oTree Virtual Machine Manager Documentation

Release 0.2.2

Tobias Raabe

Apr 24, 2018

Contents

1 oTree Virtual Machine Manager **2**
- 1.1 Overview . . . 2
- 1.2 Features . . . 2
- 1.3 Tutorials . . . 3
- 1.4 Contributing . . . 3
- 1.5 Credits . . . 3

2 Installation **4**
- 2.1 Preconfigured Systems . . . 4
- 2.2 Requirements . . . 4
- 2.3 Stable release . . . 5
- 2.4 From sources . . . 5
- 2.5 Recommendations for Additional Software . . . 6

3 Usage **7**
- 3.1 General usage . . . 7
- 3.2 First run . . . 7
- 3.3 Command line options . . . 8

4 End User Desktop **10**
- 4.1 Starting a new Project as End User . . . 11
- 4.2 Running Experiments as End User . . . 11

5 Contributing **12**
- 5.1 Types of Contributions . . . 12
- 5.2 Get Started! . . . 13
- 5.3 Pull Request Guidelines . . . 14
- 5.4 Tips . . . 14

6 Structure **15**
- 6.1 `ovmm/` . . . 15
- 6.2 `docs/` . . . 16
- 6.3 `tests/` . . . 17

7 Credits **18**
- 7.1 Development Lead . . . 18
- 7.2 Contributors . . . 18

8 History **19**
8.1 Unreleased . . . 19
8.2 0.2.2 (2017-07-21) [YANKED] . . . 20
8.3 0.2.1 (2017-07-16) . . . 20
8.4 0.2.0 (2017-07-16) . . . 20
8.5 0.1.1 (2017-03-20) . . . 20
8.6 0.1.0 (2017-03-20) . . . 20

9 Indices and tables **21**

Contents:

CHAPTER 1

oTree Virtual Machine Manager

oTree Virtual Machine Manager helps to manage user accounts.

- Free software: MIT license
- Documentation: https://otree-virtual-machine-manager.readthedocs.io.

1.1 Overview

oTree Virtual Machine Manager is a complement to the oTree Virtual Machine Image provided by Felix Albrecht and Holger Gerhardt.

Since doing research is time-consuming enough, this tool ensures that administrators of an oTree server do not waste their time on creating fully equipped user accounts and similar tedious tasks. Everything breaks down to a single commandline interface.

Managing an oTree server with multiple experiments running parallel has never been easier.

1.2 Features

Create users Creates a fully equipped experimenter account (clear project structure, virtual environment, graphical or point-and-click solutions to many oTree-related commands, samba access).

Back up user Creates a database and/or home folder backup for users upon account closure so that nothing gets lost.

Remove user Removes otree-server user accounts.

Reroutes user Depending on the network it might be necessary to assign the main port to a different user.

Behind the scenes Handles port configuration for multiple parallel user accounts on a single virtual host.

1.3 Tutorials

You can find a series of tutorial videos on Youtube.

For administrators

- Installation of ovmm
- Adding a user account with ovmm
- Restoring a database

For end users

- Using ovmm as an end user
- Resetting the database and backups as end user

1.4 Contributing

Contributions are welcome and they are greatly appreciated! Every little bit helps, and credit will always be given.

1.5 Credits

Thanks to all contributors who spent their time on making this project more helpful for everyone.

- Jonas Radbruch (Contributions to the documentation)

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

As **oTree Virtual Machine Manager** provides useful commands to administrators of the **oTree Virtual Machine Image**, it requires superuser rights to perform its actions. Therefore, install `ovmm` under the provided administrator account.

2.1 Preconfigured Systems

We created a Virtualbox image for oTree users based on **ovmm** and on "Ubuntu 16.04.2 Server 64bit" and comes with three desktops *Gnome*, *LXDE*, and *fluxbox* to allow it to work on systems with varying level RAM.

The images all contain a user account called `otreeadmin` that comes preconfigured with the set of end user commands provided by **ovmm**. The images further each contain an account `ovmmadmin` from where **ovmm** can be used to create more accounts.

You find the images here: oTree Virtual Machine Images

We also spent some time on trying to figure out what tools could be useful to the end users. Therefore all images with the following tools preinstalled.

Chromium Webbrowser

Giggle GUI for Git,

Atom Great free IDE with many plugins,

pgAdmin 3 GUI for postreSQL database systems

2.2 Requirements

The following components need to be present on your system to use `ovmm`:

ftp required for ftp access to the account

git version control

nginx webserver

postgresql database server + development packages

pip package manager for Python

python3-venv virtual environment for Python 3

redis-server remote dictionary server

samba Windows share access

screen detachable consoles

mailutils sending mails if otree stops unexpectedly

ssh remote shell access

ufw firewall

xterm Linux terminal emulator

zenity GTK 3.0 dialog handler

In most Debian based Linux environments you can install these packages from the repositories like so:

```
$ sudo apt install ftp nginx samba screen mailutils ssh ufw postgresql git␣
↪ \
                   postgresql-server-dev-all python3-pip zenity          ␣
↪ \
                   python3-venv redis-server xterm
```

Please see the documentation of your Linux distribution for help.

2.3 Stable release

To install oTree Virtual Machine Manager, run this command in your terminal:

```
$ sudo pip3 install ovmm
```

This is the preferred method to install oTree Virtual Machine Manager, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.4 From sources

The sources for oTree Virtual Machine Manager can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/tobiasraabe/otree_virtual_machine_manager
```

Or download the tarball:

```
$ curl  -OL https://github.com/tobiasraabe/otree_virtual_machine_manager/
↪tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ sudo python3 setup.py install
```

2.5 Recommendations for Additional Software

To enhance the working experience for the oTree end user we recommend to also install the following software packages.

chromium Chromium browser is the Open Source version of Chrome browser. As it is the most frequently used browser on the web it is the recommended testing environment for your oTree apps.

conky Desktop system monitor. ovmm statics provides a preconfigured conky configuration file which is unzipped into the user's $HOME directory and provides helpful system information while running oTree.

Atom Open Source IDE with many features.

pgAdmin III Graphical user interface for postgreSQL database servers. Helps with understanding oTree and data recovery if something goes wrong.

In Ubuntu and official derivatives you can install chromium, conky, and pgAdmin via the package manager, like so:

```
$ sudo apt-get install conky-all chromium-browser pgadmin3
```

It is recommended to obtain LightTable from the official website as the community packages are deprecated.

You can find the website here: http://lighttable.com

CHAPTER 3

Usage

3.1 General usage

`ovmm` is a complement to the **oTree Virtual Machine Image** provided by Felix Albrecht and Holger Gerhardt. Therefore, until now it is only tested on this specific image. But, we do provide information (*Requirements*) on how to create a similar image which can use `ovmm`.

Always run `ovmm` with `sudo` or some commands might not work.

If you need more information on the commands, run `ovmm --help`.

Run `ovmm --version` to display the version number.

3.2 First run

1. When using `ovmm` for the first time, run `sudo ovmm initialise` to configure your system.

   Note: The process leads you through the installation of required dependecies and creates necessary content files for `ovmm`.

   (a) Enter administrator's password.

   (b) Enter account information of the PostgreSQL superuser.

   By default, the superuser for a PostgreSQL database is called `postgres`. You also have to set a password for this account. This password can be new and will be set to the user by request.

2. After running the command, there is a folder called `/home/<user>/ovmm_sources/` which contains a bash script `ovmm_conf`. Perform the following checks:

   (a) Check whether the login information for the PostgreSQL database is correctly set.

(b) Define the ranges of port numbers, e.g. `OVMM_DAPHNE_RANGE`. Port ranges are stored in lists. Insert distinct values or use list comprehension for bigger ranges.

3. Next, go back to `ovmm_sources` in your home folder. You should see a file called `nginx_template`. Read the commentary to adjust the file to your network configuration.

   This file will be reused every time a new user is created. The process will fill out the user specific values and place it under `/etc/nginx/sites-avalaible`. Then, a symlink will link this file to `/etc/nginx/sites-enabled`, so that Nginx takes care of it.

4. Forelast, try out one of the innocent lookup commands like `sudo ovmm count_user` or `list_user`. If they succeed, you have correctly set up `ovmm`.

5. Last, set up a test account and verify whether everything is working including *oTree* by running one of the example experiments.

3.3 Command line options

Command line options are entered used with `sudo ovmm <command>`

3.3.1 Commands

Since command expansion is implemented, the commands after the forward slash are also available.

initialise / i Adjust your system to the needs of `ovmm`. It installs Ubuntu dependencies as well as configuration files for the Administrator.

> **Warning:**
>
> 1. This command should be executed on first run in advance of any other command.
> 2. An internet connection is needed

add_user / a Add a new user to the configuration. Several prompts will ask you about user specific information. Please, fill out the forms in accordance with the provided examples.

delete_user / d Delete a user from the configuration. Her account including her home folder is completely removed. Make sure you have a backup of all files in advance.

backup_user / b Create a backup for a given user and save it in the administrator's home directory under `/home/<admin>/ovmm_sources/user_backups`. Choose from one of three options, `all`, `db`, `home`, whether you want to make a backup of the database or the home folder. `all` is a shortcut to run a backup of both, database and home folder.

count_user / c Return the number of existing accounts and the number of possible, additional accounts.

list_user / l Give a list of user names of all currently installed users.

route_port / r Change nginx default config to route port 80 (HTTP) and port 443 (HTTPS) to specific user account in addition to user specific port. Intended for running experiments where router settings prevent the access to non-standard web-ports like 780x.

upgrade_statics / u Upgrades the statics in each existing user account with the versions in `ovmm/static/exp_env.7z`. This is done by a simple extract and overwrite.

--version Returns the version number of `ovmm`

–help / -h Shows the help page. A more detailed version is available for each command with `sudo ovmm add_user --help / -h`

CHAPTER 4

End User Desktop

Apart from the standard otree commands we created a set of commands for the end user to improve usability. The commands are available via command line and via desktop launcher providing an easy to use GUI that guides the user through the several processes. The user has the following commands available:

- `otree_startproject` creates an otree project in the folder *Projects* in the user's home directory. The created folder is a basic otree project folder WITHOUT the example apps. It further creates either a symbolic link `venv` to the default virtual python environment of the user in the project folder or creates a specific virtual environment within the project folder called `venv`. The latter allows for specialized oTree versions for each project. Lastly it copies the *_rooms* directory from the folder Templates into the project folder. *_rooms* contains a preconfigured econ101.txt for 24 clients.
- `otree_link_experiment` is used to create a symbolic link to the nginx root location. The nginx webserver looks in a predefined location for a webserver root. The command handles the linking for you and creates a symlink to `$HOME/.oTree` which is where nginx looks for your project.
- `otree_restart` helps you with your starting requirements. It offer to run `otree collectstatic` for you as well as an `otree resetdb`. Before the database reset is executed pg_dump makes a backup of the current state of the database and stores a 7zipped version of it in $HOME/DB_BackUps. Finally it runs `otree runprodserver --port=YOUR_DAPHNE_PORT` or `otree runserver YOUR_DAPHNE_PORT` for you to ensure that you use the correct port. `otree_restart` also comes with a number of arguments that help you get started more quickly.

  -c run `otree collectstatic`

  -h help

  -l run local (programming) server (runserver)

  -m Send emails when otree stops running. (Your email is configured in *otree_environ_config*.)

  -p skip queries and run `otree runprodserver --port=YOUR_PORT` directly

-r run `otree resetdb`

-s start otree in a detached screen (virtual console)

The commands are stored in $HOME/.local/bin for each user separately so that users can make adjustments for themselves if needed.

All three commands offer a console and a GUI based on GTK zenity dialogs. For calling the GUI three `*.desktop` are created in $HOME/Desktop.

When you call up a command the command will guide you through the process.

The below graphic provides an overview of the file structure created in the oTree user's home directory and the command relations.

The user should not run `otree runserver` (which is just for local testing) or `otree runprodserver` (because of the proxy settings).

4.1 Starting a new Project as End User

You need to complete 2 steps in order start a new project and connect it to the server root directory.

Step 1)

1. Execute `otree_startproject`
2. Provide a project name.
3. Choose whether to create a **specialized virtual environment** for this project.

Step 2)

1. Execute `otree_link_experiment`
2. Select the path to the project you want to run.

4.2 Running Experiments as End User

As End User you have to follow the following steps in order to run experiments.

1. Execute `otree_restart`
2. Choose whether to do a collect static or not.
3. Choose whether to do a database reset.
4. Watch the output if everything starts fine. ;-)

If you have trouble starting the experiment or you want to activate demo mode change the necessary parameters in the `otree_environ_config`. `otree_restart` sources `otree_environ_config` each time it is called. You don't need to manually source it to activate the new settings.

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/tobiasraabe/otree_virtual_machine_manager/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "FEATURE_REQUEST", "FEATURE_IDEA" and "HELP_WANTED" is open to whoever wants to implement it.

Furthermore, have a look at *Structure* to see how your implementation fits into the current design of ovmm.

5.1.4 Write Documentation

oTree Virtual Machine Manager could always use more documentation, whether as part of the official oTree Virtual Machine Manager docs, in docstrings, or even on the web in blog posts, articles, and such.

If you want to participate by writing docstrings, please, follow the guidelines for NumPy Style Python Docstrings. For a complete example, follow this link (Example NumPy Docstring).

A more general tutorial for reStructuredText can be found here.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/tobiasraabe/otree_virtual_machine_manager/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *ovmm* for local development.

1. Fork the *ovmm* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/ovmm.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv ovmm
$ cd ovmm/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass all the tests defined with tox:

```
$ tox
```

   You can test separately by typing:

```
$ tox -e ${TOXENV}
```

   The following commands test python code and documentation:

```
$ tox -e flake8
$ tox -e doc8
```

If you want to lint code and documentation, type:

```
$ tox -e linters
```

To get flake8, tox, doc8, restructuredtext_lint just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/tobiasraabe/otree_virtual_machine_manager/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_otree_virtual_machine_manager
```

CHAPTER 6

Structure

The tree structure of ovmm currently looks like this:

```
otree_virtual_machine_manager
├──docs
│
├──ovmm
│    │   cli.py
│    │   __init__.py
│    │
│    ├──commands
│    │
│    ├──config
│    │
│    ├──handlers
│    │
│    ├──prompts
│    │
│    ├──static
│
└──tests
```

In this view, we have only included the most relevant parts and cover each part separately in the following sections.

6.1 ovmm/

```
otree_virtual_machine_manager
├──ovmm
│    │   cli.py
│    │   __init__.py
│    │
│    ├──commands
│    │   │   add_user.py
```

```
│   │   │   backup_user.py
│   │   │   count_user.py
│   │   │   delete_user.py
│   │   │   initialise.py
│   │   │   list_user.py
│   │   │   route_port.py
│   │   │   upgrade_statics.py
│   │   │   __init__.py
│   │   ├──config
│   │   │   environment.py
│   │   │   static.py
│   │   │   __init__.py
│   │   ├──handlers
│   │   │   nginx.py
│   │   │   postgres.py
│   │   │   samba.py
│   │   │   __init__.py
│   │   ├──prompts
│   │   │   defaults.py
│   │   │   parsers.py
│   │   │   validators.py
│   │   │   __init__.py
│   │   ├──static
│   │       .profile
│   │       exp_env.7z
│   │       nginx_default_template
│   │       nginx_template
│   │       otree_environ_config
│   │       ovmm_conf.yml
```

`ovmm/` contains the actual python package. On top of the folder, there exists `cli.py` which is contains the command line interface. It collects all existing commands from the folder `commands/` which contains one file per command. Usually, commands require actions to be performed on the PostgreSQL, Nginx or Samba. To provide an easy interface which automatically performs checks for the integrity of the targeted service, handlers for these services are included in `handlers/`. Additionally, commands rely on prompts and validators to receive valid user input. Pre-existing functions can be found in `prompts/` as well as defaults serving as examples. `config/` contains to files for configuration values which can be static or dependent on the specific environment. At last, `static/` contains everything which needs to be copied to the executing account or to users accounts created by `ovmm`.

6.2 docs/

```
otree_virtual_machine_manager
├──docs
│   │   authors.rst
│   │   conf.py
│   │   contributing.rst
│   │   enduser.rst
│   │   history.rst
│   │   index.rst
```

```
│   │   installation.rst
│   │   make.bat
│   │   Makefile
│   │   readme.rst
│   │   structure.rst
│   │   usage.rst
│   │
│   ├───static
│   │       otree_img_sys.svg
```

The `docs/` folder contains the documentation of the project as well as this document. New documents must be included in `docs/index.rst` to be compiled as part of the documentation. Static images, etc. have to be placed in `docs/static/`.

6.3 tests/

```
otree_virtual_machine_manager
└───tests
    │   conftest.py
    │   test_cli.py
    │   test_handlers_postgres.py
    │   test_handlers_samba.py
    │   test_prompts_defaults.py
    │   test_prompts_parsers.py
    │   test_prompts_validators.py
```

The `tests/` folder contains the testing environment for `ovmm` written with pytest. `conftest.py` holds variables or function which are relevant for all tests. The other files have the following naming pattern.

1. The prefix `test_` is required to be automatically recognized as a file containing tests by pytest.
2. The middle part of the name references the subfolder of `ovmm/` to which the test file belongs. We can therefore easily see what the target of the test is.
3. The suffix references the specific file inside the subfolder of `ovmm`

CHAPTER 7

Credits

7.1 Development Lead

- Felix Albrecht <f.albrecht@uni-bonn.com> (enduser commands/statics, documentation, director)
- Tobias Raabe <tobiasraabe@uni-bonn.de> (ovmm core)

7.2 Contributors

- Jonas Radbruch (documentation)

CHAPTER 8

History

All notable changes to this project will be documented in this file.

The format is based on Keep A Changelog and this project adheres to Semantic Versioning.

8.1 Unreleased

- **Overhauled CLI (#102)**
  - new command: upgrade_statics
  - version flag with `ovmm --version`
  - Check for sudo when calling `ovmm` with error message
  - Added switch to assign user to sudo group during creation
  - Separated settings into a static and environment dependent part
  - Open ftp and samba ports with ufw during initialisation
  - `ovmm init` detects whether nginx templates exists and asks before overwriting
  - New format for HISTORY.rst
  - More information in python header file
  - Added credits
- Enhanced Github PR template (#105)
- Small adjustments to documentation tests and build (https://github.com/tobiasraabe/otree_virtual_machine_manager/commit/c5512515b615c8c83654f8d318ce6887d74bdd82)
- Dropped pyup.io and codacy (#120)
- Refinements to testing battery (#123)

8.2 0.2.2 (2017-07-21) [YANKED]

- Hotfix for v0.2.1 due to error on initialise.

8.3 0.2.1 (2017-07-16)

- Re-released v0.2.0.

8.4 0.2.0 (2017-07-16)

8.4.1 Added

- `route_port` command to change standard ports to a different user. (Thanks, Felix)

8.5 0.1.1 (2017-03-20)

- Alpha release. Re-released on PyPI :).

8.6 0.1.0 (2017-03-20)

- Alpha release. Released on PyPI.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search