MAXIMO 7612 TO MANAGE 8.6

Contents

1. Introduction

This is one of my first few upgrade PoC of a local Maximo76 setup to MAS Manage8.6 setup in IBM TechZone Cloud. I am sure there are more than one approach to upgrade, but this is just to prove ourselves at least one successful approach.

This is just a guide with real commands. Please follow the standards and procedures you must for a client setup, apart from best practices and standard procedures of IBM Maximo.

This document needs good understanding of Maximo EAM version 7.6 and its administration and fair level of understanding of MAS, OpenShift concepts and OpenShift CLI.

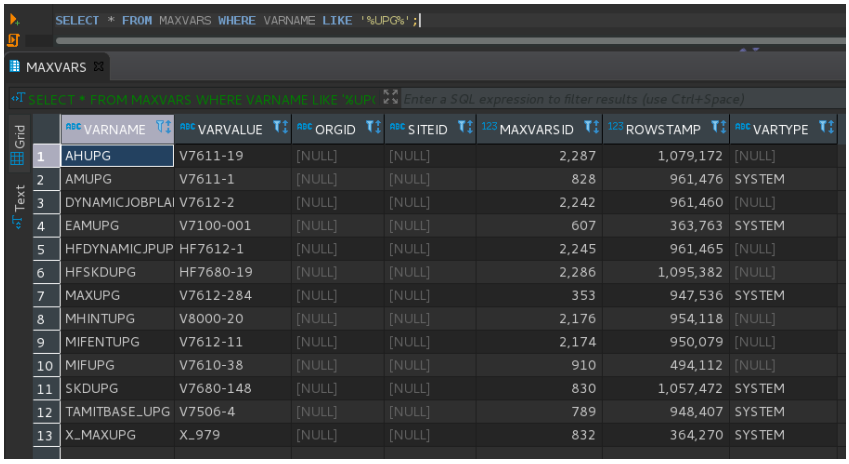
2. Working Environment

Windows laptop with two VirtualBox VMs.

VM1 - Maximo 76 Environment:

- Maximo 7612 setup
- DB2 database
- Customisation: No but outlined the steps in this document for customisations.

The screenshots were taken from Maximo76 OOTB (no customisations).



```
SELECT * FROM MAXVARS WHERE VARNAME LIKE '%UPG%';
```

MAXVARS

SELECT * FROM MAXVARS WHERE VARNAME LIKE '%UPG%' — Enter a SQL expression to filter results (use Ctrl+Space)

	VARNAME	VARVALUE	ORGID	SITEID	MAXVARSID	ROWSTAMP	VARTYPE
1	AHUPG	V7611-19	[NULL]	[NULL]	2,287	1,079,172	[NULL]
2	AMUPG	V7611-1	[NULL]	[NULL]	828	961,476	SYSTEM
3	DYNAMICJOBPLAI	V7612-2	[NULL]	[NULL]	2,242	961,460	[NULL]
4	EAMUPG	V7100-001	[NULL]	[NULL]	607	363,763	SYSTEM
5	HFDYNAMICJPUP	HF7612-1	[NULL]	[NULL]	2,245	961,465	[NULL]
6	HFSKDUPG	HF7680-19	[NULL]	[NULL]	2,286	1,095,382	[NULL]
7	MAXUPG	V7612-284	[NULL]	[NULL]	353	947,536	SYSTEM
8	MHINTUPG	V8000-20	[NULL]	[NULL]	2,176	954,118	[NULL]
9	MIFENTUPG	V7612-11	[NULL]	[NULL]	2,174	950,079	[NULL]
10	MIFUPG	V7610-38	[NULL]	[NULL]	910	494,112	[NULL]
11	SKDUPG	V7680-148	[NULL]	[NULL]	830	1,057,472	SYSTEM
12	TAMITBASE_UPG	V7506-4	[NULL]	[NULL]	789	948,407	SYSTEM
13	X_MAXUPG	X_979	[NULL]	[NULL]	832	364,270	SYSTEM

Note: Even though the Maximo76 environment referred here has other add-ons installed, we are focusing on base Maximo (MAXUPG 7612-284) upgrade.

VM2 - Ubuntu VM with OpenShift CLI installed. You can use your local laptop itself or WSL in your Windows laptop.

A common shared folder exists for these 2 VMs on the Windows Host and the shared folder path in both the VMs is same: `/media/sf_VMSharedFolder`

Cloud/TechZone - MAS Manage 8.6 Environment:

- MAS 8.10
- Manage 8.6
- DB2 database installed along with Manage 8.6 (or MAS IoT)
- Cloud/TechZone setup

3. Approach to Upgrade

There could be different approaches, depending on Maximo76 setup, size, environment constrains, client requirements, etc.

This document considers the following approach for upgrade:

- For Maximo76 steps, we use a production copy which is used only for the purpose of upgrade (not for dev, testing or production).
- For MAS Manage, we install the MAS Core and Manage as OOTB first.
- Then, we use the default installed database of MAS Manage and import the Maximo76 schema into it.
- Perform Upgrade in this Maximo76 schema of MAS environment.
- Let the OOTB Upgrade finish in MAS environment and then apply the customisations.

Other approaches:

1. Mx76 as secondary DB in MAS
   - instead of overwriting default DB2 database with Maximo76 schema, install a new (secondary) database for Maximo76.
   - Upgrade this Maximo76 database
   - Then either point Manage to this upgraded database or backup/restore this upgraded database into default Manage DB.
   - I used this approach earlier (not documented here) and it works as well.
2. DB First approach
   - In this approach, we install MAS Core
   - Install DB2 database in MAS with same names used by Manage install.
   - Import the Maximo76 schema into this DB2 database.
   - Install MAS Manage: This installation of Manage should take care of upgrading the Db2 database with Maximo76 schema.
   - Not tried this approach yet but prefer not doing this approach as we want the Manage installation to be successful with OOTB.
3. Upgrade locally
   - In a Maximo76 environment (locally), try to get the MAS Manage SMP and related artefacts as a SMP_M8 folder.
   - Point the SMP_M8 folder's DB Connection to Maximo76 database.
   - Perform the `updatedb` from the SMP_M8 folder.
   - Export this upgraded copy and import in MAS environment.
   - I've given a try to this approach but was not successful as SMP_M8 copy needs additional artefacts (like secrets) which are in OpenShift and need to figure out how to get them working in SMP_M8 folder.

4. Customisations

If your Maximo76 setup has customisations, bundle them into an archive as documented in Customizing IBM Maximo Manage - IBM Documentation, compatible with MAS Manage.

This includes and not limited to, custom product config, DBCs, MXSs, Java Classes (businessobjects, maximouiweb, tools, etc), Automation Scripts (compatible with GBS auto deploy tool), Custom XSLs, etc.

You may refer to my MAS Manage Customisation PoC here for a sample.

Note: Pease ensure the Java version match/computability when compiling custom java classes for Manage customisation archive.

5. Preparing Maximo76 DB

Perform these steps to avoid errors when importing or upgrading the Maximo76 database in MAS environment.

Note: Steps c, d, e, f can be ignored if performing this for OOTB Maximo76 database.

a. Backup DB

- Take a backup of the database before we prepare it for upgrade. Just in case, if you need to start over again

b. Integrity Check

- Run the integrity checker in Maximo76 database as is ie., with customisations, etc.
- Fix the integrity errors or run integrity checker in repair mode.
- Run the integrity checker again until all errors are fixed.

c. Stripping Off Customisations

Remove all the custom classes and applicable mbo and mbo-field registrations from Maximo76 database and bundle the customisation files in the customisation archive as mentioned above.

In the Maximo76 database perform the following:

- Update the custom classes used for persistent and non-persistent objects to the default OOTB custom classes.
- Remove/Replace the custom classes for MBO Attributes.
- Do the same for Integration custom classes
- You can leave the custom class name references in the presentation XMLs as is.
- You can keep the automation scripts as is.
- Basically, remove all custom class reference from DB tables (except from the XMLs of MAXPRESENTATION).

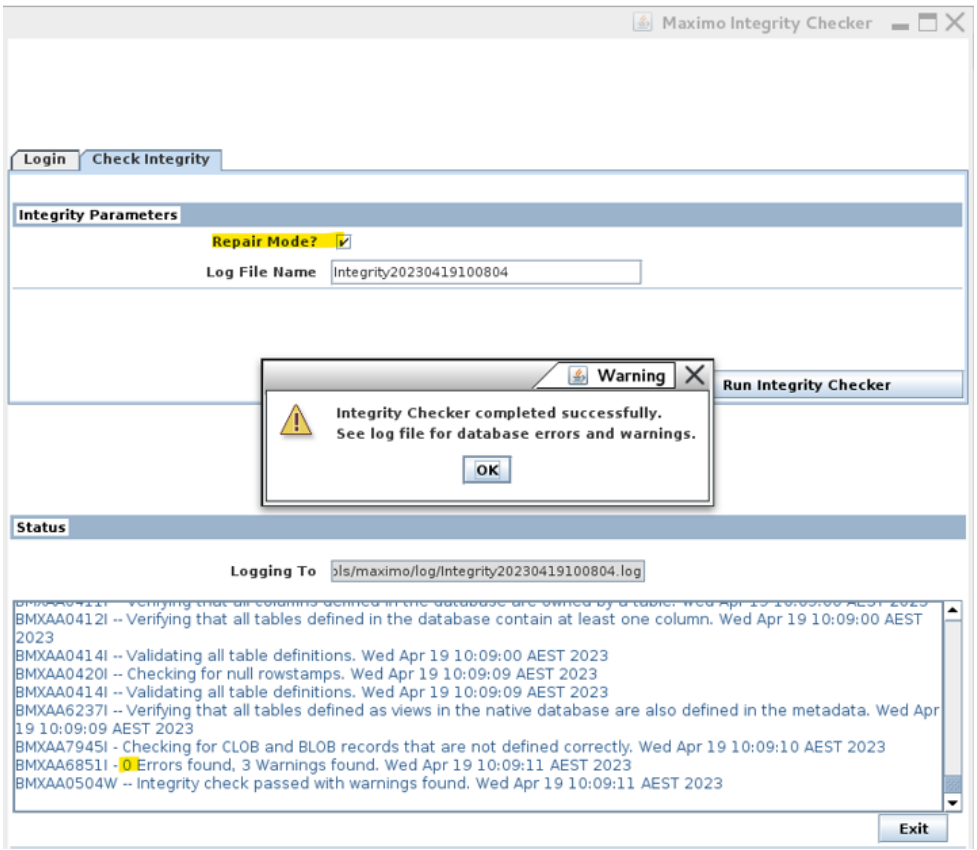
d. Perform updatedb

Run the updatedb command in Maximo 76 itself without the custom product config, so that the updatedb resets the OOTB mbo and field class definitions.

At this stage maximo76 DB has custom objects and attributes but no custom class reference (except in MAXPRESENTATION XMLs).

e. Integrity Check Again

- Run the integrity checker in Maximo76 database, this time it is without customisations.
- It should pass in the first go, else something is being missed in the above steps.



f. Fix Script

Prepare a script for the fixes which are not covered in customisation archive, eg Integration End Point URLs, Doclinks updates, etc. So that this can be run after Maximo76 is upgraded to Manage8.

6. Export Maximo76 DB

There are multiple ways of exporting DB, I used this approach to simply export the DDL and Data. This is to ensure minimal settings are affected when importing this into MAS Manage database.

These values are here for reference, they could be different in your environment:

DB install user	`ctginst1`
The name of Maximo76 database in DB2	`maxdb76`
Maximo Schema	`maximo`
File name to capture all Maximo DDL	`maxdb76.ddl`
The Maximo76 VM's shared folder path	`/media/sf_VMSharedFolder`

a. Backup vs Export

Preferred the Export-Import option over Backup-Restore. For the Maximo7612 OOTB database I was working with, the DB2 backup size (~6 GB) was way more than the Export size (tar file of ~400 MB). Plus going with the Export options with minor modifications lets you import the Schema and its objects

rather than the entire DB. The MAS DB default install had other schemas as well, which I did not want to disturb.

b. Exporting DDL and Data as tar file

Run the following commands in the terminal of Maximo76 installation (Local VM, in my case). Ensure the DB is not connected by any application, like Maximo or SQL Developer or any other app.

```
su ctginst1
mkdir /tmp/db2bkp/
cd /tmp/db2bkp
db2 terminate
db2 deactivate db maxdb76
db2look -d maxdb76 -e -z maximo -l -x -f -o maxdb76.ddl
db2move maxdb76 export -sn maximo -l ./lobs
tar -cf maxdb76exp.tar *
cd /tmp
chmod -R 777 db2bkp
exit
cp /tmp/db2bkp/maxdb76exp.tar /media/sf_VMSharedFolder/Mx76/export
```

If the commands like `db2`, `db2look` and `db2move` are not recognised, trying using their full path, eg:

```
/opt/IBM/db2/V11.1/bin/db2 deactivate db maxdb76
```

Example output:

```
[ctginst1@mx761 apr20]$ db2 terminate
DB20000I  The TERMINATE command completed successfully.

[ctginst1@mx761 apr20]$ /opt/IBM/db2/V11.1/bin/db2 terminate
DB20000I  The TERMINATE command completed successfully.

[ctginst1@mx761 apr20]$ /opt/IBM/db2/V11.1/bin/db2 deactivate db maxdb76
DB20000I  The DEACTIVATE DATABASE command completed successfully.

[ctginst1@mx761 apr20]$ /opt/IBM/db2/V11.1/bin/db2look -d maxdb76  -e -z maximo -l  -x -f  -o  maxdb76.ddl
-- No userid was specified, db2look tries to use Environment variable USER
-- USER is: CTGINST1
-- Specified SCHEMA is: MAXIMO
-- Creating DDL for table(s)

-- Schema name is ignored for the Federated Section
-- Output is sent to file: maxdb76.ddl
-- Binding package automatically ...
-- Bind is successful

[ctginst1@mx761 apr20]$ ls
maxdb76.ddl
```

```
[ctginst1@mx761 apr20]$ /opt/IBM/db2/V11.1/bin/db2move maxdb76 export -sn maximo -l ./lobs
    *****  DB2MOVE  *****
    Action:  EXPORT
    Start time:  Thu Apr 20 14:38:57 2023

    All schema names matching:  MAXIMO;
    Connecting to database MAXDB76 ... successful!  Server : DB2 Common Server V11.1.0
    Binding package automatically ... /home/ctginst1/sqllib/bnd/db2common.bnd ... successful!
    Binding package automatically ... /home/ctginst1/sqllib/bnd/db2move.bnd ... successful!

    EXPORT:     53 rows from table "MAXIMO  "."ACCOUNTDEFAULTS"
    EXPORT:      0 rows from table "MAXIMO  "."ACTCI"
    EXPORT:      0 rows from table "MAXIMO  "."ACTCIRELATION"
    ...
    ...
    ...
    EXPORT:    312 rows from table "MAXIMO  "."WPITEM"
    EXPORT:    362 rows from table "MAXIMO  "."WPLABOR"

    Disconnecting from database ... successful!

    End time:  Thu Apr 20 14:39:42 2023
```

All the above commands should be successful, unlike the following example:

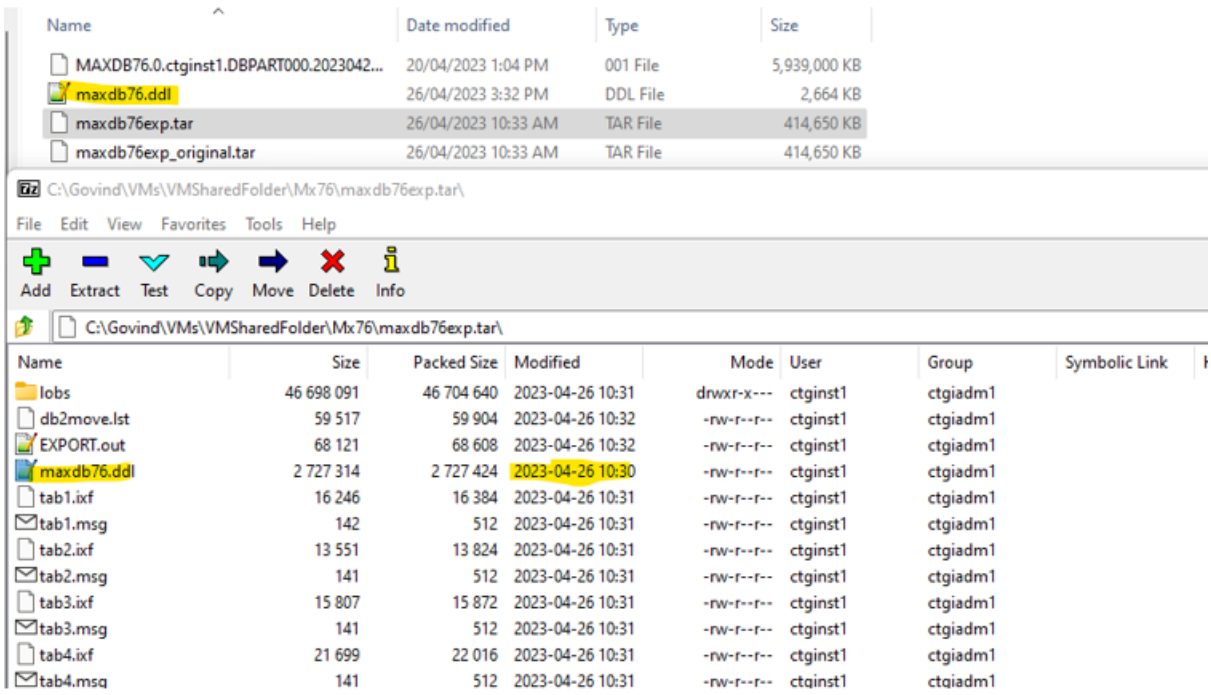
```
[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$ db2 terminate
DB20000I  The TERMINATE command completed successfully.
[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$
[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$ db2 deactivate db BLUDB
SQL1495W  Deactivate database is successful, however, there is still a
connection to the database.
[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$
```

c. Updating the tar export file

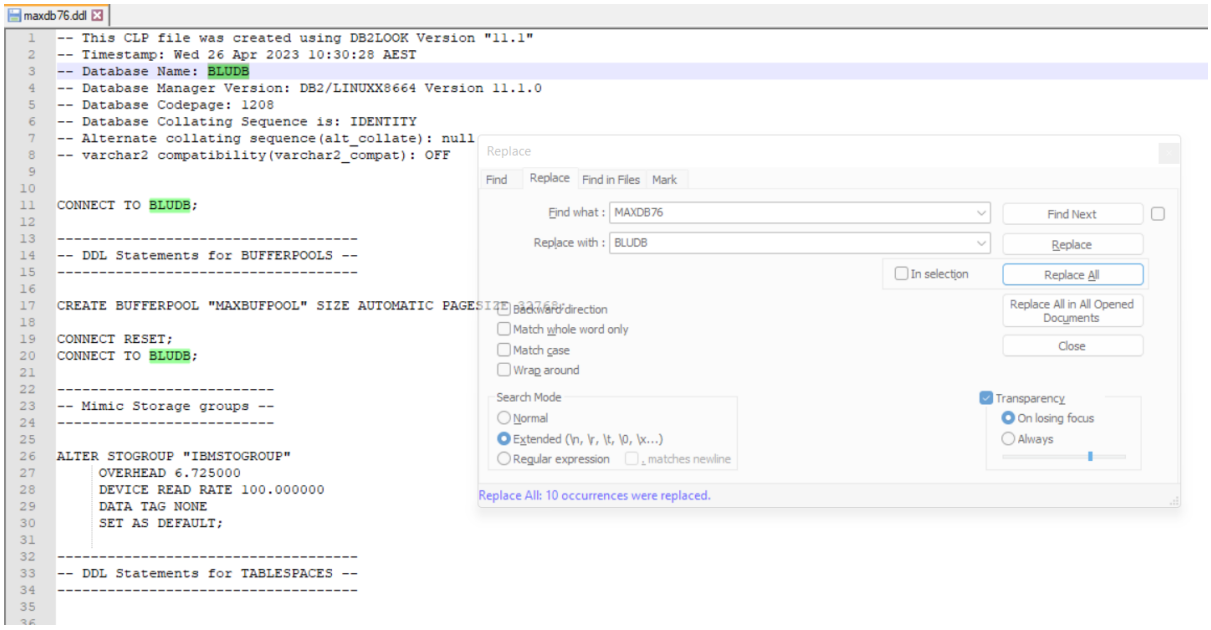
The tar file from the export is now available in the Windows Host of the Maximo76 VM.

Certain sections of the maxdb76.ddl file needs to be modified.

- Extract the maxdb76.ddl file from the tar (I am using 7zip here), after extraction:



- Find Replace the DB Name from MAXDB76 to BLUDB. The BLUDB is the default DB Name used by MAS Manage.



- Remove the following sections from the file:
  - Mimic Storage groups
  - DDL Statements for Tablespaces
  - Mimic Tablespaces

The purpose behind doing this is to keep the IBM DB2 settings close to default installed settings in the MAS environment, we just want to replace the Maximo Schema and its objects.

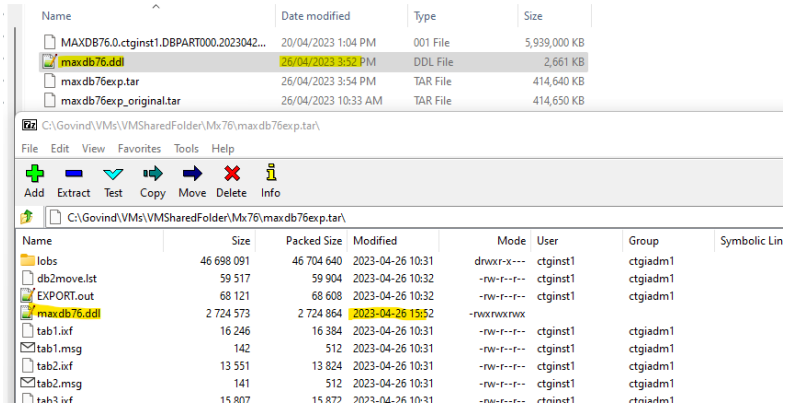
```
----------------------------
-- Mimic Storage groups --
----------------------------

ALTER STOGROUP "IBMSTOGROUP"
     OVERHEAD 6.725000
     DEVICE READ RATE 100.000000
     DATA TAG NONE
     SET AS DEFAULT;
```

```
-----------------------------------
-- DDL Statements for TABLESPACES --
-----------------------------------


CREATE REGULAR TABLESPACE "MAXDATA" IN DATABASE PARTITION GROUP IBMDEFAULTGROUP
     PAGESIZE 32768 MANAGED BY AUTOMATIC STORAGE
     USING STOGROUP "IBMSTOGROUP"
     AUTORESIZE YES
     INITIALSIZE 5000 M
     MAXSIZE NONE
     EXTENTSIZE 32
     PREFETCHSIZE AUTOMATIC
     BUFFERPOOL "MAXBUFPOOL"
     DATA TAG INHERIT
     OVERHEAD INHERIT
     TRANSFERRATE INHERIT
     DROPPED TABLE RECOVERY ON;


CREATE TEMPORARY TABLESPACE "MAXTEMP" IN DATABASE PARTITION GROUP IBMTEMPGROUP
     PAGESIZE 32768 MANAGED BY AUTOMATIC STORAGE
     USING STOGROUP "IBMSTOGROUP"
     EXTENTSIZE 32
```

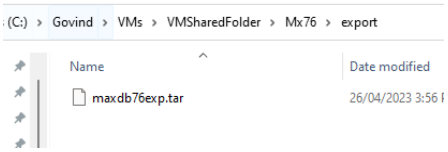
```
106  ----------------------
107  -- Mimic tablespace --
108  ----------------------
109
110  ALTER TABLESPACE "SYSCATSPACE"
111        PREFETCHSIZE AUTOMATIC
112        OVERHEAD INHERIT
113        AUTORESIZE YES
114        TRANSFERRATE INHERIT;
115
116
117  ALTER TABLESPACE "SYSCATSPACE"
118        USING STOGROUP "IBMSTOGROUP";
119
120
121  ALTER TABLESPACE "TEMPSPACE1"
122        PREFETCHSIZE AUTOMATIC
123        OVERHEAD INHERIT
124        FILE SYSTEM CACHING
125        TRANSFERRATE INHERIT;
126
127
128  ALTER TABLESPACE "USERSPACE1"
129        PREFETCHSIZE AUTOMATIC
130        OVERHEAD INHERIT
131        AUTORESIZE YES
132        TRANSFERRATE INHERIT
133        DATA TAG INHERIT;
134
135
136  ALTER TABLESPACE "USERSPACE1"
137        USING STOGROUP "IBMSTOGROUP";
138
139
140  ------------------------------------------------
141  -- DDL Statements for Schemas
142  ------------------------------------------------
143
```

Until end of “ALTER TABLESPACE …” commands.

- Save the changes to maxdb76.ddl file
- Add it back to the tar:



- Move the tar file to another directory. This new directory to contain only the exported tar file:



7. Installing MAS Core and Manage

Use the standard procedures to install MAS Core and Manage with DB2. I have used the IBM ansible playbook to install Manage.

After the install we will have just Manage application running in MAS. MAS itself has various services running in OpenShift to support Manage (and other applications like Health, Monitor, etc). No add-ons or industry solutions were chosen when installing Manage.

Following values are recorded here after the install and most of them will be used in the steps during upgrade.

MAS Instance name	`gjmasinst1`
MAS Workspace name	`masdev`

MAS DB2 Namespace	`db2u`
MAS DB2 Pod name	`c-db2w-shared-db2u-0`
MAS DB2 Container name	`db2u`
MAS DB2 Database name	`BLUDB`
MAS DB2 Manage Schema name	`MAXIMO`
MAS DB2 Username	`db2inst1`
MAS DB2 Password	`ssAcdQsVKWt8qe2`

MAS Manage Namespace	`mas-gjmasinst1-manage`
MAS Manage MAXINST Pod name	`gjmasinst1-masdev-manage-maxinst-<random_chars>-<random_chars>`
MAS Manage MAXINST Container name	`manage-maxinst-maxinst`
MAS Manage Application Pod name	`gjmasinst1-masdev-all-<random_chars>-<random_chars>`
MAS Manage Application Container name	`all`

Note: For those new to MAS environment:

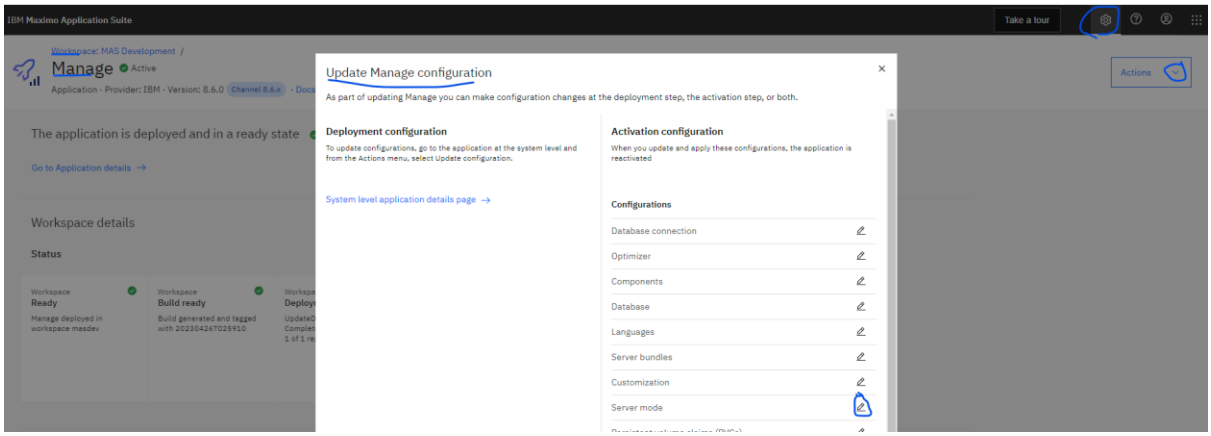
- All components of MAS run as Containers.
- One or more containers are encapsulated in a Pod
- Pod is the least deployable unit in OpenShift/Kubernetes.
- The “maxinst” pod contains the Maximo SMP folder structure
- The “c-db2w-shared-db2u-0” pod runs the DB2 engine
- The “gjmasinst1-masdev-all-b9f64c7db-gsg7f” pod runs the actual Manage application
- “all” is the name of server bundle synonymous to a Maximo 76’s “cluster of MX Servers”.
- Here “all” refers to one bundle for the purpose of UI, MIF, Cron and Reports.
- Please go through the IBM Maximo Application Suite documentation for more details.

8. Importing Maximo76 data into Manage DB

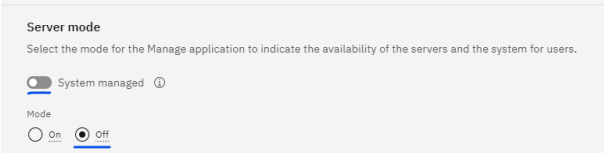
At this point, we have installed OOTB MAS Manage8.6 which has also installed the DB2 database called BLUDB. Did not choose to install demo data in MAS Manage. The Maximo76 data export however has the demo data.

a. Turn Server Mode Off

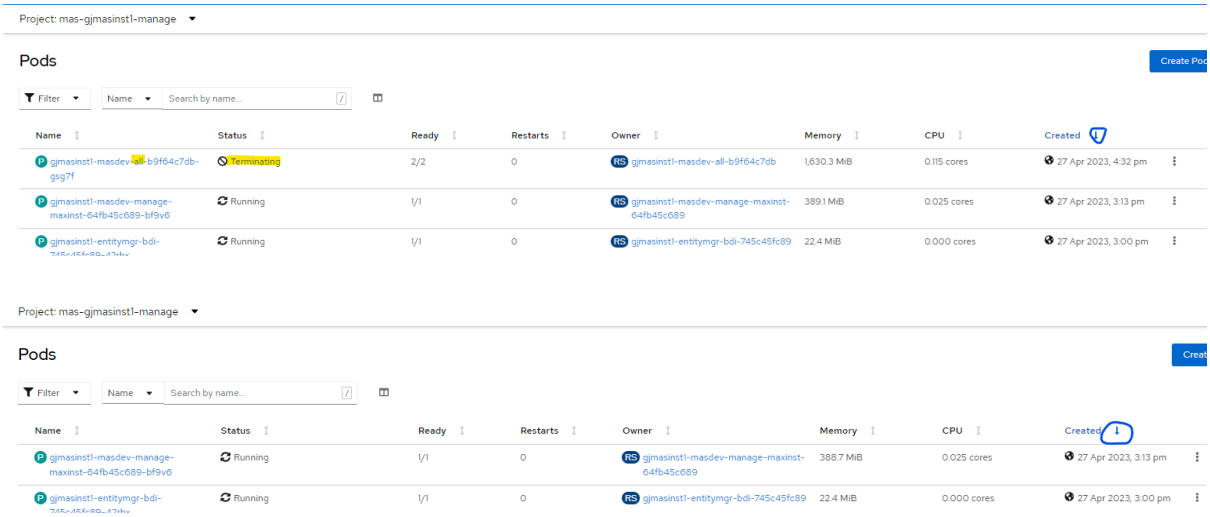
- Login to the MAS Console
- Navigate to “Update Configuration” page of Manage (MAS Console > Suite Administration > Workspace > Manage tile > Action button drop down > Update Configuration > “Server mode” pencil icon)



- Turn off the “System Managed” toggle button and Click on the “Off” option



- Scroll up the page, click on the “Apply Changes” blue button, then Confirm.
- Wait for some time, let the changes be applied.
- To verify, login to OpenShift console and verify the “all” pod under of Manage application is terminating and disappear from the list.



b. Backup MAS Manage DB

Back the database as a safe return point. I used the following commands in `c-db2w-shared-db2u-0` pod's terminal. At the end create a directory called "mx76exp" to hold the Maximo76's exported files.

```
su db2inst1
db2 terminate
db2 disconnect all
db2 deactivate db BLUDB
mkdir /mnt/backup/m8_ootb_asis
db2 backup database BLUDB to /mnt/backup/m8_ootb_asis
exit
```

Create a directory to place the Maximo76 export files:

```
mkdir /mnt/backup/mx76exp
chmod -R 777 /mnt/backup/mx76exp
```

Example:

```
sh-4.4$ su db2inst1
[db2inst1@c-db2w-shared-db2u-0 - Db2U ]$ cd

[db2inst1@c-db2w-shared-db2u-0 - Db2U db2inst1]$ pwd
/mnt/blumeta0/home/db2inst1

[db2inst1@c-db2w-shared-db2u-0 - Db2U db2inst1]$ db2 terminate
DB20000I  The TERMINATE command completed successfully.

[db2inst1@c-db2w-shared-db2u-0 - Db2U db2inst1]$ db2 deactivate db BLUDB
SQL1495W  Deactivate database is successful, however, there is still a
connection to the database.

[db2inst1@c-db2w-shared-db2u-0 - Db2U db2inst1]$ db2 disconnect all
DB20000I  The SQL DISCONNECT command completed successfully.

[db2inst1@c-db2w-shared-db2u-0 - Db2U db2inst1]$ db2 deactivate db BLUDB
DB20000I  The DEACTIVATE DATABASE command completed successfully.

[db2inst1@c-db2w-shared-db2u-0 - Db2U db2inst1]$ mkdir /mnt/backup/m8_ootb_asis

[db2inst1@c-db2w-shared-db2u-0 - Db2U db2inst1]$ db2 backup database BLUDB to /mnt/backup/m8_ootb_asis

Backup successful. The timestamp for this backup image is : 20230426051940

[db2inst1@c-db2w-shared-db2u-0 - Db2U db2inst1]$ ls -ltra /mnt/backup/m8_ootb_asis
total 1815972
drwxrwxrwx. 3 db2inst1 db2iadm1          1 Apr 26 05:19 ..
drwxr-xr-x. 2 db2inst1 db2iadm1          1 Apr 26 05:19 .
-rw-------. 1 db2inst1 db2iadm1 1859555328 Apr 26 05:20 BLUDB.0.db2inst1.DBPART000.20230426051940.001
```

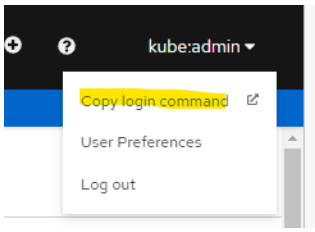
I had to issue the `deactivate` command again after issuing `db2 disconnect all`.

```
[db2inst1@c-db2w-shared-db2u-0 - Db2U db2inst1]$ exit

sh-4.4$ mkdir /mnt/backup/mx76exp

sh-4.4$ ls -ltra /mnt/backup
total 0
drwxr-xr-x. 1 root     root     91 Apr 26 02:42 ..
drwxr-xr-x. 2 db2inst1 db2iadm1  1 Apr 26 05:19 m8_ootb_asis
drwxr-xr-x. 2 db2uadm  wheel     0 Apr 26 06:08 mx76exp
drwxrwxrwx. 4 db2inst1 db2iadm1  2 Apr 26 06:08 .
```

c. Transfer the export file

Use the OpenShift CLI in VM2 to transfer the exported tar file from local setup to MAS DB2 pod's file system.

- Get the command to login from the OpenShift Console, follow the links and copy the command text:



- Login to OpenShift Console using the copied command:

```
oc login --token=sha256~qrhq6WRjmJW0TfhLhEC4geyoQGYYpYTupr5SFnGLfT0 --
server=https://api.ocp-550008e5ux-af52.cloud.techzone.ibm.com:6443
```

- Copy the folder containing the export file to DB2 Pod's backup directory:

```
oc rsync /media/sf_VMSharedFolder/Mx76/export c-db2w-shared-db2u-0:/mnt/backup/mx76exp -
c db2u -n db2u
```

```
ubuntu@Ubuntu2204OpenShift:~$ oc rsync /media/sf_VMSharedFolder/Mx76/export c-db2w-shared-db2u-0:/mnt/backup/mx76exp -c db2u -n db2u
sending incremental file list
export/
export/maxdb76exp.tar

sent 424,694,636 bytes  received 39 bytes  2,206,206.10 bytes/sec
total size is 424,590,848  speedup is 1.00
ubuntu@Ubuntu2204OpenShift:~$
```

- Verify the file transfer in DB2 Pod's terminal:

```
[db2uadm@c-db2w-shared-db2u-0 /]$ su db2inst1

[db2inst1@c-db2w-shared-db2u-0 - Db2U ]$ cd

[db2inst1@c-db2w-shared-db2u-0 - Db2U db2inst1]$ cd /mnt/backup/mx76exp/export/

[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$ ls -ltra
total 414640
-rwxrwxrwx. 1 db2uadm wheel 424590848 Apr 26 05:56 maxdb76exp.tar
drwxrwxrwx. 3 db2uadm wheel         1 Apr 26 06:09 ..
drwxrwxrwx. 2 db2uadm wheel         1 Apr 26 06:12 .
```

- In the DB2 Pod's terminal, extract the contents of the tar file:

```
su db2inst1
cd /mnt/backup/mx76exp/export/
tar -xf maxdb76exp.tar
```

d. Import Maximo76 DDL into BLUDB

While the DB2 pod's terminal is still active -or- use the OpenShift CLI to login to DB2 pod and perform the following.

- Ensure you have switched user to db2inst1, if not already. And change to the "export" folder.

```
su db2inst1
cd /mnt/backup/mx76exp/export/
```

- Drop MAXIMO Schema:

```
db2 connect to BLUDB user db2inst1 using ssAcdQsVKWt8qe2
db2 "call SYSPROC.ADMIN_DROP_SCHEMA ('MAXIMO', NULL, 'ERRORSCHEMA', 'ERRORTABLE')"
```

Example:

```
[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$ db2 connect to BLUDB user db2inst1 using ssAcdQsVKWt8qe2

   Database Connection Information

 Database server        = DB2/LINUXX8664 11.5.8.0
 SQL authorization ID   = DB2INST1
 Local database alias   = BLUDB

[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$


[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$ db2 "call SYSPROC.ADMIN_DROP_SCHEMA ('MAXIMO', NULL, 'ERRORSCHEMA', 'ERRORTABLE')"

  Value of output parameters
  --------------------------
  Parameter Name  : ERRORTABSCHEMA
  Parameter Value : -

  Parameter Name  : ERRORTAB
  Parameter Value : -

  Return Status = 0
[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$
```

The “Return Status” should be 0 for a successful drop schema command.

- Drop ERRORSCHEMA

The SYSPROC.ADMIN_DROP_SCHEMA command above creates an ERROSCHEMA in BLUDB Database. If the Return Status is 0 for this command, then you can drop the ERRORSCHEMA. It would have no objects under it for Return Status 0.

```
db2 drop schema ERRORSCHEMA restrict
```

Example:

```
[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$ db2 drop schema ERRORSCHEMA restrict
DB20000I  The SQL command completed successfully.
```

- Load DDL:

Ensure you have changed to the export directory.

```
cd /mnt/backup/mx76exp/export/
db2 connect reset
db2 disconnect all
db2 connect to BLUDB user db2inst1 using ssAcdQsVKWt8qe2
db2 -tf maxdb76.ddl > ddloutput.log
db2 connect reset
```

Here the output of DDL import is captured in “ddloutput.log”file.

Meanwhile, you can monitor the logs in DB2 pod terminal in OpenShift console:

```
for i in {1..1000}; do echo `cat /mnt/backup/mx76exp/export/ddloutput.log | wc -l` ;
sleep 1; done
```

`Ctrl+C` the above command once you see the last line of the output is not changing i.e., it has no more new lines in the ddloutput.log file.

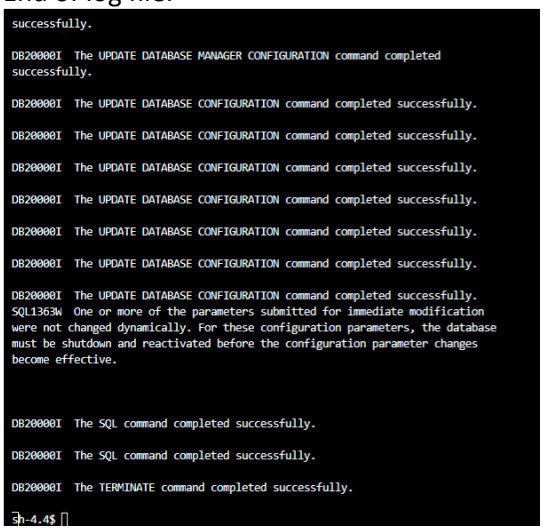
Example:

```
[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$ db2 connect reset
DB20000I  The SQL command completed successfully.

[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$ db2 disconnect all
DB20000I  The SQL DISCONNECT command completed successfully.

[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$ db2 connect to BLUDB user db2inst1 using ssAcdQsVKWt8qe2

   Database Connection Information

 Database server        = DB2/LINUXX8664 11.5.8.0
 SQL authorization ID   = DB2INST1
 Local database alias   = BLUDB

[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$ ls *.ddl
maxdb76.ddl

[db2inst1@c-db2w-shared-db2u-0 - Db2U export]$ db2 -tf maxdb76.ddl > ddloutput.log
```

ddloutput.log contents:

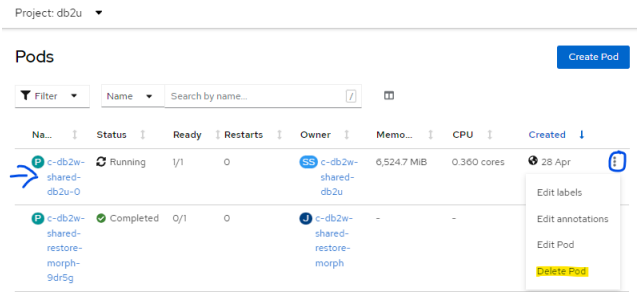
Start of log file:

End of log file:

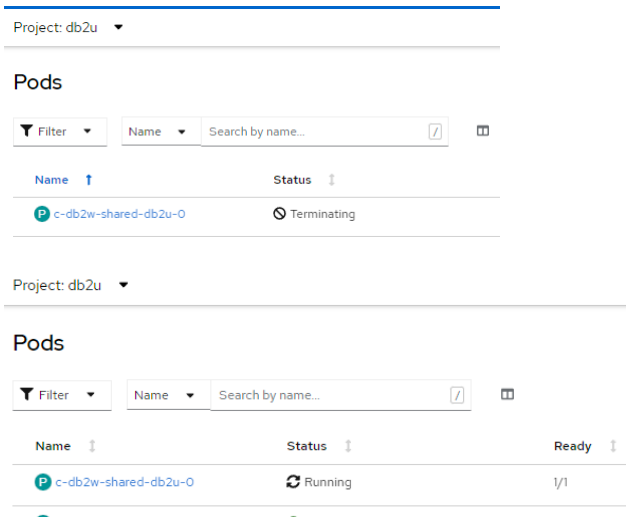


e. Restart DB2 Pod

Delete the DB2 Pod from the OpenShift Console, using the 3 dots button and Delete Pod option:



And wait for the DB2 pod to terminate in OpenShift Console and start again in “Running” status and “1/1” Ready state.



f. Import Maximo76 Data into BLUDB

- From the OpenShift CLI, login to DB2 Pod's terminal

```
oc exec --stdin --tty c-db2w-shared-db2u-0 -c db2u -n db2u -- /bin/bash
```

We use this approach because the data load might take long time and we don't want the terminal to timeout and disconnect, which happens quite often in Terminal of OpenShift Console.

- Run the following commands:

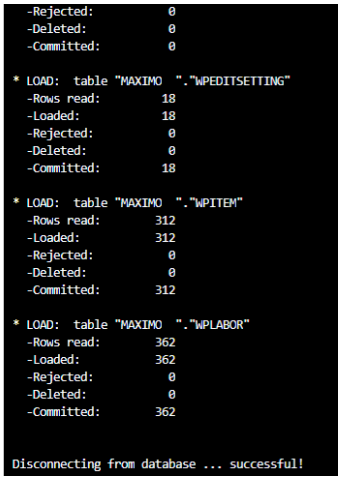
```
su db2inst1
cd /mnt/backup/mx76exp/export/
db2 terminate
db2 disconnect all
db2 deactivate db BLUDB
db2 connect to BLUDB user db2inst1 using ssAcdQsVKWt8qe2
db2 set current schema=maximo
db2move BLUDB load -l /mnt/backup/mx76exp/export/lobs/ > dmloutput.log
db2 connect reset
exit
exit
```

Note: the `db2move` command takes long time.

Meanwhile, you can monitor the logs in DB2 pod terminal in OpenShift console:

```
tail -f /mnt/backup/mx76exp/export/dmloutput.log
```

Example dmloutput.log file (end of file section):



g. Restart DB2 Pod

Restart the DB2 pod as mentioned in the step one above the previous step.

9. Upgrading Maximo76

a. Reset Crypto Keys

Using the OC CLI, login to MAXINST Pod's terminal:

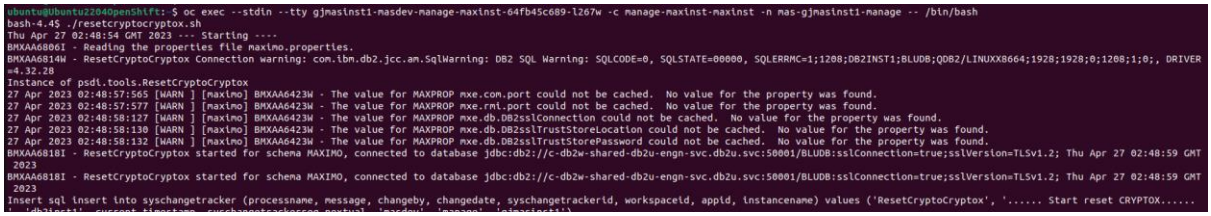
```
oc exec --stdin --tty gjmasinst1-masdev-manage-maxinst-64fb45c689-l267w -c manage-maxinst-maxinst -n mas-gjmasinst1-manage -- /bin/bash
```

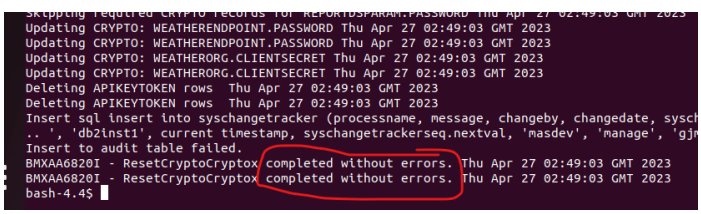
Run the following command:

```
./resetcryptocryptox.sh
```

```
exit
```

Example:





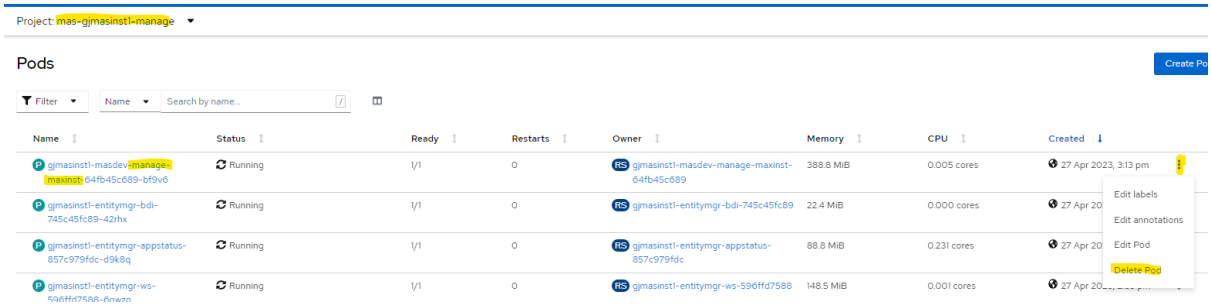
Not performing this step would most likely result in the following `javax.crypto.BadPaddingException` in next step (updatedb):

```
javax.crypto.BadPaddingException: Given final block not properly padded. Such issues can arise if a bad key is used during decryption.
        at java.base/com.sun.crypto.provider.CipherCore.unpad(CipherCore.java:1028)
        at java.base/com.sun.crypto.provider.CipherCore.fillOutputBuffer(CipherCore.java:1109)
        at java.base/com.sun.crypto.provider.CipherCore.doFinal(CipherCore.java:906)
        at java.base/com.sun.crypto.provider.DESedeCipher.engineDoFinal(DESedeCipher.java:294)
        at java.base/javax.crypto.Cipher.doFinal(Cipher.java:2202)
        at psdi.util.MXCipher.decData(MXCipher.java:293)
        at psdi.tools.ReEncryptFromSecret.doCryptoUpdates(ReEncryptFromSecret.java:210)
        at psdi.tools.ReEncryptFromSecret.updateCryptoFields(ReEncryptFromSecret.java:147)
        at psdi.tools.UpgradeEncryption.processEncryption(UpgradeEncryption.java:151)
        at psdi.tools.UpgradeEncryption.process(UpgradeEncryption.java:88)
        at psdi.tools.UpdateDB.main(UpdateDB.java:3430)
system#major  A major exception has occurred. Check the system log to see if there are any companion errors logged. Report this error to your syst
psdi.util.MXApplicationException: system#major
        at psdi.tools.ReEncryptFromSecret.doCryptoUpdates(ReEncryptFromSecret.java:260)
        at psdi.tools.ReEncryptFromSecret.updateCryptoFields(ReEncryptFromSecret.java:147)
        at psdi.tools.UpgradeEncryption.processEncryption(UpgradeEncryption.java:151)
        at psdi.tools.UpgradeEncryption.process(UpgradeEncryption.java:88)
        at psdi.tools.UpdateDB.main(UpdateDB.java:3430)
Caused by: psdi.util.MXSystemException: system#major
        at psdi.util.MXCipher.decData(MXCipher.java:299)
        at psdi.tools.ReEncryptFromSecret.doCryptoUpdates(ReEncryptFromSecret.java:210)
        ... 4 more
Caused by: javax.crypto.BadPaddingException: Given final block not properly padded. Such issues can arise if a bad key is used during decryption.
        at java.base/com.sun.crypto.provider.CipherCore.unpad(CipherCore.java:1028)
        at java.base/com.sun.crypto.provider.CipherCore.fillOutputBuffer(CipherCore.java:1109)
        at java.base/com.sun.crypto.provider.CipherCore.doFinal(CipherCore.java:906)
        at java.base/com.sun.crypto.provider.DESedeCipher.engineDoFinal(DESedeCipher.java:294)
        at java.base/javax.crypto.Cipher.doFinal(Cipher.java:2202)
        at psdi.util.MXCipher.decData(MXCipher.java:293)
        ... 5 more
psdi.util.MXSystemException: system#major
        at psdi.util.MXCipher.decData(MXCipher.java:299)
        at psdi.tools.ReEncryptFromSecret.doCryptoUpdates(ReEncryptFromSecret.java:210)
        at psdi.tools.ReEncryptFromSecret.updateCryptoFields(ReEncryptFromSecret.java:147)
        at psdi.tools.UpgradeEncryption.processEncryption(UpgradeEncryption.java:151)
        at psdi.tools.UpgradeEncryption.process(UpgradeEncryption.java:88)
        at psdi.tools.UpdateDB.main(UpdateDB.java:3430)
Caused by: javax.crypto.BadPaddingException: Given final block not properly padded. Such issues can arise if a bad key is used during decryption.
        at java.base/com.sun.crypto.provider.CipherCore.unpad(CipherCore.java:1028)
        at java.base/com.sun.crypto.provider.CipherCore.fillOutputBuffer(CipherCore.java:1109)
        at java.base/com.sun.crypto.provider.CipherCore.doFinal(CipherCore.java:906)
        at java.base/com.sun.crypto.provider.DESedeCipher.engineDoFinal(DESedeCipher.java:294)
        at java.base/javax.crypto.Cipher.doFinal(Cipher.java:2202)
        at psdi.util.MXCipher.decData(MXCipher.java:293)
        ... 5 more
[developer@mx761 maximo]$
```

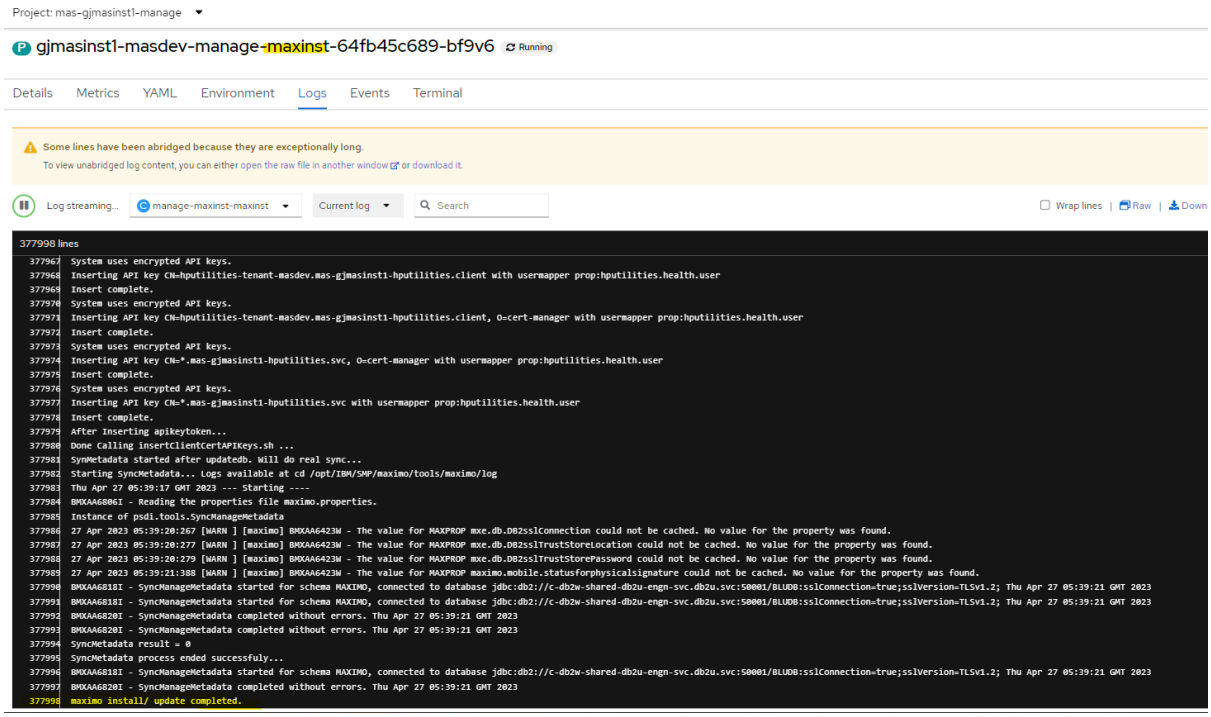
b. Trigger UpdateDB

To trigger `updatedb`, you can manually do it via `maxinst's` terminal. Or just delete the `maxinst` pod from the Mange namespace and let it start again. Instructions to delete the Pod are like deleting the DB2 pod.

MAS Manage Namespace	`mas-gjmasinst1-manage`
MAS Manage MAXINST Pod name	`gjmasinst1-masdev-manage-maxinst-<random_chars>-<random_chars>`

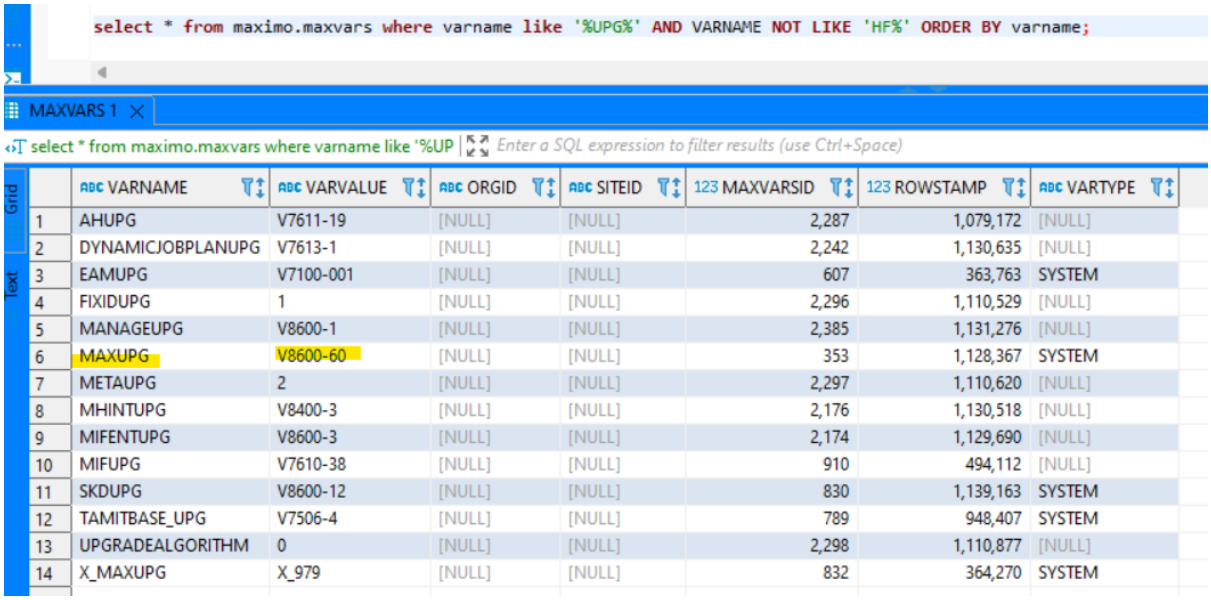


The new `maxinst` pod starts automatically, and you can monitor the logs for `updatedb` logs:



Let the `updatedb` finish successfully. You may have to troubleshoot the `updatedb` if it encounters issues and restart the `updatedb` manually or by deleting the `maxinst` pod.

The MAXVARs after the upgrade:



	VARNAME	VARVALUE	ORGID	SITEID	MAXVARSID	ROWSTAMP	VARTYPE
1	AHUPG	V7611-19	[NULL]	[NULL]	2,287	1,079,172	[NULL]
2	DYNAMICJOBPLANUPG	V7613-1	[NULL]	[NULL]	2,242	1,130,635	[NULL]
3	EAMUPG	V7100-001	[NULL]	[NULL]	607	363,763	SYSTEM
4	FIXIDUPG	1	[NULL]	[NULL]	2,296	1,110,529	[NULL]
5	MANAGEUPG	V8600-1	[NULL]	[NULL]	2,385	1,131,276	[NULL]
6	MAXUPG	V8600-60	[NULL]	[NULL]	353	1,128,367	SYSTEM
7	METAUPG	2	[NULL]	[NULL]	2,297	1,110,620	[NULL]
8	MHINTUPG	V8400-3	[NULL]	[NULL]	2,176	1,130,518	[NULL]
9	MIFENTUPG	V8600-3	[NULL]	[NULL]	2,174	1,129,690	[NULL]
10	MIFUPG	V7610-38	[NULL]	[NULL]	910	494,112	[NULL]
11	SKDUPG	V8600-12	[NULL]	[NULL]	830	1,139,163	SYSTEM
12	TAMITBASE_UPG	V7506-4	[NULL]	[NULL]	789	948,407	SYSTEM
13	UPGRADEALGORITHM	0	[NULL]	[NULL]	2,298	1,110,877	[NULL]
14	X_MAXUPG	X_979	[NULL]	[NULL]	832	364,270	SYSTEM

c. Generate APIKEYTOKEN

The `resetcryptocryptox.sh` command executed earlier would have deleted the APIKEYTOKEN table rows, we need to regenerate the rows and as per the IBM TechNote Corrupted APIKEY after upgrade MAS to 8.8 (ibm.com). We do this now, to avoid “user synchronisation” errors, thus letting you login/use Manage app.

Using the OC CLI, login to MAXINST Pod’s terminal:

```
oc exec --stdin --tty gjmasinst1-masdev-manage-maxinst-64fb45c689-l267w -c manage-maxinst-maxinst -n mas-gjmasinst1-manage -- /bin/bash
```

Run the following commands:

```
cd /opt/IBM/SMP/maximo/tools/maximo
./insertapikey.sh -v'${MANAGE_APIKEY}' -m'prop:mxe.int.dfltuser'
./insertapikey.sh -v'${MANAGE_APIKEY}, O=cert-manager' -m'prop:mxe.int.dfltuser'
./insertapikey.sh -v'${HEALTH_APIKEY}' -m'prop:mxe.MASUserSyncAgentUser'
./insertapikey.sh -v'${HEALTH_APIKEY}, O=cert-manager' -m'prop:mxe.MASUserSyncAgentUser'
./insertapikey.sh -v'${PMI_APIKEY}' -m'prop:mxe.PMIUser'
./insertapikey.sh -v'${PMI_APIKEY}, O=cert-manager' -m'prop:mxe.PMIUser'
./insertapikey.sh -v'${PMI_APIKEY_NEW}, O=cert-manager' -m'prop:mxe.PMIUser'
./insertapikey.sh -v'${PMI_APIKEY_NEW}' -m'prop:mxe.PMIUser'
./insertapikey.sh -v'${EU_APIKEY}' -m'prop:hputilities.health.user'
./insertapikey.sh -v'${EU_APIKEY}, O=cert-manager' -m'prop:hputilities.health.user'
./insertapikey.sh -v'${EU_APIKEY_NEW}, O=cert-manager' -m'prop:hputilities.health.user'
./insertapikey.sh -v'${EU_APIKEY_NEW}' -m'prop:hputilities.health.user'
exit
```

Example:

```
bash-4.4$ cd /opt/IBM/SMP/maximo/tools/maximo
bash-4.4$
bash-4.4$
bash-4.4$ ./insertapikey.sh -v'${MANAGE_APIKEY}' -m'prop:mxe.int.dfltuser'
System uses encrypted API keys.
Inserting API key CN=*.mas-gjmasinst1-manage.svc with usermapper prop:mxe.int.dfltuser
Insert complete.
bash-4.4$ ./insertapikey.sh -v'${MANAGE_APIKEY}, O=cert-manager' -m'prop:mxe.int.dfltuser'
System uses encrypted API keys.
Inserting API key CN=*.mas-gjmasinst1-manage.svc, O=cert-manager with usermapper prop:mxe.int.dfltuser
Insert complete.
bash-4.4$ ./insertapikey.sh -v'${HEALTH_APIKEY}' -m'prop:mxe.MASUserSyncAgentUser'
System uses encrypted API keys.
Inserting API key CN=*.mas-gjmasinst1-health.svc with usermapper prop:mxe.MASUserSyncAgentUser
Insert complete.
bash-4.4$ ./insertapikey.sh -v'${HEALTH_APIKEY}, O=cert-manager' -m'prop:mxe.MASUserSyncAgentUser'
System uses encrypted API keys.
Inserting API key CN=*.mas-gjmasinst1-health.svc, O=cert-manager with usermapper prop:mxe.MASUserSyncAgentUser
Insert complete.
bash-4.4$ ./insertapikey.sh -v'${PMI_APIKEY}' -m'prop:mxe.PMIUser'
System uses encrypted API keys.
Inserting API key CN=predict-tenant-masdev.mas-gjmasinst1-predict.client with usermapper prop:mxe.PMIUser
Insert complete.
bash-4.4$ ./insertapikey.sh -v'${PMI_APIKEY}, O=cert-manager' -m'prop:mxe.PMIUser'
System uses encrypted API keys.
Inserting API key CN=predict-tenant-masdev.mas-gjmasinst1-predict.client, O=cert-manager with usermapper prop:mxe.PMIUser
Insert complete.
bash-4.4$ ./insertapikey.sh -v'${PMI_APIKEY_NEW}, O=cert-manager' -m'prop:mxe.PMIUser'
System uses encrypted API keys.
Inserting API key CN=*.mas-gjmasinst1-predict-ws.svc, O=cert-manager with usermapper prop:mxe.PMIUser
Insert complete.
bash-4.4$ ./insertapikey.sh -v'${PMI_APIKEY_NEW}' -m'prop:mxe.PMIUser'
System uses encrypted API keys.
Inserting API key CN=*.mas-gjmasinst1-predict-ws.svc with usermapper prop:mxe.PMIUser
Insert complete.
bash-4.4$ ./insertapikey.sh -v'${EU_APIKEY}' -m'prop:hputilities.health.user'
System uses encrypted API keys.
Inserting API key CN=hputilities-tenant-masdev.mas-gjmasinst1-hputilities.client with usermapper prop:hputilities.health.user
Insert complete.
bash-4.4$ ./insertapikey.sh -v'${EU_APIKEY}, O=cert-manager' -m'prop:hputilities.health.user'
System uses encrypted API keys.
Inserting API key CN=hputilities-tenant-masdev.mas-gjmasinst1-hputilities.client, O=cert-manager with usermapper prop:hputilities.health.user
Insert complete.
bash-4.4$ ./insertapikey.sh -v'${EU_APIKEY_NEW}, O=cert-manager' -m'prop:hputilities.health.user'
System uses encrypted API keys.
Inserting API key CN=*.mas-gjmasinst1-hputilities.svc, O=cert-manager with usermapper prop:hputilities.health.user
Insert complete.
bash-4.4$ ./insertapikey.sh -v'${EU_APIKEY_NEW}' -m'prop:hputilities.health.user'
```

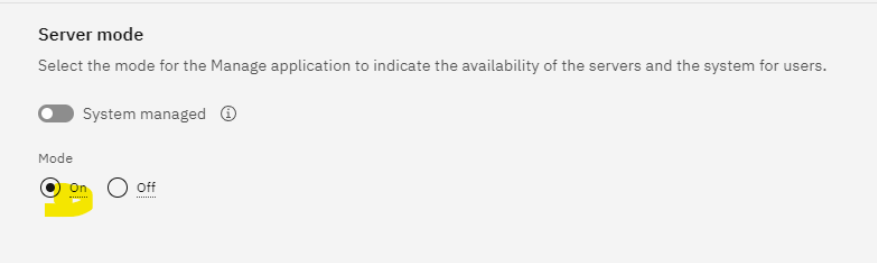
```
bash-4.4$ ./insertapikey.sh -v'${EU_APIKEY_NEW}' -m'prop:hputilities.health.user'
System uses encrypted API keys.
Inserting API key CN=*.mas-gjmasinst1-hputilities.svc with usermapper prop:hputilities.health.user
Insert complete.
bash-4.4$
```

d. Backup DB

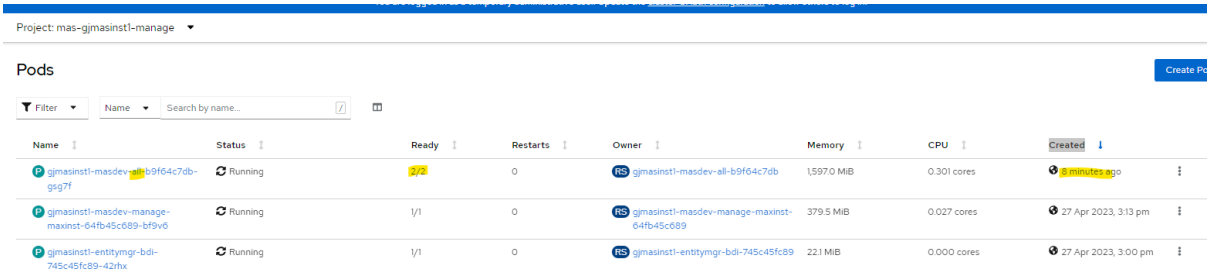
We have now completed the Maximo DB Upgrade. You may now take a DB backup using the commands described above in this document.

e. Server Mode Up

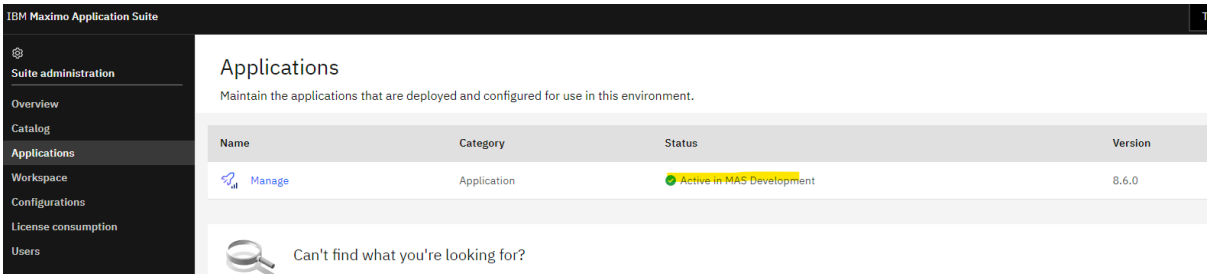
Like steps described in turning the server mode Off above, now you can turn the server mode On



After starting the activation, you can monitor the pods in manage namespace for the “all” pod status:

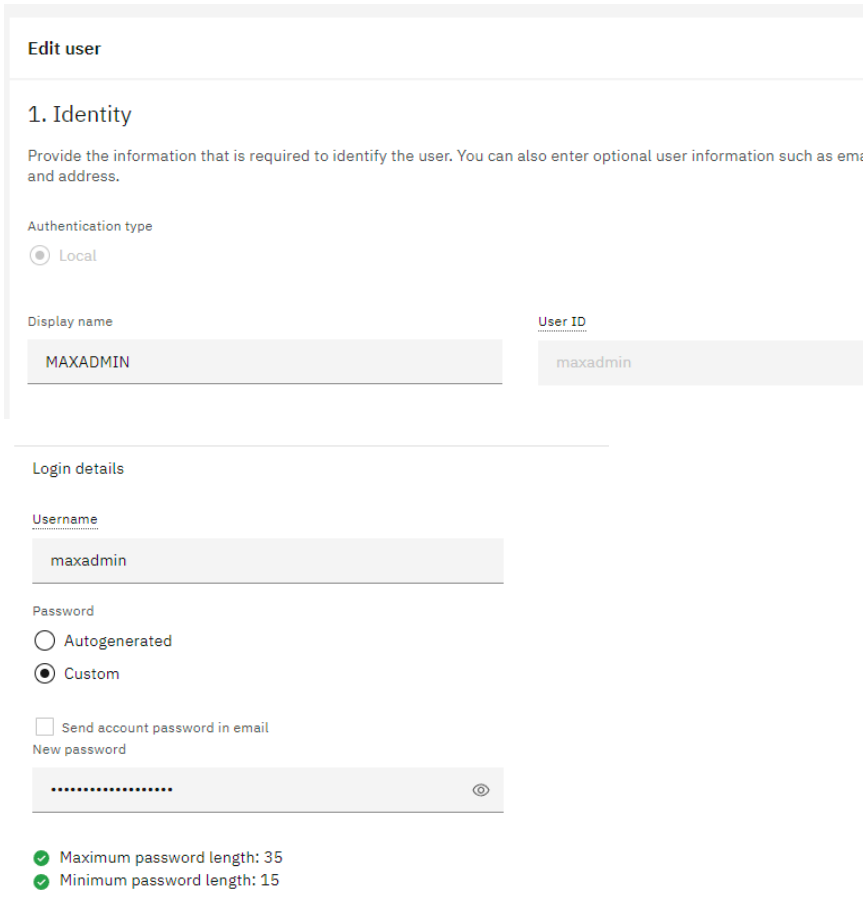


In MAS Console:

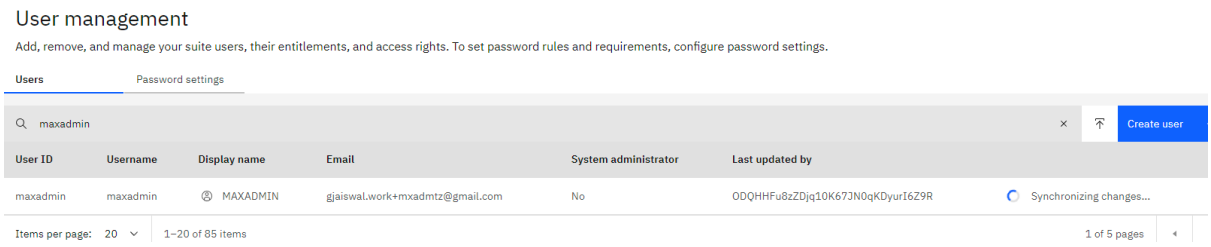


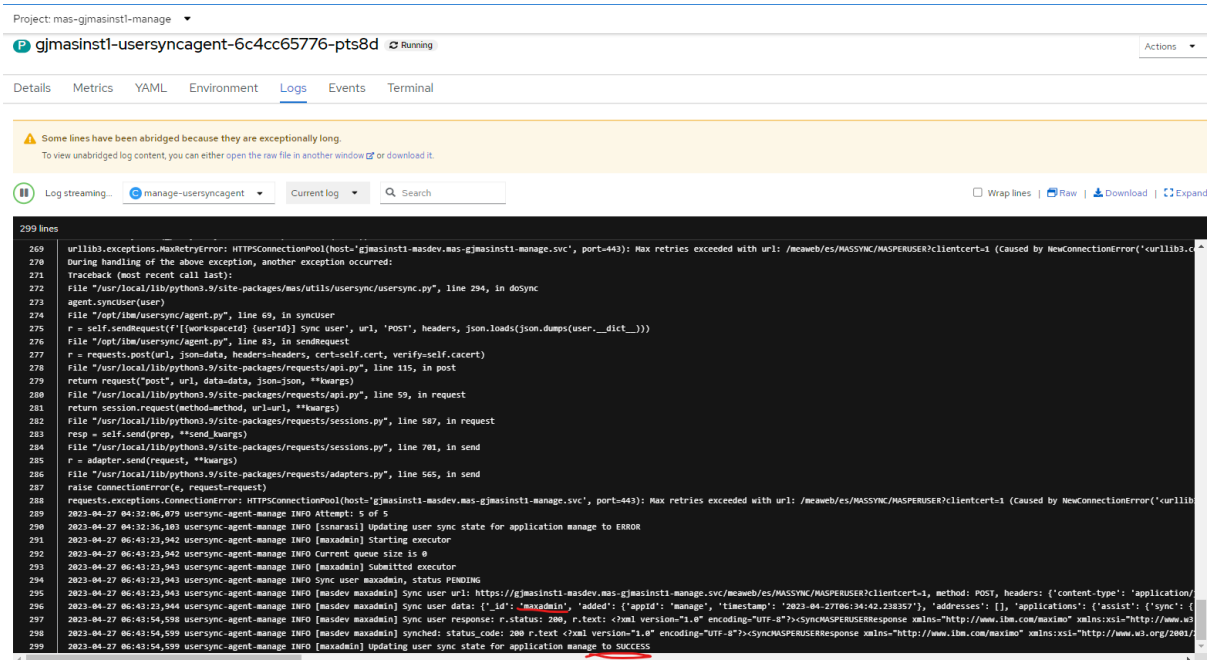
f. Setup Users

You may now setup the users, or atleast the password for maxadmin user and let it synchronise successfully.

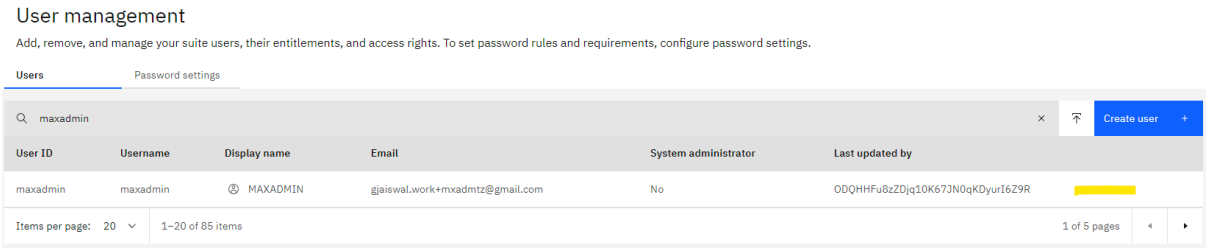


Save and let it synchronise the users.

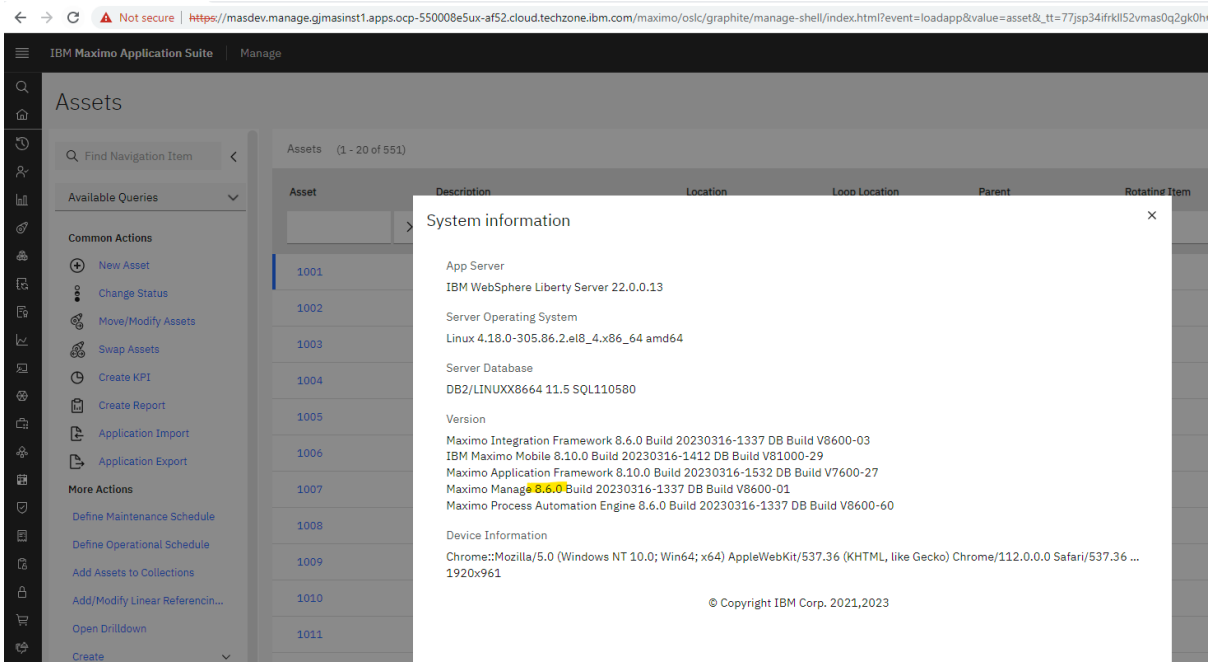




User is now synchronised:



g. Test MAXADMIN login



h. Clean-up

You may now cleanup the contents of `/mnt/backup/mx76exp` folder in the DB2 pod, as the database import and upgrade are now complete.

10. Applying Customisation

The customisation archive prepared earlier in this process can now be applied. Refer to the section above for sample deployment of customisations.

11. Next Steps

This document deals with Upgrade and Customisations, further setup like Doclinks, LDAP, SAML, Integration, etc would be needed and can be achieved using standard IBM Manage documentation.