Odoo 20 Windows developer setup: from a new laptop

OdooMatrix · Updated 20 October 2026

Online guide: https://odoomatrix.com/blogs/install-odoo-20-windows/

Use the online Copy buttons or Word/text version for exact pasting; PDF viewers may insert line wraps.

Contents

1. Start here: the exact setup you will build
2. Prepare Windows and learn where commands go
3. Install Git, Python, C++ tools, Node.js, and VS Code
4. Install PostgreSQL 16 and check the server
5. Create a separate database role for Odoo
6. Install PDF reporting and right-to-left CSS tools
7. Create the workspace and clone Odoo 20
8. Create the virtual environment and install Python dependencies
9. Generate your local Odoo configuration
10. Start Odoo and watch its startup log
11. Create your first development database
12. Configure VS Code and the debugger
13. Create and install your first custom app
14. Make a change, upgrade the module, and use Git
15. Back up and prove you can restore
16. Your daily start, stop, and restart commands
17. Optional: pgvector for Odoo AI features
18. Troubleshoot problems without starting over
19. Completion checklist and next steps

> **About this guide**
>
> This walkthrough starts with a new laptop and ends with an editable Odoo 20 source workspace, a working custom app, VS Code debugging, and a tested database restore. Follow the steps in order; every command box identifies its shell and the text explains what success looks like.

1. Start here: the exact setup you will build

This is the **developer route** for a new Windows 11 laptop with an **Intel or AMD x64 processor**. You will run Odoo 20 Community from Git source, write a custom app, debug it in VS Code, and back up your development database. You do not need a paid Odoo subscription or a GitHub account to clone the public source.

- **Python:** Python 3.12 x64, isolated in a virtual environment. Odoo 20's source requires Python 3.12 or newer; this guide deliberately selects the 3.12 dependency set rather than an arbitrary latest Python.
- **Database:** PostgreSQL 16, a separate non-superuser role, and a database named `odoo20_dev`.

- **Tools:** Git, Visual Studio 2022 C++ Build Tools, Node.js LTS, rtlcss, wkhtmltopdf, and VS Code with Python debugging.
- **Workspace:** `%USERPROFILE%\Dev\odoo20`. Commands calculate your actual Windows user folder for you.
- **Result:** a working local Odoo at `http://localhost:8069` and your own **Learning Notes** app.

Allow time for several gigabytes of tools and source downloads. A practical development budget is 8 GB RAM or more and about 20 GB free disk space; these are planning recommendations, not Odoo's published minimum specifications.

> **One route, in order.**
>
> Do not install the all-in-one Odoo Windows `.exe` alongside this guide. It already contains a runtime and can conflict with the source server's port. If you used the earlier installer or already have PostgreSQL, use the troubleshooting section before changing services. This recipe assumes a clean laptop.

Use this Windows environment for learning and development. Odoo discourages Windows production deployment. A live business deployment needs a separate hosting and operations plan.

2. Prepare Windows and learn where commands go

1. Open **Settings** → **System** → **About**. Confirm Windows 11 and an x64-based processor. An ARM64 laptop needs a different compatibility plan; do not assume these x64 installers are native ARM packages.
2. Open **Settings** → **Windows Update**, install pending updates, and restart if requested. Plug in the laptop and connect to the internet.
3. Open a password manager or another secure place to record four different credentials: PostgreSQL administrator, Odoo database role, Odoo database-manager master password, and Odoo application login. Never put actual passwords in Git.
4. Right-click **Start** → **Terminal (Admin)**, approve the UAC prompt, and choose a **Windows PowerShell** tab. Do not choose Command Prompt for the PowerShell blocks.
5. Copy only the command text inside each box, paste it into the named shell, and press Enter. Commands have no `PS C:\>` prompt prefix. Run one box at a time and check its result before continuing.

Windows PowerShell (Run as administrator)

```
winget --version
```

Expected: a version number. If Windows says `winget` is not recognized, open Microsoft Store, install or update **App Installer** by Microsoft, then close Terminal and open a fresh administrator PowerShell window.

Windows PowerShell (Run as administrator)

```
Get-Service -Name 'postgresql*','odoo*' -ErrorAction SilentlyContinue
Get-NetTCPConnection -State Listen -LocalPort 5432,8069 -ErrorAction SilentlyContinue |
    Select-Object LocalAddress,LocalPort,OwningProcess
```

Expected on a new laptop: no Odoo/PostgreSQL services and no listeners on either port. If you see an existing server, stop here and inspect it rather than reinstalling over its data. Missing-service output is not an instruction to delete anything.

3. Install Git, Python, C++ tools, Node.js, and VS Code

Run these commands in **administrator Windows PowerShell**, one box at a time. WinGet keeps its normal package hash and installer checks. Accept installer/UAC prompts where appropriate. If a command fails, resolve its error before continuing. Exit code `3010` means a restart is required.

Git for the source checkout

Windows PowerShell (Run as administrator)

```
winget install --id Git.Git --exact --source winget --accept-package-agreements --accept-source-agreements
```

Python 3.12 x64

Windows PowerShell (Run as administrator)

```
winget install --id Python.Python.3.12 --exact --source winget --architecture x64 --accept-package-agreements --accept-source-agreements
```

This package selects the latest available 3.12 Windows installer in the WinGet catalog, not the latest Python major/minor. Keep the Python launcher enabled if the installer presents that choice. Use a maintained compatible Python patch level supplied through your organization when applicable.

Visual C++ build tools

Windows PowerShell (Run as administrator)

```
winget install --id Microsoft.VisualStudio.2022.BuildTools --exact --source winget --accept-package-agreements --accept-source-agreements --override "--wait --passive --add Microsoft.VisualStudio.Workload.VCTools --includeRecommended"
```

Wait until Visual Studio Installer finishes. If you use its graphical installer instead, select **Desktop development with C++** / **C++ build tools**, including the MSVC x64 toolset and a Windows SDK. These support Python packages that need native compilation. Reboot if the installer asks you to.

Node.js LTS and VS Code

Windows PowerShell (Run as administrator)

```
winget install --id OpenJS.NodeJS.LTS --exact --source winget --architecture x64 --accept-package-agreements --accept-source-agreements
```

Windows PowerShell (Run as administrator)

```
winget install --id Microsoft.VisualStudioCode --exact --source winget --accept-package-agreements --accept-source-agreements
```

Now close every Terminal and VS Code window. Open **Start → Windows PowerShell** normally, without "Run as administrator." Continue all project work as your normal user. Fresh windows pick up the installed tools' PATH changes.

Windows PowerShell (normal window)

```
git --version
py -3.12 --version
node --version
npm.cmd --version
code.cmd --version
```

Expected: Git reports a version, Python reports `3.12.x`, and the other commands print their versions. The `.cmd` commands avoid PowerShell's script-policy issue with `npm.ps1`. Do not change your machine-wide execution policy to get through this guide.

4. Install PostgreSQL 16 and check the server

Open a separate **administrator PowerShell** window for the PostgreSQL installer:

Windows PowerShell (Run as administrator)

```
winget install --id PostgreSQL.PostgreSQL.16 --exact --source winget --interactive --accept-package-agreements --accept-source-agreements
```

If WinGet is unavailable in your organization, obtain the Windows installer through the official PostgreSQL Windows download page and select version 16, x64. Follow the same choices below.

1. Click **Next**. Keep `C:\Program Files\PostgreSQL\16` as the installation directory so subsequent commands match.
2. Keep **PostgreSQL Server**, **pgAdmin 4**, and **Command Line Tools** selected. Stack Builder is not needed for this setup.
3. Keep the default data directory unless you have a specific storage plan.
4. When asked for a password for the **postgres** database administrator, create and save a strong password. Later `psql` prompts will ask for this password; it is not your Windows login.
5. Keep port **5432** and the default locale. Click through the summary and wait for installation to finish.
6. Uncheck **Launch Stack Builder at exit** if offered, then click **Finish**.

Windows PowerShell (normal window)

```
Get-Service -Name 'postgresql-x64-16'
& 'C:\Program Files\PostgreSQL\16\bin\psql.exe' --version
```

Expected: the service is `Running`, and `psql` reports PostgreSQL `16.x`. If the service exists but is stopped, start only that service in administrator PowerShell:

Windows PowerShell (Run as administrator)

```
Start-Service -Name 'postgresql-x64-16'
```

Do not proceed until the service starts successfully. A port check by itself is not proof that database authentication works; the next step tests it.

5. Create a separate database role for Odoo

In your **normal PowerShell** window, open PostgreSQL's command-line client:

Windows PowerShell (normal window)

```
& 'C:\Program Files\PostgreSQL\16\bin\psql.exe' -h 127.0.0.1 -p 5432 -U postgres -d postgres -W
```

At **Password:**, type the PostgreSQL administrator password from step 4 and press Enter. Password characters are not displayed. After connection, the prompt becomes `postgres=#`. The following box is **SQL and psql commands, not PowerShell**:

PostgreSQL psql prompt (postgres=#)

```
CREATE ROLE odoo_dev WITH LOGIN CREATEDB NOSUPERUSER NOCREATEROLE NOREPLICATION;
\password odoo_dev
```

When `psql` asks for the new role password twice, enter and save a **different** password for `odoo_dev`. If the role already exists because you are repeating this step, do not drop it; use `\password odoo_dev` to set its password intentionally.

PostgreSQL psql prompt (postgres=#)

```
SELECT rolname, rolcreatedb, rolsuper FROM pg_roles WHERE rolname = 'odoo_dev';
\q
```

Expected: `odoo_dev`, `rolcreatedb = t`, and `rolsuper = f`. `\q` returns you to PowerShell. Test the new role using its own password:

Windows PowerShell (normal window)

```
& 'C:\Program Files\PostgreSQL\16\bin\psql.exe' -h 127.0.0.1 -p 5432 -U odoo_dev -d postgres -W -c
'SELECT current_user, version();'
```

Expected: the user is `odoo_dev` and the server version is 16.x. Odoo must not connect as the `postgres` superuser.

6. Install PDF reporting and right-to-left CSS tools

wkhtmltopdf for PDF report headers and footers

Open the official packaging release page in your browser:

Windows PowerShell (normal window)

```
Start-Process 'https://github.com/wkhtmltopdf/packaging/releases/tag/0.12.6-1'
```

1. Under **Assets**, choose the Windows x64 installer `wkhtmltox-0.12.6-1.msvc2015-win64.exe`. This is the Windows patched-Qt 0.12.6 family referenced by Odoo's reporting requirements.
2. Run the downloaded installer, approve its UAC prompt if you trust its origin, and keep `C:\Program Files\wkhtmltopdf` as the destination.
3. Finish the wizard. Do not install a random newer PDF engine as a substitute for the compatible patched-Qt build.

rtlcss for right-to-left interfaces

Windows PowerShell (normal window)

```
npm.cmd install --global rtlcss
```

This installs the CSS conversion tool used for languages such as Arabic. Add the command folders to your **user** PATH without replacing the existing PATH:

Windows PowerShell (normal window)

```
$npmPrefix = (& npm.cmd prefix --global).Trim()
$folders = @($npmPrefix, 'C:\Program Files\wkhtmltopdf\bin', 'C:\Program Files\PostgreSQL\16\bin')
$userPath = [Environment]::GetEnvironmentVariable('Path', 'User')
$pathParts = @($userPath -split ';' | Where-Object { $_ })
foreach ($folder in $folders) {
    if (-not (Test-Path -LiteralPath $folder)) { throw "Missing folder: $folder" }
    if ($pathParts -notcontains $folder) { $pathParts += $folder }
}
[Environment]::SetEnvironmentVariable('Path', ($pathParts -join ';'), 'User')
$env:Path = $env:Path + ';' + ($folders -join ';')
& 'C:\Program Files\wkhtmltopdf\bin\wkhtmltopdf.exe' --version
& (Join-Path $npmPrefix 'rtlcss.cmd') --version
& 'C:\Program Files\PostgreSQL\16\bin\pg_config.exe' --version
```

Expected: wkhtmltopdf identifies `0.12.6` with `patched qt`, rtlcss reports its version, and pg_config reports PostgreSQL 16.x. If any tool is missing, fix that installation rather than skipping the check. Close and reopen normal PowerShell once more so future tools, including VS Code, inherit the saved PATH.

7. Create the workspace and clone Odoo 20

Use a normal PowerShell window. This creates folders in your Windows profile; it does not require administrator permissions.

Windows PowerShell (normal window)

```
$project = Join-Path $env:USERPROFILE 'Dev\odoo20'
New-Item -ItemType Directory -Force -Path $project | Out-Null
Set-Location -LiteralPath $project
'custom-addons','data','logs','backups' | ForEach-Object {
    New-Item -ItemType Directory -Force -Path (Join-Path $project $_) | Out-Null
}
if (Test-Path -LiteralPath (Join-Path $project 'src')) {
    throw 'The src folder already exists. Inspect it before cloning again.'
}
git clone --branch 20.0 --single-branch --depth 1 https://github.com/odoo/odoo.git src
if ($LASTEXITCODE -ne 0) { throw 'Git clone failed. Resolve the Git error before continuing.' }
git -C .\src branch --show-current
git -C .\src rev-parse --short HEAD
```

Expected: the branch is `20.0` and Git prints a commit ID. Keep the Odoo core checkout in `src`; put your own modules in `custom-addons`. Do not modify the upstream requirements or core code just to make an install pass.

8. Create the virtual environment and install Python dependencies

Windows PowerShell (normal window)

```
$project = Join-Path $env:USERPROFILE 'Dev\odoo20'
Set-Location -LiteralPath $project
py -3.12 -m venv .venv
& .\.venv\Scripts\python.exe --version
& .\.venv\Scripts\python.exe -m pip install --upgrade pip setuptools wheel
& .\.venv\Scripts\python.exe -m pip install -r .\src\requirements.txt
if ($LASTEXITCODE -ne 0) { throw 'Dependency installation failed. Read the pip error before
starting Odoo.' }
& .\.venv\Scripts\python.exe -m pip check
& .\.venv\Scripts\python.exe -c "import psycopg2, lxml, PIL, win32api; print('Core Windows imports
OK')"
& .\.venv\Scripts\python.exe -X utf8 .\src\odoo-bin --version
```

Expected: Python 3.12.x, `No broken requirements found.`, `Core Windows imports OK`, and `Odoo Server 20.0` (the exact version suffix can vary).

Every Python command uses the virtual environment's executable directly. There is no activation step and no reason to run `Set-ExecutionPolicy Unrestricted`. If pip reports a compiler, `pg_config`, or missing-wheel problem, check steps 3 and 6 and the troubleshooting section. Do not remove dependency pins or use `--trusted-host` to hide an error.

9. Generate your local Odoo configuration

The next command asks for two passwords: the `odoo_dev` role password you already created, and a new Odoo database-manager master password. Password entry is hidden. These are not your future Odoo application login.

Windows PowerShell (normal window)

```
$project = Join-Path $env:USERPROFILE 'Dev\odoo20'
Set-Location -LiteralPath $project
$configPath = Join-Path $project 'odoo.conf'
if (Test-Path -LiteralPath $configPath) {
    throw 'odoo.conf already exists. Keep it or back it up before intentionally regenerating it.'
}
$dbSecret = Read-Host 'Enter the existing PostgreSQL password for odoo_dev' -AsSecureString
$masterSecret = Read-Host 'Create a new Odoo database-manager master password' -AsSecureString
$dbPassword = ([System.Management.Automation.PSCredential]::new('odoo_dev',
$dbSecret)).GetNetworkCredential().Password
$masterPassword = ([System.Management.Automation.PSCredential]::new('master',
$masterSecret)).GetNetworkCredential().Password
if ([string]::IsNullOrWhiteSpace($dbPassword) -or [string]::IsNullOrWhiteSpace($masterPassword)) {
    throw 'Passwords must not be empty.'
}
$basePath = $project.Replace('\', '/')
$configText = @"
[options]
admin_passwd = $masterPassword
db_host = 127.0.0.1
db_port = 5432
db_user = odoo_dev
db_password = $dbPassword
addons_path = $basePath/src/odoo/addons,$basePath/src/addons,$basePath/custom-addons
data_dir = $basePath/data
logfile = $basePath/logs/odoo.log
log_level = info
http_interface = 127.0.0.1
http_port = 8069
workers = 0
proxy_mode = False
list_db = True
bin_path = C:/Program Files/wkhtmltopdf/bin
"@
[System.IO.File]::WriteAllText($configPath, $configText, [System.Text.UTF8Encoding]::new($false))
$currentSid = [System.Security.Principal.WindowsIdentity]::GetCurrent().User.Value
& icacls.exe $configPath /inheritance:r /grant:r "*${currentSid}:(F)"
if ($LASTEXITCODE -ne 0) { throw 'Could not apply the config file permissions.' }
Remove-Variable dbPassword,masterPassword,configText,dbSecret,masterSecret
Write-Host 'Created odoo.conf. Keep its passwords private and do not commit this file.'
```

Expected: a successful file-permissions message and `Created odoo.conf`. The file is written as UTF-8 without a byte-order mark so its first line is a valid `[options]` section.

The server binds to `127.0.0.1` for this laptop, uses one Windows development process, and leaves database management enabled locally. Paths are generated automatically. The startup commands use Python's `-X utf8` mode so configuration paths and passwords are read consistently on Windows; the debugger sets the equivalent `PYTHONUTF8` environment option. The configuration contains database credentials; do not print it in a shared terminal, send it in a support message, or publish it.

10. Start Odoo and watch its startup log

Keep **Terminal A** open while using Odoo. Run:

Windows PowerShell (normal window)

```
$project = Join-Path $env:USERPROFILE 'Dev\odoo20'
Set-Location -LiteralPath $project
& .\.venv\Scripts\python.exe -X utf8 .\src\odoo-bin --config .\odoo.conf
```

The command keeps running; the PowerShell prompt does not return. This is expected. Logs go to a file, so the terminal can look quiet. Open a second **normal PowerShell** window, **Terminal B**:

Windows PowerShell (normal window)

```
$project = Join-Path $env:USERPROFILE 'Dev\odoo20'
Get-Content -LiteralPath (Join-Path $project 'logs\odoo.log') -Tail 40 -Wait
```

Expected: startup messages for Odoo 20 and an HTTP service listening on `127.0.0.1:8069`. After database initialization you should see modules-loaded messages. If you see a traceback, resolve it before continuing. Press **Ctrl+C in Terminal B** to stop following the log; this does not stop the server in Terminal A.

Windows PowerShell (normal window)

```
Test-NetConnection -ComputerName 127.0.0.1 -Port 8069
Start-Process 'http://localhost:8069/web/database/manager'
```

Expected: `TcpTestSucceeded : True` and the Odoo database-manager page in your browser. This is an internal check, not a public website. Do not add router forwarding or open a public firewall port for this walkthrough.

11. Create your first development database

On the database-manager page, choose **Create Database** if the creation form is not already visible. Fill it as follows:

- **Master Password:** the new database-manager master password from step 9.
- **Database Name:** `odoo20_dev`, exactly. Later module commands use this name.
- **Email:** your chosen local Odoo administrator login. Record it.
- **Password:** a new strong password for that application login, distinct from the PostgreSQL and master passwords.
- **Phone Number:** provide it only if needed for your intended local setup.
- **Language and Country:** select your preferred values. Country can affect localization.
- **Demo Data:** leave it unchecked for the clean custom-module walkthrough.

1. Click **Create database** once and wait for initialization. It can take several minutes; watch Terminal B's log if needed.
2. Confirm Odoo opens the application interface and you can sign out and back in with the email and application password you just chose.
3. Open Odoo's About/version information and confirm **20.0**.

> **Four different credentials, four jobs.**
>
> The **postgres** password administers PostgreSQL. The **odoo_dev** password connects Odoo to PostgreSQL. The **master password** creates, backs up, and restores Odoo databases. Your **email and application password** sign you into the selected Odoo database. Do not interchange them.

Do not initialize a database from the command line and then assume `admin/admin` is your login. This guide creates your administrator through the browser with the password you chose.

12. Configure VS Code and the debugger

Stop the Odoo process with **Ctrl+C in Terminal A** before launching it through the debugger. Only one process can own port 8069.

Windows PowerShell (normal window)

```
$project = Join-Path $env:USERPROFILE 'Dev\odoo20'
Set-Location -LiteralPath $project
code.cmd --install-extension ms-python.python
code.cmd --install-extension ms-python.debugpy
New-Item -ItemType Directory -Force -Path .\.vscode | Out-Null
$utf8 = [System.Text.UTF8Encoding]::new($false)
$settings = @'
{
  "python.terminal.activateEnvironment": false,
  "python.defaultInterpreterPath": "${workspaceFolder}/.venv/Scripts/python.exe",
  "python.analysis.extraPaths": ["${workspaceFolder}/src"]
}
'@
$launch = @'
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Odoo 20 development",
      "type": "debugpy",
      "request": "launch",
      "python": "${workspaceFolder}/.venv/Scripts/python.exe",
      "program": "${workspaceFolder}/src/odoo-bin",
      "cwd": "${workspaceFolder}",
      "args": ["--config", "${workspaceFolder}/odoo.conf", "-d", "odoo20_dev", "--dev=xml"],
      "env": {"PYTHONUTF8": "1"},
      "console": "integratedTerminal",
      "justMyCode": false
    }
  ]
}
'@
[System.IO.File]::WriteAllText((Join-Path $project '.vscode\settings.json'), $settings, $utf8)
```

```
[System.IO.File]::WriteAllText((Join-Path $project '.vscode\launch.json'), $launch, $utf8)
code.cmd $project
```

1. In VS Code, trust the workspace only if it is the folder you created.
2. Press **Ctrl+Shift+P** → **Python: Select Interpreter** → **Enter interpreter path**. Select `.venv\Scripts\python.exe` inside this workspace. This also handles any previously cached interpreter choice.
3. Open **Run and Debug**, select **Odoo 20 development**, and press **F5**. Open `http://localhost:8069` and sign in to `odoo20_dev`.
4. Use **Shift+F5** to stop the debugger before CLI module installation or another server start. Logs still go to `logs\odoo.log`.

13. Create and install your first custom app

Stop the running server first. Paste the following PowerShell block once. It creates a complete **Learning Notes** module without requiring you to guess file names, indentation, or XML placement. It will refuse to overwrite an existing module.

Windows PowerShell (normal window)

```
$project = Join-Path $env:USERPROFILE 'Dev\odoo20'
$module = Join-Path $project 'custom-addons\learning_notes'
if (Test-Path -LiteralPath $module) {
    throw 'learning_notes already exists. Keep your work instead of overwriting it.'
}
'models','security','views' | ForEach-Object {
    New-Item -ItemType Directory -Force -Path (Join-Path $module $_) | Out-Null
}
$utf8 = [System.Text.UTF8Encoding]::new($false)
[System.IO.File]::WriteAllText((Join-Path $module '__init__.py'), "from . import models`n", $utf8)
[System.IO.File]::WriteAllText((Join-Path $module 'models\__init__.py'), "from . import note`n",
$utf8)
$manifest = @'
{
    'name': 'Learning Notes',
    'version': '20.0.1.0.0',
    'category': 'Productivity',
    'summary': 'A first custom app for local Odoo development',
    'license': 'LGPL-3',
    'depends': ['base'],
    'data': [
        'security/ir.model.access.csv',
        'views/note_views.xml',
    ],
    'application': True,
    'installable': True,
}
'@
$model = @'
from odoo import fields, models


class LearningNote(models.Model):
    _name = 'learning.note'
    _description = 'Learning Note'

    name = fields.Char(string='Title', required=True)
```

```
    description = fields.Text(string='Description')
'@
$access = @'
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_learning_note_user,learning.note user,model_learning_note,base.group_user,1,1,1,1
'@
$views = @'
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <record id="view_learning_note_list" model="ir.ui.view">
        <field name="name">learning.note.list</field>
        <field name="model">learning.note</field>
        <field name="arch" type="xml">
            <list><field name="name"/><field name="create_date"/></list>
        </field>
    </record>
    <record id="view_learning_note_form" model="ir.ui.view">
        <field name="name">learning.note.form</field>
        <field name="model">learning.note</field>
        <field name="arch" type="xml">
            <form><sheet><group>
                <field name="name"/>
                <field name="description"/>
            </group></sheet></form>
        </field>
    </record>
    <record id="action_learning_note" model="ir.actions.act_window">
        <field name="name">Learning Notes</field>
        <field name="res_model">learning.note</field>
        <field name="view_mode">list,form</field>
    </record>
    <menuitem id="menu_learning_notes_root" name="Learning Notes" sequence="10"
groups="base.group_user"/>
    <menuitem id="menu_learning_notes" name="Notes" parent="menu_learning_notes_root"
action="action_learning_note" sequence="10"/>
</odoo>
'@
[System.IO.File]::WriteAllText((Join-Path $module '__manifest__.py'), $manifest, $utf8)
[System.IO.File]::WriteAllText((Join-Path $module 'models\note.py'), $model, $utf8)
[System.IO.File]::WriteAllText((Join-Path $module 'security\ir.model.access.csv'), $access, $utf8)
[System.IO.File]::WriteAllText((Join-Path $module 'views\note_views.xml'), $views, $utf8)
Get-ChildItem -LiteralPath $module -Recurse -File | Select-Object FullName
```

Expected: six files: two `__init__.py` files, the manifest, model, access-control CSV, and XML views. Odoo 20 uses `<list>` and `list,form` here; do not copy an old `<tree>` example from a tutorial for a different version.

Install the module into the database you created. This command exits after module initialization instead of leaving a second server running:

Windows PowerShell (normal window)

```
$project = Join-Path $env:USERPROFILE 'Dev\odoo20'
Set-Location -LiteralPath $project
& .\.venv\Scripts\python.exe -X utf8 .\src\odoo-bin --config .\odoo.conf -d odoo20_dev -i
learning_notes --stop-after-init
if ($LASTEXITCODE -ne 0) { throw 'Module installation failed. Read logs\odoo.log before
continuing.' }
Get-Content -LiteralPath .\logs\odoo.log -Tail 35
```

Expected: modules-loaded messages and no traceback or XML ParseError. Start Odoo again using F5 or the daily command below, sign in to `odoo20_dev`, and open **Learning Notes → Notes → New**. Create a note titled **My first Odoo 20 note**, add a description, save it, refresh the page, and reopen it.

This proves that your source checkout, Python dependencies, PostgreSQL connection, custom add-ons path, model, views, and basic access rules work together. The sample grants internal users access to all its notes; add appropriate record rules before using this pattern for private records.

14. Make a change, upgrade the module, and use Git

1. In VS Code, open `custom-addons\learning_notes\views\note_views.xml`.
2. Change the action's `<field name="name">Learning Notes</field>` to `<field name="name">My Learning Notes</field>`. Save the file.
3. Stop the running server with **Ctrl+C** or **Shift+F5**.
4. Upgrade the module, then restart the server and refresh the browser:

Windows PowerShell (normal window)

```
$project = Join-Path $env:USERPROFILE 'Dev\odoo20'
Set-Location -LiteralPath $project
& .\.venv\Scripts\python.exe -X utf8 .\src\odoo-bin --config .\odoo.conf -d odoo20_dev -u
learning_notes --stop-after-init
if ($LASTEXITCODE -ne 0) { throw 'Module upgrade failed. Check logs\odoo.log.' }
```

Expected: after restarting, the opened notes action shows **My Learning Notes**, and your previously saved record remains. Python changes generally require a restart; XML/data changes generally require an upgrade. `--dev=xml` is a development convenience, not a replacement for every module upgrade.

Track your own module and editor configuration in a separate Git repository. This excludes Odoo core, the virtual environment, databases/filestore, backups, and the secret configuration:

Windows PowerShell (normal window)

```
$project = Join-Path $env:USERPROFILE 'Dev\odoo20'
Set-Location -LiteralPath $project
$ignore = @'
.venv/
src/
data/
logs/
backups/
pgvector/
odoo.conf
*.zip
__pycache__/
*.py[cod]
'@
[System.IO.File]::WriteAllText((Join-Path $project '.gitignore'), $ignore,
[System.Text.UTF8Encoding]::new($false))
git init
git check-ignore odoo.conf
git add .gitignore custom-addons .vscode
git status --short
```

Expected: `git check-ignore` prints `odoo.conf`. The staged files are your module, editor JSON, and `.gitignore`. If data, backups, or credentials are staged, correct the ignore/staging rules before committing.

Windows PowerShell (normal window)

```
$authorName = Read-Host 'Your Git author name'
$authorEmail = Read-Host 'Your Git author email'
if ([string]::IsNullOrWhiteSpace($authorName) -or [string]::IsNullOrWhiteSpace($authorEmail)) {
throw 'Name and email are required.' }
git config --local user.name $authorName
git config --local user.email $authorEmail
git commit -m 'Add Learning Notes development module'
```

This creates a local commit. Publishing to a Git hosting service is a separate step; you do not need to publish credentials or create a GitHub account to finish this local setup.

15. Back up and prove you can restore

Stop the debugger or running server first. Restart Odoo **without `-d odoo20_dev`** for this check so you can select both the original database and the restored copy. Keep this command running in Terminal A:

Windows PowerShell (normal window)

```
$project = Join-Path $env:USERPROFILE 'Dev\odoo20'
Set-Location -LiteralPath $project
& .\.venv\Scripts\python.exe -X utf8 .\src\odoo-bin --config .\odoo.conf
```

In Terminal B, open the database manager:

Windows PowerShell (normal window)

```
Start-Process 'http://localhost:8069/web/database/manager'
```

1. Choose **Backup** for `odoo20_dev`.
2. Enter the database-manager master password, choose the **ZIP** format, and include the **filestore** when the option is shown. A PostgreSQL dump alone does not contain uploaded attachments.
3. Save the downloaded ZIP in your workspace's `backups` folder, and keep a separate copy outside the laptop for anything important.
4. Choose **Restore Database**, select the ZIP, and use the new name `odoo20_restore_check`. Do not overwrite or delete `odoo20_dev`.
5. Choose the copy/neutralized test option when offered so the restored database is treated as a separate test copy. Complete the restore with the master password.
6. Sign in to the restored database with the application credentials included in the backup. Open Learning Notes and confirm your saved note is present. If you added an attachment before the backup, verify it too.

Expected: the original development database still exists, and the restored copy contains the record. A backup file existing on disk is not enough; this restore is the completion check.

16. Your daily start, stop, and restart commands

After a laptop restart, PostgreSQL should start as a Windows service. Odoo source does **not** run automatically in this guide. Open normal PowerShell and run:

Windows PowerShell (normal window)

```
$project = Join-Path $env:USERPROFILE 'Dev\odoo20'
Set-Location -LiteralPath $project
& .\.venv\Scripts\python.exe -X utf8 .\src\odoo-bin --config .\odoo.conf -d odoo20_dev --dev=xml
```

Open `http://localhost:8069` in the browser. Keep this terminal running. Press **Ctrl+C** in that terminal to stop Odoo, then run the same command to restart it. Alternatively, open the workspace in VS Code and use F5 / Shift+F5.

Stop Odoo before shutting down PostgreSQL or doing an intentional environment change. For source updates, first back up the database and filestore and commit your custom code. Pull changes only when you are ready to review release/dependency changes; this guide does not silently update your source each time you start.

17. Optional: pgvector for Odoo AI features

The base developer workspace and Learning Notes app do not require AI-provider access. If you want to experiment with Odoo 20 features that require `pgvector`, the extension is an additional dependency. External AI services can also require their own account and configuration.

1. Close Odoo before changing the database environment.
2. From Start, find **x64 Native Tools Command Prompt for VS 2022**, right-click it, and choose **Run as administrator**.
3. These are **Command Prompt commands, not PowerShell**. Use the x64 prompt so `nmake`, the compiler, and the PostgreSQL architecture match:

x64 Native Tools Command Prompt for VS 2022 (Administrator)

```
set "PGROOT=C:\Program Files\PostgreSQL\16"
cd /d "%USERPROFILE%\Dev\odoo20"
git clone --branch v0.8.1 --depth 1 https://github.com/pgvector/pgvector.git pgvector
cd pgvector
nmake /F Makefile.win
nmake /F Makefile.win install
```

Check that compilation and installation finish without errors before continuing. Do not rerun the clone over an existing folder. Return to normal PowerShell, enable the extension in your development database using the PostgreSQL administrator account, and enter its password at the prompt:

Windows PowerShell (normal window)

```
& 'C:\Program Files\PostgreSQL\16\bin\psql.exe' -h 127.0.0.1 -p 5432 -U postgres -d odoo20_dev -W -
c 'CREATE EXTENSION IF NOT EXISTS vector;'
```

```
& 'C:\Program Files\PostgreSQL\16\bin\psql.exe' -h 127.0.0.1 -p 5432 -U postgres -d odoo20_dev -W -
c "SELECT extname, extversion FROM pg_extension WHERE extname = 'vector';"
```

Expected: a `vector` extension row. Extensions are enabled per database; repeat the deliberate database-level setup for a different test database when needed. If the compiler cannot find the PostgreSQL headers, confirm that PGROOT points to your PostgreSQL 16 installation and retry from the x64 Native Tools Command Prompt shown above.

18. Troubleshoot problems without starting over

A tool is not recognized

Close all terminal/editor windows and reopen a normal PowerShell window. Repeat the relevant version check. If `winget` itself is missing, update Microsoft's App Installer. If `npm.ps1` is blocked, use `npm.cmd`. Use the virtual environment's `python.exe` directly rather than changing execution policy.

pip says Microsoft C++ or pg_config is missing

Confirm the C++ workload and Windows SDK were installed, reboot if requested, and check that `C:\Program Files\PostgreSQL\16\bin\pg_config.exe` exists. Verify that the environment is Python 3.12 x64. Retry the original `pip install -r .\src\requirements.txt` after fixing the prerequisite; do not edit the requirement pins, disable TLS verification, or use an unrelated Python version to hide the error.

PostgreSQL password authentication failed

Repeat the role connection test in step 5. Check whether the prompt expects `postgres`, `odoo_dev`, the master password, or an application login. If the role password changed, update only `db_password` in your local configuration, preserving the `[options]` section and file permissions, then restart Odoo. Do not send the configuration to support.

Port 8069 is already in use

Stop your previous terminal/debugger process first. To identify another listener without terminating an unrelated program:

Windows PowerShell (normal window)

```
Get-NetTCPConnection -State Listen -LocalPort 8069 -ErrorAction SilentlyContinue |
    Select-Object LocalAddress,LocalPort,OwningProcess
Get-Service -Name 'odoo*' -ErrorAction SilentlyContinue
```

If this shows an old packaged Odoo installation, inspect its service name and stop only that known local test instance from `services.msc`. Do not kill every Python process or uninstall a database server as a quick fix.

The official packaged Odoo installer can register `odoo-server-20.0` and `PostgreSQL_For_Odoo`; builds can override the Odoo service name. This source-based guide instead uses the standalone

`postgresql-x64-16` service and your own foreground Python process. Packaged-installation roles and paths can differ, so inspect an existing setup rather than assuming it uses this guide's `postgres`/`odoo_dev` accounts.

The custom app is missing or an XML view fails

Confirm the module is under `custom-addons\learning_notes`, both import files exist, the manifest lists the CSV and XML, and your config includes the custom add-ons folder. Check the log for the first traceback. Use `<list>`, not the older `<tree>`, and stop the server before running the module install/upgrade command against `odoo20_dev`.

The log begins with a configuration-file error

Make sure the first section is `[options]`. The configuration-generation command writes UTF-8 without a BOM. A file saved with a BOM or copied with Markdown fence markers can fail before Odoo starts. Keep a copy before editing and never overwrite a working configuration just to repeat a tutorial.

PDF output lacks headers/footers

Verify the patched-Qt wkhtmltopdf version and its installation path from step 6. Restart Odoo after correcting PATH or `bin_path`. Check the report error in the log rather than installing a different PDF library.

A download or connection fails

Read the actual proxy, DNS, TLS, disk-space, package, or access-denied error. Use the official source or your organization's approved network configuration; do not disable package signature/hash checks or Windows security protection. A failed prerequisite is a reason to fix that step, not to continue with half the environment.

19. Completion checklist and next steps

- Git, Python 3.12, Node/npm, VS Code, C++ build tools, PostgreSQL 16, wkhtmltopdf, and rtlcss are installed.
- The source checkout is on branch `20.0`, the virtual environment passes `pip check`, and `odoo-bin --version` reports Odoo 20.
- The `odoo_dev` role can log in and create databases without being a superuser.
- Odoo starts with your local config, you can sign in to `odoo20_dev`, and only one development server owns port 8069.
- VS Code uses the correct interpreter and starts/stops the debugger.
- Your Learning Notes module installs, saves a record, and upgrades without losing that record.
- Your Git commit excludes passwords, the core checkout, databases/filestore, and backups.
- A ZIP backup has been restored under a different database name and its contents checked.

If each item is true, the local developer setup is complete. Enterprise modules, production hosting, mail delivery, payment providers, and external AI credentials are separate integrations, not hidden prerequisites

for this Community development workspace.

These steps target Odoo 20.0, Python 3.12, and PostgreSQL 16 on Windows 11 x64. Commands and examples were checked for consistency; the full installation has not been tested on a new Windows laptop. Installer screens and dependencies can change.

Continue learning

- Back up and restore your Odoo database
- Set up right-to-left languages
- Plan an Odoo implementation

← Back to all articles