Deployment with Docker and Poetry

PyBay2023

Cristian Heredia

caheredia/pybay2023
cristianheredia

This is not a deep dive into:

This is not a deep dive into:

- Docker, docker-compose

This is not a deep dive into:

- Docker, docker-compose
- Poetry

This is not a deep dive into:

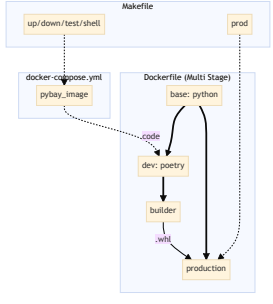
- Docker, docker-compose
- Poetry
- venv

This is not a deep dive into:

- Docker, docker-compose
- Poetry
- venv
- Makefile

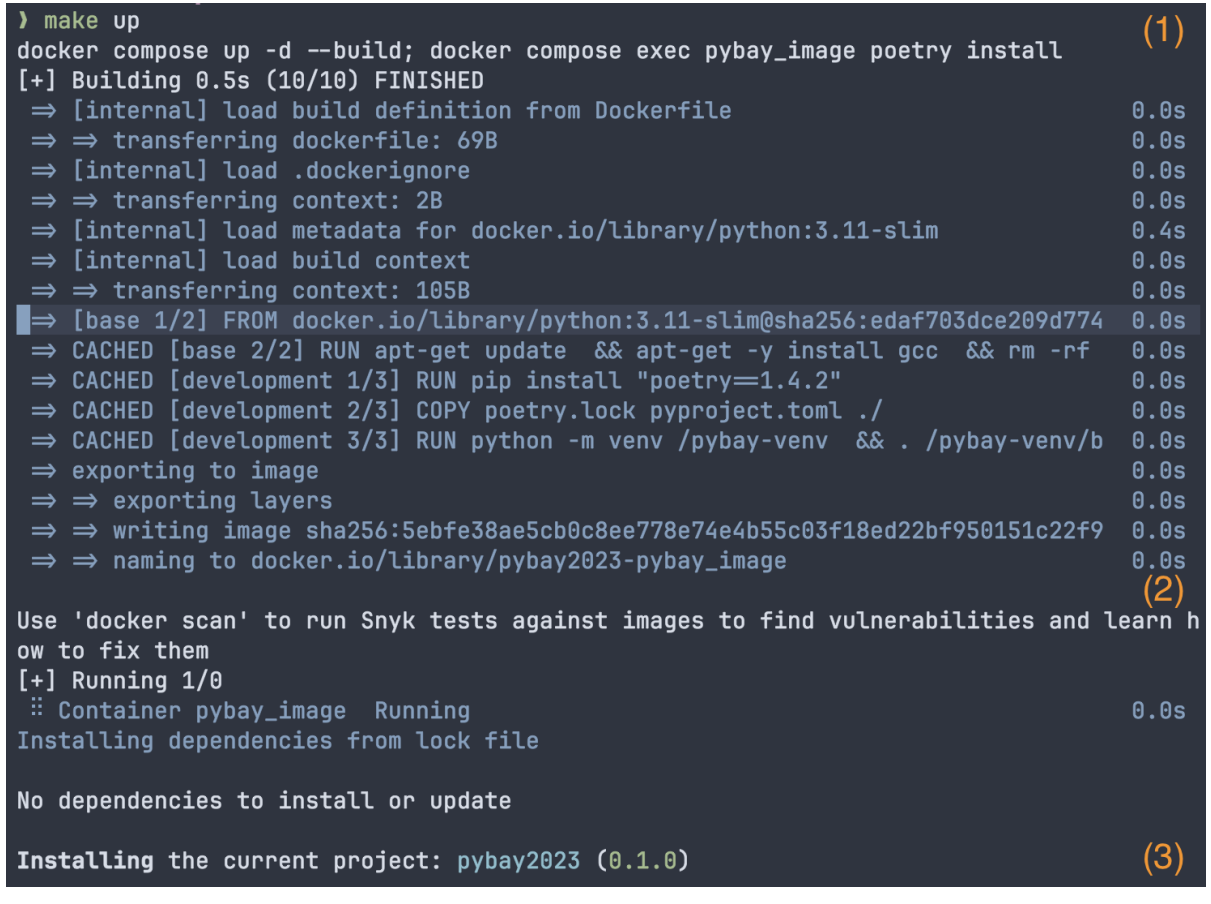
A working example on how to build a containerized Python app with Poetry and Docker


caheredia/pybay2023


Makefile
up/down/test/shell
prod
docker-compose.yml
pybay_image
Dockerfile (Multi Stage)
base: python
.code
dev: poetry
builder
.whl
production

Makefile

```
up: ## Docker compose up
    docker compose up -d --build; docker compose exec pybay_image poetry install
down: ## Docker compose down
    docker compose down --remove-orphans
shell: ## Shell into container
    docker compose exec pybay_image bash
test: ## Run static checks and tests
    docker compose exec pybay_image flake8 src/ tests/;
    docker compose exec pybay_image isort src/ tests/;
    docker compose exec pybay_image black src/ tests/;
    docker compose exec pybay_image python -m unittest discover tests
prod: ## Build production image
    docker build --file Dockerfile --target production -t pybay2023_prod_image .
```



`make up`

1. build and bring up container
2. attach `cwd` as volume
3. install source code as editable package

docker-compose.yml

```yaml
services:
  pybay_image:
    container_name: pybay_image
    build:
      context: .
      dockerfile: Dockerfile
      target: development
    volumes:
      - .:/app
    working_dir: /app
    entrypoint: ["sleep", "infinity"]
```

Dockerfile stages: Base | Dev | Prod

```
1  # BASE
2  FROM python:3.11-slim as base
3  # install gcc
4  RUN apt-get update \
5      && apt-get -y install gcc \
6      && rm -rf /var/lib/apt/lists/*
7
8  # DEVELOPMENT
9  FROM base as development
10 ENV \
11     PIP_NO_CACHE_DIR=off \
12     PIP_DISABLE_PIP_VERSION_CHECK=on \
13     PYTHONDONTWRITEBYTECODE=1 \
14     VIRTUAL_ENV=/pybay-venv
15 ENV \
16     POETRY_VIRTUALENVS_CREATE=false \
```

Dockerfile stages: Base | Dev | Prod

```
# BASE
FROM python:3.11-slim as base
# install gcc
RUN apt-get update \
    && apt-get -y install gcc \
    && rm -rf /var/lib/apt/lists/*

# DEVELOPMENT
FROM base as development
ENV \
    PIP_NO_CACHE_DIR=off \
    PIP_DISABLE_PIP_VERSION_CHECK=on \
    PYTHONDONTWRITEBYTECODE=1 \
    VIRTUAL_ENV=/pybay-venv
ENV \
    POETRY_VIRTUALENVS_CREATE=false \
```

Dockerfile stages: Base | Dev | Prod

```
 7
 8 # DEVELOPMENT
 9 FROM base as development
10 ENV \
11     PIP_NO_CACHE_DIR=off \
12     PIP_DISABLE_PIP_VERSION_CHECK=on \
13     PYTHONDONTWRITEBYTECODE=1 \
14     VIRTUAL_ENV=/pybay-venv
15 ENV \
16     POETRY_VIRTUALENVS_CREATE=false \
17     POETRY_VIRTUALENVS_IN_PROJECT=false \
18     POETRY_NO_INTERACTION=1 \
19     POETRY_VERSION=1.4.2
20
21 # install poetry
15 ENV \
16     POETRY_VIRTUALENVS_CREATE=false \
```

Dockerfile stages: Base | Dev | Prod

```
30 RUN python -m venv $VIRTUAL_ENV \
31     && . $VIRTUAL_ENV/bin/activate \
32     && poetry install
33
34 # BUILDER
35 FROM development as builder
36 WORKDIR /app
37 COPY . .
38 RUN poetry install --without dev
39 # export build
40 RUN poetry build --format wheel
41
42 # PRODUCTION
43 FROM base as production
44 WORKDIR /app
15 ENV \
16     POETRY_VIRTUALENVS_CREATE=false \
```

Dockerfile stages: Base | Dev | Prod

```
33
34 # BUILDER
35 FROM development as builder
36 WORKDIR /app
37 COPY . .
38 RUN poetry install --without dev
39 # export build
40 RUN poetry build --format wheel
41
42 # PRODUCTION
43 FROM base as production
44 WORKDIR /app
45 COPY --from=builder /app/dist/*.whl ./
46 RUN pip install ./*.whl
```

```
15 ENV \
16     POETRY_VIRTUALENVS_CREATE=false \
```

Dockerfile stages: Base | Dev | Prod

```
33
34 # BUILDER
35 FROM development as builder
36 WORKDIR /app
37 COPY . .
38 RUN poetry install --without dev
39 # export build
40 RUN poetry build --format wheel
41
42 # PRODUCTION
43 FROM base as production
44 WORKDIR /app
45 COPY --from=builder /app/dist/*.whl ./
46 RUN pip install ./*.whl
```

```
15 ENV \
16     POETRY_VIRTUALENVS_CREATE=false \
```

Dockerfile stages: Base | Dev | Prod

```
33
34 # BUILDER
35 FROM development as builder
36 WORKDIR /app
37 COPY . .
38 RUN poetry install --without dev
39 # export build
40 RUN poetry build --format wheel
41
42 # PRODUCTION
43 FROM base as production
44 WORKDIR /app
45 COPY --from=builder /app/dist/*.whl ./
46 RUN pip install ./*.whl
```

```
15 ENV \
16     POETRY_VIRTUALENVS_CREATE=false \
```

Dockerfile stages: Base | Dev | Prod

```
33
34 # BUILDER
35 FROM development as builder
36 WORKDIR /app
37 COPY . .
38 RUN poetry install --without dev
39 # export build
40 RUN poetry build --format wheel
41
42 # PRODUCTION
43 FROM base as production
44 WORKDIR /app
45 COPY --from=builder /app/dist/*.whl ./
46 RUN pip install ./*.whl
```

```
15 ENV \
16     POETRY_VIRTUALENVS_CREATE=false \
```

pyproject.toml

```toml
[tool.poetry]
name = "pybay2023"
version = "0.1.0"
description = "A demo for packaging python with Poetry, Docker, and Docker Co
authors = ["Cristian <tech+cristian@simplelogin.dedyn.io>"]
readme = "README.md"

[tool.poetry.dependencies] # main dependency group
python = "^3.9"
requests = "2.31.0"

[tool.poetry.group.dev.dependencies]
black = "23.9.1"
isort = "5.12.0"
flake8 = "6.1.0"
mypy = "1.5.1"
```

pyproject.toml

```
9  python = "^3.9"
10 requests = "2.31.0"
11
12 [tool.poetry.group.dev.dependencies]
13 black = "23.9.1"
14 isort = "5.12.0"
15 flake8 = "6.1.0"
16 mypy = "1.5.1"
17 types-requests = "2.31.0.3"
18 boto3 = "1.28.55"
19 jupyter = "1.0.0"
20 numpy = "1.25.0"
21 pandas = "2.0.0"
22
23 [tool.poetry.scripts]
24 nybav2023 = "nybav2023.service.cli:fetch_date"
```


```
15 flake8 = "6.1.0"
16 mypy = "1.5.1"
```

pyproject.toml

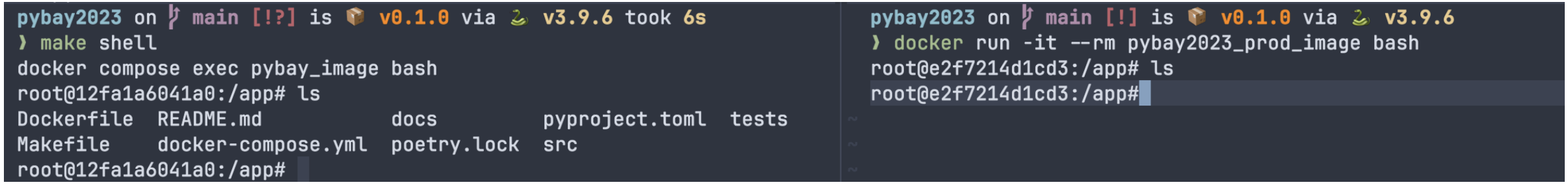
```
15 flake8 = "6.1.0"
16 mypy = "1.5.1"
17 types-requests = "2.31.0.3"
18 boto3 = "1.28.55"
19 jupyter = "1.0.0"
20 numpy = "1.25.0"
21 pandas = "2.0.0"
22
23 [tool.poetry.scripts]
24 pybay2023 = "pybay2023.service.cli:fetch_date"
25
26 [build-system]
27 requires = ["poetry-core"]
28 build-backend = "poetry.core.masonry.api"
```


```
15 flake8 = "6.1.0"
16 mypy = "1.5.1"
```

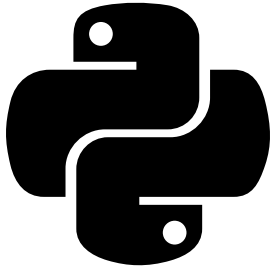
```python
from typing import Optional

import requests
from requests import Response

from pybay2023.domain.world_time.world_time import TIME_URL


def fetch_date(response: Optional[Response] = None) -> str:
    if response is None:
        response = requests.get(TIME_URL)

    response.raise_for_status()
    date_str = response.json()["dateTime"]

    print(f"PyBay2023, \nThe time is: {date_str}\n")
```

It's not just about shipping smaller production images

```
❯ docker image ls
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
pybay2023_prod_image        latest    7d6782977a99   6 seconds ago    373MB
pybay2023-pybay_image       latest    50b6e8449790   19 minutes ago   955MB
```



- smaller attack surface
- smaller production image, faster deployment
- repeatable build across multiple platforms



Good Luck!

References

- https://python-poetry.org/docs/managing-dependencies/
- https://pip.pypa.io/en/stable/cli/pip_wheel/
- https://docs.docker.com/build/building/multi-stage/
- https://pragprog.com/titles/dmpython/intuitive-python/
- https://bmaingret.github.io/blog/2021-11-15-Docker-and-Poetry