

Odoo tutorials

Odoo made easy

RECENT POSTS

[Introducing Oocademy: better Odoo e-learning for you](#)

[Installing Odoo 12 \(enterprise\) on Ubuntu](#)

[Creating building blocks in Odoo](#)

[Configuring submodules on Odoo.sh](#)

[Installing Odoo frontend and backend on different servers](#)

SUBSCRIBE TO BLOG VIA EMAIL

Enter your email address to subscribe to this blog and receive notifications of new posts by email.

CATEGORIES

[Automated actions](#)

[Building block](#)

[Buttons](#)

[Chat](#)

[Controllers](#)

[Data](#)

[erppeek](#)

[Filters](#)

[Github](#)

Introducing Oocademy: better Odoo e-learning for you

OCTOBER 1, 2019 YENTHE666 6 COMMENTS

As you've probably already noticed it has been quite a while that I've published a new tutorial here. Ever since I've started writing tutorials here I had one goal. Share as much knowledge as possible in an easy to access way that everybody understands. No matter their knowledge. After a few years on this blog I noticed that this blog wasn't cutting it anymore though. I started missing some features that became harder to ignore as time went by.

Why I started building an e-learning platform



Because of this I've decided to build an e-learning platform on my own: **Oocademy**. Oocademy is the tool that I've always wished that existed to help others and enables the missing features that I missed:

- Manage versioned tutorials/documentation as code usually changes on another Odoo version.
- Quick linking & smarter logic to find related content.
- Advanced search options for users so they can find content fast(er).
- The ability to write documentation and apps along with the tutorials.
- The ability to create exams and certificates so that developers can test their Odoo knowledge
- Keeping wishlists of all the content and keeping track of the progress of reading content.
- Many other, smaller, things.

Five years after I started writing tutorials on this blog I still have the same opinion about the Odoo world. There is far too little good content for developers to learn from. The official Odoo documentation only highlights some major functionalities, it is pretty outdated and contains very little examples. In the meanwhile some content has showed up but the speed in which content is being added is still too slow, not to mention the quality of it.

This is where Oocademy comes in. It is built to help you find the content that you need. A few months ago I've started writing tutorials, documentation and apps specifically for Odoo 13 and here I am today to announce the launch. At this point I've published 43 tutorials, 19 documentation pages, 37 apps and two exams and most of the content is specifically written for V13!

Accessing Oocademy

Installation

Mail

many2many

Odoo 10

Odoo 11

Odoo 12

Odoo 13

Odoo 8

Odoo 9

Odoo.sh

PowerBI

Python

Reports

Security

Selection

Tests

Tour

Translations

Uncategorized

Webpages

Website

ARCHIVES

October 2019

November 2018

June 2018

April 2018

January 2018

November 2017

October 2017

August 2017

July 2017

April 2017

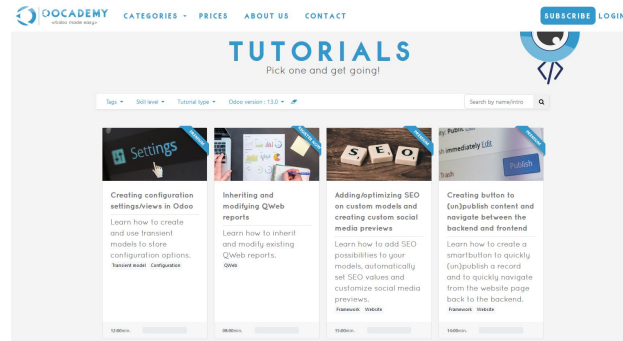
January 2017

December 2016

November 2016

October 2016

So, how does it work? On Oocademy every tutorial, documentation page, app or exam is either free, behind a login or premium content:



If the content is free you can read all of it without needing any membership or login. When it is freemium you can access it for free and you just need to register (which takes just 10 seconds and can be done from the Oocademy homepage). If it is premium content you'll need to buy membership. To celebrate the launch of Odoo 13 and all the content for V13 on Oocademy I'm giving huge discounts though. The first 100 new users can [buy premium membership starting from €10 a month](#).

Now I bet you're wondering why isn't all of the content free? Of course I wanted to keep it all free but creating a whole e-learning platform and writing all this content takes a lot of time. Just to give you an idea: I've invested +- 600 hours in this project (75 working days!) to get to this point. The fact is that any bigger platform needs to have a paid tier to make it grow and to be able to maintain it well. If I just get 200 premium users a month I can put at least two extra days of work into Oocademy, every single week.

If you don't want – or can't – pay you can register for free too though! [Just fill in your email, name and password](#) and you can start reading content within seconds.

What is the roadmap?

And this brings us to the roadmap ofcourse. I still have a whole lot planned for Oocademy in the future. The most important part is more content. The goal is to create tutorials, apps and documentation for every functionality in Odoo. Besides of the content I want to add quite some extra features:

- Code generators to create dynamic pieces of code for the main Odoo functionalities. Examples are QWeb reports, domain builders, menuitems, ...
- Webinars / tutorials written by specialists in that specific domain. For example a tutorial about workers written by Olivier Odony.
- Training courses: a chapter-like tutorial which explains a bigger functionality part by part. For example how to create webpages with controllers and custom search filters.
- A secret project that will stay a secret for now. 😊
- Do you have any ideas or suggestions? Please post them in the comments! 😊

Conclusion

I really hope to see you on Oocademy and most importantly that it helps you grow! All the tutorials that are written here will stay available and for free, forever. Gradually I'll also update all these tutorials on Oocademy so that they're up to date and working for a specific version.

Oh and as Odoo V13 is being released.. [check out our newest tutorial about how to install Odoo 13!](#)

Let us work on an Odoo community that has better content for both new and experienced developers. Our community deserves this.

Regards,
Yenthe Van Ginneken
CEO & Founder of Oocademy

August 2016
April 2016
February 2016
January 2016
November 2015
October 2015
September 2015
August 2015
July 2015
June 2015
May 2015
April 2015
March 2015
February 2015
December 2014



Share this:

Share

Like this:

Loading...

E-LEARNING OOCADAMY

Odoo

localhost:8069/web/database/selector

odoo

Database Name

Email

Password

Phone number

Language English

Country

Demo data

Create database

Installing Odoo 12 (enterprise) on Ubuntu

NOVEMBER 3, 2018 YENTHE666 19 COMMENTS

In this tutorial I will learn you how to install Odoo 12 community or enterprise on Ubuntu 18.04. The script that you will use is based on the code from André Schenkels but has been updated, upgraded and improved. Do notice that if you want to install the enterprise version that you will need to be an official partner or that you need to have bought the enterprise subscription from Odoo. Otherwise you will have no access to the Github repository for the enterprise code!

1. Downloading the script

The first step is to download my script from Github and to add the code in a new .sh file on your Ubuntu machine, wherever you'd like this. For example right under /home. Open up an Ubuntu terminal and cd to the directory where you'd like to keep the script and then create the file:

```
1 sudo wget https://raw.githubusercontent.com/Yenthe666/Install
```

If you're curious about how the whole code looks and works you can find it on my Github account.

Now open up the file and edit the parameters to your liking:

```
1 sudo nano odoo_install.sh
```

There are some things you can configure/change to your likings at the top of the script. You can choose if you wish to install Wkhtmltopdf or not, which version you'd like, where the location is and most importantly what the master admin password is.

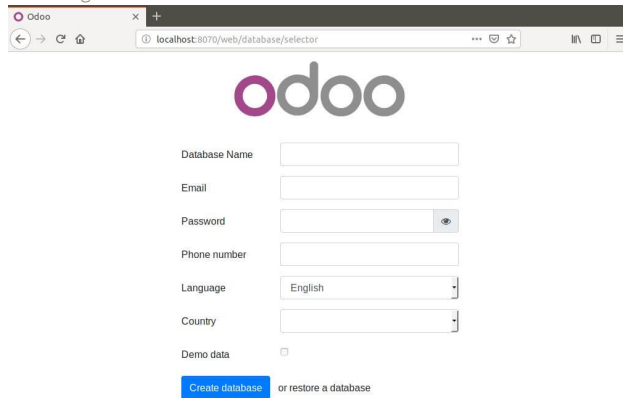
Tip: always modify this for every Odoo you install!

If you want the enterprise version of V12 you should change the line IS_ENTERPRISE to true:

```
1 IS_ENTERPRISE="True"
```

```
administrator@ubuntu:~$ sudo service odoo-server restart -c /etc/odoo-server.conf
Restarting odoo-server: odoo-server.
administrator@ubuntu:~$
```

The -c will change the configuration and memorize what you've changed under /etc/your-config-file.conf. Because my port was set to 8070 this is telling the Odoo that it should run on port 8070. When you would now open up your browser and navigate to <http://localhost:8070/> you will see it is running there:



Has this tutorial helped you, do you have any feedback or questions? Post away!



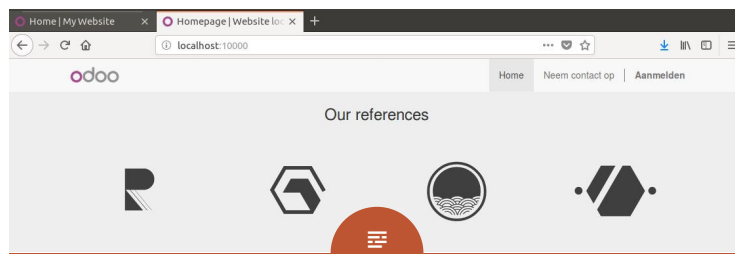
Share this:



Like this:

Loading...

• ENTERPRISE • INSTALLATION • ODOO 12



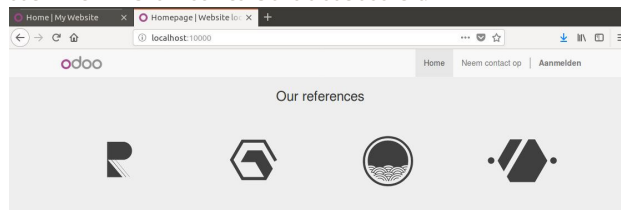
Creating building blocks in Odoo

JUNE 2, 2018 YENTHE666 LEAVE A COMMENT

In this tutorial I will learn you how to create new building blocks and how to add the blocks to the menu so that you can quickly drag and drop the blocks in the webpage.

In this tutorial I will create a new building block named "References with

title" which will show four icons and a title above it:



1. Adding the dependency

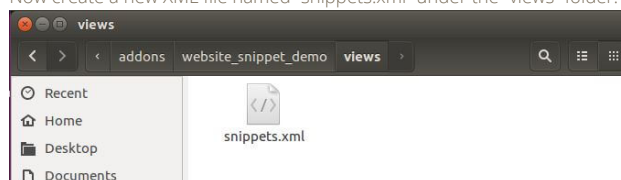
Before you can start creating a new building block you will need to create a new module and configure the manifest.py correctly. Open up your manifest.py and add the website as a dependency:

```
1 # The website module has to be installed and is needed
2 depends: ['website'],
```

Without this dependency you cannot create and add new building blocks to the Odoo website.

2. Creating a new XML file

Now create a new XML file named "snippets.xml" under the "views" folder:



In this file we will add all the code to create the building block (snippet) and to make it visible in the editor.

2.1 CREATING THE BUILDING BLOCKS

First we will need to create the building block itself. Let us create a building block that has a title (h3) saying "Our references" with four logo's under the title. Have a look at this code:

```
1 <!-- This record will create the building block
2 <template id="s_title_references" name="Referenc
3 <section class="s_references bg-gray-lighter"
4 <div class="container">
5 <div class="row">
6 <h3>
7 <center>Our references</cent
8 </h3>
9 </div>
10 <div class="row">
11 <div class="col-md-3 col-sm-3 co
12 
14 </div>
15 <div class="col-md-3 col-sm-3 co
16 
18 </div>
19 <div class="col-md-3 col-sm-3 co
20 
22 </div>
23 <div class="col-md-3 col-sm-3 co
24 
26 </div>
27 <!-- You can use your own image
28 <div class="col-md-2 col-sm-3 co
29 
31 </div>
32 </div>
33 </div>
34 </section>
35 </template>
36
```

So, what does this tell us? We first create a new XML record. After doing this we add all our code within a section block and inside this we create a container div. Within this section and container you can basically code anything you like, this is the framework for any building block. Generally when you create a building block you try to use as much **bootstrap** classes as possible. In the above example I simply made two rows. One row for the title and one row for the images. Those images are all the same width thanks to the default bootstrap classes col-md, col-sm and col-xs-6.

You're already well over halfway to your own building block! If you would install this module right now all the code would be there that is needed for

a building block, but we still have to show it in the editor so that we can use it.

2.2 ADDING THE BUILDING BLOCK TO THE EDITOR

Let us continue and add a building block preview to the editor so you can quickly find it from the editor.

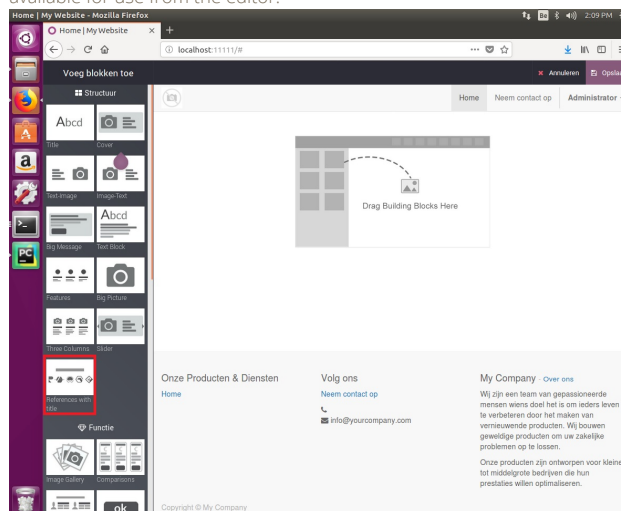
You can do this by inheriting the default “website.snippets” record and doing an xpath in the “snippet_structure” id, which holds the main structure of the editor. Have a look at this code:

```
1 <!-- This record will create a preview of the bui
2 <template id="add_title_references_to_bar" inheri
3 <xpath expr="//div[@id='snippet_structure']">
4 <div class="o_panel_body">
5 <t t-snippet="website.snippet_demo.s
6 t-thumbnail="/website/static/src/i
7 </div>
8 </xpath>
9 </template>
```

Let me explain the code a bit further. We first inherit the default ‘website.snippets’ record, which holds the link to all available snippets. By doing an xpath on ‘snippet_structure’ we’re telling Odoo to add our building block preview within the editor. In this xpath element we add a div and we use the t-snippet element made by Odoo. By doing so Odoo knows we want to add a snippet preview to the editor. Finally save this file and add it in the manifest.py file so that it is loaded:

```
1 # always loaded
2 'data': [
3 # Load the snippets (building block code) whe
4 'views/snippets.xml',
5 ]
```

When you now install the module you will see your new building block is available for use from the editor:



That is all. You’ve just made your own building block, congratulations!

3. Conclusion

Thanks to the Odoo framework it is very easy to create and use new building blocks. The functionality is so flexible and easy to use that you can create a building block for about anything. Creating and reusing building blocks in Odoo is one of the biggest strengths of the Odoo website editor.

Do you want to try the demo code and see the source code of this tutorial? You can view it on my [Github account](#).

Has this tutorial helped you, do you have any feedback or questions? Post away!



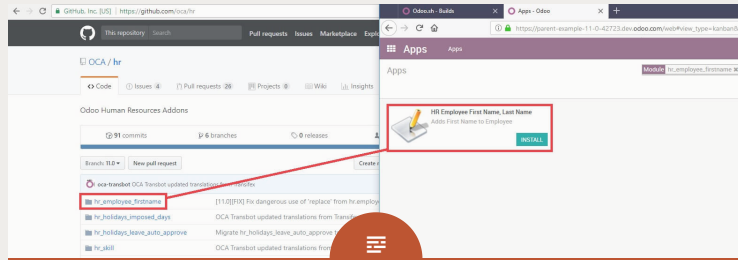
Share this:



Like this:

Loading...

BUILDING BLOCKS ODOO 10 ODOO 11 WEBSITE



Configuring submodules on Odoo.sh

APRIL 14, 2018 YENTHE666 4 COMMENTS

In this tutorial I will learn you how to create and use submodules in Odoo.sh. In this example I will create a public submodule and a private submodule and I will link them to the Odoo.sh project.

1. Introduction to submodules

Tip: If you haven't made an Odoo.sh project yet you can follow my tutorial [here](#).

So, what exactly is a submodule? A submodule is a link from one Github repository to another repository. You can see it as a virtual pointer to a specific commit in time of the remote repository.

For Odoo.sh there are two major differences. You have public submodules and private submodules. Public repositories are those that are publicly available (for example <https://github.com/odoo/odoo>). Private repositories are the repositories that are not publicly available. Usually you have private repositories when you work for a company and when you manage customer code.

There is a big difference in using public or private submodules on Odoo.sh though. When you have a public repository you can easily add the submodule and it will work. For a private repository you will need to generate a deploy key on Odoo.sh and then add it on the remote Github repository.

1.1 Why use submodules?

Now we know what a submodule is but the question is why would you use a submodule? Submodules are very handy to use if you want to include third party apps in your Odoo.sh tests. The only other way to get this code available in your Odoo.sh builds is to add all this third party code in your own repository. But what if the third party developer has made a lot of changes, fixes or improvements? If you don't use submodules you will constantly need to download the remote apps, add them to your own repository and keep track of all those changes. This is the true power of submodules. Now let us first link a public repository to the Odoo.sh project in chapter 2. In chapter 3 I will learn you how to add a private repository as a submodule.

2. Public repositories

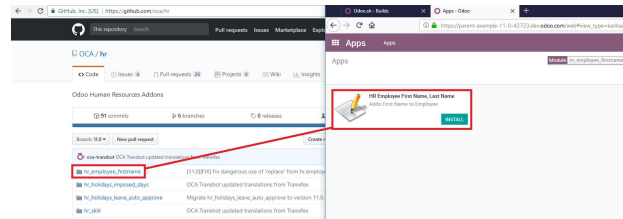
Setting up public repositories is really very easy to do. You will need to create a submodule and commit it to Github. Go to your Github repository, switch to the correct branch and add the submodule from the command line:

```

1 git submodule -b 11.0 add git@github.com:oca/hr.git
2 git add -A && git commit -m "[ADD] submodule: add OCA/hr as s
3 git push
4

```

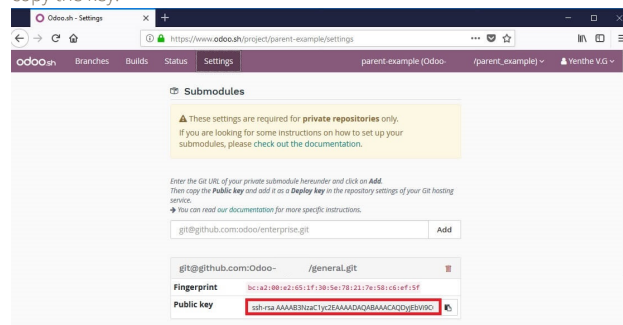
This new commit will trigger a new Odoo.sh build. After this build is ready you will see that the modules from the OCA/hr repository are available on this instance:



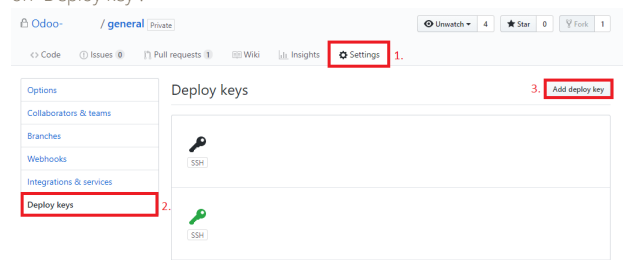
Great job! This is all you need for a public repository. If you also have a private repository and want to add it then continue to the next chapter.

3. Private repositories

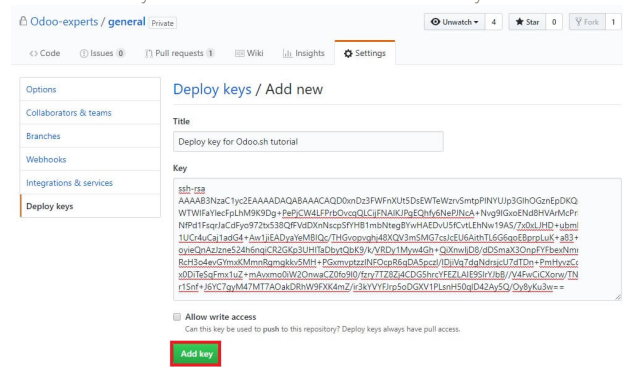
Alright now let us configure the use of a private submodule! Go back to your Odoo.sh project to the settings tab and add the link to your private repository (git@github.com:youruser/repository.git). In your repository you could now make a commit to add the repository as a submodule. For a private repository however you have some extra work. You have to copy the generated key from the Odoo.sh project and add it on Github. First copy the key:



Now go to the repository where you've linked to (so the submodule), go to the settings tab, open "Deploy keys" and add your own key here by clicking on "Deploy key":



Now add the key in the next screen and click on "Add key":



After doing this Odoo.sh can find the private repository and can access all the data it needs. The final thing you need to do is to add the submodule from the command line, commit it and push it to Github:

```

1 git submodule -b 11.0 add git@github.com:account/private_subr
2 git add -A && git commit -m "[ADD] submodule: add private rep
3 git push
4

```




If you want the community version you can just continue and keep the IS_ENTERPRISE key on "False" (which is the case by default):

```
1 IS_ENTERPRISE="False"
```

2. Making the Odoo installation file executable

The next step is to make this file executable. After you've made it executable you can execute it and everything will be installed automatically.

do this with the following command:

```
1 sudo chmod +x odoo_install.sh
```

3. Running the script

Now that the code is in your file and the file is executable you simply have to execute it with the following command:

```
1 ./odoo_install.sh
```

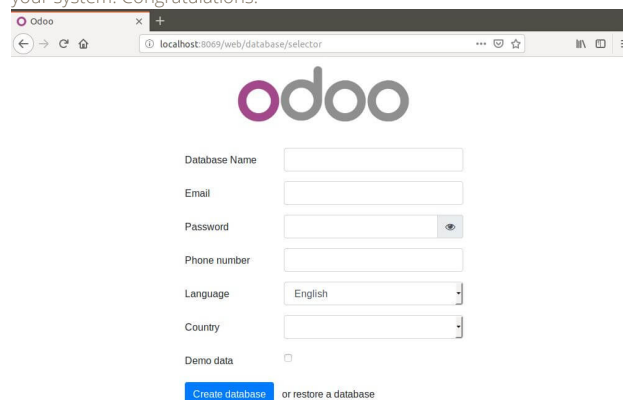
You will see that the script automatically starts updates, downloads required packages, creates the user, downloads the code from Github, ... Eventually, if you've chosen to install the enterprise version, you will need to give in your Github credentials to download the enterprise code (since this is a private repository). Fill in your details and let the script continue:

```
---- Create custom module directory ----
---- Adding Enterprise code under /odoo/enterprise/addons ----
Cloning into '/odoo/enterprise/addons'...
Username for 'https://github.com': Yenthe666
Password for 'https://Yenthe666@github.com':
```

Give the script a few minutes to configure and install everything and eventually you will see something like this:

```
administrator@ubuntu: ~
File Edit View Search Terminal Help
-----
Done! The Odoo server is up and running. Specifications:
Port: 8069
User service: odoo12e
User PostgreSQL: odoo12e
Code location: odoo12e
Addons folder: odoo12e/odoo12e-server/addons/
Start Odoo service: sudo service odoo12e-server start
Stop Odoo service: sudo service odoo12e-server stop
Restart Odoo service: sudo service odoo12e-server restart
-----
administrator@ubuntu:~$
```

You now have a fully functional Odoo V12 community or enterprise on your system! Congratulations.



4. Extra information about Odoo 12 Enterprise

Since Odoo Enterprise uses code from both <http://github.com/odoo/odoo> and <http://github.com/odoo/enterprise> we will separate the code with this script. This will make future upgrades easier and the code is nicely separated. This means that the default V12 code will be under /odoo/odoo-server/ and all the enterprise code will be under /odoo/enterprise/.

In the script you saw there was an option to change the Odoo port (OE_PORT). When you'd change this port number to 8070 in the install script it would be applied to /etc/your-config-file.conf and this would give you the ability to change the default port.

To apply these changes you should do the following:

This new commit will trigger a new Odoo.sh build. After this build is ready you will see that the modules from your other private repository are available on this instance.

4. Conclusion

Using submodules in combination with Odoo.sh is very powerfull and handy to use. Setting it up in the beginning might take some time and you can make some mistakes but in the long run it will save you a load of time and redundant code. As a result it'll improve your testing a lot as all your custom code will be automatically tested.

If you can use submodules and want to test your deployments on Odoo.sh then submodules are the way to go.

Has this tutorial helped you, do you have any feedback or questions? Post away!



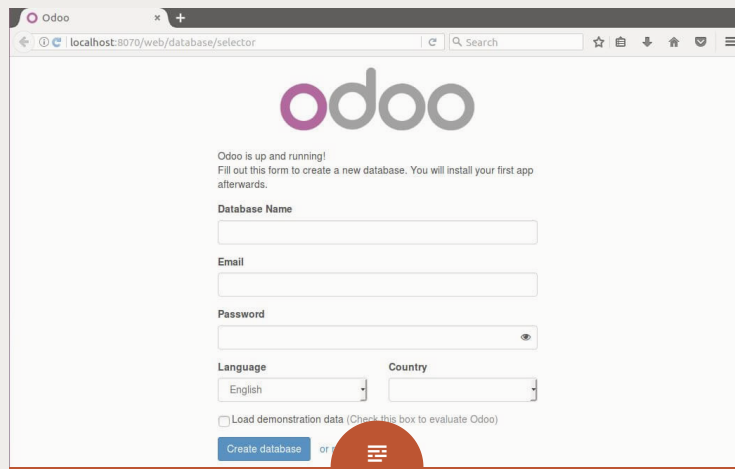
Share this:



Like this:

Loading...

• [ODOO.SH](#) • [SUBMODULE](#) • [SUBMODULES](#)



Installing Odoo frontend and backend on different servers

JANUARY 20, 2018 YENTHE666 9 COMMENTS

In this tutorial I will teach you how to separate your Odoo frontend and backend. I will install the Odoo codebase/frontend on one server and the backend (your PostgreSQL server and database) on another server. You will learn how to configure both Odoo and PostgreSQL to achieve this.

1. Introduction

In order to achieve and demonstrate this setup to you I will be working with two virtual machines.
I will refer to the Odoo codebase/frontend server as the 'frontend server'. The PostgreSQL server and database will be named 'backend server' throughout this tutorial.
My frontend server has the IP 192.168.118.167 and my backend server has the IP 192.168.118.168.

Tip: Make sure that both your servers have a fixed IP, otherwise your setup will break due to IP changes!

2. Configuring the backend server

Let us start with the configuration of the backend server. This server will contain the PostgreSQL server and the database(s).

2.1 INSTALLING POSTGRESQL

Open up a terminal on your backend server and install PostgreSQL if you haven't yet:

```
1 sudo apt-get install postgresql -y
```

2.2 CREATING THE POSTGRESQL USER

Now create a new PostgreSQL user. Make sure that the PostgreSQL username matches with the username of the user running Odoo on your frontend server. In my example I will create a new PostgreSQL user with the name "odoo11":

```
1 sudo su - postgres -c "createuser -s odoo11 -P"
```

After you execute this command the system will ask you for a password for this user. Fill in a password and confirm this password again.

Tip: Don't forget to remember this password, you'll need it later on.

2.3 CONFIGURING PG_HBA.CONF

After installing the PostgreSQL server and creating the user we now need to configure the remote connections. As our backend server will be used for the database connections our frontend server needs to be able to access it. Open up your frontend server and get the IP of the server. In my example my frontend server has the IP 192.168.118.167. Now open up your pg_hba.conf file on your backend server:

```
1 sudo nano /etc/postgresql/9.5/main/pg_hba.conf
```

Scroll in this file and search for the following line of code:

```
1 # IPv4 local connections:
2 host all all 127.0.0.1/32
```

Add a new line after the existing one which contains your frontend server its IP address. As you don't want to use the exact IP address you should use the /24 subdomain. In my example this results to 192.168.118.0/24 instead of 192.168.118.167. Your configuration file should now look like this:

```
1 # IPv4 local connections:
2 host all all 127.0.0.1/32
3 host all all 192.168.118.0/24
4
```

Finally, save your file and close it. Your PostgreSQL now knows that you want to allow connections from the backend server (IP 192.168.118.167).

2.4 CONFIGURING POSTGRESQL.CONF

Your PostgreSQL still needs to know the listen address of your frontend server too though. To achieve this we have to edit the postgresql.conf file:

```
1 sudo nano /etc/postgresql/9.5/main/postgresql.conf
```

Find the 'listen_addresses' line, which looks like this by default:

```
1 #listen_addresses = 'localhost'
```

Now add in the IP of your frontend Odoo server (in my example 192.168.118.192):

```
1 listen_addresses = 'localhost,192.168.118.168'
```

Tip: Don't forget to remove the # in this line because otherwise this line will be skipped!

Save this file and close it.

You've now applied all the changes that you need to do on the backend server. Next reload the PostgreSQL service in order to apply all the changes:

```
1 sudo service postgresql reload
```

3. Configuring the frontend server

Your backend server is done now. Switch to the frontend server and open up your Odoo configuration file.

Tip: If you haven't installed an Odoo yet you can follow my tutorial [here](#). Typically your Odoo configuration file is under /etc/ and is named your-odoo.conf:

```
1 sudo nano /etc/odoo11-server.conf
```

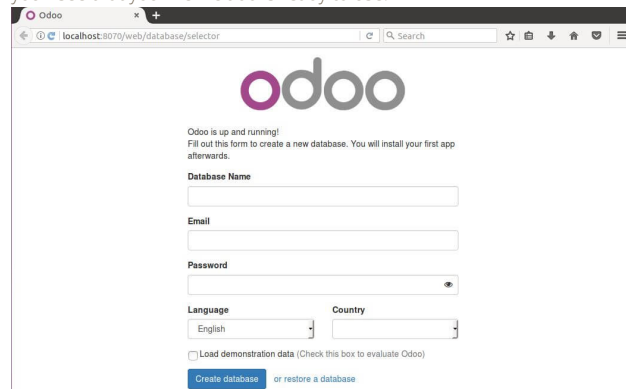
Add or change the following parameters in the configuration file:

```
1 [options]
2 admin_passwd = mysupersecretpassword
3 db_host = 192.168.118.168 # This is the IP of the backend server
4 db_port = 5432
5 db_user = odoo11 # The PostgreSQL user you've configured in c
6 db_password = yourpassword # The PostgreSQL user his password
```

Finally, restart your Odoo service to reload your Odoo configuration:

```
1 sudo service odoo11-server restart
```

When you now browse to your Odoo instance (on your frontend server) you'll see that your new Odoo is ready to use.



That is all! You've now setup your own Odoo instance where the frontend and backend are split between two servers.

4. Tips

While this is everything that you need in order to split up the backend and frontend of Odoo there are still some things to consider.

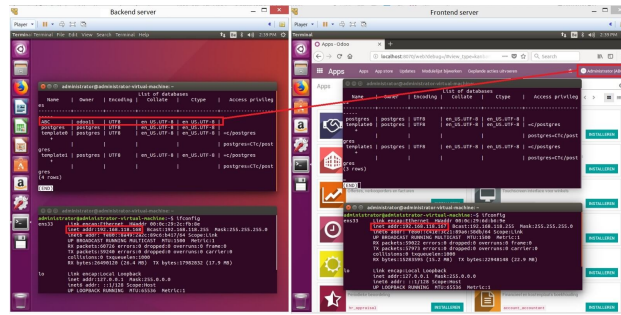
Because of security reasons you'll most likely want to encrypt all the data and place everything on SSL. This also includes the calls from the frontend server to the backend server.

After you've created a new database you should check if the database is in fact created on the backend server instead of on the frontend server. You can do this executing the following commands (one by one) on your backend server:

```
1 sudo su postgres
2 psql
3 \l
```

After running these commands you can see an overview of your databases and to what user the database belongs. In my example you can see a new database named 'ABC' which is owned by my Odoo/PostgreSQL user

'odoo11':



5. Conclusion

Setting up an Odoo instance where the frontend and the backend are split is actually quite easy. Due to the built-in parameters from Odoo it is very easy to configure the frontend side. When you split the frontend and backend up and apply extra security – such as HTTPS connections and VLAN's – you'll have a safer and more controllable Odoo instance. If you'd like to learn more about the deployment of servers and the default Odoo parameters you can have a look at the [official documentation](#).

Has this tutorial helped you, do you have any feedback or questions? Post away!



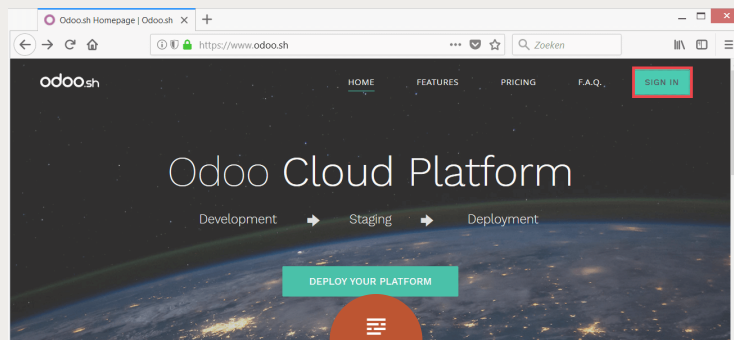
Share this:



Like this:

Loading...

• ODOD 10 • ODOD 11 • ODOD 9 • POSTGRESQL • SECURITY



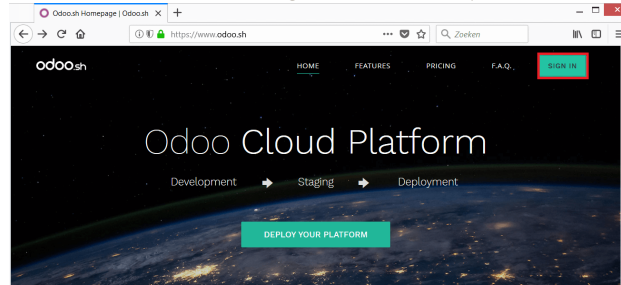
Configuring and using Odoo.sh

JANUARY 19, 2018 YENTHE666 3 COMMENTS

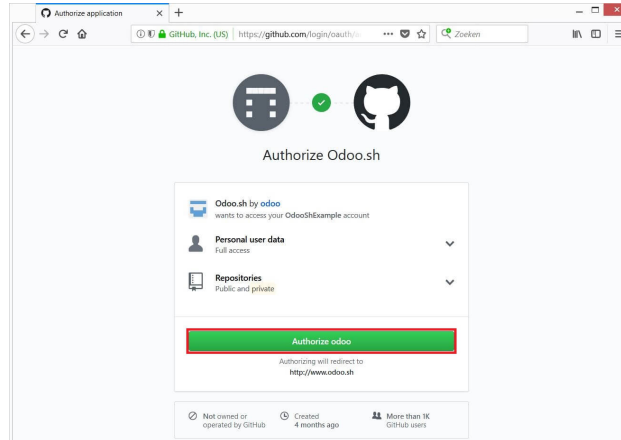
In this tutorial I will learn you how to setup an [Odoo.sh](#) account, how to configure Odoo.sh and how to tests your code automatically with Odoo.sh. In this example I will create a new account, create a new repository and add code to the Github repository in order to explain how Odoo.sh works.

1. Creating an Odoo.sh account

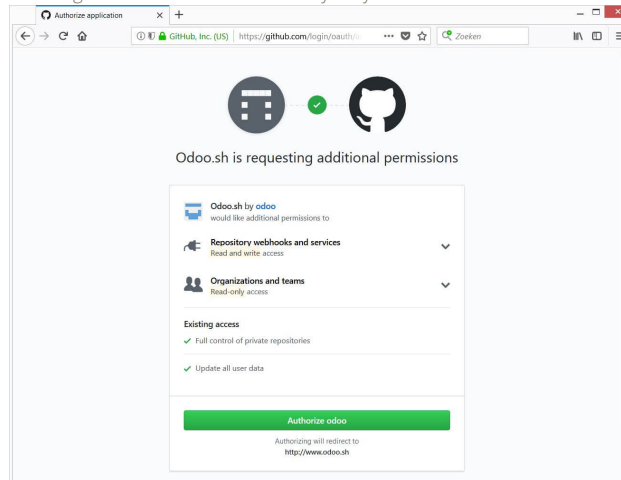
Go to [odoo.sh](https://www.odoo.sh) and click on the “Sign in” button at the top:



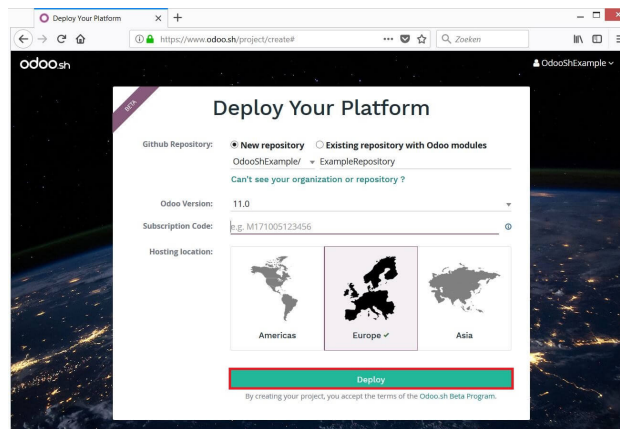
When you click on the “Sign in button” you’ll get an authorize screen from Github. If you’re not yet logged in on Github it will ask you to login, if you’re already logged in on Github you’ll get the authorize screen. Click on “Authorize odoo”:



After you click on “Authorize odoo” Github will ask for additional permissions. Click on “Authorize Odoo” again to give the additional permissions. Because of these additional permissions Odoo.sh can follow all changes and handle it automatically for you.



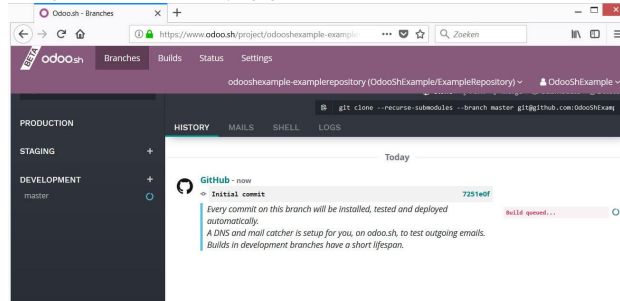
Now Odoo will ask you to deploy your platform. Choose an existing repository if you want to use it or create a new one. In this tutorial I will create a new repository to show everything in detail. Choose a repository name. Next choose the Odoo version you want to test against and finally provide your enterprise (or partner) license. The hosting location is up to you. Finally click on “Deploy”:



That's it! You've just registered your own Odoosh account and connected Github to Odoosh!

2. The Odoosh main screen

After you've clicked on "Deploy" you'll now see the main screen of Odoosh.

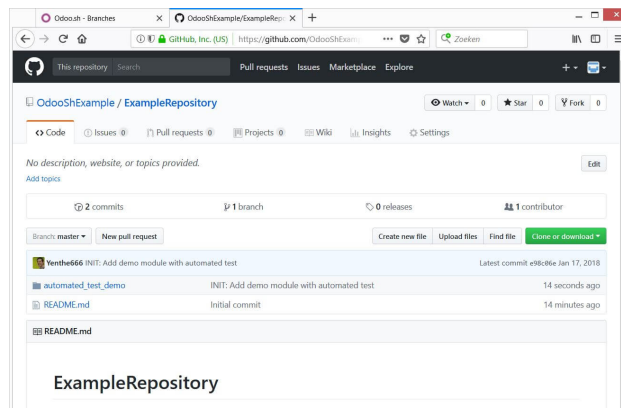


In the left menu you will see the title "DEVELOPMENT". Under this section you will see every branch you've made on this Github repository. If I would create a second branch named "11.0" I would see "11.0" and "master" in the left menu. At the top you'll see a main menubar with the options "Branches", "Builds", "Status" and "Settings".

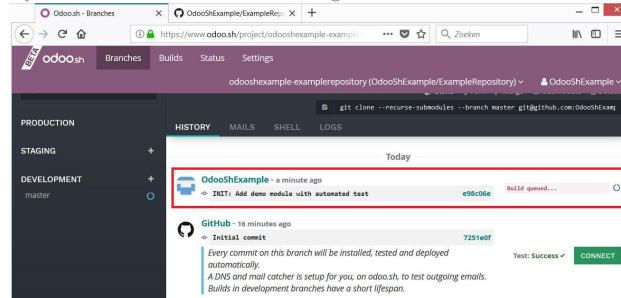
- The **Branches** tab opens the main page, from where you can see everything related to your branches. This includes the mails, shell access to the test instance and access to the logs.
- The **Builds** tab opens a page where you can see all your test instances. This is almost identical to the Odoo runbot (at runbot.odoo.com).
- The **Status** tab opens a page where you can see all the statistics of the Odoosh platform. It shows you the uptime and the status of all the servers.
- The **Settings** tab opens a page where you can configure advanced settings. You can add collaborators, set a project name (and URL), add submodules and much more.

3. Github

Now open up your Github and go to your (just created) repository. Add a commit to the repository so that it contains some (new) code. In my example I will push a module to Github that contains an automated test so I can show you how tests work and what happens. If you'd like to do the same you can take my example module from [Github](#). The moment that you make a commit to the Github repository Odoosh will detect this and it will start up a test environment because of the new commit. My Github after making a commit:

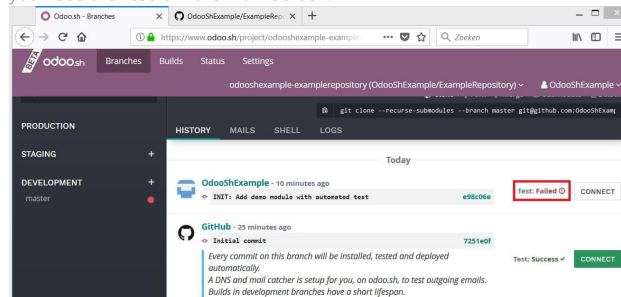


My Odoo.sh a few seconds after making this commit:

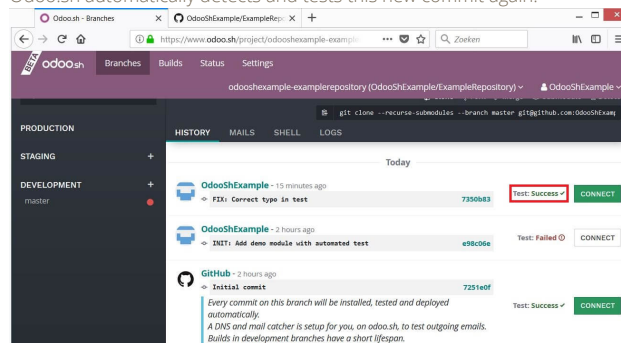


4. Checking the commit

Now switch back to Odoo.sh. After a few minutes your new commit will produce a new test instance that is ready and built. In my example I've deliberately added an error in my last commit so as a result you can see the test fail. When the test instance is done, typically after a few minutes, you'll see the result in the main screen:

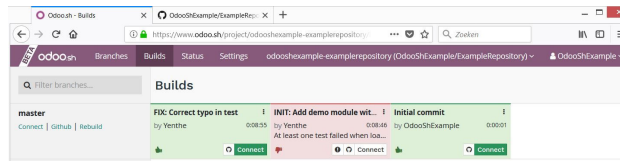


So now my test has failed it means I must have done something wrong. I've figured out what was wrong and I correct my test. After correcting this test I make a new commit to Github. After a few seconds you'll see that Odoo.sh automatically detects and tests this new commit again:



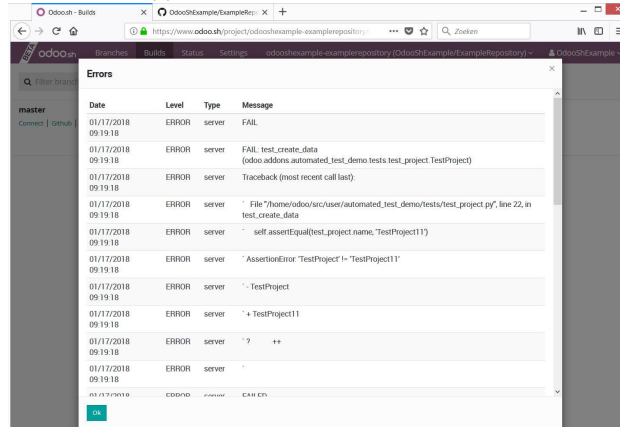
5. Odoo.sh builds and test instances

Finally go to the builds page by clicking on "Builds" in the top menu bar. After you click on "Builds" you will see an overview of all your test instances and if they succeeded or not:



From this page you can directly connect to a test instance in order to test functionalities or ‘play’ with Odoo.

When a test has failed (such as “INIT: Add demo module wit...” in my screenshot) you can click on the exclamation icon, next to the connect button, to see why your instance has failed:



Tip: If you want to see the full log of the instance you can do this by clicking on the “...” icon at the top of a build and selecting “Logs”. From here you can download and view all logs. Alternatively you can also do this from the “Branches” page by clicking on the “Logs” menu.

6. Conclusion

As Odoo.sh is very big and has a lot of options I've decided to split the content in two tutorials. Because of this the first blog post, this one, is only about basic operations. You can find a second tutorial about how to configure public and private submodules [here](#).

Odoo.sh is a very powerful platform to use. Odoo has invested a lot of time into Odoo.sh and due to this it is an advanced system that allows you to quickly (and easily!) use test instances with Odoo. Thanks to Odoo.sh you no longer need your own runbot instance and complex hardware setup as you can get it all out of the box.

Has this tutorial helped you, do you have any feedback or questions? Post away!



Share this:



Like this:

Loading...



