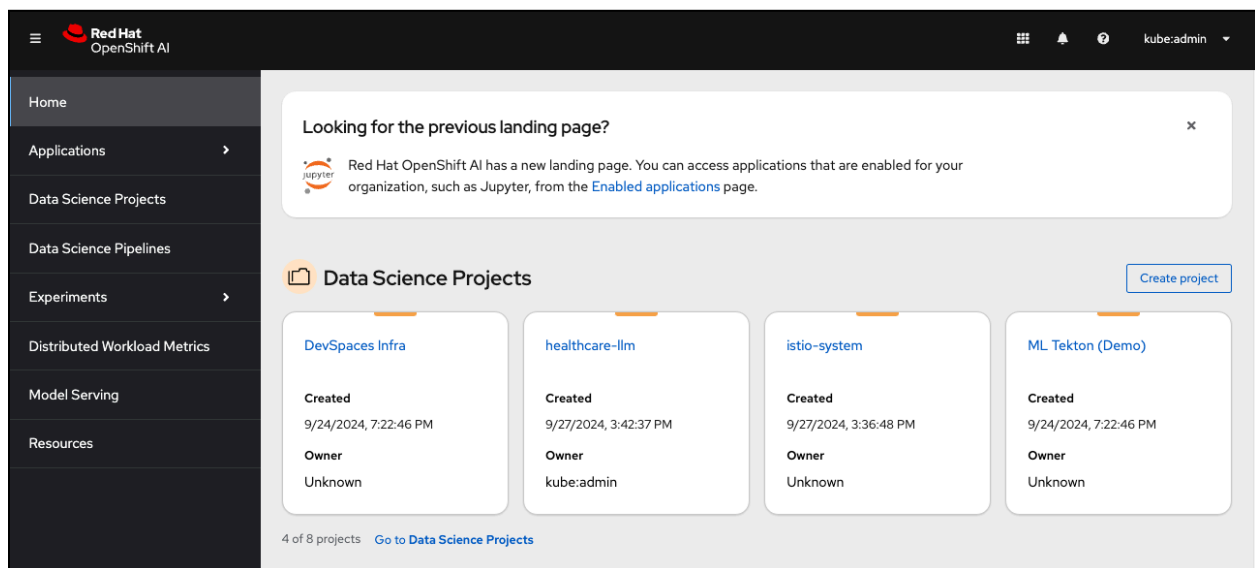


The screenshot shows the Red Hat OpenShift login interface. At the top left is the Red Hat OpenShift logo. The main heading is "Log in to your account". Below this are two input fields: "Username" with the value "userX" and "Password" with masked characters. A blue "Log in" button is positioned below the password field. At the bottom left, there is a "Welcome to Red Hat OpenShift" message. The right side of the image is a large white rectangle, likely representing a second window or a placeholder for the dashboard view after login.

- Because the password is so simple (openshift), your browser might display a scary message such as:



- It is safe here to ignore this message when it pops up.
- After you authenticate, the result should look like:



If you got this far and saw all that, congratulations, you properly connected to the OpenShift AI Dashboard! We are now ready to start the lab.

Creating your project and pipeline server

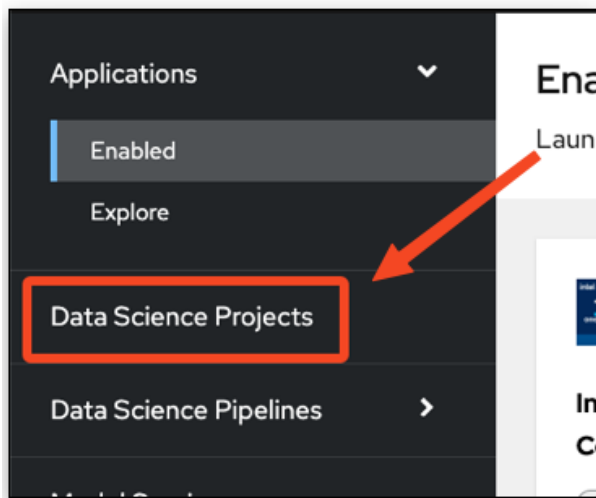
As a preliminary step, each of you is going to:

1. Create a Data Science project
 - this will help keep your things together
2. Launch a Workbench
 - we will use it to review content and notebooks
3. Clone the git repo into your Workbench
 - this contains all the code from the prototype

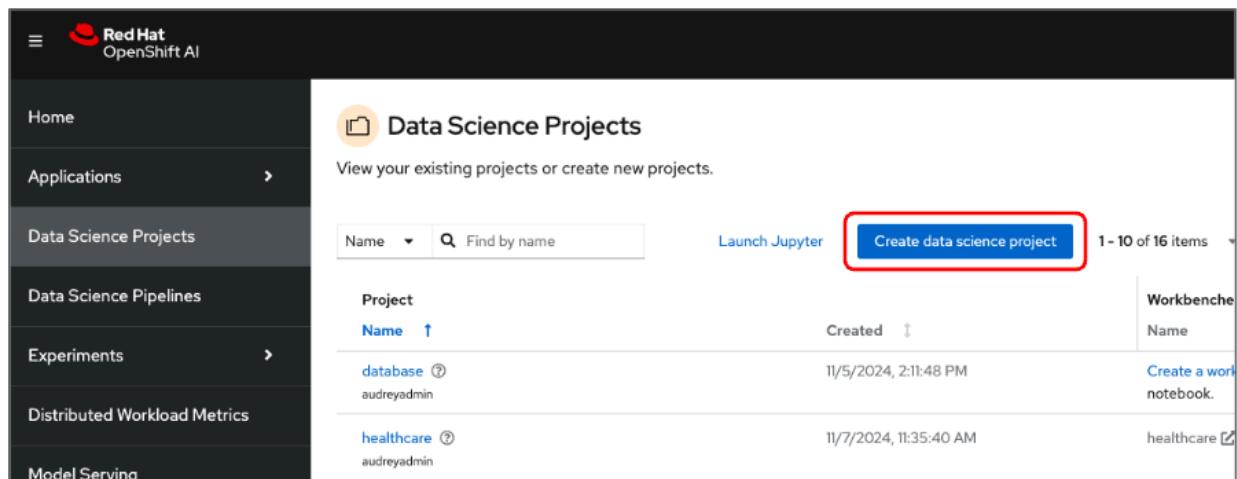
The instructions below will guide you through these steps. Follow them carefully.

Create a project

- First, in the OpenShift AI Dashboard application, navigate to the Data Science Projects menu on the left:



- Click the “Create data science project” button



- Create a project with the same number as in your user id
 - You have been assigned a unique user ID: userX
 - You need to now create a project with the name: patient-chartsX

Create data science project

Name *

Resource name * ?

Must consist of lower case alphanumeric characters or '-', and must start and end with an alphanumeric character

Description

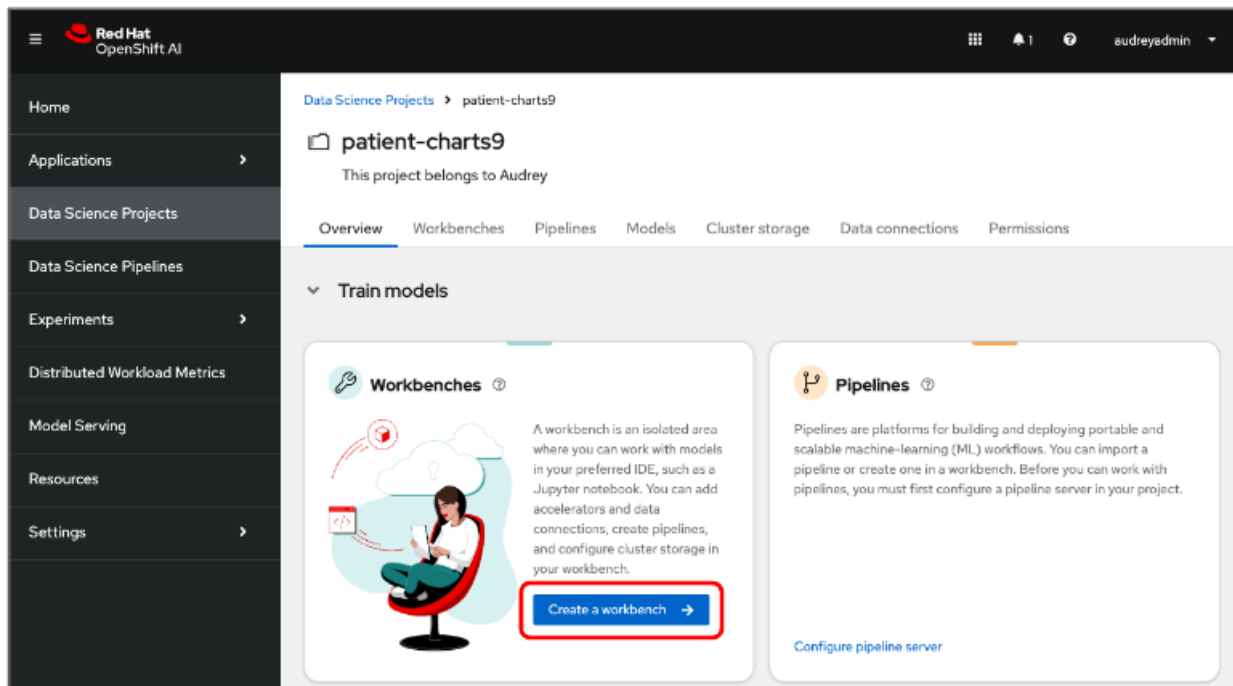
Create

Cancel

IMPORTANT

Your project name should **NOT** be **patient-chartsX**. For you, X should be a number instead.)

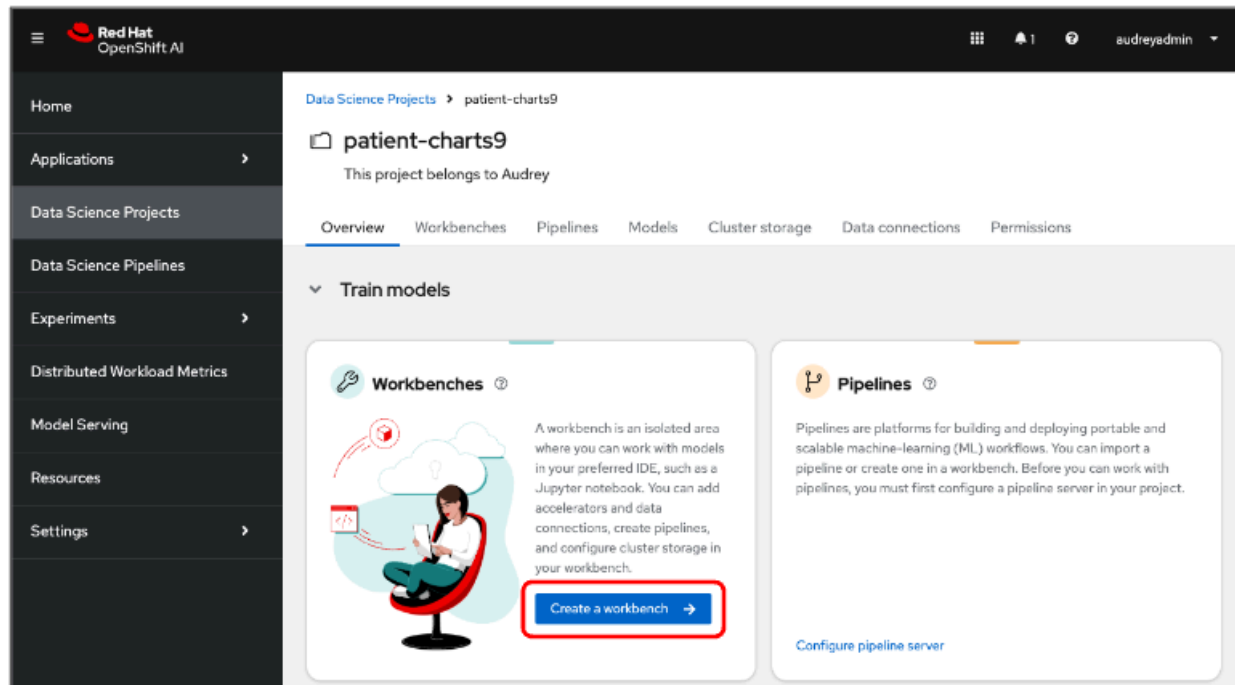
- Leave the resource name unchanged
- Optionally, enter your first and last name in the description of the project.
- You will be brought back to the “Overview” tab in the Red Hat OpenShift AI dashboard



Next you will create a workbench in your project. Click the “Create a Workbench” button to get started.

Creating a workbench

From the Overview tab, Click the “Create a Workbench” button to get started.



- Make sure your workbench has the following characteristics:
 - Choose a name for it, like: `patient-charts`
 - Image Selection: `Standard Data Science`
 - Container Size: `Medium`
 - Accelerator: `None`

IMPORTANT

Do not press the “Create workbench” button - yet!

Name *

patient-charts9

Description

My workbench

Notebook image

Image selection *

Standard Data Science

Version selection *

2024.1

Hover over a version to view its included packages.

[? View package information](#)

Deployment size

Container size

Medium

Accelerator

Next we need to add your unstructured.io API keys as “**Environment variables**”. Scroll down the webpage until you see the “Environment variables” section. Click “**Add variable**”

Environment variables

[+ Add variable](#)

Cluster storage

i Cluster storage will mount to /

☒ Create new persistent storage

This creates storage that is retained when logged out.

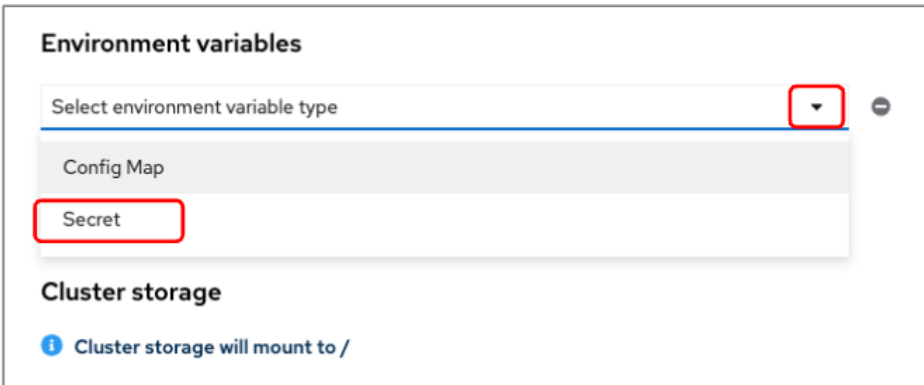
Name *

patient-charts9

IMPORTANT

Before you proceed, make certain that you have an unstructured.io API Key and URL. You can register for them at the unstructured.io website.

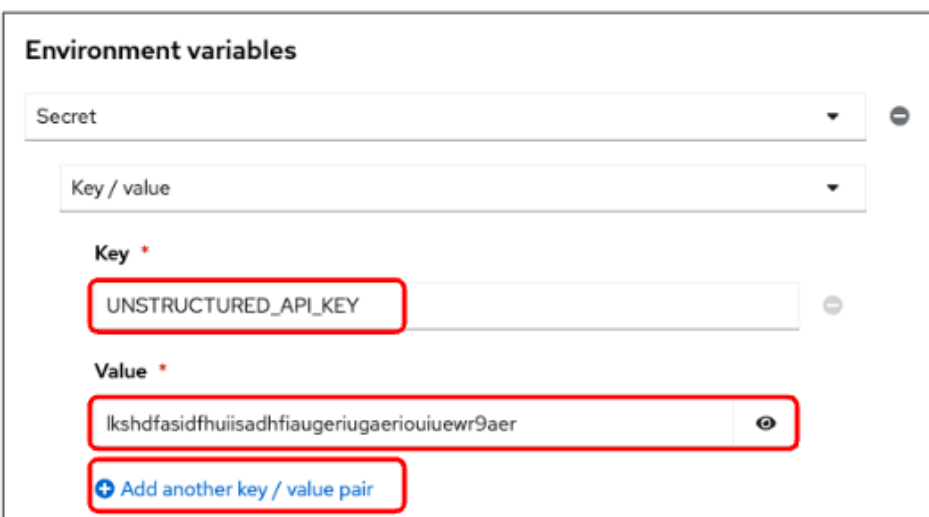
Click the drop down menu "Select environment variable type", and choose "Secret"



The screenshot shows the 'Environment variables' section of a configuration interface. A dropdown menu titled 'Select environment variable type' is open, showing two options: 'Config Map' and 'Secret'. The 'Secret' option is highlighted with a red box. Below the dropdown, the 'Cluster storage' section is visible, with a note that 'Cluster storage will mount to /'.

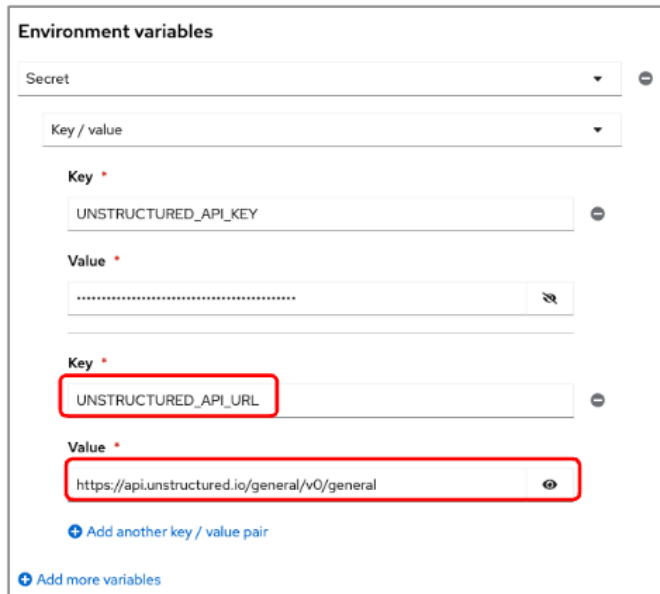
Next, click the drop down menu "Select one", and choose "Key / value".

This is where you will enter your unstructured.io API key and URL. Enter your "UNSTRUCTURED_API_KEY" first.



The screenshot shows the 'Environment variables' section of a configuration interface. The 'Secret' dropdown is selected. Below it, the 'Key / value' dropdown is open, showing 'Key / value' as the selected option. The 'Key' field is labeled 'Key *' and contains the text 'UNSTRUCTURED_API_KEY'. The 'Value' field is labeled 'Value *' and contains a long alphanumeric string. Both fields are highlighted with red boxes. At the bottom, there is a button labeled '+ Add another key / value pair'.

Once you have entered the UNSTRUCTURED_API_KEY, click “Add another key / value pair” You will now create a key called UNSTRUCTURED_API_URL. Add the URL you were given when you signed up for an API key and URL. It may look something like this: `https://api.unstructured.io/general/v0/general`



Environment variables

Secret

Key / value

Key *

UNSTRUCTURED_API_KEY

Value *

Key *

UNSTRUCTURED_API_URL

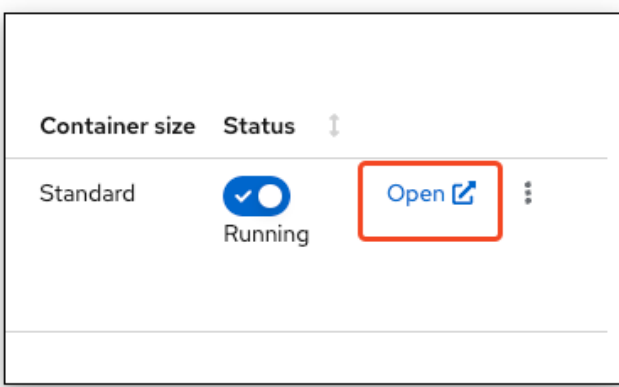
Value *



https://api.unstructured.io/general/v0/general

+ Add another key / value pair

+ Add more variables

- You will not need to modify any other Workbench settings (such as Storage)
- **Click** the “**Create workbench**” button.
- Wait for your workbench to be fully started
- Once it is, click the [Open](#) link to connect to it.



Container size	Status	
Standard	 Running	Open 

- Authenticate with the same credentials as earlier

- You will be asked to accept the following settings. Select “Allow”

Authorize Access

Service account `my-workbench` in project `user8` is requesting permission to access your account (`user8`)

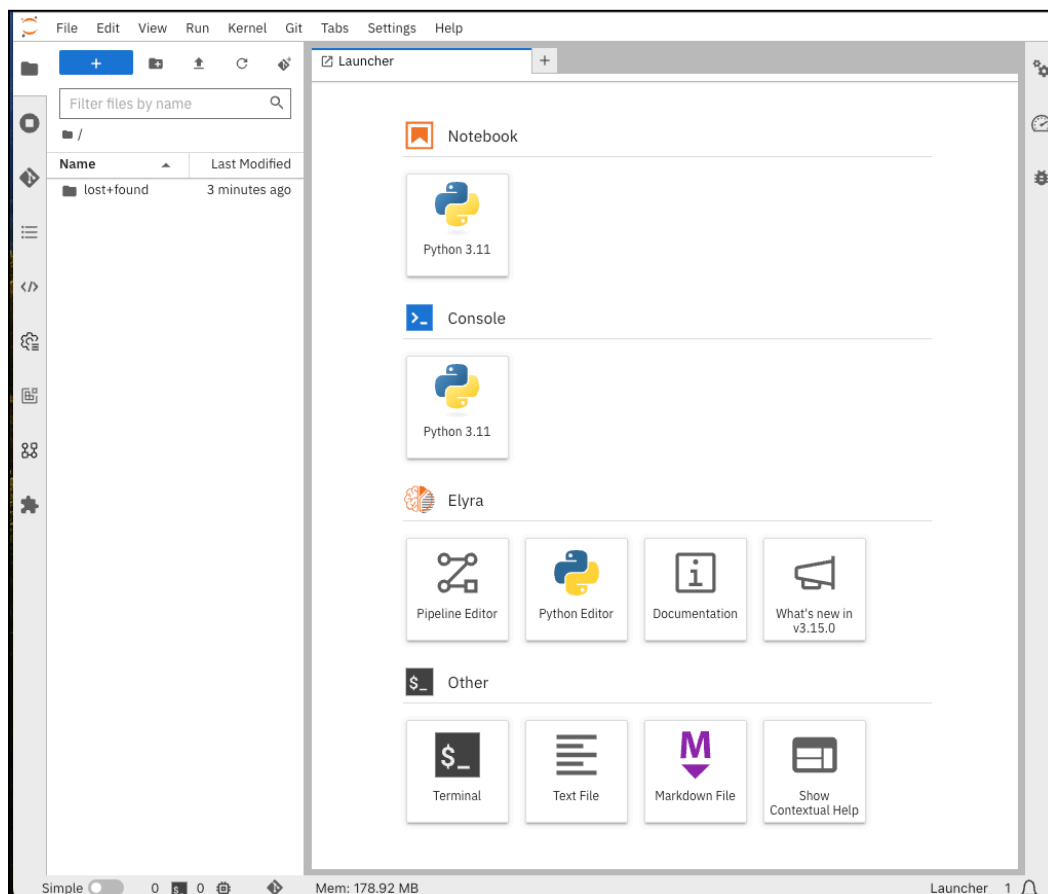
Requested permissions

- ☒ **user:info**
Read-only access to your user information (including username, identities, and group membership)
- ☒ **user:check-access**
Read-only access to view your privileges (for example, "can I create builds?")

You will be redirected to <https://my-workbench-user8.apps.cluster-858zv.sandbox2788.opentlc.com/oauth/callback>

Allow selected permissionsDeny

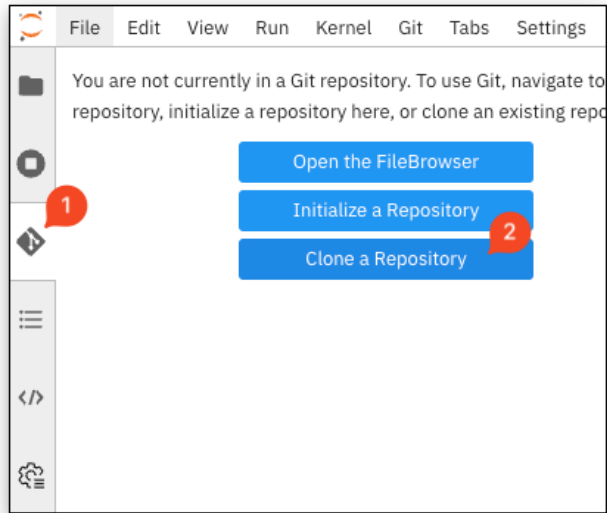
- You will be taken into a Jupyter Lab environment:



Git-Clone the common repo

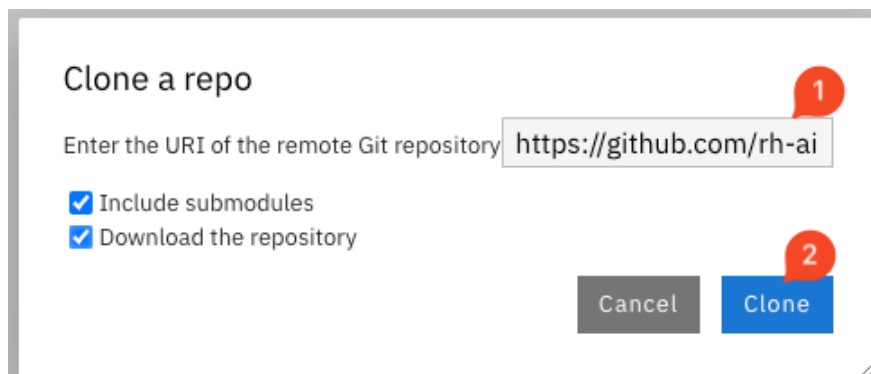
We will clone the content of our Git repo so that you can access all the materials that were created as part of our prototyping exercise.

- Using the Git UI:
 - Open the Git UI in Jupyter:



- Enter the URL of the Git repo:

<https://github.com/redhat-na-ssa/datasci-patient-charts>



At this point, your project is ready for the work we want to do in it.

Visit Information

Provider Information

Encounter Provider

Department

Name

Address

Authorizing Provider

Phone

Fax

Questions/Answers

BILL AREA (PKA PROVIDER ID)

What is the Bill Area for the visit?

Level of Service

Level of Service OFFICE/OUTPAT VISIT

Reason for Visit

Chief Complaints

- Follow-up
- Results

Visit Diagnoses

- Medication monitoring encounter
- Calculation of dose
- Elevated LFTs (primary)

MED GENERAL INTERNAL MEDICINE BA [160301]

Condition	One year before (%)	One year after (%)
No stimulation	21 (64.6%)	25 (79.4%) *
One stimulation	14 (43.8%)	11 (34.4%)
No stimulation (one or more stimulations)	24 (74.2%)	8 (25.0%) **

intensive and a little harder to run locally. Therefore, we will use apis to access a model created by 'Unstructured.io'.

*If you have not already done so, obtain a **UNSTRUCTURED_API_KEY** and **UNSTRUCTURED_API_URL** from unstructured.io Please go to the following [link](#) and submit a request for free API keys. After submitting your request, you will receive a 'Welcome to Unstructured' email that will contain your api key and url.

Notebook 1-normalize content

In this exercise, we will use a notebook to investigate how we normalize pdf data.

From the `datasci-patient-charts` folder, please open the notebook called `01-normalize-content.ipynb` and follow the instructions. Also make certain to read the notes provided within the notebook.

When done, you can close the notebook and head to the next section (Metadata extraction & chunking).

Metadata extraction & chunking

In this exercise, we will use a notebook to investigate how we extract metadata from our pdf and chunk it.

We 'chunk' the data so that it can be stored in a vector database for future LLM usage. We will conduct a hybrid search where we will ask a question about a particular medical section in our patient chart.

Note: we will use '**chromadb**' which is an in-memory (vector) database for a similarity search. If you don't know what a vector database is, read the following Wikipedia article.

'A vector database, vector store or vector search engine is a [database](#) that can store vectors (fixed-length lists of numbers) along with other data items. Vector databases typically implement one or more [Approximate Nearest Neighbor](#) (ANN) algorithms,^{[1][2]} so that one can search the database with a query vector to retrieve the closest matching database records.'

Vectors are mathematical representations of data in a high-dimensional space. In this space, each dimension corresponds to a [feature](#) of the data, with the number of dimensions ranging from a few hundred to tens of thousands, depending on the complexity of the data being represented. A vector's position in this space represents its characteristics. Words, phrases, or entire documents, as well as images, audio, and other types of data, can all be vectorized.^[3]

These feature vectors may be computed from the raw data using machine learning methods such as [feature extraction](#) algorithms, [word embeddings](#)^[4] or [deep learning](#) networks. The goal is that semantically similar data items receive feature vectors close to each other.

Vector databases can be used for [similarity search](#), [multi-modal search](#), [recommendations engines](#), [large language models](#) (LLMs), etc.^[5]

Vector databases are also often used to implement [Retrieval-Augmented Generation](#) (RAG), a method to improve domain-specific responses of large language models. The retrieval component of a RAG can be any search system, but is most often implemented as a vector database. Text documents describing the domain of interest are collected, and for each document or document section, a feature vector (known as an "[embedding](#)") is computed, typically using a deep learning network, and stored in a vector database. Given a user prompt, the feature vector of the prompt is computed, and the database is queried to retrieve the most relevant documents. These are then automatically added into the context window of the large language model, and the large language model proceeds to create a response to the prompt given this context.^[6] [Wikipedia, 2024](#)

We will conduct a hybrid search where we will ask a question about a particular medical section in our patient chart.

When we preprocess our pdf, we will get a metadata field called '**parent id**' that attaches each element to a title which is the title of a section. And you'll use that metadata in order to construct a hybrid search where you search for content within a particular section or a chapter in this case, within the document.

Notebook 2-metadata extraction & chunking

From the `datasci-patient-charts` folder, please open the notebook called `02-metadata-extraction-chunking.ipynb` and follow the instructions. Also make certain to read the notes provided within the notebook.

You will see the following code sections/results in the notebook:

1. Run the document through the Unstructured.io API

```
Run the document through the Unstructured API

[69]: filename = "example_files/CP_CHRT_C_G4M3BA_De-identified.pdf"

with open(filename, "rb") as f:
    files=shared.Files(
        content=f.read(),
        file_name=filename,
    )

req = shared.PartitionParameters(files=files)

[70]: try:
      resp = s.general.partition(req)
      except SDKError as e:
          print(e)

[71]: JSON(json.dumps(resp.elements[0:29], indent=2))

[71]: ▼ root [] 29 items
      ▼ 0
      type "Title"
      element_id "8065fe19f6274b393812d5ec683501ed"
      text "PAST MEDICAL HISTORY"
      ► metadata
      ▼ 1
      type "NarrativeText"
      element_id "4add12fae58d7c7be8c7224cd8ca5810"
      text "has a past medical history of Anxiety, Disease of thyroid gland, Migraine, Obstructive sleep apnea i Prediabetes."
      ► metadata
      ▼ 2
      type "NarrativeText"
      element_id "6c4310489304396f7ecd527b21e5bfa0"
      text "She has no past medical history of Angina pectoris (CMS/HCC), Arthritis, Asthma, Atrial fibrillation (CMS/HCC), Awareness under anesthesia, Basal cell carcinoma, Cancer (CMS/HCC), Chronic kidney disease, Chronic pain disorder, Chronic renal failure, COPD (chronic obstructive pulmonary disease) (CMS/HCC), Deep vein thrombosis (CMS/HCC), Delayed emergence from general anesthesia, Depression, Diabetes mellitus type I (CMS/HCC), Diabetic retinopathy (CMS/HCC), Dry eyes, Epilepsy (CMS/HCC), Eye trauma, GERD (gastroesophageal reflux disease), Glaucoma, Hard to intubate, Heart disease, Heart murmur, Hiatal hernia, HIV disease (CMS/HCC), Hypertension, Hypertensive retinopathy, Infectious viral hepatitis, Macular degeneration, Malignant hyperthermia, Melanoma (CMS/HCC), Mitral valve prolapse, Motion sickness, Myocardial infarction (CMS/HCC), Parkinson's disease (CMS/HCC), Peptic ulceration, PONV (postoperative nausea and vom
```

2. Find the Elements associated with Patient Chart Sections

Find elements associated with chapters

```
[73]: [x for x in resp.elements if x['type'] == 'Title' and 'surgical history' in x['text'].lower()]

[73]: [{'type': 'Title',
      'element_id': '8826b0eb28717fcdab587ef17aad048',
      'text': 'SURGICAL HISTORY',
      'metadata': {'languages': ['eng']},
      'page_number': 1,
      'filename': 'CP_CHRT_C_G4M3BA_De-identified.pdf',
      'filetype': 'application/pdf'}]]

[74]: chapters = [
      "PAST MEDICAL HISTORY",
      "VACCINE HISTORY",
      "SURGICAL HISTORY",
      "SOCIAL HISTORY",
      "VITALS",
      "VIDEO EXAM VIA TELEMEDICINE",
      "ASSESSMENT & PLAN",
      "FOLLOW UP",
      "SIGNATURE",
    ]

[75]: chapter_ids = {}
      for element in resp.elements:
          for chapter in chapters:
              if element["text"] == chapter and element["type"] == "Title":
                  chapter_ids[element["element_id"]] = chapter
                  break

[79]: chapter_ids

[79]: {'8065fe19f6274b393812d5ec683501ed': 'PAST MEDICAL HISTORY',
      '9490c17b6936e11e95415f1f2402a20e': 'VACCINE HISTORY',
      '8826b0eb28717fcdab587ef17aad048': 'SURGICAL HISTORY',
      'c49eae348c580cbac4c1818d5c476d8': 'SOCIAL HISTORY',
      '03dee2ccc09edd9fd56174730feaf38': 'VITALS',
      'f35d3a414cc9e51b70144b41a04f012e': 'VIDEO EXAM VIA TELEMEDICINE',
      'a13ea5a32e955df03cd3fd79303acf64': 'ASSESSMENT & PLAN',
      'b0f68271eef797f6ba310a59ad53ee1d': 'FOLLOW UP',
      '45a7c54c84f0cbde87f6c2f6d221058f': 'SIGNATURE'}
```

3. Load the documents into a vector database & see the associated elements in the vector database.

Load documents into a vector db

```
[81]: client = chromadb.PersistentClient(path="chroma_tmp", settings=chromadb.Settings(allow_reset=True))
      client.reset()
```

```
[81]: True
```

```
[82]: collection = client.create_collection(
      name="medical_chart",
      metadata={"hnsw:space": "cosine"}
      )
```

 **Note (Wait Time)** : The following block can take a few minutes to complete.

```
[83]: for element in resp.elements:
      parent_id = element["metadata"].get("parent_id")
      chapter = chapter_ids.get(parent_id, "")
      collection.add(
          documents=[element["text"]],
          ids=[element["element_id"]],
          metadatas=[{"chapter": chapter}]
      )
```

See the elements in Vector DB

```
[84]: results = collection.peek()
      print(results["documents"])
```

```
['VITALS', 'Procedure: neck exploration with para thyroidectomy - Surgeon: 1 - Service: ENT', '2. H
ypothyroidism, unspecified type', 'Surgeon: - Service: Ophthalmology', 'care. Patient's question(s)
were answered', '1. Polio 2. Measles 3. ChickenPox 4. COVID 5. Tetanus, Diphtheria 6. Flu Shot 7. Y
ellow Fever 8. Rabies', '2. Pr strabismus surg', 'respiratory distress. Cough was not audible duri
ng the video visit.', 'LORazepam; TAKE 1 TO 2 TABLETS BY MOUTH EVERY 6 HOURS AS NEEDED -FOR ANXIETY
Dispense: 30 tablet; Refill: 0', 'SIGNATURE']
```

4. Perform a Hybrid Search with metadata

Perform a hybrid search with metadata

```
[85]: result = collection.query(
      query_texts=["Did patient have a skin graft?"],
      n_results=2,
      where={"chapter": "SURGICAL HISTORY"},
    )
    print(json.dumps(result, indent=2))

    {
      "ids": [
        [
          "065f00a9071c0b85490e660a20fe4f79",
          "869ef926671b9229bef3fe88d110f52f"
        ]
      ],
      "distances": [
        [
          0.6499665575757073,
          0.8157684454896992
        ]
      ],
      "metadatas": [
        [
          {
            "chapter": "SURGICAL HISTORY"
          },
          {
            "chapter": "SURGICAL HISTORY"
          }
        ]
      ],
      "embeddings": null,
      "documents": [
        [
          "Procedure: neck exploration with para thyroidectomy - Surgeon: 1 - Service: ENT",
          "1. Pr explore parathyroid glands n/a"
        ]
      ],
      "uris": null,
      "data": null
    }
```

5. Start Chunking Content

Chunking Content

```
[86]: elements = dict_to_elements(resp.elements)

[87]: chunks = chunk_by_title(
    elements,
    combine_text_under_n_chars=100,
    max_characters=3000,
)

[88]: JSON(json.dumps(chunks[0].to_dict(), indent=2))

[88]: ▼ root Find...
    type "CompositeElement"
    element_id "b756d3a7d81256e790f06f0dddc425ee"
    text "PAST MEDICAL HISTORY has a past medical history of Anxiety, Disease of thyroid gland, Migraine, Obstructive sleep apnea, Prediabetes. She has no past medical history of Angina pectoris (CMS/HCC), Arthritis, Asthma, Atrial fibrillation (CMS/HCC), Awareness under anesthesia, Basal cell carcinoma, Cancer (CMS/HCC), Chronic kidney disease, Chronic pain disorder, Chronic renal failure, COPD (chronic obstructive pulmonary disease) (CMS/HCC), Deep vein thrombosis (CMS/HCC), Delayed emergence from general anesthesia, Depression, Diabetes mellitus type I (CMS/HCC), Diabetic retinopathy (CMS/HCC), Dry eyes, Epilepsy (CMS/HCC), Eye trauma, GERD (gastroesophageal reflux disease), Glaucoma, Hard to intubate, Heart disease, Heart murmur, Hiatal hernia, HIV disease (CMS/HCC), Hypertension, Hypertensive retinopathy, Infectious viral hepatitis, Macular degeneration, Malignant hyperthermia, Melanoma (CMS/HCC), Mitral valve prolapse, Motion sickness, Myocardial infarction (CMS/HCC), Parkinson's disease (CMS/HCC), Peptic ulceration, PONV (postoperative nausea and vomiting), Pseudocholinesterase deficiency, attack), Tuberculosis, Type 2 diabetes mellitus (CMS/HCC), or Valvular disease. PAST"
    ► metadata

[89]: len(elements)

[89]: 39

[90]: len(chunks)

[90]: 8
```

Why is data preprocessing for LLMs so challenging?

Structuring data is getting easier but there are still many challenges, especially in the Healthcare industry with the sheer number of different document types. And even though we are seeing a number of different structuring document techniques, in healthcare data preprocessing is still challenging.

1. **Content Cues** – different document types have different cues for element types (visual, markdown)
2. **Standardization Need** – to process content from different document types, they need to be standardized
3. **Extraction Variability** – Different document formats may require different extraction approaches (i.e. forms vs journal articles)
4. **Metadata Insight** – In many cases, extracting metadata requires an understanding of document structure.

With that in mind, there are great opportunities in preprocessing and using healthcare data.