



SeicentoBilling Installation Guide

Software Release v20.1.1

Inhaltsverzeichnis

1. Einführung	3
1.1. Konventionen	3
1.2. SeicentoBilling Distributionen	3
1.3. Systemanforderungen	3
2. Installation des Docker Images	4
2.1. Anforderungen	4
2.2. Pre-Installation Tasks	5
2.2.1. Github clone Seicentobilling-cmdline	5
2.3. Initiale Datenbank erstellen	7
2.3.1. Anpassen .env Datei	7
2.3.1. Starten Image SeicentoBilling-cmdline	8
2.1. Starten SeicentoBilling	9
2.1.1. Anpassen .env Datei für seicentobilling	9
2.1.2. docker ps	9
2.1.3. Login Screen	10
3. Optionen	11
3.1. JasperReports	11
3.2. Pre-Installation Tasks	11
3.2.1. Github clone js-docker	11
3.2.2. Installation	13
3.3. Reverse Proxy NGinx	17
3.3.1. nginx.conf	17
3.3.2. Verschlüsselung mit letsencrypt	18

1. Einführung

SeicentoBilling ist eine einfache Weblösung für Leistungsrapportierung, Spesen und Rechnungsstellung. Die App ist seit mehreren Jahren produktiv im Einsatz.

Die Sourcen der Lösung sind auf [Github](#) unter der Apache 2.0 Lizenz verfügbar. Ein Docker Image kann von [Dockerhub](#) bezogen werden.

History:

Datum	Wer	Bemerkung
23.03.2020	Muri Josef	Eröffnet für Version 20.1.1

1.1.Konventionen

Dieses Dokument verwendet folgende Konventionen.

Konvention	Beschreibung
<i>docker-compose up</i>	Kommando, dass in einer Shell eingegeben wird.

1.2.SeicentoBilling Distributionen

SeicentoBilling kann grundsätzlich auf mehrere Arten installiert/gestartet werden.

- Am einfachsten ist dabei die Verwendung des angebotenen Docker Images auf Dockerhub.
- Es ist auch möglich die Sourcen von Github zu beziehen und ein WAR File zu erstellen oder die Lösung in der Entwicklungsumgebung (Rapidclipse) zu starten.

1.3.Systemanforderungen

Die folgende Tabelle enthält die empfohlenen minimal Anforderungen. Das beinhaltet auch eine Datenbank wie Microsoft SQL oder Postgresql.

Resource	Minimum	Empfohlen
Disk	10 GB freier Diskplatz	40 GB+
RAM	4 GB	12 GB+
CPU	2 Core	2.5 Ghz+ multi Core für Windows, Mac und Linux

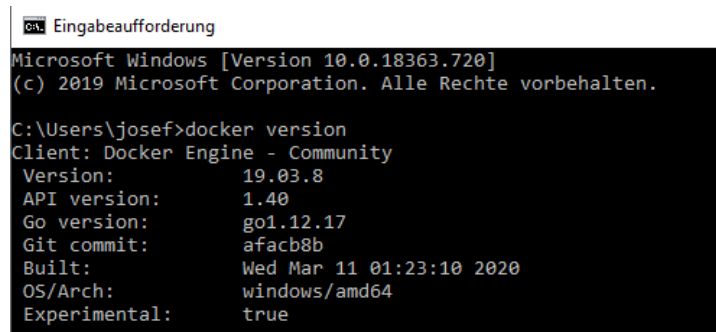
2. Installation des Docker Images

Die schnellste Methode besteht darin die Angebotenen Docker-Images auf Dockerhub zu verwenden.

2.1. Anforderungen

Bevor mit der Installation begonnen werden kann müssen auf dem Zielcomputer folgende Komponenten vorhanden sein:

- Docker / docker-compose (ab Version 18.x)
- Ein Datenbankserver:
 - Microsoft SQL ab Version 12
 - Postgresql ab Version 11
- Git-Client (optional)



```
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\josef>docker version
Client: Docker Engine - Community
Version:          19.03.8
API version:      1.40
Go version:       go1.12.17
Git commit:       afacb8b
Built:            Wed Mar 11 01:23:10 2020
OS/Arch:          windows/amd64
Experimental:     true
```

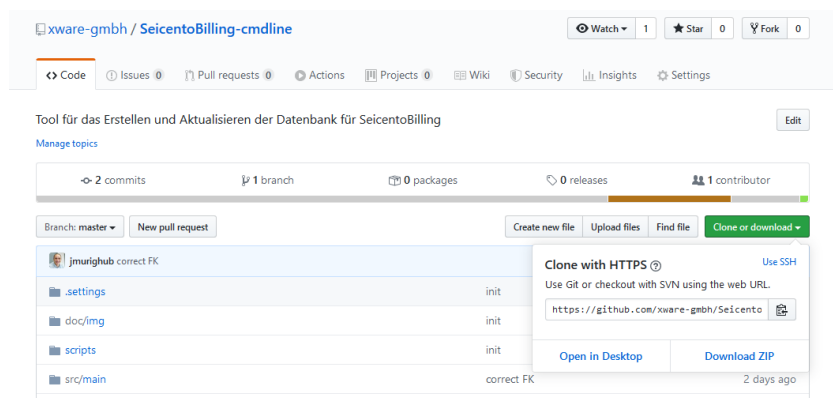
Docker ist in der Version 19.03. vorhanden.

2.2.Pre-Installation Tasks

Bevor mit der eigentlichen Installation begonnen werden kann muss man noch folgende Daten herunterladen.

- SeicentoBilling-cmdline von Github (<https://github.com/xware-gmbh/SeicentoBilling-cmdline>)

2.2.1. Github clone Seicentobilling-cmdline



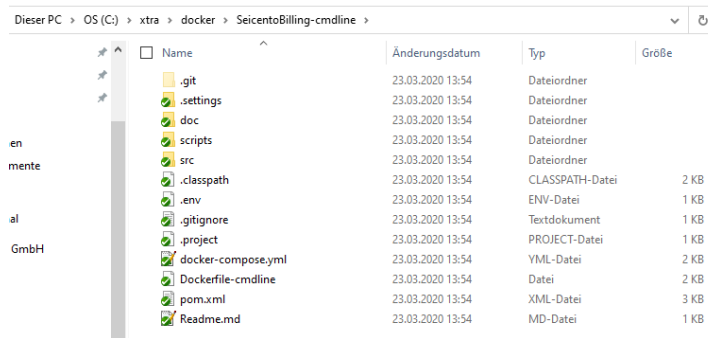
- Mittels Git clone (`git clone https://github.com/xware-gmbh/SeicentoBilling-cmdline.git`) oder via download ZIP die Dateien in ein lokales Verzeichnis kopieren (z.B. c:\xtra\docker)

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. Alle Rechte vorbehalten.

C:\xtra\docker>git clone https://github.com/xware-gmbh/SeicentoBilling-cmdline.git
Cloning into 'SeicentoBilling-cmdline'...
remote: Enumerating objects: 76, done.
remote: Counting objects: 100% (76/76), done.
remote: Compressing objects: 100% (56/56), done.
remote: Total 76 (delta 5), reused 73 (delta 2), pack-reused 0
Unpacking objects: 100% (76/76), done.

C:\xtra\docker>
```

folgende Daten sollten nachher vorhanden sein:



Name	Änderungsdatum	Typ	Größe
.git	23.03.2020 13:54	Dateiordner	
.settings	23.03.2020 13:54	Dateiordner	
doc	23.03.2020 13:54	Dateiordner	
scripts	23.03.2020 13:54	Dateiordner	
src	23.03.2020 13:54	Dateiordner	
.classpath	23.03.2020 13:54	CLASSPATH-Datei	2 KB
.env	23.03.2020 13:54	ENV-Datei	1 KB
.gitignore	23.03.2020 13:54	Textdokument	1 KB
.project	23.03.2020 13:54	PROJECT-Datei	1 KB
docker-compose.yml	23.03.2020 13:54	YML-Datei	2 KB
Dockerfile-cmdline	23.03.2020 13:54	Datei	2 KB
pom.xml	23.03.2020 13:54	XML-Datei	3 KB
Readme.md	23.03.2020 13:54	MD-Datei	1 KB

Anschliessend im erstellten Verzeichnis (C:/xtra/docker/SeicentoBilling-cmdline) ein Cmd Shell öffnen.

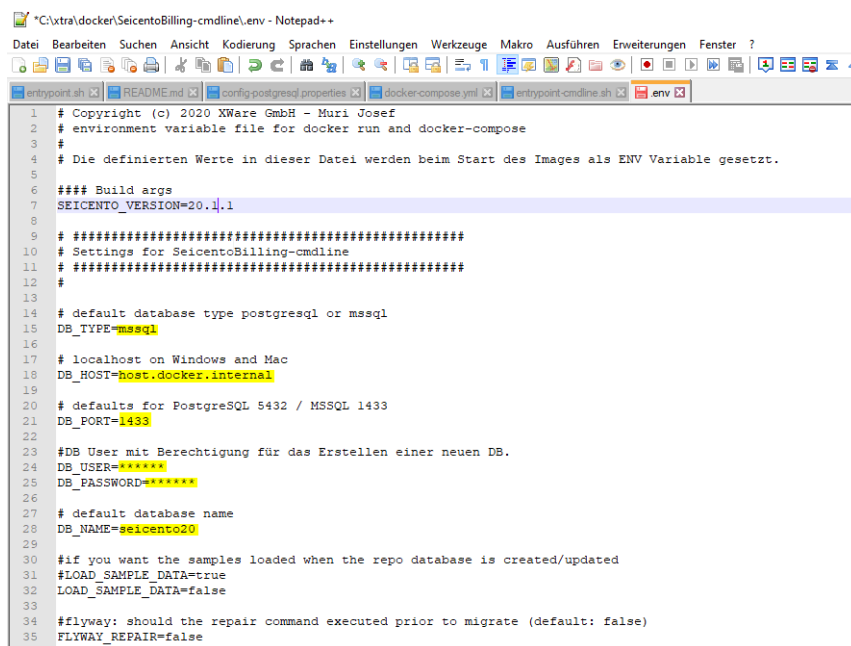
2.3. Initiale Datenbank erstellen

Um die Datenbank für Seicento zu initialisieren ist wie folgt vorzugehen:

1. Anpassen .env File mit DB Credentials
2. Starten Docker Image mit docker-compose

2.3.1. Anpassen .env Datei

Im Verzeichnis C:/xtra/docker/SeicentoBilling-cmdline befindet sich eine Datei «.env». Diese ist mittels eines Editor anzupassen.



```
1 # Copyright (c) 2020 XWare GmbH - Muri Josef
2 # environment variable file for docker run and docker-compose
3 #
4 # Die definierten Werte in dieser Datei werden beim Start des Images als ENV Variable gesetzt.
5
6 ### Build args
7 SEICENTO_VERSION=20.1.1
8
9 #####
10 # Settings for SeicentoBilling-cmdline
11 # #####
12 #
13
14 # default database type postgresql or mssql
15 DB_TYPE=mssql
16
17 # localhost on Windows and Mac
18 DB_HOST=host.docker.internal
19
20 # defaults for PostgreSQL 5432 / MSSQL 1433
21 DB_PORT=1433
22
23 #DB User mit Berechtigung für das Erstellen einer neuen DB.
24 DB_USER=*****
25 DB_PASSWORD=*****
26
27 # default database name
28 DB_NAME=seicento20
29
30 #if you want the samples loaded when the repo database is created/updated
31 #LOAD_SAMPLE_DATA=true
32 LOAD_SAMPLE_DATA=false
33
34 #flyway: should the repair command executed prior to migrate (default: false)
35 FLYWAY_REPAIR=false
```

Die gelb markierten Werte sind zu überprüfen bzw. zu setzen. ACHTUNG: der DB User muss über die Berechtigung verfügen um eine neue DB anlegen zu dürfen.

Im obigen Beispiel würde eine neue MSSQL DB angelegt auf der aktuellen Maschine, mit dem Namen seicento20.

2.3.1. Starten Image SeicentoBilling-cmdline

1. Im Verzeichnis C:/xtra/docker/SeicentoBilling-cmdline eine Cmd Shell öffnen.
2. Ausführen von «*docker-compose up*»

Mit dem Kommando «*docker-compose up*» wird nun das Image von Dockerhub geholt und gestartet. Dabei werden die Werte aus der .env Datei berücksichtigt. Wenn alles korrekt ist sollte das in etwa so aussehen:

```
C:\Windows\System32\cmd.exe

C:\Users\josef\workspace\git\seicentobilling-cmdline>docker-compose up
Recreating seicentobilling-cmdline_seicentobilling-cmdline_1 ... done
Attaching to seicentobilling-cmdline_seicentobilling-cmdline_1
seicentobilling-cmdline_1 | seicentobilling-cmdline check DB postgresql on host host.docker.internal 5432
seicentobilling-cmdline_1 | -----
seicentobilling-cmdline_1 | Start check Database...
seicentobilling-cmdline_1 | Mar 17, 2020 12:23:06 PM org.flywaydb.core.Flyway setLocations
seicentobilling-cmdline_1 | jdbc:postgresql://host.docker.internal:5432/
seicentobilling-cmdline_1 | Database does already exist.
seicentobilling-cmdline_1 | -----
seicentobilling-cmdline_1 | Start flyway - check tables...
seicentobilling-cmdline_1 | jdbc:postgresql://host.docker.internal:5432/seicento
seicentobilling-cmdline_1 | Mar 17, 2020 12:23:06 PM org.flywaydb.core.internal.database.DatabaseFactory createDatabase
seicentobilling-cmdline_1 | WARNING: Direct configuration of the Flyway object has been deprecated and will be removed in Flyway 6.0. Use Flyway.configure() instead.
seicentobilling-cmdline_1 | Mar 17, 2020 12:23:06 PM org.flywaydb.core.Flyway setSchemas
seicentobilling-cmdline_1 | WARNING: Direct configuration of the Flyway object has been deprecated and will be removed in Flyway 6.0. Use Flyway.configure() instead.
seicentobilling-cmdline_1 | Mar 17, 2020 12:23:06 PM org.flywaydb.core.internal.license.VersionPrinter printVersionOnly
seicentobilling-cmdline_1 | INFO: Flyway Community Edition 5.2.4 by Boxfuse
seicentobilling-cmdline_1 | Mar 17, 2020 12:23:06 PM org.flywaydb.core.internal.database.DatabaseFactory createDatabase
seicentobilling-cmdline_1 | INFO: Database: jdbc:postgresql://host.docker.internal:5432/seicento (PostgreSQL 11.7)
seicentobilling-cmdline_1 | Mar 17, 2020 12:23:06 PM org.flywaydb.core.internal.database.base.Database recommendFlywayUpgradeIfNecessaryForMajorVersion
seicentobilling-cmdline_1 | WARNING: Flyway upgrade recommended: PostgreSQL 11.7 is newer than this version of Flyway and support has not been tested.
seicentobilling-cmdline_1 | Mar 17, 2020 12:23:06 PM org.flywaydb.core.internal.command.DbValidate validate
seicentobilling-cmdline_1 | INFO: Successfully validated 5 migrations (execution time 00:00.124s)
seicentobilling-cmdline_1 | Mar 17, 2020 12:23:06 PM org.flywaydb.core.internal.command.DbMigrate migrateGroup
seicentobilling-cmdline_1 | INFO: Current version of schema "dbo": 1.02
seicentobilling-cmdline_1 | Mar 17, 2020 12:23:06 PM org.flywaydb.core.internal.command.DbMigrate logSummary
seicentobilling-cmdline_1 | INFO: Schema "dbo" is up to date. No migration necessary.
seicentobilling-cmdline_1 | Mar 17, 2020 12:23:06 PM org.flywaydb.core.internal.database.base.Database recommendFlywayUpgradeIfNecessaryForMajorVersion
seicentobilling-cmdline_1 | WARNING: Flyway upgrade recommended: PostgreSQL 11.7 is newer than this version of Flyway and support has not been tested.
seicentobilling-cmdline_1 | -----
seicentobilling-cmdline_1 | | Category | Version | Description | Type | Installed On | State |
seicentobilling-cmdline_1 | |-----|-----|-----|-----|-----|-----|
seicentobilling-cmdline_1 | | R | 1.0 | << Flyway Schema Creati | SCHEMA | 2020-03-18 13:49:41 | SUCCESS |
seicentobilling-cmdline_1 | | | 1.0 | Initial Tables | SQL | 2020-03-18 14:04:57 | SUCCESS |
seicentobilling-cmdline_1 | | | 1.01 | InsertValues | SQL | 2020-03-18 14:04:57 | SUCCESS |
seicentobilling-cmdline_1 | | | 1.02 | Procedures | SQL | 2020-03-18 14:04:57 | SUCCESS |
seicentobilling-cmdline_1 | | R | | Initial CodeValues | SQL | 2020-03-18 14:04:57 | SUCCESS |
seicentobilling-cmdline_1 | |-----|-----|-----|-----|-----|
seicentobilling-cmdline_1 | [END]
seicentobilling-cmdline_1 exited with code 0
C:\Users\josef\workspace\git\seicentobilling-cmdline>
```

Die Datenbank ist nun erstellt und verfügbar für SeicentoBilling.

2.1. Starten SeicentoBilling

Nachdem die Datenbank nun verfügbar ist müssen wir nun noch SeicentoBilling konfigurieren und starten. Dies geschieht mit den folgenden Schritten.

1. Kopieren .env Datei in den Unterordner «seicentobilling»
2. Bearbeiten .env Datei in Unterordner «seicentobilling»
3. In der Cmd-Shell wechseln in den Unterordner «seicentobilling» (*cd seicentobilling*)
4. Ausführen von «*docker-compose up -d*»
5. Starten App in Browser (<http://localhost:8080/SeicentoBilling>)

2.1.1. Anpassen .env Datei für seicentobilling

```
# #####
# Settings for SeicentoBilling
# #####
#
APP_STAGE=TEST

# sample for postgresql: jdbc:postgresql://host.docker.internal:5432/se[DBNAME]
DB_URL_TEST=jdbc:sqlserver://host.docker.internal:1433;database=[DBNAME];encrypt=true;trustServerCertificate=false;hostNameInCertificate=.database.windows.net;loginTimeout=30;

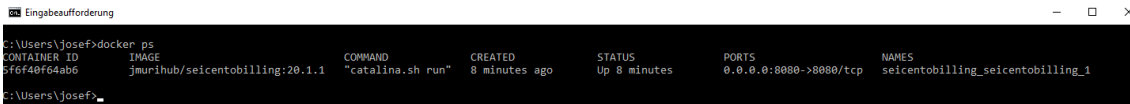
#DB User mit Lese/Schreibrecht auf DB.
DB_USER_TEST=[DBUSER]
DB_PWD_TEST=[DBPASSWORD]

#Mögliche Werte: azure, local (default: azure)
SEICENTO_LOGIN_METHOD=local

#Die folgenden Werte sind nur zu setzen bei LOGIN_METHOD=azure
tenantid=[Company.com]
clientid=[Azure ClientID]
clientkey=[Azure Clients Key]
```

Die gelb markierten Felder sind zu überprüfen bzw. anzupassen. Dabei können einzelne Werte aus dem Bereich von SeicentoBilling-cmdline übernommen werden.

2.1.2. docker ps

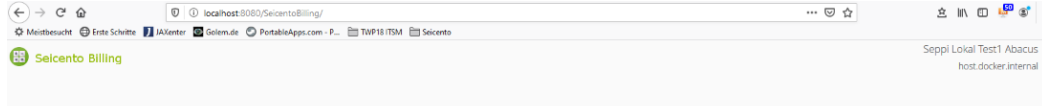


```
Eingabeaufforderung
c:\Users\josef>docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                    NAMES
3f6f40f64ab6   jmurihub/seicentobilling:20.1.1    "catalina.sh run"       8 minutes ago Up 8 minutes   0.0.0.0:8080->8080/tcp   seicentobilling_seicentobilling_1
c:\Users\josef>
```

Nach erfolgreichem Start des Images kann mit «*docker ps*» überprüft werden ob dieses läuft.

2.1.3. Login Screen

Auf dem Login Screen wird das Login für das «lokale Login» verlangt. Standardmässig gibt es hierfür den



User: demo mit Pw: changme.

Bei der Konfiguration für Azure (AAD) erscheint hier nur ein grüner Login Button, welcher dann auf die Login-page von Microsoft weiter leitet.

3. Optionen

3.1.JasperReports

JasperReports bzw. JasperServer ist ein Produkt aus dem Hause von Tibco. Wir verwenden die Community Edition von JasperServer/JasperReports. Aktuell in der Version 7.5.

Vor der Installation ist noch zu definieren, wo Jasper seine Daten ablegt. Mit der Community Edition gibt es hierfür 2 Möglichkeiten:

- Postgresql DB Server
- Postgresql als Docker Image

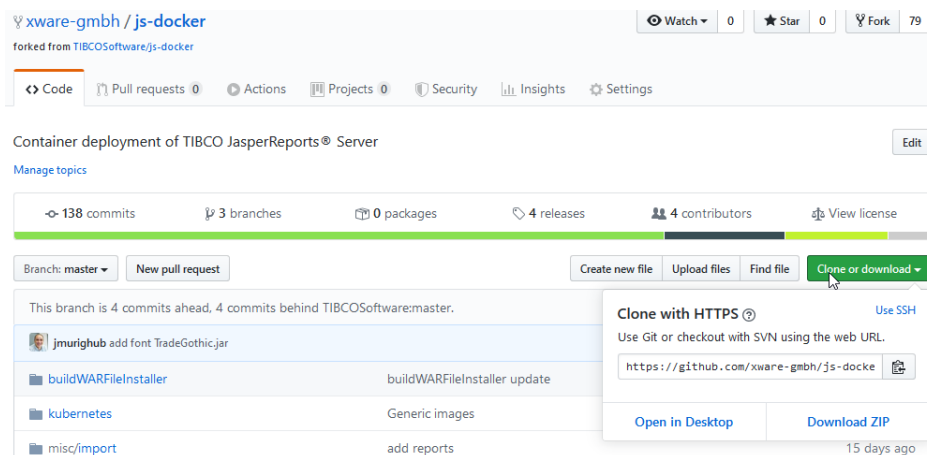
Im folgenden ist die Variante mit einem dedizierten DB Server beschrieben.

3.2.Pre-Installation Tasks

Bevor mit der eigentlichen Installation begonnen werden kann muss man noch folgende Daten herunterladen.

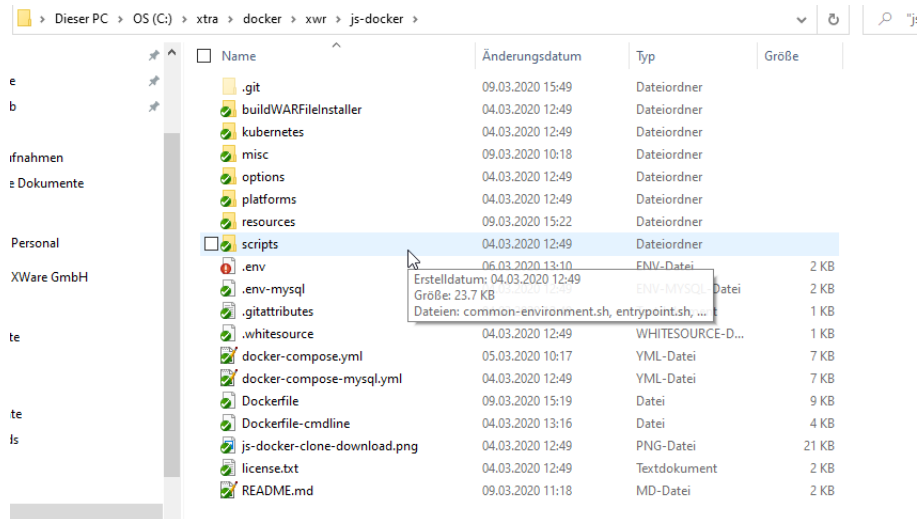
- js-docker von Github (<https://github.com/xware-gmbh/js-docker>)

3.2.1. Github clone js-docker



- Mittels Git clone (`git clone https://github.com/xware-gmbh/js-docker.git`) oder via download ZIP die Dateien in ein lokales Verzeichnis kopieren (z.B. c:/xtra/docker)

Danach sollte das lokale Verzeichnis in etwa so aussehen:

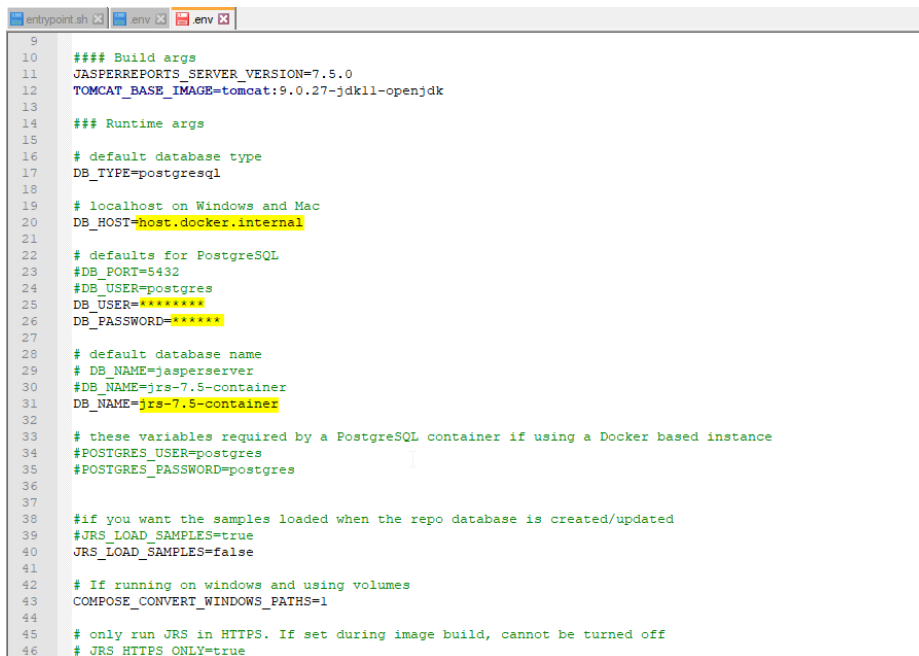


Anschliessend im erstellten Verzeichnis (C:/xtra/docker/js-docker) eine cmd Shell öffnen.

3.2.2. Installation

1. Anpassen/Ändern .env Datei
2. Starten docker image
3. Login als Admin (jasperadmin)
4. Import Seicento Reports (*.zip)
5. Setzen Passwort für jasperadmin
6. Konfiguration DB Verbindung für Reports

Anpassen/Ändern .env Datei



```
9
10 ##### Build args
11 JASPERREPORTS_SERVER_VERSION=7.5.0
12 TOMCAT_BASE_IMAGE=tomcat:9.0.27-jdk11-openjdk
13
14 ##### Runtime args
15
16 # default database type
17 DB_TYPE=postgresql
18
19 # localhost on Windows and Mac
20 DB_HOST=host.docker.internal
21
22 # defaults for PostgreSQL
23 #DB_PORT=5432
24 #DB_USER=postgres
25 DB_USER=postgres
26 DB_PASSWORD=postgres
27
28 # default database name
29 # DB_NAME=jasperserver
30 #DB_NAME=jrs-7.5-container
31 DB_NAME=jrs-7.5-container
32
33 # these variables required by a PostgreSQL container if using a Docker based instance
34 #POSTGRES_USER=postgres
35 #POSTGRES_PASSWORD=postgres
36
37
38 #if you want the samples loaded when the repo database is created/updated
39 #JRS_LOAD_SAMPLES=true
40 JRS_LOAD_SAMPLES=false
41
42 # If running on windows and using volumes
43 COMPOSE_CONVERT_WINDOWS_PATHS=1
44
45 # only run JRS in HTTPS. If set during image build, cannot be turned off
46 # JRS_HTTPS_ONLY=true
```

Die gelb markierten Werte sind zu überprüfen bzw. zu setzen. ACHTUNG: der DB User muss über die Berechtigung verfügen um eine neue DB anlegen zu dürfen.

Im obigen Beispiel würde eine neue Postgresql DB angelegt auf der aktuellen Maschine, mit dem Namen jrs-7.5-container.

Starten docker image

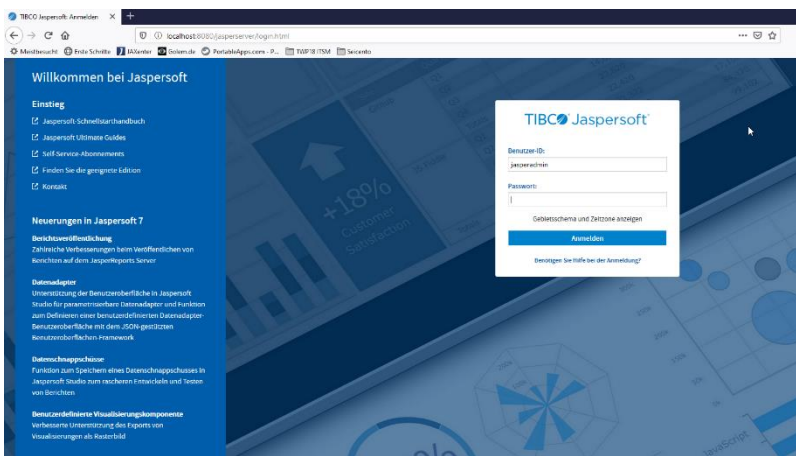
- In der Shell zu c:/xtra/docker/js-docker navigieren (`cd c:/xtra/docker/js-docker`)
- Image Starten (`docker-compose up`)

Auf der Konsole kann nun verfolgt werden, wie 2 Images gestartet werden. Das Erste erstellt die initiale Datenbank für Jasper. Das Zweite Image startet Jasperserver.

```

C:\Windows\System32\cmd.exe - docker-compose up
d:\usr\local\tomcat\temp
jasperserver-cp_1 | 23-Mar-2020 20:01:20.662 INFORMATION [main] org.apache.catalina.core.AprLifecycleListener.lifecycleEvent Loaded APR based Apache T
Native library [1.2.23] using APR version [1.5.2].
jasperserver-cp_1 | 23-Mar-2020 20:01:20.662 INFORMATION [main] org.apache.catalina.core.AprLifecycleListener.lifecycleEvent APR capabilities: IPv6 [t
, sendfile [true], accept filters [false], random [true].
jasperserver-cp_1 | 23-Mar-2020 20:01:20.662 INFORMATION [main] org.apache.catalina.core.AprLifecycleListener.lifecycleEvent APR/OpenSSL configuration
eAprConnector [false], useOpenSSL [true]
jasperserver-cp_1 | 23-Mar-2020 20:01:20.671 INFORMATION [main] org.apache.catalina.core.AprLifecycleListener.lifecycleEvent InitializeSSL OpenSSL successfully initi
ed [OpenSSL 1.1.0] 10 Sep 2019]
jasperserver-cp_1 | 23-Mar-2020 20:01:21.093 INFORMATION [main] org.apache.coyote.AbstractProtocol.init Initialisiere ProtocolHandler["http-nio-8080"]
jasperserver-cp_1 | 23-Mar-2020 20:01:21.124 INFORMATION [main] org.apache.coyote.AbstractProtocol.init Initialisiere ProtocolHandler["ajp-nio-8009"]
jasperserver-cp_1 | 23-Mar-2020 20:01:21.127 INFORMATION [main] org.apache.coyote.AbstractProtocol.init Initialisiere ProtocolHandler["https-openssl-n
443"]
jasperserver-cp_1 | 23-Mar-2020 20:01:21.388 INFORMATION [main] org.apache.catalina.startup.Catalina.load Server initialization in [1.043] milliseconds
jasperserver-cp_1 | 23-Mar-2020 20:01:21.484 INFORMATION [main] org.apache.catalina.core.StandardService.startInternal Starting service [Catalina]
jasperserver-cp_1 | 23-Mar-2020 20:01:21.464 INFORMATION [main] org.apache.catalina.core.StandardEngine.startInternal Starting Servlet engine: [Apache
cat/9.0.27]
jasperserver-cp_1 | 23-Mar-2020 20:01:21.472 INFORMATION [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploye Web-Applikations-Verzei
l (/usr/local/tomcat/webapps/jasperserver)
jasperserver-cp_1 | 23-Mar-2020 20:01:31.836 INFORMATION [main] org.apache.jasper.servlet.TldScanner.scanJars At least one JAR was scanned for TLDs ye
ntained no TLDs. Enable debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs durin
g scanning can improve startup time and JSP compilation time.
jasperserver-cp_1 | 23-Mar-2020 20:01:55.410 INFORMATION [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application d
tory [/usr/local/tomcat/webapps/jasperserver] has finished in [33.937] ms
jasperserver-cp_1 | 23-Mar-2020 20:01:55.445 INFORMATION [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
jasperserver-cp_1 | 23-Mar-2020 20:01:55.429 INFORMATION [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["ajp-nio-8009"]
jasperserver-cp_1 | 23-Mar-2020 20:01:55.441 INFORMATION [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["https-openssl-nio-
jasperserver-cp_1 | 23-Mar-2020 20:01:55.453 INFORMATION [main] org.apache.catalina.startup.Catalina.start Server startup in [34.064] milliseconds
  
```

Wenn dies ohne Fehler durchläuft kann Jasper im Browser geöffnet werden unter <http://localhost:8080/jasperserver>



Für die Erstanmeldung ist der Standard User/Pw zu verwenden (jasperadmin).

Import Seicento Reports (*.zip)

Wenn man sich erfolgreich mit dem Benutzer jasperadmin anmelden konnte, kann man nun die Reports für SeicentoBilling importieren.

Der Import Screen ist in Jasper unter Verwalten/Servereinstellungen/Importieren zu finden.

Die Reports sind im ZIP C:/xtra/docker/jd-docker/misc/import/JS_SeicentoSalary_Demo_2020.zip zu finden. Die Datei durch drücken des Knopfes «Durchsuchen» auswählen und danach den «Importieren» Knopf drücken.

WICHTIG: Der Import kann 1-2 Minuten dauern. Es werden auch die Benutzer/Passwörter überschrieben. Deshalb ist es wichtig, nach dem Import das Passwort wieder zu setzen.

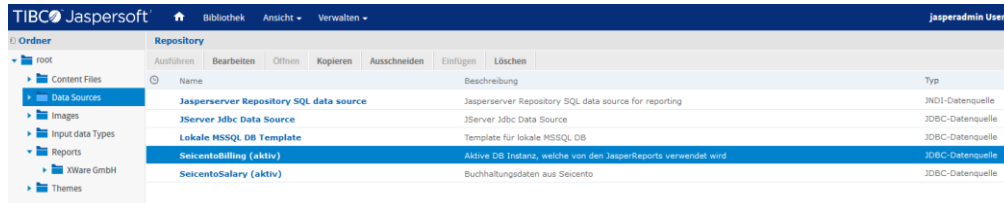
Passwort für jasperadmin setzen

Wie weiter oben vermerkt werden durch den Import die Passwörter neu gesetzt. Somit muss man zwingend das Passwort neu setzen vor dem logout, ansonsten man sich nicht mehr anmelden kann.

Der Screen ist in Jasper unter Verwalten/Benutzer zu finden. Danach Benutzer «jasperadmin» auswählen und «Bearbeiten» klicken.

Konfigurieren DB Verbindung für Reports

Damit die importierten Reports auf Daten zugreifen können, muss diese Verbindung noch überprüft und angepasst werden.



Name	Beschreibung	Typ
Jasperserver Repository SQL data source	Jasperserver Repository SQL data source for reporting	JNDI-Datenquelle
JServer jdbc Data Source	JServer jdbc Data Source	JDBC-Datenquelle
Lokale MSSQL DB Template	Template für lokale MSSQL DB	JDBC-Datenquelle
SeicentoBilling (aktiv)	Aktive DB Instanz, welche von den JasperReports verwendet wird	JDBC-Datenquelle
SeicentoSalary (aktiv)	Buchhaltungsdaten aus Seicento	JDBC-Datenquelle

Die JDBC Verbindung unter «Data Source» ist zu bearbeiten. Die gelb markierten Felder sind zu überprüfen bzw. zu setzen um auf die Daten von SeicentoBilling zugreifen zu können. Nachdem die Verbindung erfolgreich getestet wurde, können nun Reports (Reports/XWare GmbH) gestartet werden.

SeicentoBilling (aktiv)

Datenquelleneigenschaften bearbeiten

Typ:

JDBC-Treiber:
 Treiber auswählen...

Host (erforderlich):

Port (erforderlich):

Datenbank (erforderlich):

URL (erforderlich):

Hinweis: jdbc:postgresql://localhost:5432/mydb Erforderliche URL

Benutzername:

Passwort:

Zeitzone:
 ▼
Hinweis: Ändern Sie die Zeitzoneinstellung nur, wenn Sie sicher sind, dass die Daten des Datenbank-Zeitstempels falsch sind.

Verbindung testen

3.3.Reverse Proxy NGinx

[Nginx](#) ist ein Reverse Proxy der sehr einfach mit Docker verwendet werden kann. Dies zusammen mit letsencrypt ermöglicht es mit dem Reverse Proxy mittels https zu kommunizieren.

Verwendung von nginx.conf als Template im Zusammenhang mit docker-compose.yml für den Start von Reverse Proxy und SeicentoBilling

3.3.1. nginx.conf

Im folgenden ein Beispiel für die Konfiguration von nginx für SeicentoBilling und Jasper.

```
worker_processes 2;

events { worker_connections 1024; }

http {
    sendfile on;

    upstream billing-stream {
        server svc-billing:8080;           #container port, not host port
    }

    # upstream jasper-stream {
    #     server svc.jasper:8080;           #container port, not host port
    # }
    #redirect http to https

    server {
        listen 8089 default_server;
        listen [::]:8089 default_server;

        server_name _;

        # SeicentoBilling
        location /SeicentoBilling/ {
            proxy_pass            http://billing-stream/SeicentoBilling/;

            proxy_redirect        default;

            proxy_set_header      Host $host;
            proxy_set_header      X-Real-IP $remote_addr;
            proxy_set_header      X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header      X-Forwarded-Host $server_name;
        }

        # Jasper
        location /jasper/ {
            proxy_pass            http://127.0.0.1:17080/jasperserver/;
            #proxy_pass            http://jasper-stream/jasperserver/;

            proxy_redirect        default;

            proxy_set_header      Host $host;
            proxy_set_header      X-Real-IP $remote_addr;
            proxy_set_header      X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header      X-Forwarded-Host $server_name;
        }
    }
}
```

```
# redirect Root
location / {
    proxy_pass          http://billing-stream/SeicentoBilling/;
    proxy_redirect      default;

    proxy_set_header    Host $host;
    proxy_set_header    X-Real-IP $remote_addr;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Host $server_name;
}

}
```

3.3.2. Verschlüsselung mit letsencrypt

<https://www.humankode.com/ssl/how-to-set-up-free-ssl-certificates-from-lets-encrypt-using-docker-and-nginx>