

3 - Data Migration Step-by-step - Loading

FIRST STEPS

1. Login to Prod. Is there a weekly backup running, encoded as UTF-8, in Setup > Data Export
 - Nope

Select encoding UTF-8 and click "Export Now". This will take hours.

Turn that weekly stuff on.

Make sure the client KNOWS it's on.

Make sure they have a strategy for downloading the ZIP file that is generated by the extract weekly.
 - Yup
 - Is it UTF-8 and has run in the last 48 hours ?
 - Yup

Confer with the client to see if additional backup files are needed.

Otherwise, you're good.
 - Nope

If the export isn't UTF-8, it's worthless.

If it's more than 48h old, confer with the client to see if additional backup files are needed. In all cases, you should consider doing a new, manual export.

SERIOUSLY MAKE SURE YOU CHANGE THE ENCODING. Salesforce has some dumb rule of not defaulting to UTF-8. YOU NEED UTF-8. Accents and đấầ ãĩ s exist. Turns out people like accents and non-roman alphabets, who knew?
 - If Data Export is not an option because it has run too recently, or because the encoding was wrong, you can also do your export by using whatever tool you want to Query all the relevant tables. Remember to set UTF-8 as the encoding on both export and import.- 2. Check the org code and automation
 - Seriously, look over all triggers that can fire when you upload the data.

You don't want to be that consultant that sent a notification email to 50000 people. Just check the triggers, WFs, PBs, and see what they do.

If you can't read triggers, ask a dev to help you.

Yes, Check the Workflows and Process Builders too. They can send Emails as well.

- Check Process Builders again. Are there a lot that are firing on an object you are loading ? Make note of that for later, you may have to deactivate them.
3. Check data volume.
- Is there enough space in the org to accommodate the extra data ? (this should be pre-project checks, but check it again)
 - Are volumes to load enough to cause problems API-call wise ?
If so, you may need to consider using the BULK jobs instead of normal operations
 - In case data volumes are REALLY big, you will need to abide by LDV (large data volume) best practices, including not doing upserts, deferring sharing calculations, and grouping records by Parent record and owner before uploading. Full list of these is available in the pdf linked above and [here](#).

PREPARING THE JOBS

Before creating a job, ask yourself which job type is best.

Upsert is great but is very resource intensive, and is more prone to RECORD_LOCK than other operation types. It also takes longer to complete.

Maybe think about using the BULK Api.

In all cases, study what operation you do and make sure it is the right one.

Once that is done...

You are able to create insert, upsert, query and deletion jobs, and change select parts of it. That's because you are using a real data loading tool.

This is important because this means you can:

- Create a new Sandbox
- In whatever tool you're using, create the operations you will do, and name them so you know in which order you need to trigger them.
- Prepare each job, point them to a sandbox.
- Do a dummy load in sandbox. Make sure to set the start line to something near the end so you don't clog the sandbox up with all the data.
- Make sure everything looks fine.

If something fails, you correct the TRANSFORMATION, not the file, except in cases where it would be prohibitively long to do so. Meaning if you have to redo the load, you can run the same scripts you did before to have a nice CSV to upload.

GETTING READY TO DO THAT DATA OPERATION

This may sound stupid but warn your client, the PM, the end users that you're doing a data load. There's nothing worse than losing data or seeing stuff change without knowing why. Make sure key stakeholders are aware of the operation, the start time, and the estimated end time. Plus, you need them to check the data afterwards to ensure it's fine.

You've got backups of every single table in the Production org.

Even if you KNOW you do, you open the backups and check they are not corrupt or unreadable. Untested backups are no backups.

You know what all automations are going to do if you leave them on.

You talked with the client about possible impacts, and the client is ready to check the data after you finish your operations.

You set up, with the client, a timeframe in which to do the data operation.

If the data operation impacts tables that users work on normally, you freeze all those users during that timeframe.

Remember to deactivate any PB, WF, APEX that can impact the migration. You didn't study them just to forget them.

If this is an LDV job, take into account any considerations listed above.

DATA OPERATION

1. Go to your tool and edit the Sandbox jobs.
2. Edit the job Login to point to production
3. Save all the jobs.
4. You run, in order, the jobs you prepared.

When the number of failures is low enough, study the failure files, take any corrective action necessary, then use those files as a new source for a new data load operation.

Continue this loop until the number of rejects is tolerable.

This will ensure that if some reason you need to redo the entire operation, you can take the same steps in a much easier fashion.

Once you are done, take the failure files, study them, and prepare a recap email detailing failures and why they failed. It's their data, they have a right to know.

POST-MIGRATION

- Make sure everything looks fine, that you carried everything over.

- Warn their PM that the migration is done and request testing from their side.
- If you deactivated Workflows or PBs or something so the migration passes, **ACTIVATE THEM BACK AGAIN.**
- Unfreeze users if needed.

Go drink champagne.

IF SHIT DOESN'T LOOK RIGHT

You have a backup. Don't panic.

- Identify WTF is causing data to be wrong.
- Fix that.
- Get your backup, restore data to where it was before the fuckup. Ideally, only restore affected fields. If needed, restore everything.
- Redo the data load if needed.

Revision #2

Created 7 July 2019 18:22:11 by Windyo

Updated 10 March 2020 10:56:41 by thejamesjames