Appendix

A Installation Guide

A.1 Prerequisites

- Docker, supporting compose files version >= 3.9
- Valid GraphDB SE/EE license file
- NodeJS (for development)

IMPORTANT: To ease deployment, all containers are pre-built and uploaded to Dockerhub. If you are a new maintainer of the project, go to `frontend/` or `backend/`, rebuild the images with your own tags and push them to your own Docker registry. Finally, use these tags in the compose-files in `deployment/`.

A.2 Deployment

This section explains the steps necessary to deploy the application to a production environment. For local development, see appendix A.3.

API, database, backend, or whole application:

1. Navigate to `deployment/`
2. Add your valid GraphDB license file to the license directory, renaming it graphb.license
3. Copy .env.example and rename it .env. If needed, change the variables. If you're running the database- and API-containers separately (not from the same compose file), make sure that `GRAPHDB_BASE_URL` points to an address on which the API can reach the database.
4. In `build_frontend.Dockerfile`, set `REACT_APP_BACKEND_URL` to the publicly reachable address of the API.
5. Run either...
    - Whole application: `docker compose up`
    - Frontend only: `docker compose -f docker-compose-frontend.yml up`
    - Database only: `docker compose -f docker-compose-db.yml up`
    - Whole backend: `docker compose -f docker-compose-backend.yml up`
6. When the cluster is running and ready, go to http://localhost:7200 (or the externally reachable URL).
7. On the left side of the screen, go to "Setup" and then "Users and Access" (figure fig. 14 and fig. 15).
8. Click "Create new user" and make a user with credentials matching the `GRAPHDB_USERNAME` and `GRAPHDB_PASSWORD` variables set in `.env`. Make sure the user has Read-access to the `TK_SDG`-repository (figure fig. 16).
9. To enable the GDC-functionalities, you'll need to populate the database with data and goals. This can be done by running `utils/datagen.py` after setting up the backend (see `utils/README.md` for more details).
10. If you're running the API, whole backend, or whole application, restart it to reconnect the API with the updated credentials.

Only API or frontend:

1. See step 3 and 4 above.
2. Run `docker compose -f docker-compose-api.yml up`
   or `docker compose -f docker-compose-frontend.yml up`

Default ports:

- DB: 7200
- API: 3100
- Frontend: 80

A.3 Local development

Backend

1. Make sure Docker is running on your computer
2. Place your `graphdb.license` file in `backend/database/conf`
3. Copy the `backend/.env.example`-file, and rename the copy `.env`
4. In `backend/`, run `docker-compose -f docker-compose-backend.yml up`
5. When the Docker-cluster is running, go to http://localhost:7200. On the left side of the screen, go to "Setup" and then "Users and Access" (fig. 14 and fig. 15). Click "Create new user" and make a user with credentials matching the `GRAPHDB_USERNAME` and `GRAPHDB_PASSWORD` fields in `backend/.env`. Make sure the user has Read-access to the `TK_SDG`-repository (figure fig. 16).
6. Stop the docker-compose cluster, and repeat step 4.
7. The backend should now be running, and accessible at http://localhost:3001

Frontend

1. Make sure the API is available on http://localhost:3001 (follow the steps above)
2. Copy the `frontend/.env.example`-file, and rename the copy `.env`
3. In `frontend/`, run docker-compose up –build
4. The app should now be accessible at http://localhost

Frontend (development) You can also develop using a devserver on a Docker container. It uses Docker-volumes to ensure that the container's workspace matches the codebase on your computer. This means that hot-reloading will work the same way as if you ran the devserver locally.

To use the development container, follow these steps:

1. Make sure the API is available on http://localhost:3001
2. Copy the `frontend/.env.example`-file, and rename the copy `.env` (if you haven't done so already)
3. In `frontend/`, run docker-compose -f docker-compose-dev.yml up –build
4. The app should now be accessible at http://localhost:3000

Local devservers When developing the application, it's often easier (and required for the API) to develop locally with the hot-reloadable NodeJS devservers. This can be done by installing the environment with `yarn` in `/frontend` or `/backend`, before starting the desired development server with `yarn start`. For the frontend, this requires a running backend reachable on the `REACT_APP_BACKEND_URL` set in `/frontend/.env`, and for the API, a running GraphDB instance reachable on the `GRAPHDB_BASE_URL` set in `/backend/.env`.

A.4 If changes are made to the ontology

Sometimes, changes to the `/backend/database/ontology/SDG_Ontology.owl` file are made. To implement these into your database, you have to do the following:

1. Before composing the backend Docker containers in Backend step 4, run `docker-compose -f docker-compose-bac`
   This forces the containers to rebuild with the new ontology.
2. Continue with step 5 to 7 in the backend setup guide.

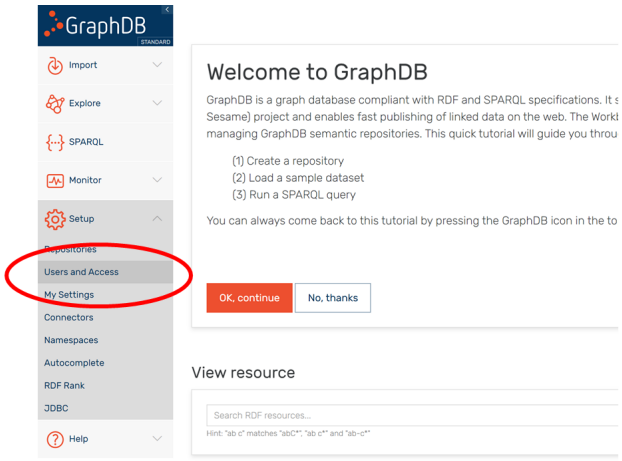

Figure 14: Access “Users and Access” page

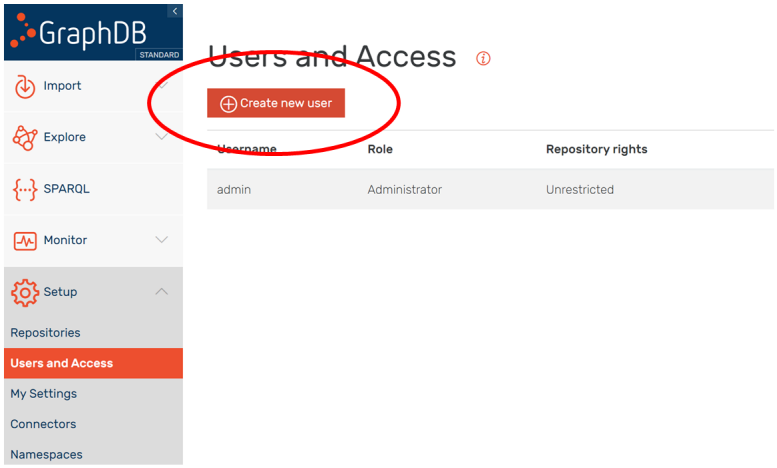

Figure 15: Click “Create new user”

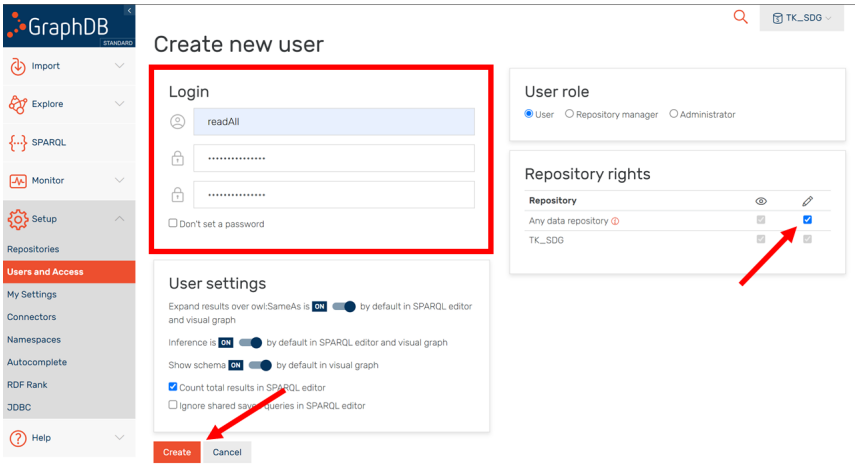

Figure 16: Fill out form