

Set up an Odoo 20 development environment with VS Code on Ubuntu

A from-scratch install of Odoo 20 Community on Ubuntu 26.04 LTS. PostgreSQL, wkhtmltopdf and the Python environment are set up by hand, then VS Code is configured to run, debug and live-reload Odoo — and we build a first module.

OS
Ubuntu 26.04

ODOO
20.0 CE

PYTHON
3.14

POSTGRESQL
18

WKHTMLTOPDF
0.12.6.1

EDITOR
VS Code

Contents

01	Update Ubuntu	02	Install build dependencies
03	Install PostgreSQL 18 and create the odoo role	04	Install wkhtmltopdf
05	Install Node.js and rtlcss	06	Clone the Odoo 20 source
07	Create the virtual environment	08	Write odoo.conf
09	Create the demo database	10	Install VS Code
11	Run and debug Odoo from VS Code	12	Scaffold your first module
13	Start, stop and reset scripts	—	Troubleshooting
—	What's next & custom development		

WHAT YOU'LL END UP WITH

Folder layout

```
~/odoo/
├── 20.0/           # Odoo source code (git clone)
├── custom-addons/  # your own modules
│   └── hello_odoo/ # the module we build in step 12
├── venv/           # Python virtual environment
├── odoo.conf       # server configuration
├── start.sh  stop.sh  reset-demo.sh
└── .vscode/        # launch.json + settings.json
```

HOW TO COPY THE COMMANDS

- Each dark box is ready to paste. There is no `$` prompt to remove.
- A line ending in `\` continues on the next line — copy the whole box, not one line of it.
- Paste boxes **one at a time** and read the output before the next one.

LOCAL DEVELOPMENT ONLY

This guide uses throwaway credentials (`odoo` / `odoo` , `admin` / `admin`) so they can appear on screen. Never use them on a server anyone else can reach.

01 Update Ubuntu

Start from an up-to-date system so package versions match the rest of this guide.

BASH

```
sudo apt update
sudo apt upgrade -y
```

- `apt update` refreshes the list of available packages — it installs nothing.
- `apt upgrade -y` installs the newer versions; `-y` answers yes.
- A “restart services” screen? Accept the defaults. New kernel installed? Reboot before step 2.

02 Install build dependencies

pip compiles several of Odoo's Python packages from source, and these system packages give it the compiler and headers it needs.

BASH

```
sudo apt install -y git python3-dev python3-venv python3-pip build-essential \
    libpq-dev libxml2-dev libxslt1-dev libldap2-dev libsasl2-dev \
    libjpeg-dev zlib1g-dev liblcms2-dev libffi-dev libssl-dev
python3 --version
```

EXPECTED OUTPUT

Python 3.14.x

PACKAGE	WHY OD OO NEEDS IT
python3-dev, python3-venv	Python headers for compiling packages, and virtual environments
build-essential	The C compiler and <code>make</code>
libpq-dev	Needed to build <code>psycopg2</code> , the PostgreSQL driver
libxml2-dev, libxslt1-dev	Needed by <code>lxml</code> , which parses Odoo's XML views and reports
libldap2-dev, libsasl2-dev	Needed by <code>python-ldap</code> , for LDAP login
libjpeg-dev, zlib1g-dev, liblcms2-dev	Needed by <code>Pillow</code> for image handling
libffi-dev, libssl-dev	Needed by <code>cryptography</code> and <code>pyopenssl</code>

NO EXTRA PYTHON NEEDED

Odoo 20 supports Python 3.12 to 3.14, and its `requirements.txt` has version pins for Ubuntu 26.04 (“Resolute”). Ubuntu's own Python 3.14 works as it is, with no deadsnakes PPA. The command is `python3` — plain `python` only works once the venv is active (step 7).

Install PostgreSQL 18 and create the `odoo` role

Odoo stores everything in PostgreSQL. Ubuntu 26.04 ships PostgreSQL 18, so no extra repository is needed.

BASH

```
sudo apt install -y postgresql postgresql-client
pg_lsclusters
```

EXPECTED OUTPUT

Ver	Cluster	Port	Status	Owner	Data directory
18	main	5432	online	postgres	/var/lib/postgresql/18/main

Next, create a database role for Odoo. Type `odoo` when it asks for a password (nothing appears as you type — that's normal):

BASH

```
sudo -u postgres createuser --createdb --no-createrole --no-superuser --pwprompt odoo
```

Test the login over TCP, which is how Odoo will connect, then check the role's rights:

BASH

```
psql -h localhost -U odoo -d postgres -c "select current_user;"
sudo -u postgres psql -c "\du odoo"
```

EXPECTED OUTPUT

```
current_user
```

```
-----
odoo
```

```
Role name | Attributes
```

```
-----+-----
odoo      | Create DB
```

WHY NOT SUPERUSER?

Many guides make the Odoo role a superuser. It doesn't need to be one: create-database rights are enough, and the extensions Odoo uses (`pg_trgm` , `unaccent`) are “trusted” in PostgreSQL 13 and later, so a database owner can install them. `\du` should show **Create DB** only.

04

Install wkhtmltopdf

Odoo 20 still renders PDF reports (invoices, quotes, delivery slips) with wkhtmltopdf, through its auto-installed `base_report_wkhtmltox` module. Ubuntu 26.04 doesn't package it, so use the official build with patched Qt.

BASH

BASE keeps the long URL on one line

```
cd ~/Downloads
BASE=https://github.com/wkhtmltopdf/packaging/releases/download/0.12.6.1-3
wget $BASE/wkhtmltox_0.12.6.1-3.jammy_amd64.deb
ls -lh wkhtmltox_*.deb
sudo apt install -y ./wkhtmltox_0.12.6.1-3.jammy_amd64.deb
wkhtmltopdf --version
```

EXPECTED OUTPUT

wkhtmltopdf 0.12.6.1 (with patched qt)

- `ls -lh` confirms the download worked (about 20–25 MB) before you install it.
- The `./` tells apt to install the local file and fetch the fonts it depends on.

A 404 FROM WGET?

The URL was split across two lines when you copied it, so wget asked for half a URL. Paste the box above as it is — `BASE` keeps every line short.

AVOID THE OLD 0.12.5 BUILD

Some guides install the 0.12.5 build for Ubuntu 18.04 together with `libssl1.1` from the old Ubuntu archive. That version of OpenSSL is end-of-life, so don't. If the jammy (22.04) package won't install, try `wkhtmltox_0.12.6.1-3.bookworm_amd64.deb` from the same release page.

05

Install Node.js and rtlcss

Odoo uses `rtlcss` to flip its stylesheets for right-to-left languages such as Arabic and Hebrew. You don't need `less`, because Odoo has compiled SCSS with libsass for years.

BASH

```
sudo apt install -y nodejs npm
sudo npm install -g rtlcss
rtlcss --version
```

`-g` installs it system-wide, so Odoo finds it on the PATH. An `EACCES` error means `sudo` was left out.

06

Clone the Odoo 20 source

A shallow clone of the `20.0` branch only, without the full history, is much faster to download (about 200 MB).

BASH

copy both clone lines together

```
mkdir -p ~/odoo/custom-addons
git clone https://github.com/odoo/odoo.git \
  --depth 1 --branch 20.0 --single-branch ~/odoo/20.0
```

BASH

```
cd ~/odoo/20.0 && git log -1 --oneline && git branch --show-current && cd ~
ls ~/odoo
```

EXPECTED OUTPUT

```
17ff827a (grafted, HEAD → 20.0, origin/20.0) ...
20.0
20.0 custom-addons
```

FLAG

WHAT IT DOES

<code>--depth 1</code>	Takes only the latest commit — no years of history
<code>--branch 20.0</code>	The Odoo 20 branch, not master
<code>--single-branch</code>	Skips every other branch
<code>~/odoo/20.0</code>	Where the code lands. “grafted” in git log is normal for a shallow clone

CLONED INTO THE WRONG FOLDER?

If the destination got separated from the command, git clones into `./odoo` in the current folder instead. No need to download again — move it: `mv -n ~/Downloads/odoo ~/odoo/20.0`

07

Create the virtual environment and install requirements

A virtual environment keeps Odoo's Python packages separate from Ubuntu's own.

BASH

```
python3 -m venv ~/odoo/venv
source ~/odoo/venv/bin/activate
pip install --upgrade pip wheel setuptools
pip install -r ~/odoo/20.0/requirements.txt
pip install watchdog
which python
```

EXPECTED OUTPUT

```
Successfully installed Babel-2.17.0 ... psycopg2-2.9.10 ... zeep-4.3.1
/home/<you>/odoo/venv/bin/python
```

After `activate`, the prompt starts with `(venv)`. Several packages (`psycopg2`, `python-ldap`, `gevent`) compile from source, so this takes a few minutes — only an `ERROR` at the very end matters. `watchdog` isn't in `requirements.txt`, but it's what lets `--dev=reload` restart the server when you save a Python file.

Write odoo.conf

A config file saves you typing a dozen flags every time you start Odoo. You don't open an editor: paste the **whole box** into the terminal, from `cat` down to `EOF`. The terminal writes the file and replaces `$HOME` with your real home folder, because Odoo needs absolute paths.

BASH

paste the whole box at once

```
cat > ~/odoo/odoo.conf <<EOF
[options]
; Local development only: throwaway credentials
admin_passwd = odoo
db_host = localhost
db_port = 5432
db_user = odoo
db_password = odoo
dbfilter = ^demo$
addons_path = $HOME/odoo/20.0/addons,$HOME/odoo/custom-addons
http_port = 8069
dev_mode = reload,qweb,xml
EOF
```

BASH

```
cat ~/odoo/odoo.conf
```

You should see 12 lines, each on its own line, and `addons_path` should start with `/home/` — not `$HOME`.

KEY	WHAT IT DOES
admin_passwd	The master password for the database manager page (<code>/web/database/manager</code>)
db_host / db_port / db_user / db_password	The <code>odoo</code> role from step 3, over TCP
dbfilter	Only shows the <code>demo</code> database, so you're never asked to pick one
addons_path	Where Odoo looks for modules: the core addons first, then your own
dev_mode	<code>reload</code> restarts the server when Python files change. <code>qweb</code> and <code>xml</code> read views straight from disk, so template edits show up without upgrading the module.

STUCK AT A > PROMPT?

The `EOF` line didn't get pasted. Type `EOF` on its own line and press Enter. If `cat` shows `$HOME` literally, the first line was typed as `<<'EOF'` — run the box again with a plain `<<EOF`.

Until `custom-addons` contains a module, Odoo logs *“invalid addons directory... skipped”* for it. That warning is harmless, and step 12 fixes it.

Create the demo database

The database is created **in the terminal**, not in the browser. Demo data gives you sample companies, contacts, products and sales orders — ideal for tutorials. Odoo 20 no longer loads demo data by default, so ask for it explicitly.

Always start from the `odoo` folder with the venv active:

BASH

```
cd ~/odoo
source venv/bin/activate
```

YOUR PROMPT SHOULD LOOK LIKE

```
(venv) you@machine:~/odoo$
```

BASH

```
./20.0/odoo-bin db -c odoo.conf init demo --with-demo
# optional: set the demo company's country, e.g. --country GB
```

Then start the server:

BASH

```
./20.0/odoo-bin -c odoo.conf
```

EXPECTED OUTPUT

```
... HTTP service (werkzeug) running on ...:8069
```

Open **<http://localhost:8069>** yourself — the browser doesn't open on its own. You go straight to the login page, not the database manager, because the database already exists and `dbfilter` picks it for you. Sign in with **admin / admin**.

COMMAND PART	WHAT IT DOES
<code>db -c odoo.conf init demo</code>	Odoo 20's built-in database command: creates <code>demo</code> and installs the base modules
<code>--with-demo</code>	Loads sample data into it
<code>--country GB</code>	Optional: sets the country on the main company
<code>--force</code>	Drops and recreates <code>demo</code> if it already exists

Stop the server with `Ctrl + C`. From step 11 on, you start it from VS Code instead.

Install VS Code

Install Microsoft's official `.deb` package. Check first with `code --version` — if a version prints, skip to the extensions. During the install it asks whether to add Microsoft's apt repository: answer **Yes**, so VS Code updates along with the rest of Ubuntu.

```
BASH the quotes matter: & breaks the URL without them

cd ~/Downloads
wget -O code_amd64.deb \
  "https://code.visualstudio.com/sha/download?build=stable&os=linux-deb-x64"
sudo apt install -y ./code_amd64.deb
```

Install the Python extension. It brings Pylance (code completion) and debugpy (the debugger) with it. The XML extension helps when editing Odoo views.

```
BASH

code --install-extension ms-python.python
code --install-extension redhat.vscode-xml
code ~/odoo
```

If VS Code asks whether you trust the authors of this folder, it's your own folder — answer yes.

Run and debug Odoo from VS Code

This is the VS Code version of a PyCharm run configuration: run `odoo-bin` with the venv's Python and your config file. Create the two files from VS Code's terminal (`Ctrl + ``):

```
BASH

mkdir -p ~/odoo/.vscode
code ~/odoo/.vscode/settings.json ~/odoo/.vscode/launch.json
```

~/odoo/.vscode/settings.json

```
JSON

{
  "python.defaultInterpreterPath": "${workspaceFolder}/venv/bin/python",
  "python.analysis.extraPaths": ["${workspaceFolder}/20.0"],
  "files.watcherExclude": { "**/venv/**": true }
}
```

- **defaultInterpreterPath** — use the Python from our venv.
- **extraPaths** — code completion understands `from odoo import models`.
- **watcherExclude** — don't watch the thousands of files in `venv`.

~/odoo/.vscode/ launch.json

JSON

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Odoo 20: run (auto-reload)",
      "type": "debugpy",
      "request": "launch",
      "program": "${workspaceFolder}/20.0/odoo-bin",
      "args": ["-c", "${workspaceFolder}/odoo.conf"],
      "python": "${workspaceFolder}/venv/bin/python",
      "console": "integratedTerminal",
      "justMyCode": false
    },
    {
      "name": "Odoo 20: debug (breakpoints)",
      "type": "debugpy",
      "request": "launch",
      "program": "${workspaceFolder}/20.0/odoo-bin",
      "args": ["-c", "${workspaceFolder}/odoo.conf", "--dev=qweb,xml"],
      "python": "${workspaceFolder}/venv/bin/python",
      "console": "integratedTerminal",
      "justMyCode": false
    },
    {
      "name": "Odoo 20: upgrade a module",
      "type": "debugpy",
      "request": "launch",
      "program": "${workspaceFolder}/20.0/odoo-bin",
      "args": ["-c", "${workspaceFolder}/odoo.conf",
        "-d", "demo", "-u", "${input:module}"],
      "python": "${workspaceFolder}/venv/bin/python",
      "console": "integratedTerminal",
      "justMyCode": false
    }
  ],
  "inputs": [
    { "id": "module", "type": "promptString", "description": "Module to upgrade" }
  ]
}
```

WHY TWO RUN CONFIGURATIONS?

With **reload** on, Odoo restarts itself as a new process whenever a file changes, and the debugger loses its connection. The “debug” configuration turns reload off so breakpoints keep working. Use **run** while writing code and **debug** while you're investigating something. **Upgrade a module** asks for a module name and runs **-u** on the demo database.

Try it: your first breakpoint

1. Check the bottom-right of VS Code shows the **venv** Python. If not: `Ctrl+Shift+P` → *Python: Select Interpreter* → `./venv/bin/python`.
2. Press `F5`, pick **Odoo 20: run (auto-reload)**, and refresh `localhost:8069` — the Odoo log now appears inside VS Code. Stop it with `Shift+F5`.
3. In Odoo, go to **Apps** and install **Contacts**.
4. In VS Code press `Ctrl+P`, open `20.0/odoo/addons/base/models/res_partner.py`, find `def write(self, vals):` and click in the margin to set a breakpoint.
5. Press `F5`, pick **Odoo 20: debug (breakpoints)**. In the browser, change a contact's phone number and **Save**.
6. VS Code stops on your line; `vals` holds exactly the fields you changed. Press `F5` to continue.

12 Scaffold your first module

`scaffold` generates a module skeleton for you: a manifest, models, views, security rules and demo data. Stop Odoo first (`Shift+F5`).

```
BASH
cd ~/odoo && source venv/bin/activate
./20.0/odoo-bin scaffold hello_odoo custom-addons
```

FILE	WHAT IT'S FOR
<code>__manifest__.py</code>	The module's ID card: name, version, depends, data files
<code>models/models.py</code>	Your Python models
<code>views/views.xml</code>	List view, action and menus
<code>security/ir.model.access.csv</code>	Who may read and write your model
<code>controllers/, demo/</code>	Web routes and sample data

Install it:

1. Start Odoo (`F5` → **run**). The “invalid addons directory” warning is gone now.
2. Open `http://localhost:8069/odoo?debug=1` — **developer mode** is needed to see the next menu.
3. **Apps** → **Update Apps List** → Update.
4. Remove the “Apps” filter, search `hello_odoo` and click **Activate**.

HEADS-UP: THE SCAFFOLD IS ALL COMMENTED OUT

Odoo 20's template ships every model, view and access rule commented out, so the module installs as an empty shell. The next page turns it into a working app.

Make `hello_odoo` do something

1. In `__manifest__.py`, remove the `#` in front of the security line so it reads `'security/ir.model.access.csv',`
2. Replace everything in `models/models.py`:

PYTHON

```
from odoo import models, fields, api

class HelloOdoo(models.Model):
    _name = 'hello_odoo.hello_odoo'
    _description = 'Hello Odoo'

    name = fields.Char(required=True)
    value = fields.Integer()
    value2 = fields.Float(compute="_value_pc", store=True)
    description = fields.Text()

    @api.depends('value')
    def _value_pc(self):
        for record in self:
            record.value2 = float(record.value) / 100
```

3. Replace everything in `views/views.xml`:

XML

```
<odoo>
  <record id="hello_odoo_list" model="ir.ui.view">
    <field name="name">hello_odoo list</field>
    <field name="model">hello_odoo.hello_odoo</field>
    <field name="arch" type="xml">
      <list>
        <field name="name" />
        <field name="value" />
        <field name="value2" />
      </list>
    </field>
  </record>

  <record id="hello_odoo_action" model="ir.actions.act_window">
    <field name="name">Hello Odoo</field>
    <field name="res_model">hello_odoo.hello_odoo</field>
    <field name="view_mode">list,form</field>
  </record>

  <menuitem id="hello_odoo_menu_root" name="Hello Odoo" action="hello_odoo_action" />
</odoo>
```

4. Press `F5` → **Odoo 20: upgrade a module**, type `hello_odoo`, then refresh the browser. A **Hello Odoo** app appears on the home screen. Create a record with Value `50` and Value2 computes to `0.5` — a computed field in a dozen lines of Python.

13 Start, stop and reset scripts

These are shortcuts for recording. `reset-demo.sh` drops the `demo` database and its filestore and creates it again with fresh demo data, so every take starts from the same clean state. Paste each box as a whole.

BASH

start.sh

```
cat > ~/odoo/start.sh <<'EOF'
#!/usr/bin/env bash
exec "$HOME/odoo/venv/bin/python" "$HOME/odoo/20.0/odoo-bin" \
  -c "$HOME/odoo/odoo.conf" "$@"
EOF
```

BASH

stop.sh

```
cat > ~/odoo/stop.sh <<'EOF'
#!/usr/bin/env bash
pkill -f "odoo-bin -c $HOME/odoo/odoo.conf" \
  && echo "Odoo stopped" || echo "Odoo was not running"
EOF
```

BASH

reset-demo.sh

```
cat > ~/odoo/reset-demo.sh <<'EOF'
#!/usr/bin/env bash
set -e
"$HOME/odoo/stop.sh" || true
"$HOME/odoo/venv/bin/python" "$HOME/odoo/20.0/odoo-bin" db \
  -c "$HOME/odoo/odoo.conf" init demo --with-demo --force
echo "demo database recreated. Login: admin / admin"
EOF
```

BASH

```
chmod +x ~/odoo/*.sh
```

SCRIPT

USE IT TO

~/odoo/start.sh

Start Odoo from any terminal — no need to activate the venv. Extra flags pass through, e.g. `~/odoo/start.sh -u hello_odoo`

~/odoo/stop.sh

Stop a running Odoo, including one started from VS Code

~/odoo/reset-demo.sh

Get a fresh `demo` database before every take

`db init --force` is built into Odoo 20. It drops only the database you name (`demo`) and leaves any others alone. The `'EOF'` in quotes keeps `$HOME` inside the scripts, so they work for any user.

Troubleshooting

SYMPTOM	FIX
\$: command not found	You copied the <code>\$</code> prompt from an older guide. Paste only the command.
wget ... ERROR 404	The URL broke across two lines when copied. Use the boxes in this guide (step 4, step 10) as they are.
git cloned into ./odoo	The destination path got separated. Move it: <code>mv -n ~/Downloads/odoo ~/odoo/20.0</code>
python: command not found	Use <code>python3</code> , or activate the venv first: <code>source ~/odoo/venv/bin/activate</code>
pg_config not found	<code>libpq-dev</code> is missing (step 2). Don't switch to psycopg2-binary.
password authentication failed	The role's password doesn't match <code>odoo.conf</code> . Reset it: <code>sudo -u postgres psql -c "ALTER ROLE odoo PASSWORD 'odoo';"</code>
database "demo" already exists	Add <code>--force</code> , or run <code>~/odoo/reset-demo.sh</code>
No module named ...	The venv isn't active. <code>cd ~/odoo && source venv/bin/activate</code>
Address already in use	Something else is using port 8069. Run <code>~/odoo/stop.sh</code> , or change <code>http_port</code> .
PDFs show as HTML	wkhtmltopdf isn't on your PATH. Check with <code>wkhtmltopdf --version</code> (step 4).
Breakpoints ignored	Use the “debug (breakpoints)” configuration, which has reload turned off.
inotify watch limit reached	Raise it once: <code>echo fs.inotify.max_user_watches=524288 sudo tee /etc/sysctl.d/60-inotify.conf && sudo sysctl --system</code>
No “Update Apps List” menu	Turn on developer mode: <code>localhost:8069/odoo?debug=1</code>
Module not in Apps	Check its folder is in <code>addons_path</code> , restart Odoo, Update Apps List , and remove the “Apps” filter.

WHAT WE TESTED

Recorded on Ubuntu 26.04 with Python 3.14.4, PostgreSQL 18, Odoo 20.0 Community and wkhtmltopdf 0.12.6.1 (patched Qt). psycopg2 2.9.10, python-ldap 3.4.4 and gevent 24.11.1 all compiled cleanly from `requirements.txt`.

What's next

You now have Odoo 20 running from source, a PostgreSQL 18 database with demo data, VS Code running and debugging Odoo with breakpoints and auto-reload, and your first custom module installed. That's the same setup professional Odoo developers use every day.

- Add fields, views and business logic to `hello_odoo`
- Read Odoo's own modules in `20.0/addons` — the best documentation there is
- Use `reset-demo.sh` whenever you want a clean database

ODOOISTIC MEMBERSHIP

Get the source code

Join the Odooistic channel membership on YouTube for extra features — especially the **source code** for the modules we build in the tutorials, ready to drop into your own `custom-addons`.

Look for the **Join** button on the Odooistic channel.

CUSTOM DEVELOPMENT

Need something built for your business?

Odooistic builds custom Odoo modules, integrations, migrations and performance work.

contact@odooistic.co.uk

SOURCES

odoo/odoo 20.0 on GitHub (requirements.txt, release.py, odoo/cli/db.py, odoo/cli/scaffold.py, odoo/tools/config.py) · wkhtmltopdf/packaging releases · Inspired by Cybrosys's PyCharm guide, rewritten for VS Code and Ubuntu 26.04.