Get data into Label Studio

Get data into Label Studio by importing files, referencing URLs, or syncing with cloud or database storage.

- If your data is stored in a cloud storage bucket, see Sync data from cloud or database storage.
- If your data is stored in a Redis database, see Sync data from cloud or database storage.
- If your data is stored at internet-accessible URLs, in files, or directories, import it from the Label Studio UI.
- If your data is stored locally, import it into Label Studio.
- If your data contains predictions or pre-annotations, see Import pre-annotated data into Label Studio.

Enterprise

If your data is stored in Google Cloud, AWS, or Azure, you can import your unstructured data as a dataset in Label Studio Enterprise.

From here, you can use semantic search and similarity search to curate data for labeling, which can then be added to different projects as tasks. For more information, see Data Discovery overview.

General guidelines for importing data

- It's best to keep about 100k tasks / 100k annotations per project for optimal performance.
- Avoid frequent imports because each new import requires lengthy background operations. One import per 30 seconds will work without overloads.

> ⚠ Warning
>
> For large projects or business critical projects, do not upload media files through the Label Studio interface. This is especially true for files such as images, audio, video, timeseries, etc.
>
> Uploading data through the Label Studio UI works fine for proof of concept projects, but it is not recommended for larger projects. You will also face challenges when you want export your data or move it to another Label Studio instance or even just redeploy Label Studio. Finally, Label Studio is not designed as a hosting service at scale and does not have backups for imported media resources.
>
> We strongly recommend that you configure source storage instead.

Types of data you can import into Label Studio

You can import many types of data, including text, timeseries, audio, and image data. The file types supported depend on the type of data.

Data type	Supported file types
Audio	.flac, .m4a, .mp3, .ogg, .wav
HyperText (HTML)	.html, .htm, .xml
Images	.bmp, .gif, .jpg, .png, .svg, .webp
Paragraphs (Dialogue)	.json
Structured data	.csv, .tsv
Text	.txt, .json
Time series	.csv, .tsv, .json
Tasks with multiple data types	.csv, .tsv, .json
Video	.mp4, .webm

If you don't see a supported data or file type that you want to import, please let us know by submitting an issue to the Label Studio Repository.

How to import your data

The most secure and reliable method to import your data is to store the data outside of Label Studio and import references to the data using URLs. You can import a list of URLs in a TXT, CSV, or TSV file, or reference the URLs in JSON task format.

If you're importing audio, image, or video data, you must use URLs to refer to those data types.

If you're importing HTML, text, dialogue, or timeseries data using the `<HyperText>`, `<Text>`, `<Paragraphs>`, or `<TimeSeries>` tags in your labeling configuration, you can either load data directly, or load data from a URL.

- To load data from a URL, specify `valueType="url"` in your labeling configuration.
- To load data directly into the Label Studio database, specify `valueType="text"` for `HyperText` or `Text` data, or `valueType="json"` for `Paragraph` or `TimeSeries` data.

Note

If you load data from a URL, the data is not saved in Label Studio. If you want an annotated task export to include the data that you annotated, you must import the data into the Label Studio database without using URL references, or combine the data with the annotations after exporting.

▶ **Click to expand example configurations with each valueType**

How to retrieve data

There are several steps to retrieve the data to display in the `Object` tag. The data retrieval is also used in dynamic choices and labels. Use the following parameters in the `Object` tag.

`value` (required)

The `value` parameter represents the source of the data. It can be plain text or a step of complex data retrieval system. It can be denoted using the following forms: `value` (required)

Variables

In most cases, the `Object` tag has the value with one variable (prefixed with a $) in it.

For example, `<Audio value="$audio" ... />` seeks the “audio” field in the imported JSON object:

```json
{
  "data": {
    "audio": "https://host.name/myaudio.wav"
  }
}
```

Plain text

The value parameter can be a string. It is useful for `Header` and `Text`.

Also, you can use the content of the tag as value. It is useful for descriptive text tags and is applied for `Label` and `Choice`.

For example:

```xml
<Header>Label audio:</Header>
<Header value="Label only fully visible cars" />
<Text name="instruction" value="Label only fully visible cars" />
<Label>cat</Label>

<Choice>other</Choice>
```

Other cases

1. The `value` parameter can be a text containing variables prefixed by $.

   For example:

   ```xml
   <Header value="url: $image"/>
   ```

2. The `value` parameter can also refer to nested data in arrays and dicts (`$texts[2]` and `$audio.url`).

   For example:

   ```xml
   <Image name="image" value="$images[0]"/>
   ```

`valueType` (optional)

The `valueType` parameter defines how to treat the data retrieved from the previous steps. There are two options such as the “url” and raw data. Currently the raw data input can be “text” or “json”. The “text” is used for `HyperText` and `Text` tags and “json” is used for `TimeSeries` tag.

For example:

- Using “url”: `<Text name="text1" value="$text" valueType="url"/>` displays the text loaded by the URL.
- Using “text”: `<Text name="text" value="$text" valueType="text"/>` displays the URL without loading the text.

`resolver` (optional)

Use this parameter to retrieve data from multi-column csv on S3 or other cloud storage. Label Studio can retrieve it only in run-time, so it’s secure.

If you import a file with a list of tasks, and every task in this list is a link to another file in the storage. In this case, you can use the `resolver` parameter to retrieve the content of these files from a storage.

Use Case

There is a list of tasks, where the “remote” field of every task is a link to a CSV file in the storage. Every CSV file has a “text” column with text to be labeled. Every CSV file has a “text” column with text to be labeled. For example:

Tasks:

```json
[
    { "remote": "s3://bucket/text1.csv" },
    { "remote": "s3://bucket/text2.csv" }
]
```

CSV file:

```csv
id;text
12;The most flexible data annotation tool. Quickly installable. Build custom
```

Solution

To retrieve the file, use the following parameters:

1. `value="$remote"`: The URL to CSV on S3 is in "remote" field of task data. If you use the `resolver` parameter the `value` is always treated as URL, so you don't need to set `valueType`.
2. `resolver="csv|separator=;|column=text"`: Load this file in run-time, parse it as CSV, and get the "text" column from the first row.
3. Display the result.

Syntax

The syntax for the `resolver` parameter consists of a list of options separated by a `|` symbol.

The first option is the type of file.

> **Note**
>
> Currently, only CSV files are supported.

The remaining options are parameters of the specified file type with optional values. The parameters for CSV files are:

- `headless`: A CSV file does not have headers (this parameter is boolean and can't have a value).
- `separator=;`: CSV separator, usually can be detected automatically.
- `column=1`: In `headless` mode use zero-based index, otherwise use column name.

For example, `resolver="csv|headless|separator=;|column=1"`

How to format your data to import it

Label Studio treats different file types different ways.

If you want to import multiple types of data to label at the same time, for example, images with captions or audio recordings with transcripts, you must use the basic Label Studio JSON format.

You can also use a CSV file or a JSON list of tasks to point to URLs with the data, rather than directly importing the data if you need to import thousands of files. You can import files containing up to 250,000 tasks or up to 50MB in size into Label Studio.

If you're specifying data in a cloud storage bucket or container, and you don't want to sync cloud storage, create and specify presigned URLs for Amazon S3 storage, signed URLs for Google Cloud Storage, or shared access signatures for Microsoft Azure in a JSON, CSV, TSV or TXT file.

Basic Label Studio JSON format

The best way to import data into Label Studio is to use a JSON-formatted list of tasks. The `data` key of the JSON file references each task as an entry in a JSON dictionary. If there is no `data` key, Label Studio interprets the entire JSON file as one task.

In the `data` JSON dictionary, use key-value pairs that correspond to the source key expected by the object tag in the label configuration that you set up for your project.

Depending on the type of object tag, Label Studio interprets field values differently:

- `<Text value="$key">`: `value` is interpreted as plain text.
- `<HyperText value="$key">`: `value` is interpreted as HTML markup.
- `<HyperText value="$key" encoding="base64">`: `value` is interpreted as a base64 encoded HTML markup.
- `<Audio value="$key">`: `value` is interpreted as a valid URL to an audio file with CORS policy enabled on the server side.
- `<Image value="$key">`: `value` is interpreted as a valid URL to an image file
- `<TimeSeries value="$key">`: `value` is interpreted as a valid URL to a CSV/TSV file if `valueType="url"`, otherwise it is interpreted as a JSON dictionary with column arrays: `"value": {"first_column": [...], ...}` if `valueType="json"`. See more about how to use valueType.

You can add other, optional keys to the JSON file.

JSON key	Description
annotations	Optional. List of annotations exported from Label Studio. Label Studio's annotation format allows you to import annotation results in order to use them in subsequent labeling tasks.
predictions	Optional. List of model prediction results, where each result is saved using Label Studio's prediction format. Import predictions for automatic task pre-labeling and active learning. See Import predicted labels into Label Studio

See Relevant JSON property descriptions in the export documentation for more details about the JSON format of exported tasks.

Example JSON format

For an example text classification project, you can set up a label config like the following:

```xml
<View>
  <Text name="message" value="$my_text"/>
  <Choices name="sentiment_class" toName="message">
    <Choice value="Positive"/>
    <Choice value="Neutral"/>
    <Choice value="Negative"/>
  </Choices>
</View>
```

You can then import text tasks to label that match the following JSON format:

```yaml
[{
  # "data" must contain the "my_text" field defined in the text labeling con
  "data": {
    "my_text": "Opossums are great",
    "ref_id": 456,
    "meta_info": {
      "timestamp": "2020-03-09 18:15:28.212882",
      "location": "North Pole"
    }
  },

  # annotations are not required and are the list of annotation results matc
```

```
  "annotations": [{
    "result": [{
      "from_name": "sentiment_class",
      "to_name": "message",
      "type": "choices",
      "readonly": false,
      "hidden": false,
      "value": {
        "choices": ["Positive"]
      }
    }]
  }],

  # "predictions" are pretty similar to "annotations"
  # except that they also include some ML-related fields like a prediction "
  "predictions": [{
    "result": [{
      "from_name": "sentiment_class",
      "to_name": "message",
      "type": "choices",
      "readonly": false,
      "hidden": false,
      "value": {
        "choices": ["Neutral"]
      }
    }],
  # score is used for active learning sampling mode
    "score": 0.95
  }]
}]
```

If you're placing JSON files in cloud storage, place 1 task in each JSON file in the storage bucket. If you want to upload a JSON file from your machine directly into Label Studio, you can place multiple tasks in one JSON file and import it using Label Studio GUI (Data Manager => Import button).

Example JSON with multiple tasks

You can place multiple tasks in one JSON file if you're uploading the JSON file using Label Studio Import Dialog only (Data Manager => Import button).

▶ **To place multiple tasks in one JSON file, use this JSON format example**

Example JSON for older versions of Label Studio

If you're still using a Label Studio version earlier than 1.0.0, refer to this example JSON format.

▶ **For versions of Label Studio earlier than 1.0.0, use this JSON format example.**

Import CSV or TSV data

When you import a CSV / TSV formatted text file, Label Studio interprets the column names are as task data keys that correspond to the labeling config you set up:

```csv
my_text,optional_field
this is a first task,123
this is a second task,456
```

Note

If your labeling config has a `TimeSeries` tag, Label Studio interprets the CSV/TSV as time series data when you import it. This CSV/TSV is hosted as a resource file and Label Studio automatically creates a task with a link to the uploaded CSV/TSV.

Plain text

Import data as plain text. Label Studio interprets each line in a plain text file as a separate data labeling task.

You might use plain text for labeling tasks if you have only one stream of input data, and only one object tag specified in your label config.

```text
this is a first task
this is a second task
```

If you want to import entire plain text files without each line becoming a new labeling task, customize the labeling configuration to specify `valueType="url"` in the Text tag. See the Text tag documentation. See more about how to use the valueType field.

Import HTML data

You can import `HyperText` data in HTML-formatted files and annotate them in Label Studio. When you directly import HTML files, the content is minified by compressing the text, removing whitespace and other nonfunctional data in the HTML code. Annotations that you create are applied to the minified version of the HTML.

If you want to label HTML files without minifying the data, you can do one of the following:

- Import the HTML files as BLOB storage from external cloud storage such as Amazon S3 or Google Cloud Storage.
- Update the `HyperText` tag in your labeling configuration to specify `valueType="url"` as described in How to import your data on this page.

Import data from a local directory

To import data from a local directory, you have two options:

- Run a web server to generate URLs for the files, then upload a file that references the URLs to Label Studio.
- Add the file directory as a source or target local storage connection in the Label Studio UI.

Run a web server to generate URLs to local files

To run a web server to generate URLs for the files, you can refer to this provided helper shell script in the Label Studio repository or write your own script. Use that script to do the following:

1. On the machine with the file directory that you want Label Studio to import, call the helper script and specify a regex pattern to match the files that you want to import. In this example, the script identifies files with the JPG file extension:

```bash
./script/serve_local_files.sh <directory/with/files> *.jpg
```

The script collects the links to the files provided by that HTTP server and saves them to a `files.txt` file with one URL per line.

2. Import the file with URLs into Label Studio using the Label Studio UI.

> **Note**
>
> You must keep the web server running while you perform your data labeling so that the URLs remain accessible to Label Studio.

If your labeling configuration supports HyperText or multiple data types, use the Label Studio JSON format to specify the local file locations instead of a `txt` file. See an example of this format.

If you serve your data from an HTTP server created like follows: `python -m http.server 8081 -d`, you might need to set up CORS for that server so that Label Studio can access the data files successfully. If needed, run the following from the command line:

```bash
npm install http-server -g
http-server -p 3000 --cors
```

Add the file directory as source storage in the Label Studio UI

If you're running Label Studio on Docker and want to add local file storage, you need to mount the file directory and set up environment variables. See Run Label Studio on Docker and use local storage.

Import data from the Label Studio UI

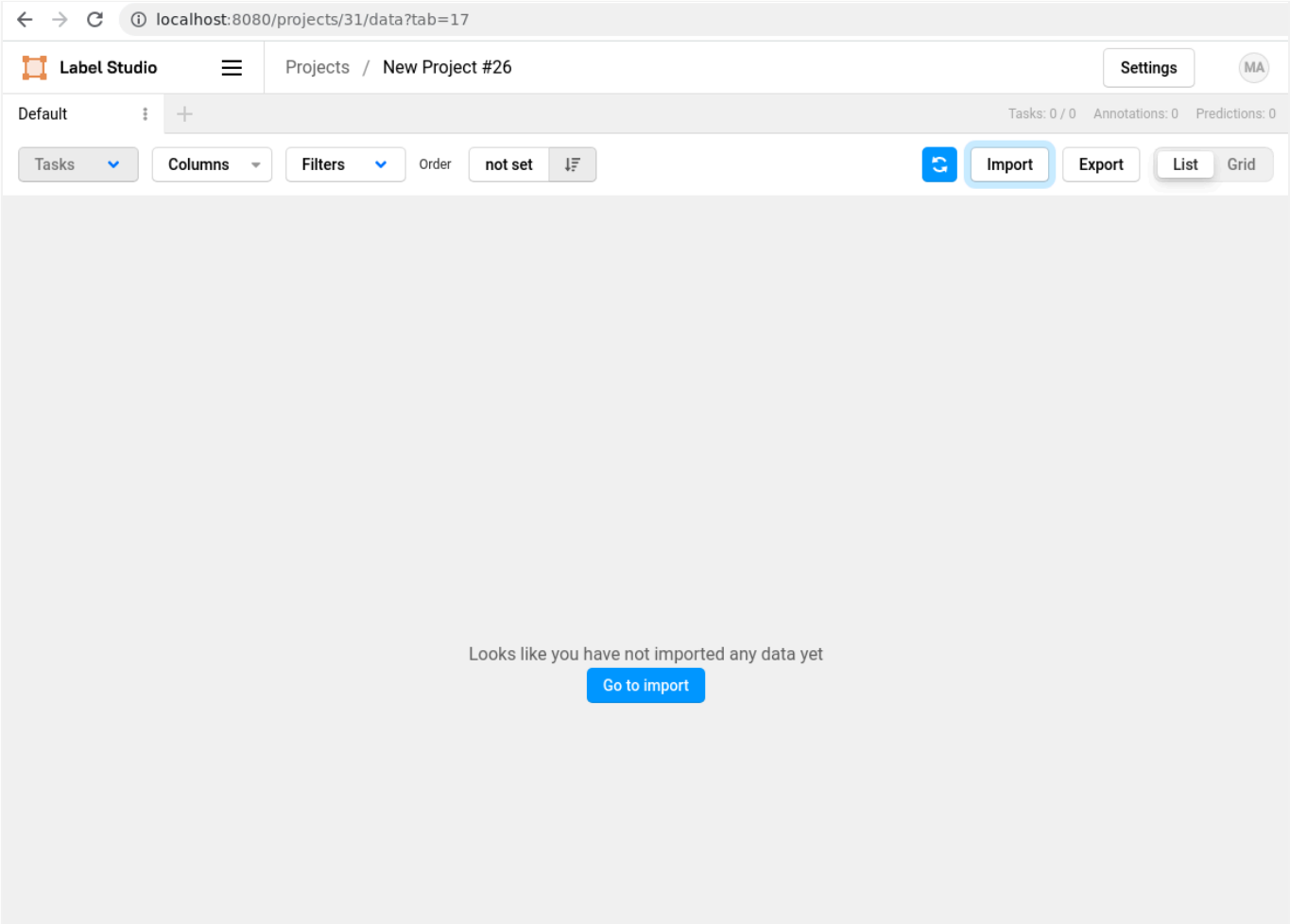
> **Warning**
>
> For large projects or business critical projects, do not upload media files through the Label Studio interface. This is especially true for files such as images, audio, video, timeseries, etc.

> Uploading data through the Label Studio UI works fine for proof of concept projects, but it is not recommended for larger projects. You will also face challenges when you want export your data or move it to another Label Studio instance or even just redeploy Label Studio. Finally, Label Studio is not designed as a hosting service at scale and does not have backups for imported media resources.
>
> We strongly recommend that you configure source storage instead.

To import data from the Label Studio UI, do the following:

1. On the Label Studio UI, open the Data Manager page for a specific project.
2. Click **Import** to open the Import dialog.
3. Import your data from files or URLs.

Data that you import is project-specific.



Import data using the API

Import your data using the Label Studio API. See the API documentation for importing tasks.

Import data from the command line

In versions of Label Studio earlier than 1.0.0, you can import data from a local directory using the command line.

To import data from the command line, do the following:

1. Start Label Studio and use command line arguments to specify the path to the data and format of the data.
   For example:
   `label-studio init --input-path my_tasks.json --input-format json`
2. Open the Label Studio UI and confirm that your data was properly imported.

You can use the `--input-path` argument to specify a file or directory with the data that you want to label. You can specify other data formats using the `--input-format` argument. For example run the following command to start Label Studio and import audio files from a local directory:

```bash
label-studio init my-project --input-path=my/audios/dir --input-format=audio
```

⚠ Warning

The `--allow-serving-local-files` argument is intended for use only with locally-running instances of Label Studio. Avoid using it for remote servers unless you are sure what you're doing.

By default, Label Studio expects JSON-formatted tasks using the Basic Label Studio JSON format.

If you add more files to a local directory after Label Studio starts, you must restart Label Studio to import the tasks in the additional files.

WAS THIS USEFUL?

Yes No

CONTRIBUTE TO THE DOCS

Our docs are open source. See something that could be improved? Submit a pull request.

Make a contribution

CAN'T FIND WHAT YOU'RE LOOKING FOR?

Submit a GitHub Issue

Join Slack Community

HumanSignal

PRODUCTS

Community Edition

Enterprise

Pricing

COMMUNITY

Blog

Newsletter

Slack

Webinars

DOCUMENTATION

Quickstart

API Reference

SDK Reference

Customizable Tags

Labeling Templates

COMPANY