Odoo Redundancy Server Script Suite

Simon Rundstedt

October 28, 2021

Contents

1 Introduction

This is a small suite of scripts to keep an Odoo redundancy server as up to speed as possible as applicable at Vertel AB.

It will keep the *Odoo* parts of the installation close to the state on the designated production machine as possible with the notable exception of database replication which is beyond the scope of the script.

2 Setup and limits

2.1 Setup

The standby redundancy server is meant to be as close to the production server in terms of setup and configuration to the extent reasonably possible.

1. Basic setup of *production* and *standby* machine are as similar possible. In the context of this manual they are expected to be setup as clones or near clones before configuration as production and standby machine make them diverge.[1]
2. Odoo and Postgresql has been installed with Odootools and no changes has been made to the default Odoo filestore or Postgresql data directories.
3. Postgresql database replication is already setup and Postgresql on the standby machine is in standby mode.
4. Odoo as a service is disabled on the standby machine.

2.2 Limits

There are of course limits in how up to date the standby machine can be compared to the production server. In order to keep network and production server load down different parts of sync are made on different schedules. The different sync frequencies can cause some discrepancies between the state on the production and standy server at any one point in time. The general assumption is that updates to files are made less often than the DB and that some files such as installed packages rarely are updated.

- The database is continously replicated to the standby server. The delay is dependent on the chosen replication method and can range from seconds to minutes or more.
- Updates to the filestore are expected to done less frequently. The script suite install a cronjob which sync files hourly. Note: Some files, like generated invoice PDF's, are generated from database data, and other, like product images, need to be uploaded and have no direct representation in the DB.
- `apt` and `pip` packages and configuration files are expected to seldom be updated. Sync of them should be done during deployment of changes on the production server. The script suite install a cronjob which eventually will sync them as well.

[1] IE: Same disk size, same OS, same packages installed, same locations of important files.

3 RECOVERY

The worst has happened and the standby need to be put into production mode.

The process of *transitioning from standby to production mode is destructive* in the sense that the standby DB will go from RO mode to RW mode and immediately diverge from the production servers state.

Reverting the process, while not strictly impossible, should be viewed as difficult and the standby-to-production transition to be a one way street.

3.1 Checklist

Note that the scripts themselves provide no functionality to actually error check the standby installation and only aid to transition between standby and production mode. The checklist below can't cover all details or possible configurations and should be seen a starting point of a separate recovery process checklist.

1. Don't panic. Open a document for note taking during the recovery process.
2. Exhaust options to continue using the current production server.
3. Shut down the current production server if running. This is important since if Odoo is still running it can feasably continue with automatic tasks such as sending emails and depending on the DB replication setup it might continue archiving data. Make note of the shutdown time.[2]
4. (Recommended) Shut down the standby and make a backup image of it. This might help troubleshooting later if the standby machine doesn't work.
5. » **On the standby machine:** Run `sudo ot-disable-standby-mode` . The command disable Postgresql's standby mode, the scripte suits sync commands, and start Odoo.
6. Take note of the last file syncs. The file defined as `SUMMARYFILE` in the suites configuration file (`/etc/odoo/ot-redundancy-sync.conf`) contain timestamps of when the sync commands last where last run with exit code 0 (success). This will be useful in order to figure out how much data and work has been lost.
7. Login into Odoo and perform a smoke test looking for any obvious errors. Especially regarding:
   - the last updates to sale orders, inventory operations, invoices etc
   - any functionality using files created or updated since the last filestore sync (`ot-attachment-sync`) since it is the most likely to have some discrepancies between the DB and filestore content.

   Depending on the Odoo configuration it might difficult to login using the standby machines IP-address.
8. Repoint any reverse proxies, webservers, or DNS-records to the standby machine and verify the standby server is reachable as intendended.
9. Inform the relevant people about what have happened and the current state of the Odoo installation from the users perspective.

All going well the standby server has now been put into production mode and as few as possible will have noticed the hickups.

3.2 Standby server contain errors after recovery

This can happen for a multiple reasons. A non exhaustive list can be:

- the production server has been writing bad data to the standby server.
  If the standby server has been getting corrupted data from the production server there is little one could do other than recover from a backup. This should be done to the production server if possible.

[2] Though if it is running in a broken state, the broken state have probably already been synked to the standby machine

- the delay between db and file sync has caused issues:
  - DB and filestore out of sync.
    Regenerating asset bundles might help some errors. Expect up to an hours files to be lost. Note many files are generated from DB data and the data isn't lost per se.
  - Packages installed on the production server hasn't been installed on the standby.
    Packages are synced, but not nearly as often as the DB or filestore. This highlights the need of keeping the package sync as a part of the deployment process to the production server.
- additional services, other than Postgresql and Odoo, might need to be (re)started on the standby before they are fully operational.
  This document can't cover extra services or programs installed and need to be covered as a part of a separate checklist.

4 Installation and usage

4.1 Installation

4.1.1 Prerequisites

See section 2.1 for more information about the setup.

The script requires Odoo (>13.0) and Postgresql (>=12) installed on both production and standby machine. It is *strongly* recommended they are installed with `odootools`. If it is not make sure the Odoo filestore, and all Python modules are stored in the default directories.

The scripts has as of 2021-10-21 only been tested on Odoo 14.0 but should work with at least 13.0 too.

It is strongly recommended to setup Postgresql replication before installing the scripts since Odoo is worthless without a DB data.

4.1.2 Preparation

For the scripts to work `root` on the standby machine need to able to connect to the production server with a user having read access to Odoo files. By default this is the remote Odoo user `odoo` since `odootools` already enable a login shell for that user.

1. (On standby) Create SSH-keys as `root`: `sudo ssh-keygen -t ed25519`
2. (On standby) Print public key: `sudo cat /root/.ssh/id_ed25519.pub`
3. (On production) Copy keys to user `odoo`'s `authorized_keys` file. It might need to be created first:

   ```
   adam: sudo su odoo
   odoo: cd ~
   odoo: mkdir .ssh
   odoo: chmod 700 ~/.ssh  # Too easy to forget permissions.
   odoo: echo "KEY-FROM-STANDBY" >> .ssh/authorized_keys
   odoo: chmod 640 .ssh/authorized_keys
   ```

4. (On standby) Test SSH-key: `sudo ssh odoo@REMOTE-IP`

4.1.3 Installation

If installing from source, run `make` from the source folder and a deb-file should have been generated in the deb sub folder.

Copy the deb file to the standby server and install with:

```
adam: sudo dpkg -i ot_redundancy_XXX.deb # Replace XXX with version and architecture
adam: sudo ot-configure
```

Run the configuration script, `ot-configure`, and answer the prompts.