Steps for In-situ data porting

Content

1. Introduction ….. 1

2. Machine Requirements ….. 1

3. Prerequisites ….. 2

4. Step 1: Upgrading from Gingko to Hawthorn ….. 3

4.1 Generate database reports for the old release ….. 3

4.2 Upgrade Data from Old release to New release ….. 4

4.3 Generate database reports for the new release ….. 10

4.4 Compare data of old and new releases ….. 10

4.4.1 Steps to compare data reports of both the releases ….. 11

4.4.2 Comparison Study ….. 12

4.4.3 Observations during the DryRun round in the production environment; and their solution …..14

5. Step 2: From Hawthorn.2 to Ironwood.master ….. 16

5.1 Comparison Study of Hawthorn and Ironwood ….. 17

6 General Remark ….. 19

6.1 Last go-live round report analysis ( Ficus to Gingko) for reference ….. 19

6.2 Last go-live round Mongo report analysis ( Ficus to Gingko) ….. 20

7. Migrating Custom-Field-data of the Registration page ….. 21

8. Additional Checking ….. 22

9. Troubleshooting ….. 23

10. Extra Information ….. 25

11. References ….. 26

12. Appendix ….. 27

Steps for In-situ data porting

# Introduction

This document describes the steps for moving and upgrading data from the *current release* of Open edX to the next release.

Here we describe the steps of moving and upgrading of data from **Gingko.2** to **Ironwood.master** which will be done in two steps as there is a Hawthorn release available before Ironword.master.

Step 1: Upgrading from Gingko.2 to Hawthorn.2.

Step 2: Upgrading from Hawthorn.2 to Ironwood.master.

These steps will be described in later sections. In the later section of the document, we will be using the terminology **old release** for *current release and* ***new release*** for *next release* terms used in the beginning of the document.

# Machine Requirements

Below are the machines required for the process.

* A **Ginkgo.2** machine with live production data dump. It must have the same configurations and settings as in the live production Gingko machine. Also, the **Custom Fields for the registration page** (Version.0) application must be installed in the Gingko.2 machine.
* A **Hawthorn.2** new machine with the same configurations and settings as it will be in the live production machine with the live data of the Ginkgo.2(MySQL and Mongo) machine. The **Custom fields for the registration page** application should not be installed.
* An **Ironwood.master** new machine with the same configurations and settings as it will be in the live production machine. The **Custom fields for the registration page** application should not be installed.
* A **remote machine** that has all the privileges to access the above machines and their databases(MySQL and Mongo)

# Prerequisites

1. Stop the services of the Ginkgo.2, Hawthorn.2 and Ironwood.master machines and verify if all the services are stopped.

*# To stop the services*

***sudo /edx/bin/supervisorctl stop all***

*# To Confirm the status of the services*

***sudo /edx/bin/supervisorctl status all***

1. Check whether MySQL **root user permission is given** **to all the databases** of three machines( Gingko, Hawthorn, Ironwood) and can be accessed by the remote machine. Make sure root user permissions must be created in all the databases,

*# On the Remote Machine*

***mysql -u<new\_username> -p -h<host-ip-address>***

***show databases;***

*# Exit from MySQL*

1. Check the character set of all user databases, all user tables and columns must be utf8.

*# To*Check the character set of all databases.

***select \* from SCHEMATA;***

*# To*Check the character set of all tables of all databases.

***SELECT count(\*), table\_collation, table\_schema***

***FROM information\_schema.`TABLES`***

***group by table\_collation, table\_schema***

***order by table\_schema;***

*# To*Check the character set of all columns of all tables of all databases.

***SELECT count(\*), character\_set\_name, TABLE\_NAME, table\_schema***

***FROM information\_schema.`COLUMNS`***

***group by character\_set\_name, TABLE\_NAME, table\_schema***

***order by table\_schema, TABLE\_NAME;***

1. Check whether **root user permission to Mongo databases** of three machines( Gingko, Hawthorn, Ironwood) is available and can be accessed by the remote machine. Make sure root user permissions must be created in all theMongo servers,

*# On the Remote Machine*

***mongo -u <user\_name> -p <password> <host-ip-address>:<mongo-port>/<db\_name> --authenticationDatabase admin***

***show dbs;***

*# Exit from Mongo*

1. Confirm Is the SECRET\_KEY added in the configuration files /edx/app/edxapp/lms.auth.json, /edx/app/edxapp/cms.auth.json and my-passwords.yml etc. of the ironwood machine?

***sudo vi /edx/app/edxapp/lms.auth.json***

***sudo vi /edx/app/edxapp/cms.auth.json;***

# Step 1: Upgrading from Gingko to Hawthorn

Upgrading from Gingko.2 to Hawthorn.2 is an intermediate step for going to Ironwood.master, We will consider

Ginkgo.2 : as Old Release( which was mentioned as current release in the beginning of document).

Hawthorn.2 : as a New Release or next release.

The upgradation process is divided into four parts.

1. **Generate database reports for the old release:**  Run data report script to generate data reports of old release (Ginkgo.2) data.
2. **Upgrade data from the old release to the new release:** This section is vital, it has the steps to upgrade the data from the old release(Ginkgo.2) to the new release(Hawthorn.2). This section can also be used independently.
3. **Generate database reports for the new release:** Run the data report script to generate reports of the new release (Hawthorn.2).
4. **Compare data of old and new releases:**  Compare the data reports of the old release(Ginkgo.2) machine and reports of the new release(Hawthorn.2) machine to verify and confirm the up-gradation process from the old release(Ginkgo.2) to the new release(Hawthorn.2) has been done successfully.

The data migration process is described in detail in the further sections. This Step-1 has been prepared for upgrading from the old release(Ginkgo.2) to the new release (Hawthorn.2) keeping in mind that it can be used in the subsequent releases using a very similar approach.

# Generate database reports for the old release

## In this section, requirements and process of creating reports for the data available in the old release machine is explained. The script (report script) is prepared to generate different reports. The report script can be run from the remote machine. The generated data reports will be stored in a defined location mentioned in the parameter.config file. The result of this report will be used for comparing the similar report’s result generated after porting data in the new release.

Do the steps mentioned in prerequisites and then follow the process given below to generate the database reports from the remote machine.

1. Clone the git repository:

*# Make a folder named*  ***migration*** *on the remote**machine*

***mkdir migration***

***cd migration***

***git clone http://<username>@gitlab.cse.iitb.ac.in/IITBombayX-Utilities/iitbombayxdbupgrade.git***

***cd iitbombayxdbupgrade***

***git checkout Ginkgo2ToIronwood***

1. Run the **data report script** for generating MySQL and Mongo database reports by following the steps below.
   1. *Change directory*

***cd Script/ReportScript***

* 1. *Set the parameters*

*# Set the parameters in* ***parameter.config*** *file and run the script as per the instructions given in README.md file,*

***vi parameter.config***

For the data report from **G**inkgo to **H**awthorn, set *UpgradingType="****GH****"*, result path, MySQL server detail(*IP, port and user credential*) and Mongo server detail(*IP, port and user credential*) of the old release machine as per the instruction in the ***parameter.config*** file.

After changing the parameters, save and exit.

* 1. Run the **data report script**

***bash data\_report.sh***

# Upgrade Data from Old release to New release

This section describes the requirements and process of upgrading data from old release( Ginkgo.2) to new release (Hawthorn.2). This section can also be used independently.The upgradation process is the same as described in the OpenEdX upgradation document.

The recommended approach to upgrade an existing installation of the Open edX Ginkgo release to the Hawthorn release is by making a fresh installation of the Hawthorn release on a new machine, and move your data and settings to it.

To move and upgrade your Ginkgo data onto a Hawthorn installation, follow these steps.

1. Be sure that your Ginkgo installation is on the latest open-release/ginkgo.master. This ensures that your database is fully migrated and ready for upgrade to Hawthorn. *We are using Gingko.2 in place of Gingko.master*
2. Stop all services on the Ginkgo machine( old release).
3. Dump the data on the **Ginkgo machine( old release)**. The script is available that will dump the MySQL and Mongo databases into a single .tgz file. Below are the steps.

* *Make sure there is enough space (as much as the data size of MySQL and Mongo databases ) for a data dump. To* check the available free disk space

***df -h***

***free***

* *Make a folder.*

*# Make the “mig” folder on the maximum available space path.*

***mkdir mig***

* *Change* the directory

***cd mig***

* *Fetch the the folder “iitbombayxdbupgrade/Script/DumpScript” from the remote machine, run the following command.*

***scp -r <user\_remote\_machine>@<IP\_remote\_machine>:<path\_of\_DumpScript\_folder> .***

* Change the directory

***cd DumpScript***

* Set MySQL server detail(*IP, port and user credential*) and Mongo server detail(*IP, port and user credential*) of the old release machine in the ***parameter.config*** file. R*un the script as per the instructions given in README.md file*

***vi parameter.config***

After changing the parameters, save and exit

* *Run the script as per the instructions given in README.md file. (script is provided by Open edX)*

***bash dump.sh***

**Note:** After running the above script, check the current working directory.

* A folder is created, with a name "mongo-dump-<timestamp -yyyymmddThhmmss>", eg., "mongo-dump-20180814T164638". It contains files having a dump of mongo databases.
* A .sql file with a name "mysql-structure-data-<timestamp-yyyymmddThhmmss>.sql" is created, eg., "mysql-structure-data-20180814T164638.sql". It is a dump of all MySQL user databases.
* A .tgz file is created, with a name "openedx-data-<timestamp -yyyymmddThhmmss>", eg., "openedx-data-20180814T164638.tgz", which is tar of all dump of MySQL and Mongo databases ( mentioned above)).

1. Make sure enough space is there in **New release machine( hawthorn)**
2. To check the available free disk space on the new release(Hawthorn.2) machine for restoring the dump of the old release(Ginkgo.2)

***df -h***

***free***

1. create a folder ‘migrationDump’ in the new release machine and go to the directory.

***mkdir migrationDump***

***cd migrationDump***

1. Copy the generated .tgz data file here in the new release( hawthorn) machine.
2. Stop all services on the new release (Hawthorn ) machine.
3. Restore the old release(Ginkgo.2) data to the new release(Hawthorn.2) machine. Use the following commands in the terminal

***tar -xvf openedx-data-<20180814T164638>.tgz***

***mysql -uroot -p < mysql-structure-data-<20180814T164638>.sql***

***mongorestore -u admin -p password -h localhost --authenticationDatabase admin --drop -d edxapp mongo-dump-<20180814T164638>/edxapp***

***mongorestore -u admin -p password -h localhost --authenticationDatabase admin --drop -d cs\_comments\_service mongo-dump-<20180814T164638>/ cs\_comments\_service***

# *After restore, delete the unzip files to release the disk space.*

***rm mysql-structure-data-20180814T164638.sql***

***rm -r mongo-dump-20180814T164638/***

1. Verify the character set of all user databases and all user tables and all user columns. It should have an **“utf8”** character set.
2. To migrate data from old release (Ginkgo) to new release (Hawthorn), we need to drop the database tables used by djcelery. These tables should be empty in the Ginkgo data, so it is safe to drop them. The edx-platform application has a management command to check that they are empty and drop them. While dropping the database tables used by djcelery using the management command, errors were received due to the foreign key constraint and some missing tables on the Hawthon.2 and Ironwood.master machine in the development environment. To avoid the errors, we tried a solution which is described below
   1. Restore all djcelery tables
   2. Set the foreign key constraint OFF of MySQL database.
   3. Run the management command provided by OpenedX to drop the database tables used by djcelery.
   4. Set the foreign key constraint ON of MySQL database.

The solutions mentioned above are explained below

1. Restore all the djcelery tables:

*#To restore all djcelery tables, Copy the file “iitbombayxdbupgrade/Dump/Data/celery\_tables.sql” from the git directory of the remote machine, to Gingko machine and run following commands*

***scp -r <user\_remote\_machine>@<IP\_remote\_machine>:<path\_of\_celery\_tables\_file> .***

***mysql -uroot -p < celery\_tables.sql***

The below steps are used for migrating the restore data for the Single-Instance machine in the development environment. For migrating to the live-production- data in a distributed instance in the production environment refer to the points mentioned at the end of this subsection.

1. Before dropping djcelery tables, set FOREIGN\_KEY\_CHECKS to OFF in MySQL.

***mysql -uroot -p***

***select @@global.FOREIGN\_KEY\_CHECKS;***

***select @@FOREIGN\_KEY\_CHECKS;***

***SET global FOREIGN\_KEY\_CHECKS=0;***

***SET FOREIGN\_KEY\_CHECKS=0;***

***select @@global.FOREIGN\_KEY\_CHECKS;***

***select @@FOREIGN\_KEY\_CHECKS;***

***exit***

1. The edx-platform application has a management command to check that they are empty and drop them:

***sudo su - -s /bin/bash edxapp***

***. edxapp\_env***

***cd edx-platform/***

***python manage.py lms drop\_djcelery\_tables --settings=aws***

***exit***

1. Reset the foreign key constraint of MySQL database.

***mysql -uroot -p***

***SET global FOREIGN\_KEY\_CHECKS=1;***

***SET FOREIGN\_KEY\_CHECKS=1;***

***select @@global.FOREIGN\_KEY\_CHECKS;***

***select @@FOREIGN\_KEY\_CHECKS;***

***exit***

1. Verify the openedx version in ‘/var/tmp/configuration’ directory: output should be “**\* (HEAD detached at open-release/hawthorn.2)**”.
   1. Go to the directory ‘/var/tmp/configuration’

***cd /var/tmp/configuration***

* 1. Check the git branch

***git branch***

# *If it is not \* (HEAD detached at open-release/hawthorn.2), then*

***git checkout open-release/hawthorn.2***

* 1. Check ‘git diff’ in “/var/tmp/configuration" directory. There should be no difference.

***git diff***

* 1. Go to the home directory

***cd***

1. Be sure internet service is on, to check login on internet proxy server as

***lynx internet.iitb.ac.in***

1. Set permission to /edx/app/edx\_ansible/edx\_ansible/util/install/native.sh file.

# assign all permission to the file

***sudo chmod 777 /edx/app/edx\_ansible/edx\_ansible/util/install/native.sh***

1. Set an environment variable. it should be as git branch of “/var/tmp/configuration” git repository :

***export OPENEDX\_RELEASE=open-release/hawthorn.2***

# check by printing variable

***echo $OPENEDX\_RELEASE***

1. Set an environment variable **OPENEDX\_RELEASE=open-release/hawthorn.2** before line “CONFIGURATION\_VERSION=${CONFIGURATION\_VERSION-$OPENEDX\_RELEASE}” in /edx/app/edx\_ansible/edx\_ansible/util/install/native.sh file.

***sudo vi /edx/app/edx\_ansible/edx\_ansible/util/install/native.sh***

***OPENEDX\_RELEASE=open-release/hawthorn.2***

***echo ${OPENEDX\_RELEASE}***

***CONFIGURATION\_VERSION=${CONFIGURATION\_VERSION-$OPENEDX\_RELEASE}***

1. Run the new release(Hawthorn) migration script, which will upgrade old release(Ginkgo) data to be valid for new release(Hawthorn):

***/edx/app/edx\_ansible/edx\_ansible/util/install/native.sh --tags migrate***

It may ask for following :

* **Mail configuration**: Select **no configuration** option.
* **Configuring grub-pc**: Select **no configuration** option.
* **Some package updated version:** Select **currently-installed version** option.
* Password for edx user

1. This section can run independently. If this section is used independently( if data report is not to be generated) then **restart all services**. Otherwise, **skip this step** and go to the next section to generate the data report for the new release.

*#to restart all the services*

***sudo /edx/bin/supervisorctl start all***

**Upgradation Process** for the **live-production-data** for the distributed instance in the production environment using ansible command. Ansible commands are run by the sysadmin team.

1. Confirm the git branch of the repository used.
2. Enable logs in the “ansible” for analysis before the migration process starts.
3. Drop the database tables used by djcelery.
4. As per a single instance machine in the development environment, the sequence of tasks to upgrade data using the openedx “***native.sh***” script is given below and the same can be followed for a distributed environment.
   1. Edxapp (lms and cms)
   2. insights
   3. forum
   4. xqueue

# Generate database reports for the new release:

## In this section, requirements and process of creating database reports of the ported data in the new release is explained. The script(report script) can be used to generate different reports. and can be run from the remote machine. The generated database reports will be stored in a defined location mentioned in the parameter.config file. This report will be used for comparing the similar reports generated before porting data of the old release.

**Check the prerequisites. Follow the steps mentioned in section 4.1** for generating the database reports for the new release from the remote machine. Use the new machine credentials instead of old machine credentials wherever required, generate data reports on the old release path, so that it will be easier to compare.

# Compare data of old and new releases

This section compares the data reports of the old release (generated in 4.1) and the new release ( generated in 4.3) to verify if the migration of data is proper. The script is prepared which will compare the generated reports of both the releases and shows the differences if any. It also compares the found differences with expected differences and generates a conclusion.

The expected difference is the changes in the data from the old release to the new release, that has been noted after porting data in the development setup. These changes are kept in the folders which are inside the “ReportScript” folder.

For the data report from the ***G***inkgo.2 data to ***H***awthorn.2 S*et UpgradingType="****GH****"* in the ***parameter.config*** *file.*

# Steps to compare data reports of both the releases

1. Check whether the database report results of the old release(Ginkgo) machine and the new release(Hawthorn) machine is available in the git repository.
2. Clone the git repository: If you already have a clone of the git repository on the remote machine then skip the below steps to clone the git repository.

*# Make folder migration on the remote**machine*

***mkdir migration***

***cd migration***

***git clone http://<username>@gitlab.cse.iitb.ac.in/IITBombayX-Utilities/iitbombayxdbupgrade.git***

***cd iitbombayxdbupgrade***

***git checkout Ginkgo2ToIronwood***

1. Copy Report Results: Go to the git repository directory copied in the remote machine. Check if the data report results(“ReportResult**GH/**Before” and “ReportResult**GH/**After”) is available in the result folder(It is set in parameter.config file) which was created in the section 4.1 or the section 4.3.

**Run script:** Run the **findDiff script** for getting the differences between the database report result available in the “Before" folder and in the “After" folder.

***cd Scripts/ReportScript***

To compare the data report from **G**inkgo to **H**awthorn, set *UpgradingType="****GH****"*and result path as per the instruction in the ***parameter.config*** file.

# *Set the parameter in the parameter.config file as per instructions given in the "README.md" file,*

*vi* ***parameter.config***

*Save and exit, and run the script.*

***bash findDiff.sh***

**Output:** After running this script, the “Conclusion” folder is created in the result path which is set in the “parameter.config” file. The Conclusion folder has the result of the comparison.

# Comparison Study:

In the live setup, data is distributed on more than one server. It depends on the size of data and the server setup. Below study has been done for a single instance machine.

**For MySQL:** The table only shows the results which we need to verify.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Before migration** | **After migration** | **Reason** | **Action to be taken** |
| difference in the number of rows in django\_migrations tables of all database(diff\_rows\_django\_migrations.txt) |  |  | No difference is expected.  If there is any difference, compare the “<result-path>/ReportResult**GH**/After/mysqlReports\_<ip\_timestamp>/rows\_django\_migrations.txt" file with “iitbombayxdbupgrade/Script/ReportScript/MySQL\_ExpectedResultGH/rows\_django\_migrations.txt" file.  If the rows count is more in “<result-path>/ReportResultGH/After/mysqlReports\_<ip\_timestamp>/rows\_django\_migrations.txt" file than the "iitbombayxdbupgrade/Script/ReportScript/MySQL\_ExpectedResultGH/rows\_django\_migrations.txt" file, then we can continue migration otherwise, we should stop migration.  Need to identify the reasons for the difference. |  |
| Not deleted tables |  |  | No issue, if some tables aren't deleted. |  |

Table 1. MySQL table data comparison ( gingko to hawthorn): Expected

**General remark:** Need to study the differences that got in last go-live round

**For MongoDB:** The table only shows the results which we need to verify.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Before migration** | **After migration** | **Reason** | **Action to be taken** |
| “system.indexes” collection of the “cs\_comments\_service” database. | 6 | 0 | It is due to the Mongo version change. | No |
| “system.indexes” collection of the “edxapp” database. | 8 | 0 | It is due to the Mongo version change. | No |
| diff\_diff\_output\_cs\_comments\_service.txt |  |  | If it is due to the timestamp and IP of the machine. | No |
| diff\_diff\_output\_edxapp.txt |  |  | If it is due to the timestamp and IP of the machine. | No |

Table 2. Mongo table data comparison ( gingko to hawthorn): Expected

# Observations during the DryRun round in the production environment; and their solution

Discovery database has actually 103 tables of “utf8” character set for the Ginkgo release, but we found 96 tables with "latin1" character set in the production setup. It didn't have user data so we replaced it with a fresh discovery database for the Ginkgo.2 release.

The solutions mentioned above are explained below

Replace the discovery database:

*#To restore the* discovery *tables, Copy the file “iitbombayxdbupgrade/Dump/Fresh/Ginkgo2/G27/mysql-structure-data-discovery-20190906T113755.sql” from the git directory of the remote machine, to the Hawthorn machine and run following commands*

***scp -r <user\_remote\_machine>@<IP\_remote\_machine>:<path\_of\_discovery\_dump\_file> .***

***mysql -uroot -p < mysql-structure-data-discovery-20190906T113755.sql***

**General Remark:** Need to study the differences that got in last go-live round

# 

# Step 2: From Hawthorn.2 to Ironwood.master

After upgrading and verifying the live data from Ginkgo.2 to Hawthorn.2 , the same approach has to be followed for upgrading data from Hawthorn.2 to Ironwood.master. For this process our old release and new release will be as below,

* Old Release: Hawthorn.2
* New Release: Ironwood.master

The Upgradation process is the same we did in section 4. We need to add custom fields in the registration page after upgrading data from old release to new release.

1. **Generate database reports for the old release:** Run data report script to generate reports for the old release (Hawthorn.2) data, follow all the steps mentioned in Section 4.1. Use Hawthorn credential details in place of gingko details and set *UpgradingType="HI"* in the parameter.config file to generate the data reports.
2. **Upgrade data from the old release to the new release:** To upgrade data of the old release(Hawthorn.2) to the new release(Ironwood.master), this section is vital. Follow all the steps mentioned in Section 4.2. Make sure to change the parameter mentioned in point 12. ***git branch of the "/var/tmp/configuration" directory should be "open-release/ironwood.master"****.*
3. **Generate database reports for the new release**: Run data report script to generate reports for the new release (Ironwood.master) data. Follow all the steps mentioned in Section 4.3. Use Ironwood.master details in place of Hawthorn.2 details and set *UpgradingType="HI"* in the parameter.config file to generate the data reports.
4. **Comparing data of old and new releases:** Compare the data reports of the old release(Hawthorn.2) machine and new release(Ironwood.master) machine to verify the up-gradation process as mentioned in Section 4.4.
5. ***Custom-Field-Data of the registration page:*** After upgrading data of old release Hawthorn.2 to the Ironwood.master, migrate ***Custom-Field-Data of the registration page*** by following the steps mentioned in section 6.

### 

# Comparison Study of Hawthorn and Ironwood

**For MySQL:** The table only shows the results which we need to verify.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Before migration** | **After migration** | **Reason** | **Action to be taken** |
| “auth\_user" table of the "edxapp" database |  | +1 | Ironwood.master has an additional JWT user, so there is an extra user in the table | No |
| difference in Mongo and MySQL users(diff\_diff\_mongo\_mysql\_user\_diff.txt) |  | +1 | Ironwood.master has an additional JWT user, so there is an extra user in the table | No |
| “waffle\_switch” table of “discovery” database |  | -1 | 5th entry “log\_course\_search\_queries” is deleted “waffle\_switch” table of “discovery” database in Ironwood.master. It is the expected difference. | No |
| difference in the number of rows in django\_migrations tables of all database(diff\_rows\_django\_migrations.txt) |  |  | No difference is expected.  If there is any difference, compare the “<result-path>/ReportResultGH/After/mysqlReports\_<ip\_timestamp>/rows\_django\_migrations.txt" file with “iitbombayxdbupgrade/Script/ReportScript/MySQL\_ExpectedResultGH/rows\_django\_migrations.txt" file.  If the rows count is more in “<result-path>/ReportResultGH/After/mysqlReports\_<ip\_timestamp>/rows\_django\_migrations.txt" file than the "iitbombayxdbupgrade/Script/ReportScript/MySQL\_ExpectedResultGH/rows\_django\_migrations.txt" file, then we can continue migration otherwise, we should stop migration.  Need to identify the reasons for the difference. | Yes |
| Not deleted tables |  |  | No issue, if some tables aren't deleted. | No |

Table 3. MySQL table data comparison (Hawthorn to Ironwood): Expected

**General remark:** For the "edxapp" database, there might be some differences due to the celery tables or due to the extra tables in production data. For other types of differences, we need to identify the reason behind it.

**For MongoDB:** The table only shows the results which we need to verify.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Before migration** | **After migration** | **Reason** | **Action to be taken** |
| diff\_diff\_output\_cs\_comments\_service.txt |  |  | If it is due to the timestamp and IP of the machine. | No |
| diff\_diff\_output\_edxapp.txt |  |  | If it is due to the timestamp and IP of the machine. | No |

Table 4. Mongo table data comparison (Hawthorn to Ironwood): Expected

# GeneralRemark**:**

### Last go-live round report analysis ( Ficus to Gingko) for reference

In upgrading the data from Ficus.4 to Ginkgo.2, three servers were used; one for Mongo; one for the "edxapp\_csmh" MySQL database; and one for all other databases of the MySQL.

**Findings in MySQL database:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Finding** | **Reason** | **Action taken** |
| New Database found | edx\_notes\_api |  | No |
| Blank database found | analytics-api  dashboard  programs |  | No |
| Missing table in database | discovery(96 instead of 103)  Missing tables in discovery database:  course\_metadata\_courseentitlement  course\_metadata\_courserun\_video\_translation\_languages  course\_metadata\_program\_instructor\_ordering  course\_metadata\_subjecttranslation  course\_metadata\_topic  course\_metadata\_topictranslation  publisher\_courseentitlement |  | No |
| Missing entries in Table | “waffle\_switch” table of discovery database  Following two entries were missing:   * use\_company\_name\_as\_utm\_source\_value * auto\_course\_about\_page\_creation |  | No |
| Database on different server | “edxapp\_csmh” | In the live environment, distributed architecture has been used for storing data. While upgrading data from ficus to Ginkgo, we observed the "edxapp\_csmh" MySQL database was located on a different server. | Yes, We ran the data report again for the “edxapp\_csmh” database server. Do the comments for the Mongo database script calls in the"data\_report.sh" script. |

Table 5. MySQL table data comparison for Ficus to Gingko release

### Last go-live round Mongo report analysis ( Ficus to Gingko)

|  |  |  |
| --- | --- | --- |
| **Test** | **Finding** | **Action taken** |
| Extra system.users collection in edxapp & cs\_comments\_service both user databases. | System Users should be dropped after migration but were present. | No |
| Extra system.indexes collection in edxapp & cs\_comments\_service both user databases. | The indexes related to the system.users collection were also present in Ginkgo.2.  These indexes were related to System Users. | No |

Table 6. Mongo collection data comparison for Ficus to Gingko release

While upgrading the data from ficus to Ginkgo , we observed some differences in live data and development setup data and hence the "finddiff.sh" script produced the differences in some files.

**Example**: Following files had differences while upgrading from Ficus to Gingko

*diff\_diff\_UGC\_analytics-api\_auth\_permission.txt diff\_diff\_UGC\_analytics-api\_django\_content\_type.txt diff\_diff\_UGC\_dashboard\_auth\_permission.txt diff\_diff\_UGC\_dashboard\_django\_content\_type.txt diff\_diff\_UGC\_dashboard\_django\_migrations.txt diff\_diff\_UGC\_edxapp\_csmh\_django\_migrations.txt*

*diff\_diff\_upgraded\_rows\_analytics-api.txt*

*diff\_diff\_upgraded\_rows\_dashboard.txt* in *~~/home/test/migration/dbutils/scripts/ReportConclusion/At\_20190717T163423/Mysql/~~*<path>folder.

It was due to the presence of a blank database, and also one database on the different server.

**Conclusion:** The similar differences can be found in the upgradation process of Ginkgo.2 to Hawthorn.2 and Hawthorn.2 to Ironwood.master. We need to study the comparison carefully to complete the upgradation process successfully.

# Migrating Custom-Field-data of the Registration page

This subsection describes the process to migrate the data of custom-field tables from the Ginkgo.2 release to the Ironwood.master release.

1. Remove old migration data from the MySQL database

*Go to the git repository directory copied on the remote machine*

***cd Script/Reg\_custom\_field\_data\_mig***

*# Set the parameters ( IP, MySQL database port and user credential of the Ironwood.master machine) in the parameter.config file as per instructions given in README.md file.*

***vi parameter.config***

*Save and exit.*

*#Run the script*

***bash remove\_G2\_custom\_reg\_data.sh***

1. Install the "Custom field for the registration page" Version.1 application on the Ironwood.master machine using “Custom fields for Registration page” Document.

**Reference**: https://docs.google.com/document/d/1YUfqAOB61gnxX-XV2oy7ncAoMqdLC5KB/edit#heading=h.gjdgxs

1. Migrate the custom-field tables data from the Ginkgo release to the Ironwood release.

*# Go to the git repository directory copied on the remote machine*

***cd Script/Reg\_custom\_field\_data\_mig***

*# Run the script*

***bash mig\_custom\_reg\_data\_GtoI.sh***

1. Restart all services in the new release machine(Ironwood.master) **only if you do not required to run the data report script** otherwise do not restart and follow the instructions provided in section 4.3 to generate data report for the new release Ironwood.master

*#to restart all the services*

***sudo /edx/bin/supervisorctl start all***

# 

# Additional Checking

1. Note down the git branches of git repositories, git branches should be as per the release version.

**Note:** For a single instance machine(Development Environment), Use the git branch script to check the git branches.

Go to the git repository directory copied on the remote machine

***cd Script/***

copy ListGitBranch folder on the single instance machine

***cd ListGitBranch***

and run the git branch script

**bash listGitBranch.sh**

1. To count records of all tables of user databases(Mysql and Mongo), use the record count script as follow

Go to the git repository directory copied on the remote machine

***cd Script/RecordCountScript***

*# Set the parameters ( IP, MySQL database port and user credential of the machine) in the parameter.config file as per instructions given in README.md file.*

***vi parameter.config***

*# Save and exit.*

And run the git branch script

***bash count\_rows.sh***

# Troubleshooting

* 1. The native.sh migration script provided by OpenedX for the single instance machine uses IP 127.0.0.1 as localhost to upgrade MySQL data. So the MySQL database must be accessible using host IP 127.0.01 from the localhost of the Open edX machine. If it is not so, the following error will occur.

Error: Error in the task “TASK [insights : migrate]” as below:

***"django.db.utils.OperationalError: (2003, \"Can't connect to MySQL server on '127.0.0.1' (111)\")"], "stdout": "", "stdout\_lines": []}***

Solution: Make sure that you are able to connect to mysql as follows:

**mysql -h 127.0.0.1 -u root -p**

***#****If not able to connect, modify the /etc/mysql/mysql.conf.d/mysqld.cnf file as below.*

***vi /etc/mysql/mysql.conf.d/mysqld.cnf***

***bind-address = 0.0.0.0***

#Restart the MySQL services

***sudo /etc/init.d/mysql restart***

* 1. **To change the** access of the library of the course:

**Library Access:** In the file */edx/app/edxapp/cms.env.json*, the default value for “ENABLE\_CREATOR\_GROUP” is set as *True* under “FEATURES“ .

#This can be checked by

***$ cd /edx/app/edxapp/***

***$ sudo vi cms.env.json***

# search for “ENABLE\_CREATOR\_GROUP“

#Value of parameter

**"FEATURES": {**

**"ENABLE\_CREATOR\_GROUP": True,**

**}**

This restricts the access of the library to non-admin users. If a requirement is there to access the library by a staff user, then change this parameter as **False** in the file */edx/app/edxapp/cms.env.json*

***$ cd /edx/app//***

***$ sudo vi cms.env.json***

# search for “ENABLE\_CREATOR\_GROUP“

# Set parameter

***"FEATURES": {***

***'ENABLE\_CREATOR\_GROUP': False,***

***}***

# Exit

#Restart the cms services as

***$ sudo /edx/bin/supervisorctl restart cms***

* 1. The following error occured during running the data report script:

**ERROR 1 (HY000) at line 1: Can't create/write to file '/tmp/#sql\_6cb\_0.MYI' (Errcode: 28 - No space left on device)**

This error occurs when you don't have enough space in the partition. Usually MYSQL uses ***/tmp*** on linux servers. This may happen with some queries because the lookup was either returning a lot of data, or possibly even just sifting through a lot of data creating big temp files.

**Solution**:

Check inode using following command

***df -i***

Remove tmp files or make available enough space on the machine. Restart VM to remove temporary files:

***sudo Init 6***

* 1. The following error occurred during running the ansible migration script on the Ginkgo machine:

**OperationalError: (1050, \"Table 'thumbnail\_kvstore' already exists\") ... " Applying thumbnail.0001\_initial..."**

**Solution:**

Drop a table thumbnail\_kvstore from the ecommerce MySQL database and re-run ansible command.

* 1. The following error occurred during running the ansible migration script on the Ironwood.master machine:

**ProgrammingError: (1146, \"Table 'edxapp.content\_type\_gating\_contenttypegatingconfig' doesn't exist\")... Applying ccx.0006\_set\_display\_name\_as\_override**

**Solution:**

Re-Run migration command.

# **Extra Information**

**Meld is a visual diff and merge tool targeted at developers. Meld helps you compare files, directories, and version controlled projects. It provides two- and three-way comparison of both files and directories.**

**We can use the “diff” terminal command to compare files also ...**

**To compare the schema of MySQL database using the software “Meld Diff Viewer”**

Using the software “Meld Diff Viewer”, do the Comparison between the **“<path\_of\_result>/ReportResult<*UpgradingType*>/After/<mysqlReports\_<ip\_timestamp>/mysql-structure-<database\_name>.sql”** files and the **“Script/DumpMySQLSchema<Intial\_letter\_of\_new\_ OpenedX\_release>/mysql-structure-<database\_name>.sql”** files.

**Note:**

1. The “Script/DumpMySQLSchemaH” directory has the expected ported schema of the Hawthorn.2 release.
2. The “Script/DumpMySQLSchemaI” directory has the expected ported schema of the Ironwood.master release.
3. Mostly the schema file has the difference, it is due to the timestamp and IP of the machine stored in both the files. If there is a difference other than timestamp and IP in the schema file then it needs to be identified.

# References

1. <https://edx.readthedocs.io/projects/edx-installing-configuring-and-running/en/latest/platform_releases/hawthorn.html#id15>
2. <https://edx.readthedocs.io/projects/edx-installing-configuring-and-running/en/latest/platform_releases/ironwood.html#id15>
3. https://edx.readthedocs.io/projects/edx-installing-configuring-and-running/en/open-release-ironwood.master/configuration/customize\_registration\_page.html
4. https://github.com/open-craft/custom-form-app
5. https://docs.google.com/document/d/1YUfqAOB61gnxX-XV2oy7ncAoMqdLC5KB/edit#heading=h.gjdgxs

# 

# Appendix

Below are the details of scripts used for creating data reports.

For the data up-gradation from Ginkgo to Hawthorn we use "UpgradingType" is "GH", set UpgradingType="GH" in the "parameter.config" file. For the data up-gradation from Hawthorn to Ironwood.master we use "UpgradingType" is "HI", set UpgradingType="HI" in the "parameter.config" file. "UpgradingType" is concatenated with the folder name, to specify the folder has been used by the "UpgradingType" upgrades.

Following are the list of folders used by the "data\_report.sh" script and the "findDiff.sh" script.

* MongoFiles:

This folder has two files mongoqueries\_cs\_comment.js and mongoqueries\_edxapp.js. These files are used by the mongodata\_report.sh script to run the data report for Mongo database.

* MySQLfilesGH and MySQLfilesHI:

These folders have four files(new\_tables, common\_tables, static\_tables, and deleted\_tables) for each user MySQL database. These files are used by the mysqldata\_report.sh script to run the data report for MySQL database.

* MySQLNewTable\_ExpectedResultGH and MySQLNewTable\_ExpectedResultHI:

These folders have the database files, these files have the count of rows which should have in the new tables of the database on the new release machine. It is used by the "findDiff.sh" script.

* MySQL\_ExpectedResultGH and MySQL\_ExpectedResultHI:

These folders have a file, this file has the count of rows of django\_migrations tables of all databases which should have in the django\_migrations tables of all databases on the new release machine. It is used by the "findDiff.sh" script

* MySQLdiff\_ExpectedResultGH and MySQLdiff\_ExpectedResultHI:

These folders have diff files, these files have the difference in between the old release and the new release if any, which should have on the new release machine. It is used by the "findDiff.sh" script

* MySQLdiff\_UpgradedTable\_ExpectedResultGH and MySQLdiff\_UpgradedTable\_ExpectedResultHI:

These folders have diff files, these files have the expected difference in the row count of the upgraded table of the MySQL databases from the old release to the new release if any, which should have on the new release machine. It is used by the "findDiff.sh" script

* MySQLdiff\_UGC\_ExpectedResultGH and MySQLdiff\_UGC\_ExpectedResultHI:

These folders have diff files, these files have the expected difference in the upgraded table of the MySQL database from the old release to the new release if any, which should have on the new release machine. It is used by the "findDiff.sh" script.

* Mongodiff\_ExpectedResultGH and Mongodiff\_ExpectedResultHI:

These folders have diff files, these files have the expected difference in the Mongo database reports from the old release to the new release, which should have on the new release machine. It is used by the "findDiff.sh" script.

**Details of the data studied during the migration process**

Following data is checked in the report.

* **In MySQL:**

No. of user tables present in each databases,

Check the character set of all user database schema

Auto\_Increment count of the tables of MySQL user database

Number of rows in the table of the MySQL user databases

* I**n Mongo:**

Check the collections, users and its indices, total records and indices in each collection.