DTaaS

Digital Twin as a Service (DTaaS)

DTaaS Development Team

Table of contents

1. What is DTaaS?

The Digital Twin as a Service (DTaaS) software platform is useful to **Build, Use and Share** digital twins (DTs).

🦾 **Build**: DTs are built on DTaaS using reusable DT assets available on the platform.

🧑‍🔬 👷 **Use**: Run your DTs on DTaaS.

🤝 **Share**: Share ready-to-use DTs with other users. It is also possible to share the services offered by one DT with other users.

There is an overview of DTaaS available in the form of slides, video, and feature walkthrough.

1.1 License

This software is owned by The INTO-CPS Association and is available under the INTO-CPS License.

DTaaS software platform uses third-party open-source software. These software components have their own licenses.

2. User

2.1 Motivation

How can DT software platforms enable users collaborate to:

- Build digital twins (DTs)
- Use DTs themselves
- Share DTs with other users
- Provide the existing DTs as Service to other users

In addition, how can the DT software platforms:

- Support DT lifecycle
- Scale up rather than scale down (flexible convention over configuration)

2.1.1 Existing Approaches

There are quite a few solutions proposed in the recent past to solve this problem. Some of them are:

- Focus on data from Physical Twins (PTs) to perform analysis, diagnosis, planning etc…
- Share DT assets across the upstream, downstream etc….
- Evaluate different models of PT
- DevOps for Cyber Physical Systems (CPS)
- Scale DT / execution of DT / ensemble of related DTs
- Support for PT product lifecycle

2.1.2 Our Approach

- Support for transition from existing workflows to DT frameworks
- Create DTs from reusable assets
- Enable users to share DT assets
- Offer DTs as a Service
- Integrate the DTs with external software systems
- Separate configurations of independent DT components

2.2 Overview

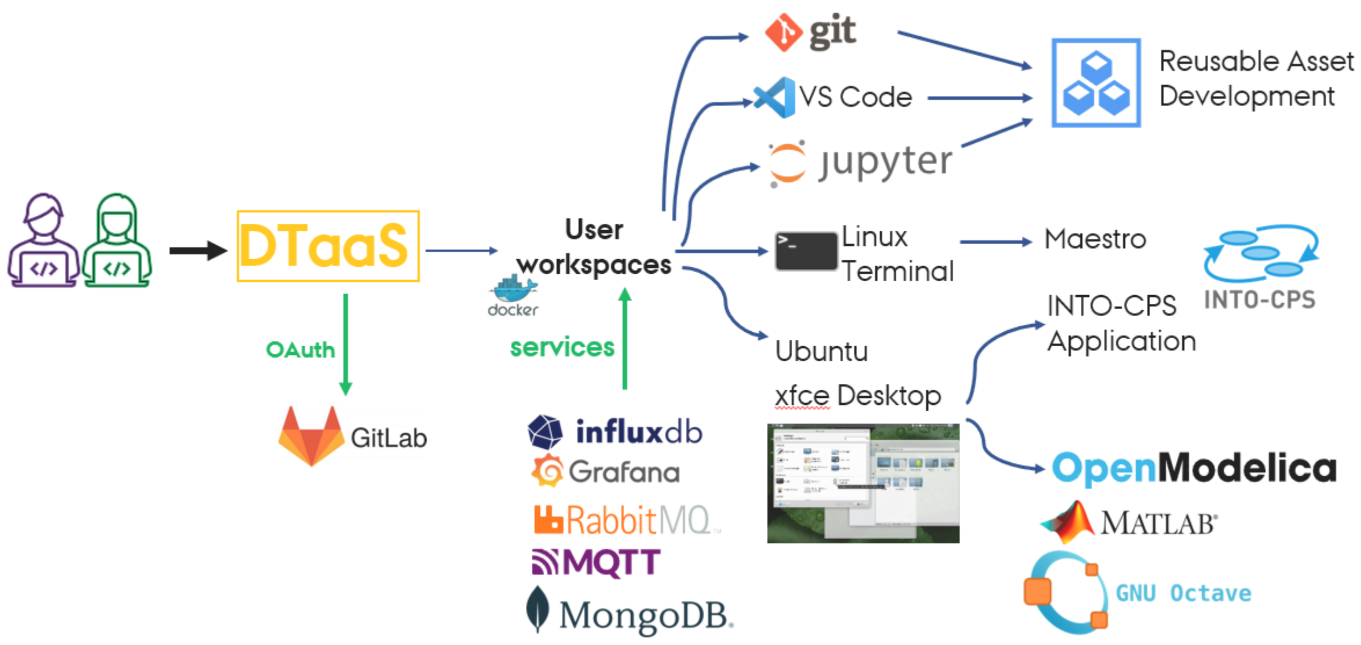
2.2.1 Advantages

The DTaaS software platform provides certain advantages to users:

- Support for different kinds of Digital Twins
- CFD, Simulink, co-simulation, FEM, ROM, ML etc.
- Integrates with other Digital Twin frameworks
- Facilitate availability of Digital Twin as a Service
- Collaboration and reuse
- Private workspaces for verification of reusable assets, trial run DTs
- Cost effectiveness

2.2.2 Software Features

Each installation of DTaaS platform comes with the features highlighted in the following picture.



All the users have dedicated workspaces. These workspaces are dockerized versions of Linux Desktops. The user desktops are isolated so the installations and customizations done in one user workspace do not effect the other user workspaces.

Each user workspace comes with some development tools pre-installed. These tools are directly accessible from web browser. The following tools are available at present:

Tool	Advantage
Jupyter Lab	Provides flexible creation and use of digital twins and their components from web browser. All the native Jupyterlab usecases are supported here.
Jupyter Notebook	Useful for web-based management of their files (library assets)
VS Code in the browser	A popular IDE for software development. Users can develop their digital twin-related assets here.
ungit	An interactive git client. Users can work with git repositories from web browser

In addition, users have access to xfce-based remote desktop via VNC client. The VNC client is available right in the web browser. The xfce supported desktop software can also be run in their workspace.

The DTaaS software platform has some pre-installed services available. The currently available services are:

Service	Advantage
InfluxDB	Time-series database primarly for storing time-series data from physical twins. The digital twins can use an already existing data. Users can also create visualization dashboards for their digital twins.
RabbitMQ	Communication broker for communication between physical and digital twins
Grafana	Visualization dashboards for their digital twins.
MQTT	Lightweight data transfer broker for IoT devices / physical twins feeding data into digital twins.
MongoDB	NoSQL document database for storing metadata of data from physical twins

In addition, the workspaces are connected to the Internet so all the Digital Twins running in the workspace can interact with both the internal and external services.

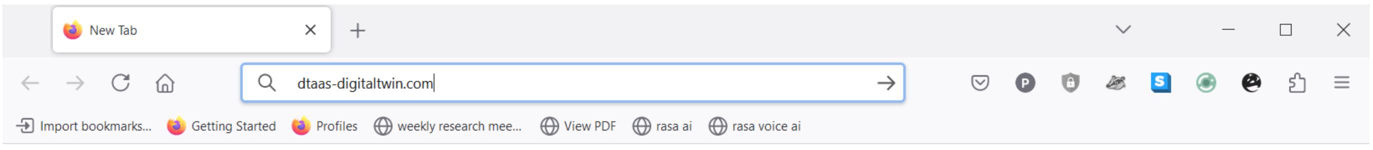
The users can publish and reuse the digital twin assets available on the platform. In addition, users can run their digital twins and make these live digital twins available as services to their clients. The clients need not be users of the DTaaS software installation.

2.3 DTaaS Website Screenshots

This page contains a screenshot driven preview of the website serving the DTaaS software platform.

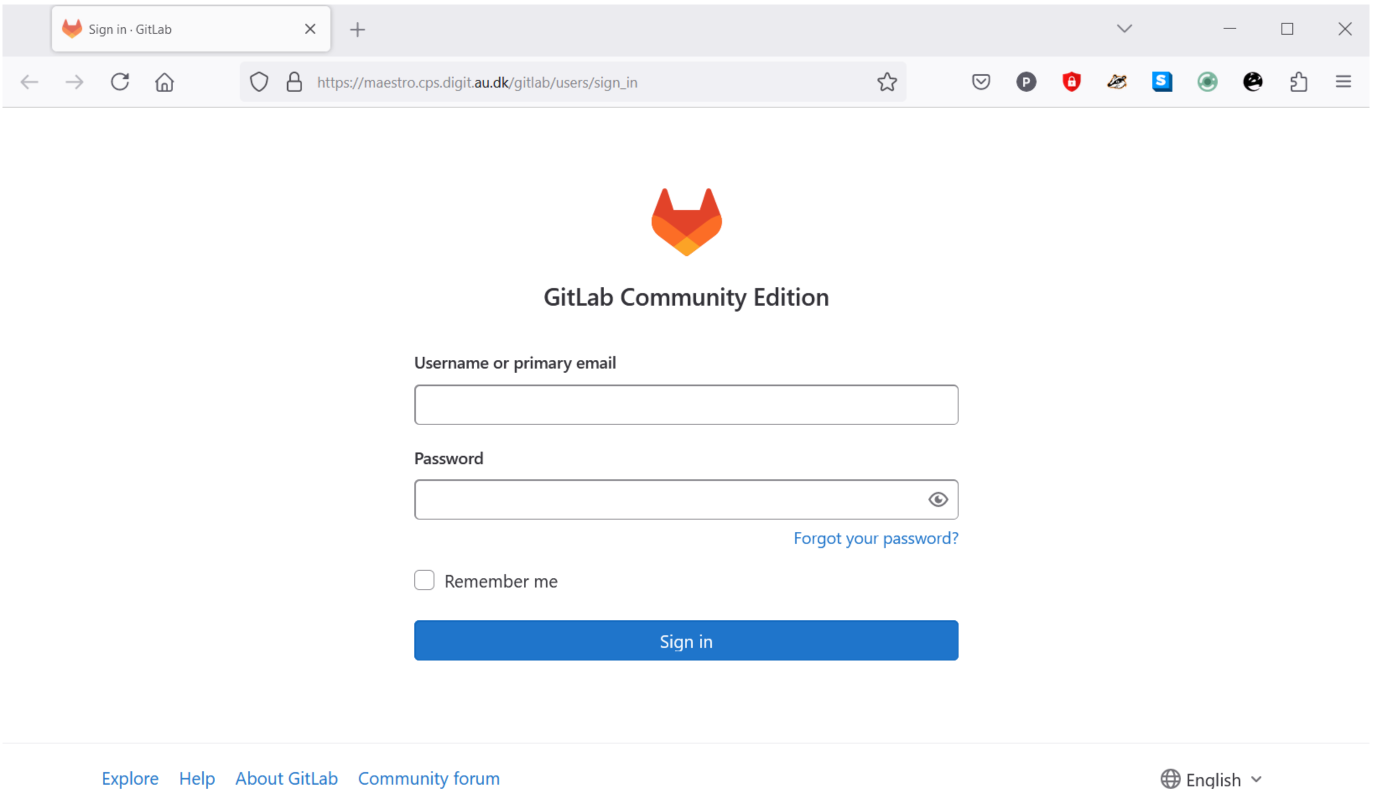
2.3.1 Visit the DTaaS Application

Start off by simply visiting the website of the DTaaS instance for which you are a user.



2.3.2 Redirected to Authorization Provider

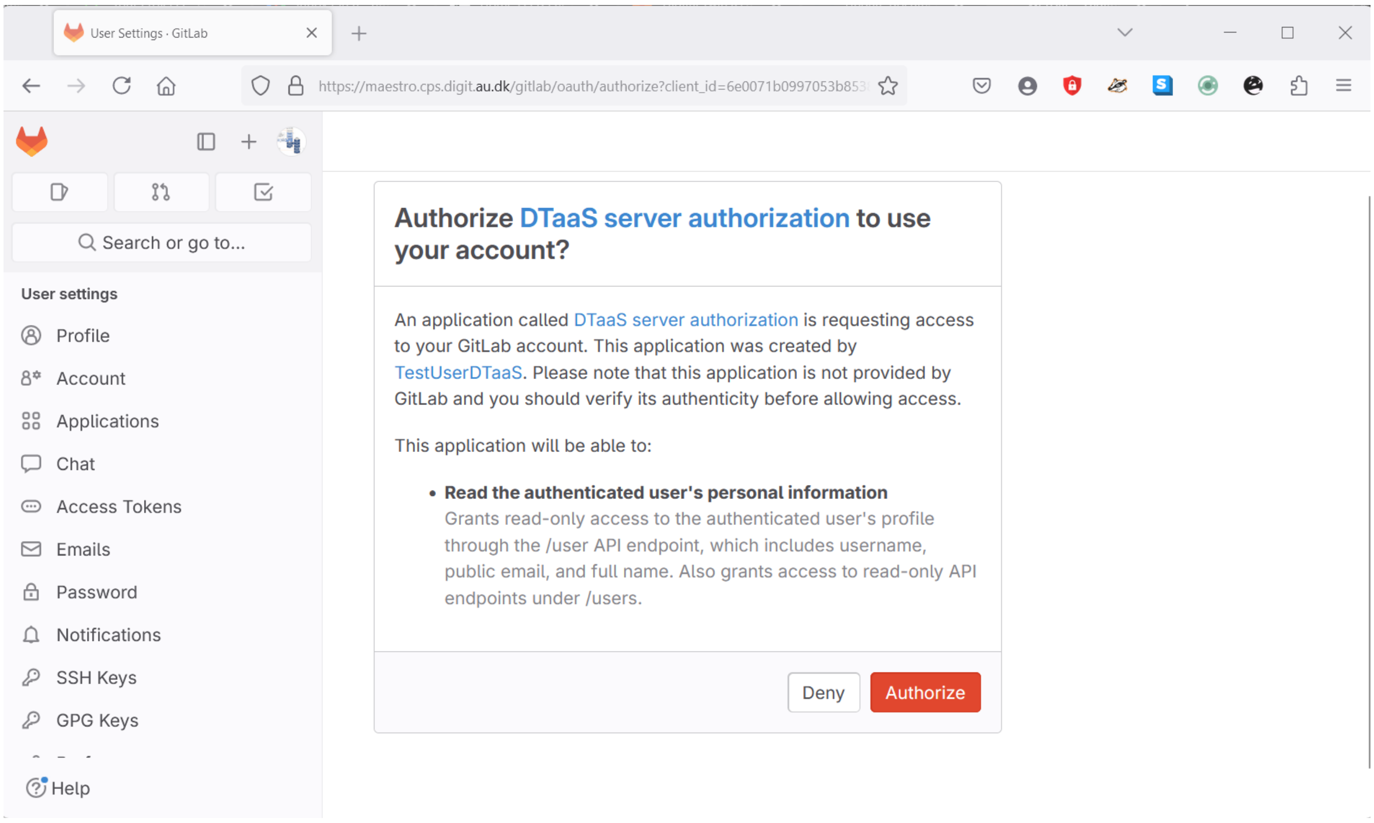
You will be redirected to the Gitlab Authorization for DTaaS.



Enter your email/username and password. If the email ID registered with DTaaS, is the same as your Google Login email ID, you can also opt to sign in using Google.

You will be redirected to the OAuth Application page.

2.3.3 Permit DTaaS Server to Use Gitlab



Authorize DTaaS server authorization to use your account?

An application called DTaaS server authorization is requesting access to your GitLab account. This application was created by TestUserDTaaS. Please note that this application is not provided by GitLab and you should verify its authenticity before allowing access.

This application will be able to:

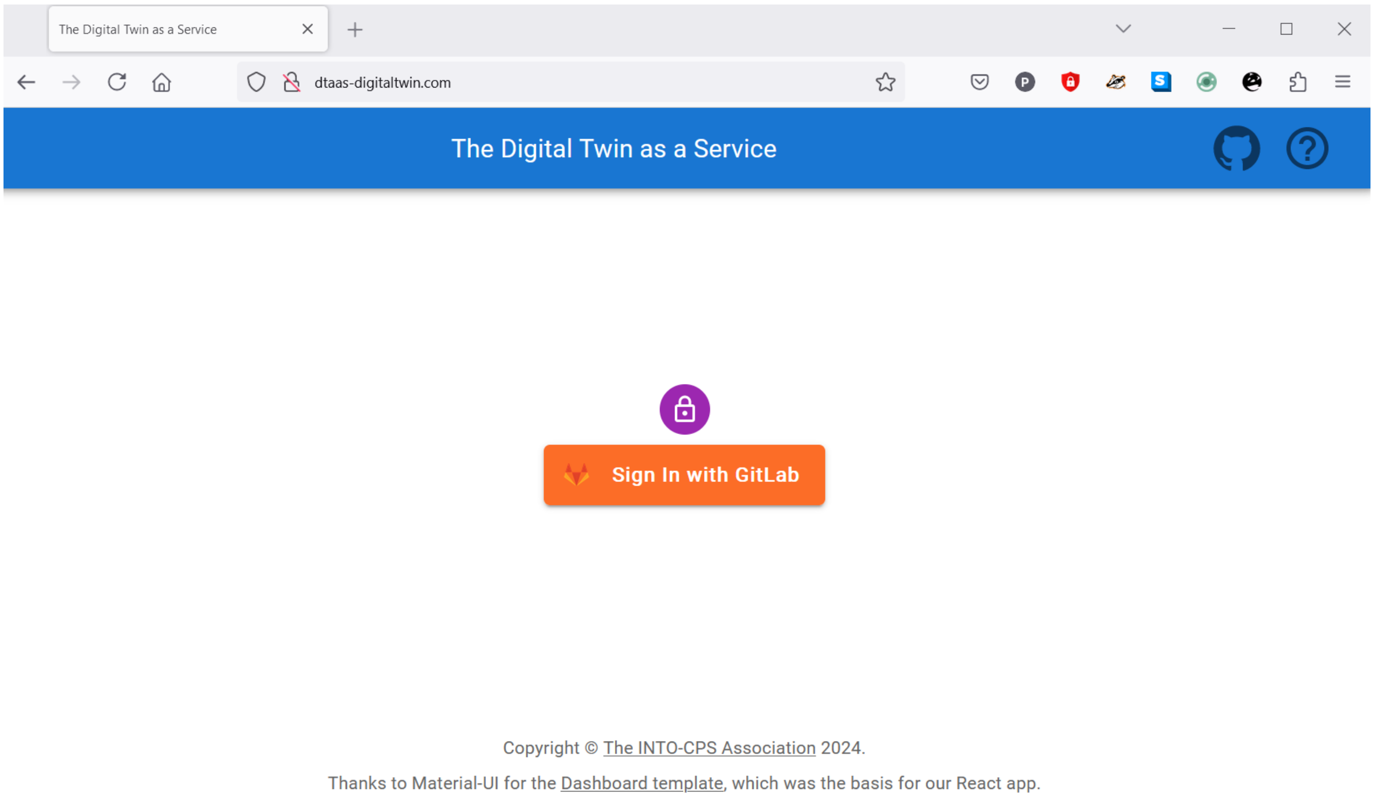
- **Read the authenticated user's personal information**
  Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.

Deny Authorize

Click on Authorize to allow the OAuth application to access the information connected to your Gitlab account. This is a necessary step.

You are now logged into the DTaaS server. You will be redirected to the login page of the DTaaS website.

The DTaaS website uses an additional layer of security - the third-party authorization protocol known as OAuth. This protocol provides secure access to a DTaaS installation if users have a working active accounts at the selected OAuth service provider. This also uses Gitlab as OAuth provider.



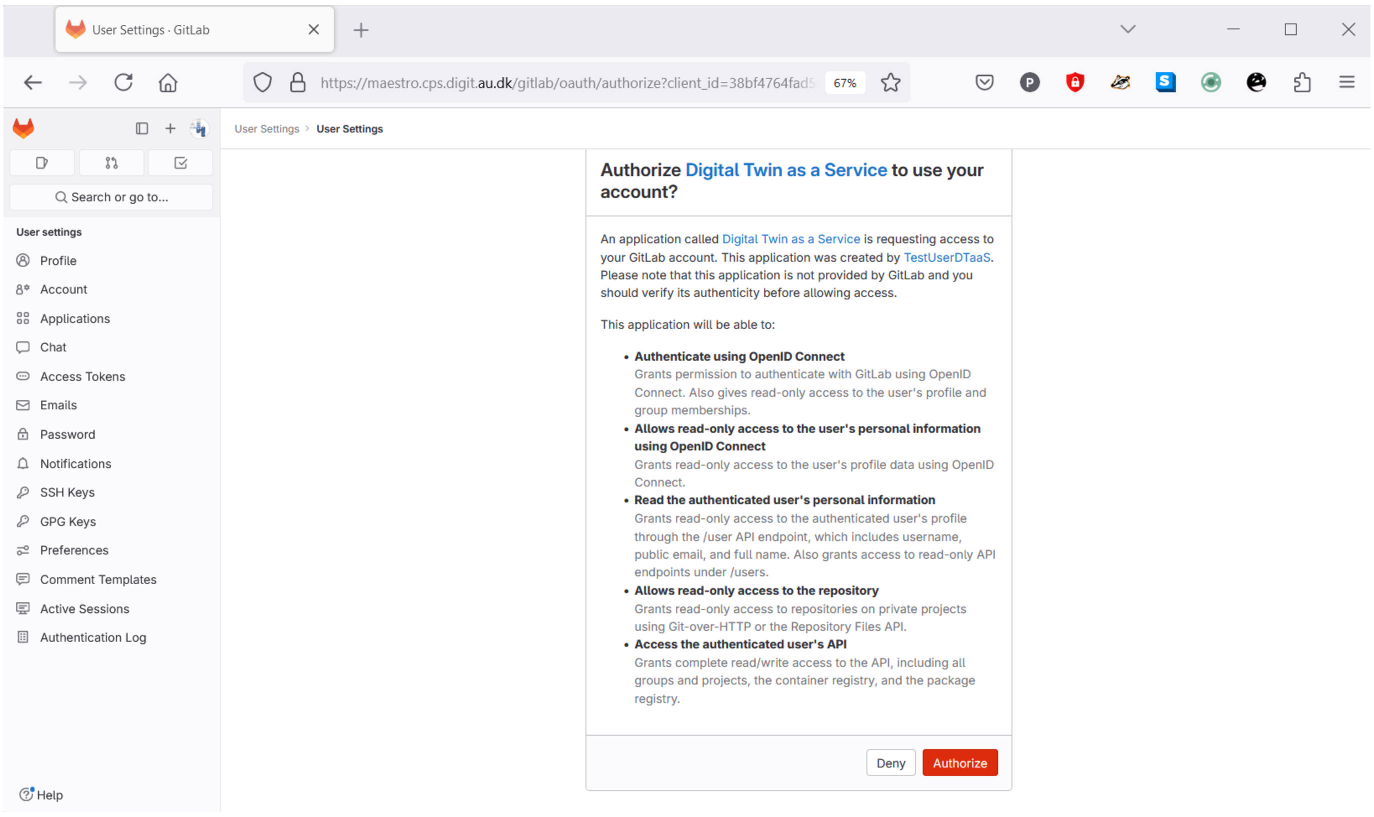
Copyright © The INTO-CPS Association 2024.

Thanks to Material-UI for the Dashboard template, which was the basis for our React app.

You can see the Gitlab signin button. A click on this button takes you to Gitlab instance providing authorization for DTaaS. You will not need to sign in to Gitlab again, unless you have explicitly logged out of your Gitlab account.

2.3.4 Permit DTaaS Website to Use Gitlab

The DTaaS website needs your permission to use your Gitlab account for authorization. Click on **Authorize** button.



Authorize Digital Twin as a Service to use your account?

An application called Digital Twin as a Service is requesting access to your GitLab account. This application was created by TestUserDTaaS. Please note that this application is not provided by GitLab and you should verify its authenticity before allowing access.

This application will be able to:

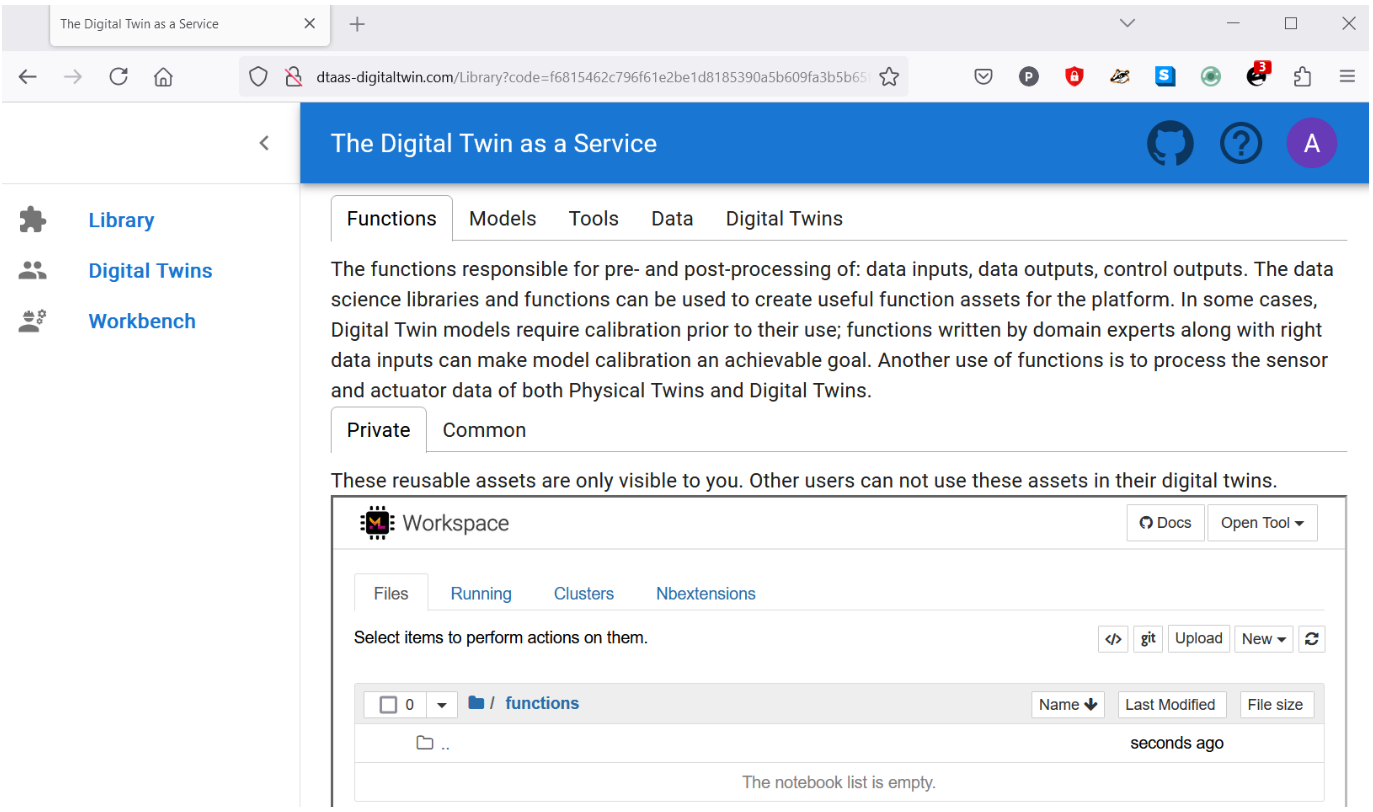
- **Authenticate using OpenID Connect**
  Grants permission to authenticate with GitLab using OpenID Connect. Also gives read-only access to the user's profile and group memberships.
- **Allows read-only access to the user's personal information using OpenID Connect**
  Grants read-only access to the user's profile data using OpenID Connect.
- **Read the authenticated user's personal information**
  Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
- **Allows read-only access to the repository**
  Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
- **Access the authenticated user's API**
  Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.

After successful authorization, you will be redirected to the **Library** page of the DTaaS website.

There are two icons on the top-right of the webpage. The hyperlink on **question mark icon** redirects to help page while the hyperlink on **github icon** redirects to github code repository.

2.3.5 Overview of menu items

The menu is hidden by default. Only the icons of menu items are visible. You can click on the ☰ icon in the top-left corner of the page to see the menu.



The functions responsible for pre- and post-processing of: data inputs, data outputs, control outputs. The data science libraries and functions can be used to create useful function assets for the platform. In some cases, Digital Twin models require calibration prior to their use; functions written by domain experts along with right data inputs can make model calibration an achievable goal. Another use of functions is to process the sensor and actuator data of both Physical Twins and Digital Twins.

These reusable assets are only visible to you. Other users can not use these assets in their digital twins.

Select items to perform actions on them.

The notebook list is empty.

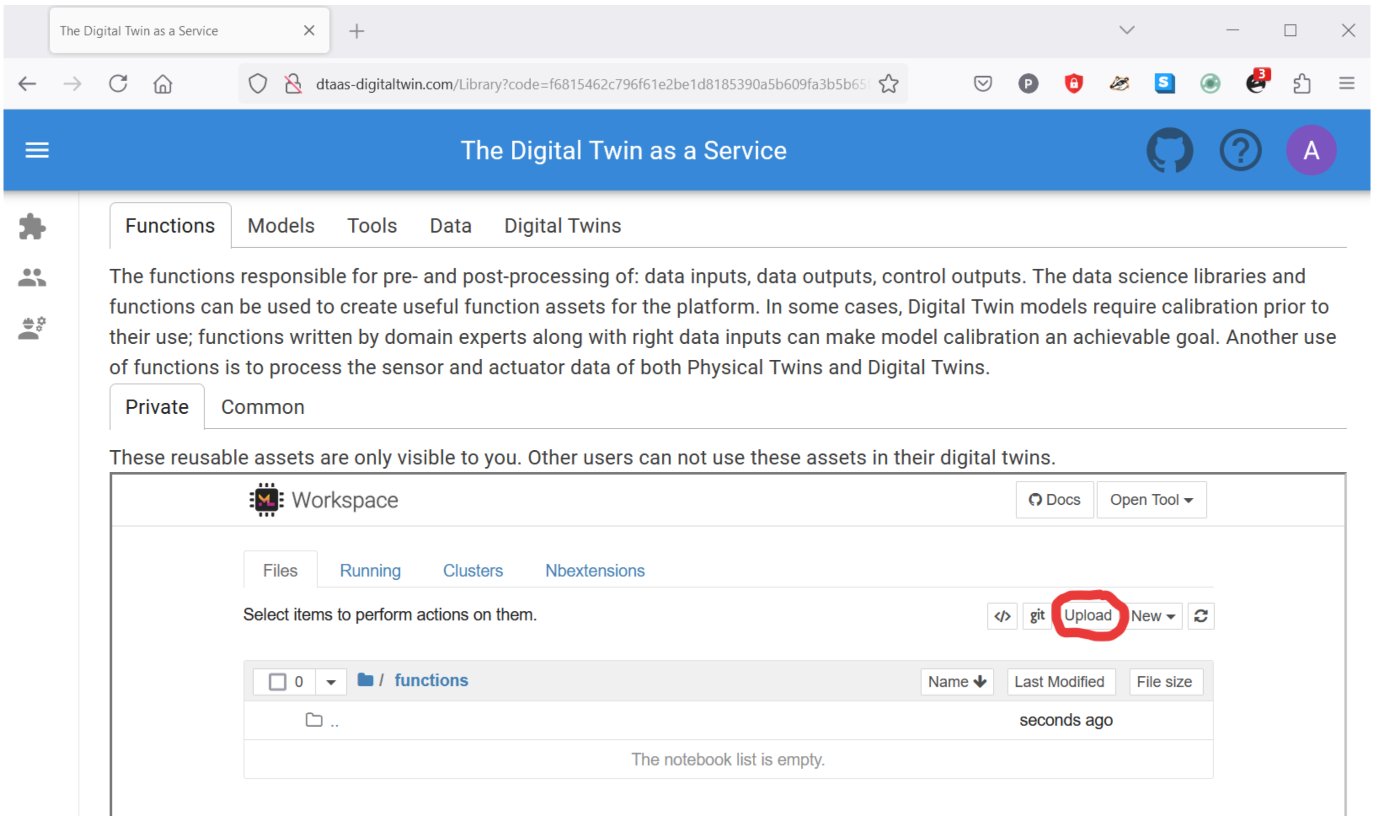
There are three menu items:

Library: for management of reusable library assets. You can upload, download, create and modify new files on this page.

Digital Twins: for management of digital twins. You are presented with the Jupyter Lab page from which you can run the digital twins.

Workbench: Not all digital twins can be managed within Jupyter Lab. You have more tools at your disposal on this page.

2.3.6 Library tabs and their help text



The functions responsible for pre- and post-processing of: data inputs, data outputs, control outputs. The data science libraries and functions can be used to create useful function assets for the platform. In some cases, Digital Twin models require calibration prior to their use; functions written by domain experts along with right data inputs can make model calibration an achievable goal. Another use of functions is to process the sensor and actuator data of both Physical Twins and Digital Twins.

These reusable assets are only visible to you. Other users can not use these assets in their digital twins.

Select items to perform actions on them.

The notebook list is empty.

You can see the file manager and five tabs above the library manager. Each tab provides help text to guide users in the use of different directories in their workspace.

Functions

The functions responsible for pre- and post-processing of: data inputs, data outputs, control outputs. The data science libraries and functions can be used to create useful function assets for the platform. In some cases, Digital Twin models require calibration prior to their use; functions written by domain experts along with right data inputs can make model calibration an achievable goal. Another use of functions is to process the sensor and actuator data of both Physical Twins and Digital Twins.

Data

The data sources and sinks available to a digital twins. Typical examples of data sources are sensor measurements from Physical Twins, and test data provided by manufacturers for calibration of models. Typical examples of data sinks are visualization software, external users and data storage services. There exist special outputs such as events, and commands which are akin to control outputs from a Digital Twin. These control outputs usually go to Physical Twins, but they can also go to another Digital Twin.

Models

The model assets are used to describe different aspects of Physical Twins and their environment, at different levels of abstraction. Therefore, it is possible to have multiple models for the same Physical Twin. For example, a flexible robot used in a car production plant may have structural model(s) which will be useful in tracking the wear and tear of parts. The same robot can have a behavioural model(s) describing the safety guarantees provided by the robot manufacturer. The same robot can also have a functional model(s) describing the part manufacturing capabilities of the robot.

Tools

The software tool assets are software used to create, evaluate and analyze models. These tools are executed on top of a computing platforms, i.e., an operating system, or virtual machines like Java virtual machine, or inside docker containers. The tools tend to be platform specific, making them less reusable than models. A tool can be packaged to run on a local or distributed virtual machine environments thus allowing selection of most suitable execution environment for a Digital Twin. Most models require tools to evaluate them in the context of data inputs. There exist cases where executable packages are run as binaries in a computing environment. Each of these packages are a pre-packaged combination of models and tools put together to create a ready to use Digital Twins.

Digital Twins

These are ready to use digital twins created by one or more users. These digital twins can be reconfigured later for specific use cases.

In addition to the five directories, there is also **common** directory in which five sub-directories exist. These sub-directories are: data, functions, models, tools and digital twins.

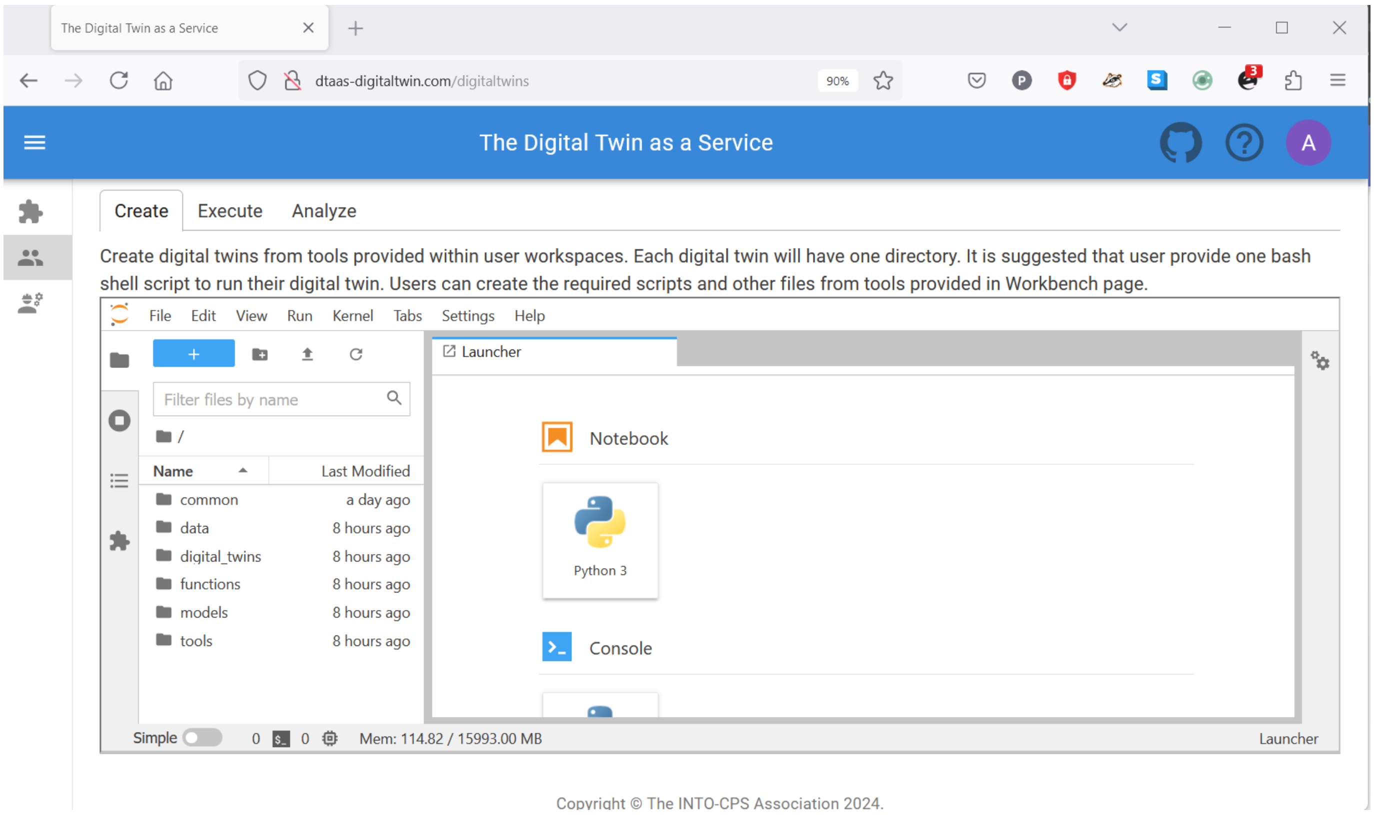
Common Assets

The common directory again has four sub-directories: - data - functions - functions - models - tools - digital twins The assets common to all users are placed in **common**.

The items used by more than one user are placed in **common**. The items in the **common** directory are available to all users. Further explanation of directory structure and placement of reusable assets within the the directory structure is in the assets page

The file manager is based on Jupyter notebook and all the tasks you can perform in the Jupyter Notebook can be undertaken here.

2.3.7 Digital Twins page



Create digital twins from tools provided within user workspaces. Each digital twin will have one directory. It is suggested that user provide one bash shell script to run their digital twin. Users can create the required scripts and other files from tools provided in Workbench page.

Name	Last Modified
common	a day ago
data	8 hours ago
digital_twins	8 hours ago
functions	8 hours ago
models	8 hours ago
tools	8 hours ago

Mem: 114.82 / 15993.00 MB

Copyright © The INTO-CPS Association 2024.

The digital twins page has three tabs and the central pane opens Jupyter lab. There are three tabs with helpful instructions on the suggested tasks you can undertake in the **Create - Execute - Analyze** life cycle phases of digital twin. You can see more explanation on the life cycle phases of digital twin.

Create

Create digital twins from tools provided within user workspaces. Each digital twin will have one directory. It is suggested that user provide one bash shell script to run their digital twin. Users can create the required scripts and other files from tools provided in Workbench page.

Execute

Digital twins are executed from within user workspaces. The given bash script gets executed from digital twin directory. Terminal-based digital twins can be executed from VSCode and graphical digital twins can be executed from VNC GUI. The results of execution can be placed in the data directory.

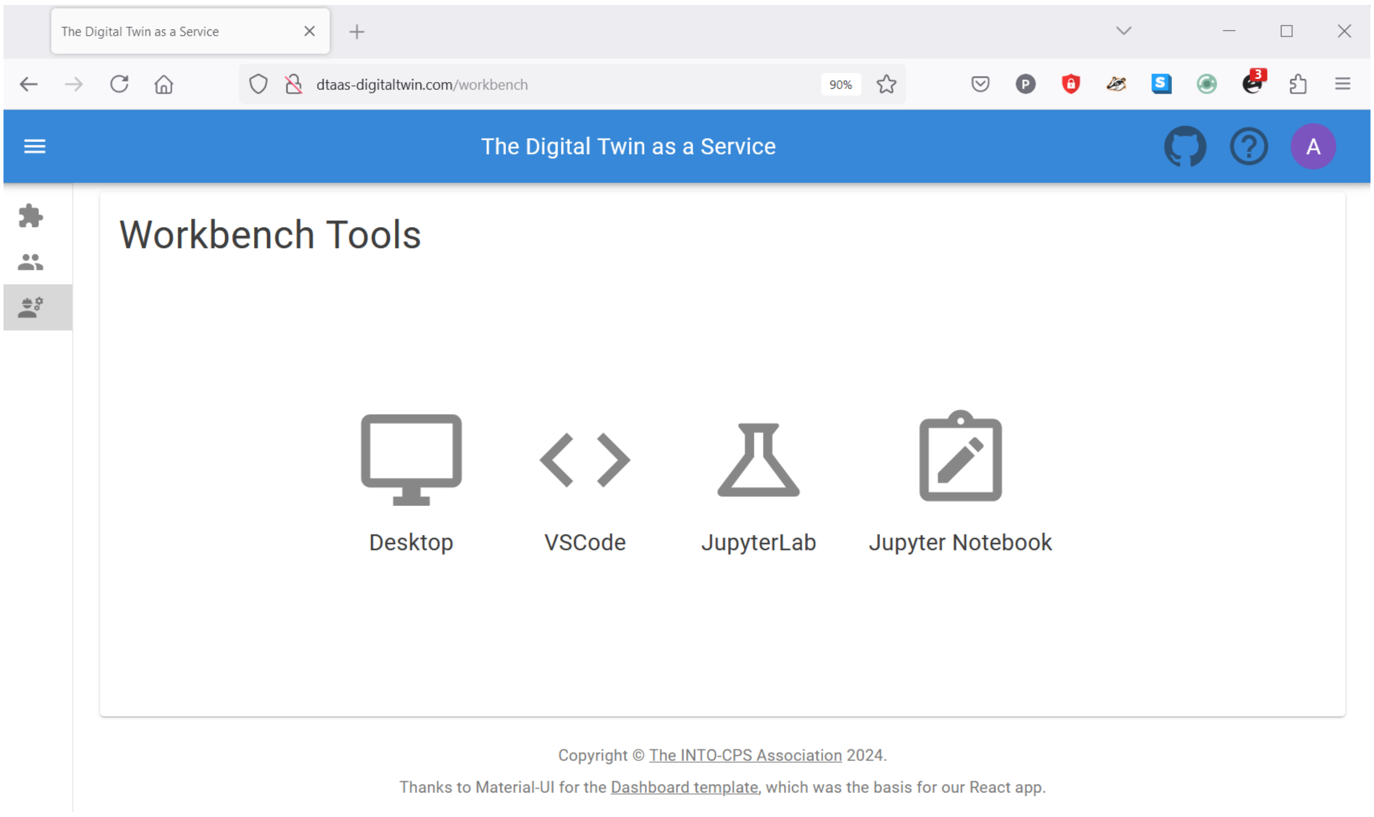
Analyze

The analysis of digital twins requires running of digital twin script from user workspace. The execution results placed within data directory are processed by analysis scripts and results are placed back in the data directory. These scripts can either be executed from VSCode and graphical results or can be executed from VNC GUI. The analysis of digital twins requires running of digital twin script from user workspace. The execution results placed within data directory are processed by analysis scripts and results are placed back in the data directory. These scripts can either be executed from VSCode and graphical results or can be executed from VNC GUI.

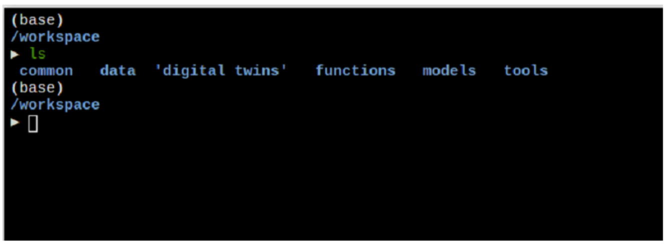
The reusable assets (files) seen in the file manager are available in the Jupyter Lab. In addition, there is a git plugin installed in the Jupyter Lab using which you can link your files with the external git repositories.

2.3.8 Workbench

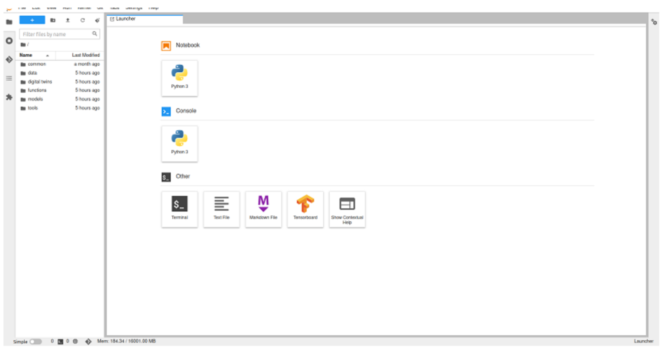
The **workbench** page provides links to four integrated tools.

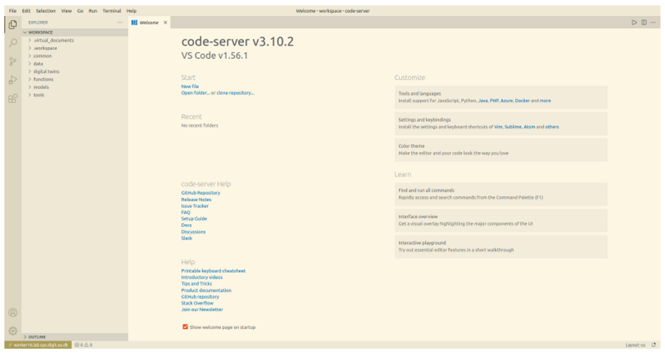

The Digital Twin as a Service
dtaas-digitaltwin.com/workbench
90%
The Digital Twin as a Service
A
Workbench Tools
Desktop
VSCode
JupyterLab
Jupyter Notebook
Copyright © The INTO-CPS Association 2024.
Thanks to Material-UI for the Dashboard template, which was the basis for our React app.

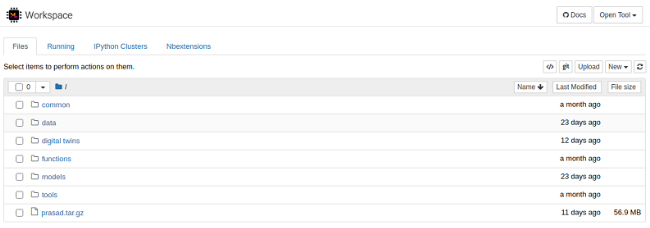
The hyperlinks open in new browser tab. The screenshots of pages opened in new browser are:



```
(base)
/workspace
▶ ls
 common   data  'digital twins'   functions   models   tools
(base)
/workspace
▶ ▯
```





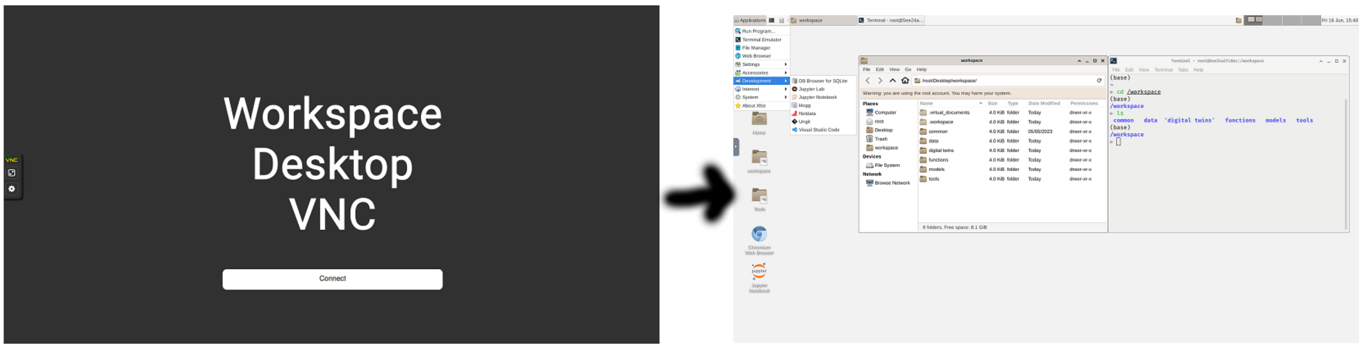


Workspace

Files | Running | IPython Clusters | Nbextensions

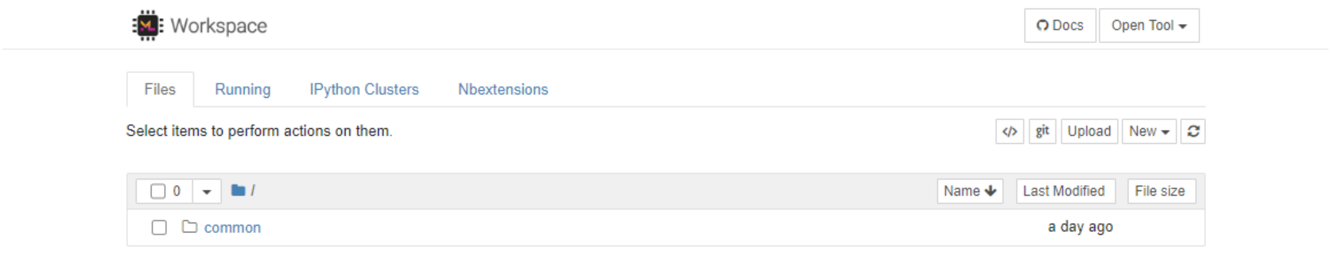
Select items to perform actions on them.

Name	Last Modified	File size
common	a month ago	
data	23 days ago	
digital twins	12 days ago	
functions	a month ago	
models	23 days ago	
tools	a month ago	
prasad.tar.gz	11 days ago	56.9 MB



Terminal

The Terminal hyperlink does not exist on workbench page. If you want terminal. Please use the tools dropdown in the Jupyter Notebook.



Workspace

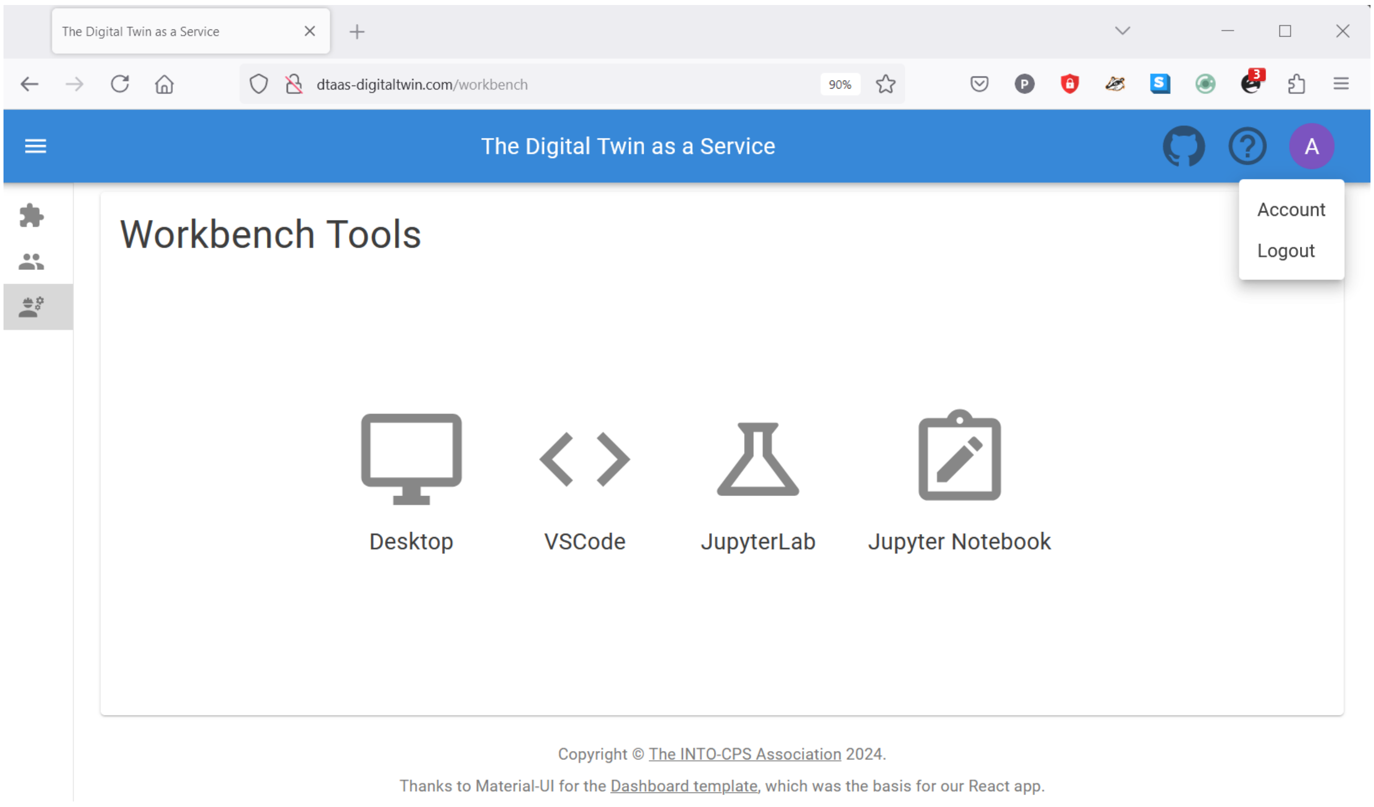
Docs | Open Tool

Files | Running | IPython Clusters | Nbextensions

Select items to perform actions on them.

Name	Last Modified	File size
common	a day ago	

2.3.9 Finally logout


The Digital Twin as a Service
dtaas-digitaltwin.com/workbench
90%
The Digital Twin as a Service
A
Account
Logout
Workbench Tools
Desktop
VSCode
JupyterLab
Jupyter Notebook
Copyright © The INTO-CPS Association 2024.
Thanks to Material-UI for the Dashboard template, which was the basis for our React app.

You have to close the browser in order to completely exit the DTaaS software platform.

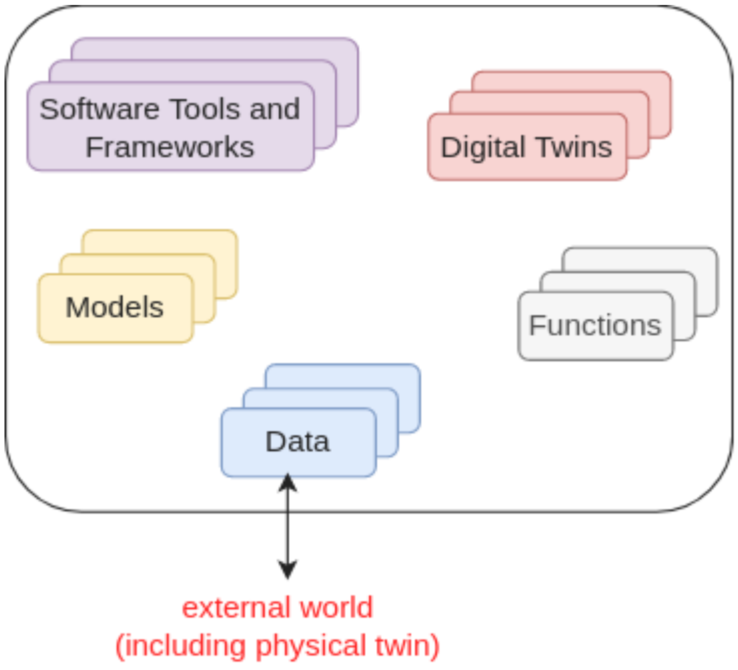
2.4 Reusable Assets

2.4.1 Reusable Assets

The reusability of digital twin assets makes it easy for users to work with the digital twins. The reusability of assets is a fundamental feature of the platform.

Kinds of Reusable Assets

The DTaaS software categorizes all the reusable library assets into six categories:



DATA

The data sources and sinks available to a digital twins. Typical examples of data sources are sensor measurements from Physical Twins, and test data provided by manufacturers for calibration of models. Typical examples of data sinks are visualization software, external users and data storage services. There exist special outputs such as events, and commands which are akin to control outputs from a Digital Twin. These control outputs usually go to Physical Twins, but they can also go to another Digital Twin.

MODELS

The model assets are used to describe different aspects of Physical Twins and their environment, at different levels of abstraction. Therefore, it is possible to have multiple models for the same Physical Twin. For example, a flexible robot used in a car production plant may have structural model(s) which will be useful in tracking the wear and tear of parts. The same robot can have a behavioural model(s) describing the safety guarantees provided by the robot manufacturer. The same robot can also have a functional model(s) describing the part manufacturing capabilities of the robot.

TOOLS

The software tool assets are software used to create, evaluate and analyze models. These tools are executed on top of a computing platforms, i.e., an operating system, or virtual machines like Java virtual machine, or inside docker containers. The tools tend to be platform specific, making them less reusable than models. A tool can be packaged to run on a local or distributed virtual machine environments thus allowing selection of most suitable execution environment for a Digital Twin. Most models require tools to evaluate them in the context of data inputs. There exist cases where executable packages are run as binaries in a computing environment. Each of these packages are a pre-packaged combination of models and tools put together to create a ready to use Digital Twins.

FUNCTIONS

The functions responsible for pre- and post-processing of: data inputs, data outputs, control outputs. The data science libraries and functions can be used to create useful function assets for the platform. In some cases, Digital Twin models require calibration prior to their use; functions written by domain experts along with right data inputs can make model calibration an achievable goal. Another use of functions is to process the sensor and actuator data of both Physical Twins and Digital Twins.

DIGITAL TWINS

These are ready to use digital twins created by one or more users. These digital twins can be reconfigured later for specific use cases.

File System Structure

Each user has their assets put into five different directories named above. In addition, there will also be common library assets that all users have access to. A simplified example of the structure is as follows:

```
workspace/
  data/
    data1/ (ex: sensor)
      filename (ex: sensor.csv)
      README.md
    data2/ (ex: turbine)
      README.md (remote source; no local file)
    ...
  digital_twins/
    digital_twin-1/ (ex: incubator)
      code and config
      README.md (usage instructions)
    digital_twin-2/ (ex: mass spring damper)
      code and config
      README.md (usage instructions)
    digital_twin-3/ (ex: model swap)
      code and config
      README.md (usage instructions)
    ...
  functions/
    function1/ (ex: graphs)
      filename (ex: graphs.py)
      README.md
    function2/ (ex: statistics)
      filename (ex: statistics.py)
      README.md
    ...
  models/
    model1/ (ex: spring)
      filename (ex: spring.fmu)
      README.md
    model2/ (ex: building)
      filename (ex: building.skp)
      README.md
    model3/ (ex: rabbitmq)
      filename (ex: rabbitmq.fmu)
      README.md
    ...
  tools/
    tool1/ (ex: maestro)
      filename (ex: maestro.jar)
      README.md
    ...
  common/
    data/
    functions/
    models/
    tools/
```

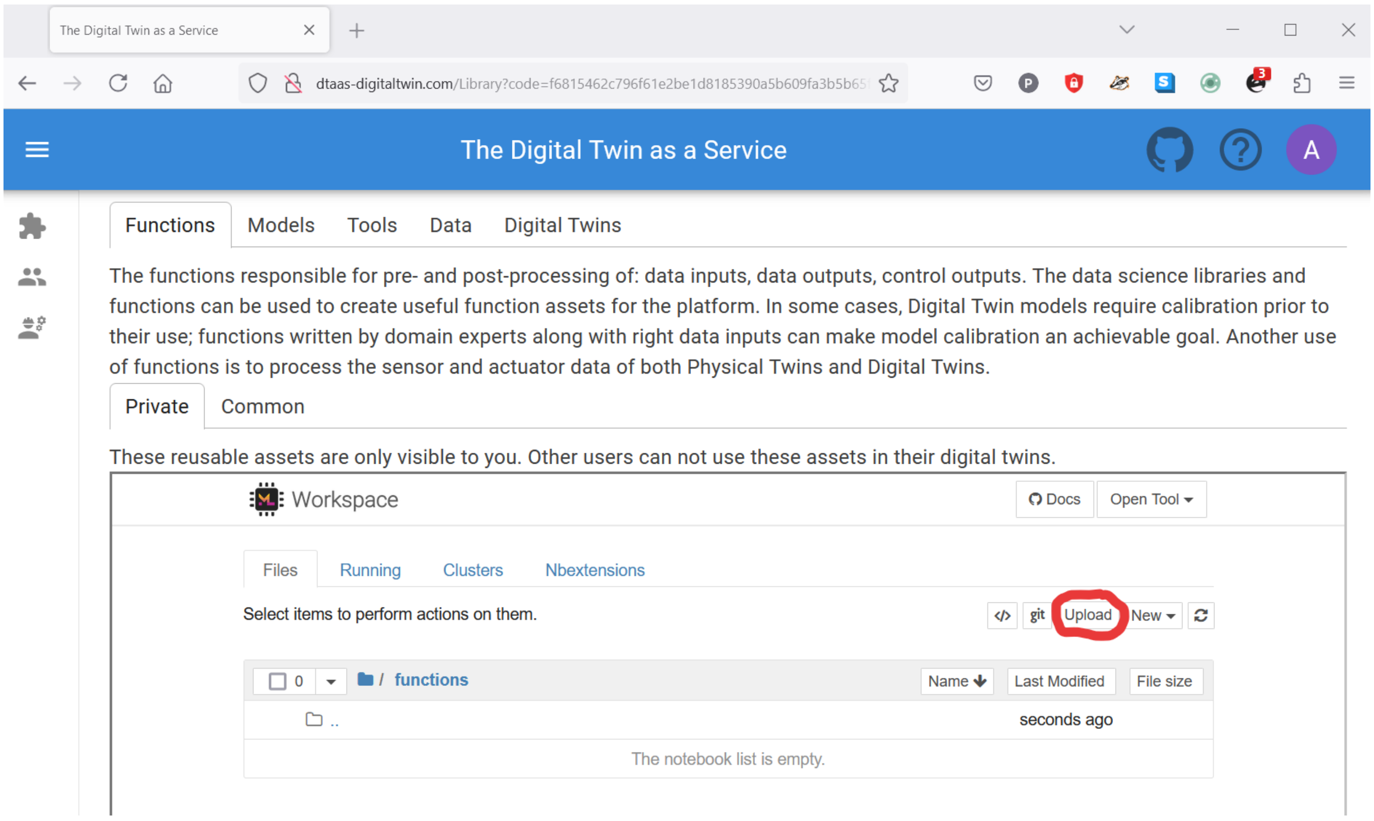
Tip

The DTaaS is agnostic to the format of your assets. The only requirement is that they are files which can be uploaded on the Library page. Any directories can be compressed as one file and uploaded. You can decompress the file into a directory from a Terminal or xfce Desktop available on the Workbench page.

A recommended file system structure for storing assets is also available in DTaaS examples.

Upload Assets

Users can upload assets into their workspace using Library page of the website.



The functions responsible for pre- and post-processing of: data inputs, data outputs, control outputs. The data science libraries and functions can be used to create useful function assets for the platform. In some cases, Digital Twin models require calibration prior to their use; functions written by domain experts along with right data inputs can make model calibration an achievable goal. Another use of functions is to process the sensor and actuator data of both Physical Twins and Digital Twins.

These reusable assets are only visible to you. Other users can not use these assets in their digital twins.

Select items to perform actions on them.

The notebook list is empty.

You can go into a directory and click on the **upload** button to upload a file or a directory into your workspace. This asset is then available in all the workbench tools you can use. You can also create new assets on the page by clicking on **new** drop down menu. This is a simple web interface which allows you to create text-based files. You need to upload other files using **upload** button.

The user workbench has the following services:

- Jupyter Notebook and Lab
- VS Code
- XFCE Desktop Environment available via VNC
- Terminal

Users can also bring their DT assets into user workspaces from outside using any of the above mentioned services. The developers using *git* repositories can clone from and push to remote git servers. Users can also use widely used file transfer protocols such as FTP, and SCP to bring the required DT assets into their workspaces.

2.4.2 Library Microservice

The lib microservice is responsible for handling and serving the contents of library assets of the DTaaS platform. It provides API endpoints for clients to query, and fetch these assets.

This document provides instructions for using the library microservice.

Please see assets for a suggested storage conventions of your library assets.

Once the assets are stored in the library, they become available in user workspace.

Application Programming Interface (API)

The lib microservice application provides services at two end points:

GraphQL API Endpoint: `http://foo.com/lib`

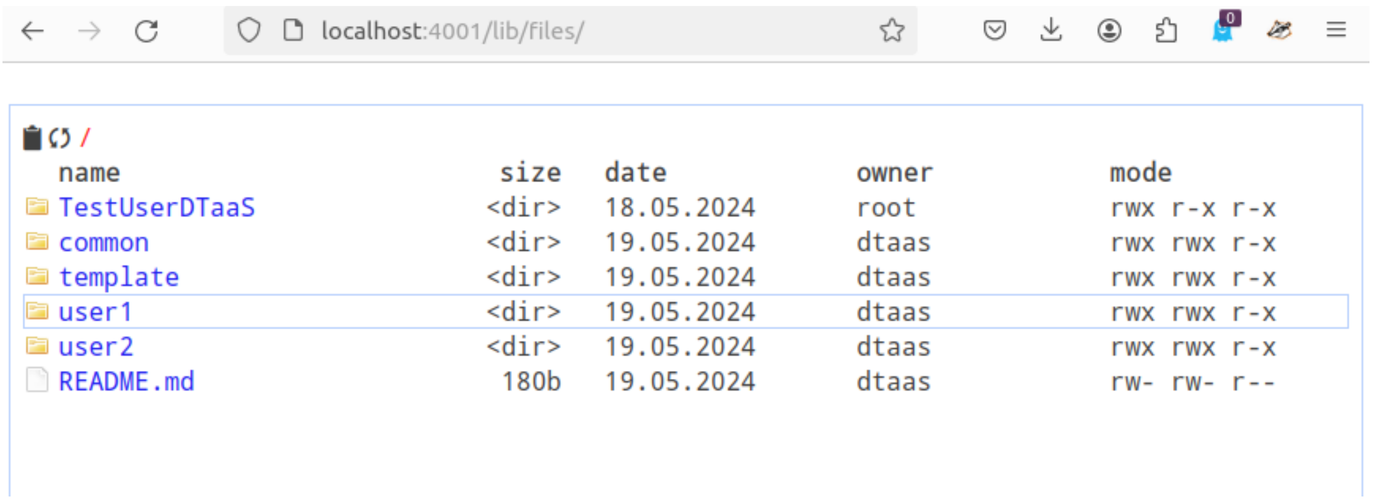
HTTP Endpoint: `http://foo.com/lib/files`

HTTP PROTOCOL

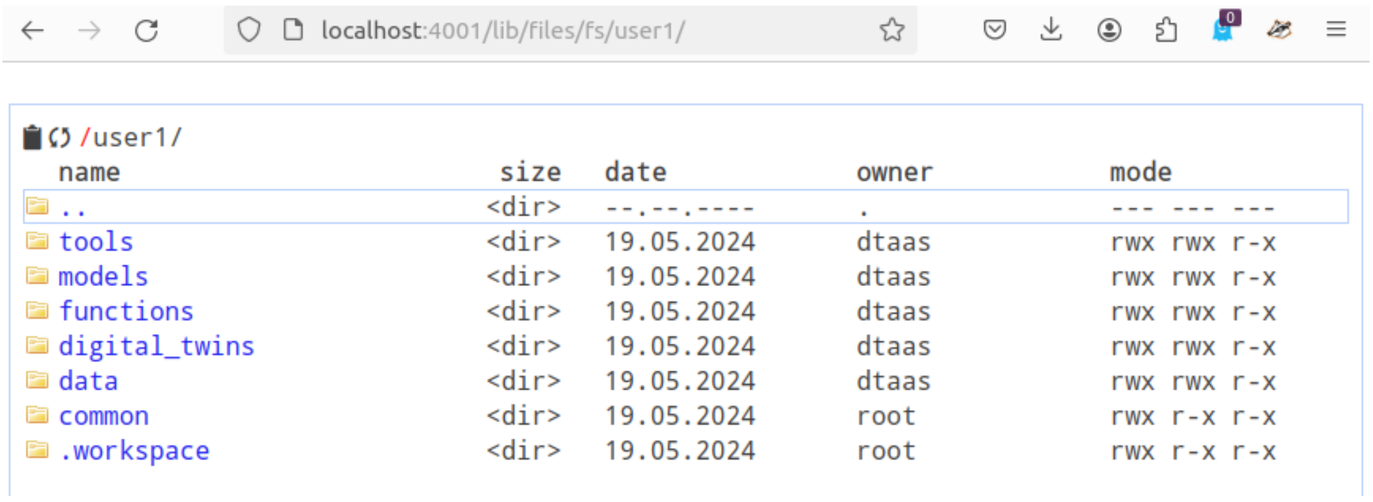
Endpoint: `localhost:PORT/lib/files`

This option needs to be enabled with `-H http.json` flag. The regular file upload and download options become available.

Here are sample screenshots.



```
/
name                 size   date        owner   mode
TestUserDTaaS        <dir>  18.05.2024  root    rwx r-x r-x
common               <dir>  19.05.2024  dtaas   rwx rwx r-x
template             <dir>  19.05.2024  dtaas   rwx rwx r-x
user1                <dir>  19.05.2024  dtaas   rwx rwx r-x
user2                <dir>  19.05.2024  dtaas   rwx rwx r-x
README.md            180b   19.05.2024  dtaas   rw- rw- r--
```



```
/user1/
name                 size   date        owner   mode
..                   <dir>  --.--.----  .       --- --- ---
tools                <dir>  19.05.2024  dtaas   rwx rwx r-x
models               <dir>  19.05.2024  dtaas   rwx rwx r-x
functions            <dir>  19.05.2024  dtaas   rwx rwx r-x
digital_twins        <dir>  19.05.2024  dtaas   rwx rwx r-x
data                 <dir>  19.05.2024  dtaas   rwx rwx r-x
common               <dir>  19.05.2024  root    rwx r-x r-x
.workspace           <dir>  19.05.2024  root    rwx r-x r-x
```

GRAPHQL PROTOCOL

Endpoint: `localhost:PORT/lib`

The `http://foo.com/lib` URL opens a graphql playground.

You can check the query schema and try sample queries here. The graphql queries need to be sent as HTTP POST requests and get responses.

The library microservice services two API calls:

- Provide a list of contents for a directory
- Fetch a file from the available files

The API calls are accepted over GraphQL and HTTP API end points. The format of the accepted queries are:

PROVIDE LIST OF CONTENTS FOR A DIRECTORY

To retrieve a list of files in a directory, use the following GraphQL query.

Replace `path` with the desired directory path.

send requests to: https://foo.com/lib

GraphQL Query **GraphQL Response** **HTTP Request** **HTTP Response**

```
query {
  listDirectory(path: "user1") {
    repository {
      tree {
        blobs {
          edges {
            node {
              name
              type
            }
          }
        }
        trees {
          edges {
            node {
              name
              type
            }
          }
        }
      }
    }
  }
}
```

```
{
  "data": {
    "listDirectory": {
      "repository": {
        "tree": {
          "blobs": {
            "edges": []
          },
          "trees": {
            "edges": [
              {
                "node": {
                  "name": "common",
                  "type": "tree"
                }
              },
              {
                "node": {
                  "name": "data",
                  "type": "tree"
                }
              },
              {
                "node": {
                  "name": "digital twins",
                  "type": "tree"
                }
              },
              {
                "node": {
                  "name": "functions",
                  "type": "tree"
                }
              },
              {
                "node": {
                  "name": "models",
                  "type": "tree"
                }
              },
              {
                "node": {
                  "name": "tools",
                  "type": "tree"
                }
              }
            ]
          }
        }
      }
    }
  }
}
```

```
POST /lib HTTP/1.1
Host: foo.com
Content-Type: application/json
Content-Length: 388

{
  "query":"query {\n  listDirectory(path: \"user1\") {\n    repository {\n      tree {\n        blobs {\n          edges {\n            node {\n              name\n              type\n            }\n          }\n        }\n        trees {\n          edges {\n            node {\n              name\n              type\n            }\n          }\n        }\n      }\n    }\n  }\n}"
}
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Connection: close
Content-Length: 306
Content-Type: application/json; charset=utf-8
Date: Tue, 26 Sep 2023 20:26:49 GMT
X-Powered-By: Express
{"data":{"listDirectory":{"repository":{"tree":{"blobs":{"edges":[]},"trees":{"edges":[{"node":{"name":"data","type":"tree"}},{"node":{"name":"digital twins","type":"tree"}},{"node":{"name":"functions","type":"tree"}},{"node":{"name":"models","type":"tree"}},{"node":{"name":"tools","type":"tree"}}]}}}}}}
```