

Introduction

Prerequisites:

Install VS code : <https://code.visualstudio.com/>

NodeJS : <https://nodejs.org/en/download>

Git : <https://git-scm.com/>

Automated steps

The steps can be automatically executed with the following demo code:

1. Create a folder for this Summer School session
2. Navigate to the folder (e.g. `cd C:\Documents\SummerSchool\Lecture_federation`)

```
cd C:\Documents\SummerSchool\Lecture_federation
```

3. Clone the following repository / SSoLDAC branch

```
git clone -b SSoLDAC2023 https://github.com/LBD-Hackers/consolid-demo.git demo
```

4. Navigate to the demo folder

```
cd demo
```

5. Install the necessary packages

```
npm install
```

Part I : initialisation

Download and install the Community Solid Server

1. Clone the Solid Server from Github (alternatively: download as ZIP)

```
git clone https://github.com/CommunitySolidServer/CommunitySolidServer solid
```

2. Navigate to the directory where the Solid Server was cloned (e.g. solid)

```
cd solid
```

3. Install the necessary packages

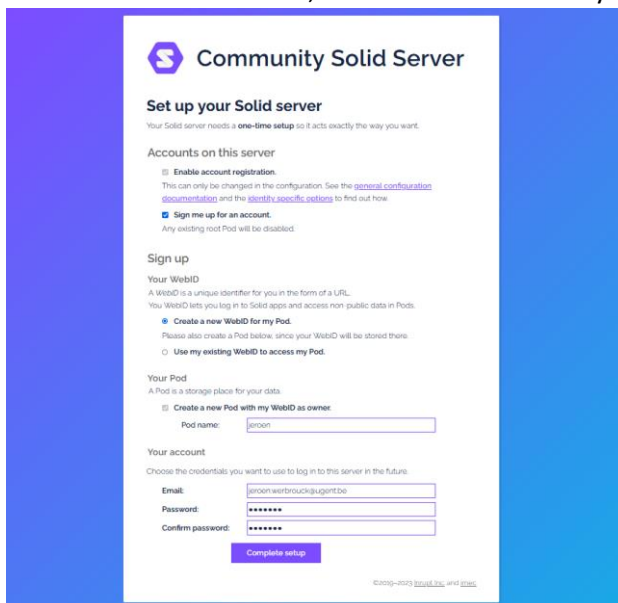
```
npm install
```

First run & account creation

1. Start the Solid server

```
npm run start:file
```

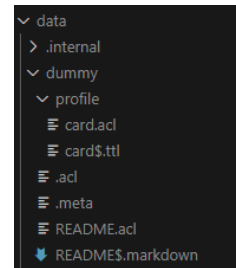
2. Navigate to <http://localhost:3000>. You will be redirected to <http://localhost:3000/setup>.
3. Check “Sign me up for an account”. The first Pod will be a dummy Pod, we will automatically create the Pods of the AEC stakeholders later.
4. Enter a name for your Pod (e.g. “dummy”)
5. Enter account credentials, which will be linked to your WebID. It should look like this:



6. Click “complete setup”
7. Open the folder with the Solid Server in VS code

As we chose the startup option “npm run start:file” earlier, our Pods and their data will be preserved in the file system. A new directory has been created to organise the data in the Pods, called “/data”.

The Solid Server will mirror the content of the corresponding Pod folders corresponding to the Linked Data Platform (LDP) specification. This means that all resources will be accessible via HTTP URLs as well. The URLs contain the parent folders until the root of the Pod. For example, after initialisation every Pod contains already one subfolder, called “profile”, containing the “card”. This is the resource linked to the URL of the WebID of the Pod owner.



For example, in the situation depicted in the image above, `data/dummy/profile/card$.ttl` will be hosted at <http://localhost:3000/dummy/profile/card>. The access control rules which state that the public has READ access and the owner has READ, WRITE and CONTROL access are defined in the `card.acl` resource

! The `$.`(extension) means that the extension will be omitted in the URL.

! CONTROL access means that a user has the rights to edit the access control document

Create Project Pods

To automatically execute the steps of this part, execute the script “`createAccounts.js`” of the ConSolid demo repository (see page 1). Note that at least one Pod must have been created already as the first Pod creation also creates some essential server resources for the first time. We did that with the dummy Pod.

With this step, 3 Pods will be created and populated with Project files, one for the architect, one for the MEP engineer, and one for the owner. We’ll create them on the same Solid server, but of course in reality they may be hosted by different Pod providers.

1. Execute the following commands in a terminal.

```
cd .\src\  
node .\createAccounts.js
```

This script also registers a “Project Registry” in the WebID card of each of the Pods. This will allow us later to discover and filter the projects on a Pod. The project registry (here called “`myProjects`”) is also created as a TTL file describing a DCAT Catalog.

```
<#me> <https://w3id.org/consolid#hasProjectCatalogue> </architect/myProjects>.
```

The input data for this script is a JSON file in the demo repository called “`setup.json`”.

Create project

This step will create a project catalogue for the project on each of the Pods and some sample datasets. The sample datasets will be registered in the project catalogue – now they can be discovered as part of the project. In DCAT, ‘datasets’ refer to the metadata of a specific resource. The real resource is called a ‘distribution’ of this dataset. The demo project contains the following distributions:

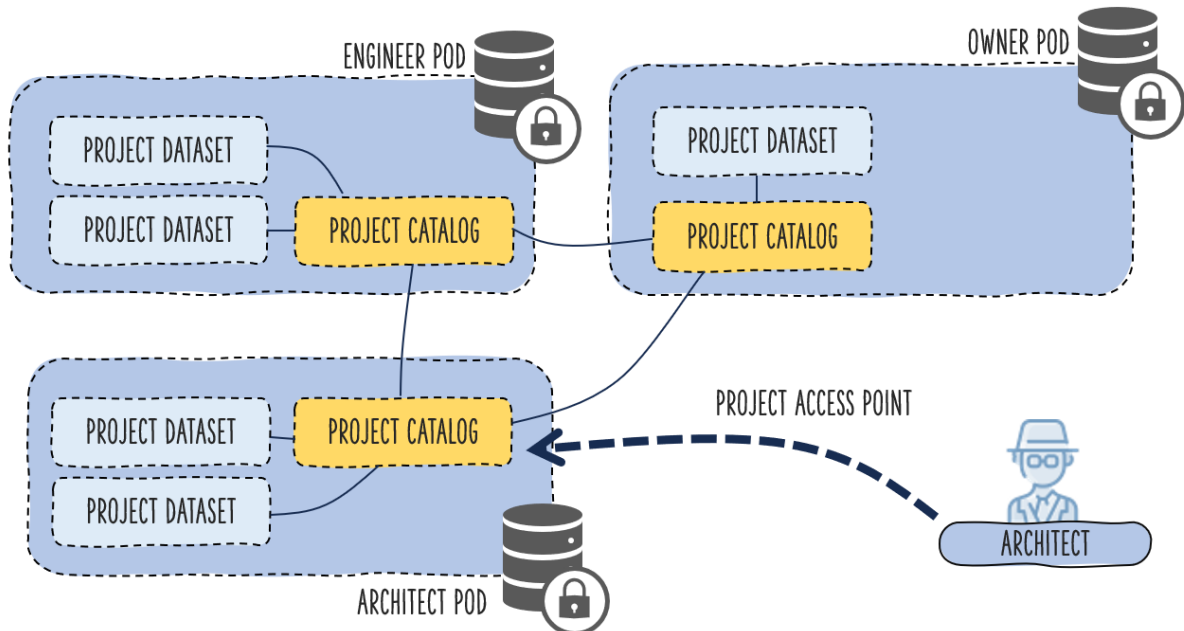
- Architect:
 - o TTL semantics of the Duplex architectural model
 - o glTF geometry of the Duplex architectural model
- MEP
 - o TTL semantics of the Duplex MEP model
 - o glTF geometry of the Duplex MEP model
- Owner:
 - o A JPEG image of their house being damaged

The media types of these distributions are included in the metadata. We will later in this tutorial use this to find specific project datasets.

1. Execute the following command (while in the repository of the demo):

```
node .\createProject.js
```

The structure of the interconnected catalogues on the stakeholder Pods will look like this:



LEGENDA

\\ = dcat:dataset

A dcat:Catalog points to its dataset with the property *dcat:dataset*. Since instances of dcat:Catalog are also instances of dcat:Dataset, the same property can be used for a catalogue to aggregate another catalogue. Recursively following all *dcat:dataset* properties from one project access point, we can thus discover all the datasets in a project, using this simple query:

```
SELECT * WHERE {  
  # the "+" sign means that a chain is created of this property  
  <url-of-the-access-point-catalogue> dcat:dataset+ ?dataset .  
  ?dataset dcat:distribution ?distribution .  
}
```

Variants of this query can be used to filter dataset metadata.

Querying the federated project

Now execute the following script in the demo repository

```
cd .\src\  
node .\queries\findMyProjects.js
```

This script will go to a given WebID and discover the projects they are participating in, using the following query, executed with Comunica/link-traversal:

```
SELECT DISTINCT * WHERE {  
  ?s <https://w3id.org/consolid#hasProjectCatalogue> ?catalogue .  
  ?catalogue <http://www.w3.org/ns/dcat#dataset> ?dataset .  
}
```

A second script will propagate through the catalogs, searching for all the files which are of glTF format. The result from the previous query (i.e. the project access point URL) can be used as input for this query

```
cd .\src\  
node .\queries\findProjectResources.js
```

```
SELECT DISTINCT * WHERE {  
  
  #the "+" sign means that a chain is created of this property  
  <url-of-project-access-point> dcat:dataset+ ?dataset .  
  ?dataset dcat:distribution ?distribution .  
  
  ?distribution dcat:mediaType <https://www.iana.org/assignments/media-  
types/model/gltf+json> .  
}
```