oTree Server Setup

Christian Koenig-Kersting

May 06, 2021

Contents

This documentation contains all the information neccessary to setup oTree on a fresh Debian 9 install. The rough process is as follows:

- install required software
- setup an otree user and set privileges
- install Python 3.6.x from sources
- initialize the PostgreSQL database
- setup Git with post-receive hooks
- setup nginx as a reverse proxy

The individual chapters cover the steps in more detail.

CHAPTER 1

Contents

1.1 Preparations

It is assumed that `root` can login to a standard shell. Furthermore, it is assumed that the server is reachable via SSH. If it is not, install openssh-server as `root`:

```
apt-get update
apt-get install openssh-server
```

The following steps will reference installation scripts. These need to be downloaded, extracted and made executable. As `root` run:

```
wget https://github.com/chkgk/otree_setup_scripts/archive/master.zip
unzip master.zip
cd otree_setup_scripts-master
chmod +x setup_otree_server.sh
```

You are now ready for the first step of the installation.

1.2 Quick Setup

Read *Preparations* **first!**

1.2.1 Basic setup as root

Run as `root`:

```
./setup_otree_server.sh
```

Enter passwords along the way. After the script finishes, you have two options: You can continue the installation with or without setting up SSL certificates.

1.2.2 Userspace setup without SSL

Run as `otree`:

```
./1_continue_setup.sh
```

This will automatically execute the following scripts one after the other:

```
./2_install_python.sh
./3_setup_database.sh
./4_setup_git.sh
./5_setup_nginx.sh
```

Finally it will remove `otree` from the sudo group.

1.2.3 Userspace setup with SSL

Run as `otree`:

```
./1_continue_setup_ssl.sh
```

This will automatically execute the following scripts one after the other:

```
./2_install_python.sh
./3_setup_database.sh
./4_setup_git.sh
./5_setup_nginx_ssl.sh
```

Finally it will remove `otree` from the sudo group.

1.3 Step 1: Software, Users and Privileges

This step requires the installation scripts to be downloaded and currently being logged in as `root`. Read *Preparations* if this is not the case.

1.3.1 Software

First we install the neccessary software using Debian's package manager:

```
apt-get update
apt-get install -y libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm zlib1g-
↪dev libssl-dev libncurses5-dev libncursesw5-dev xz-utils tk-dev postgresql␣
↪postgresql-contrib redis-server git supervisor nginx sudo
```

This includes:

- PostgreSQL Database
- Redis Datastore
- Git Version Control System
- Nginx HTTP and reverse proxy server
- Supervisor Process Control System

- Sudo

1.3.2 User and privileges

Typically, you do not want to work as `root`. Accordingly, we will setup a user `otree` and allow this user to run specific commands as `root` through `sudo`:

```
adduser otree
adduser otree sudo
```

This creates the user `otree` and adds him to the sudo group which allows him to run arbitrary commands as `root`. This is just for the setup. We will remove `otree` from the sudo group at the end of the installation. In the end, we want `otree` to only be able to run specific commands which are needed to control the supervisor daemon. We will now prepare the sudoers file accordingly:

```
nano /etc/sudoers.d/otree_supervisor
```

Copy in the following lines:

```
# allow otree user to start, stop and restart supervisor
otree ALL=NOPASSWD:/usr/sbin/service supervisor start
otree ALL=NOPASSWD:/usr/sbin/service supervisor stop
otree ALL=NOPASSWD:/usr/sbin/service supervisor restart
```

Save and exit nano with [ctrl] + [s] and [ctrl] + [x].

The three lines grant `otree` the right to start, stop, and restart supervisor. There will be more information on supervisor in a later installation step.

1.3.3 Prepare userspace installation

Now it is time to continue the installation from the userspace. Before we can do so, however, we need to copy over the installation scripts to `otree`'s home directory:

```
mv *.sh /home/otree/
chmod +x /home/otree/*.sh
chown otree:otree /home/otree/*.sh
```

Continue with step 2.

1.4 Step 2: Python 3.6

The Debian 9 stable branch includes Python up to version 3.5. While oTree runs on this version, there are some bugs related to the presentation of the monitoring table. Columns appear to be scrambled. To avoid this, we install Python 3.6 from sources.

1.4.1 Get Python 3.6

Log in as `otree` and navigate to your home directory. Download and unpack Python 3.6 (any point release is fine, change the following commands accordingly):

```
cd ~
wget https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tgz
tar  zxvf Python-3.6.5.tgz
rm Python-3.6.5.tgz
```

1.4.2 Configuration and building

Change working directory, then configure and make Python 3.6:

```
cd Python-3.6.5
./configure --with-ensurepip=install --enable-optimizations
make -j8
```

This might take a while. Time to get some coffee.

1.4.3 Installation

By default `python` points to `python2.7`. We will add python 3.6 as an alternative, but give the old association highest priority. We will have to use `python3.6` explicitly (rather than just `python` or `python3`), but at least we are not breaking anything.

```
sudo make altinstall
sudo update-alternatives --install /usr/bin/python python /usr/bin/python2.7 50
sudo update-alternatives --install /usr/bin/python python /usr/local/bin/python3.6 40
sudo update-alternatives --install /usr/bin/python python /usr/bin/python3.5 30
```

Leave and remove working directory:

```
cd ..
sudo rm -rf Python3.6.5
```

1.4.4 Virtual environment

OTree will run in its own virtual environment. This allows us to keep it separate from the system Python installation and avoids dependency clutter. We will create an initial virtual environment:

```
python3.6 -m venv venv_otree
source venv_otree/bin/activate
```

Finally, we make sure that the oTree virtual environment is automatically activated whenever `otree` logs in:

```
echo "source /home/otree/venv_otree/bin/activate" >>/home/otree/.otree_env
echo "source /home/otree/.otree_env" >> /home/otree/.bashrc
```

This complete the second step.

1.5 Step 3: Database and Supervisor

We now setup the PostgreSQL database and prepare a supervisor script which makes sure that the oTree web server is automatically started and restarted if if fails for any reason.

1.5.1 PostgreSQL

We start by setting up the database and a database user for oTree:

```
sudo -i -u postgres psql postgres -c "CREATE DATABASE django_db;"
sudo -i -u postgres psql postgres -c "CREATE USER otree_user WITH PASSWORD 'CHANGE_
↪THIS_PASSWORD_TO_YOUR_LIKING';"
sudo -i -u postgres psql postgres -c "GRANT ALL PRIVILEGES ON DATABASE django_db TO
↪$db_user;"
```

Note: You can use `tr -cd '[:alnum:]' < /dev/urandom | fold -w30 | head -n1` to quickly generate a random 30 character password from the command line.

1.5.2 Supervisor

We need to make sure that the oTree web server is started on system boot. It should also automatically restart, if it fails for any reason. To do so, we setup a supervisor script:

```
sudo nano /etc/supervisor/condf.d/otree.conf
```

Include the following lines:

```
[program:otree]
command=/home/otree/venv_otree/bin/otree runprodserver 8000
directory=/home/otree/oTree
stdout_logfile=/home/otree/otree-supervisor.log
stderr_logfile=/home/otree/otree-supervisor-errors.log
autostart=true
autorestart=true
environment=
    PATH="/home/otree/venv_otree/bin/:%(ENV_PATH)s",
    DATABASE_URL="postgres://otree_user:YOUR_DATABASE_PASSWORD@localhost/django_db",
    OTREE_ADMIN_PASSWORD="YOUR_WEBINTERFACE_ADMIN_PASSWORD",
    OTREE_PRODUCTION=1,
    OTREE_AUTH_LEVEL=STUDY,
```

Make sure to replace `YOUR_DATABASE_PASSWORD` with the one you set above. Also come up with a password for the oTree web interface and replace `YOUR_WEBINTERFACE_ADMIN_PASSWORD` accordingly. Finally, save the file and exit nano with [ctrl] + [s] and [ctrl] + [x].

1.5.3 Environment Variables

Finally, we want `otree` to be able to manually reset the database and invoke other oTree commands. We need to make sure that the credentials are available as environment variables whenever `otree` logs in.

Edit `/home/otree/.otree_env` with nano:

```
nano /home/otree/.otree_env
```

Add the following lines to the end:

```
export DATABASE_URL="postgres://otree_user:YOUR_DATABASE_PASSWORD@localhost/django_db"
export OTREE_ADMIN_PASSWORD="YOUR_WEBINTERFACE_ADMIN_PASSWORD"
export OTREE_PRODUCTION=1
export OTREE_AUTH_LEVEL=STUDY
```

Again, make sure to replace `YOUR_DATABASE_PASSWORD` with the one you set above. Also come up with a password for the oTree web interface and replace `YOUR_WEBINTERFACE_ADMIN_PASSWORD` accordingly. Finally, save the file and exit nano with [ctrl] + [s] and [ctrl] + [x].

Continue with step 4.

1.6 Step 4: Git Repository

We now set up the Git repository and a specific post-receive script that makes it easy for experimenters to deploy experiments on the server.

1.6.1 Directories

First we will create two directories: `oTree` will contain the live oTree project. `oTree.git` will be the repository that experimenters can push their experiments to. After creating the directories, we initialize an empty Git repository in the latter.

```
mkdir /home/otree/oTree
mkdir /home/otree/oTree.git
cd /home/otree/oTree.git
git init --bare
cd ~
```

1.6.2 Post-Receive Script

Experimenters can push their oTree experiments to the Git repository we have just created. Now we set up a script that is executed after every push to the repository. It will take care of the following steps:

- wipe the current virtual environment
- re-create the virtual environment
- install current versions of all required packages
- attempt to migrate the existing database if migrations are specified by the experimenter
- reset the database in all other cases
- restart the web server

Edit `oTree.git/hooks/post-receive` with nano:

```
nano /home/otree/oTree.git/hooks/post-receive
```

Add the following script:

```
#!/bin/bash
GIT_WORK_TREE="/home/otree/oTree"
VENV_DIR="/home/otree/venv_otree"
export GIT_WORK_TREE
git checkout -f

if [[ -d "$VENV_DIR" ]]; then
    echo "[log] - Cleaning virtualenv"
    rm -rf $VENV_DIR
```

(continues on next page)

(continued from previous page)

```
    echo "[log] - Finished creating virtualenv"
fi

# recreate venv
echo "[log] - create venv"
python3.6 -m venv $VENV_DIR

# activate
echo "[log] - activate venv"
echo $VENV_DIR
source $VENV_DIR/bin/activate
source /home/otree/.otree_env

# install requirements
echo "[log] - install requirements"
pip install -U pip
pip install -r $GIT_WORK_TREE/requirements.txt

echo "[log] - Staring DB migration"
cd $GIT_WORK_TREE
if [[ -d "$GIT_WORK_TREE/otree_core_migrations" ]]
    then
        echo "[log] - detected migrations in otree project dir"
        echo "[log] - attempting migrations"
        otree migrate
        echo "[log] - migrations done"
    else
        echo "[log] - no migrations defined"
        echo "[log] - resetting db"
        otree resetdb --noinput
        echo "[log] - database reset"
fi

cd ..
echo "[log] - Finished DB migration "

echo "[log] - restart services"
sudo /usr/sbin/service supervisor restart
```

Finally, we make the script executable:

```
chmod +x /home/otree/oTree.git/hooks/post-receive
```

Continue with step 5.

1.7 Step 5: Nginx

We now set up nginx to serve as a reverse proxy. This allows us to easily set up SSL certificates.

1.7.1 Setup Without SSL

We need to remove the default nginx config and add a new one for oTree. Let's remove the old one first:

```
sudo rm /etc/nginx/sites-enabled/default
```

Then we create a new one using nano:

```
sudo nano /etc/nginx/sites-enabled/otree
```

Copy in the following script:

```
map $http_upgrade $connection_upgrade {
    default   upgrade;
    ''        close;
}

server {
    listen 80;
    server_name _;

    location / {
        proxy_pass http://localhost:8000;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-Port $server_port;
        proxy_set_header Host $host;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header        Connection $connection_upgrade;
        proxy_set_header        X-Real-IP $remote_addr;
        proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header        X-Forwarded-Host $server_name;
    }
}
```

Save the file and exit nano with [ctrl] + [s] and [ctrl] + [x].

1.7.2 Setup with SSL

We need to remove the default nginx config and add a new one for oTree. Let's remove the old one first:

```
sudo rm /etc/nginx/sites-enabled/default
```

Then we create a new one using nano:

```
sudo nano /etc/nginx/sites-enabled/otree
```

Copy in the following script:

```
map $http_upgrade $connection_upgrade {
    default   upgrade;
    ''        close;
}

server {
    listen 80;
    server_name _;
    return 301 https://$server_name$request_uri;
}

server {
```

(continues on next page)

(continued from previous page)

```
    listen 443 ssl;
    server_name _;

    ssl_certificate PATH_TO_SSL_CERTIFICATE_PEM;
    ssl_certificate_key PATH_TO_SSL_KEY;

    location / {
        proxy_pass http://localhost:8000;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-Port $server_port;
        proxy_set_header Host $host;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header        Connection $connection_upgrade;
        proxy_set_header        X-Real-IP $remote_addr;
        proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header        X-Forwarded-Host $server_name;
    }
}
```

Make sure to replace `PATH_TO_SSL_CERTIFICATE_PEM` and `PATH_TO_SSL_KEY` with the appropriate paths to your SSL certificates. Finally, save the file and exit nano with [ctrl] + [s] and [ctrl] + [x].

This concludes Step 5.

1.8 Final Steps

The final step is to lock down the `otree` user account. Throughout the installation, it had full sudo privileges. To remove them invoke:

```
sudo deluser otree sudo
```

This removes `otree` from the sudo group. The group change only takes effect after logging out and in again.

1.9 Experimenter Usage

The server should now be set up completely. To get an experiment running, experimenters can add the server as a remote to their local git repository and then push their version to the remote repository.

1.9.1 Git Setup on experimenter's computer

If the experimenter is already using Git as their version control system, you can skip this step.

First, install Git from https://git-scm.com/downloads.

Then, navigate to your oTree project folder and initialize the git repository. You only need to do this once for each oTree project folder:

```
git init
```

They will also have to configure their username and e-mail address:

```
git config --global user.name "your name"
git config --global user.email "test@exam"
```

1.9.2 Setting up the remote repository

To add the oTree server as a remote invoke the following command from your oTree project / local git repository directory:

```
git remote add production ssh://otree@SERVER_URL/home/otree/oTree.git
```

Make sure to replace `SERVER_URL` with the URL the server is accessible from.

1.9.3 Pushing to production

When the experiment is ready to go live, experimenters can use the typical workflow of first staging and then commiting their changes before finally pushing them to production:

```
git add .
git commit -am "your commit message here"
git push production master
```

Users will have to know and enter the password for the `otree` user account. A few moments later oTree's web interface should be available from any browser pointed to the server's URL.

1.9.4 A note of caution

If the experimenter does not setup their oTree project to include database migrations the database will be reset after each push. **This means that data currently stored in the database will be erased.**

CHAPTER 2

Indices

- genindex
- modindex
- search