Global Enablement & Learning

Configure Db2 Access From SAS Viya

Introduction

This document is about configuring SAS/ACCESS to Db2 (including Data Connector for Db2) on SAS Viya.

In the following instructions, pay attention each time to the user to be used for running linux commands: "as `jumpuser`" (a sudoer) or "as `sas`". It is important in the process.

These instructions use sample paths. Please adapt them to your environment.

We assume these instructions are run on a machine connected to the Kubernetes cluster holding SAS Viya. This could be a Kubernetes client machine or a jump host. This is defined as "SAS Viya jump host" in this instructions.

Optional: Deploy Db2 Community Edition Docker Image

To be able to test access to Db2 from SAS Viya, we need a Db2 database. If you don't have one, you can install one following these steps.

NB: Instructions are adapted from https://www.ibm.com/docs/en/db2/11.5?topic=system-linux.

Requirement: docker installed

On your SAS Viya jump host, as `jumpuser`:

```
cd
mkdir db2
cd db2
docker pull icr.io/db2_community/db2

tee .env_list > /dev/null << EOF
LICENSE=accept
DB2INSTANCE=db2inst1
DB2INST1_PASSWORD=lnxsas
DBNAME=sasdb
BLU=false
ENABLE_ORACLE_COMPATIBILITY=false
UPDATEAVAIL=NO
TO_CREATE_SAMPLEDB=false
REPODB=false
IS_OSXFS=false
PERSISTENT_HOME=true
HADR_ENABLED=false
ETCD_ENDPOINT=
ETCD_USERNAME=
ETCD_PASSWORD=
EOF

docker run \
   -h db2server \
   --name db2server \
   --restart=always \
   --detach \
   --privileged=true \
   -p 50000:50000 \
   --env-file .env_list \
   -v ${PWD}/database:/database \
   icr.io/db2_community/db2

docker ps | grep db2server
```

You should have now a running Db2 instance in a container on your jump host.

Wait a minute or two before connecting to the container.

Let's connect to it and setup a sample database:

```
docker exec -ti db2server bash -c "su - db2inst1"
```

You are inside the container.

Run the following commands, one at a time. If the first one fails (`db2sampl`), wait a little bit (the database might not be ready yet) and run it again:

```
db2sampl -force -sql
db2 connect to sample
db2 "select * from department"
db2 connect reset
```

Exit from the container:

```
exit
```

Install the Db2 Client on the Jump Host

Review the "Configuring SAS/ACCESS and Data Connectors for SAS Viya 4" README File

This README file is available in the https://my.sas.com portal for your order or in your deployments assets once you have extracted them ($deploy/sas-bases/examples/data-access/README.md)

Review the "*SAS/ACCESS Interface to DB2*" section.

Download the Db2 Client

We need to download the Db2 client.

NB: Instructions are adapted from https://www.ibm.com/docs/en/db2/11.5?topic=clients-installing-data-server-unix-linux.

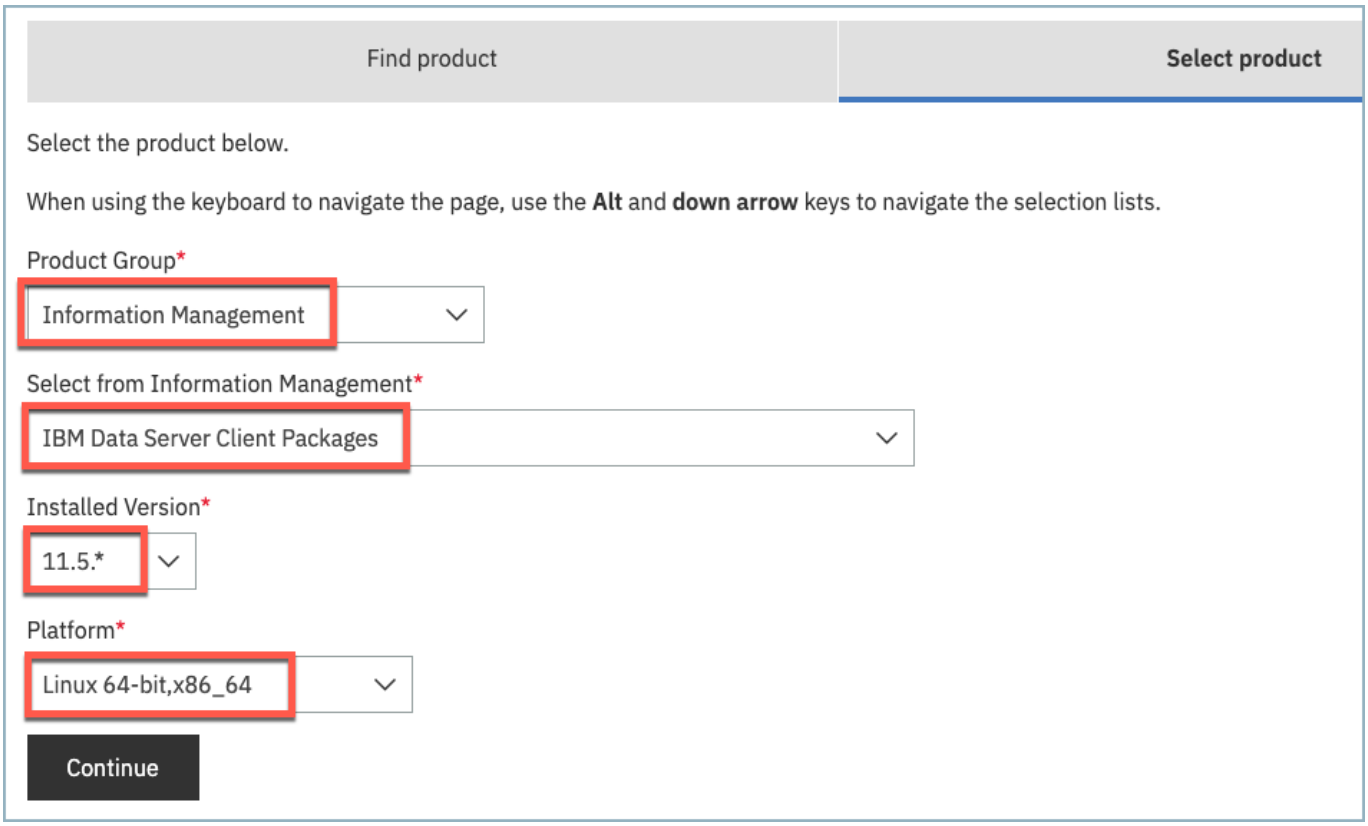
Open https://www.ibm.com/support/fixcentral/.

Login with your IBM profile.

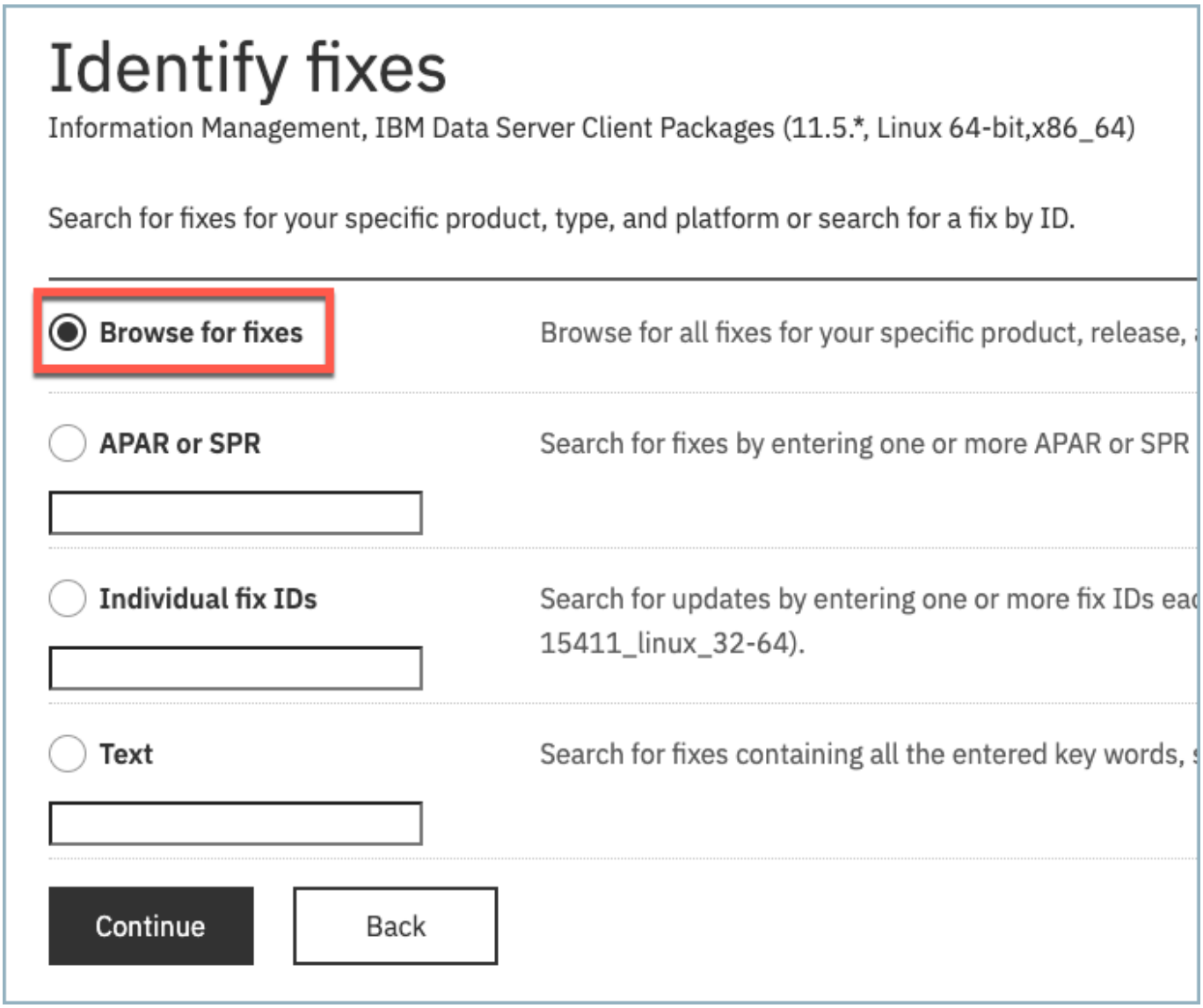
Select:

- Product Group: Information Management
- Select from Information Management: IBM Data Server Client Packages
- Installed Version: 11.5.*
- Platform: Linux 64-bit, x86_64

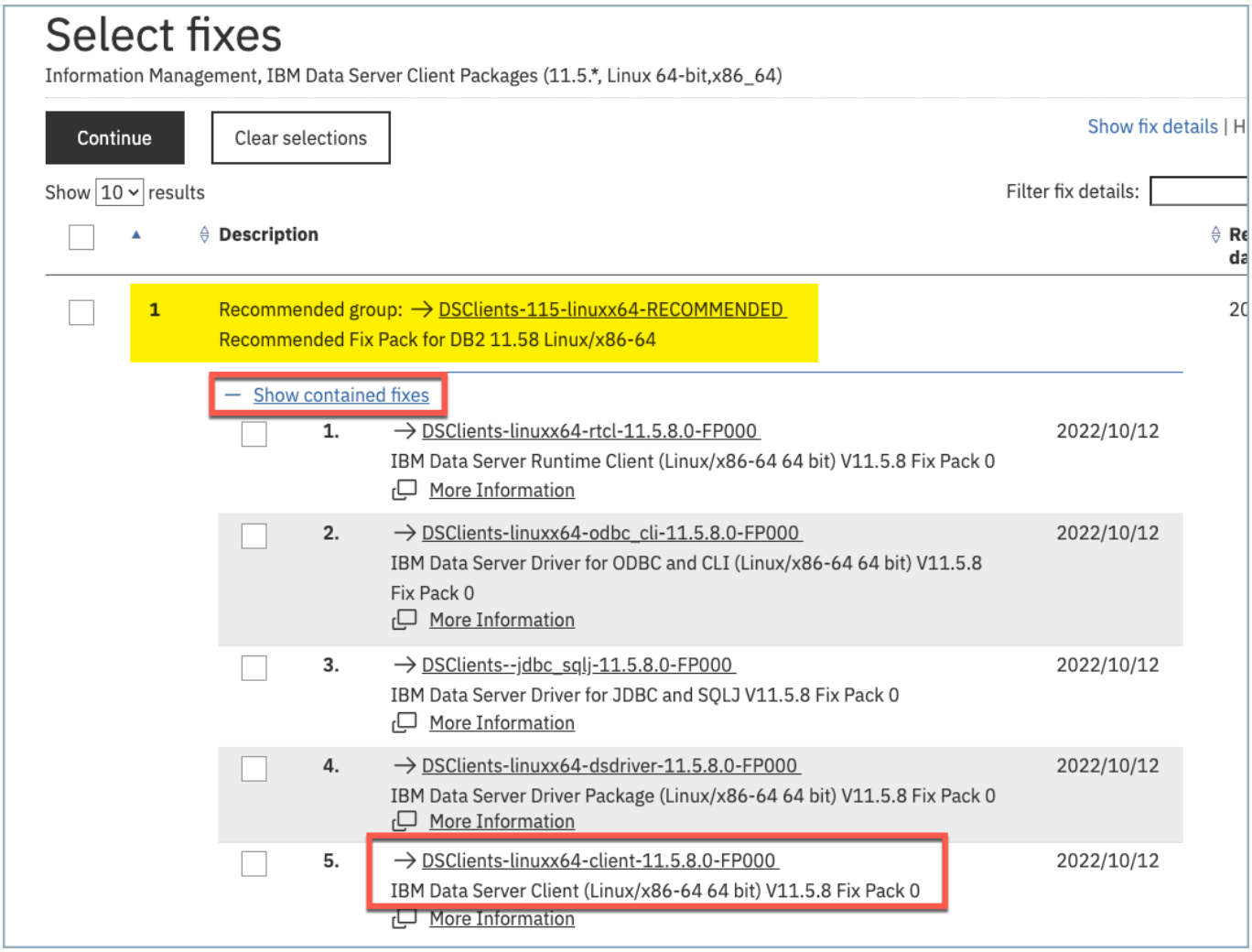
And click "Continue".



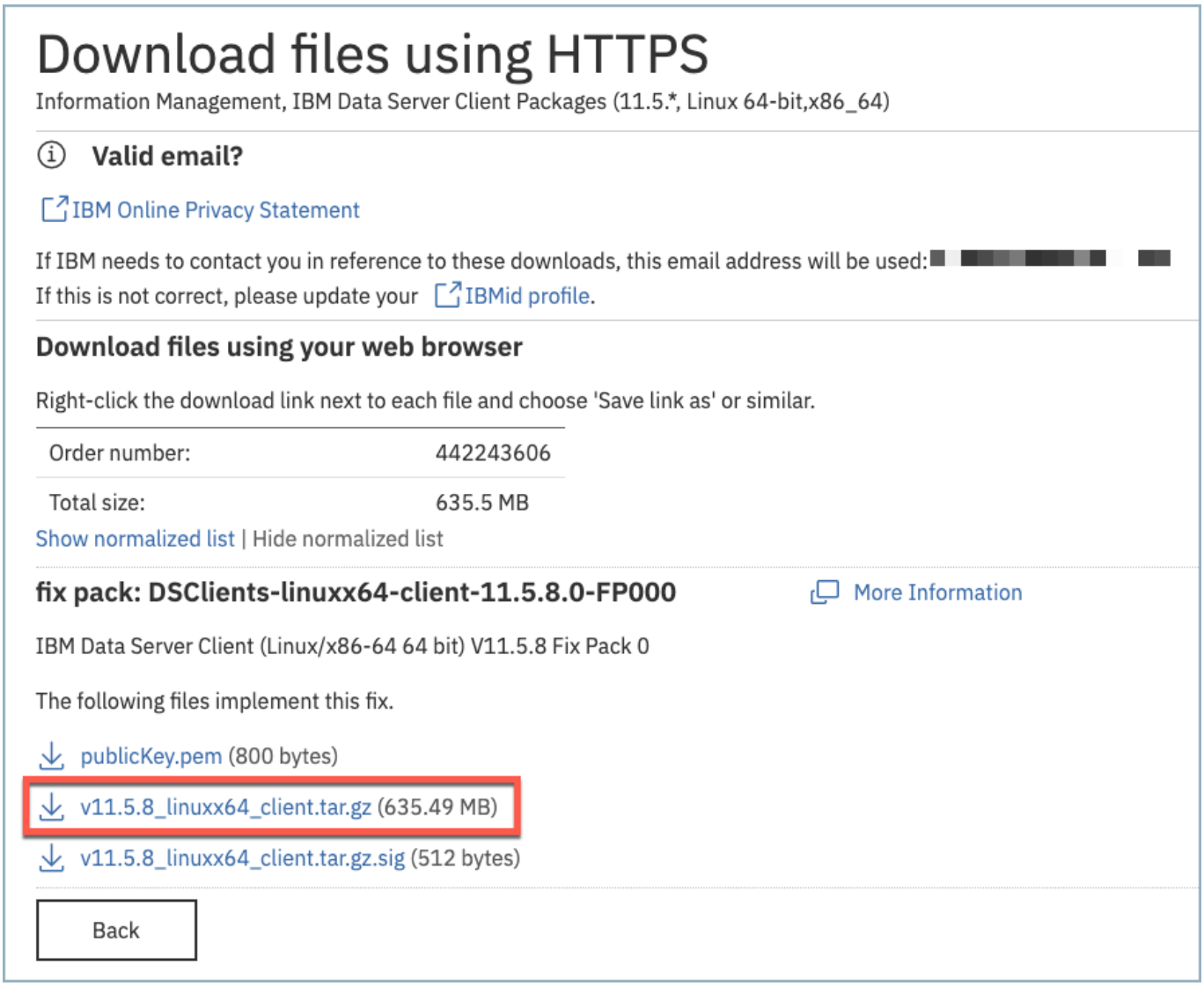
Select "Browse for fixes" and "Continue":



Expand the recommended group ("Show contained fixes") and identify the "IBM Data Server Client":



Finally click on the "v11.5.8_linuxx64_client.tar.gz" to download it:



Once downloaded, make it available on your "SAS Viya jump host" in `/tmp`.

Create a SAS User

Create (or reuse) a system user that has a uid and gid value of "1001". The name "sas" is not essential but will be used in these instructions. Replace by the username you chose. Uid and gid are important.

On your "SAS Viya jump host", as `jumpuser`:

```
sudo groupadd -g 1001 sas
sudo useradd -u 1001 -g 1001 sas -m
sudo passwd sas
# Choose a password for this account
```

Install Db2 Client as SAS user

On your "SAS Viya jump host", log in as `sas`, untar the client (uploaded previously in /tmp) in the `sas` home directory:

```
cd
tar -xvf /tmp/v*linuxx64_client.tar.gz
```

Install the Db2 client silently:

```
cd
./client/db2_install -b $HOME/sqllib -f sysreq -y
```

NB: `-f sysreq` is used to not check the minimum requirements on our sandbox since CentOS is not up-to-date.

Configure and Validate Db2 Client against Db2 Community Edition

On your "SAS Viya jump host", as `sas`, source the Db2 client profile:

```
. $HOME/sqllib/db2profile
```

Define the Db2 Community Edition database and dsn running locally (setup earlier):

```
db2cli writecfg add -database sample -host $(hostname -f) -port 50000
db2cli writecfg add -dsn sampledsn -database sample -host $(hostname -f) -
port 50000

cat $HOME/sqllib/cfg/db2dsdriver.cfg
```

Validate access to this dsn:

```
db2cli validate -dsn sampledsn -connect -user db2inst1 -passwd lnxsas
```

Make the Db2 Client Accessible from SAS Viya

SAS/ACCESS Configuration Note

Some SAS/ACCESS engines on SAS Compute and Data Connectors on CAS require a shared persistent location (usually NFS) for holding database client files.

If you haven't done so in your SAS Viya environment, you need to setup that location.

See SAS Viya deployment README file (https://my.sas.com/) and samples:

- `$deploy/sas-bases/examples/sas-compute-server/configure/compute-server-add-nfs-mount.yaml`

- `$deploy/sas-bases/examples/cas/configure/cas-add-nfs-mount.yaml`

In our sandbox, a single NFS shared directory is mounted on both the Compute Server (compute-server-add-nfs-mount.yaml) and the CAS nodes (cas-add-nfs-mount.yaml). This is easier for maintenance (only one place to maintain the clients).

Then, once you have setup the shared persistent location for both CAS and the SAS Compute Server, you need to set environment variables required by some engines.

This is also described in the SAS Viya deployment README file (https://my.sas.com/) and samples:

- `$deploy/sas-bases/examples/data-access/README.md`
- `$deploy/sas-bases/examples/sas-access.properties`

Mount the Location of SAS/ACCESS Clients

You need to mount this NFS directory on your "SAS Viya jump host" EXACTLY (same path) as it is mounted on both the SAS Compute Server and the CAS nodes.

For example, if your `compute-server-add-nfs-mount.yaml` contains:

```
- op: add
  path:  /template/spec/volumes/-
  value:
    name: sas-viya-geldm-volume
    nfs:
      path: /shared/gelcontent
      server: my.nfs.server.com
- op: add
  path: /template/spec/containers/0/volumeMounts/-
  value:
    name: sas-viya-geldm-volume
    mountPath: /gelcontent
```

Then you will mount `my.nfs.server.com:/shared/gelcontent` as `/gelcontent` on the jump host.

On your "SAS Viya jump host", as `jumpuser`, prepare the target folder:

```
sudo mkdir -p /gelcontent
```

Mount the NFS server directory on `/gelcontent`:

```
sudo mount -t nfs my.nfs.server.com:/shared/gelcontent /gelcontent
```

Check if you see the NFS contents:

```
ls -al /gelcontent
```

Prepare some Directories for the Shared Db2 Client

On your "SAS Viya jump host", as `jumpuser`:

```
mkdir -p /gelcontent/access-clients/db2client
mkdir -p /gelcontent/access-clients/db2
sudo chmod -R 755 /gelcontent/access-clients/db2client
sudo chmod -R 755 /gelcontent/access-clients/db2
```

Assign these directories to the `sas` (or 1001 uid) user:

```
sudo chown sas:sas /gelcontent/access-clients/db2client
sudo chown sas:sas /gelcontent/access-clients/db2
```

Satisfy Requirements for Upcoming Tasks

On your "SAS Viya jump host", as `jumpuser`:

```
# perl-env will be needed by Db2 db2ccprf utility
sudo yum install perl-Env -y
```

Copy Db2 Client Files from the Jump Host to the Shared Location

On your "SAS Viya jump host", as `sas`:

```
cp -R -L $HOME/sqllib /gelcontent/access-clients/db2client/sqllib
chmod -R 755 /gelcontent/access-clients/db2client/
```

Adjust the Network-Mounted Client

On your "SAS Viya jump host", as `sas`, modify db2profile to take into account the new paths:

```
export DB2_NET_CLIENT_PATH=/gelcontent/access-clients/db2client/sqllib

sed -i 's|^DB2DIR=.*|DB2DIR='$DB2_NET_CLIENT_PATH'|g'
$DB2_NET_CLIENT_PATH/db2profile
sed -i 's|^INSTHOME=.*|INSTHOME=/gelcontent/access-clients/db2|g'
$DB2_NET_CLIENT_PATH/db2profile
```

```
# DB2INSTANCE should already be set to the user who installed the client
(sas for us)
# DB2INSTANCE=<user-who_installed-db2>
```

Check the `$DB2_NET_CLIENT_PATH/db2profile` file.

On your "SAS Viya jump host", as `sas`, copy the client profile:

```
source $DB2_NET_CLIENT_PATH/db2profile
export DB2_APPL_DATA_PATH=/gelcontent/access-clients/db2
export DB2_APPL_CFG_PATH=/gelcontent/access-clients/db2
$DB2_NET_CLIENT_PATH/bin/db2ccprf -f -t /gelcontent/access-clients/db2
chmod -R 755 /gelcontent/access-clients/db2
```

Check the `/gelcontent/access-clients/db2` directory:

```
ls -al /gelcontent/access-clients/db2
```

Update SAS Viya

NB: There are multiple ways to deploy SAS Viya (https://communities.sas.com/t5/SAS-Communities-Library/New-SAS-Viya-Deployment-Methods/ta-p/856206). Adapt the following instructions based on your deployment method. They are to be run in the environment that was used to deploy SAS Viya (jump host or Kubernetes client, etc.).

First, see SAS/ACCESS Configuration Note.

Then, on your "SAS Viya jump host", as `jumpuser`, create/modify the `sas-access.properties` file in your SAS Viya deployment directory:

```
# cd $deploy/site-config/data-access
cd ~/project/deploy/gelenv/site-config/data-access

DB2CLIENTPATH=/gelcontent/access-clients/db2client/sqllib
DB2PATH=/gelcontent/access-clients/db2

tee sas-access-db2.properties > /dev/null << EOF

##########################
# SAS/ACCESS to DB2
##########################
CUR_INSTHOME=
CUR_INSTNAME=
DASWORKDIR=
DB2DIR=${DB2CLIENTPATH}
DB2INSTANCE=sas
DB2LIB=${DB2CLIENTPATH}
```

```
DB2_HOME=${DB2CLIENTPATH}
DB2_NET_CLIENT_PATH=${DB2CLIENTPATH}
IBM_DB_DIR=${DB2CLIENTPATH}
IBM_DB_HOME=${DB2CLIENTPATH}
IBM_DB_INCLUDE=${DB2CLIENTPATH}
IBM_DB_LIB=${DB2CLIENTPATH}
INSTHOME=${DB2PATH}
INST_DIR=${DB2CLIENTPATH}
PREV_DB2_PATH=
DB2=${DB2CLIENTPATH}/lib64:${DB2CLIENTPATH}/lib64/gskit:${DB2CLIENTPATH}/l
ib32
DB2_BIN=${DB2CLIENTPATH}/bin:${DB2CLIENTPATH}/adm:${DB2CLIENTPATH}/misc

EOF

cat sas-access-db2.properties >> sas-access.properties
```

NB: the `DB2INSTANCE` variable is purposely set to `sas` which is the predefined user that SAS Viya uses behind the scenes. It is mapped automatically to the uid=1001 created earlier, regardless of the username chosen in step Create a SAS User. So the value must be `sas`, not the username created earlier if it is different than `sas`.

Check the `sas-access.properties` file:

```
cat sas-access-db2.properties
```

You are ready to update SAS Viya deployment. Go into your SAS Viya deployment directory and run the following commands:

```
# cd $deploy
cd ~/project/deploy/gelenv/

# Rebuild
kustomize build -o site.yaml

# Apply cluster-api resources to the cluster. As an administrator with
cluster permissions, run
kubectl apply --selector="sas.com/admin=cluster-api" --server-side --
force-conflicts -f site.yaml

kubectl wait --for condition=established --timeout=60s -l
"sas.com/admin=cluster-api" crd

# As an administrator with cluster permissions, run
kubectl apply --selector="sas.com/admin=cluster-wide" -f site.yaml

#As an administrator with local cluster permissions, run
kubectl apply --selector="sas.com/admin=cluster-local" -f site.yaml --
prune
```

```
#As an administrator with namespace permissions, run
kubectl apply --selector="sas.com/admin=namespace" -f site.yaml --prune

# Restart CAS
kubectl delete pods -l casoperator.sas.com/server=default

# Wait for CAS to be ready
kubectl wait --for=condition=ready pod -l
casoperator.sas.com/server=default --timeout=600s
```

If you have an existing SAS Studio Compute session already started, sign-out from SAS Studio and sign back in.

Test SAS/ACCESS to Db2

In SAS Studio, run the following program, step-by-step, to check SAS/ACCESS to Db2 features:

```
/* Check some environment variables */
%put %sysget(DB2) ;
%put %sysget(INSTHOME) ;
%put %sysget(INST_DIR) ;

/*** SAS Compute Server ***/

/* Native Db2 */

/* sampledsn is the dsn defined earlier in db2dsdriver.cfg */
libname ndb2 db2 database="sampledsn" user="db2inst1" password="lnxsas" ;

/* Other syntaxes */
/*
libname ndb2 db2 noprompt="DSN=sampledsn;UID=db2inst1;PWD=lnxsas;" ;
libname ndb2 db2
noprompt="Hostname=db2server;Database=sample;ServiceName=50000;UID=db2inst
1;PWD=lnxsas;" ;
*/

/* List Db2 tables */
proc datasets lib=ndb2 ;
quit ;

/* Read Db2 table */
data emp ;
   set ndb2.emp ;
run ;

/* Write Db2 table */
/* Drop target if exists */
proc datasets lib=ndb2 nowarn nolist ;
   delete from_sas ;
```

```
quit ;
data ndb2.from_sas ;
   set sashelp.prdsale ;
run ;

/* List Db2 tables */
/* Check FROM_SAS presence */
proc datasets lib=ndb2 ;
quit ;

/* In-database procedure */
/* Check the log for SQL push-down */
options sastrace=",,,d" sastraceloc=saslog nostsuffix ;
proc tabulate data=ndb2.sales ;
   var sales ;
   class sales_person region ;
   tables sales_person="Sales Person", region="Sales per
Region"*sales=""*sum="" ;
run ;

/* Bulkload */
/* Create sample table */
data prdsale ;
   set sashelp.prdsale ;
   do i=1 to 1000 ;
      output ;
   end ;
run ;
/* Drop target if exists */
proc datasets lib=ndb2 nowarn nolist ;
   delete from_sas_bulkload ;
quit ;
proc append base=ndb2.from_sas_bulkload(bulkload=yes bl_method=cliload)
data=prdsale ;
run ;

/* List Db2 tables */
/* Check FROM_SAS_BULKLOAD presence */
proc datasets lib=ndb2 ;
quit ;


/*** CAS ***/

cas mysession ;

/* Drop caslib if exists */
proc cas ;
   dropcaslib / caslib="casndb2" quiet=true ;
quit ;

/* Native Db2 */

/* sampledsn is the dsn defined earlier in db2dsdriver.cfg */
```

```
caslib casndb2 datasource=(srctype="db2" database="sampledsn"
   username="db2inst1" password="lnxsas" schema="DB2INST1")
   libref=casndb2 ;

/* Other syntaxes */
/*
caslib casndb2 datasource=(srctype="db2"
conOpts="DSN=sampledsn;UID=db2inst1;PWD=lnxsas;"
   schema="DB2INST1")
   libref=casndb2 ;
caslib casndb2 datasource=(srctype="db2"
conOpts="Hostname=db2server;Database=sample;ServiceName=50000;UID=db2inst1
;PWD=lnxsas;"
   schema="DB2INST1")
   libref=casndb2 ;
*/

/* List Db2 tables */
proc casutil incaslib="casndb2" outcaslib="casndb2" ;
   list files ;
quit ;

/* Load Db2 table */
proc casutil incaslib="casndb2" outcaslib="casndb2" ;
   load casdata="EMP" casout="CASEMP" replace ;
   list tables ;
quit ;

/* Write Db2 table */
proc casutil incaslib="casndb2" outcaslib="casndb2" ;
   load data=sashelp.prdsale casout="prdsale" replace ;
   save casdata="prdsale" casout="FROM_CAS" replace ;
   /* Check FROM_CAS presence */
   list files ;
quit ;

/* In-database fedSQL */
/* Check the log for offloaded SQL statement */
proc fedsql sessref=mySession _method ;
   create table casndb2.sales_agg{options replace=true replication=0} as
   select sales.sales_person, sales.region, sum(sales.sales) as sum_sales
   from casndb2."SALES" as sales
   group by sales_person, region ;
quit ;

/* Multi-Node Load Db2 table */
/* CAS MPP only - multiple CAS workers */
/* Use 99 to trigger a warning in the log */
proc casutil incaslib="casndb2" outcaslib="casndb2" ;
   load casdata="EMP_ACT" casout="EMP_ACT" options=(numreadnodes=99)
replace ;
quit ;
proc cas ;
   tableDetails / caslib="casndb2" name="EMP_ACT" level="node" ;
```

```
quit ;

/* Multi-Node Write Db2 table */
/* CAS MPP only - multiple CAS workers */
/* Use 99 to trigger a warning in the log */
proc casutil incaslib="casndb2" outcaslib="casndb2" ;
   load data=sashelp.prdsale casout="prdsale" replace ;
   save casdata="prdsale" casout="FROM_CAS_MULTINODE" options=
(numwritenodes=99) replace;
   /* Check FROM_CAS_MULTINODE presence */
   list files ;
quit ;

cas mysession terminate ;
```

Optional: Enable JDBC Access to DB2

This is an alternative route to the native Db2 access described above. This route does not have all the capabilities of the native Db2 access (in-database push-down, bulkload, performance, etc.). It's just simpler to setup.

Setup SAS Viya for JDBC

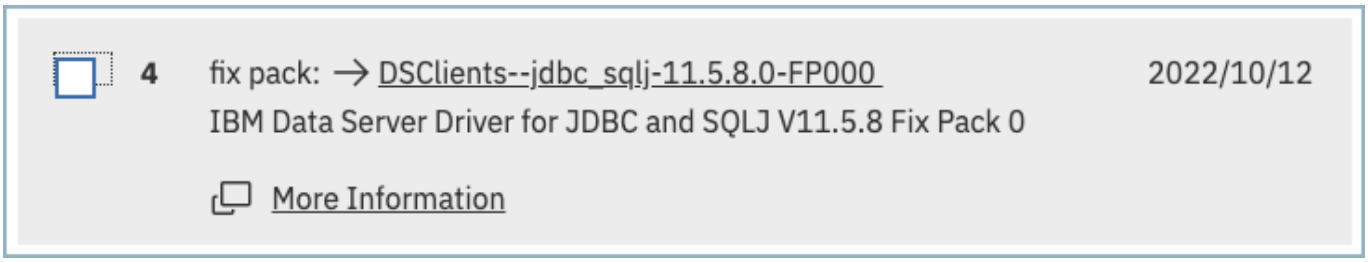
Basically, to make a JDBC driver available in your SAS environment, you only need to add it to a volume mounted as `/data-drivers/jdbc`. Then it will be automatically added to the CLASSPATH.

More information at https://communities.sas.com/t5/SAS-Communities-Library/JDBC-Drivers-Deployment-Made-Easier-SAS-Viya-2021-2-4/ta-p/802050.

Get the Db2 JDBC Driver

You can download it directly from the IBM web site:

https://www.ibm.com/support/fixcentral/swg/selectFixes?parent=ibm%7EInformation%20Management&product=ibm/Information+Management/IBM+Data+Server+Client+Packages&release=11.5.*&platform=Linux+64-bit,x86_64&function=all



Or if you have already installed the Db2 Client, you can find it in `<Db2-client-install-dir>/java/db2jcc4.jar`

Copy the Db2 JDBC Driver

Identify where the `/data-drivers/jdbc` mounted directory is mounted from. Typically, this has been configured through some yaml files.

On your "SAS Viya jump host", as `jumpuser`, run:

```
# cat $deploy/site-config/compute/<custom-name>.yaml
cat ~/project/deploy/gelenv/site-config/compute/compute-server-add-jdbc-
nfs-mount.yaml

# cat $deploy/site-config/cas/<custom-name>.yaml
cat ~/project/deploy/gelenv/site-config/cas/cas-add-jdbc-nfs-mount.yaml
```

You should see something like this in both cases (Compute and CAS):

```
    nfs:
      path: /shared/gelcontent/access-clients/jdbc
      server: my.nfs.server.com
```

Remember we have mounted it on the "SAS Viya jump host" earlier in this section Mount the Location of SAS/ACCESS Clients so it is already available through NFS.

On your "SAS Viya jump host", as `jumpuser`, copy the Db2 JDBC Driver from the Db2 client to the `/data-drivers/jdbc` mounted directory:

```
cp /gelcontent/access-clients/db2client/sqllib/java/db2jcc4.jar
/gelcontent/access-clients/jdbc/.
```

Check if the driver has been correctly copied.

```
ll /gelcontent/access-clients/jdbc/
```

Test the Availability of the Driver in SAS

In SAS Studio, run the following program to check that the Db2 driver is available:

```
options sastrace="d,,," sastraceloc=saslog nostsuffix ;

libname test jdbc url="jdbc:dummy://server:9999/database" ;
```

This fails expectedly. But you should see in the SAS log that `/data-drivers/jdbc/db2jcc4.jar` is available:

```
JDBC: attempting JDBC connection: jdbc:dummy://server:9999/database
/opt/sas/viya/home/SASFoundation/lib/access/hadoop/access-hadoop-
```

```
hivehelper-2.1.30.jar
/opt/sas/viya/home/lib64/accessclients/jdbc/cdata.jdbc.apachehive.jar
/opt/sas/viya/home/lib64/accessclients/jdbc/cdata.jdbc.databricks.jar
/opt/sas/viya/home/lib64/accessclients/jdbc/cdata.jdbc.facebook.jar
/opt/sas/viya/home/lib64/accessclients/jdbc/cdata.jdbc.googleanalytics.jar
/opt/sas/viya/home/lib64/accessclients/jdbc/cdata.jdbc.googledrive.jar
/opt/sas/viya/home/lib64/accessclients/jdbc/cdata.jdbc.odata.jar
/opt/sas/viya/home/lib64/accessclients/jdbc/cdata.jdbc.onedrive.jar
/opt/sas/viya/home/lib64/accessclients/jdbc/cdata.jdbc.sparksql.jar
/opt/sas/viya/home/lib64/accessclients/jdbc/cdata.jdbc.twitter.jar
/opt/sas/viya/home/lib64/accessclients/jdbc/cdata.jdbc.youtubeanalytics.ja
r
/data-drivers/jdbc/db2jcc4.jar
```

Test SAS/ACCESS to JDBC for Db2

In SAS Studio, run the following program, step-by-step, to check SAS/ACCESS to JDBC features:

```
/* Reset trace option */
options sastrace=off ;

/*** SAS Compute Server ***/

/* Access to Db2 through JDBC */

libname jdb2 jdbc url="jdbc:db2://db2server:50000/sample"
   class="com.ibm.db2.jcc.DB2Driver" user="db2inst1" password="lnxsas"
   ;

/* List Db2 tables */
proc datasets lib=jdb2 ;
quit ;

/* Read Db2 table */
data emp ;
   set jdb2.emp ;
run ;

/* Write Db2 table */
/* Drop target if exists */
proc datasets lib=jdb2 nowarn nolist ;
   delete from_sas_jdbc ;
quit ;
data jdb2.from_sas_jdbc ;
   set sashelp.prdsale ;
run ;

/* List Db2 tables */
/* Check FROM_SAS_JDBC presence */
proc datasets lib=jdb2 ;
quit ;
```

```
/* In-database procedure */
/* Check the log for SQL push-down */
/* !!! No pushdown with JDBC !!! */
options sastrace=",,,d" sastraceloc=saslog nostsuffix ;
proc tabulate data=jdb2.sales ;
   var sales ;
   class sales_person region ;
   tables sales_person="Sales Person", region="Sales per
Region"*sales=""*sum="" ;
run ;


/*** CAS ***/

cas mysession ;

/* Drop caslib if exists */
proc cas ;
   dropcaslib / caslib="casjdb2" quiet=true ;
quit ;

/* Access to Db2 through JDBC */

caslib casjdb2 datasource=(srctype="jdbc"
url="jdbc:db2://db2server:50000/sample"
   user="db2inst1" password="lnxsas" schema="DB2INST1")
   libref=casjdb2 ;

/* List Db2 tables */
proc casutil incaslib="casjdb2" outcaslib="casjdb2" ;
   list files ;
quit ;

/* Load Db2 table */
proc casutil incaslib="casjdb2" outcaslib="casjdb2" ;
   load casdata="EMP" casout="CASEMP" replace ;
   list tables ;
quit ;

/* Write Db2 table */
proc casutil incaslib="casjdb2" outcaslib="casjdb2" ;
   load data=sashelp.prdsale casout="prdsale" replace ;
   save casdata="prdsale" casout="FROM_CAS_JDBC" replace ;
   /* Check FROM_CAS_JDBC presence */
   list files ;
quit ;

/* In-database fedSQL */
/* Check the log for offloaded SQL statement */
proc fedsql sessref=mySession _method ;
   create table casjdb2.sales_agg{options replace=true replication=0} as
   select sales.sales_person, sales.region, sum(sales.sales) as sum_sales
   from casjdb2."SALES" as sales
```

```
   group by sales_person, region ;
quit ;

/* Multi-Node Load Db2 table */
/* CAS MPP only - multiple CAS workers */
/* Use 99 to trigger a warning in the log */
proc casutil incaslib="casjdb2" outcaslib="casjdb2" ;
   load casdata="EMP_ACT" casout="EMP_ACT" options=(numreadnodes=99)
replace ;
quit ;
proc cas ;
   tableDetails / caslib="casjdb2" name="EMP_ACT" level="node" ;
quit ;

/* Multi-Node Write Db2 table */
/* CAS MPP only - multiple CAS workers */
/* Use 99 to trigger a warning in the log */
proc casutil incaslib="casjdb2" outcaslib="casjdb2" ;
   load data=sashelp.prdsale casout="prdsale" replace ;
   save casdata="prdsale" casout="FROM_CAS_JDBC_MULTINODE" options=
(numwritenodes=99) replace ;
   /* Check FROM_CAS_JDBC_MULTINODE presence */
   list files ;
quit ;

cas mysession terminate ;
```