

JADN Sandbox User Guide

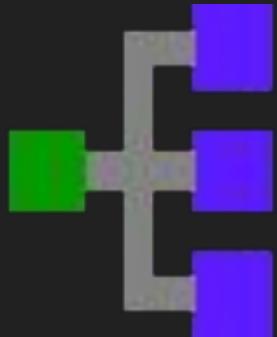


Table of Contents

JADN Sandbox Capabilities

[Schema Creation](#)

[How to get started...](#)

[Schema Visualization](#)

[Schema Translation](#)

[Data Creation](#)

[Data Validation](#)

[Example Data Generation](#)

[Schema Transformation](#)

[Other Features](#)

Docker Desktop

[How to Get the JADN Sandbox Image](#)

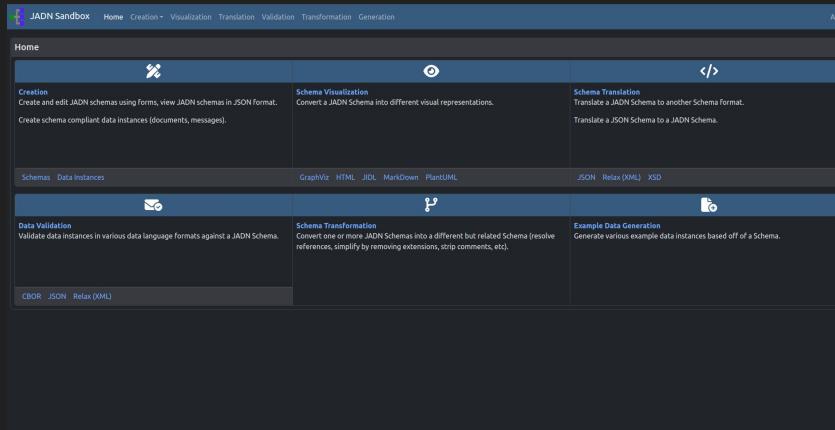
[How to Run the JADN Sandbox Image](#)

[How to Update the JADN Sandbox Image](#)

[How to Stop the JADN Sandbox Image](#)

[How to Learn More about Docker](#)

JADN Sandbox Capabilities



JADN Sandbox Table of Contents

Schema Creation

Feature: Editor Style

To Start

Add Data

Edit Data

Feature: Types Outline

More Functions and Features

Schema Transformation

Result: A Resolved Schema

Result: Strip Comments

Other Features

Theme Switcher

Add Custom Files

Remove Custom Files

Download Files

How to get started...

Schema Visualization

Results: Selecting One Language

Results: Selecting Multiple Languages

Schema Translation

Data Creation

View JSON/Creator

Data Validation

Invalid Results

Valid Results

Example Data Generation

Results

Schema Creation

Create valid JADN Schemas

JADN Sandbox Home Creation Visualization Translation Validation Transformation Generation About

Home Schema Creation Data Creation

Creation
Create and edit JADN schemas using forms, view JADN schemas in JSON format.
Create schema compliant data instances (documents, messages).

Schema Visualization
Convert a JADN Schema into different visual representations.

Schema Translation
Translate a JADN Schema to another Schema format.
Translate a JSON Schema to a JADN Schema.

Schemas Data Instances GraphViz HTML JIDL MarkDown PlantUML JSON Relax (XML) XSD

Data Validation
Validate data instances in various data language formats against a JADN Schema.
CBOR JSON Relax (XML)

Schema Transformation
Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc.).

Example Data Generation
Generate various example data instances based off of a Schema.

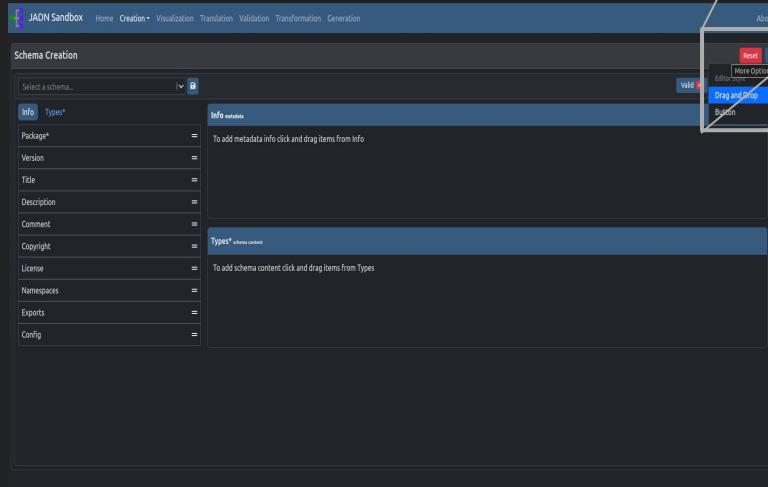
localhost:8082/create/schema v0.10.0_1705587211962

Go to: Creation
When the drop down menu appears, select Schema Creation

Feature: Editor Style

Choose your schema creation editor style:

- Drag and Drop (default)
- Button



Drag and drop style (default)

This image contains two side-by-side screenshots of the JADN Sandbox Schema Creation interface, demonstrating the 'Button' editor style. Both screenshots show the same basic layout with fields for schema selection, package, version, title, etc., and a large blue 'Drag and Drop' button. In the left screenshot, a tooltip above the 'Drag and Drop' button shows the 'Editor Style' options: 'Drag and Drop' (selected) and 'Button'. In the right screenshot, the 'Drag and Drop' button has been replaced by a smaller, rounded rectangular button with the text 'Button' on it. The entire interface has a dark-themed background.

Button style

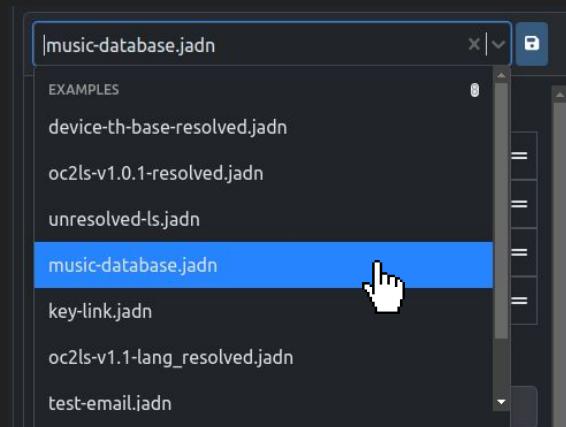
To start...

- 1 Select an example schema
or
- 2 Upload a syntactically valid JADN Schema
or
- 3 From scratch (blank schema)

The screenshot shows the JADN Sandbox interface with the 'Creation' tab selected. In the 'Schema Creation' section, there is a dropdown menu labeled 'Select a schema...' containing several schema files: 'oc2ls-v1.0.1-resolved.jadn', 'unresolved-ls.jadn', 'music-database.jadn' (which is currently selected), 'key-link.jadn', 'oc2ls-v1.1-lang_resolved.jadn', 'test-email.jadn', and 'start-up-template.jadn'. Below the dropdown is a button labeled 'Upload Custom File ...'. To the right of the schema selection area, there are two sections: 'Info metadata' and 'Types* schema content', each with a note: 'To add metadata info click and d' and 'To add schema content click and d' respectively.

Option 1 to Start: Select a Schema

Selecting a preloaded Schema from the drop down menu will automatically generate the Schema in the Schema Creator.



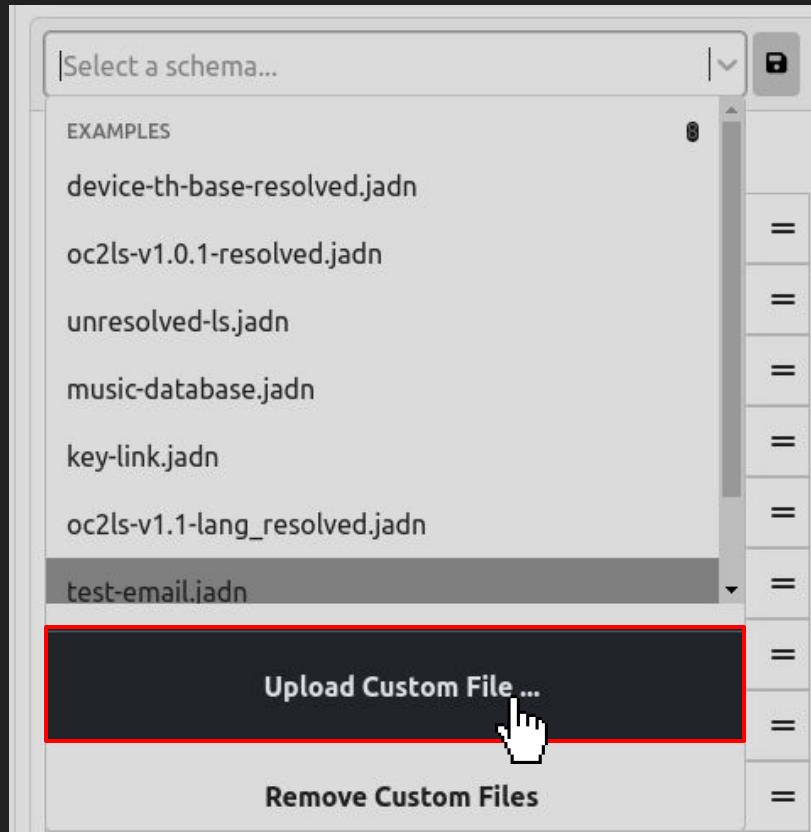
The screenshot shows the JADN editor interface with the following details:

- Title:** music-database.jadn
- Valid:** Valid (green)
- View JSON:** View in JSON (blue)
- Info Tab:** Active tab, showing metadata fields:
 - Comment
 - Copyright
 - Namespaces
 - Config
- Types Tab:** Shows a list of type definitions:
 - Library
 - Barcode
- Outline Tab:** Shows the outline of the schema.
- Search Bar:** Search... (placeholder)
- Exports Section:** Type definitions exported by this module:
 - Library

Option ② to Start: Upload a Schema

Click ‘Upload Customer File...’ to upload a file using the computer’s file loader

This will upload any JADN file and then attempt to validate it.

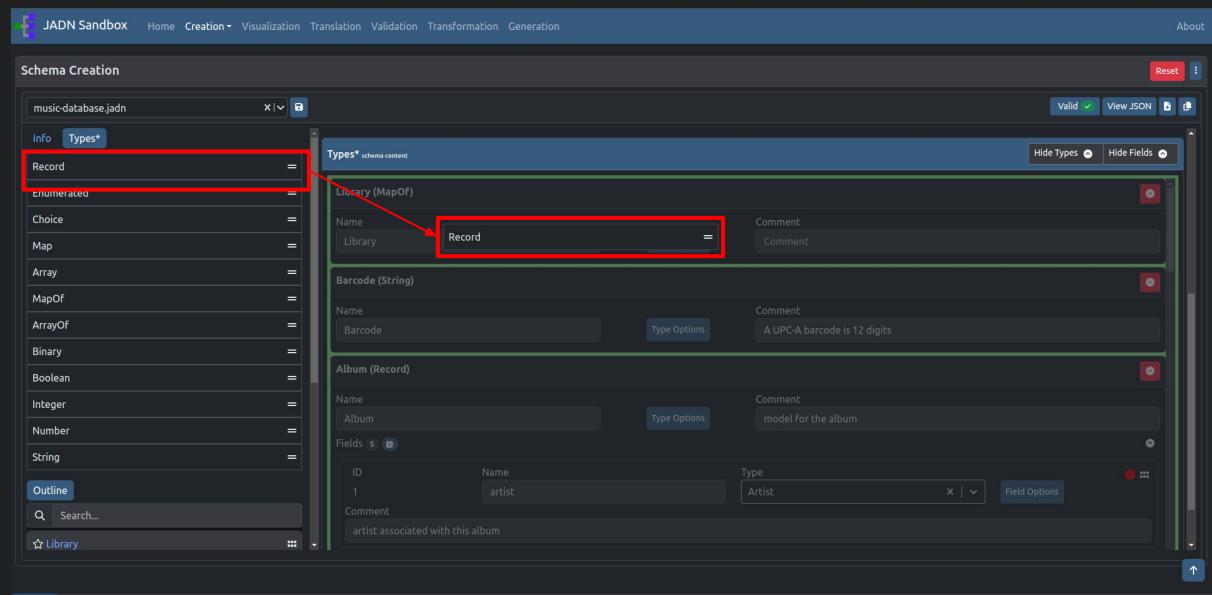


Add Data: Drag and Drop Style

Using the default drag and drop style creator, drag items from left to right.

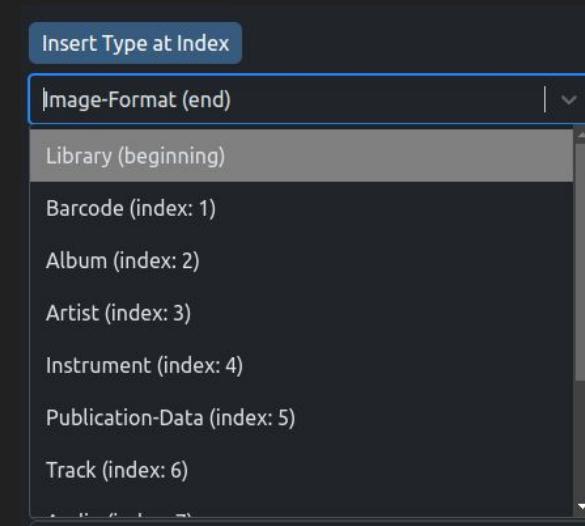
Drop the card when backgrounds are highlighted green.

When dragging, the background will change colors (in this case, to dark gray) to indicate where you can drop the card. It will then change to a green background when you are able to drop the card.

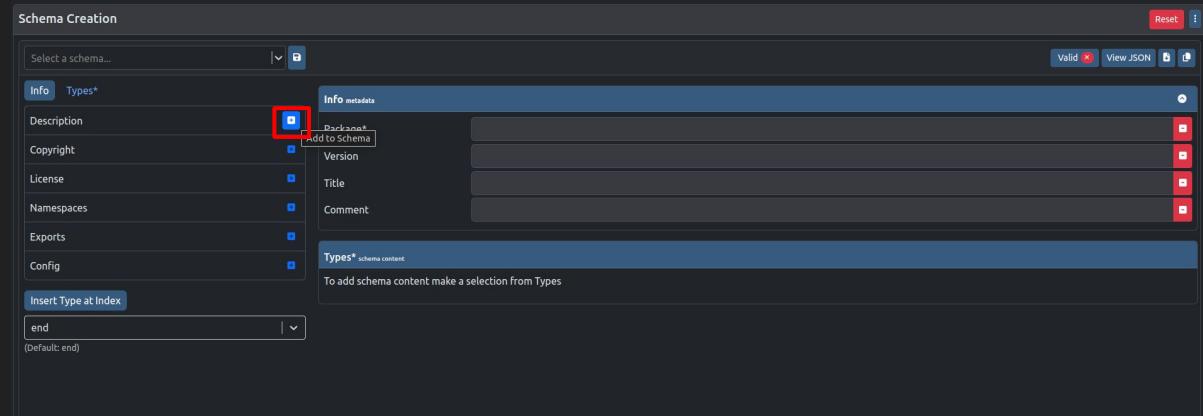


Add Data: Button Style

The ‘Insert Type at Index’ allows you to select where you would like to add your Type. Types are added to the end of the Schema by default.



Click to add Info or Type to the schema.

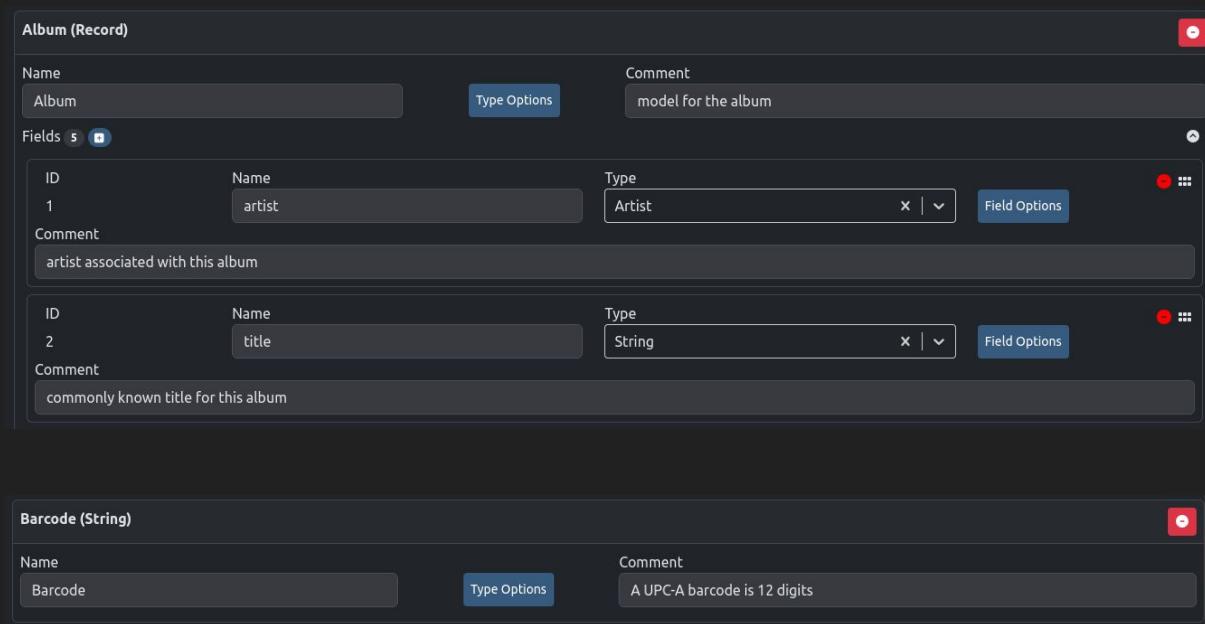


Edit Data: Add, change, or remove Types

The cards dropped on the right can be edited by clicking within input fields or removed .

Fields, if applicable, can be:

- Added,
- removed,
- hidden, and
- easily rearranged.



The screenshot shows a data editor interface with three cards:

- Album (Record)**:
 - Name: **Album** (with **Type Options** button)
 - Comment: **model for the album**
 - Fields (5):
 - ID: 1, Name: **artist**, Type: **Artist** (with **Field Options** button)
 - Comment: **artist associated with this album**
- Barcode (String)**:
 - Name: **Barcode** (with **Type Options** button)
 - Comment: **A UPC-A barcode is 12 digits**
- Unnamed Card**:
 - ID: 2, Name: **title**, Type: **String** (with **Field Options** button)
 - Comment: **commonly known title for this album**

Feature: Schema Types Outline

You can easily search or rearrange the Types within the Outline.



- ★ The cards within the outline can also be starred for future reference.

Clicking on the Type Name will take you to the card in the Types section of the schema on the right.

A screenshot of a software interface titled "Outline". It features a search bar at the top with the placeholder "Search...". Below the search bar is a list of schema types, each preceded by a star icon and a three-dot menu icon. The items in the list are: Library, Barcode, Album, Artist, Instrument, Publication-Data, Track, Audio, and Audio-Format. The "Album" item is highlighted with a blue background.

A screenshot of a software interface titled "Outline". It features a search bar at the top with the placeholder "Search...". Below the search bar is a list of schema types, each preceded by a star icon and a three-dot menu icon. The items in the list are: Library, Barcode, Album, Artist, Instrument, Publication-Data, Track, Audio, and Audio-Format. The "Album" item is highlighted with a blue background.

Functions

As you build your schema, you can view the JSON code by clicking on 'View JSON' .

This code can be:

- Downloaded to your local storage
- Copied to the clipboard
- Saved onto the file uploader for easy selection

Other features include:

- Checked for validation
- Reset to start over
- Scroll to top (appears when you scroll down on window)

```
1. {  
2.   "info": {  
3.     "title": "Music Library",  
4.     "package": "http://fake-audio.org/music-lib",  
5.     "version": "1.0",  
6.     "description": "This information model defines a library of audio tracks, organized by album.",  
7.     "license": "CC0-1.0",  
8.     "exports": ["Library", "Album", "Track"]  
9.   },  
10.  "types": [  
11.    {"Library": "MapOf", "/*Barcode": "String", "Album": "[1]", "", []},  
12.    {"Barcode": "String", "/*Value": "String", "A UPC/A barcode is 12 digits", []},  
13.    {"Album": "Record", "/*Label": "String", "The label for the album", []},  
14.    {"Artist": "Artist", "/*Label": "String", "Artist associated with this album"},  
15.    {"title": "String", "/*Label": "Commonly known title for this album"},  
16.    {"Publication-Data": "Object", "/*Label": "Metadata about publication"},  
17.    {"Cover-Art": "Image", "/*Label": "Small thumbnail track description"},  
18.    {"cover_art": "Cover-Art", [], "cover art image for this album"}  
19.  ],  
20.  {"Artist": "Record", [], "interesting information about the performers", [  
21.    {"name": "String", "/*Label": "Who is this person"},  
22.    {"instruments": "ArrayOf", "/*Instrument": "String", "and what do they play"},  
23.  ],  
24.  {"Instrument": "Enumerated", [], "collection of instruments (non-exhaustive)", [  
25.    {"vocals": ""},  
26.    {"guitar": ""},  
27.    {"bass": ""},  
28.    {"drums": ""},  
29.    {"keyboards": ""},  
30.    {"percussion": ""},  
31.    {"brass": ""},  
32.    {"woodwinds": ""},  
33.    {"harmonica": ""}],  
34.  },  
35.  {"Publication-Data": "Record", [], "who and when of publication", [  
36.    {"label": "String", "/*Label": "Name of record label"},  
37.    {"rel_date": "String", "/*Label": "Date", "when did they let this drop"},  
38.  ]},  
39.  {"Track": "Record", [], "information about the individual audio tracks", [  
40.    {"t_number": "Number", "/*Label": "track sequence number"}  
41.  ]}  
42.}
```

Click 'View Form' [View Form](#) to return to the Schema Creator.

The following slide shows you how to begin interacting
with the page.

This applies to the following pages:

Schema Visualization

Schema Translation

Data Creation

Data Validation

Example Data Generation

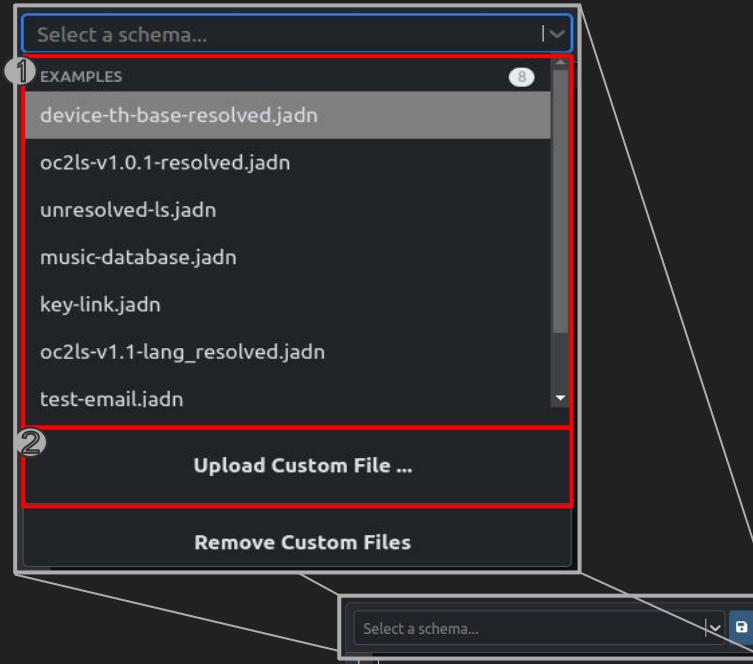
Start with the File Loader :

Valid ✓

- ① Select a valid example schema
or
- ② Upload a valid JADN Schema
or
- ③ Type/ Paste in a valid JADN Schema in the code editor

You can also:

-  - format the Schema
-  - copy to clipboard
-  [Save the Schema to the file loader](#)



③

Valid ✘  

Schema Visualization

View JADN schema in another language format:
JIDL¹, HTML, Markdown, PlantUML, GraphViz

1. [JADN Interface Definition Language \(IDL\)](#) is a textual representation of JADN type definitions

Home

 Creation Create and edit JADN schemas using forms, view JADN schemas in JSON format. Create schema compliant data instances (documents, messages).	 Schema Visualization Convert a JADN Schema into different visual representations.	 Schema Translation Translate a JADN Schema to another Schema format. Translate a JSON Schema to a JADN Schema.
Schemas Data Instances	GraphViz HTML JIDL MarkDown PlantUML	JSON Relax (XML) XSD
 Data Validation Validate data instances in various data language formats against a JADN Schema.	 Schema Transformation Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc).	 Example Data Generation Generate various example data instances based off of a Schema.
CBOR JSON Relax (XML)		

Go to: Visualization

To Start : [click here](#)

Note: Only a JADN Schema can be uploaded.

After selecting a valid schema, select the desired languages you would like to convert the schema to.

Options on this page are for visual representation of JADN data. For Schema Translations (like to JSON or XML) [click here](#)

Visualize to...(select at least one) | 

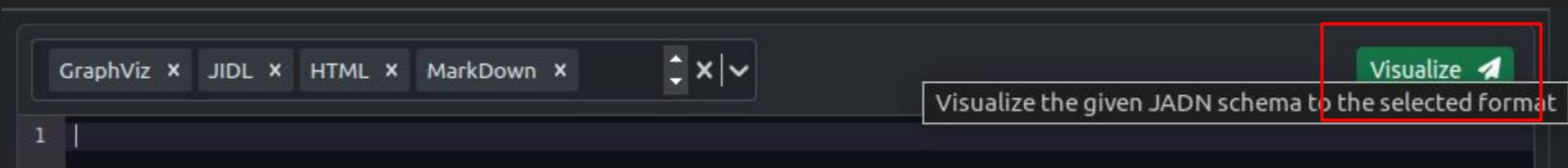


- GraphViz
- HTML
- JIDL
- MarkDown
- PlantUML

Visualize

Once you have selected the languages, the green paper airplane button should light up allowing you to proceed to visualize the schema to the selected messages.

Click the Visualize button to proceed.



Results: Selecting one language

The result will show you the code in the selected language.

The following features allow you to interact with the visualization as you need.



- Split view (currently shown) : To view the code and visualization simultaneously



- Pop out : See the visualization in a new tab window



- Download visualization as PDF



- Download visualization as an image



- Download code



- Copy code to Clipboard

HTML x ✖ | ↻

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Music Library</title>
6     <style type="text/css"> /* Theme Colors - CC3322 as base using Adobe Kuler */
7   /* PDF Styles */
8   @page {
9     size: letter portrait;
10    @frame content_frame {
11      top: 16pt;
12      left: 16pt;
13    }
14  }
15  @media print {
16    #schema {
17      max-width: auto;
18    }
19  }
20  /* Standard Styles */
21  div>
```

Schema

```
package: "http://fake-audio.org/music-lib"
version: "1.0"
  title: "Music Library"
description: "This information model defines a library of audio tracks, organized by album"
  license: "CC0-1.0"
  exports: ["Library", "Album", "Track"]
  config: {}
```

Compound Types

Library

Library (MapOf(Barcode, Album)[1..*])

Album

model for the album

Results: Selecting multiple languages

You also have the ability to visualize multiple languages at once.

You can view the code for one or multiple languages at once by clicking on the header of each section.

Each language has its own applicable features as seen in each section.

The screenshot shows a web-based interface for visualizing code across multiple languages. At the top, there are tabs for GraphViz, HTML, JIDL, and MarkDown, each with a close button. Below the tabs, there are four sections: GraphViz, HTML, JIDL, and MarkDown, each with its own set of icons for file operations and sharing. The HTML section is currently active, indicated by a red border around its tab and a blue border around its content area. The content area displays the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Music Library</title>
6     <style type="text/css"> /* Theme Colors - CC3322 as base using Adobe Kuler */
7   /* PDF Styles */
8   @page {
9     size: letter portrait;
10    @frame content_frame {
11      top: 16pt;
12      left: 16pt;
13    }
14  }
15  @media print {
16    #schema {
17      max-width: auto;
18    }
19  }
20  /* Standard Styles */
21 div#body {
```

Below the HTML section, the JIDL, MarkDown, and PlantUML sections are visible, each with their own set of icons.

Schema Translation

Convert JADN Schema to JSON , Relax XML, or XSD.
Convert JSON Schema to JADN.



Home

**Creation**

Create and edit JADN schemas using forms, view JADN schemas in JSON format.

Create schema compliant data instances (documents, messages).

**Schema Visualization**

Convert a JADN Schema into different visual representations.

**Schema Translation**

Translate a JADN Schema to another Schema format.

Translate a JSON Schema to a JADN Schema.

Schemas Data Instances

GraphViz HTML JIDL MarkDown PlantUML

JSON Relax (XML) XSD

**Data Validation**

Validate data instances in various data language formats against a JADN Schema.

CBOR JSON Relax (XML)

**Schema Transformation**

Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc).

**Example Data Generation**

Generate various example data instances based off of a Schema.

Go to: Translation

To Start : [click here](#)

Note: A JSON or JADN Schema can be uploaded.

Similar to Schema Visualization,
Schema Translation allows you to
translate your schema into your
desired language.

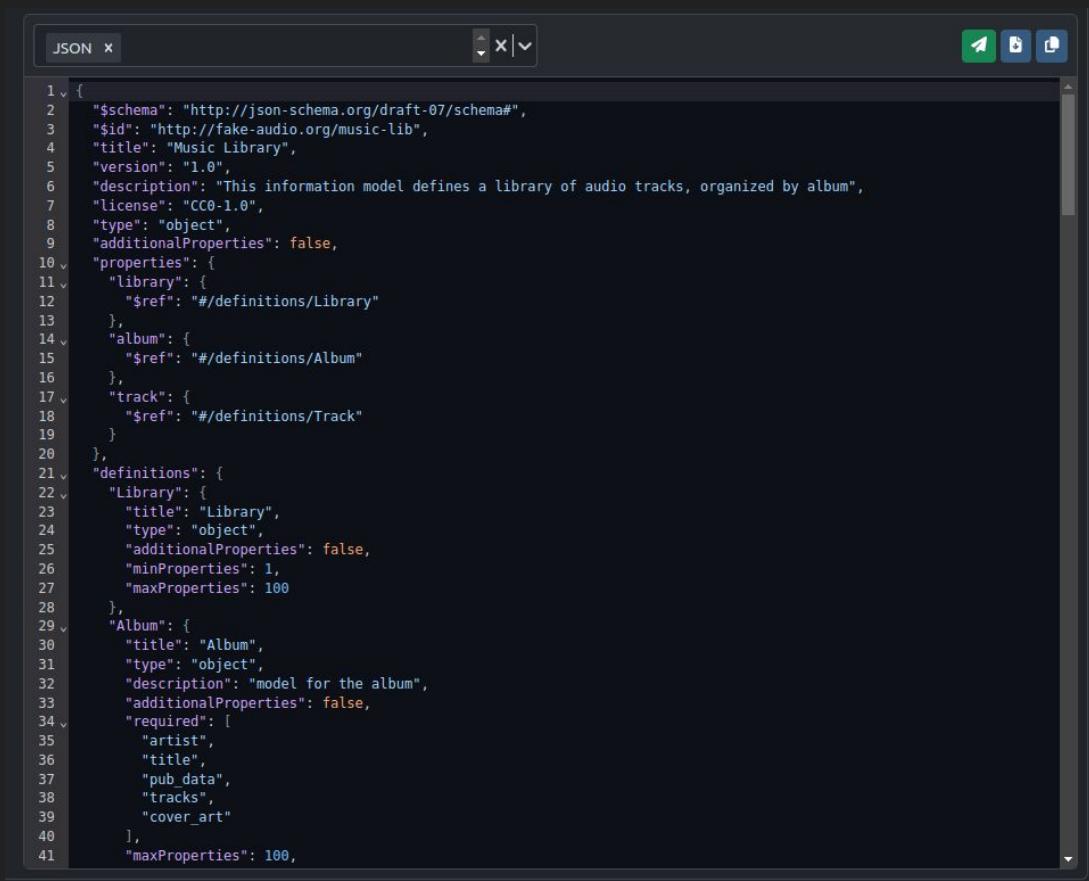
When selecting a schema, make sure
to select the type of schema you want
to translate from: JADN or JSON.



The screenshot shows the Schema Translation interface. On the left, there is a code editor window titled "music-database.jadn" containing JADN schema code. On the right, there is another code editor window titled "JSON x" containing JSON schema code. At the top center, there is a dropdown menu with the option "jadn" highlighted with a red box. Below the dropdown are several icons: a file icon, a "Valid" button, and a refresh/cancel icon.

```
music-database.jadn
1 v {
2 v   "info": {
3 v     "title": "Music Library",
4 v     "package": "http://fake-audio.org/music-lib".
1 v {
2 v   "$schema": "http://json-schema.org/draft-07/schema#",
3 v   "$id": "http://fake-audio.org/music-lib",
4 v   "title": "Music Library",
5 v   "version": "1.0",
6 v   "description": "This information model defines a library of audio tracks, organized by album",
7 v   "license": "CC0-1.0",
8 v   "type": "object",
9 v   "additionalProperties": false,
10 v   "properties": {
11 v     "library": {
12 v       "$ref": "#/definitions/Library"
13 v     },
14 v     "album": {
15 v       "$ref": "#/definitions/Album"
16 v     },
17 v     "track": {
18 v       "$ref": "#/definitions/Track"
19 v     }
20 v   },
21 v   "definitions": {
22 v     "Library": {
23 v       "title": "Library",
24 v       "type": "object",
25 v       "additionalProperties": false,
26 v       "minProperties": 1,
27 v       "maxProperties": 100
28 v     },
29 v     "Album": {
30 v       "title": "Album",
31 v       "type": "object",
32 v       "description": "model for the album",
33 v       "additionalProperties": false,
34 v       "required": [
35 v         "artist",
36 v         "title",
37 v         "pub_data",
38 v         "tracks",
39 v         "cover_art"
40 v       ],
41 v       "maxProperties": 100,

```



The screenshot shows the Schema Translation interface. On the right, there is a code editor window titled "JSON x" containing JSON schema code. The code is identical to the one in the "music-database.jadn" editor. At the top right of the "JSON x" window, there are three icons: a magnifying glass, a save icon, and a refresh/cancel icon.

```
JSON x
1 v {
2 v   "$schema": "http://json-schema.org/draft-07/schema#",
3 v   "$id": "http://fake-audio.org/music-lib",
4 v   "title": "Music Library",
5 v   "version": "1.0",
6 v   "description": "This information model defines a library of audio tracks, organized by album",
7 v   "license": "CC0-1.0",
8 v   "type": "object",
9 v   "additionalProperties": false,
10 v   "properties": {
11 v     "library": {
12 v       "$ref": "#/definitions/Library"
13 v     },
14 v     "album": {
15 v       "$ref": "#/definitions/Album"
16 v     },
17 v     "track": {
18 v       "$ref": "#/definitions/Track"
19 v     }
20 v   },
21 v   "definitions": {
22 v     "Library": {
23 v       "title": "Library",
24 v       "type": "object",
25 v       "additionalProperties": false,
26 v       "minProperties": 1,
27 v       "maxProperties": 100
28 v     },
29 v     "Album": {
30 v       "title": "Album",
31 v       "type": "object",
32 v       "description": "model for the album",
33 v       "additionalProperties": false,
34 v       "required": [
35 v         "artist",
36 v         "title",
37 v         "pub_data",
38 v         "tracks",
39 v         "cover_art"
40 v       ],
41 v       "maxProperties": 100,
```

Result: Translated Schema to one language

Result: Translated Schema to multiple languages

You can translate from JADN to JSON and XML (XSD or Relax) or from JSON to JADN.

Schema Translation

Reset

The screenshot displays a schema translation tool with three main tabs: JADN, XSD, and JSON. The JADN tab on the left shows a JSON-like code editor for a 'music-database.jadn' schema. The XSD tab in the center shows the corresponding XML schema definition. The JSON tab on the right shows the schema translated into JSON format. The interface includes a toolbar at the top with various icons for file operations and validation.

JADN Tab Content:

```
1  {
2    "info": {
3      "title": "Music Library",
4      "package": "http://fake-audio.org/music-lib",
5      "version": "1.0",
6      "description": "This information model defines a library of audio tracks, organized by album",
7      "license": "CC0-1.0",
8      "exports": ["Library", "Album", "Track"]
9    },
10   "types": [
11     ["Library", "MapOf", ["+Barcode", "*Album", "[1]", "", []],
12     ["Barcode", "String", ["*\\d{12}"], "A UPC-A barcode is 12 digits", []],
13     ["Album", "Record", [], "model for the album", [
14       [1, "artist", "Artist", [], "artist associated with this album"],
15       [2, "title", "String", [], "commonly known title for this album"],
16       [3, "pub_data", "Publication-Data", [], "metadata about album publication"],
17       [4, "tracks", "ArrayOf", ["*Track", "J0"], "individual track descriptions"],
18       [5, "cover_art", "Cover-Art", [], "cover art image for this album"]
19     ],
20     ["Artist", "Record", [], "interesting information about the performers", [
21       [1, "artist_name", "String", [], "who is this person"],
22       [2, "instruments", "ArrayOf", ["*Instrument", "J0"], "and what do they play"]
23     ],
24     ["Instrument", "Enumerated", [], "collection of instruments (non-exhaustive)", [
25       [1, "vocals", ""],
26       [2, "guitar", ""],
27       [3, "bass", ""],
28       [4, "drums", ""],
29       [5, "keyboards", ""],
30       [6, "percussion", ""],
31       [7, "brass", ""],
32       [8, "woodwinds", ""],
33       [9, "harmonica", ""]
34     ],
35     ["Publication-Data", "Record", [], "who and when of publication", [
36       [1, "label", "String", [], "name of record label"],
37       [2, "rel_date", "String", ["*/date"], "and when did they let this drop"]
38     ],
39     ["Track", "Record", [], "information about the individual audio tracks", [
40       [1, "t_number", "Number", [], "track sequence number"],
41       [2, "title", "String", [], "track title"],
42     ]
43   ]
44 }
```

XSD Tab Content:

```
1 <xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema" xmlns:jadn="jadn_base_types">
2   <xss:import schemaLocation="../data/xsd/jadn_base_types.xsd" namespace="jadn_base_types" />
3   <xss:complexType name="Library">
4     <xss:sequence>
5       <xss:element id="library_map" name="library_map" minOccurs="1" maxOccurs="unbounded">
6         <xss:complexType>
7           <xss:sequence>
8             <xss:element id="library_map_barcode" name="Barcode" />
9           </xss:sequence>
10          </xss:complexType>
11        </xss:element>
12      </xss:sequence>
13    </xss:complexType>
14  </xss:complexType>
15  <xss:simpleType name="Barcode">
16    <xss:annotation>
17      <xss:documentation>A UPC-A barcode is 12 digits</xss:documentation>
18    </xss:annotation>
19    <xss:restriction base="jadn:String">
20      <xss:pattern value="\d{12}" />
21    </xss:restriction>
22  </xss:simpleType>
23
```

JSON Tab Content:

```
1  {
2    "$schema": "http://json-schema.org/draft-07/schema#",
3    "$id": "http://fake-audio.org/music-lib",
4    "title": "Music Library",
5    "version": "1.0",
6    "description": "This information model defines a library of audio tracks, organized by album",
7    "license": "CC0-1.0",
8    "type": "object",
9    "additionalProperties": false,
10   "properties": {
11     "library": {
12       "$ref": "#/definitions/Library"
13     }
14   }
15 }
```

Data Creation

Create valid data instances from a JADN Schema

The screenshot shows the JADN Sandbox application interface. At the top, there is a navigation bar with links: Home, Creation, Visualization, Translation, Validation, Transformation, Generation, and About. The 'Creation' link is highlighted with a red box. A dropdown menu for 'Creation' is open, showing two options: 'Schema Creation' and 'Data Creation', also both highlighted with red boxes.

Home

Creation

Schema Creation

Data Creation

Schema Visualization

Convert a JADN Schema into different visual representations.

Schema Translation

Translate a JADN Schema to another Schema format.

Schemas Data Instances

GraphViz HTML JIDL MarkDown PlantUML

JSON Relax (XML) XSD

Data Validation

Validate data instances in various data language formats against a JADN Schema.

CBOR JSON Relax (XML)

Schema Transformation

Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc).

Example Data Generation

Generate various example data instances based off of a Schema.

localhost:8082/create/message v0.10.0_1705587211962

Go to: Creation
When the drop down menu appears, select Data Creation

To Start : [click here](#)

Note: Only a JADN Schema can be uploaded.

After selecting a schema, you should be able to select a data type.

Note: This drop down menu is based on the Schema exports.

Once you have selected a data type, a form will appear to help guide you in building data that is valid against the selected schema.

The screenshot shows a schema editor interface with a dark theme. At the top, there is a header bar with the title 'Library' and several icons: 'View JSON', a save icon, and a refresh/cancel icon. Below the header, a dropdown menu is open, showing three options: 'Library', 'Album', and 'Track'. The 'Library' option is highlighted with a blue selection bar. Below the dropdown, there is a form area with four fields:

- barcode***: A UPC-A barcode is 12 digits.
Input field: \d{12}
- album***: model for the album.
Input field: (empty)
- artist***: artist associated with this album.
Input field: (empty)
- artist_name***: who is this person
Input field: (empty)

View JSON / Creator

After filling out the form, you can click 'View JSON' to see the data you have created in JSON.

[View JSON](#)

Click 'View Creator' to go back to the form.

[View Creator](#)

The JSON can also be:



- Downloaded to your local storage



- Copied to the clipboard

```
1 v {
2 v   "Library": {
3 v     "123456789123": {
4 v       "artist": {
5 v         "artist_name": "the Beatles",
6 v         "instruments": ["guitar", "guitar"]
7 v       },
8 v       "title": "Hey Jude"
9 v     }
10 v   }
11 }
```

Data Validation

Validate data instances in JSON, XML, or CBOR against a
JADN Schema

Home

**Creation**

Create and edit JADN schemas using forms, view JADN schemas in JSON format.

Create schema compliant data instances (documents, messages).

**Schema Visualization**

Convert a JADN Schema into different visual representations.

**Schema Translation**

Translate a JADN Schema to another Schema format.

Translate a JSON Schema to a JADN Schema.

Schemas Data Instances

GraphViz HTML JIDL MarkDown PlantUML

JSON Relax (XML) XSD

**Data Validation**

Validate data instances in various data language formats against a JADN Schema.

CBOR JSON Relax (XML)

**Schema Transformation**

Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc).

**Example Data Generation**

Generate various example data instances based off of a Schema.

Go to: Validation

To Start : [click here](#)

Note: Only a JADN Schema can be uploaded.

① Select data to validate.

② Make sure the correct data format is selected.

③ Make sure the correct data type being validated is selected.

In this example, the data is a json file being validated against the Library type from the music-database.jadn

The screenshot shows a user interface for validating JADN data. At the top, there are three dropdown menus labeled 1, 2, and 3, each with a red box around it. Menu 1 contains 'query_pairs_a.json'. Menu 2 contains 'json'. Menu 3 contains 'Library'. To the right of these menus is a green 'Validate' button with a checkmark icon, and below it are three small blue icons. Below the menus is a code editor window displaying the following JSON code:

```
1 {  
2   "action": "query",  
3   "target": {  
4     "features": [  
5       "pairs"  
6     ]  
7   }  
8 }
```

Select data to validate

Starting by uploading data:

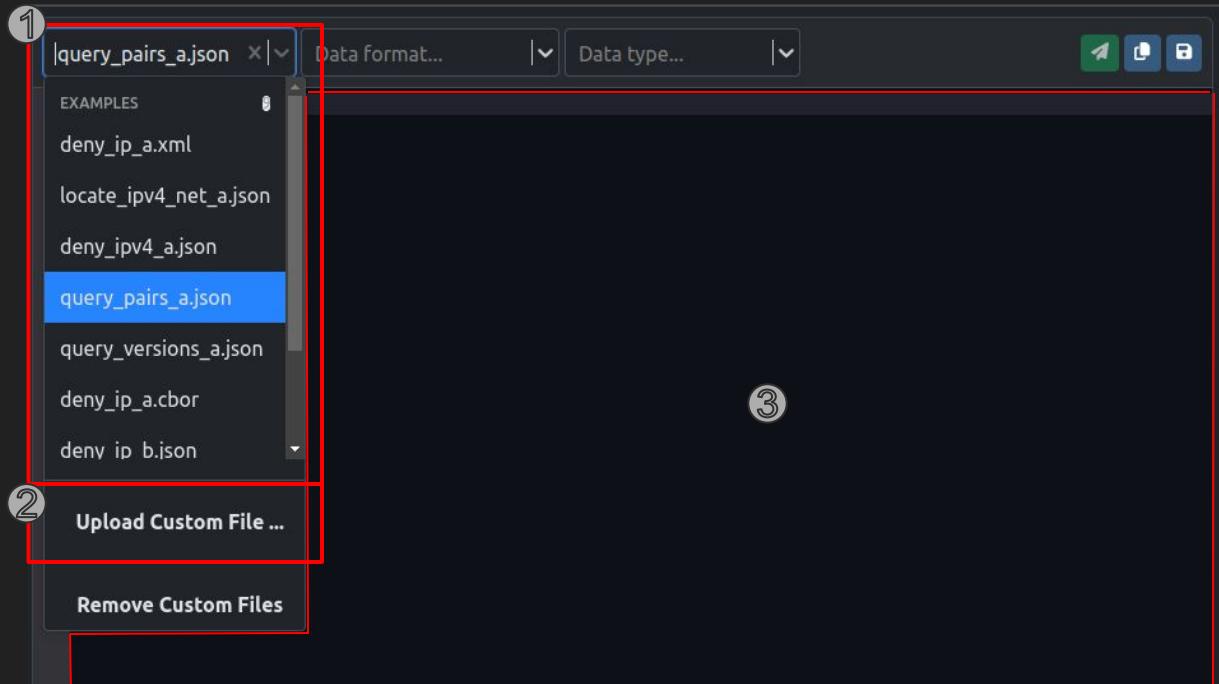
- ① Select example data

or

- ② Upload a custom file

or

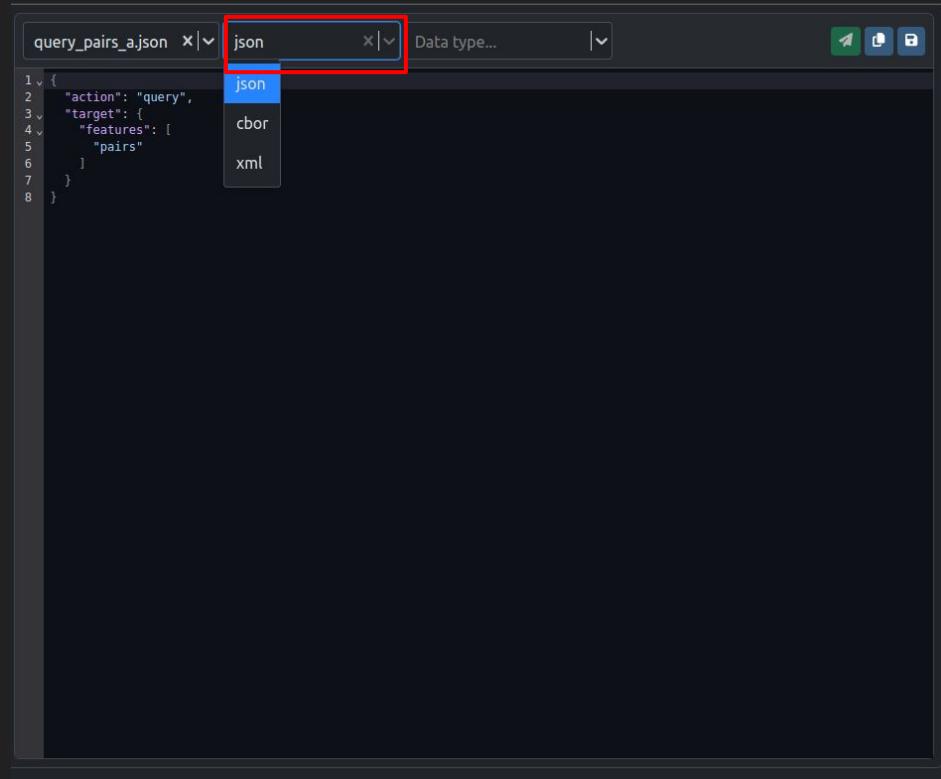
- ③ type/paste directly into code editor.



Make sure to select the correct data format

The JADN Sandbox can currently validate JSON, CBOR, and XML data against a JADN Schema.

If using a data file, the data format drop down may be pre-selected.



Make sure the correct data type is selected

The Data types available to validate against are based off of the exports from the JADN Schema file.

In this example, notice that the uploaded schema music-database.jadn has exports that correspond to the valid datatypes:

Library, Album, Track

```
music-database.jadn
1 v {
2 v   "info": {
3 v     "title": "Music Library",
4 v     "package": "http://fake-audio.org/music-lib",
5 v     "version": "1.0",
6 v     "description": "This information model defines a library of audio tracks, organized by album",
7 v     "license": "CC0-1.0",
8 v     "exports": ["Library", "Album", "Track"]
9 v   },
10 v }
```

```
locate_ipv4_net_a.json
1 v {
2 v   "action": "locate",
3 v   "target": {
4 v     "ipv4_net": "MTI3LjAuMC4x/30"
5 v   }
6 v }
```

Library

Album

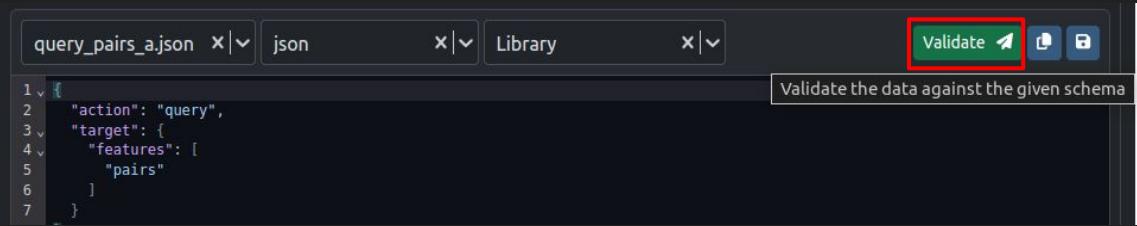
Track

Validate

Once you have selected:

- A Schema,
- Data,
- Data format, and
- Data type

Click the bright, green “Validate” button.



The screenshot shows a software interface for validating JSON data. At the top, there are tabs for "query_pairs_a.json" (selected), "json", and "Library". On the right side, there is a green "Validate" button with a white checkmark icon, which is highlighted with a red rectangular box. Below the tabs, there is a status bar with the text "Validate the data against the given schema". The main area contains a code editor with the following JSON code:

```
1 v
2   "action": "query",
3   "target": {
4     "features": [
5       "pairs"
6     ]
7   }
```

Invalid Results

An invalid data file will return errors to help fix the data.

The screenshot shows a code editor window with a dark theme. The title bar includes tabs for 'query_pairs_a.json', 'json', and 'Library'. On the right side of the title bar are icons for saving, opening, and closing. The main area displays the following JSON code:

```
1 {  
2   "action": "query",  
3   "target": {  
4     "features": [  
5       "pairs"  
6     ]  
7   }  
8 }
```

Below the code, a series of five red error notifications are listed, each with an exclamation icon and a message:

- field required at artist
- object of type
'builtin_function_or_method' has
no len() at title
- field required at pub_data
- field required at tracks
- field required at cover_art

A 'Clear All' button is located at the bottom right of the error list.

Valid Results

A valid data file will return a green success notification.

The screenshot shows a JSON validation interface with the following details:

- File type: json
- Validation status: Validation success (indicated by a green checkmark icon)
- Content: A JSON object representing an album. It includes fields for artist, title, publication data, tracks (with specific track details like title and length), and cover art.

```
1 v {
2 v   "artist": {
3 v     "artist_name": "the Beatles",
4 v     "instruments": ["vocals"]
5 v   },
6 v   "title": "Hey Jude",
7 v   "pub_data": {
8 v     "label": "Capital Records",
9 v     "rel_date": "1979-05-11"
10 v   },
11 v   "tracks": [
12 v     {
13 v       "t_number": 1,
14 v       "title": "Can't Buy Me Love",
15 v       "length": "00:02:19",
16 v       "featured": [],
17 v       "audio": {
18 v         "a_format": "MP3",
19 v         "a_content": "dGhlIEJlYXRsZXM="
20 v       }
21     }
22   ],
23 v   "cover_art": {
24 v     "i_format": "JPG",
25 v     "i_content": "dGhlIEJlYXRsZXM="
26 v   }
27 }
```

Schema Transformation

Strip comments from Schemas
or Resolve a schema with references



Home

**Creation**

Create and edit JADN schemas using forms, view JADN schemas in JSON format.

Create schema compliant data instances (documents, messages).

**Schema Visualization**

Convert a JADN Schema into different visual representations.

**Schema Translation**

Translate a JADN Schema to another Schema format.

Translate a JSON Schema to a JADN Schema.

Schemas Data Instances

GraphViz HTML JIDL MarkDown PlantUML

JSON Relax (XML) XSD

**Data Validation**

Validate data instances in various data language formats against a JADN Schema.

CBOR JSON Relax (XML)

**Schema Transformation**

Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc).

**Example Data Generation**

Generate various example data instances based off of a Schema.

Theme ▾

v0.10.0_1705587211962

Go to: Transformation

To Start :

Select at least one schema to upload.

Notice that there is no code editor. Therefore, schemas must be uploaded.

Here you can see 2 schemas selected to start:

- unresolved-ls.jadn
- oc2ls-v1.1-lang_resolved.jadn

The screenshot shows a schema editor interface with two tabs open:

- unresolved-ls.jadn**: This tab displays the JSON content of the 'unresolved-ls.jadn' schema. It includes sections for info, package, version, title, description, namespaces, exports, and types. The types section defines several schema elements like 'Language-Spec-Types', 'Record', 'Artifact', 'Command-ID', and 'Device'.
- oc2ls-v1.1-lang_resolved.jadn**: This tab displays the JSON content of the 'oc2ls-v1.1-lang_resolved.jadn' schema. It includes sections for info, package, title, description, exports, and types. The types section defines the 'OpenC2-Command' and 'OpenC2-Response' schema elements.

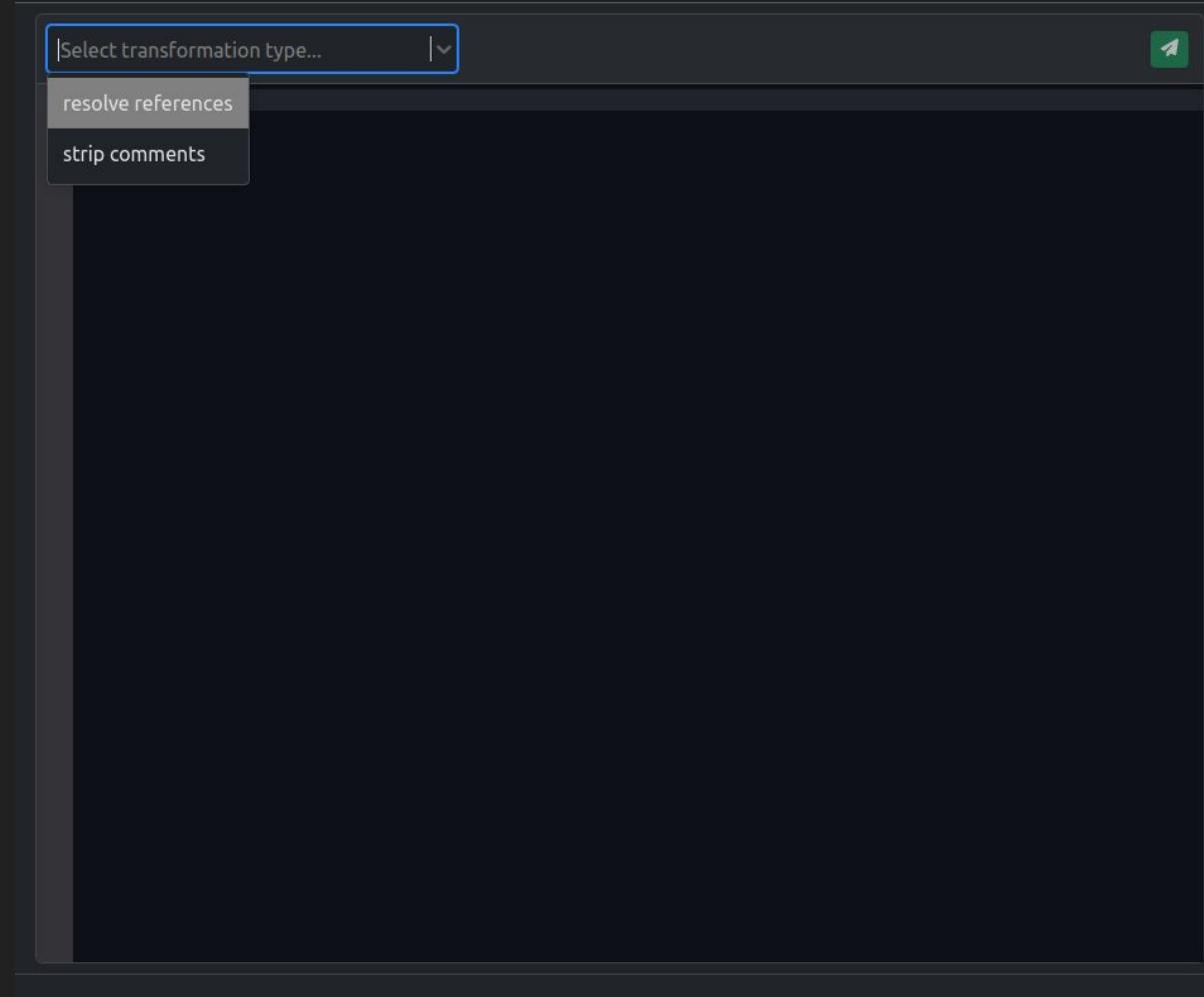
After selecting the desired schemas, you can select your desired transformation:

Resolve references

- Create a single JADN schema from an unresolved schema and its imports

Strip comments

- Remove comments from a JADN schema to reduce size or visual clutter



Resolve References ¹

For resolving references, you will need to select a base file. This file is the file you would like to be resolved.

The other uploaded files will be used to help resolve the chosen base file.

In this example, the base file is :
unresolved-ls.jadn

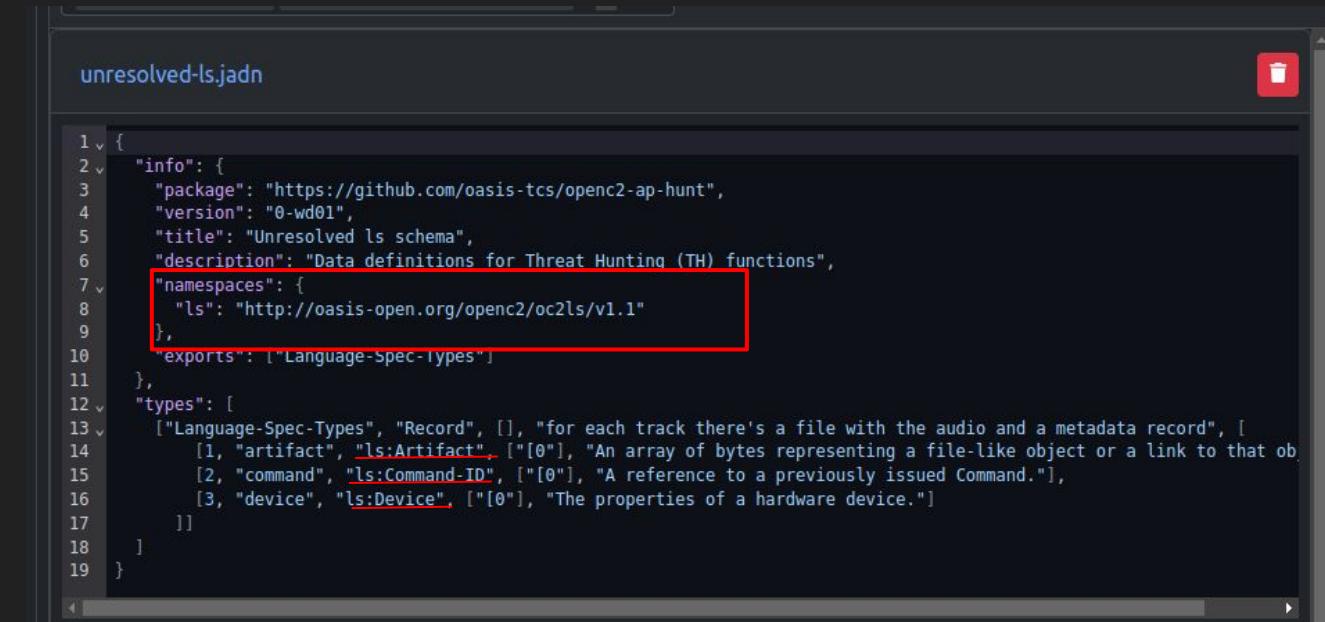
1. A resolved Schema is a schema without references.

The screenshot shows the OpenSCAP interface with the 'Resolve References' dialog open. The dialog has a red border and contains the text 'resolve references' and 'x | v'. To the right of the dialog is a dropdown menu labeled 'Select base file...' with two options: 'unresolved-ls.jadn' and 'oc2ls-v1.1-lang_resolved.jadn'. Below the dialog is a code editor window titled 'unresolved-ls.jadn' containing the following JSON-like schema definition:

```
1 v {  
2 v   "info": {  
3 v     "package": "https://github.com/oasis-tcs/openc2-ap-hunt",  
4 v     "version": "0-wd01",  
5 v     "title": "Unresolved ls schema",  
6 v     "description": "Data definitions for Threat Hunting (TH) functions",  
7 v     "namespaces": {  
8 v       "ls": "http://oasis-open.org/openc2/oc2ls/v1.1"  
9 v     },  
10 v    "exports": ["Language-Spec-Types"]  
11 v  },  
12 v  "types": [  
13 v    ["Language-Spec-Types", "Record", [], "for each track there's a file with the audio and a metadata record", [  
14 v      [1, "artifact", "ls:Artifact", "[0]", "An array of bytes representing a file-like object or a link to that ob  
15 v      [2, "command", "ls:Command-ID", "[0]", "A reference to a previously issued Command."],  
16 v      [3, "device", "ls:Device", "[0]", "The properties of a hardware device."]  
17 v    ]]  
18 v  ]  
19 v}
```

What is an unresolved Schema?

A Schema is unresolved if it has namespaces (references to types defined in other schemas).



```
unresolved-ls.jadn
1 v {
2 v   "info": {
3 v     "package": "https://github.com/oasis-tcs/openc2-ap-hunt",
4 v     "version": "0-wd01",
5 v     "title": "Unresolved ls schema",
6 v     "description": "Data definitions for Threat Hunting (TH) functions",
7 v     "namespaces": {
8 v       "ls": "http://oasis-open.org/openc2/oc2ls/v1.1"
9 v     },
10 v     "exports": ["Language-Spec-Types"]
11 },
12 v   "types": [
13 v     ["Language-Spec-Types", "Record", [], "for each track there's a file with the audio and a metadata record", [
14 v       [1, "artifact", "ls:Artifact", "[[0]", "An array of bytes representing a file-like object or a link to that ob
15 v       [2, "command", "ls:Command-ID", "[[0]", "A reference to a previously issued Command."]
16 v       [3, "device", "ls:Device", "[[0]", "The properties of a hardware device."]
17 v     ]]
18   ]
19 }
```

Click Transform to resolve the base file.



Result: A resolved Schema

Notice there is no namespace and that the types from the unresolved schema can be referenced within the resolved schema.

Ls: Artifact → Artifact

Ls: Command-ID → Command-ID

Ls: Device → Device

resolve references X | v unresolved-ls.jadn X | v

```
1 v {
2 v   "info": {
3 v     "package": "https://github.com/oasis-tcs/openc2-ap-hunt",
4 v     "version": "0-wd01",
5 v     "title": "Unresolved ls schema",
6 v     "description": "Data definitions for Threat Hunting (TH) functions",
7 v     "exports": ["Language-Spec-Types"]
8 },
9
10 v   "types": [
11 v     ["Language-Spec-Types", "Record", [], "for each track there's a file with the audio and a metadata record", [
12 v       [1, "artifact", "Artifact", ["[0]", "An array of bytes representing a file-like object or a link to that object."],
13 v       [2, "command", "Command-ID", ["[0]", "A reference to a previously issued Command."],
14 v       [3, "device", "Device", ["[0]", "The properties of a hardware device."]
15 ],
16
17 v     ["Artifact", "Record", [{"1": "", [
18 v       [1, "mime_type", "String", ["[0]", "Permitted values specified in the IANA Media Types registry, [RFC6838]"],
19 v       [2, "payload", "Payload", ["[0]", "Choice of literal content or URL"]],
20 v       [3, "hashes", "Hashes", ["[0]", "Hashes of the payload content"]]
21 ],
22
23 v     ["Device", "Map", [{"1": "", [
24 v       [1, "hostname", "Hostname", ["[0]", "A hostname that can be used to connect to this device over a network"],
25 v       [2, "idn_hostname", "IDN-Hostname", ["[0]", "An internationalized hostname that can be used to connect to this devi
26 v       [3, "device_id", "String", ["[0]", "An identifier that refers to this device within an inventory or management syst
27 ],
28
29 v     ["URI", "String", ["/uri"], "Uniform Resource Identifier, [RFC3986]", []],
30
31 v     ["Hashes", "Map", [{"1": "Cryptographic hash values", [
32 v       [1, "md5", "Binary", ["x", "[16]", "[0]", "MD5 hash as defined in [RFC1321]"],
33 v       [2, "sha1", "Binary", ["x", "[16]", "[0]", "SHA1 hash as defined in [RFC6234]"]
34 ]]]]
```

Result: Strip Comments

When stripping comments, results will return all schemas without comments.

The screenshot shows a code editor interface with two tabs. The top bar has a search field containing "strip comments" with a red box around it. The tabs are labeled "unresolved-ls" and "oc2ls-v1.1-lang_resolved".

unresolved-ls

```
1 v {
2 v   "info": {
3     "package": "https://github.com/oasis-tcs/openc2-ap-hunt",
4     "version": "0-wd01",
5     "title": "Unresolved ls schema",
6     "description": "Data definitions for Threat Hunting (TH) functions",
7     "namespaces": {
8       "ls": "http://oasis-open.org/openc2/oc2ls/v1.1"
9     },
10    "exports": ["Language-Spec-Types"]
11  },
12  "types": [
13    ["Language-Spec-Types", "Record", [], "", [
14      [1, "artifact", "ls_Artifact", "[[0"]], ""],
15      [2, "command", "ls_Command_ID", "[["0"]], ""],
16      [3, "device", "ls_Device", "[[0"]], ""
17    ]]
18  ]
19 }
```

oc2ls-v1.1-lang_resolved

```
1 v {
2 v   "info": {
3     "package": "http://oasis-open.org/openc2/oc2ls/v1.1",
4     "title": "OpenC2 Language Profile",
5     "description": "Language Profile from the OpenC2 Language Specification version 1.1",
6     "exports": ["OpenC2-Command", "OpenC2-Response"]
7   },
8   "types": [
9     ["OpenC2-Command", "Record", [], "", [
10       [1, "action", "Action", [], ""],
11       [2, "target", "Target", [], ""],
12       [3, "args", "Args", "[[0"]], ""
13     ]]
14   ]
15 }
```

Example Data Generation

Automatically create valid data instances given the JADN Schema



Home

**Creation**

Create and edit JADN schemas using forms, view JADN schemas in JSON format.

Create schema compliant data instances (documents, messages).

**Schema Visualization**

Convert a JADN Schema into different visual representations.

**Schema Translation**

Translate a JADN Schema to another Schema format.

Translate a JSON Schema to a JADN Schema.

Schemas Data Instances

GraphViz HTML JIDL MarkDown PlantUML

JSON Relax (XML) XSD

**Data Validation**

Validate data instances in various data language formats against a JADN Schema.

CBOR JSON Relax (XML)

**Schema Transformation**

Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc).

**Example Data Generation**

Generate various example data instances based off of a Schema.

Go to: Generation

To Start : [click here](#)

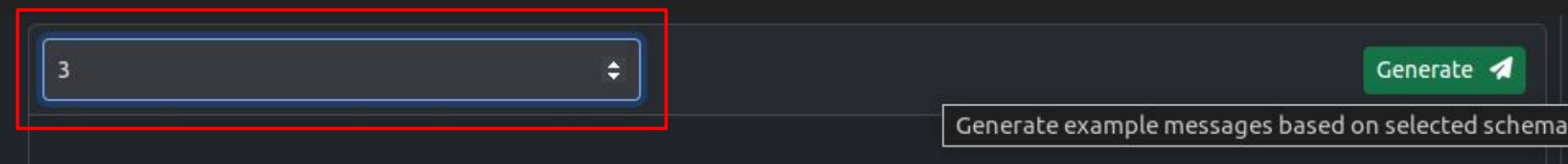
Note: Only a JADN Schema can be uploaded.

Enter the desired number of examples you would like generated. This number should be between 1 - 10.

(In this example, we have selected 3.)

After you input a number, click the green “Generate” button.

Note: The button will only be clickable if you have a valid JADN schema and the number of examples desired.



Warning: Example Generation may fail due to large Schema data. As a result, the application may lock itself. In this case, please restart docker container.

Results

If successfully generated, you will be able to view and hide the example data generated.

For each example, you can:



- Download to local storage



- Save the file to data uploader
(The file can then be found for easy access in the Data Validation page)



- Copy to clipboard

3

Data Example #1

```
1 v {  
2 v   "artist": {  
3 v     "artist_name": "consectetur magna dolor",  
4 v     "instruments": [  
5 v       [  
6 v         "harmonica",  
7 v         "brass",  
8 v         "keyboards",  
9 v         "guitar",  
10 v        "woodwinds",  
11 v        "guitar",  
12 v        "harmonica",  
13 v        "bass",  
14 v        "harmonica",  
15 v        "drums",  
16 v        "percussion",  
17 v        "drums",  
18 v        "vocals",  
19 v        "brass",  
20 v        "bass",  
21 v        "bass"
```

Data Example #2

```
1 v {  
2 v   "t_number": -4919358.7109660655,  
3 v   "title": "in",  
4 v   "length": "21:54:30.163Z",  
5 v   "featured": [  
6 v     [  
7 v       "artist_name": "magna nisi minim est",  
8 v       "instruments": [  
9 v         "
```

Data Example #3

```
1 v {  
2 v   "t_number": -4919358.7109660655,  
3 v   "title": "in",  
4 v   "length": "21:54:30.163Z",  
5 v   "featured": [  
6 v     [  
7 v       "artist_name": "magna nisi minim est",  
8 v       "instruments": [  
9 v         "
```

Other Features

JADN Sandbox Home Creation Visualization Translation Validation Transformation Generation About

Home

Creation
Create and edit JADN schemas using forms, view JADN schemas in JSON format.
Create schema compliant data instances (documents, messages).

Schemas Data Instances

Data Validation
Validate data instances in various data language formats against a JADN Schema.

CBOR JSON Relax (XML)

Schema Visualization
Convert a JADN Schema into different visual representations.

Schema Translation
Translate a JADN Schema to another Schema format.

Home

Creation
Create and edit JADN schemas using forms, view JADN schemas in JSON format.
Create schema compliant data instances (documents, messages).

Schemas Data Instances

Data Validation
Validate data instances in various data language formats against a JADN Schema.

CBOR JSON Relax (XML)

Schema Visualization
Convert a JADN Schema into different visual representations.

Schema Translation
Translate a JADN Schema to another Schema format.

GraphViz HTML JIDL MarkDown PlantUML

JSON Relax (XML) XSD

Schema Transformation
Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc.).

Example Data Generation
Generate various example data instances based off of a Schema.

Theme ▾

Light
Dark

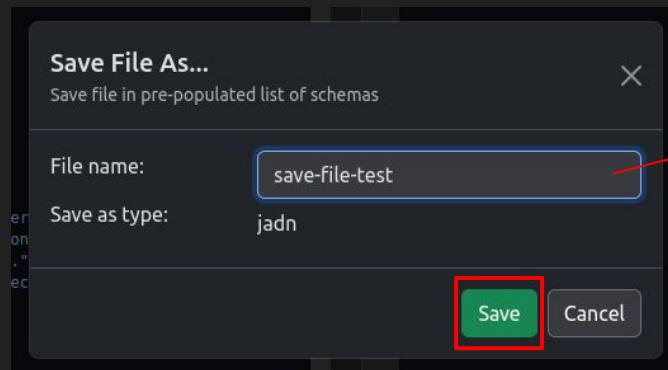
v0.9.1 1702496478525

Theme Switcher: Choose between light and dark mode.

Note: Dark is the default theme.



Add custom files (Save as...)



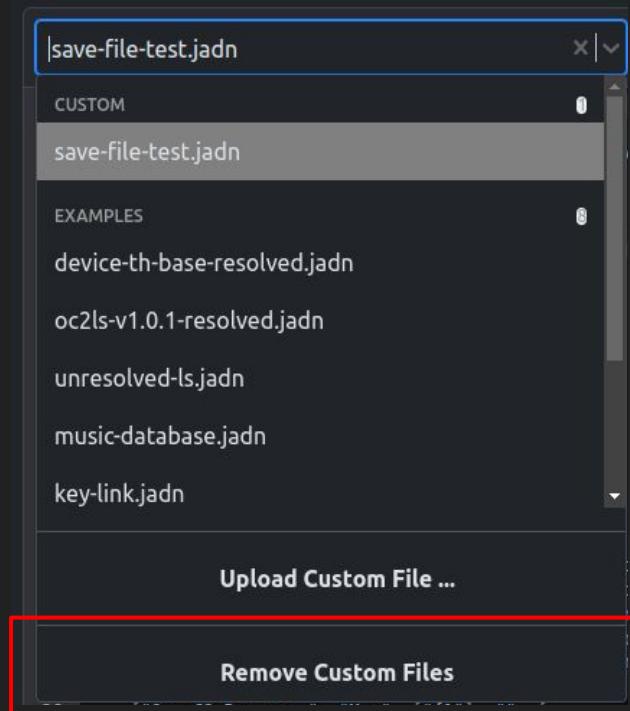
CUSTOM
save-file-test.jadn
EXAMPLES
device-th-base-resolved.jadn
oc2ls-v1.0.1-resolved.jadn
unresolved-ls.jadn
music-database.jadn
key-link.jadn

Upload Custom File ...

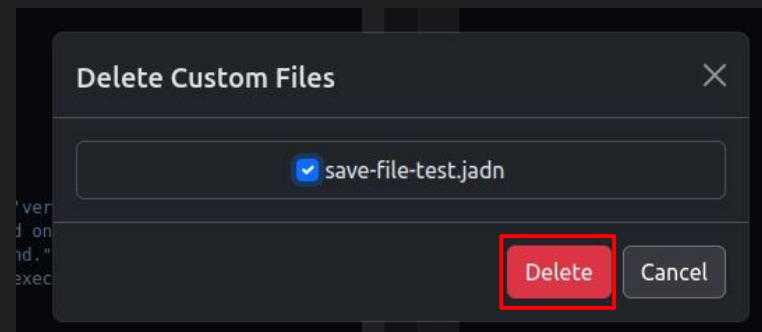
Remove Custom Files

The file saver will allow you to name and save data into a file to be found in the file loader for future use.

Delete custom files

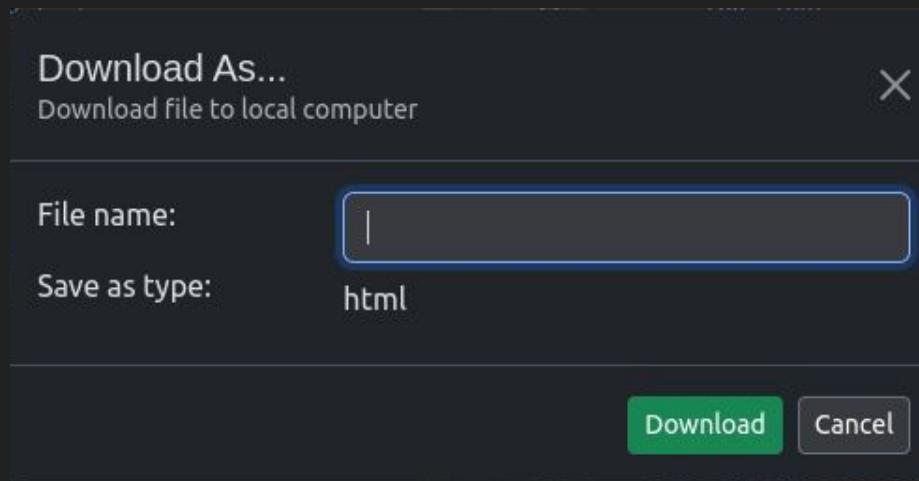


Custom files can be removed from the file loader if no longer needed.





Download files to local storage (Download As..)



Files can be named and saved to your Downloads Folder.



Docker Desktop Start Up Guide

JADN Sandbox

Docker Table of Contents

[How to Get the JADN Sandbox Image](#)

[How to Run the JADN Sandbox Image](#)

- [From Images](#)¹
- [From Containers](#)²
- [From Search Bar](#)

[How to Update the JADN Sandbox Image](#)

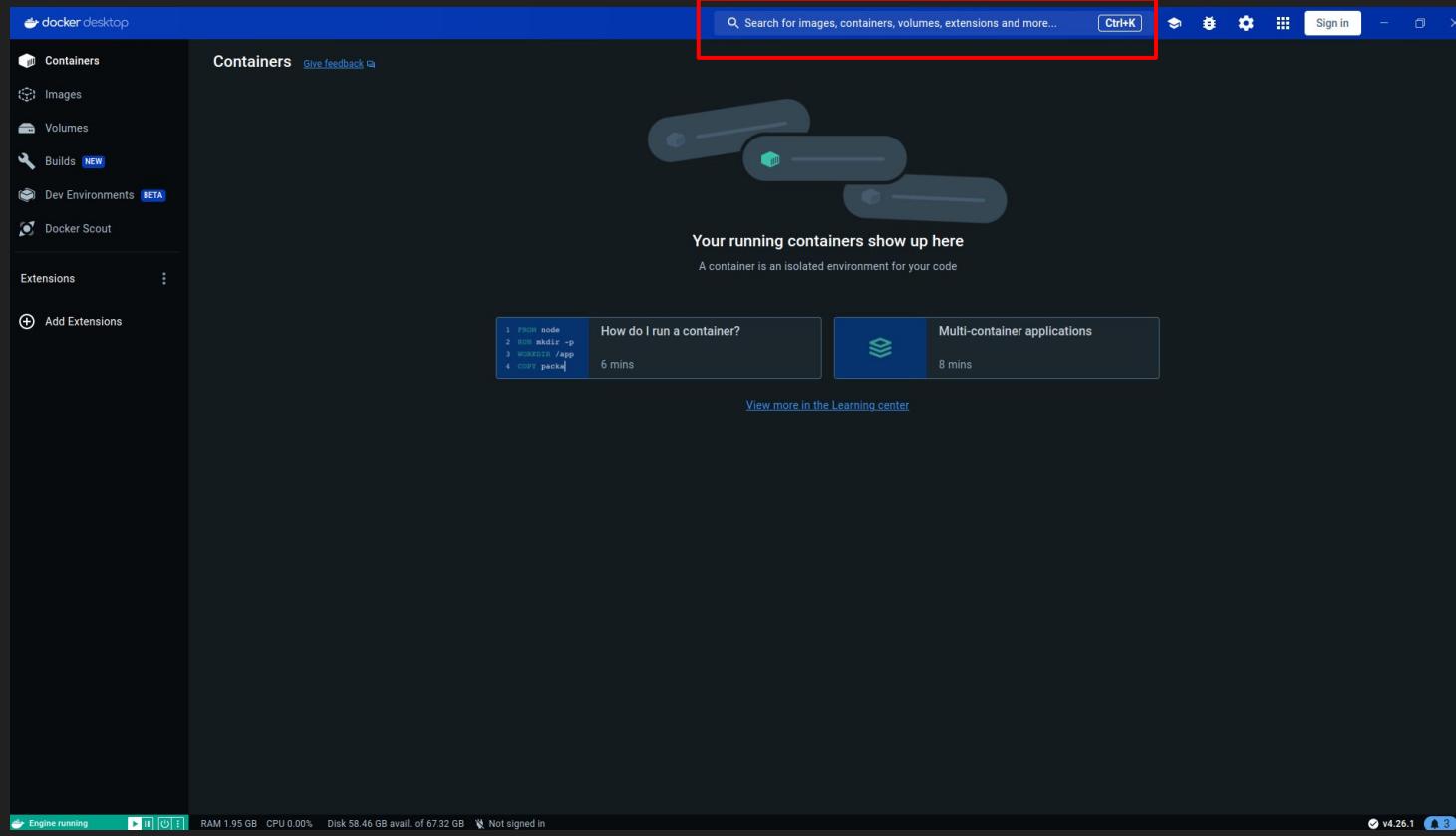
- [From Images](#)
- [From Search Bar](#)

[How to Stop the JADN Sandbox Image](#)

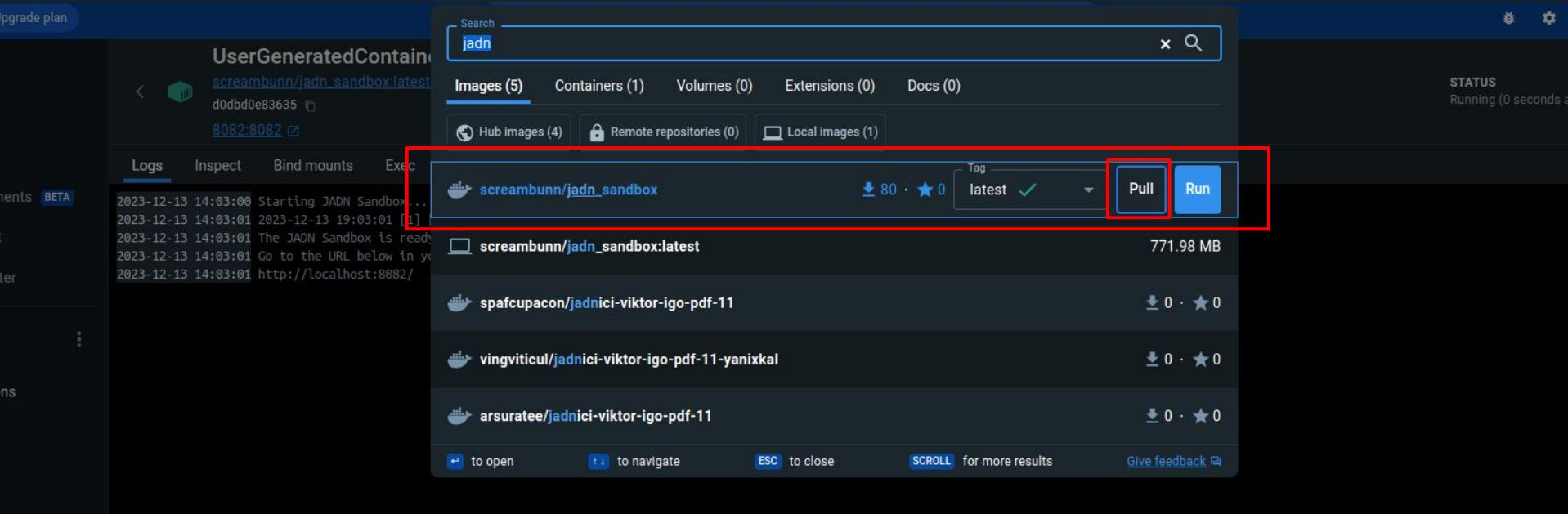
[How to Learn More about Docker](#)

1. A Docker image is like a set of instructions. It is the template loaded onto the container to run it.
2. A Docker container is a self-contained, runnable software application or service created from a docker image.

How to Get the JADN Sandbox Image



Click in the search bar, search: jadn



Then, click Pull: screambunn/jadn_sandbox

How to Run the JADN Sandbox Image

[From Images](#)

[From Containers](#)

[From Search Bar](#)

From Images

Docker Desktop Upgrade plan

Search for images, containers, volumes, extensions and more... Ctrl+K

Containers

Images Give feedback

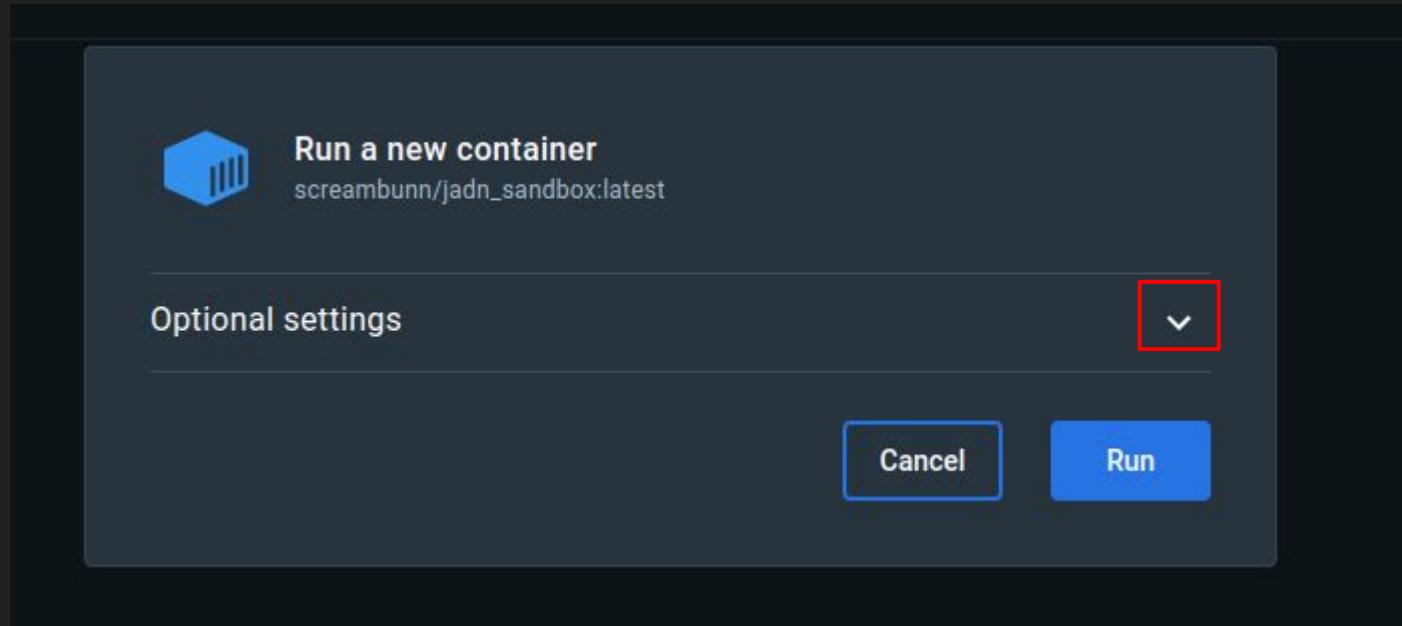
Local Hub Artifactory EARLY ACCESS

0 Bytes / 1.41 GB in use 1 images Last refresh: 2 hours ago

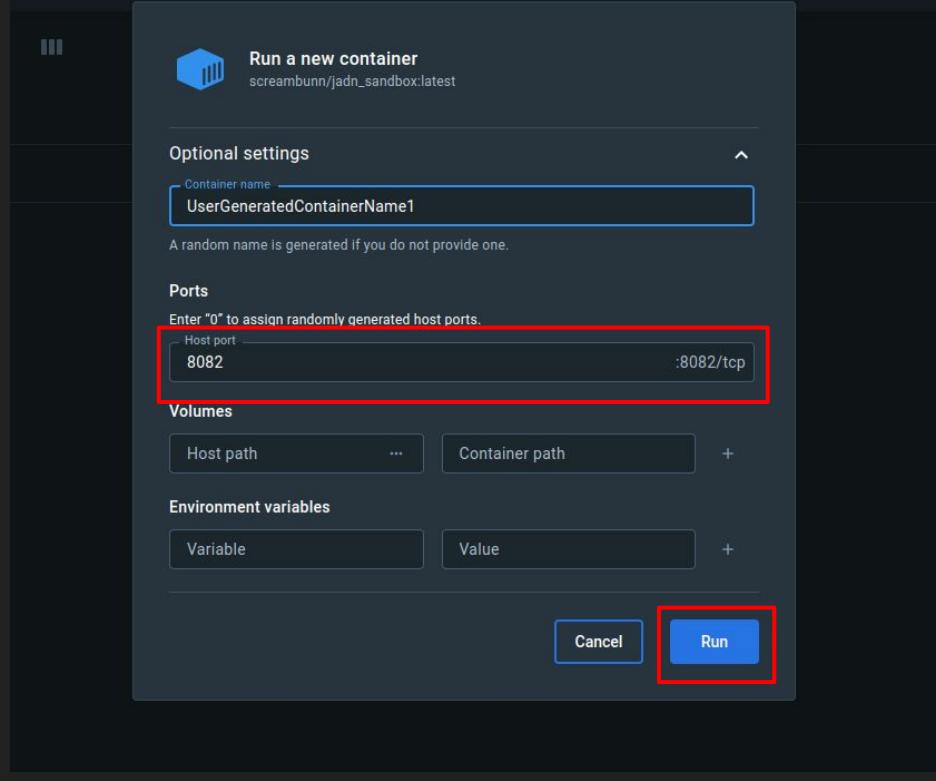
Search

Name	Tag	Status	Created	Size	Actions
screambunn/jadn_sandbox	latest	Unused	22 hours ago	771.98 MB	Run
b6d3e6af3fb0					

Under Images, find the desired image and click the play button to run the image.



After clicking the play button, this window will appear.
You will need to enter information on the container to be able to run the image.
Click the carrot icon (v) to expand the Optional settings.



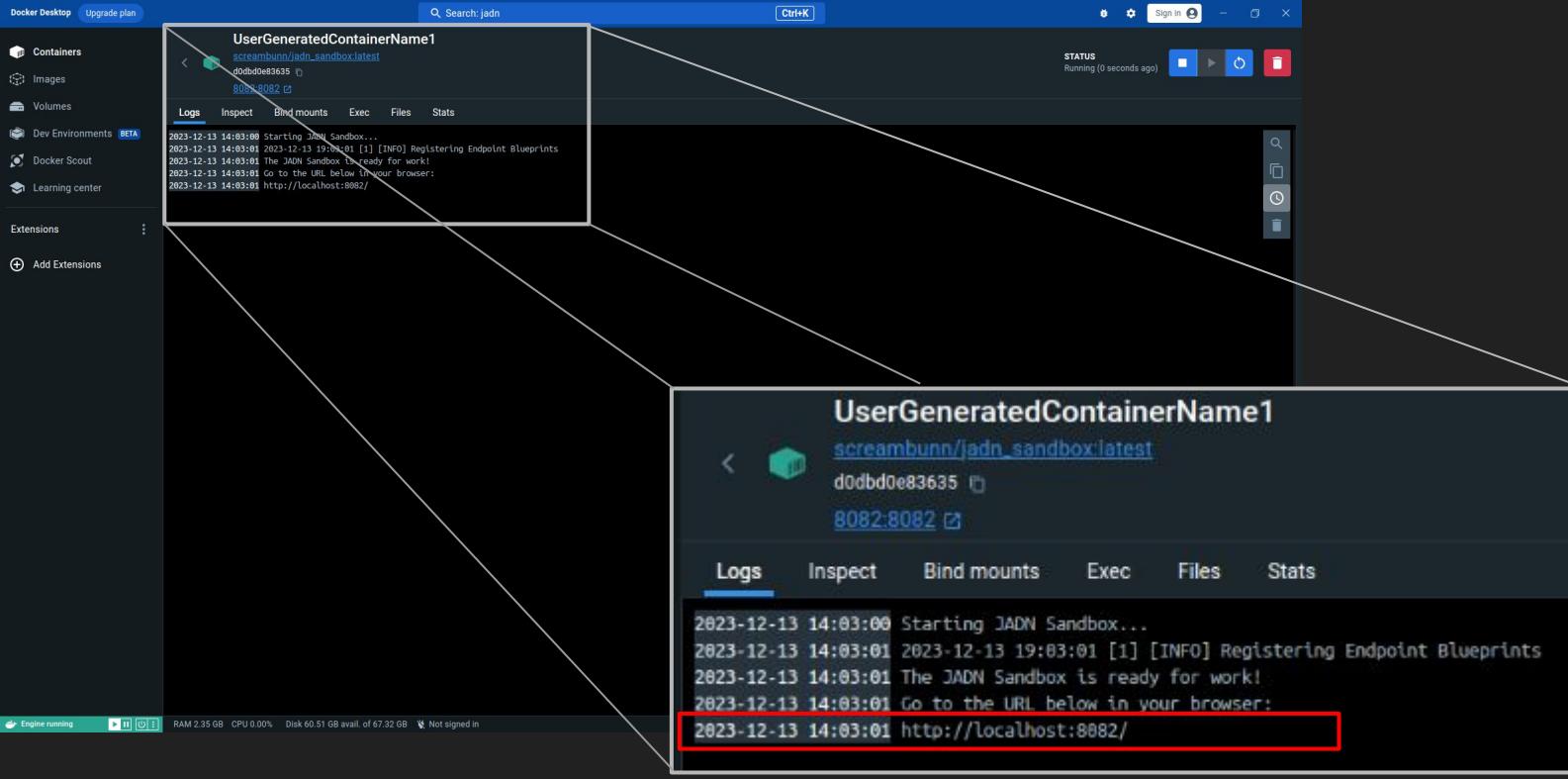
After clicking the carrot icon, you will see the settings as shown.

The Container Name is optional, but has been provided as an example (*UserGeneratedContainerName1*).

Note: no spaces allowed in the container name.

REQUIRED INFORMATION: Host Port - Enter: **8082**

Click Run.



After clicking on Run, a container will appear.

Click on the link provided to open the image in a browser window.

You can also enter the link itself in the browser:

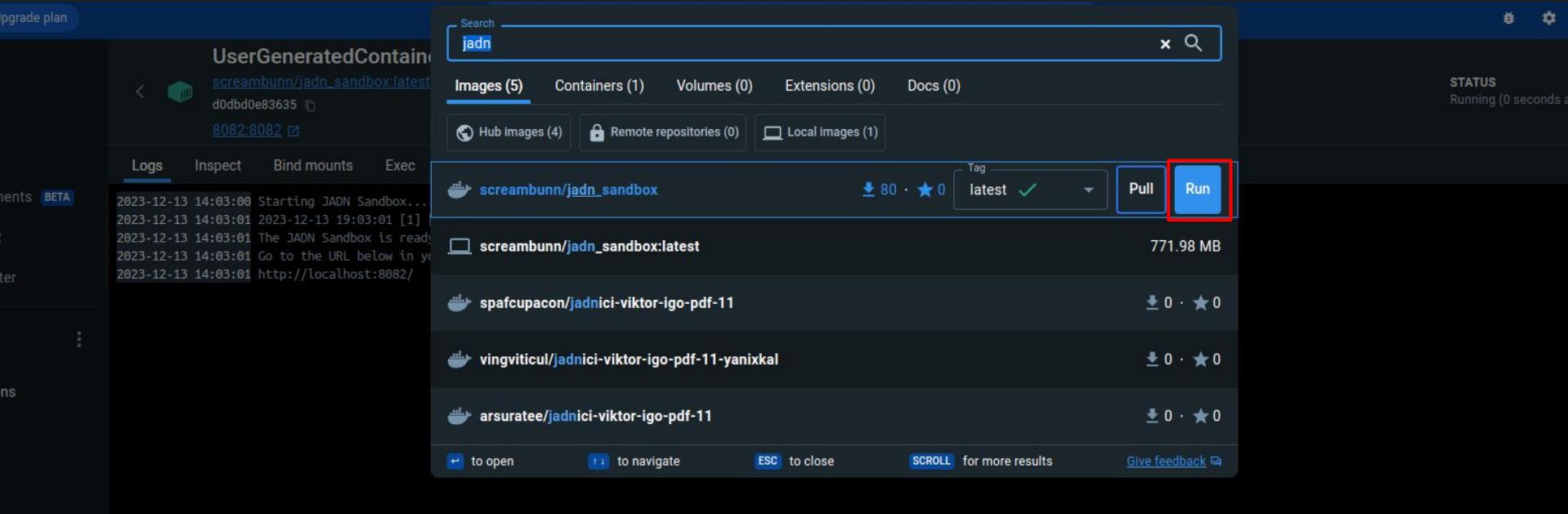
<http://localhost:8082/>

The screenshot shows the Docker Desktop application interface. On the left sidebar, there are icons for Containers, Images, Volumes, Dev Environments (BETA), Docker Scout (EARLY ACCESS), Learning center, Extensions, and a button to Add Extensions. The main area displays a container named 'amazing_brahmagupta' with the image 'screambunn/jadn_sandbox:latest'. The container's status is 'Running (39 minutes ago)'. A red box highlights the 'Stop' button in the top right corner of the container card. Below the container card, there are tabs for Logs, Inspect, Bind mounts, Terminal, Files, and Stats. The 'Logs' tab is selected, showing the following log entries:

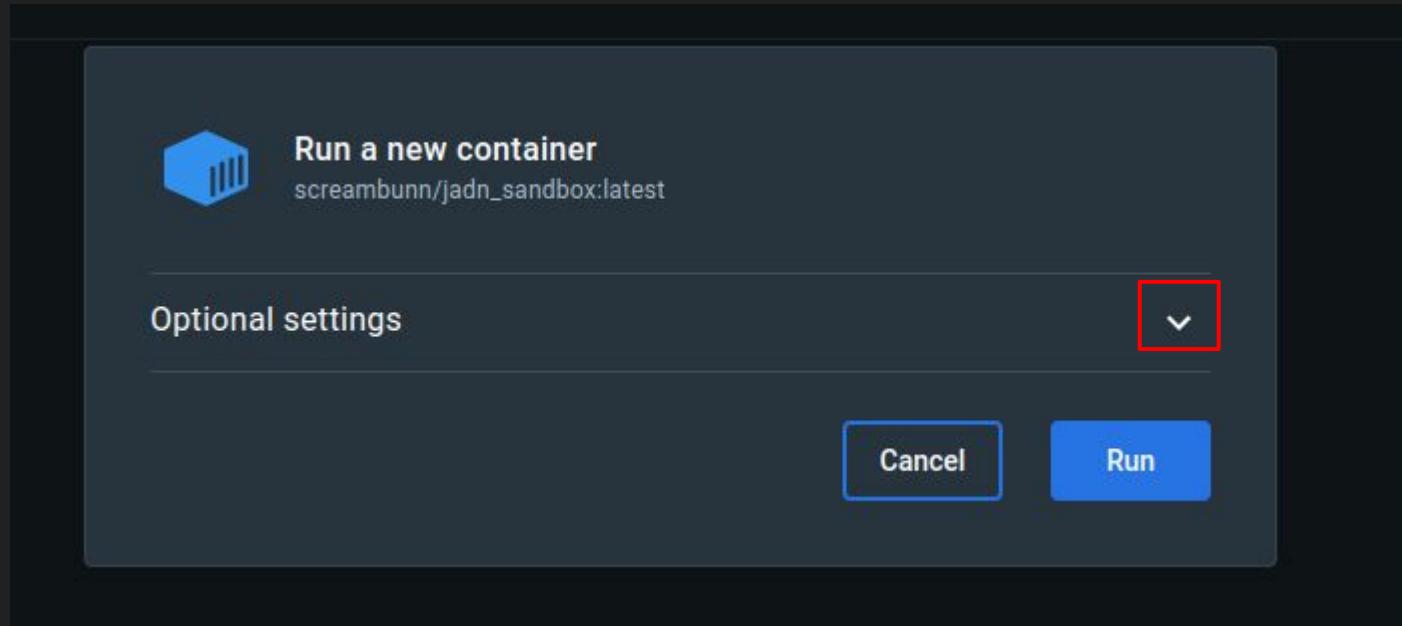
```
2024-02-07 14:28:23 Starting JADN Sandbox...
2024-02-07 14:28:23 2024-02-07 19:28:23 [1] [INFO] Registering Endpoint Blueprints
2024-02-07 14:28:24 The JADN Sandbox is ready for work!
2024-02-07 14:28:24 Go to the URL below in your browser:
2024-02-07 14:28:24 http://localhost:8082/
```

To stop the container, click the stop button.

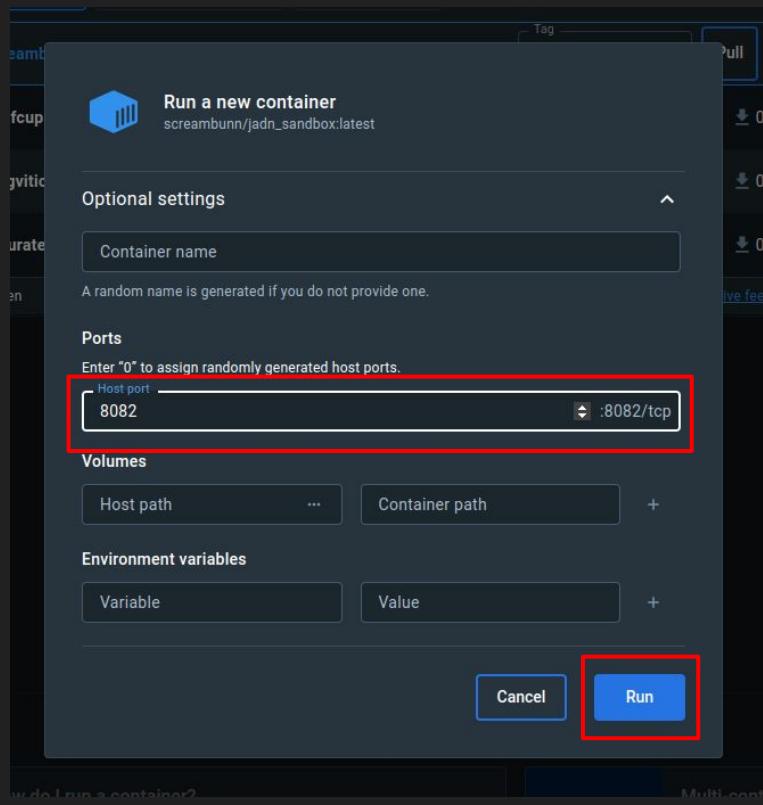
From the Search Bar



In the search bar, search: jadn
Then, click Run: screambunn/jadn_sandbox



After clicking the play button, this window will appear.
You will need to enter information on the container to be able to run the image.
Click the carrot icon (v) to expand the Optional settings.



After clicking the carrot icon, you will see the settings as shown.

The Container Name is optional.

Note: no spaces allowed in the container name.

REQUIRED INFORMATION: Host Port - Enter: **8082**

Click Run.

Docker Desktop Update to latest

Search for images, containers, volumes, extensions and more... **⌘K**

amazing_brahmagupta

screambunn/jadn_sandbox:latest

a3868fda2581

8082:8082

STATUS
Running (39 minutes ago)

Logs Inspect Bind mounts Terminal Files Stats Stop

2024-02-07 14:28:23 Starting JADN Sandbox...
2024-02-07 14:28:23 2024-02-07 19:28:23 [1] [INFO] Registering Endpoint Blueprints
2024-02-07 14:28:24 The JADN Sandbox is ready for work!
2024-02-07 14:28:24 Go to the URL below in your browser:
2024-02-07 14:28:24 <http://localhost:8082/>

Containers Images Volumes Dev Environments **BETA** Docker Scout **EARLY ACCESS** Learning center Extensions Add Extensions

After clicking on Run, a container will appear.
Click on the link provided to open the image in a browser window.
You can also enter the link itself in the browser:
<http://localhost:8082/>

From Containers

Docker Desktop Upgrade plan

Q Search: jadn Ctrl+K

Containers Images Volumes Dev Environments BETA Docker Scout Learning center Extensions Add Extensions

Containers Give feedback

Container CPU usage 0.04% / 800% (8 cores allocated) Container memory usage 119.9MB / 3.62GB Show charts

Search Only show running containers

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
elegant_murdock 0592632c37e8	screambunn/jadn_sandbox:latest	Exited	0.04%	8082:8082	26 seconds ago	
stupefied_rosalind 6fe0cec2accd	screambunn/jadn_sandbox:latest	Running	0%	8082:8082	0 seconds ago	

The screenshot shows the Docker Desktop interface with the 'Containers' tab selected. At the top, there are performance metrics for CPU usage (0.04% / 800%) and memory usage (119.9MB / 3.62GB). Below these are search and filter options. The main area displays a table of running containers. One container, 'stupefied_rosalind', is highlighted with a red box around its status cell ('Running') and another red box around its stop button in the actions column.

First, make sure to stop any running containers.

Docker Desktop Upgrade plan

Search: jadn Ctrl+K

Containers Give feedback

Container CPU usage ⓘ Container memory usage ⓘ Show charts ▾

No containers are running.

Search Only show running containers

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
elegant_murdock 0592632c37e8	screambunn/jadn_sandbox:latest	Exited	N/A	8082:8082	2 minutes ago	
stupefied_rosalind 6fe0cec2accd	screambunn/jadn_sandbox:latest	Exited	N/A	8082:8082	4 seconds ago	

Containers Images Volumes Dev Environments BETA Docker Scout Learning center Extensions Add Extensions

Click play on the desired container and then click on link to see terminal of running container.

Docker Desktop Upgrade plan

Search: jadn Ctrl+K

elegant_murdock

screambunn/jadn_sandbox:latest

0592632c37e8

8082:8082

STATUS Running (32 seconds ago)

Logs Inspect Bind mounts Exec Files Stats

2023-12-21 09:36:10 Starting JADN Sandbox...

2023-12-21 09:36:10 2023-12-21 14:36:10 [1] [INFO] Registering Endpoint Blueprints

2023-12-21 09:36:10 The JADN Sandbox is ready for work!

2023-12-21 09:36:10 Go to the URL below in your browser:

2023-12-21 09:36:10 http://localhost:8082/

2023-12-21 09:38:58 Starting JADN Sandbox...

2023-12-21 09:38:58 2023-12-21 14:38:58 [1] [INFO] Registering Endpoint Blueprints

2023-12-21 09:38:58 The JADN Sandbox is ready for work!

2023-12-21 09:38:58 Go to the URL below in your browser:

2023-12-21 09:38:58 http://localhost:8082/

2023-12-21 09:41:35 Starting JADN Sandbox...

2023-12-21 09:41:35 2023-12-21 14:41:35 [1] [INFO] Registering Endpoint Blueprints

2023-12-21 09:41:35 The JADN Sandbox is ready for work!

2023-12-21 09:41:35 Go to the URL below in your browser:

2023-12-21 09:41:35 http://localhost:8082/

Click on the link provided to open the image in a browser window.

You can also enter the link itself in the browser:

<http://localhost:8082/>

How to Update the JADN Sandbox Image

[From Images](#)

[From Search Bar](#)

From Images

Docker Desktop Upgrade plan

Containers [Give feedback](#)

Images

Volumes

Dev Environments **BETA**

Docker Scout

Learning center

Extensions ⋮

Add Extensions

Search: jadn Ctrl+K

Container CPU usage ⓘ Container memory usage ⓘ

No containers are running.

No containers are running.

Show charts ⌂

<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	kind_cori f5070dd74397	screambunn/jadn_sandbox:latest	Exited	N/A	8082:8082	31 seconds ago	⋮ Delete

OPTIONAL: Delete all containers related to the image (screambunn/jadn_sandbox:latest) by clicking the delete button (trash can icon) under Action.

Note: Containers use the image that it was created with. Even though the image is named the same, its versioning is different. Therefore older containers will be out of date.

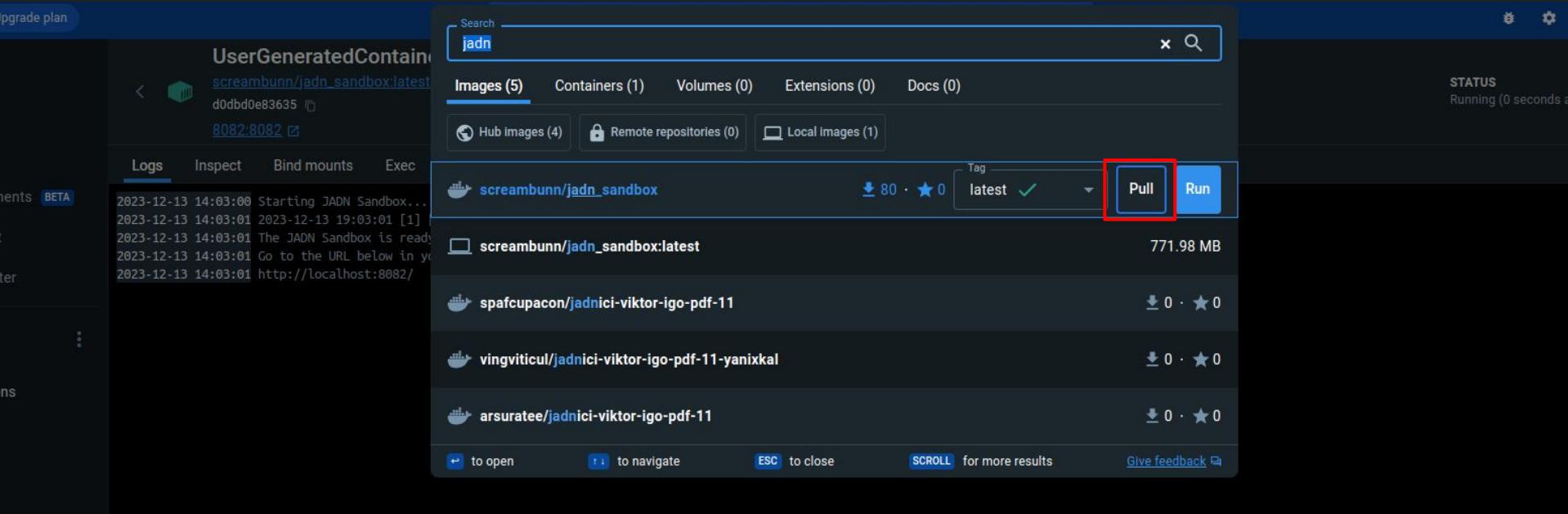
The screenshot shows the Docker Desktop interface. The left sidebar includes options like Containers, Images (which is selected and highlighted with a red box), Volumes, Dev Environments (Beta), Docker Scout, Learning center, Extensions, and Add Extensions. The main area is titled 'Images' with a search bar and a 'Ctrl+K' keyboard shortcut. It displays a table of images with columns for Name, Tag, Status, Created, Size, and Actions. A single image entry is shown: 'screambunn/jadn_sandbox' (tag latest) - Unused, created 22 hours ago, size 771.93 MB. A context menu is open over this entry, also highlighted with a red box, showing options: 'View packages and CVEs', 'Pull' (selected and highlighted with a red box), and 'Push to Hub'.

Name	Tag	Status	Created	Size	Actions
screambunn/jadn_sandbox	latest	Unused	22 hours ago	771.93 MB	<ul style="list-style-type: none">▶⋮✖<small>View packages and CVEs</small><small>Pull</small> (Selected)<small>Push to Hub</small>

Under Images, Go to Actions and click the 3 dotted icon.
A menu drop down will appear. Click Pull.
Then go through the [steps](#) to run the image.

Pulling image..

From the Search Bar



In the search bar, search: jadn
Then, click Pull: screambunn/jadn_sandbox

How to Stop the JADN Sandbox Image

Docker Desktop Upgrade plan

Search for images, containers, volumes, extensions and more... Ctrl+K

Containers Give feedback ⓘ

Container CPU usage ⓘ Container memory usage ⓘ Show charts ▾

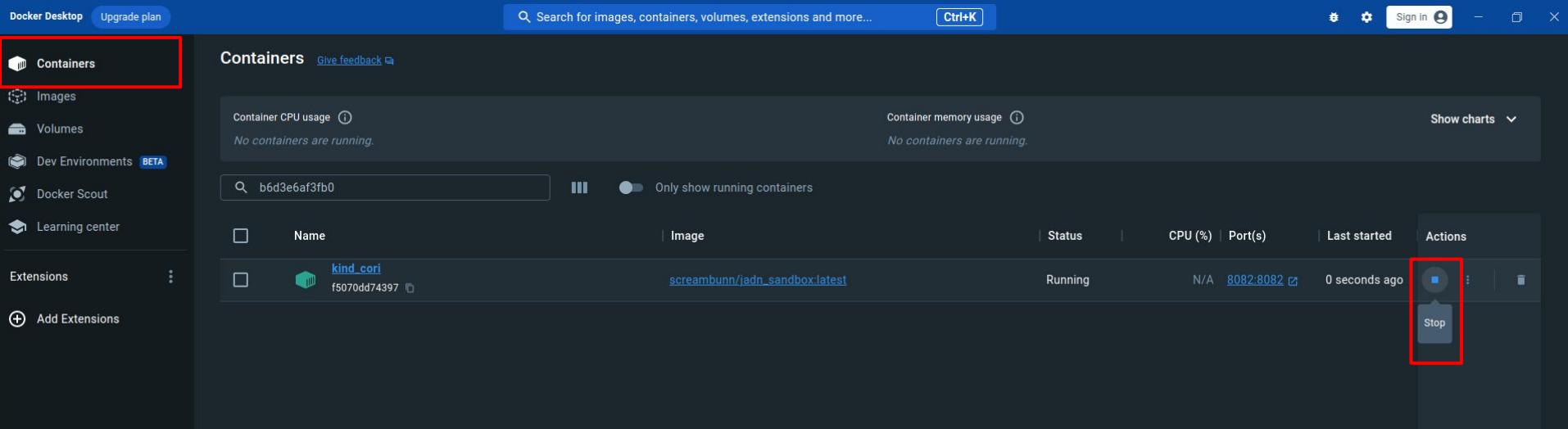
No containers are running.

b6d3e6af3fb0

Only show running containers

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
kind_cori f5070dd74397	screambunn/jadn_sandbox:latest	Running	N/A	8082:8082 ↗	0 seconds ago	 Stop

Add Extensions



Click the stop button under Action in Containers.

Containers

Images

Volumes

 Dev Environments BETA Docker Scout EARLY ACCESS

Learning center

Extensions

Add Extensions

sad_ritchie

[screambunn/jadn_sandbox:latest](#)

00bd8f6577d3

STATUS

Running (0 seconds ago)

[Logs](#) [Inspect](#) [Bind mounts](#) [Terminal](#) [Files](#) [Stats](#)

```
2024-02-07 15:21:40 Starting JADN Sandbox...
2024-02-07 15:21:41 2024-02-07 20:21:41 [1] [INFO] Registering Endpoint Blueprints
2024-02-07 15:21:41 The JADN Sandbox is ready for work!
2024-02-07 15:21:41 Go to the URL below in your browser:
2024-02-07 15:21:41 http://localhost:8082/
2024-02-07 15:21:48 Starting JADN Sandbox...
2024-02-07 15:21:48 2024-02-07 20:21:48 [1] [INFO] Registering Endpoint Blueprints
2024-02-07 15:21:48 The JADN Sandbox is ready for work!
2024-02-07 15:21:48 Go to the URL below in your browser:
2024-02-07 15:21:48 http://localhost:8082/
2024-02-07 15:21:52 Starting JADN Sandbox...
2024-02-07 15:21:52 2024-02-07 20:21:52 [1] [INFO] Registering Endpoint Blueprints
2024-02-07 15:21:52 The JADN Sandbox is ready for work!
2024-02-07 15:21:52 Go to the URL below in your browser:
2024-02-07 15:21:52 http://localhost:8082/
```



Alternatively, Click the stop button within the running container

How to Learn More about Docker

- Containers
- Images
- Volumes
- Dev Environments BETA
- Docker Scout
- Learning center**

Learning center [Give feedback](#)

Walkthroughs

Quick hands-on guides to show you around

```
1 FROM node
2 RUN mkdir -p /app
3 WORKDIR /app
4 COPY package.json .
```

How do I run a container?

6 mins



Multi-container applications

6 mins

[View all](#)

AI/ML guides

Get started with AI/ML using Docker



GenAI Stack

Start your GenAI application using Neo4j, Langchain, Ollama, Python, and Docker Compose

Docker for beginners by language

45 min guides written for different programming languages



NodeJS



Python



Go



Java



C# (.NET)



Rust

[Request a guide](#)

Samples

Go to the Learning center to learn more about Docker.