Docker

> ⚠ **Corporate Edition only!**
>
> The installation procedure described here applies to iteraplan Corporate Edition only. Lite Edition users should follow the instructions in the bundled README.txt file.

The iteraplan Corporate Edition is also available as Docker image. The image can be pulled from the iteratec Docker Registry as image `iteraplan.docker.iteratec.de/iteraplan-corporate:VERSION` where VERSION is the iteraplan version, for example 6.3.1.

Usage

iteraplan should be orchestrated with Docker Compose. An example Docker Compose file `iteraplan.yml` is included in the `docker-compose` directory of the iteraplan Docker package available at our download site.

Docker Compose is written in python and can be installed through python's Package Installer `pip` if it's not already installed. `pip install docker-compose` will install this for you. If pip is missing on your system you need to install it first.

Once the Docker Compose file is configured according to the following sections, iteraplan can be started from the directory the Docker Compose file is in via command line:

```
docker-compose -f iteraplan.yml up -d
```

Once the application is fully loaded within the Docker container, it can be accessed on the ports given in the Docker Compose file's "ports" section. For example at `http://localhost:8066/` in case of the default port mapping of the provided example Docker Compose file.

Database Configuration

Database configuration works by setting environment variables in the Docker Compose file.

`DB_TYPE`	{mysql|oracle|mssqlserver} - Database vendor
`DB_HOST`	The host name or IP address of the DB server
`DB_PORT`	The port of the DB server
`DB_NAME`	The name of the database, or the Oracle Service name in case of DB_TYPE=oracle
`SID`	Only used for DB_TYPE=oracle, **instead** of DB_NAME. If DB_NAME is set, SID is ignored even if it is given.
`DB_USER`	The user to access the iteraplan database
`DB_PASS`	The password for DB_USER

The iTURM database connection has its own variables, corresponding to the iteraplan database variables:

`LOGIN_DB_TYPE`	{mysql|oracle|mssqlserver} - Database vendor
`LOGIN_DB_HOST`	The host name or IP address of the DB server
`LOGIN_DB_PORT`	The port of the DB server
`LOGIN_DB_NAME`	The name of the database, or the Oracle Service name in case of DB_TYPE=oracle
`LOGIN_SID`	Only used for DB_TYPE=oracle, **instead** of DB_NAME. If DB_NAME is set, SID is ignored even if it is given.
`LOGIN_DB_USER`	The user to access the iteraplan database
`LOGIN_DB_PASS`	The password for DB_USER

JDBC Connector

See also Database drivers.

The JDBC connector file needs to be mounted into the `/usr/local/tomcat/lib` folder of the iteraplan Docker container. Examples are given in the example Docker Compose file, for connectors present in the same directory as the Docker Compose file.

Important: The jdbc connector file requires read permission for user 1001 (tomcat inside the docker contrainer runs as this user).

LDAP configuration

Authentication schemes can be selected by setting the environment variable `AUTH` accordingly. Possible values are "iturm", "ldap", "sso" or "x509", corresponding to the Authentication variants described in the section User Authentication and sub pages.

If another scheme than "iturm" is chosen, an authentication configuration file iteraplan-auth.properties has to be provided to configure the LDAP connection. If no LDAP configuration file is present already, a template can be retrieved from the iteraplan Docker image:

```
docker run --rm iteraplan.docker.iteratec.io/iteraplan-corporate:6.3.1 cat /iteradocker/iteraplan-auth.
properties > iteraplan-auth.properties
```

After the file was altered according to the LDAP connection settings, it has to be mounted back to the Docker container. The example Docker Compose file contains an according line in the volumes section of the iteraplan service.

Tomcat configuration

The amount of memory to be used for iteraplan can be configured in the environmental variable CATALINA_OPTS. Our sample docker compose files contain a default setting:

```
services:
  iteraplan:
    ...
    environment:
      ...
      # Memory settings for tomcat
      - CATALINA_OPTS=-Xmx2g -XX:MaxPermSize=256m
```

If the file `server.xml` of the Tomcat configuration needs to be changed for iteraplan, this can be done similar to the `iteraplan-auth.properties` configuration mentioned above:

```
docker run --rm iteraplan.docker.iteratec.io/iteraplan-corporate:6.3.1 cat /usr/local/tomcat/conf/server.xml >
server.xml
```

After the file was altered, it has to be mounted back to the Docker container. The example Docker Compose file contains an according line in the volumes section of the iteraplan service.

Shared files

Some files used or created by iteraplan need to be accessible outside of the Docker container. Examples are logfiles, or scripts for the Graphics Reactor or Plugin scripts.

For this purpose, the following directories within the Docker container can be mounted as Docker Volumes:

- /var/iteraplan/logs
- /usr/local/tomcat/logs
- /var/iturm/logs
- /var/iteraplan/indexes
- /usr/local/tomcat/webapps/ROOT/WEB-INF/classes/reactor

By saving files in these directories outside of the Docker Container, they can be easily accessed, and easily kept when upgrading to a different iteraplan version.

In the example Docker Compose file, those directories are all mounted to sub directories of the folder the Docker Compose file is in.

Important: The mapped local directories require write and execute permission for user 1001 (tomcat inside the docker contrainer runs as this user).

Troubleshooting

Docker Compose reports:

```
ERROR: Error: Status 400 trying to pull repository iteraplan-corporate: "{\n  \"errors\" : [ {\n    \"status\"
: 400,\n    \"message\" : \"Unsupported docker v1 repository request for 'iteradocker'\"\n  } ]\n}"
```

Possible reasons:

- Not logged in to the Docker repository (eg with: `docker login iteraplan.docker.iteratec.io`)
- Docker image or tag are not correctly given

Error connecting to the database

Possible reasons:

- JDBC database connector missing
- JDBC database connector jar file not mounted correctly to the container
- Database connection details entered in the Docker Compose file are wrong

Error writing log files

Possible reasons:

- The mapped local folders does not have write and execute permissions for user 1001.