Demo Server Handbook

This guide should enable the user to set u a running Data-Ware-House (DWH) for the test of Clinical Document Architecture (CDAs) containing patient information. The guide is based on Docker, which is a virtualization layer on top of a operation system (OS). It is available for all major operating systems including Windows, Mac OS and LInux based distributions.

Prerequisites: Docker and Docker Compose

for the installation of Docker and Docker compose we refer to the official documentation from Docker:

- Docker engine
- Docker Compose

> 💡 Docker is the virtualization layer on top of your operating system. It allows the installation of so called containers, which include their own operating system and are not able to interfere or connect to your host operating system besides explicit configuration allows it. Docker container are started using the command line interface (CLI). For the simple start and maintenance of docker containers, docker-compose is required, which allows the configuration of such containers in a compose.yml configuration file.

This guide enables users to set up and run a **demo Data Warehouse (DWH)** to test **Clinical Document Architecture (CDA)** files that contain anonymized or test-patient data. The system uses **Docker**, which is a popular tool to run applications in isolated environments called containers.

What is Docker?

Docker is a program that helps you run applications in isolated “containers” on your computer. Each container includes all necessary files (e.g., Java, databases, web servers), so you don’t have to install them yourself.

You can think of a Docker container like a “ready-to-go” virtual mini-computer that runs just one job. This is especially useful for software like DWH systems, which normally require complicated installations resulting in possible dependency issues.

Step 1: Install Docker and Docker Compose

Docker is available for:

- Windows
- macOS
- Linux (Ubuntu, Fedora, etc.)

Install Docker by following these official instructions:

- Docker Engine: https://docs.docker.com/engine/install/ (>25.X)
- Docker Compose: https://docs.docker.com/compose/install/ (docker compose > 2.39.x)

Alternatively, you can use the latest version of Docker Desktop including the Docker Engine as well as Docker Compose: https://www.docker.com/products/docker-desktop/

Once installed, test it with this command in your terminal (Linux/macOS) or PowerShell (Windows):

```
docker --version
docker compose version
```

If you see version numbers, Docker is installed correctly.

Step 2: Download and Configure the DWH Demo System

We will now download and start the preconfigured **AKTIN DWH demo system**, which includes:

- a PostgreSQL database
- a WildFly Java application server
- a Web interface (i2b2)

1. Download the Docker Compose configuration

💡 There is no specific folder from which you need to run the following commands, but to start the docker container again later, you need to be with your Command Line Interface at the folder in which the `compose.yml` is stored. You can check in the Terminal the current folder you are in currently with `pwd`, and using the `cd` command you can navigate to a specific folder such as: `cd /home/username/Documents/docker-composefiles/`.

You don't need to be in a specific folder to run Docker commands *in general*.

However, to (re)start a Docker Compose setup later, you **must be in the directory that contains the `compose.yml` (or `docker-compose.yml`) file** — otherwise Docker Compose won't know which project to start.

Use these terminal commands to navigate:

- `pwd` → shows your *current* working directory
- `cd <path>` → change into a specific directory

Example:

```
cd /home/username/Documents/docker-composefiles/
```

Open a terminal and run (Unix, e.g., Ubuntu):

```
curl -LO https://github.com/aktin/docker-aktin-dwh/releases/latest/download/compose.yml
```

or in Powershell (Windows):

```
Invoke-WebRequest -Uri "https://github.com/aktin/docker-aktin-dwh/releases/latest/download/compose.yml" -OutFile "compose.yml"
```

This will download the `compose.yml` file that defines all necessary components of the system and stores it in the folder you run the command from.

2. Create a secret password file

This password will be used by the database:

```
echo 'mysecretpassword' > secret.txt
```

You can replace `mysecretpassword` with a stronger password if you like.

3. Start the system

Now run the following command to start the containers:

```
docker compose up -d
```

The `-d` option tells Docker to run everything in the background. The command automatically pulls, i.e. downloads the required docker container images and runs them. If you remove the `-d` flag, then you will see the log output of the container, however, the container will stop as soon as you close the terminal and won't restart when you reboot the computer. If the command finishes without a clear error message, then the start of the docker container has been successful.

4. Open the web interface

After about 1–2 minutes (depending on your machine), open your browser and visit:

```
http://localhost/webclient
```

You should see the **i2b2 web interface.**

Alternatively, visit:

```
http://localhost/aktin/admin
```

For the **Data Warehouse Manger** which won't function properly unless you setup your own AKTIN Broker Test instance, which is out of scope for this tutorial.

Step 3: Understand the Components

Here's what each part of the demo system does:

Component	Role	Description
database	PostgreSQL DB	Stores medical documents in i2b2/AKTIN format
wildfly	Java App Server	Backend logic and data services
httpd	Apache Web Server	Provides web interface and reverse proxy functionality

Consequently, errors when sending sample CDAs to the DWH will be reported from the wildfly container.

Step 4: Stop or Restart the System

To stop the system:

```
docker compose down
```

> 💡 You need to run the docker compose command from the directory in which the `compose.yml` is stored.

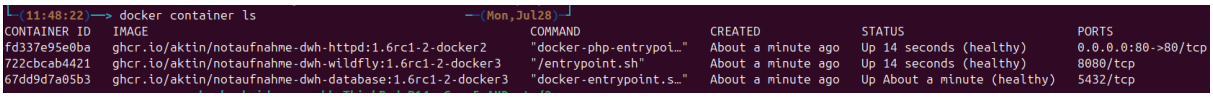
Step 5: Checking actively running containers

> 💡 If you stopped previously the container in Step 4, please start the container again, using the `docker compose up -d` command.

To show the currently running container:

```
docker container ls
```

You should see an output like this:



You can look a the log for a container for a given container id:

```
docker logs 722cbcab4421
```

Sending a CDA to the DWH

Once the DWH system is running on your computer (via Docker), you can test it by sending a **Clinical Document Architecture (CDA)** file to it. A CDA is a type of XML file that contains structured patient information, often used in medical documentation and interoperability. You can use the following file for testing purposes:

test.xml

To do this, we use a tool called **curl** (a command-line tool for sending data to web servers) or a graphical tool like **Postman** (a GUI app for sending HTTP requests).

But before sending the CDA file, we first need to **get an access token**. This token is a kind of password that tells the system who you are and allows you to submit documents securely.

In the next steps, we will:

1. Request an **authorization token** from the system (this is automated via a `curl` command),
2. Use that token to upload the CDA document file to the DWH.

You only need to do this once per session. Once you have the token, you can reuse it for uploading multiple CDA files.

Authorization:

To get an authorization key run the following:

```
curl --location 'http://localhost/aktin/admin/rest/auth/login' \
--header 'Content-Type: application/json' \
--data '{"username":"i2b2","password":"demouser"}'
```

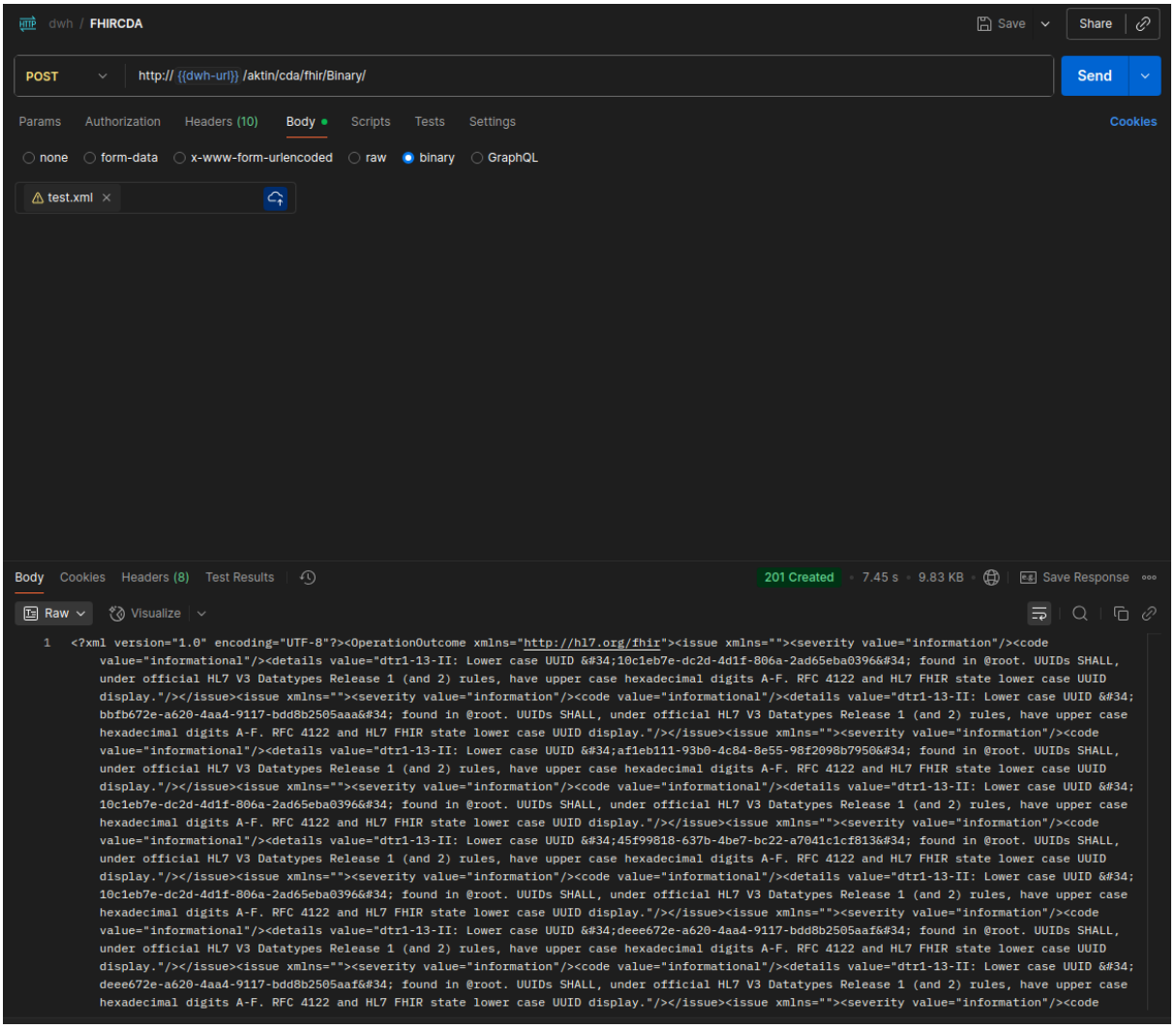
You should get returned the authorization key such as `c96f4b61-e898-4924-9021-3c9e646d3c18`.

Send a CDA to the DWH:

Assuming the test file such as `test.xml` is stored in `/home/user/Documents/test.xml` the CDA can be send to the DWH by the following POST request:

```
curl --location 'http://localhost/aktin/cda/fhir/Binary/' \
--header 'Content-Type: application/xml' \
--header 'Authorization: Bearer c96f4b61-e898-4924-9021-3c9e646d3c18' \
--data-binary '@/home/user/Documents/test.xml'
```

or In POSTMAN with the Bearer Token set to the authorization key:



The HTTP status code includes if the request was successful (code 2XX) and includes additional warnings and errors in case of failure (code 4XX or 5XX). In case of internal server errors caused by a wrongly formatted CDA (status code 5XX) the wildfly container log may contain more information.