Installationsanleitung Pseudonymisierungsdienst mit Transaction ID

Im folgenden werden die Schritte aufgelistet, welche die benötigte Software installieren, um den Pseudonymiserungsdienst zu installieren und betreiben. Alle Schritte dieser Anleitung werden als Root unter RedHat vorgenommen. Bei anderen Distributionen können die Befehle abweichen. Hier verwendete Benutzernamen und Passwörter, abseits von "Root" sind als Platzhalter zu betrachten und durch eigens erzeugte zu ersetzen. Auch sollte immer auf Versionsnummern in den Befelhen geachtet werden, diese können sich im Laufe der Zeit ändern (Z.B.: Neue Version von OpenJDK 8).

1. Postgresql-Installation und setup:
    a. **Postgresql 1**

```
yum -y update

yum install https://download.postgresql.org/pub/repos/yum/11/redhat/rhel-7-x86_64/pgdg-redhat11-11-2.noarch.rpm

yum -y install postgresql11

yum -y install postgresql11-server

yum -y install postgresql11-contrib
```

    b. Postgresql initialiserien:

```
/usr/pgsql-11/bin/postgresql-11-setup initdb
```

    c. Postgressql service aktivieren:

```
systemctl enable postgresql-11
```

    d. Postgresql service starten:

```
systemctl start postgresql-11
```

    e. UUID-Extenssion installieren:
        i.

```
su postgres

psql
```

        ii. in psql-bash:

```
create database pseudonymizationservice;

create user pseudouser;

alter user pseudouser with encrypted password 'qwertz';

grant all privileges on database pseudonymizationservice to pseudouser;

\c pseudonymizationservice

CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
```

    f. Konfiiguration für die Remotverbindung durch PG-Admin:
        i. In postgresql.conf folgende Änderung vornehmen (Pfad: /var/lib/pgsql/11/data/postgresql.conf):

postgresql.con

```
listen_addresses = '*'

Performance-Optimierung (Angaben sind Richtwerte, man sollte nicht vergessen,
dass der Pseudonymiserungsdienst und der Tomcat auch Speicher brauchen):

"shared_buffers" sollte auf 25% des genutzen RAMs stehen (Mehr RAM, mehr Buffer)
"effective_cache_size" auf bis zu 50% des System-RAMs
"work_mem" = Höherer Wert für Abfrage effizienter (Join etc.)
```

ii. Remote Zugriff ermöglichen:

(HBA_CONF) /var/lib/pgsql/11/data/pg_hba.conf

```
In pg_hba.conf Änderungen vornehmen. Es müüsen die zugriffbereichtigten IP-Adressen
eingestellt werden. Freigabe aller IP-Adressen erfolgt mit der IP "0.0.0.0/0 ".
METHOD muss auf TRUST gesetzt werden.

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     ident
# IPv4 local connections:
host    all             all             134.106.48.64/32        trust
host    all             all             127.0.0.1/32            trust
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     ident
# IPv4 local connections:
host    all             all             134.106.48.64/32        trust
host    all             all             127.0.0.1/32            trust
```

iii. Postgresql neustarten mit:

```
systemctl restart postgresql-11
```

iv. Firewall freigeben:

```
firewall-cmd --permanent --zone=public --add-port=5432/tcp
firewall-cmd --permanent --zone=public --add-port=5432/udp
firewall-cmd --reload
```

g. Tabellen in der Datenbank:

i. **API-Key-Table**

```
CREATE TABLE api_key_table
(
  owner character varying(255) NOT NULL,
  api_key character varying(255) NOT NULL,
  CONSTRAINT owner_key PRIMARY KEY (owner, api_key)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE api_key_table
  OWNER TO pseudouser;
```

ii. **PID-Table**

```
CREATE TABLE pid_table
(
  pid character varying(255) NOT NULL,
  psn character varying(255),
  CONSTRAINT pid_key PRIMARY KEY (pid)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE pid_table
  OWNER TO pseudouser;
```

iii. **Reserve-TempID-Table**

```
CREATE TABLE reserve_temp_id
(
  temp_id character varying(255) NOT NULL DEFAULT ('TMP_ID_'::text || uuid_generate_v4()),
  "row" bigint,
  CONSTRAINT reserve_temp_id_pkey PRIMARY KEY (temp_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE reserve_temp_id
  OWNER TO pseudouser;
```

iv. **Temp_id_pool Table**

```
CREATE TABLE temp_id_pool
(
  temp_id character varying(255) NOT NULL,
  "row" bigint,
  CONSTRAINT temp_id_pool_key PRIMARY KEY (temp_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE temp_id_pool
  OWNER TO pseudouser;
```

v. **TempID-Table**

```
CREATE TABLE temp_id_table
(
  tempid character varying(255) NOT NULL,
  pid character varying(255),
  transaction_id character varying(255),
  "timestamp" timestamp without time zone,
  CONSTRAINT tempid_key PRIMARY KEY (tempid),
  CONSTRAINT tempid_unique UNIQUE (tempid)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE temp_id_table
  OWNER TO pseudouser;
```

vi. **Completed PSN Transaction Table**

```
CREATE TABLE completed_psn_reception
(
  completed_transaction_id character varying(255) NOT NULL,
  delete_straight_away boolean,
  completed_timestamp timestamp without time zone,
  temp_ids_deleted_timestamp timestamp without time zone,
  temp_ids_deleted boolean,
  CONSTRAINT id_key PRIMARY KEY (completed_transaction_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE completed_psn_reception
  OWNER TO pseudouser;
```

vii. **Received Transactions Table**

```
CREATE TABLE received_transactions_table
(
  transaction_id character varying(255) NOT NULL,
  id_received_timestamp timestamp without time zone,
  transaction_completed boolean,
  transaction_completed_timestamp timestamp without time zone,
  CONSTRAINT trans_id_key PRIMARY KEY (transaction_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE received_transactions_table
  OWNER TO pseudouser;
```

h. Reserve tempid einfügen (Die Tabelle temp_id_pool will immer mit 20 Millionen IDs gefüllt sein, es empfihiehlt sich immer ein Mehrfaches davon als Reserve zu generieren):

i. **Reserve_TempID-Insert**

```
INSERT INTO reserve_temp_id("row") SELECT * FROM generate_series(1, 100000000);
```

i. Temp_id_pool füllen (Muss nur intial gemacht werden, das erledigt im betrieb ein Hintergrund Thread):

```
INSERT INTO temp_id_pool SELECT * FROM reserve_temp_id LIMIT 20000000

DELETE FROM reserve_temp_id USING temp_id_pool WHERE (reserve_temp_id.temp_id = temp_id_pool.
temp_id)
```

2. Java-Installation (Bitte Befehle an konkrete aktuelle Java Version anpassen):

a.
```
yum -y install java-1.8.0-openjdk.x86_64 java-1.8.0-openjdk-devel.x86_64

update-alternatives --config java
```

b. HOME_PATH setzen:

i.
```
nano /etc/environment
```

ii. folgendes einfügen: JAVA_HOME="/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7_6.x86_64/jre"

iii. Anschließend speichern und Editor beenden

iv.
```
nano ~/.bash_profile
```

v. folgendes einfügen: export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7_6.x86_64/jre export PATH=$JAVA_HOME/bin:$PATH

vi. Anschließend speichern und Editor beenden

vii. Bash-Befehle neu laden:

viii.
```
source ~/.bash_profile
```

ix. JAVA_HOME überprüfen:echo $JAVA_HOME

3. Tomcat-Installation:
   a. Tomcat-User anlegen, Tomcat herunterladen und installieren

```
useradd -m -U -d /opt/tomcat -s /bin/false tomcat

cd /tmp

wget https://www-eu.apache.org/dist/tomcat/tomcat-9/v9.0.19/bin/apache-tomcat-9.0.19.tar.gz

tar -xf apache-tomcat-9.0.19.tar.gz

mv apache-tomcat-9.0.19 /opt/tomcat/

ln -s /opt/tomcat/apache-tomcat-9.0.19 /opt/tomcat/latest

chown -R tomcat: /opt/tomcat

chmod +x /opt/tomcat/latest/bin/*.sh
```

   b. nano /etc/systemd/system/tomcat.service Inhalt:

tomcat.service

```
[Unit]
Description=Tomcat 9 servlet container
After=network.target

[Service]
Type=forking

User=tomcat
Group=tomcat

Environment="JAVA_HOME=/usr/lib/jvm/jre"
Environment="JAVA_OPTS=-Djava.security.egd=file:///dev/urandom"

Environment="CATALINA_BASE=/opt/tomcat/latest"
Environment="CATALINA_HOME=/opt/tomcat/latest"
Environment="CATALINA_PID=/opt/tomcat/latest/temp/tomcat.pid"
Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"

ExecStart=/opt/tomcat/latest/bin/startup.sh
ExecStop=/opt/tomcat/latest/bin/shutdown.sh

[Install]
WantedBy=multi-user.target
```

   c. Seytem-service daemon neustraten, Tomcat Service aktivieren und starten

```
systemctl daemon-reload
systemctl enable tomcat
systemctl start tomcat
```

   d. Tomcat-User anlegen

```
nano /opt/tomcat/latest/conf/tomcat-users.xml
```

Beispiel:

tomcat-users.xml

```
<tomcat-users>
<!--
    Comments
-->
   <role rolename="admin-gui"/>
   <role rolename="manager-gui"/>
   <user username="admin" password="admin_password" roles="admin-gui,manager-gui"/>
</tomcat-users>
```

e. Tomcat-Manager von überall erreichbar machen (`<tomcat>/webapps/manager/META-INF/context.xml`):

context.xml

```
<Context antiResourceLocking="false" privileged="true" >
<!--
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
-->
</Context>
```

f. Nach Änderungenan Tomcat-Einstellungen immer den Server neu starten:

```
systemctl restart tomcat
```

4. Proxy-Einstellung für TLS-Verschlüsselung per Apache-Proxy: Apache
   a. Apache installieren:

```
yum -y install httpd
```

   b. mod_ssl installieren:

```
yum -y install mod_ssl openssl
```

   c. Portfreigaben (Beispiel, hängen von den jeweils verwendeten ports ab):

```
firewall-cmd --permanent --add-port=80/tcp
firewall-cmd --permanent --add-port=443/tcp

firewall-cmd --reload
```

   d. Beispiel einer ssl.conf (/etc/httpd/conf.d/*ssl.conf*):

SSL.conf

```
#
# When we also provide SSL we have to listen to the
# the HTTPS port in addition.
#
Listen 8443

##
##  SSL Global Context
##
```

```
##  All SSL configuration in this context applies both to
##  the main server and all SSL-enabled virtual hosts.
##

#   Pass Phrase Dialog:
#   Configure the pass phrase gathering process.
#   The filtering dialog program (`builtin' is a internal
#   terminal dialog) has to provide the pass phrase on stdout.
SSLPassPhraseDialog exec:/usr/libexec/httpd-ssl-pass-dialog

#   Inter-Process Session Cache:
#   Configure the SSL Session Cache: First the mechanism
#   to use and second the expiring timeout (in seconds).
SSLSessionCache         shmcb:/run/httpd/sslcache(512000)
SSLSessionCacheTimeout  300

#   Pseudo Random Number Generator (PRNG):
#   Configure one or more sources to seed the PRNG of the
#   SSL library. The seed data should be of good random quality.
#   WARNING! On some platforms /dev/random blocks if not enough entropy
#   is available. This means you then cannot use the /dev/random device
#   because it would lead to very long connection times (as long as
#   it requires to make more entropy available). But usually those
#   platforms additionally provide a /dev/urandom device which doesn't
#   block. So, if available, use this one instead. Read the mod_ssl User
#   Manual for more details.
SSLRandomSeed startup file:/dev/urandom  256
SSLRandomSeed connect builtin
#SSLRandomSeed startup file:/dev/random  512
#SSLRandomSeed connect file:/dev/random  512
#SSLRandomSeed connect file:/dev/urandom 512

#
# Use "SSLCryptoDevice" to enable any supported hardware
# accelerators. Use "openssl engine -v" to list supported
# engine names.  NOTE: If you enable an accelerator and the
# server does not start, consult the error logs and ensure
# your accelerator is functioning properly.
#
SSLCryptoDevice builtin
#SSLCryptoDevice ubsec

##
## SSL Virtual Host Context
##

<VirtualHost pseudoservice01.virt.uni-oldenburg.de:8443>

# General setup for the virtual host, inherited from global configuration
DocumentRoot "/opt/tomcat/webapps/PseudonymizationService"
ServerName pseudoservice01.virt.uni-oldenburg.de

<Directory "opt/tomcat/">
    AllowOverride All
    Options All
    Require all granted
    Options Indexes FollowSymLinks
</Directory>

ProxyPass /PseudonymizationService/PSD http://localhost:8080/PseudonymizationService/PSD


ProxyPassReverse /Pseudonymizationservice/PSD http://localhost:8080/PseudonymizationService/PSD


# Use separate log files for the SSL virtual host; note that LogLevel
# is not inherited from httpd.conf.
ErrorLog logs/ssl_error_log
TransferLog logs/ssl_access_log
LogLevel warn
```

```
#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on
# SSLProxyEngine on

#   SSL Protocol support:
# List the enable protocol levels with which clients will be able to
# connect.  Disable SSLv2 access by default:
SSLProtocol all -SSLv2 -SSLv3

#   SSL Cipher Suite:
#   List the ciphers that the client is permitted to negotiate.
#   See the mod_ssl documentation for a complete list.
   SSLCipherSuite HIGH:3DES:!aNULL:!MD5:!SEED:!IDEA

#   Speed-optimized SSL Cipher configuration:
#   If speed is your main concern (on busy HTTPS servers e.g.),
#   you might want to force clients to specific, performance
#   optimized ciphers. In this case, prepend those ciphers
#   to the SSLCipherSuite list, and enable SSLHonorCipherOrder.
#   Caveat: by giving precedence to RC4-SHA and AES128-SHA
#   (as in the example below), most connections will no longer
#   have perfect forward secrecy - if the server's key is
#   compromised, captures of past or future traffic must be
#   considered compromised, too.
#SSLCipherSuite RC4-SHA:AES128-SHA:HIGH:MEDIUM:!aNULL:!MD5
#SSLHonorCipherOrder on

#   Server Certificate:
# Point SSLCertificateFile at a PEM encoded certificate.  If
# the certificate is encrypted, then you will be prompted for a
# pass phrase.  Note that a kill -HUP will prompt again.  A new
# certificate can be generated using the genkey(1) command.
SSLCertificateFile /etc/pki/tls/certs/pseudoservice01.virt.uni-oldenburg.de.crt

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile /etc/pki/tls/private/pseudoservice01.virt.uni-oldenburg.de.key

#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
#   concatenation of PEM encoded CA certificates which form the
#   certificate chain for the server certificate. Alternatively
#   the referenced file can be the same as SSLCertificateFile
#   when the CA certificates are directly appended to the server
#   certificate for convinience.
SSLCertificateChainFile /etc/pki/tls/certs/server-chain-g2.crt

#   Certificate Authority (CA):
#   Set the CA certificate verification path where to find CA
#   certificates for client authentication or alternatively one
#   huge file containing all of them (file must be PEM encoded)
#SSLCACertificateFile /etc/pki/tls/certs/ca-bundle.crt

#   Client Authentication (Type):
#   Client certificate verification type and depth.  Types are
#   none, optional, require and optional_no_ca.  Depth is a
#   number which specifies how deeply to verify the certificate
#   issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth  10

#   Access Control:
#   With SSLRequire you can do per-directory access control based
#   on arbitrary complex boolean expressions containing server
#   variable checks and other lookup directives.  The syntax is a
#   mixture between C and Perl.  See the mod_ssl documentation
#   for more details.
```

```
#<Location />
#SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)/ \
#            and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
#            and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
#            and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
#            and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20       ) \
#           or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
#</Location>

#   SSL Engine Options:
#   Set various options for the SSL engine.
#   o FakeBasicAuth:
#     Translate the client X.509 into a Basic Authorisation.  This means that
#     the standard Auth/DBMAuth methods can be used for access control.  The
#     user name is the `one line' version of the client's X.509 certificate.
#     Note that no password is obtained from the user. Every entry in the user
#     file needs this password: `xxj31ZMTZzkVA'.
#   o ExportCertData:
#     This exports two additional environment variables: SSL_CLIENT_CERT and
#     SSL_SERVER_CERT. These contain the PEM-encoded certificates of the
#     server (always existing) and the client (only existing when client
#     authentication is used). This can be used to import the certificates
#     into CGI scripts.
#   o StdEnvVars:
#     This exports the standard SSL/TLS related `SSL_*' environment variables.
#     Per default this exportation is switched off for performance reasons,
#     because the extraction step is an expensive operation and is usually
#     useless for serving static content. So one usually enables the
#     exportation for CGI and SSI requests only.
#   o StrictRequire:
#     This denies access when "SSLRequireSSL" or "SSLRequire" applied even
#     under a "Satisfy any" situation, i.e. when it applies access is denied
#     and no other module can change it.
#   o OptRenegotiate:
#     This enables optimized SSL connection renegotiation handling when SSL
#     directives are used in per-directory context.
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
<Files ~ "\.(cgi|shtml|phtml|php3?)$">
    SSLOptions +StdEnvVars
</Files>
<Directory "/var/www/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>

#   SSL Protocol Adjustments:
#   The safe and default but still SSL/TLS standard compliant shutdown
#   approach is that mod_ssl sends the close notify alert but doesn't wait for
#   the close notify alert from client. When you need a different shutdown
#   approach you can use one of the following variables:
#   o ssl-unclean-shutdown:
#     This forces an unclean shutdown when the connection is closed, i.e. no
#     SSL close notify alert is send or allowed to received.  This violates
#     the SSL/TLS standard but is needed for some brain-dead browsers. Use
#     this when you receive I/O errors because of the standard approach where
#     mod_ssl sends the close notify alert.
#   o ssl-accurate-shutdown:
#     This forces an accurate shutdown when the connection is closed, i.e. a
#     SSL close notify alert is send and mod_ssl waits for the close notify
#     alert of the client. This is 100% SSL/TLS standard compliant, but in
#     practice often causes hanging connections with brain-dead browsers. Use
#     this only for browsers where you know that their SSL implementation
#     works correctly.
#   Notice: Most problems of broken clients are also related to the HTTP
#   keep-alive facility, so you usually additionally want to disable
#   keep-alive for those clients, too. Use variable "nokeepalive" for this.
#   Similarly, one has to force some clients to use HTTP/1.0 to workaround
#   their broken HTTP/1.1 implementation. Use variables "downgrade-1.0" and
#   "force-response-1.0" for this.
BrowserMatch "MSIE [2-5]" \
         nokeepalive ssl-unclean-shutdown \
         downgrade-1.0 force-response-1.0
```

```
#   Per-Server Logging:
#   The home of a custom SSL log file. Use this when you want a
#   compact non-error SSL logfile on a virtual host basis.
CustomLog logs/ssl_request_log \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>
```

HTTPD Service aktivieren und starten

```
systemctl enable httpd
systemctl start httpd
```

e. Falls Kommunikation zwischen Apache und Tomcat mit 503 fehlschlägt:

```
/usr/sbin/setsebool -P httpd_can_network_connect true
```

5. Allgemeine Hinweise zur Erstellung und Betrieb des Dienstes:
   a. Build-Reihenfolge: PseudonymizationTool und dann PseudonymizationService
   b. Rollen für Die Authentifizierungsschlüssel:

Rollen

```
dataOwner -> Dateneigner
dataReceiver -> Datenempfänger
```

   c. ...