



Open in app

Get started



Published in plusteam



Julia Suarez

Follow

May 19, 2021 · 6 min read · Listen



Save



How to run odoo tests with Pycharm

There is an example repository with an addon and all the scripts mentioned here:

<https://github.com/JSilversun/odoo-testing-example>

If you are an odoo developer, you probably would like to write some unit tests like regular pytest/unit test suites so you can debug, get coverage reports and enhance your own developer experience.

So, how can we achieve it?

Well, let's start by using a docker-compose file to get our odoo and database running:

```
1  version: '2'
2
3  services:
4    web:
5      container_name: plusteam-odoo-web
6      build:
7        context: .
8        dockerfile: Dockerfile
9      depends_on:
10       - db
11      ports:
12       - "8069:8069"
13      volumes:
14       - odoo-web-data:/var/lib/odoo
15       - ./config:/etc/odoo
16       - ./addons:/mnt/extra-addons
```





Open in app

Get started

```

22     - "5432:5432"
23     environment:
24     - POSTGRES_DB=postgres
25     - POSTGRES_PASSWORD=odoo
26     - POSTGRES_USER=odoo
27     - PGDATA=/var/lib/postgresql/data/pgdata
28     volumes:
29     - odoo-db-data:/var/lib/postgresql/data/pgdata
30 volumes:
31     odoo-web-data:
32     odoo-db-data:

```

Note: The `--dev all` args will allow your odoo instance to restart once it detects changes on your addons or related volumes code.

And a `Dockerfile` like this:

```

1 FROM odoo:14
2 USER root
3 RUN apt-get update && \
4     apt-get install -y --no-install-recommends \
5     python3-pip
6 RUN mkdir -p /coverage/all && mkdir -p /coverage/local && chown -R odoo /coverage
7 RUN pip3 install pytest-odoo coverage pytest-html
8 USER odoo

```

Dockerfile hosted with ❤ by GitHub

[view raw](#)

In this file we see how we extend an odoo image, we install some packages like `pytest-odoo` `coverage` and `pytest-html` to generate coverage reports and run odoo's tests with Pycharm. Also, a coverage directory is created with proper permissions so the coverage files can be copied from the docker container to the host machine later.

Now we run:

```
docker-compose up -d
```



[Open in app](#)[Get started](#)

```
~/Code/plus-odoo feature/poc*  
> docker ps  
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES  
ddc5730bcd56   odoo:14   "/entrypoint.sh --de..." 2 minutes ago  Up 2 minutes  0.0.0.0:8069->8069/tcp, 8071-8072/tcp  plusteam-odoo-web  
1f717cd8ab9d   postgres:13 "docker-entrypoint.s..." 2 minutes ago  Up 2 minutes  0.0.0.0:5432->5432/tcp              plusteam-odoo-db
```

Now we can go to `http://localhost:8069` and we will see something like this:

The screenshot shows the Odoo installation wizard interface. At the top is the Odoo logo. Below it is a yellow warning box stating: "Warning, your Odoo database manager is not protected. To secure it, we have generated the following master password for it: vveu-tv36-c5g3. You can change it below but be sure to remember it, it will be asked for future operations on databases." Below the warning box are several input fields: "Master Password" (with a toggle to show/hide), "Database Name" (filled with "odoo"), "Email" (filled with "odoo@example.com"), "Password" (with a toggle to show/hide), "Phone number", "Language" (dropdown menu set to "English (US)"), "Country" (dropdown menu set to "Venezuela"), and "Demo data" (checkbox). At the bottom, there is a blue button labeled "Create database" followed by the text "or restore a database".

Feel free to change the connection details, just make sure you remember them.

At this point you will have your odoo instance running, you were able to use a docker compose to start an odoo instance with a Postgres database, it's easy to change the odoo version you need with the image tag.

Now the question will be, if we have the odoo instance running in a docker, how can we run the tests?

I created a `Makefile` to teach you how to generate coverage reports or set up a database only for testing purposes:

```
1  export PGPASSWORD:=odoo  
2  
3  addon_scaffold:  
4      docker exec -it --user root plusteam-odoo-web /usr/bin/odoo scaffold $(ADDON_NAME) /mnt/extr  
5
```





Open in app

Get started

```

11  -docker exec -it -u root plusteam-odoo-web coverage run /usr/bin/odoo -d db_test --test-enab
12  docker exec -it -u root plusteam-odoo-web coverage html -d /coverage/all
13  docker cp plusteam-odoo-web:/coverage/all coverage
14
15  init_test_db:
16  docker stop plusteam-odoo-web
17  docker exec -t plusteam-odoo-db psql -U odoo -d postgres -c "DROP DATABASE IF EXISTS db_test
18  docker exec -t plusteam-odoo-db psql -U odoo -d postgres -c "CREATE DATABASE db_test"
19  docker start plusteam-odoo-web
20  docker exec -u root -t plusteam-odoo-web odoo -i all -d db_test --stop-after-init
21  docker exec -u root -t plusteam-odoo-web odoo -i plus_poc -d db_test --stop-after-init

```

First, we need to set up a database only to run the test so we don't use the main database we use for our manual development, you can run `make init_test_db` to create a test database or reset it.

Start by connecting to the container using:

```
docker exec -it plusteam-odoo-web bash
```

Now you can run:

```
odoo --help
```

This is the odoo binary, if you take a look at all the options it provides, you will be able to see a test section with all the flags available.

Run the tests with:

```
odoo --test-enable -d db_test -p 8001
```

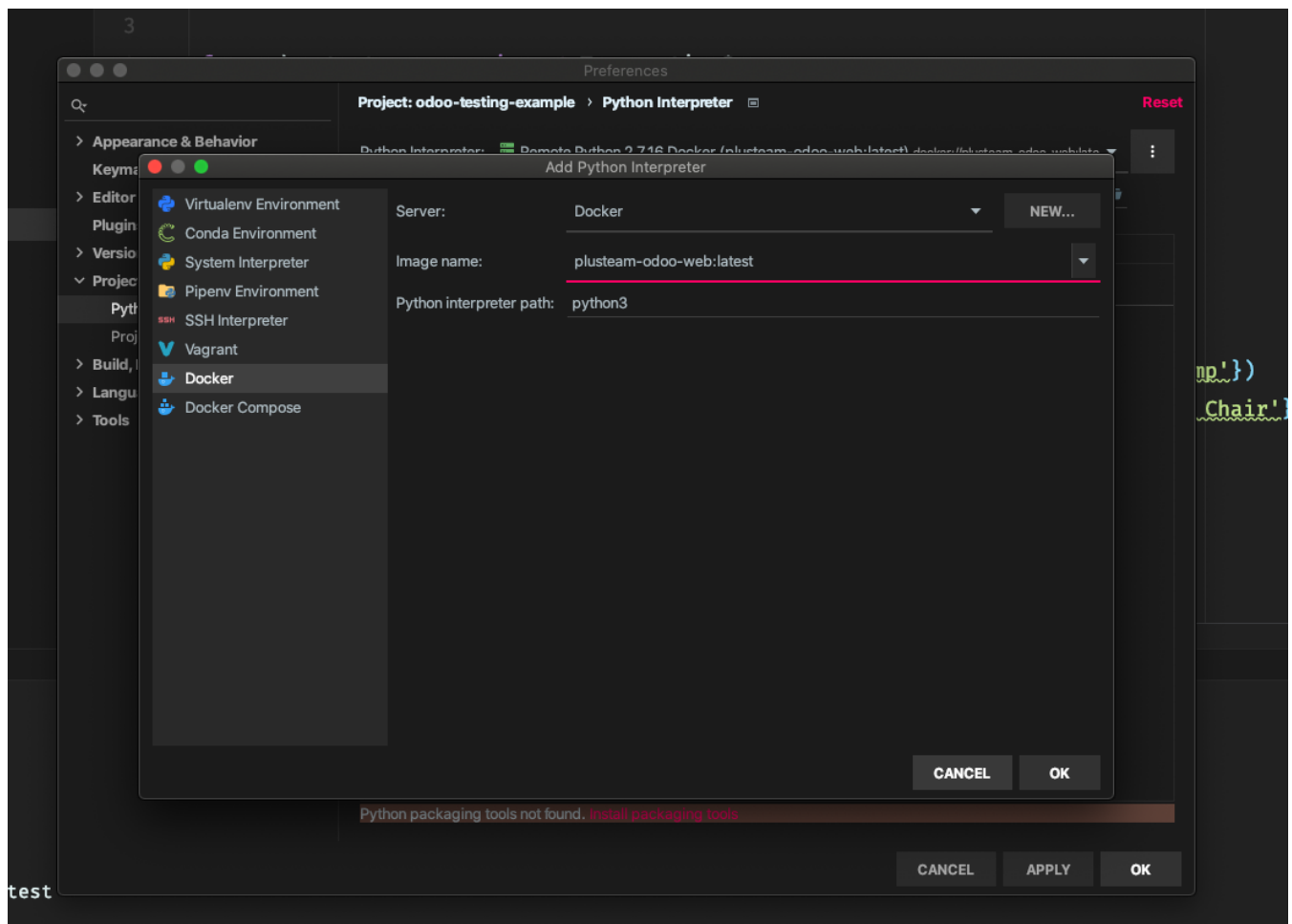


[Open in app](#)[Get started](#)

This is the “default” way to run odoo’s test, however, since this runs in a docker container, it can have a terrible developer experience since you can’t use your IDE to debug them.

So, how can we run our tests using Pycharm?

Go to Settings > Project > Python Interpreter



Add a new python interpreter, select docker, and specify the image name of the service. If you can’t find the image name for your customized odoo image you can generate an image with a name like this:

```
docker build -t <image_name> .
```





Open in app

Get started

A summary of how remote interpreters work in Pycharm: They will use the image name to launch a docker container anytime you run something through your IDE, for example:

When you open the Python console of your IDE like this:

```
Python Console
import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/Users/js/Code/plus-odoo', '/Users/js/Code/plus-odoo'])

PyDev console: starting.
Python 3.7.3 (default, Jan 22 2021, 20:04:44)
[GCC 8.3.0] on linux

>>>
```

Pycharm will start a docker container based on the image you provided:

```
~/Code/plus-odoo feature/poc*
> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
61f4bedc95be	61474514af2e	"python3 /opt/.pycha..."	About a minute ago	Up About a minute	8069/tcp, 8071-8072/tcp, 0.0.0.0:52866->52866/tcp	brave_albattani

This is why the image used needs to have available `pytest-odoo` so we can run our tests later.

This container was started automatically by Pycharm as soon as you open the python console, the same will happen if you having any python script using a remote interpreter.

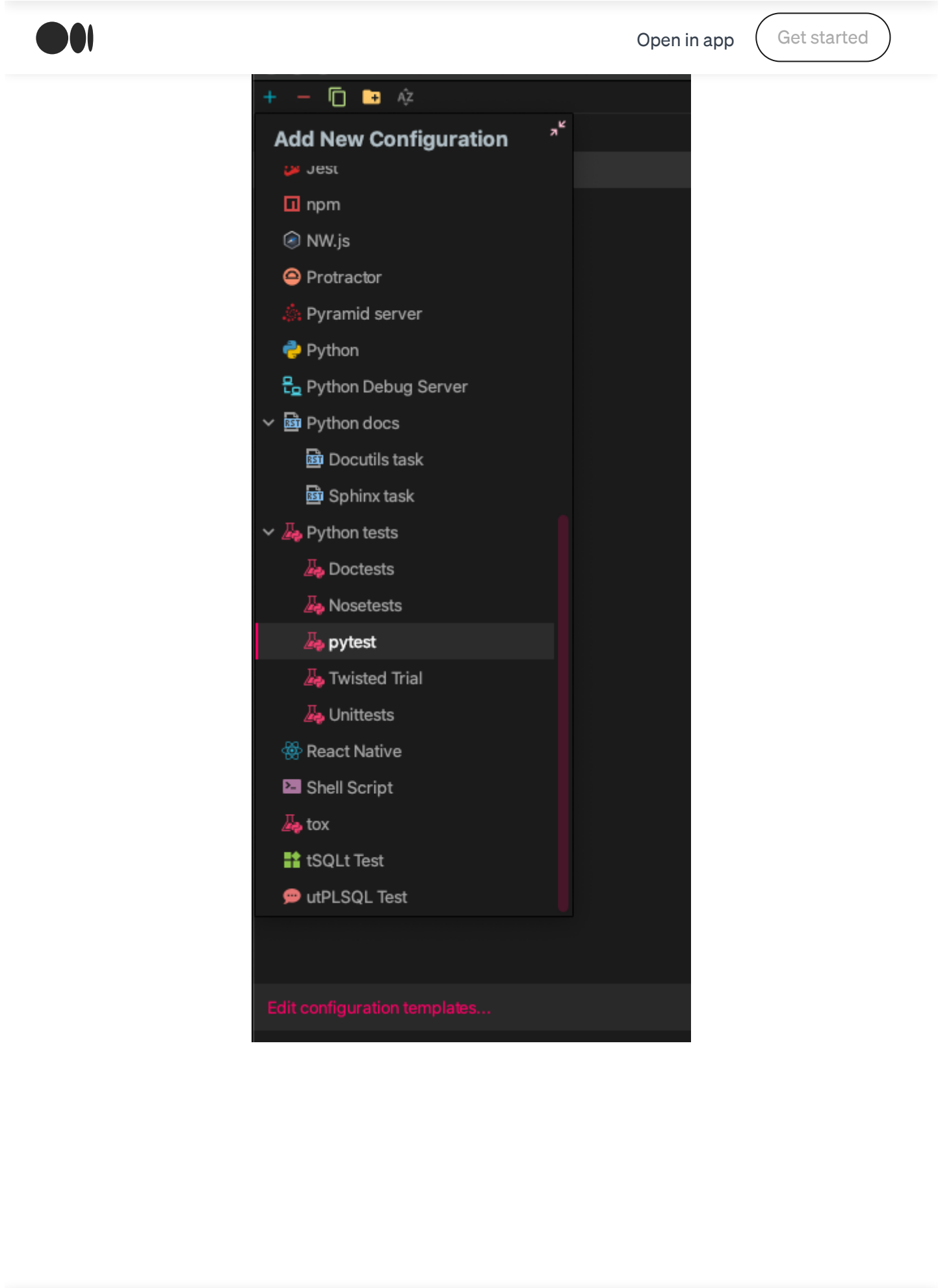
You can connect to the docker container as you would do with any kind of container like this:

```
docker exec -it <container_name|container_id> bash
```

Then you will realize that your container looks a bit different than you might expect:

```
odoo@61f4bedc95be:/opt/project$ ls
Dockerfile Dockerfile.web Makefile OdooWithBoto README.md addons boto-test config db.dump docker-compose.yml odoo.conf set_env.sh src up wait-for-psql.py
odoo@61f4bedc95be:/opt/project$
```

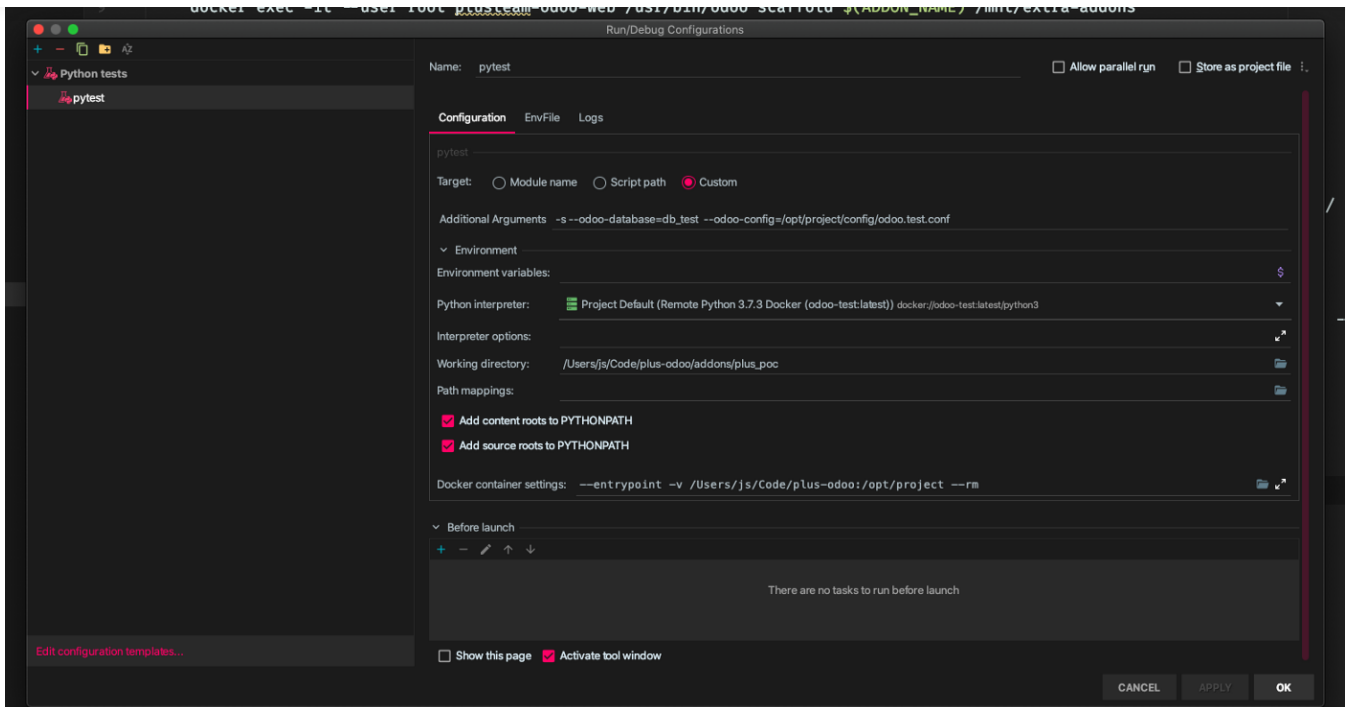






Open in app

Get started



It's important to note that Pycharm won't use the volume bindings specified in the `docker-compose` file, this means the `odoo.conf` file from the `config` folder won't be copied to the folder odoo expects and this leads to problems when we run our tests because odoo won't know where the addons are located, to solve this, we specified an `odoo.conf` file used only for tests which main difference is mapping the addons path to `/opt/project/addons` since this is the directory where Pycharm will copy the entire project's code by default.

Note: Review your database connection to make sure the `odoo.test.conf` has proper credentials for the test database.

With these settings now our tests work as expected!





Open in app

Get started

```

1  # -*- coding: utf-8 -*-
2  # Part of Odoo. See LICENSE file for full copyright and licensing details.
3
4  from odoo.tests.common import TransactionCase
5
6
7  class TestPricelist(TransactionCase):
8
9
10     def setUp(self):
11         super(TestPricelist, self).setUp()
12
13         self.datacard = self.env['product.product'].create({'name': 'Office Lamp'})
14         self.usb_adapter = self.env['product.product'].create({'name': 'Office Chair'})
15         self.uom_ton = self.env.ref('uom.product_uom_ton')
16         self.uom_unit_id = self.ref('uom.product_uom_unit')
17         self.uom_dozen_id = self.ref('uom.product_uom_dozen')
18         self.uom_kgm_id = self.ref('uom.product_uom_kgm')

```

Tests passed: 2 of 2 tests - 278 ms

d1356ca03b41:python3 -u /opt/.pycharm_helpers/pycharm/_jb_pytest_runner.py --s --odoo-database=db_test --odoo-config=/opt/project/config/odoo

Testing started at 7:17 PM ...

Launching pytest with arguments -s --odoo-database=db_test --odoo-config=/opt/project/config/odoo.test.conf --no-header --no-summary -q in /op

test session starts

collecting ... collected 2 items

addons/plus_poc/tests/test_pricelist.py::TestPricelist::test_10_discount

addons/plus_poc/tests/test_pricelist.py::TestPricelist::test_20_pricelist_uom

We can even debug our tests:

```

1  # -*- coding: utf-8 -*-
2  # Part of Odoo. See LICENSE file for full copyright and licensing details.
3
4  from odoo.tests.common import TransactionCase
5
6
7  class TestPricelist(TransactionCase):
8
9
10     def setUp(self):
11         super(TestPricelist, self).setUp()
12
13         self.datacard = self.env['product.product'].create({'name': 'Office Lamp'})
14         self.usb_adapter = self.env['product.product'].create({'name': 'Office Chair'})
15         self.uom_ton = self.env.ref('uom.product_uom_ton')
16         self.uom_unit_id = self.ref('uom.product_uom_unit')
17         self.uom_dozen_id = self.ref('uom.product_uom_dozen')

```

Debug: pytest x

Debugger

Frames

Variables

self = (TestPricelist) test_10_discount (odoo.addons.plus_poc.tests.test_pricelist.TestPricelist)

Special Variables

TestPricelist::setUp()

Coverage

Coverage for custom addons



consider the tests inside your custom addons, since I don't know yet how to tell pytest to run all the tests inside odoo's binary.

```
~/Code/plus-odoo feature/poc 58s
> make generate_local_coverage_report
docker exec -it plusteam-odoo-web pytest -s --odoo-database=db_test --html=/coverage/local/report.html /mnt/extra-addons/
===== test session starts =====
platform linux -- Python 3.7.3, pytest-6.2.4, py-1.10.0, pluggy-0.13.1
rootdir: /mnt/extra-addons
plugins: metadata-1.11.0, html-3.1.1, odoo-0.6.0
collected 2 items

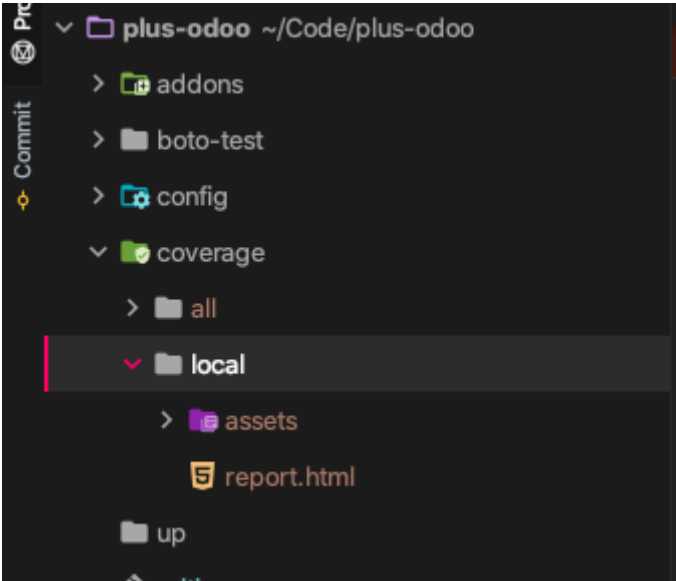
mnt/extra-addons/plus_poc/tests/test_pricelist.py

===== warnings summary =====
plus_poc/tests/test_pricelist.py::TestPricelist::test_10_discount
  /usr/lib/python3/dist-packages/jinja2/sandbox.py:82: DeprecationWarning: Using or importing the ABCs from 'collections' instead of from 'collections.abc' is deprecated, and in 3.8 it
  will stop working
    from collections import MutableSet, MutableMapping, MutableSequence

usr/local/lib/python3.7/dist-packages/_pytest/cache/provider.py:428
  /usr/local/lib/python3.7/dist-packages/_pytest/cache/provider.py:428: PytestCacheWarning: cache could not write path /mnt/extra-addons/.pytest_cache/v/cache/nodeids
    config.cache.set("cache/nodeids", sorted(self.cached_nodeids))

usr/local/lib/python3.7/dist-packages/_pytest/stepwise.py:49
```

How it looks like when you run the script



After the script finishes, you will get the report in the coverage/local folder

report.html

Report generated on 12-May-2021 at 04:53:06 by pytest-html v3.1.1

Environment

Packages

["pluggy": "0.13.1", "py": "1.10.0", "pytest": "6.2.4"]

Platform

Linux-4.19.130-boot2docker x86_64-with-debian-10.9

Plugins

["html": "3.1.1", "metadata": "1.11.0", "odoo": "0.6.0"]

Python

3.7.3

Summary

2 tests ran in 1.10 seconds.

(Un)check the boxes to filter the results.

☒ 2 passed, ☐ 0 skipped, ☐ 0 failed, ☐ 0 errors, ☐ 0 expected failures, ☐ 0 unexpected passes

Results

Show all details / Hide all details

Result	Test	Duration	Links
Passed (show details)	plus_poc/tests/test_pricelist.py::TestPricelist::test_10_discount	0.87	



Open in app

Get started

Coverage for all tests

I also read about how can I can generate a coverage report for all tests even the ones inside odoo's binary.

The process is similar:

```

10 docker exec -it --user root plusteam-odoo-web /usr/bin/odoo scartold &({AUXUN_NAME} /mnt/extra-addons
11
12 rebuild:
13     docker-compose down
14     docker-compose up -d --build
15
16 generate_local_coverage_report:
17     docker exec -it plusteam-odoo-web pytest -s --odoo-database=db_test --html=/coverage/local/report.html /mnt/extra-addons/
18     docker cp plusteam-odoo-web:/coverage/local coverage
19
20 generate_coverage_report:
21     -docker exec -it -u root plusteam-odoo-web coverage run /usr/bin/odoo -d db_test --test-enable -p 8001 --stop-after-init --log-level=test
22     docker exec -it -u root plusteam-odoo-web coverage html -d /coverage/all
23     docker cp plusteam-odoo-web:/coverage/all coverage
  
```

```

~/Code/plus-odoo feature/poc*
> make generate_coverage_report
docker exec -it -u root plusteam-odoo-web coverage run /usr/bin/odoo -d db_test --test-enable -p 8001 --stop-after-init --log-level=test
Running as user 'root' is a security risk.
2021-05-12 04:56:06,658 69 INFO ? odoo: Odoo version 14.0-20210506
2021-05-12 04:56:06,659 69 INFO ? odoo: Using configuration file at /etc/odoo/odoo.conf
2021-05-12 04:56:06,659 69 INFO ? odoo: addons paths: ['/usr/lib/python3/dist-packages/odoo/addons', '/var/lib/odoo/.local/share/Odoo/addons/14.0', '/mnt/extra-addons']
2021-05-12 04:56:06,660 69 INFO ? odoo: database: odoo@docker.lan:5432
2021-05-12 04:56:06,879 69 INFO ? odoo.addons.base.models.ir_actions_report: Will use the Wkhtmltopdf binary at /usr/local/bin/wkhtmltopdf
2021-05-12 04:56:06,967 69 INFO ? odoo.service.server: HTTP service (werkzeug) running on d059b9c136f3:8001
2021-05-12 04:56:06,977 69 INFO db_test odoo.modules.loading: loading 1 modules...
2021-05-12 04:56:07,167 69 INFO db_test odoo.addons.base.tests.test_acl: Starting TestACL.test_field_crud_restriction ...
2021-05-12 04:56:07,327 69 INFO db_test odoo.addons.base.tests.test_acl: Starting TestACL.test_field_visibility_restriction ...
  
```

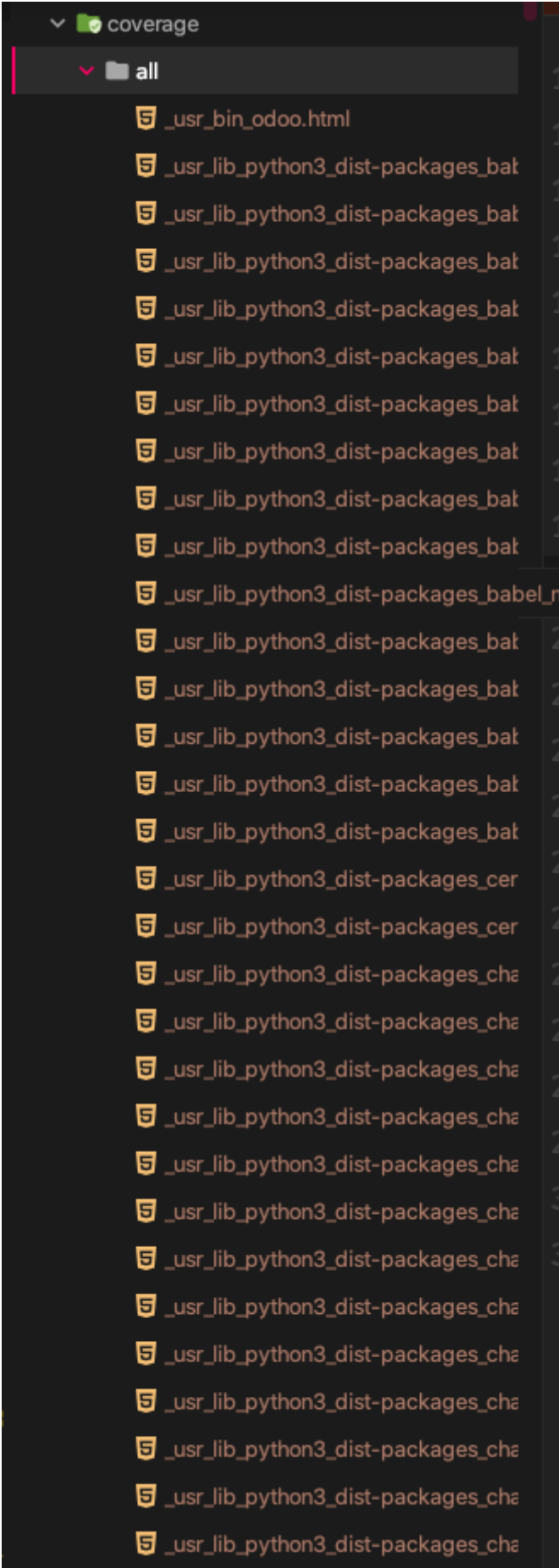
However this process will take more time, once it's done, you can see the html report in the `coverage/all` folder.






Open in app

Get started





Open in app

Get started

< > ↺ ① File | /Users/js/Code/plus-odoo/coverage/all/index.html

Coverage report: 42%

Module	statements	missing	excluded	coverage
/usr/bin/odoo	4	0	0	100%
/usr/lib/python3/dist-packages/PIL/BmpImagePlugin.py	131	73	0	44%
/usr/lib/python3/dist-packages/PIL/GifImagePlugin.py	429	227	0	47%
/usr/lib/python3/dist-packages/PIL/GimpGradientFile.py	65	51	0	22%
/usr/lib/python3/dist-packages/PIL/GimpPaletteFile.py	26	20	0	23%
/usr/lib/python3/dist-packages/PIL/IcoImagePlugin.py	125	14	0	89%
/usr/lib/python3/dist-packages/PIL/Image.py	1183	642	0	46%
/usr/lib/python3/dist-packages/PIL/ImageChops.py	65	46	0	29%
/usr/lib/python3/dist-packages/PIL/ImageColor.py	51	46	0	10%
/usr/lib/python3/dist-packages/PIL/ImageDraw.py	246	184	0	25%
/usr/lib/python3/dist-packages/PIL/ImageFile.py	334	186	0	44%
/usr/lib/python3/dist-packages/PIL/ImageFont.py	162	128	0	21%
/usr/lib/python3/dist-packages/PIL/ImageMode.py	24	1	0	96%
/usr/lib/python3/dist-packages/PIL/ImageOps.py	240	212	0	12%
/usr/lib/python3/dist-packages/PIL/ImagePalette.py	117	80	0	32%
/usr/lib/python3/dist-packages/PIL/ImageSequence.py	23	17	0	26%
/usr/lib/python3/dist-packages/PIL/JpegImagePlugin.py	402	189	0	53%
/usr/lib/python3/dist-packages/PIL/JpegPresets.py	1	0	0	100%
/usr/lib/python3/dist-packages/PIL/PaletteFile.py	24	19	0	21%
/usr/lib/python3/dist-packages/PIL/PngImagePlugin.py	479	234	0	51%
/usr/lib/python3/dist-packages/PIL/PpmImagePlugin.py	86	71	0	17%
/usr/lib/python3/dist-packages/PIL/TiffImagePlugin.py	951	452	0	52%
/usr/lib/python3/dist-packages/PIL/TiffTags.py	40	0	0	100%
/usr/lib/python3/dist-packages/PIL/_init_.py	5	0	0	100%
/usr/lib/python3/dist-packages/PIL/_binary.py	31	8	0	74%
/usr/lib/python3/dist-packages/PIL/_util.py	19	6	0	68%
/usr/lib/python3/dist-packages/PIL/_version.py	1	0	0	100%
/usr/lib/python3/dist-packages/PyPDF2/_init_.py	5	0	0	100%
/usr/lib/python3/dist-packages/PyPDF2/_version.py	1	0	0	100%
/usr/lib/python3/dist-packages/PyPDF2/filters.py	264	232	0	12%
/usr/lib/python3/dist-packages/PyPDF2/generic.py	737	335	0	55%
/usr/lib/python3/dist-packages/PyPDF2/merger.py	288	254	0	12%
/usr/lib/python3/dist-packages/PyPDF2/pagerange.py	58	33	0	43%
/usr/lib/python3/dist-packages/PyPDF2/pdf.py	1544	936	0	39%
/usr/lib/python3/dist-packages/PyPDF2/utils.py	165	44	0	73%
/usr/lib/python3/dist-packages/babel/_init_.py	3	0	0	100%
/usr/lib/python3/dist-packages/babel/_compat.py	51	26	0	49%
/usr/lib/python3/dist-packages/babel/core.py	290	120	0	59%

This is how the report looks like.

Conclusion

Hope this article was useful to enhance your developer experience with odoo, docker definitely changes the way developers work since we can have open source projects running with few lines of code in a docker-compose file and even better, we can customize images to add things missing like libraries to allow us running tests.

Pycharm’s remote interpreters is an amazing feature that lets us take full advantage of docker images to debug or run scripts.



Open in app

Get started

Get the Medium app

