

Installing Odoo 15 on Ubuntu 20.04



By Greg Moss

Founder of [OdoolInnerCircle.com](https://odoolinnercircle.com)

©2021 by First Class Ventures LLC, Greg Moss,
OdooClass.com, OdooInnerCircle.com

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by an information storage and retrieval system - except by a reviewer who may quote brief passages in a review to be printed in a magazine or newspaper - without permission in writing from the publisher.

Contents

Introduction	4
How this Book is Organized	5
Installation Requirements and Pre-Requisites.....	5
Using the built-in text editor Nano	5
Pasting Code into Nano.....	5
Saving your changes.....	5
Quitting out of Nano	5
Turning off your Browser Caching	6
Turning off the Browser Caching in Chrome.....	6
Developer Installations vs Staging / Production Instances	6
Developer Installations	6
Staging Installations	6
Production Installations	7
Updating Ubuntu and Installing the First Set of Required Packages	8
Creating a System User	8
Installing and Configuring PostgreSQL.....	8
Installing Wkhtmltopdf	9
Downloading, Installing, and Configuring Odoo 15 in a Virtual Environment.....	9
Install the Required Odoo Python dependencies	10
Initial test of our Odoo installation.....	10
Viewing your Odoo Instance in the Browser	11
Configuring Odoo to run custom add- on modules	11
Creating a configuration file for our Odoo installation.....	12
Creating a Systemd unit file to auto-start Odoo when the server reboots	13
Testing the Installation	15
Configuring Nginx to access Odoo through a secured SSL domain name	15
Configure your DNS entries to point your domain name to the Ubuntu server IP address	15
Installing Nginx.....	15
Getting and Installing your FREE SSL Certificate	16
Install the Certbot package	16
Obtaining the SSL certificate.....	16
Creating code snippets to hold our SSL configuration details	17

Acquiring the SSL Certificate	18
Auto-renew your SSL Certificate	19
Modifying your Nginx configuration to access your Odoo installation with the SSL certificate	20
Change the Odoo Configuration File to use proxy mode	22
Improving security by changing the binding interface	23
Enabling your Odoo installation for multiple worker processes	23
Worker number calculation	24
RAM memory size calculation.....	24
Calculate the RAM memory consumption based on the number of workers	24
FINAL TESTING	25
Why you want to use the most recent versions of Odoo	25
Conclusion.....	26
Odoo Resources	26

Introduction

In this eBook, you will get step by step instructions on how to install Odoo 15 on an Ubuntu server. This server could be hosted in the cloud or locally. It is highly recommended that you use Ubuntu for your Odoo installations. While you can install Odoo on Windows or Mac, there are several limitations. For one, you won't be able to use multiple threads or workers. This severely limits you.

Also, this installation will use a virtual environment for the Odoo installation. This helps avoid conflicts with other services and other Odoo versions you may wish to install on the server.

If you get stuck or have problems, make sure you have installed all the pre-requisites. Also, make sure you are in the right user account for each given step in the process and that you have the virtual environment activated when it is needed. This becomes clear when you follow my steps in order.

Finally, if you get stuck, don't panic. I've installed Odoo literally hundreds, if not thousands of times. The first dozen times you do it, it may be a little confusing. Just like any major project, implementing Odoo requires a special set of skills.

Expecting to do it all perfectly without prior experience would be like trying to re-build an engine for a car when you have no experience in automotive repair. Or trying to build a house from the ground up if you have never built anything more complicated than a table.

ERP systems are complex. So, if you have not implemented at least 4 or 5 business systems previously, I highly encourage you to reach out to an Odoo consultant or partner to assist you. In the end you will save yourself many hours of frustration. It is easy for people who are not experienced with system integrations to underestimate the skillset that is required.

You are always welcome to contact me at gregmoss@odooClass.com if you have questions or would like assistance in configuring your Odoo Installation.

All the best,

A handwritten signature in dark ink that reads "Greg Moss". The script is casual and slightly slanted to the right.

Founder, OdooClass.com

How this Book is Organized

All commands you must type into the terminal will be displayed in the typeface below:

```
sudo command options - all commands that you will need to enter will appear in this typeface
```

When commands wrap to multiple lines (as shown above), do not enter a line break unless one is indicated by a wide vertical space between the lines, like so:

```
sudo command a
```

```
sudo command b
```

During the installation, you will sometimes have to copy and paste commands into text files. This content is identified with a light orange background to make it easier to know exactly what you need to paste. For example:

```
Example Line 1
```

```
Example Line 2
```

```
Example Line 3
```

Installation Requirements and Pre-Requisites

It is expected that you have an Ubuntu server set up and ready for the Odoo installation.

I recommend using a cloud solution from AWS, Digital Ocean, or a similar provider. You can also, if you choose, host Ubuntu as a local virtual machine in VMWare. This will possibly save you a little bit of money, and the setup and installation are going to be the same.

Using the built-in text editor Nano

This installation guide uses Nano which is built into Ubuntu. If you are unfamiliar with it, here are few simple tips to help you get started.

Pasting Code into Nano

To paste text into Nano, first you will want to use the arrows on your keyboard to position the cursor where you want to paste the text. You can't use your mouse to position the cursor in Nano. Once the cursor is where you want it, right click your mouse anywhere in the editor to paste the text into the file.

Saving your changes

You can save changes in Nano by using Ctrl-o. This will prompt you to provide a file name. In this guide you can always accept the default name.

Quitting out of Nano

You can quit Nano by using Ctrl-x

Turning off your Browser Caching

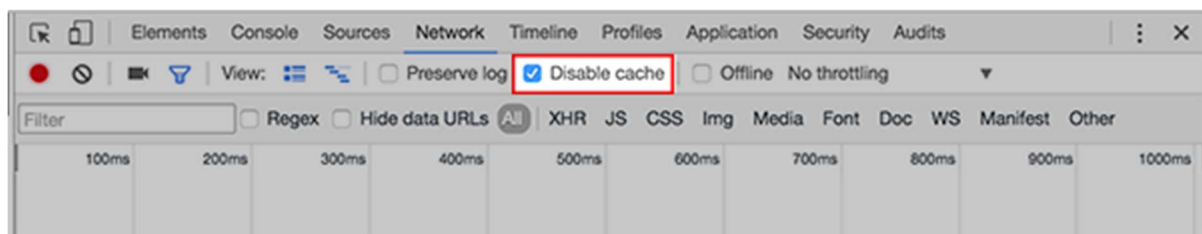
It is recommended that during an Odoo installation that you turn off the caching in your browser. This is particularly important during the configuration of Nginx and the SSL certificates. While caching is important for improving efficiency and speed of your day-to-day browsing, it can make it more difficult to test and troubleshoot your Odoo installation.

The browser doesn't explicitly get notified when you make changes to nginx configuration files. Therefore, the browser can serve up files that are cached instead of going back to the server. Another scenario that could cause problems is if you must make changes to the DNS entries on your host. The browser could rely on the cache making it harder to troubleshoot what is going on. In general, it is a good idea to turn off browser caching during your Odoo installation.

Turning off the Browser Caching in Chrome

While in chrome, open the developer tools. Find it under **Menu > More Tools > Developer Tools**. For shortcuts use Cmd + Opt + I on a Mac or F12 on Windows.

Once the developer tools are open, there is a set of tabs along the top. Select network.



Along the top of the network panel, there's a checkbox that says "Disable Caching." This disables browser-level caching, but only if the DevTools are open. So, it won't affect your normal browsing, but while working with the developer tools you won't have to worry about stale content.

Developer Installations vs Staging / Production Instances

This eBook is designed to give you a comprehensive guide on how to install Odoo 15 for a variety of scenarios. One main determination you will be making when starting an Odoo installation is if the installation is to be used for development or if it is to be used for staging or production.

Developer Installations

When you are customizing Odoo or building Odoo applications you want an Odoo installation that makes it fast and easy to restart the server. You want an installation that is easy to manage within your IDE. This means that you will likely not want to configure the installation as a service or a reverse proxy under nginx.

Staging Installations

The purpose of a staging installation is to create an environment that is as close to the production setup as possible. A staging environment should be a mirror image of the live production environment. You use this environment to test modifications, upgrades, and features to make sure all is working exactly as expected before you move those changes to a production server.

NOTE: It is **highly** recommended that you always have a staging instance as part of any Odoo production pipeline. Even if you are using stock out-of-the-box Odoo, you will want a staging server to test patches, configuration changes, and to provide an environment for users to learn and practice the business processes.

Production Installations

A production installation is arguably the most important installation to get rock solid. Take extra time to configure the necessary parameters and make sure that you thoroughly test the installation under the maximum loads you can expect while it is in live operation. You will nearly always want to configure a production installation for multi-process mode. There are details later in the book that will give you guidelines on how to determine what is the best configuration given the number of users, the number of cpu's, and the available RAM on the server.

WARNING: will want to make sure that if you are using signed SSL certificates that you complete the steps in this guide to auto-renew the certificate. Otherwise the certificate will expire in 90 days and the production instance will be inaccessible until you get a new certificate.

Updating Ubuntu and Installing the First Set of Required Packages

Let's Get Started!

Log into your Ubuntu as a sudo user and update the server applications:

```
sudo apt update
```

Next, upgrade the packages with the following command:

```
sudo apt upgrade
```

Now we will install the packages that are required before Odoo installation. These include Git, Pip, Node.js, and several others:

```
sudo apt install git python3-pip build-essential wget python3-dev python3-venv  
python3-wheel libxslt-dev libzip-dev libldap2-dev libsasl2-dev python3-setuptools  
node-less
```

```
sudo apt install postgresql postgresql-server-dev-12 build-essential python3-pillow  
python3-lxml python3-dev python3-pip python3-setuptools npm nodejs git gdebi  
libldap2-dev libsasl2-dev libxml2-dev python3-wheel python3-venv libxslt1-dev node-  
less libjpeg-dev -y
```

```
sudo pg_ctlcluster 12 main start
```

Creating a System User

It is usually not a good idea to run Odoo under a user that has sudo permissions. This is a security step you should follow and is a good practice.

Therefore, we will create a system user in which we will run our Odoo instance. While you can name this user whatever you wish, we are going to name the user 'odoo15' with the home directory located at /opt/odoo15:

```
sudo useradd -m -d /opt/odoo15 -U -r -s /bin/bash odoo15
```

NOTE: If you do decide to use a different name other than 'odoo15', make sure that you also use the exact same name when you create the PostgreSQL user in the next step!

Configuring PostgreSQL

In this installation, the postgresql server was installed during the initial packages installation in the previous step. Now, after PostgreSQL is installed, we will now create a PostgreSQL user that uses the same name as the system user we created previously. If you followed this installation guide so far and used the same name I did, then the postgres user will be 'odoo15'. Use the following command to set up the Postgres user:

```
sudo su - postgres -c "createuser -s odoo15"
```

Installing Wkhtmltopdf

Note: In my most recent video walkthrough I skip this step until we have verified that we can start the Odoo Server and successfully create a database.

Wkhtmltopdf is an open source command line tool that renders HTML into PDF. While you can get Odoo up and running without this step, you must complete it if you wish to print PDF reports from Odoo. Across many versions of Odoo, this installation has required a step separate from the many other Odoo dependencies.

To install Wkhtmltopdf, we begin with a wget command. This is a basic Ubuntu command that allows you to download any file from the Internet to your Ubuntu server:

```
sudo wget https://github.com/wkhtmltopdf/packaging/releases/download/0.12.6-1/wkhtmltox_0.12.6-1.focal_amd64.deb
```

NOTE: It is possible that this link may always not be available. There are no guarantees when it comes to these Github resources and their pathing. If there are errors in this previous command, it is likely that the resource has moved or there was trouble downloading it. Should you find yourself with this problem, you can use Google to find an alternative installation. Do a search for 'Odoo 15 wkhtmltopdf install' or something similar.

Once the Wkhtmltopdf package has been downloaded, you can install it using the following command:

```
sudo apt install ./wkhtmltox_0.12.6-1.focal_amd64.deb
```

Downloading, Installing, and Configuring Odoo 15 in a Virtual Environment

As was discussed in the introduction, we want to install Odoo into a virtual Python environment so it can be isolated from other services and applications that you may wish to install on your Ubuntu server.

First, change to user "odoo15":

```
sudo su - odoo15
```

This is a very important step, and you should see in the command window that your user is now 'odoo15'. If you don't do the above step, the source code will still be downloaded, but it will have the wrong permissions.

Next, we'll clone the Odoo 15 source code from Github into a local directory on the Ubuntu server:

```
git clone https://www.github.com/odoo/odoo --depth 1 --branch 15.0 /opt/odoo15/odoo
```

Once the source code has been downloaded, we want to create our Python virtual environment that we will use to install all the Odoo dependencies:

```
cd /opt/odoo15
```

```
python3 -m venv odoo-venv
```

We can now activate the Odoo virtual environment with the following command:

```
source odoo-venv/bin/activate
```

Once again you will see the command line change to show us that we are in the Odoo virtual environment.

Install the Required Odoo Python dependencies

Use the following command to download and install the core Python dependencies:

```
pip3 install wheel
```

```
pip3 install -r odoo/requirements.txt
```

NOTE: If you encounter any errors during the installation, you will want to double-check your steps up to this point. Typically, the problem is related to not having the right user (odoo15) or not having the virtual environment activated.

Once all the dependencies are installed, we can test our Odoo installation.

Initial test of our Odoo installation

This is somewhat of an optional step, but I typically do it just to make sure that Odoo can start up and can create a new database.

NOTE: You will want to make sure port 8069 is open on your Ubuntu server. Most cloud services such as AWS provide a way for you to do this through their security tools. Even if port 8069 is not open, you can still see in the terminal that Odoo is booting up and running, you just won't be able to actually test the Odoo installation in your browser.

To test the installation, navigate to the odoo source directory:

```
cd /opt/odoo15/odoo
```

Next, start Odoo by using the following command:

```
./odoo-bin
```


If Odoo starts correctly, you will see output that indicates Odoo is running on port 8069.

You can now test the installation (if you have port 8069 open).

Viewing your Odoo Instance in the Browser

Open your browser (I recommend Chrome), and type the following into the address bar:
`http://<your_domain_or_IP_address>:8069`

If successful, you'll be prompted to create an Odoo database. The screen should look something like this:



Master Password

Database Name

Email

Password

Phone number

Language

English (US)

Country

Demo data

Create database

 or restore a database

To stop Odoo from running and continue the installation, go back to your terminal window and hit Ctrl-c on your keyboard.

Configuring Odoo to run custom add-on modules

First, we want to deactivate our virtual environment with the following command:

`deactivate`

NOTE: If you wish to test Odoo again you must navigate to the Odoo15 directory using `cd /opt/odoo15` and activate the virtual environment using `source odoo-venv/bin/activate`.

Next, we want to create a new directory that will hold any 3rd party add-on modules you may wish to install, as well as any custom Odoo applications you may wish to create yourself. Create the directory using the following command:

```
mkdir /opt/odoo15/odoo-custom-addons
```

Right now, we are not going to do anything with this directly. It will be important when we create our Odoo configuration file. In the configuration file, we will create an `addons_path` parameter that will point to this directory.

Now we want to switch back to our sudo user to continue configuring the Odoo installation:

```
exit
```

NOTE: This is another step that is often forgotten. Make sure you are back to the sudo user and not still using the odoo15 user before continuing further.

Creating a configuration file for our Odoo installation

While we were able to start up Odoo with the simple `./odoo-bin` command, we'll want to create a configuration file to specify parameters and switches for starting Odoo. This will also allow us to make it easier to configure Odoo to automatically start whenever the server is rebooted.

Create a configuration file using the following command:

```
sudo nano /etc/odoo15.conf
```

This will bring up the Nano editor that is built into Ubuntu. You can use any text editor you like.

Next, copy the following code into the text editor:

```
[options]
; Specify the password that allows database management:
admin_passwd = admin_passwd
db_host = False
db_port = False
db_user = odoo15
db_password = False
addons_path = /opt/odoo15/odoo/addons,/opt/odoo15/odoo-custom-addons
; This is the default port. It is specified here as you will want to set this if you
are running Odoo on an alternate port.
xmlrpc_port =8069
```

```
; This is the default longpolling port. Like the xmlrpc_port we are specifying this
port for completeness
longpolling_port = 8072
; If you plan on setting up nginx it is advised to specify multiple workers in the
configuration. If you don't set this to workers > 1 then you could run into problems
when you specify the long polling blocks in the nginx config file.
workers = 2
; You will want to add a dbfilter to your config file if you have more than one
database. The ; means the command is commented out. Remove the ; and specify the
database so that your Odoo installation knows exactly which database to use for the
instance.
; dbfilter = [your database]
```

WARNING: MAKE SURE YOU CHANGE THE admin_passwd TO A SECURE PASSWORD. Otherwise anyone who comes to your Odoo installation can create, backup or even delete your Odoo databases!

NOTE: There are comments (lines with a ; at the beginning) in the config file that provide tips on why you should use a specific switch in the config file.

Creating a Systemd unit file to auto-start Odoo when the server reboots

Once again, we want to create a file with Nano (or any text editor of your choice). Use the following command to create the file:

```
sudo nano /etc/systemd/system/odoo15.service
```

Copy the following contents into the text editor.

```
[Unit]
Description=Odoo15
Requires=postgresql.service
After=network.target postgresql.service

[Service]
Type=simple
SyslogIdentifier=odoo15
PermissionsStartOnly=true
User=odoo15
Group=odoo15
```

```
ExecStart=/opt/odoo15/odoo-venv/bin/python3 /opt/odoo15/odoo/odoo-bin -c
/etc/odoo15.conf
StandardOutput=journal+console

[Install]
WantedBy=multi-user.target
```

Now we must notify Ubuntu's system services that we have created this new file.

Use the following command:

```
sudo systemctl daemon-reload
```

Now we can start the Odoo server using the Odoo service files we've created. This will also enable Odoo to start up if the server is rebooted:

```
sudo systemctl enable --now odoo15
```

Let's also verify that the Odoo service is running using the following command:

```
sudo systemctl status odoo15
```

The output should look something like below, indicating that the Odoo service is active and running:

- odoo15.service

Loaded: loaded (/etc/systemd/system/odoo15.service; enabled; vendor preset: enabled)

Active: active (running) since Sat 2020-02-10 20:06:23 UTC; 3s ago

Main PID: 6431 (python3)

Tasks: 4 (limit: 1553)

CGroup: /system.slice/odoo15.service

└─6431 /opt/odoo15/odoo-venv/bin/python3 /opt/odoo15/odoo/odoo-bin -c /etc/odoo15.conf

NOTE: Your output will not be exactly like above because your installation will have its own unique process IDs.

If you would like to see messages that are logged by the Odoo service, use can use the following command:

```
sudo journalctl -u odoo15
```

Testing the Installation

Open your browser (again, I recommend Chrome), and type the following into the address bar:

`http://<your_domain_or_IP_address>:8069`

You should be prompted to create an Odoo database. If you have already tested the installation in a previous step, you may instead get taken to the database you have already installed.

Configuring Nginx to access Odoo through a secured SSL domain name

Currently Odoo is just accessible through an IP address running on port 8069. Now if you are doing development work with this Odoo installation, then this could in fact be the end of your installation. Instead you would now turn to setting up your development environment.

However, if you intend for this Odoo installation to be available by domain name, then you will want to set up SSL and configure Nginx to access Odoo.

Configure your DNS entries to point your domain name to the Ubuntu server IP address

This is done through your domain provider. Every domain registrar offers a way in which you can point your domain name to a given IP address. You will want to complete this step as soon as possible, because it can take anywhere from a few minutes to several hours before the domain name will properly resolve to the IP address. In a worst-case scenario, it can take up to 24 hours.

IMPORTANT NOTE: For this guide, we will be using YOURWEBSITE.COM as the domain name. So anywhere you see 'YOURWEBSITE.COM' you will want to replace that with your own domain name.

As the guide is written, it is assumed you are using a root domain "YOURWEBSITE.COM" but that you would also want "WWW.YOURWEBSITE.COM" configured as well.

However, if your Odoo website will be using a subdomain, you may want to remove all references to WWW.YOURWEBSITE.COM as part of your configuration.

Installing Nginx

First, you will want to make sure that you have port 80 and port 443 open on your Ubuntu server, same as was required for port 8069.

Next, we will want to install Nginx on the Ubuntu server. To do this, use the following command:

```
sudo apt install nginx
```

At this point, you can now check to make sure that Nginx has been successfully installed and that the service is running by using the following command:


```
sudo systemctl status nginx
```

You should see output (albeit somewhat cryptic) that will confirm that the Nginx service is running.

Also, you should now be able to verify through your browser that Nginx has been successfully installed by putting the following in the address bar of your browser:

http://YOURWEBSITE.COM

If everything has been setup correctly, you should see the following screen:

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Getting and Installing your FREE SSL Certificate

There was a time where it was expensive to get a secure SSL Certificate. Fortunately, now there is a great free option from Let's Encrypt. Follow the steps below to get it all set up.

WARNING: This is another area in which people new to installing Odoo can get tripped up. Follow the steps correctly and be prepared to do a bit of troubleshooting if it is one of your first couple of times doing it.

Install the Certbot package

We install the Certbot package that will register and configure the SSL certificate using the following command:

```
sudo apt install certbot
```

Next, we will use the command below to create a strong key that will provide a high level of security:

```
sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
```

Obtaining the SSL certificate

Now we must take some steps to configure our server so that Let's Encrypt can trust it. This process makes sure someone else can't spoof your domain name and create a bogus certificate. It is fine if you don't understand all the commands and what they are doing. Basically, it is creating a temporary configuration so that the webserver can communicate with Let's Encrypt's server to verify your server's authenticity.

```
sudo mkdir -p /var/lib/letsencrypt/.well-known
```

```
sudo chgrp www-data /var/lib/letsencrypt
sudo chmod g+s /var/lib/letsencrypt
```

Creating code snippets to hold our SSL configuration details

What we are doing in this step is creating two files to hold our configuration details (known as server blocks in Nginx) so that they are easier to manage and reuse.

We do this by creating a text file using Nano, just like we did for the Odoo conf and system files:

```
sudo nano /etc/nginx/snippets/letsencrypt.conf
```

Then paste the following code into the text editor:

```
location ^~ /.well-known/acme-challenge/ {
    allow all;
    root /var/lib/letsencrypt/;
    default_type "text/plain";
    try_files $uri =404;
}
```

Next, we will create a snippet file to hold details about the SSL configuration:

```
sudo nano /etc/nginx/snippets/ssl.conf
```

Now paste the following code into that file:

```
ssl_dhparam /etc/ssl/certs/dhparam.pem;
ssl_session_timeout 1d;
ssl_session_cache shared:SSL:50m;
ssl_session_tickets off;
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
ssl_ciphers 'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-
AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-
RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-
ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-
SHA384:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-
RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-
RSA-AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-
SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-
SHA:AES256-SHA:DES-CBC3-SHA:!DSS';
ssl_prefer_server_ciphers on;
```

```
ssl_stapling on;
ssl_stapling_verify on;
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 30s;
add_header Strict-Transport-Security "max-age=15768000; includeSubdomains; preload";
add_header X-Frame-Options SAMEORIGIN;
add_header X-Content-Type-Options nosniff;
```

Now that we have our snippet files created, we can make an Nginx configuration file to use them to get our SSL certificate:

```
sudo nano /etc/nginx/sites-available/YOURWEBSITE.COM
```

WARNING: Make sure you use your own domain name in place of YOURWEBSITE.COM!

Paste the following code into your text editor:

```
server {
    listen 80;
    server_name YOURWEBSITE.COM www.YOURWEBSITE.COM;
    include snippets/letsencrypt.conf;
}
```

WARNING: Again, make sure you replace YOURWEBSITE.COM with your domain name. If you fail to do this Nginx will not be able to pick your domain name, and you won't get your SSL certificate!

Finally, we need to enable your website in Nginx. Nginx uses a separate directory to keep active sites vs. sites you are configuring (or 'available' sites). We will link to our file so we are not duplicating it. Use the following command:

```
sudo ln -s /etc/nginx/sites-available/YOURWEBSITE.COM /etc/nginx/sites-enabled/
```

WARNING: Remember. You must change YOURWEBSITE.COM to your domain name. You won't get a warning if you fail to do this. It just simply won't work!

To see the new website definition in Nginx, we must restart the Nginx server with the following command:

```
sudo systemctl restart nginx
```

NOTE: If you get any errors or problems, you will need to go back and make sure that you followed all the steps exactly up to this point!

Acquiring the SSL Certificate

Now is the moment of truth! It's time to execute the command that will get the SSL certificate, and store it on your Ubuntu server:

```
sudo certbot certonly --agree-tos --email admin@YOURWEBSITE.COM --webroot -w  
/var/lib/letsencrypt/ -d YOURWEBSITE.COM -d www.YOURWEBSITE.COM
```

WARNING: As always, change out YOURWEBSITE.COM and, in this case, also use a real email address that you can access.

If everything worked as it was supposed to, you should see output that resembles the following:

IMPORTANT NOTES:

- Congratulations! Your certificate and chain have been saved at:
/etc/letsencrypt/live/YOURWEBSITE.COM/fullchain.pem
Your key file has been saved at:
/etc/letsencrypt/live/YOURWEBSITE.COM/privkey.pem
Your cert will expire on 2020-05-10. To obtain a new or tweaked version of this certificate in the future, simply run certbot again. To non-interactively renew *all* of your certificates, run "certbot renew"
- Your account credentials have been saved in your Certbot configuration directory at /etc/letsencrypt. You should make a secure backup of this folder now. This configuration directory will also contain certificates and private keys obtained by Certbot so making regular backups of this folder is ideal.
- If you like Certbot, please consider supporting our work by:

Donating to ISRG / Let's Encrypt: <https://letsencrypt.org/donate>
Donating to EFF: <https://eff.org/donate-le>

Auto-renew your SSL Certificate

While the certificate is going to work perfectly fine now, Let's Encrypt requires that you renew the certificate every three months. So if you fail to do this next step, at some point the SSL certificate will expire, and your entire Odoo installation will be inaccessible through your web browser.

Fortunately setting up auto-renewal is easy. Simply enter the following command to edit the Certbot configuration file:

```
sudo nano /etc/cron.d/certbot
```

Then we want to APPEND the following string to the END of the listing in that file:

```
--renew-hook "systemctl reload nginx"
```

Note: You DO want to include the double quotes. If you have made the edit correctly, the entire line in the cerbot file should look something like this:

```
0 */12 * * * root test -x /usr/bin/certbot -a \! -d /run/systemd/system && perl -e  
'sleep int(rand(3600))' && certbot -q renew --renew-hook "systemctl reload nginx"
```

Modifying your Nginx configuration to access your Odoo installation with the SSL certificate

So now that we have our SSL certificate, we can update our Nginx domain definition for secure access and get it talking to our Odoo installation.

```
sudo nano /etc/nginx/sites-available/YOURWEBSITE.COM
```

Replace the existing contents of the file with the contents below:

Warning: This is a long code block that spans across several pages. Make sure you get it all.

```
# Odoo servers
upstream odooserver {
    server 127.0.0.1:8069;
}

upstream odoochat {
    server 127.0.0.1:8072;
}

# HTTP -> HTTPS
server {
    listen [::]:80;
    listen 80;
    server_name www.YOURWEBSITE.COM YOURWEBSITE.COM;
    return 301 https://YOURWEBSITE.COM$request_uri;
}

# WWW -> NON WWW
server {
    listen [::]:443 ssl;
    listen 443 ssl http2;
    server_name www.YOURWEBSITE.COM;

    ssl_certificate /etc/letsencrypt/live/YOURWEBSITE.COM/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/YOURWEBSITE.COM/privkey.pem;
    ssl_trusted_certificate /etc/letsencrypt/live/YOURWEBSITE.COM/chain.pem;
    include snippets/ssl.conf;
    include snippets/letsencrypt.conf;

    return 301 https://YOURWEBSITE.COM$request_uri;
}

server {
    listen 443 ssl http2;
    server_name YOURWEBSITE.COM;
```

```

proxy_read_timeout 720s;
proxy_connect_timeout 720s;
proxy_send_timeout 720s;

# Proxy headers
proxy_set_header X-Forwarded-Host $host;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_set_header X-Real-IP $remote_addr;

# SSL parameters
ssl_certificate /etc/letsencrypt/live/YOURWEBSITE.COM/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/YOURWEBSITE.COM/privkey.pem;
ssl_trusted_certificate /etc/letsencrypt/live/YOURWEBSITE.COM/chain.pem;
include snippets/ssl.conf;
include snippets/letsencrypt.conf;

# log files
access_log /var/log/nginx/odoo.access.log;
error_log /var/log/nginx/odoo.error.log;

# Specifies the maximum accepted body size of a client request,
# as indicated by the request header Content-Length.
client_max_body_size 200m;

# increase proxy buffer to handle some odoo web requests
proxy_buffers 16 64k;
proxy_buffer_size 128k;

# Handle longpoll requests
location /longpolling {
    proxy_pass http://odoochat;
}

# Handle / requests
location / {
    proxy_redirect off;
    proxy_pass http://odooserver;
}

# Cache static files
location ~* /web/static/ {
    proxy_cache_valid 200 90m;
    proxy_buffering on;
    expires 864000;
    proxy_pass http://odooserver;
}

```

```
# Gzip
gzip_types text/css text/less text/plain text/xml application/xml
application/json application/javascript;
gzip on;
}
```

WARNING: I know I've said it many times before! But, don't forget to replace YOURWEBSITE.COM with your own Odoo domain.

Once you have saved the file, restart the Nginx service:

```
sudo systemctl restart nginx
```

At this point, you should not see any errors from the service starting.

Change the Odoo Configuration File to use proxy mode

Finally, we need to change the Odoo configuration file to use the proxy. I've had everything work fine without this step, but it is still a good practice!

Use this command to edit the configuration file:

```
sudo nano /etc/odoo15.conf
```

Then add this line to the file:

```
proxy_mode = True
```

NOTE: Leave all the other options in the conf file in place!

Now we want to restart the Odoo service for the changes to our config file to take effect:

```
sudo systemctl restart odoo15
```

If you have followed all the steps exactly (and everything worked as it should), you will be able to access your Odoo instance by going to your web browser and simply going to your domain:

<https://YOURWEBSITE.COM>

NOTE: If your domain is not resolving, or if you are having problems, you will need to review the prior steps. Also remember that it can take a few minutes or even a few hours for the domain name to resolve to the correct IP address. If you are still having trouble, you will likely want to start this entire section over from when you first installed Nginx.

TIP: This is certainly one of the more challenging sections of completing an Odoo installation. If you are not highly technical or this is your first couple of times installing Odoo, don't be afraid to ask for professional assistance. In the end, your time is valuable, and going it alone can make you pull out your hair.

Improving security by changing the binding interface

Odoo will still run perfectly fine without doing this step. But for best practices, it can be good to disable access to Odoo on port 8069.

To disable direct access to the Odoo instance, you can either block the port 8069 for all public interfaces or force Odoo to listen only on the local interface.

In this guide, we will force Odoo to only listen on 127.0.0.1.

Open the configuration file using the following command:

```
sudo /etc/odoo15.conf
```

Then add the following two lines to the end of the file:

```
xmlrpc_interface = 127.0.0.1
netrpc_interface = 127.0.0.1
```

Now you will want to restart the Odoo server so it can restart using these new changes to the Odoo configuration file:

```
sudo systemctl restart odoo15
```

Enabling your Odoo installation for multiple worker processes

By default, Odoo is only running with one worker process. That means if multiple users are using Odoo, they are all sharing the same process. This has a dramatic impact on performance as you add additional users. For production deployments, it is highly recommended to configure Odoo using multiple workers.

To enable multiple workers in Odoo, you will begin by editing the Odoo configuration file with the following command:

```
sudo /etc/odoo15.conf
```

Unlike most of our other configurations, the values you will use here to set up multiple workers will be based on your server's CPUs and memory, and the number of users that you anticipate will be accessing the Odoo installation.

The following instructions are from Odoo's recommended guidelines:

Worker number calculation

Theoretical maximal number of worker = $(\text{system_cpus} * 2) + 1$

1 worker can serve ≈ 6 concurrent users

Cron workers also require CPU

RAM memory size calculation

We will consider that 20% of all requests are heavy requests, and 80% are lighter ones. Heavy requests are using around 1 GB of RAM while the lighter ones are using around 150 MB of RAM

Needed RAM = $\text{number_of_workers} * ((\text{light_worker_ratio} * \text{light_worker_ram_estimation}) + (\text{heavy_worker_ratio} * \text{heavy_worker_ram_estimation}))$

If you do not know how many CPUs you have on your system, use the following grep command:

```
grep -c ^processor /proc/cpuinfo
```

So for example, if you have a system with 4 CPU cores, 8 GB of RAM memory, and 30 concurrent Odoo users, here is the calculation:

$30 \text{ users} / 6 = 5$ (5 is theoretical number of workers needed)

$(4 * 2) + 1 = 9$ (9 is the theoretical maximum number of workers)

Based on the calculation above, you can use 5 workers + 1 worker as the cron worker, making a total of 6 workers.

Calculate the RAM memory consumption based on the number of workers

$\text{RAM} = 6 * ((0.8 * 150) + (0.2 * 1024)) \approx 2 \text{ GB of RAM}$

The calculation shows that the Odoo installation will need around 2GB of RAM.

Now we can edit the configuration file and append our calculated values:

```
limit_memory_hard = 2684354560
limit_memory_soft = 2147483648
limit_request = 8192
limit_time_cpu = 600
limit_time_real = 1200
max_cron_threads = 1
workers = 5
```

Now restart the Odoo service:

```
sudo systemctl restart odoo15
```

FINAL TESTING

Test your installation one more time! Create a database, install an Odoo app, and test that PDF reports work as they should.

Before you go live and tell the world, test the installation from a few different locations and browsers. You can use selected users or project stakeholders to test the installation as well.

All of this testing is to ensure that the domain has propagated. Otherwise, there may be people who have DNS servers that are not updated. To be sure, I would wait 24-48 hours before anything that could cost you money, reputation, etc. Don't tell the world to go there until you are ready!

For example, if you are planning to run \$10,000 in Facebook ads to promote your brand new Odoo website, you want to make 100% sure that it is accessible from all locations, not just from the location in which you are testing.

Finally, reboot the Ubuntu server. You want to make sure that the Odoo server automatically starts up, and that it is still running after the reboot.

Why you want to use the most recent versions of Odoo

I'm including this section in the installation guide as I am often asked about Odoo upgrades and maintenance of older systems. It is not a desirable situation to have some straggling clients on Odoo 9, others on Odoo 10 or 11, and then another batch on Odoo 12 or 13. In fact with Odoo you really never want clients using versions that are 2-3 versions back from the current. Significant progress is made each upgrade cycle in improving the architecture, performance, security, and usability of the software. Clients stuck on Odoo 10 or earlier are going to have significantly less performance. Odoo 15 is literally 900% faster than Odoo 10 or before. It is 300% faster than even just Odoo 12 in a lot of cases.

Odoo 15 makes HUGE gains in performance making it the ideal version for any client to be on. Improvements in the ORM and how data is cached dramatically improve the speed of nearly all installations. While it is not always feasible in complex installations to stay with them most recent version, you certainly never want to take systems into production under significantly older versions of Odoo if possible. This is or the simple fact that it will only be more difficult in the future to keep the software updated and maintained.

But even with what I have said, always make sure the business comes first. If everything is working fine and performance is great, and people are happy with the usability you don't need to rush an upgrade. The most important thing to remember is that you should always have a long-term plan on how you will manage Odoo upgrades.

Conclusion

If you made it this far, and everything is working as expected, CONGRATULATIONS! There are some rather complex steps in this process, and it isn't easy for those new to Ubuntu or who are not experienced technology professionals.

If you got stuck and were unable to get it all going, don't give up! Try again. Feel free to reach out for assistance. You can email me personally at gregmoss@odooClass.com, and I will be glad to help guide you through the process and make sure everything is configured properly.

Odoo Resources

I invite you to check out my various Odoo Resources:

Odoo Courses: <https://OdooClass.com>

Odoo Coaching and Live Training: <https://OdooInnerCircle.com>

Face Book: <https://www.facebook.com/odooClass/>

Linked-in: <https://www.linkedin.com/in/gregmoss/>

YouTube: [OdooClass on Youtube](#)

Books: [Working with Odoo](#)

[Working with Odoo 10](#)

[Working with Odoo 11](#)

[Learn Odoo](#)