EnergySHR Portal v4

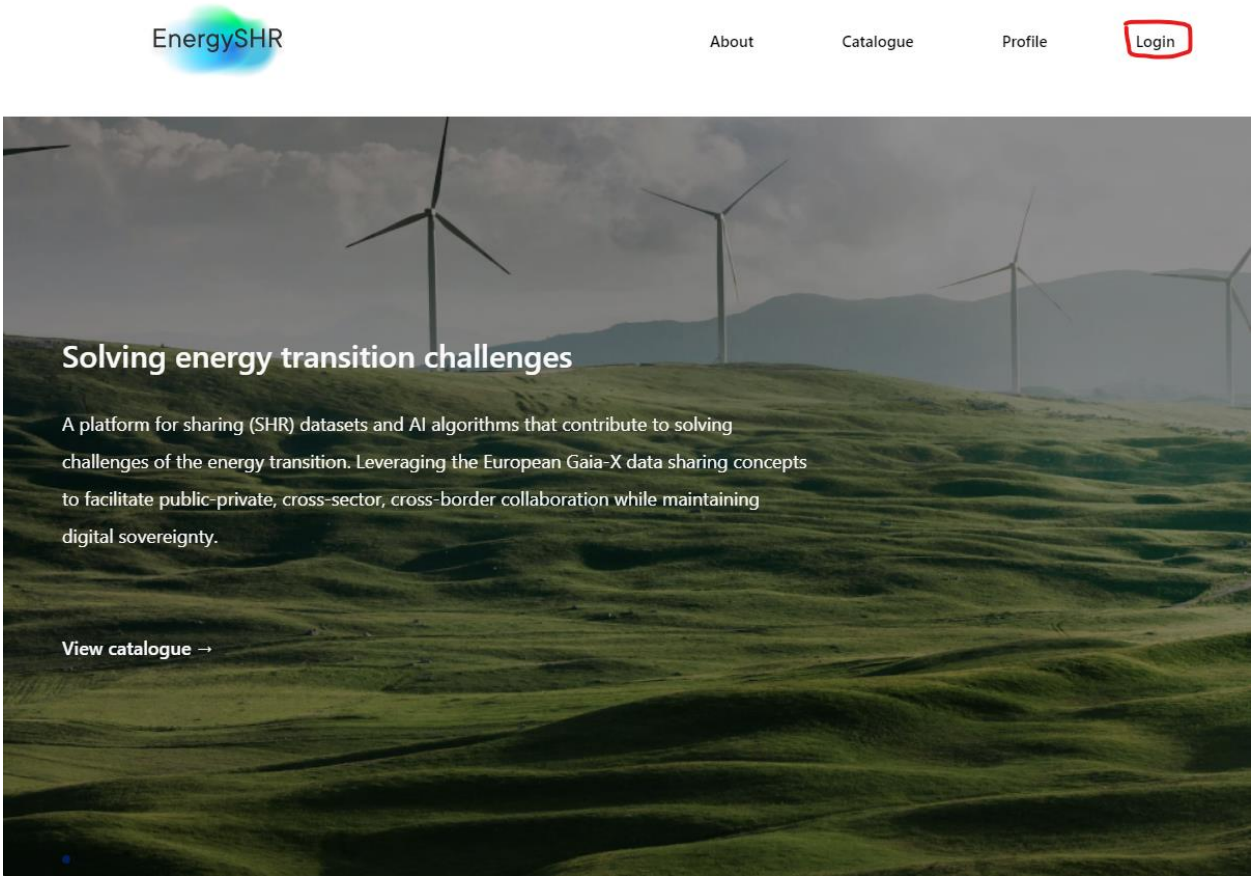
Publish an Algorithm

Manual

Publishing an Algorithm

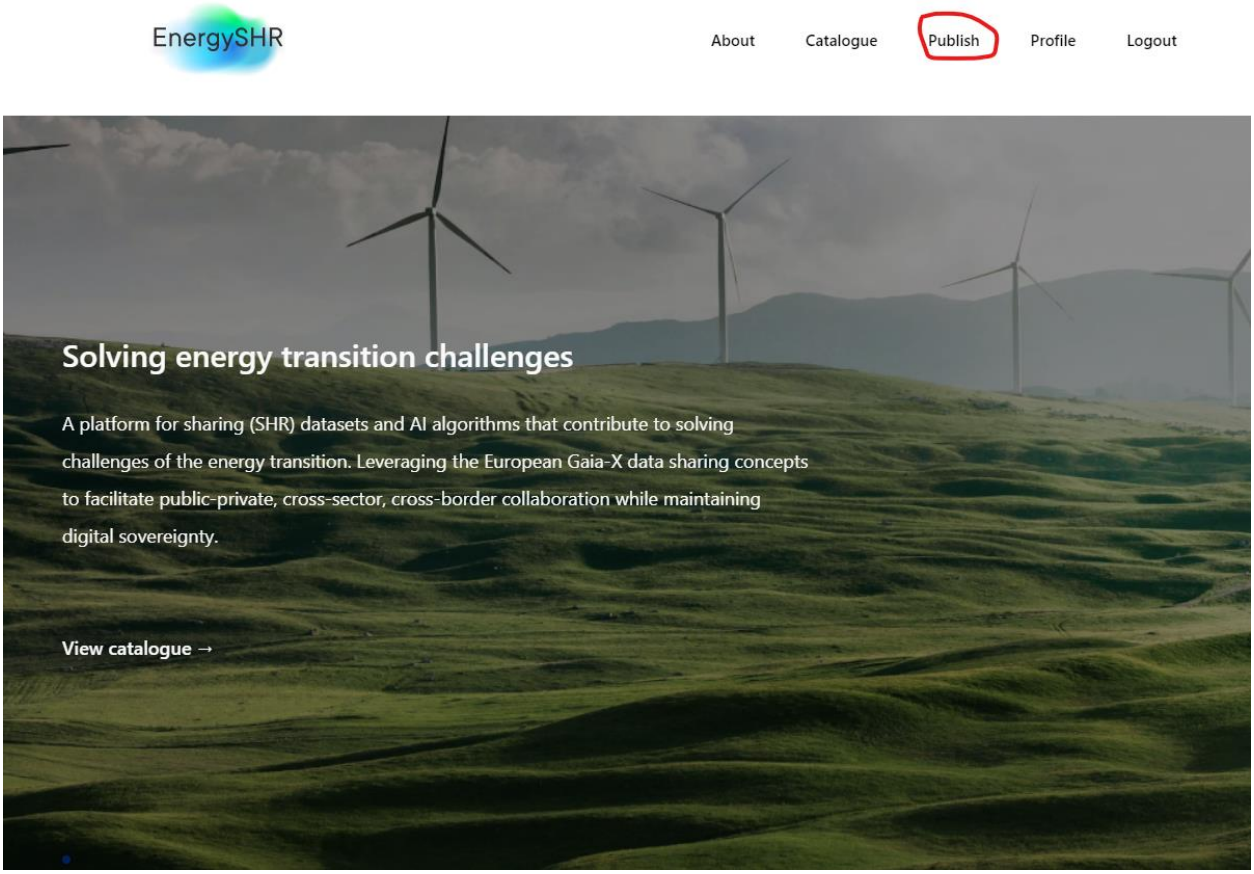
Access the portal

On the landing page, make sure to be authenticated (Logged in)



Click on Publish

After the application is connected to the wallet (now the network and wallet address is shown instead of Connect Wallet), click on publish.



Metadata

Fill in details

- Asset Type (mandatory) → Choose Algorithm
- Title (mandatory) → Enter a title to identify the algorithm
- Description (mandatory) → Enter the description of the asset. It's possible to use markdown to make it more visually appealing.
- Tags (optional) → Enter existing or new tag(s)
- Docker Image (mandatory): Choose the docker image used to run the algorithm
    - Choose node:latest if your algorithm will be javascript code or if the algorithm needs to be downloadable only (see remark below)
    - Choose python:latest if your algorithm will be Python code
    - Choose custom, if you will be using a different language or of the other 2 docker images do not suffice (see below)
    -
- Agree to the terms and conditions (mandatory) -> tick the checkbox

Downloadable Algorithm without Docker

Choose node:latest as Docker Image, even if you will be hosting a downloadable algorithm (see next section) that does not need compute to data and has no docker image. Currently the portal requires you to choose a Docker image for any algorithm, since the Compute to Data environment depends on Docker images. However you will have to decide in a next step whether the algorithm is Downloadable, or only Compute to Data. A Downloadable algorithm doesn't need docker obviously, but the current portal requires you to fill out a docker image regardless. Any programming language and code used in the algorithm is fine. It doesn't have to match the docker image (node/javascript), simply because the docker image isn't being used to begin with.

Custom Docker Image

As mentioned you can use a custom Docker image in case the node:latest or python:latest do not satisfy your needs. How to exactly create a docker image is out of scope for this manual. In order to successfully build a custom docker image, you will need to install docker on your local machine. You can find getting started guides as well as installation guides on the docker website. Once you have installed docker on your machine, you will have to create a so called Dockerfile (see the getting started guide for an explanation).

It is important to note that the Dockerfile will typically host all the libraries and/or executables in a pre-packaged environment that your algorithm is depending on. It typically however should not host the algorithm itself. The reason is that in current version of the portal it is not easy to replace

the Docker image after it is published, except for retracting your algorithm and publishing a new version. An algorithm/script itself, however is easy to update for any published algorithm,

This link is an example of such a Dockerfile. It is extending a python base image, then installs a library called libmagic (Imagemagic) Copies some supporting files to the /app folder and installs python modules required for the execution of the algorithm,

It is important to notice that the Compute2Data environment expects certain INPUT and OUTPUT files in pre-defined locations. This is needed to make sure that all algorithms and datasets can work together. Obviously that doesn't mean that you would be able to run any algorithm on any dataset, as an algorithm typically is very specific about its input data. It means that any algorithm and dataset are compatible in where they expect their data to be.

`/data/inputs`	read	Storage for input data sets, accessible only to the algorithm running in the pod. Contents will be the files themselves, inside indexed folders e.g. `/data/inputs/{did}/{service_id}`.
`/data/ddos`	read	Storage for all DDOs involved in compute job (input data set + algorithm). Contents will json files containing the DDO structure.
`/data/outputs`	read/write	Storage for all of the algorithm's output files. They are uploaded on some form of cloud storage, and URLs are sent back to the consumer.
`/data/logs/`	read/write	All algorithm output (such as `print`, `console.log`, etc.) is stored in a file located in this folder. They are stored and sent to the consumer as well.

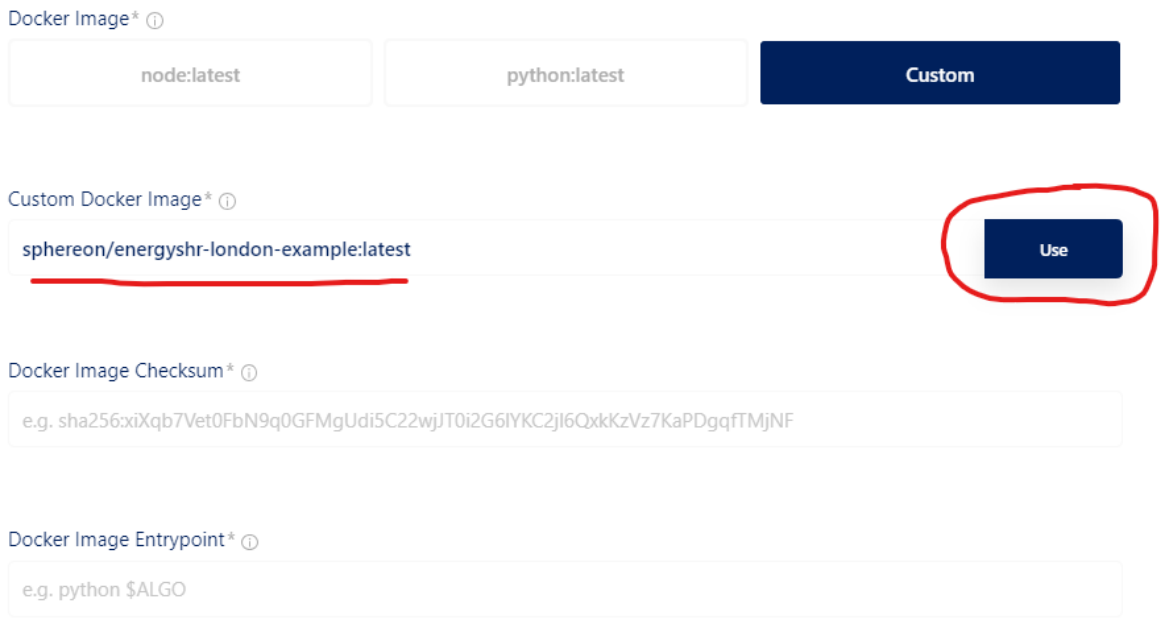
It is important to know that the input dataset typically gets mounted without the original filename in a path like /data/inputs/{did}/0, where 0 is the actual file. So you cannot rely on file-extension to determine proper types for instance. This script nicely shows it is using mimetype/mediatype detection in Python to determine whether the input dataset is of type CSV or ZIP.

After having created a Dockerfile, you will need to build and tag the docker file into an image. Then you will need to publish the Docker image with a docker repository, like for instance https://hub.docker.com When you have published your image with the respective repository you will get the image name and/or full link to the image.

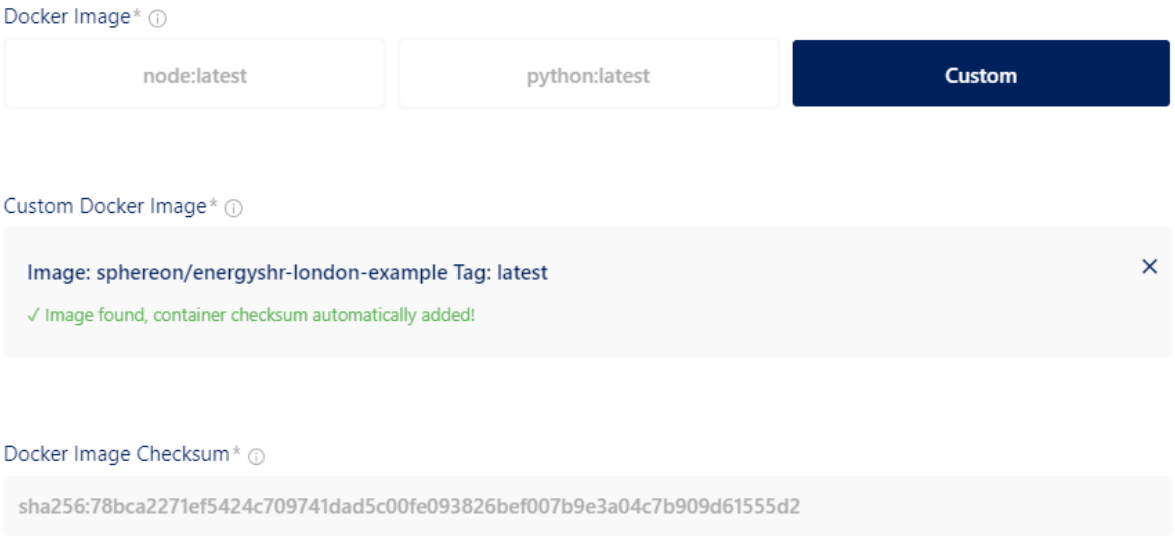
You will need this link when providing the "Custom Docker Image" value. Like shown below.

Note: If you host your docker image on hub.docker.com you do not have to provide the full url. Docker will use this repository by default. For any other repository you will have to provide the full url, which the respective repository will show for your image. You will also have to provide

the tag/version of the Docker image. In the example below this is “latest”. Again the repository will provide you the value to fill out.



After you have filled in the docker image, click on the ‘Use” button. Now the Docker Image checksum will be automatically filled.



The last field to fill out is “Docker Image Entrypoint”. This will be the main entry point that will be used by the Docker image. In other words, this will be the actual algorithm you will provide in step 2. The Docker image itself merely contains the libraries, executables, programs and/or supporting files needed to run the algorithm. The actual algorithm will be downloaded from a URL. This means you can update the algorithm over time. However the docker image currently cannot be changed afterwards. If you do need to use a different Docker image at a later point in time, you will have to publish a new Algorithm.

For the “Docker Image Entrypoint” it is important to know that you can and should use one variable, called $ALGO. This variable will be replaced with the file location of the downloaded algorithm, which will be made available in the docker container.

So the EntryPoint typically starts with a scrip interpreter, or executable, followed by the $ALGO variable. Some examples:

- /bin/sh -c $ALGO
- node $ALGO
- python3.6 $ALGO

After all required fields are filled in and the fields that require validation are successfully validated, the continue button will be enabled and we are ready to go to the next tab (click on continue).

When developing your algorithm you can use this example as inspiration. It is a nice example of not depending on fixed INPUT dataset files, by using mime type detection. It also logs to the log directory and can be run locally without the compute 2 data environment using scripts. Especially the run-local.sh script nicely shows, how it sets an EntryPoint via the command line when run on your local Docker environment, given that the Dockerfile should not contain an EntryPoint when published to the Compute to Data environment.

EnergySHR

EnergySHR

About Catalogue Publish Profile Logout

Publish into GEN-X Testnet

Highlight the important features of your dataset or algorithm to make it more discoverable and catch the interest of data consumers.

1 Metadata 2 Access 3 Pricing 4 Preview 5 Submit

Data NFT*

GX Data NFT — GX-NFT

This NFT represents an asset in Ocean Protocol v4 ecosystems.

Asset Type*

Dataset

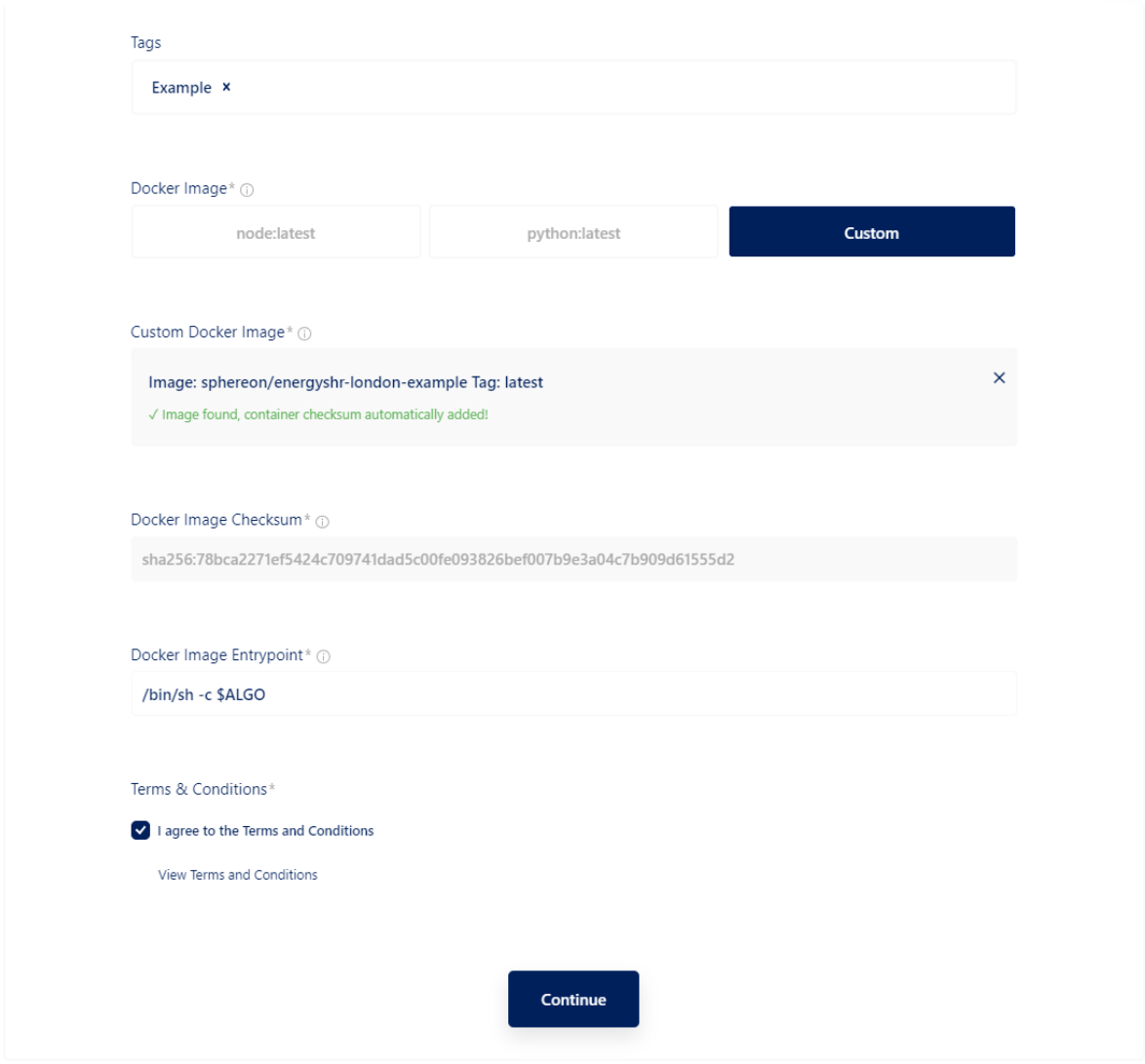
Algorithm

Title*

Example algorithm

Description*

This is my example Algorithm with [markdown](https://example.com) support



Access

- Algorithm privacy → “Keep my algorithm private” means it can only be used for Compute2Data. If you uncheck this checkbox the algorithm can be downloaded like a regular dataset. In that case the algorithm cannot be used in a Compute2Data setting, meaning users can only download the algorithm. The algorithm will never show up as

algorithm, when managing datasets that are available for Compute2Data (see also the note about docker images when providing a non-private, downloadable algorithm above)

- Provider URL(mandatory) → Provide your custom provider service URL or use the Delta DAO provider, which is the default
- File (mandatory) → Provide the actual algorithm which may be an IPFS CID, ARWEAVE Transaction ID or an URL. After providing the details, click on the Validate button!
- Sample file (optional) → Provide a sample of the algorithm output
- Timeout (mandatory) → How long the algorithm will be available after purchase. May Be Forever, 1 Day, 1 Week, 1 Month or 1 Year
- License (optional) → License type
- Terms and Conditions (optional) → Terms and conditions of usage of the dataset. If not filled out, the default terms of the portal will be applicable.

After all required fields are filled in and the fields that require validation are successfully validated (clicking on the validate button), the continue button will be enabled and we are ready to go to the next tab (click on continue).

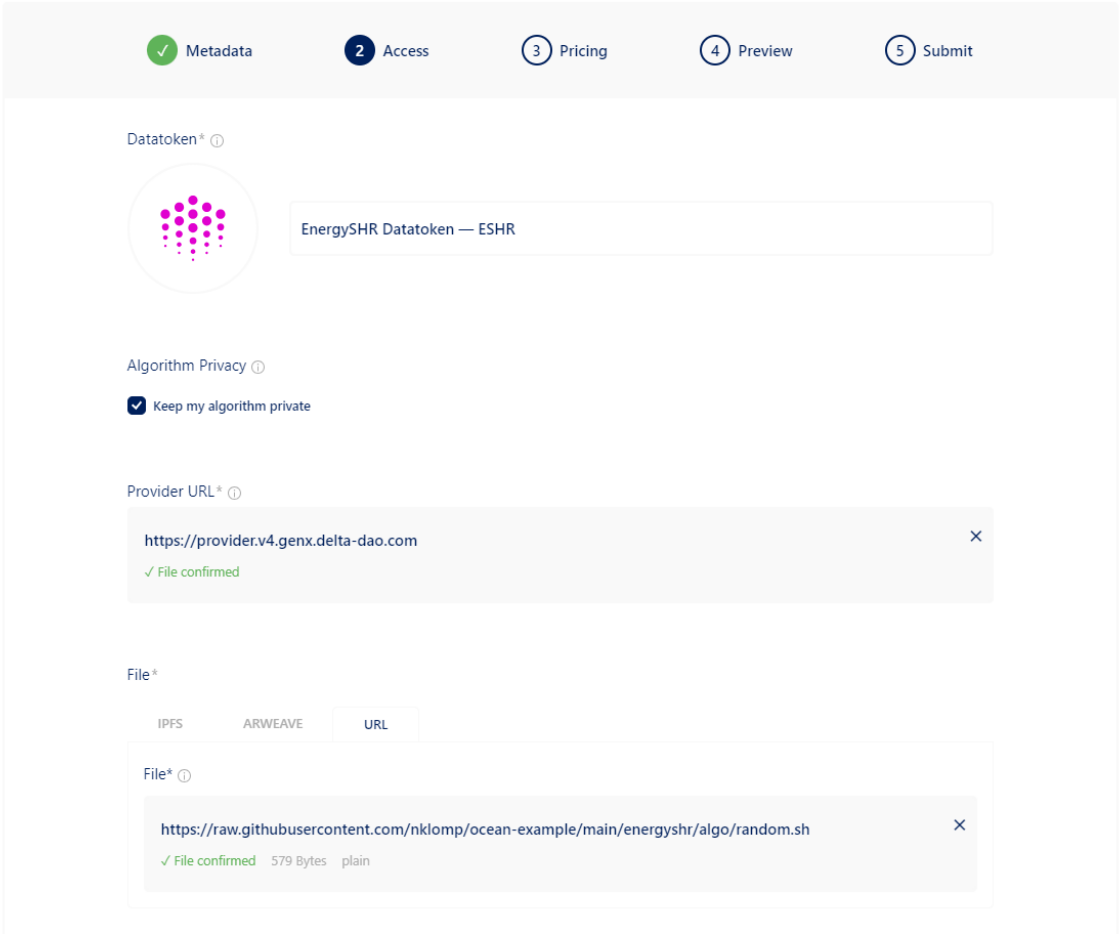
EnergySHR

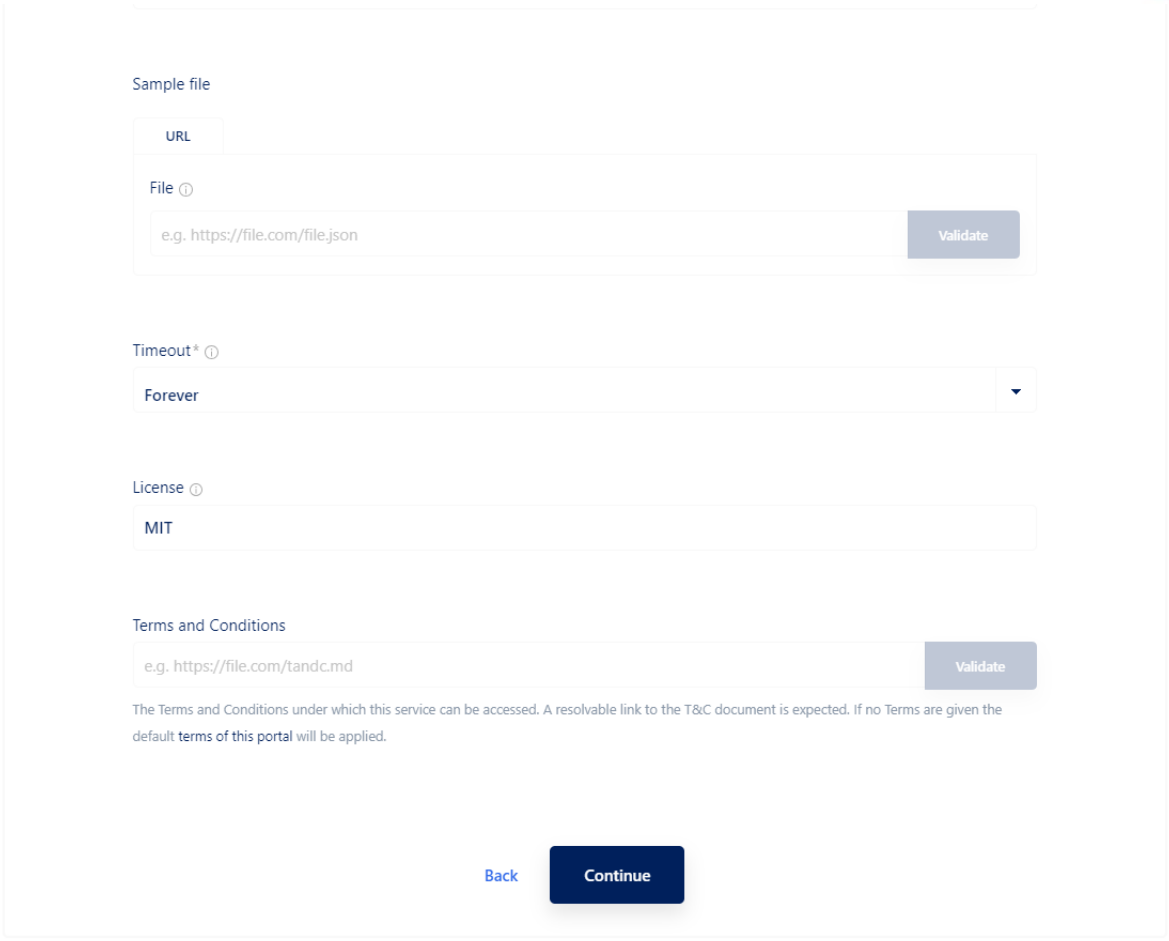
EnergySHR

About Catalogue Publish Profile Logout

Publish into GEN-X Testnet

Highlight the important features of your dataset or algorithm to make it more discoverable and catch the interest of data consumers.





Pricing

- Free → Acknowledge that although your asset is free, network fees are still to be paid

After the check box is selected, the continue button will be enabled, click on it to move to the next tab.

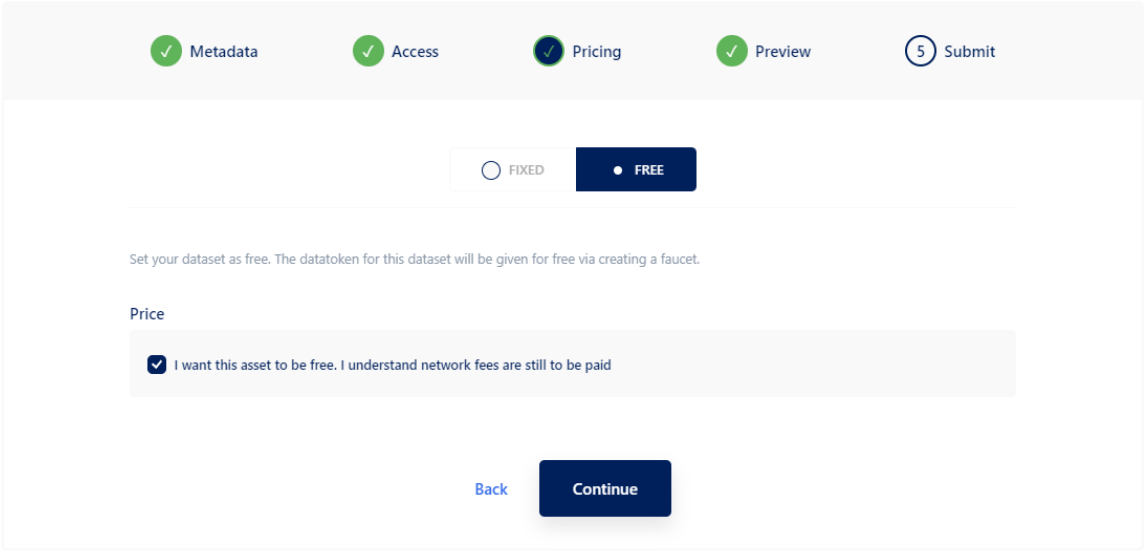
EnergySHR

EnergySHR

About Catalogue Publish Profile Logout

Publish into GEN-X Testnet

Highlight the important features of your dataset or algorithm to make it more discoverable and catch the interest of data consumers.

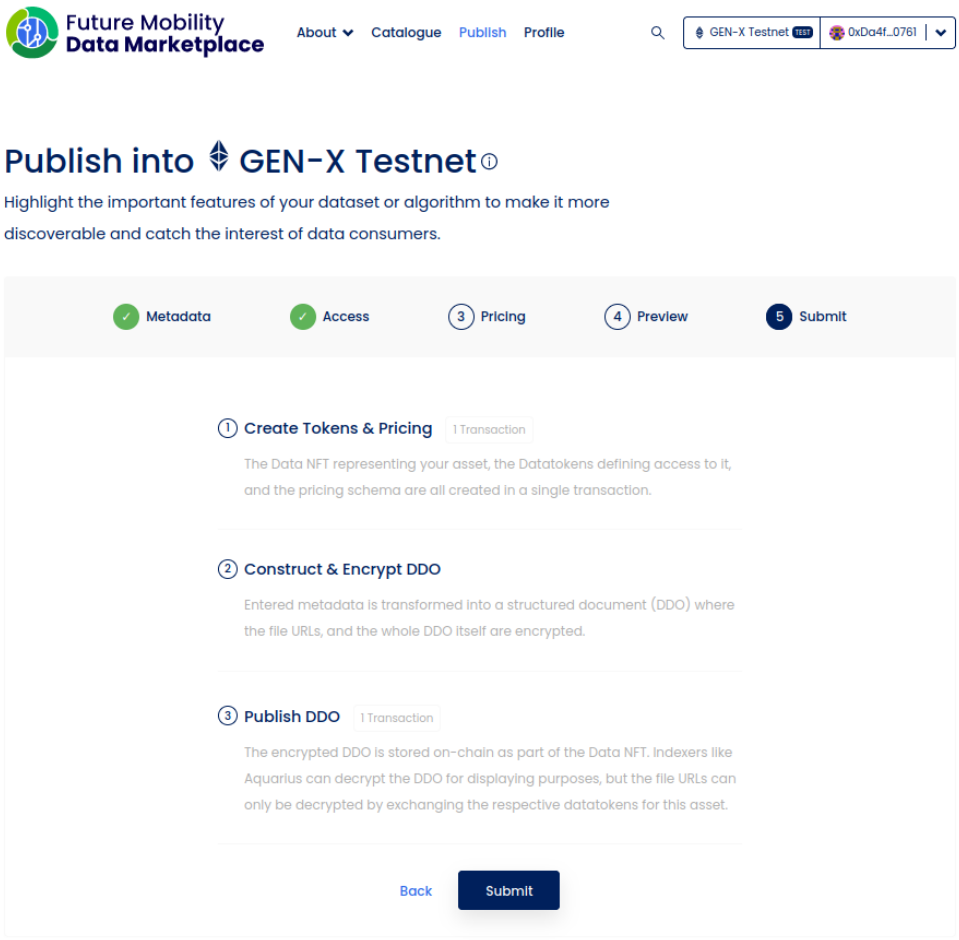


Preview

Check if everything is fine and ready to publish. If it's ready to publish click on continue.

Submit

Click on submit. Step one and three require 1 transaction each and will require confirmation to subtract ocean tokens from your wallet. This wallet however is hidden/headless. Meaning it is operated on your organizations behalf and thus you will not have interact with the wallet yourself directyly



The asset is published.