DOCKER AND PYTHON

Making them play nicely and securely for Data Science and Machine Learning

TANIA ALLARD, PHD

Sr. Developer Advocate @Microsoft.

ixek | https://bit.ly/europython-ml-docker

@ixek

@trallard

trallard.dev

THESE SLIDES

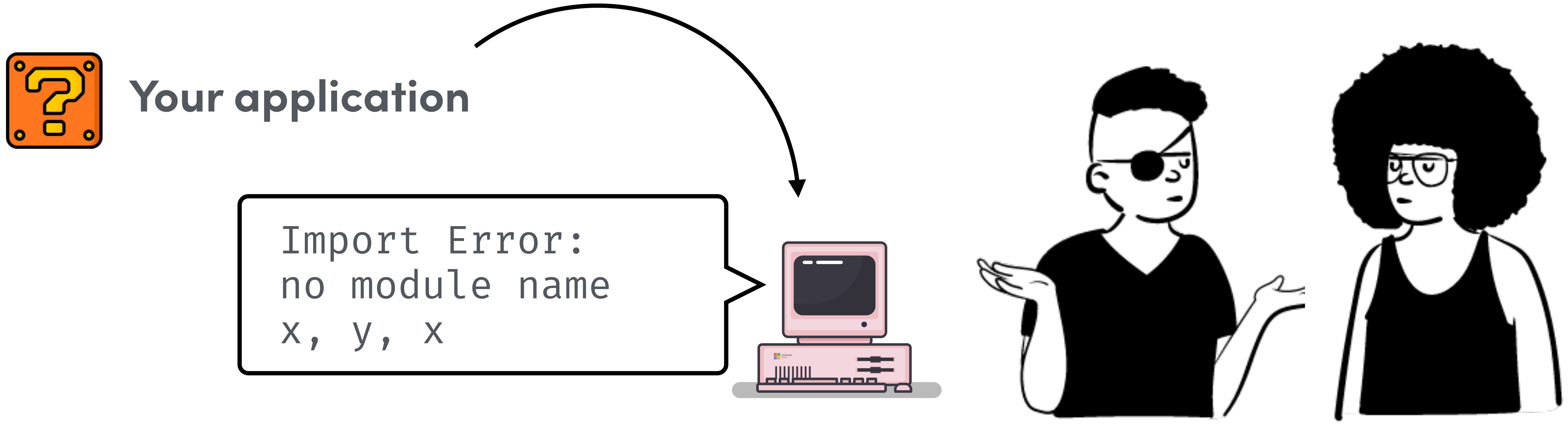
https://bit.ly/europython-ml-docker

WHAT YOU'LL LEARN TODAY

- Why using Docker?
- Docker for Data Science and Machine Learning
- Security and performance
- Do not reinvent the wheel, automate
- Tips and trick to use Docker

WHY DOCKER?

DEV LIFE WITHOUT DOCKER OR CONTAINERS



How are your users or colleagues meant to know what dependencies they need?

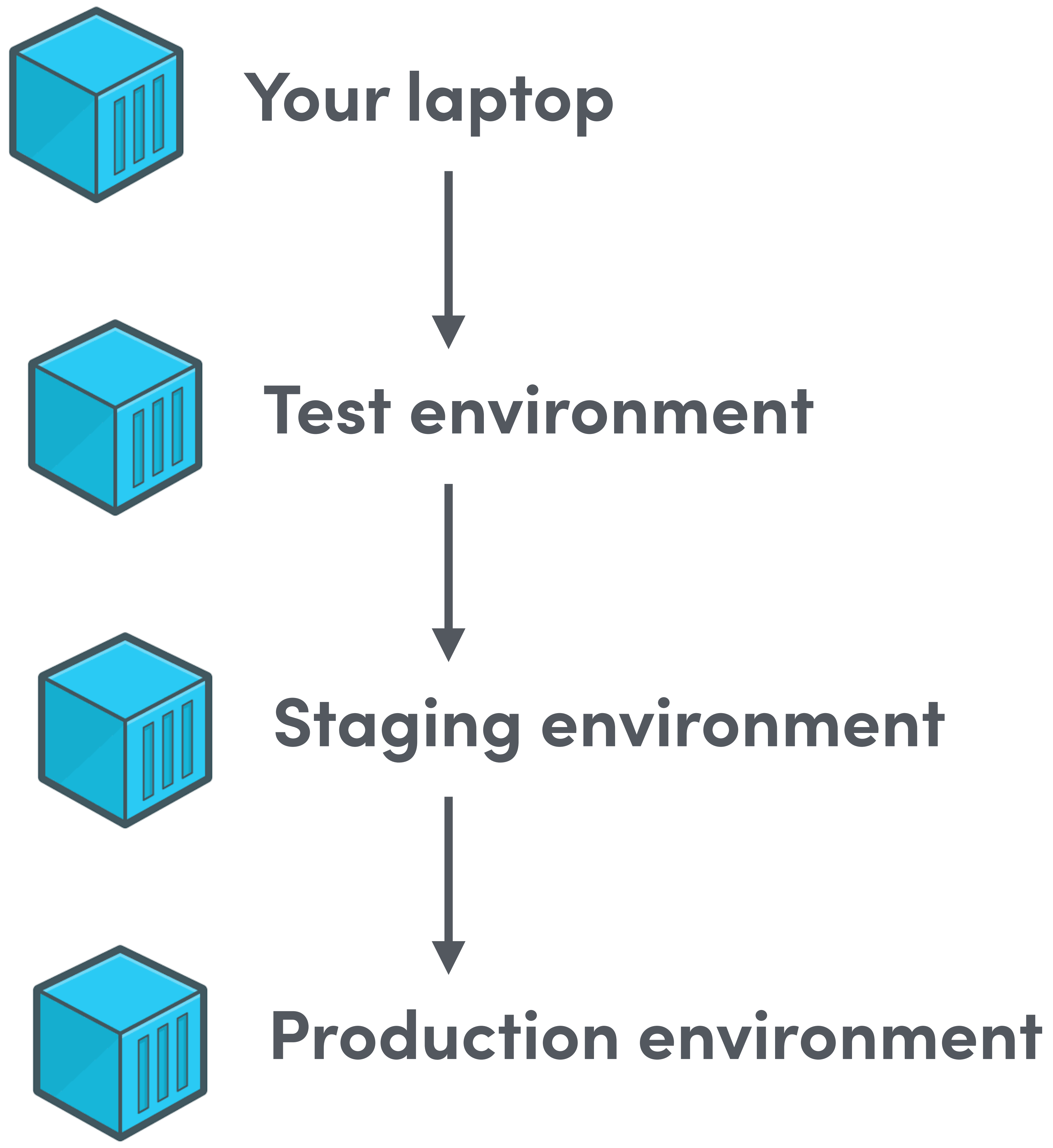
WHAT IS DOCKER?

A tool that helps you to create, deploy and run your applications or projects by using containers.

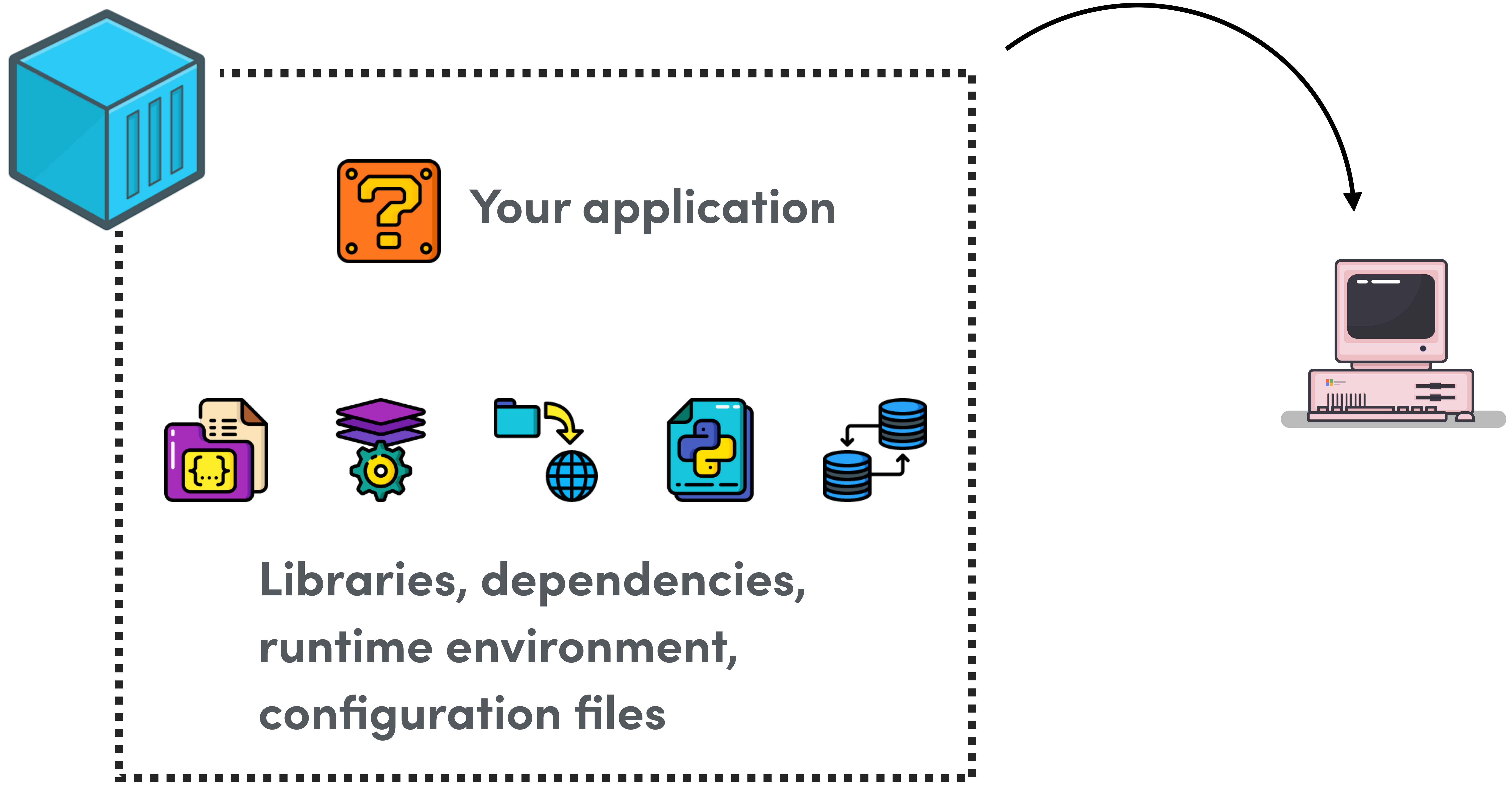
This is a container

HOW DO CONTAINERS HELP ME?

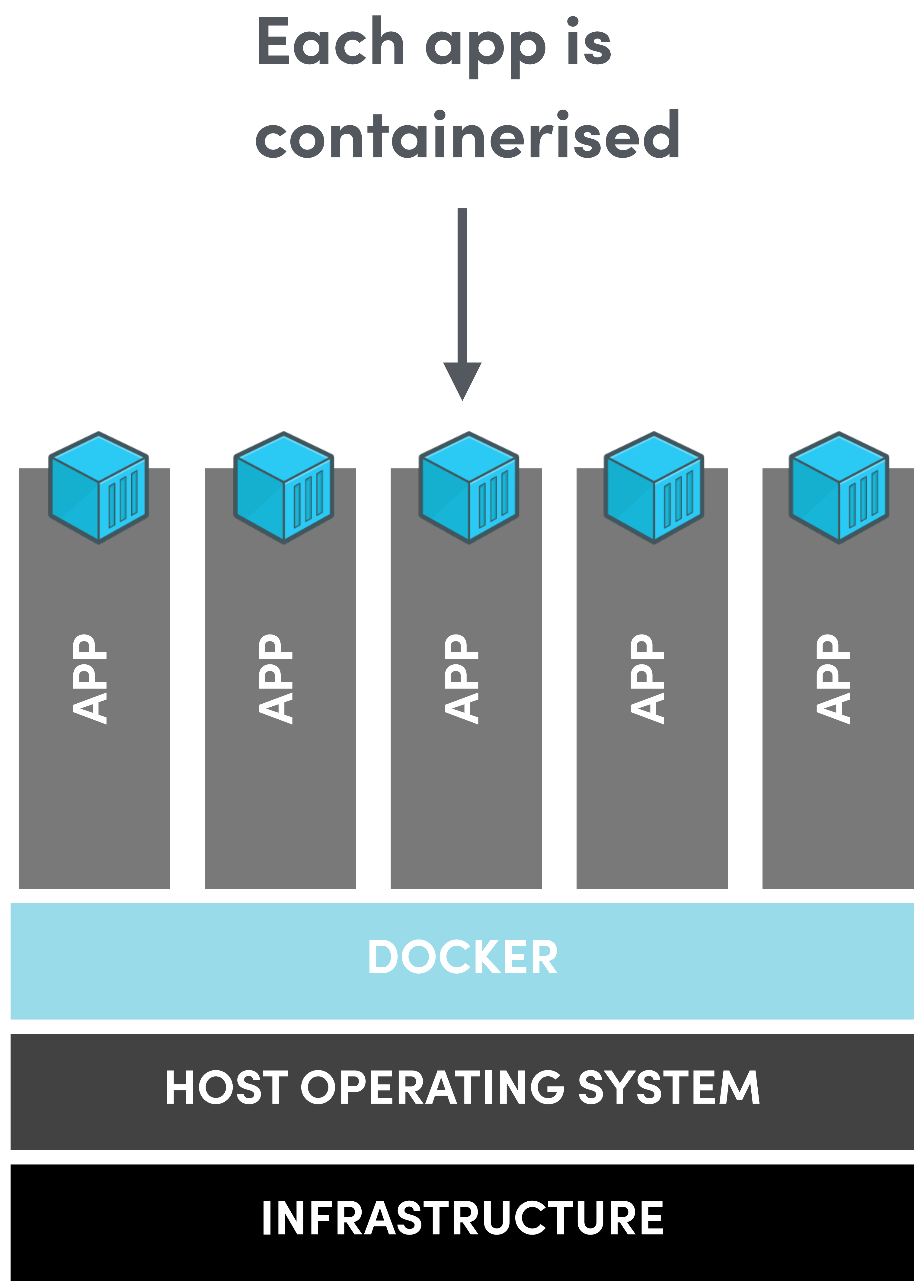
They provide a solution to the problem of how to get software to run reliably when moved from one computing environment to another



DEV LIFE WITH CONTAINERS

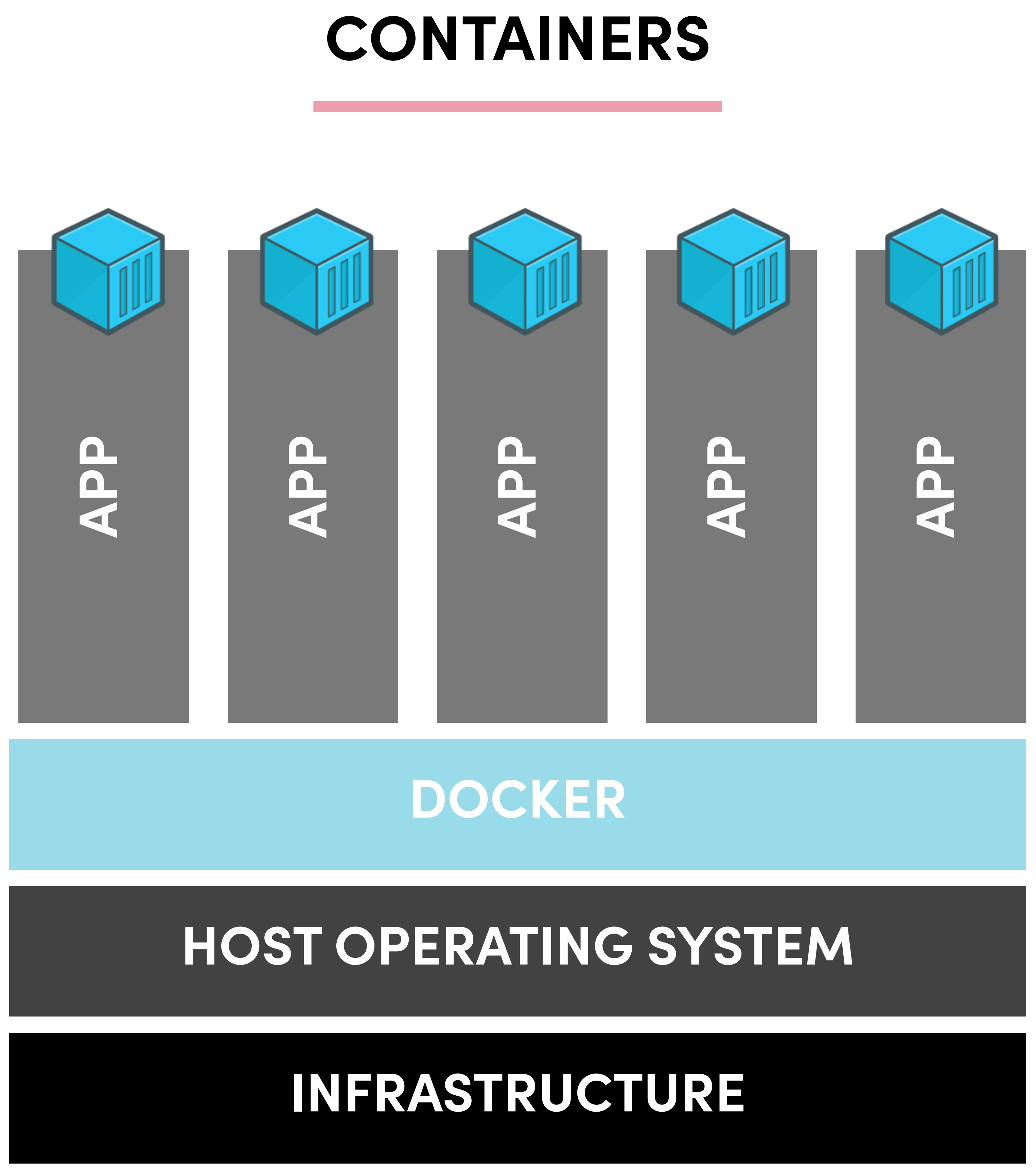


THAT SOUNDS A LOT LIKE A VIRTUAL MACHINE



At the app level:
Each runs as an isolated process

THAT SOUNDS A LOT LIKE A VIRTUAL MACHINE



Full OS + app + binaries + libraries

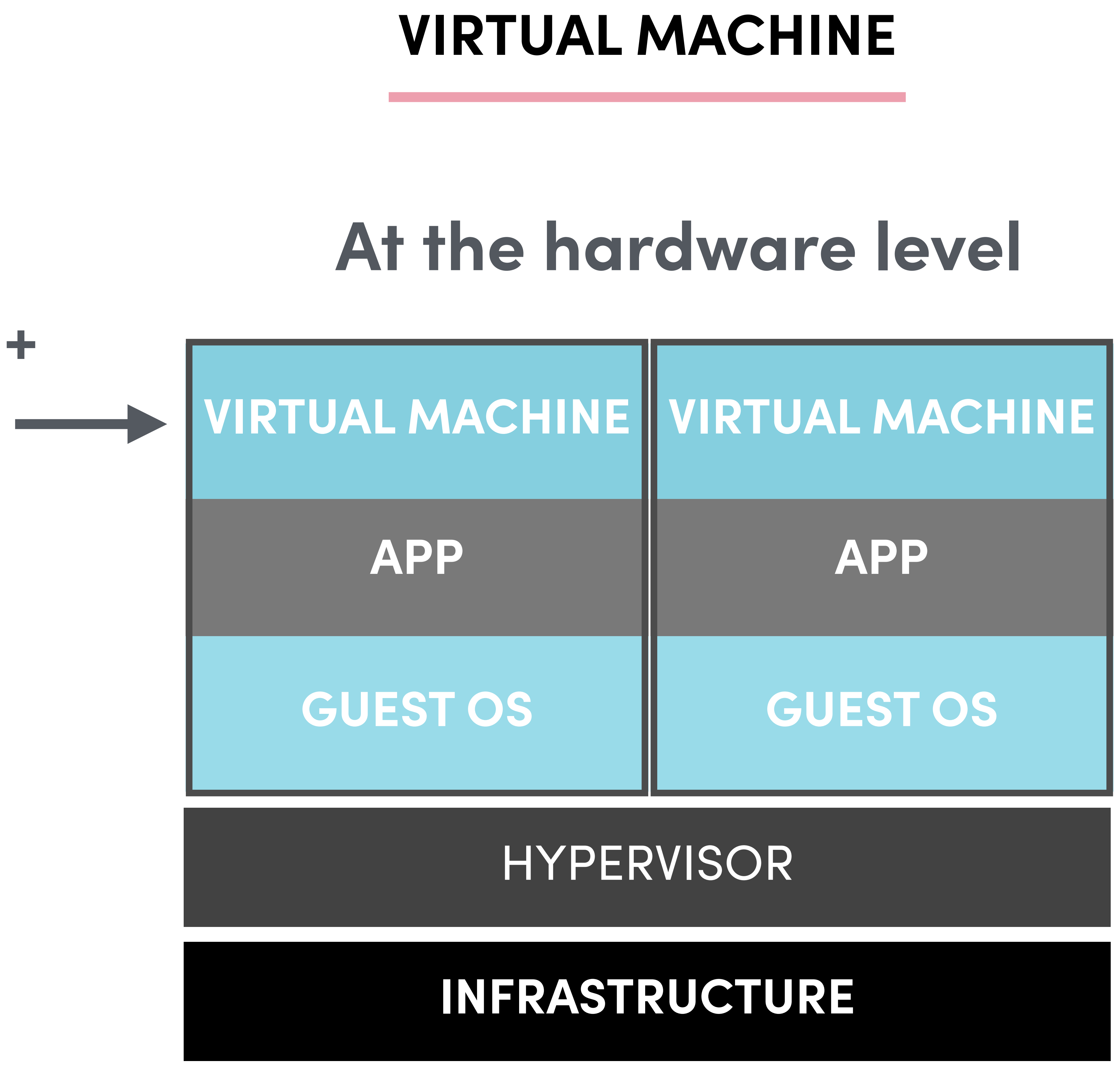
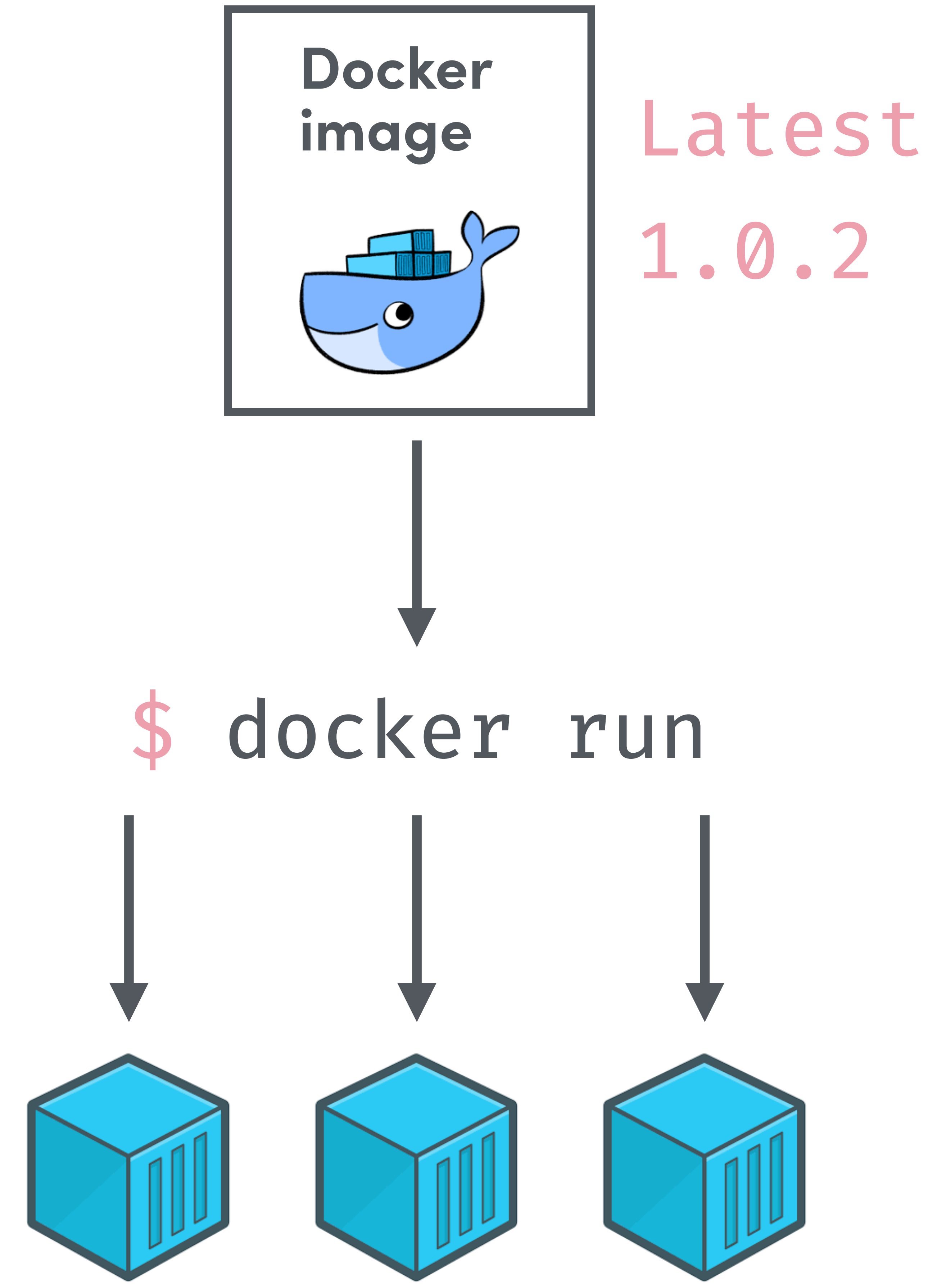


IMAGE VS CONTAINER

- Image: archive with all the data needed to run the app
- When you run an image it creates a container

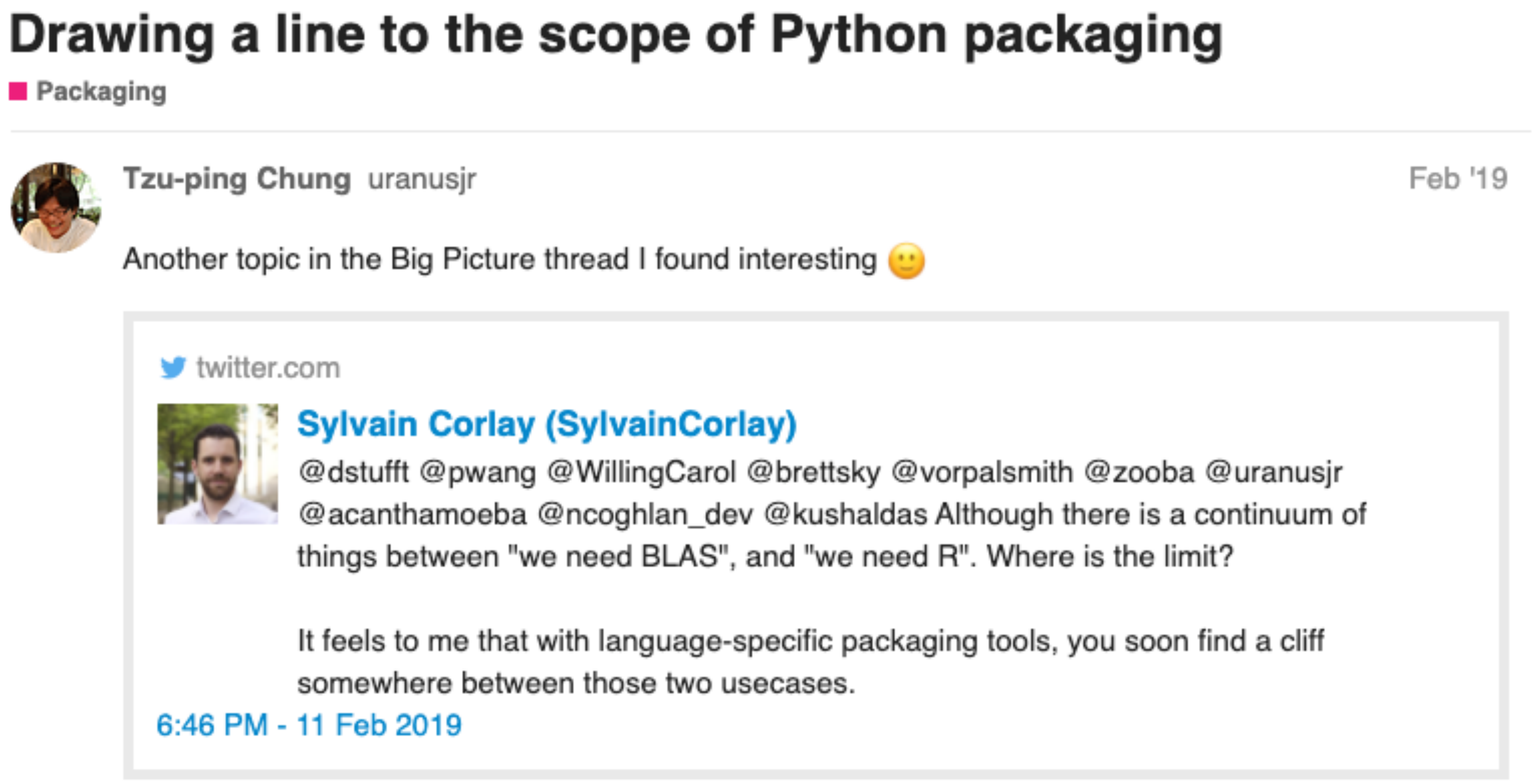


COMMON PAIN POINTS IN DS AND ML

- Complex setups / dependencies
- Reliance on data / databases
- Fast evolving projects (iterative R&D process)
- Docker is complex and can take a lot of time to upskill
- Are containers secure enough for my data / model /algorithm?

DOCKER FOR DATA SCIENCE AND MACHINE LEARNING

HOW IS IT DIFFERENT FROM WEB APPS FOR EXAMPLE?


Drawing a line to the scope of Python packaging

Packaging

Tzu-ping Chung uranusjr Feb '19

Another topic in the Big Picture thread I found interesting 🙂

twitter.com

Sylvain Corlay (SylvainCorlay)

@dstufft @pwang @WillingCarol @brettsky @vorpalsmith @zooba @uranusjr @acanthamoeba @ncoghlan_dev @kushaldas Although there is a continuum of things between "we need BLAS", and "we need R". Where is the limit?

It feels to me that with language-specific packaging tools, you soon find a cliff somewhere between those two usecases.

6:46 PM - 11 Feb 2019

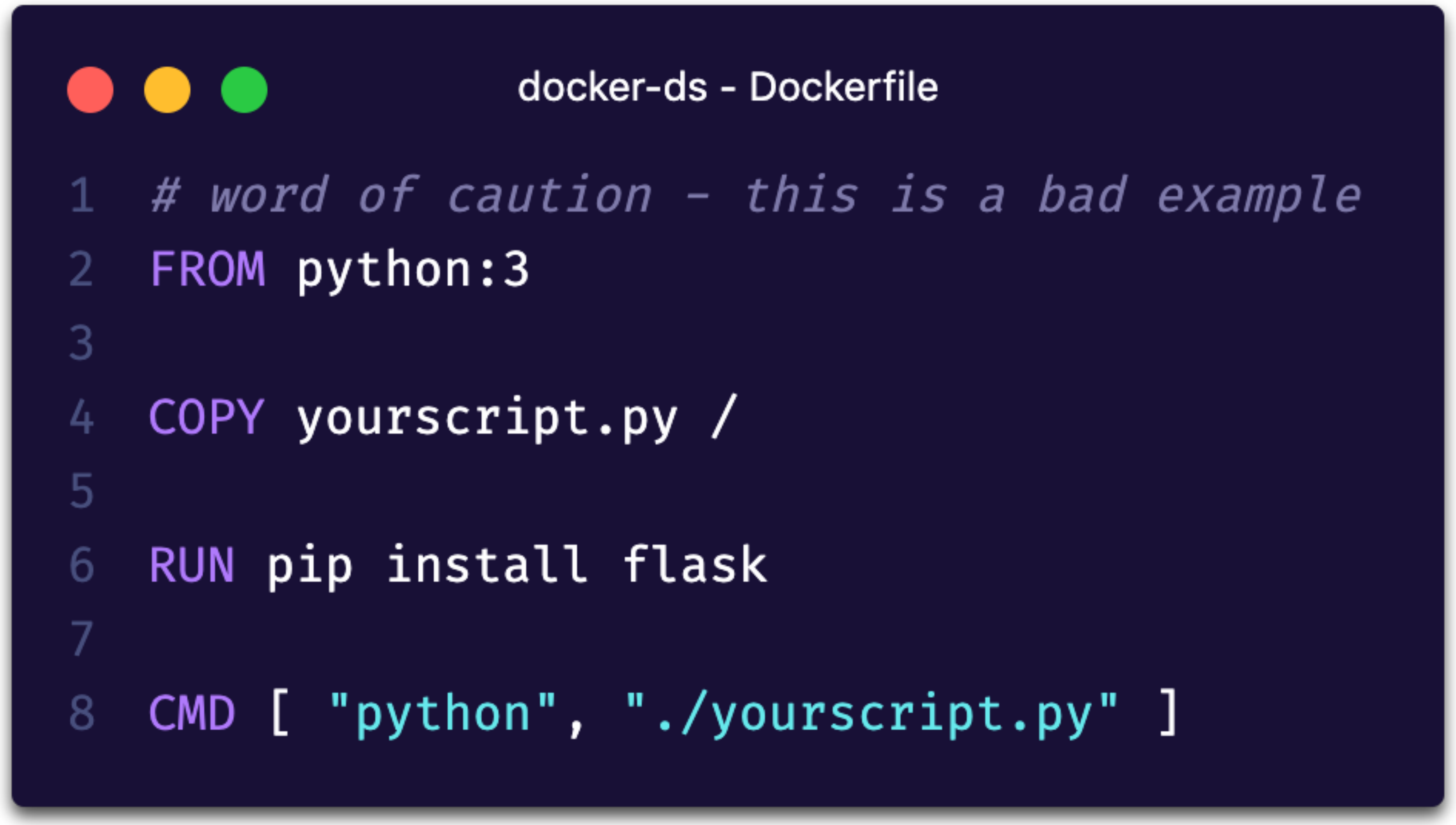
https://twitter.com/dstufft/status/1095164069802397696

HOW IS IT DIFFERENT FROM WEB APPS FOR EXAMPLE?

- Not every deliverable is an app
- Not every deliverable is a model either
- Heavily relies on data
- Mixture of wheels and compiled packages
- Security access levels - for data and software
- Mixture of stakeholders: data scientists, software engineers, ML engineers

BUILDING DOCKER IMAGES

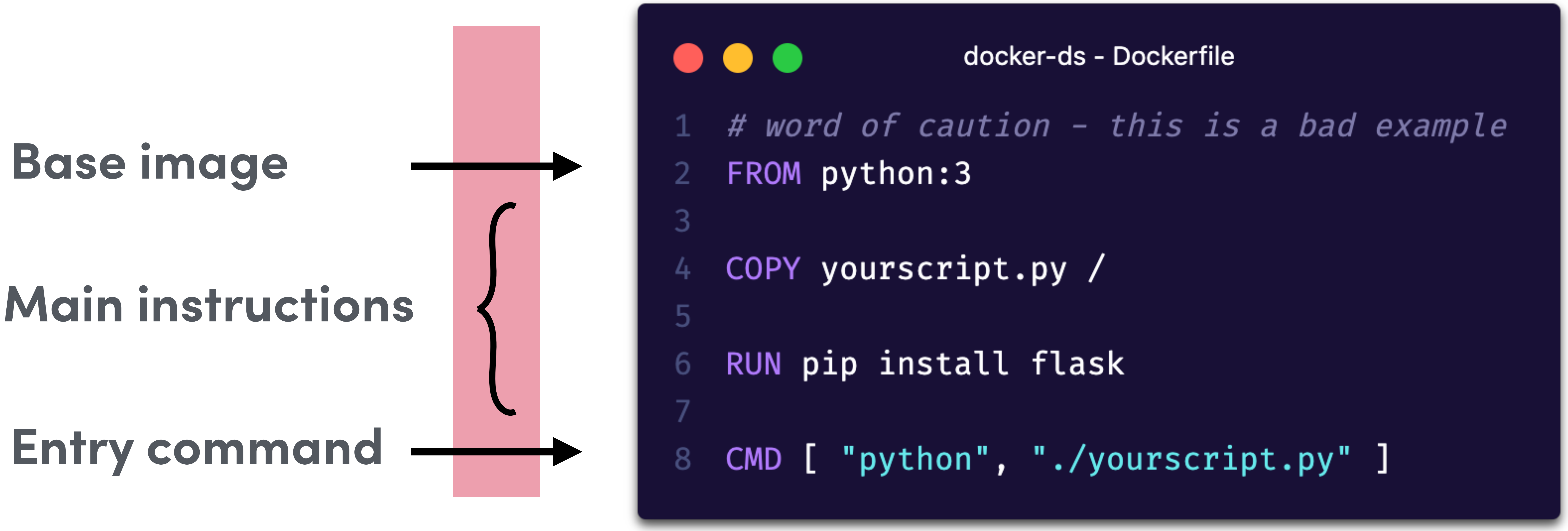
Dockerfiles are used to create Docker images by providing a set of instructions to install software, configure your image or copy files



docker-ds - Dockerfile

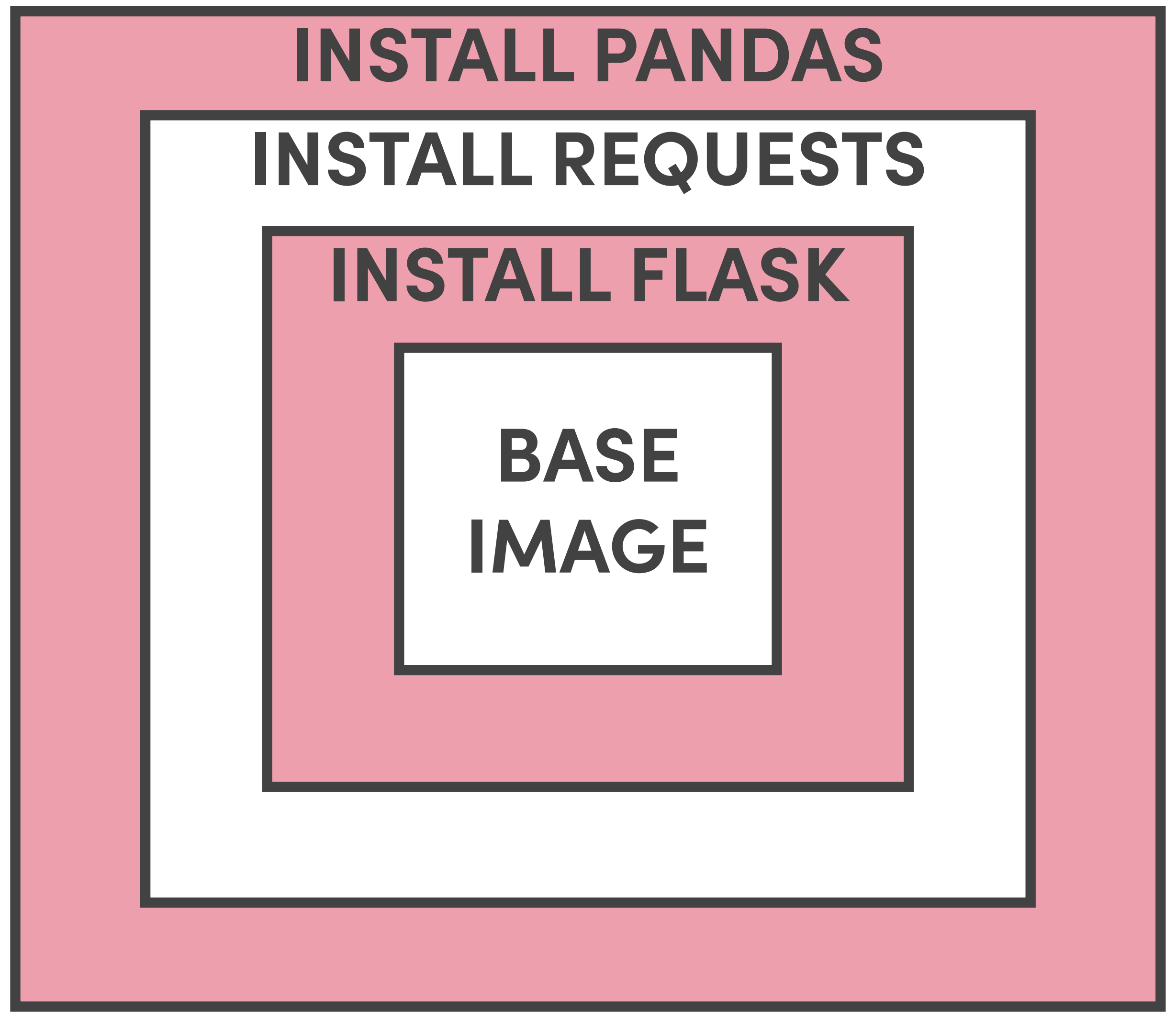
```
# word of caution - this is a bad example
FROM python:3

COPY yourscript.py /

RUN pip install flask

CMD [ "python", "./yourscript.py" ]
```

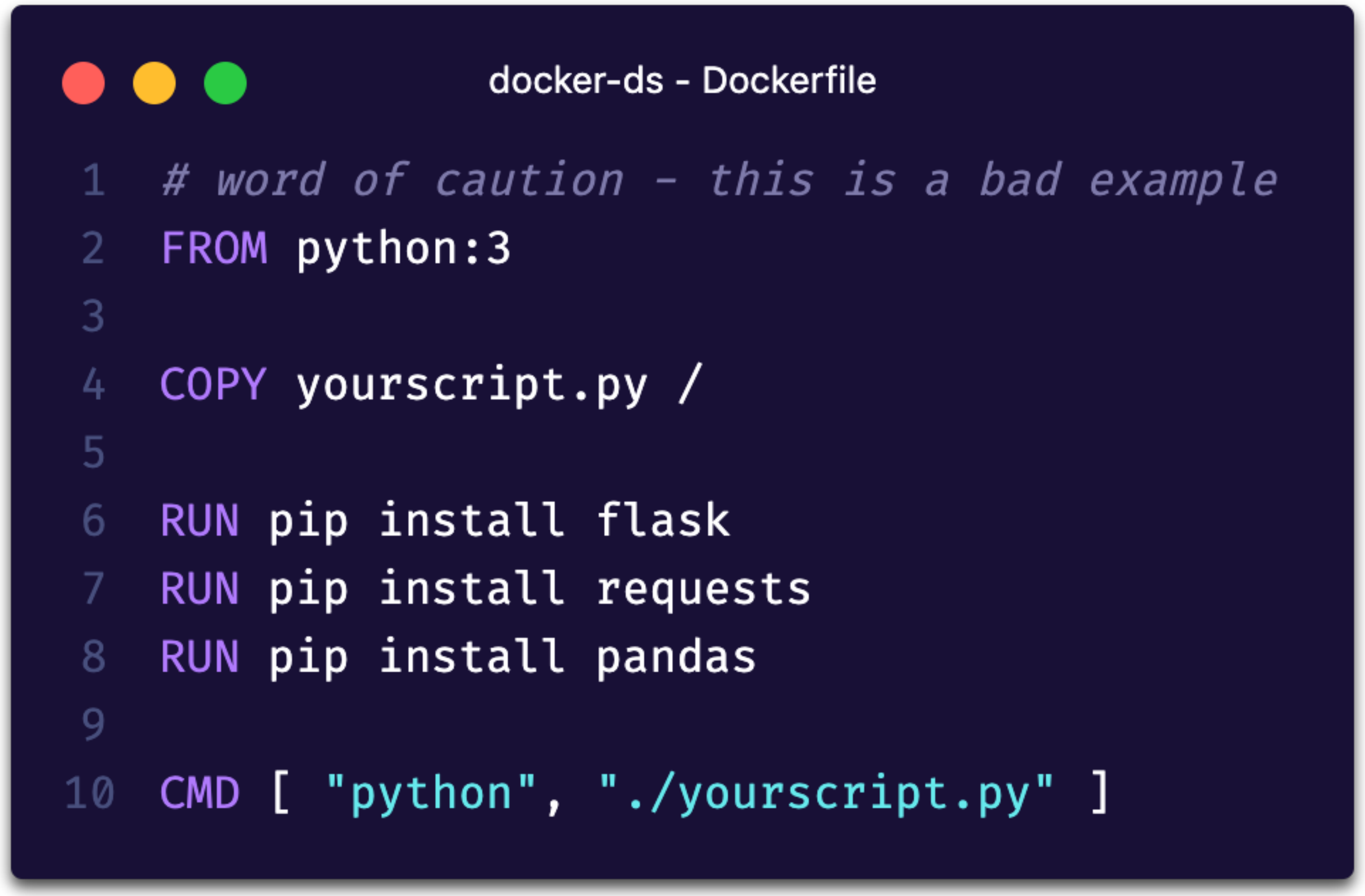
DISSECTING DOCKER IMAGES



```
# word of caution - this is a bad example
FROM python:3

COPY yourscript.py /

RUN pip install flask

CMD [ "python", "./yourscript.py" ]
```

DISSECTING DOCKER IMAGES



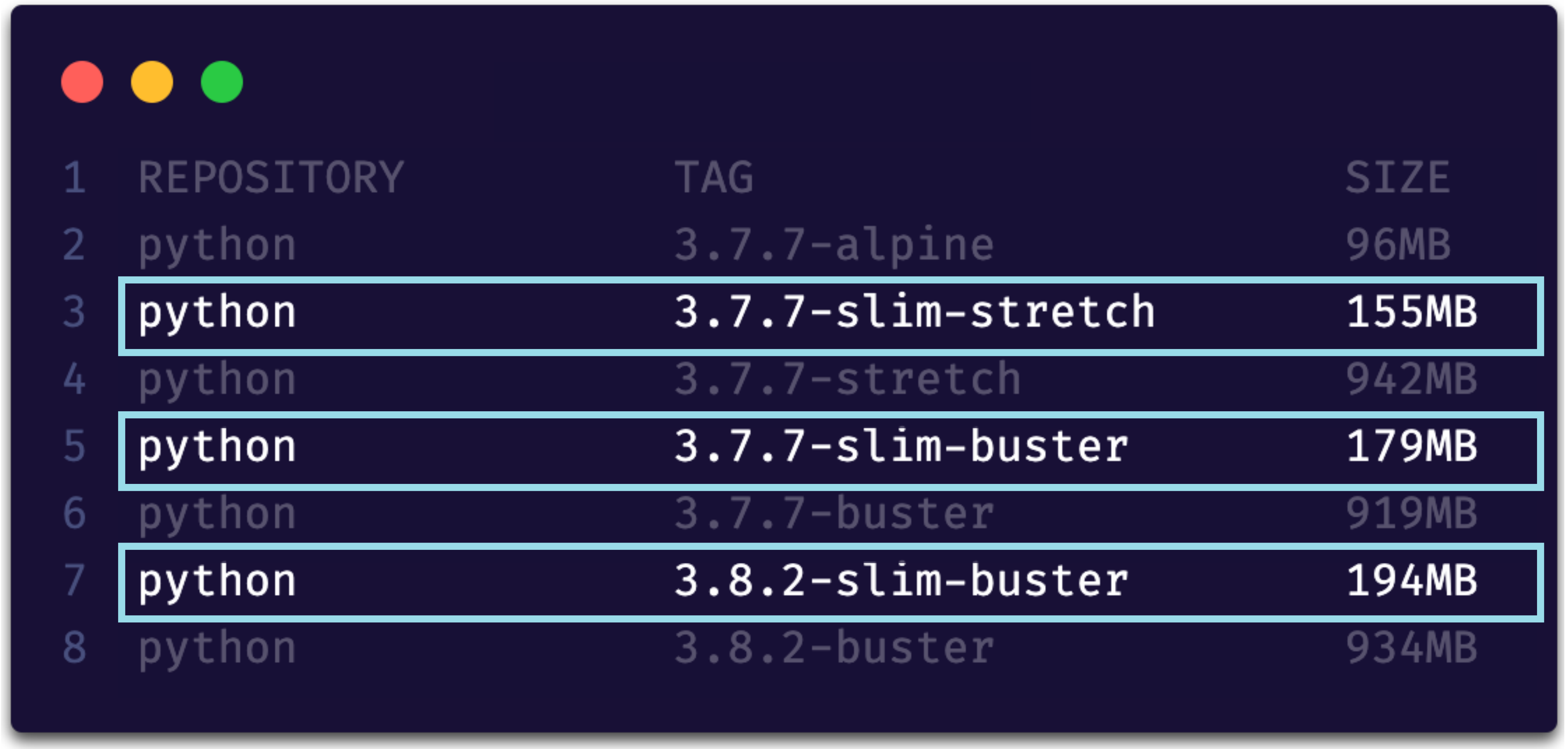


```
# word of caution - this is a bad example
FROM python:3

COPY yourscript.py /

RUN pip install flask
RUN pip install requests
RUN pip install pandas

CMD [ "python", "./yourscript.py" ]
```

Each instruction creates
A layer
(like an onion)

CHOOSING THE BEST BASE IMAGE

If building from scratch use the
official Python images



```
REPOSITORY    TAG                      SIZE
python        3.7.7-alpine             96MB
python        3.7.7-slim-stretch       155MB
python        3.7.7-stretch            942MB
python        3.7.7-slim-buster        179MB
python        3.7.7-buster             919MB
python        3.8.2-slim-buster        194MB
python        3.8.2-buster             934MB
```

https://hub.docker.com/_/python

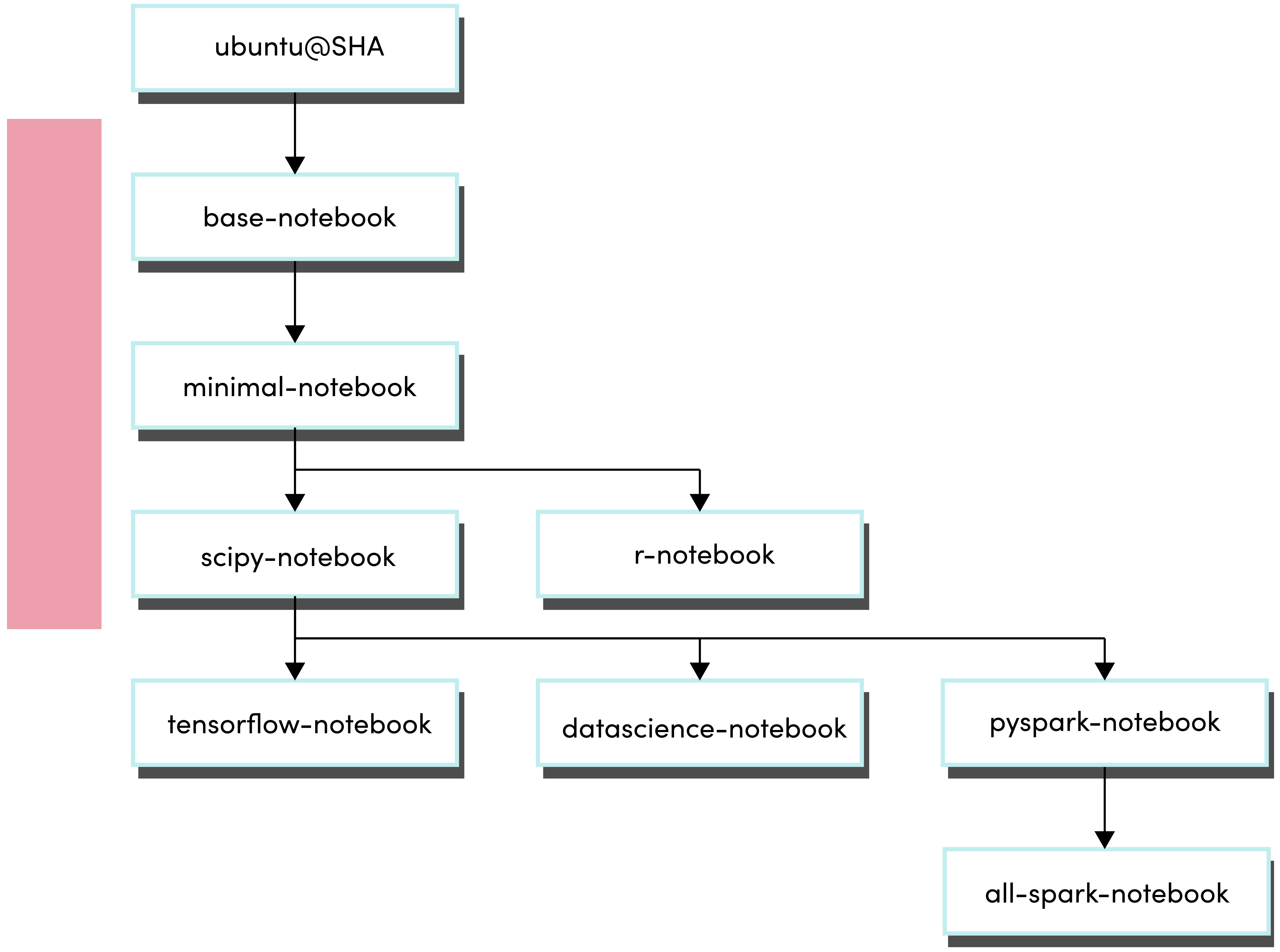
https://github.com/docker-library/docs/tree/master/python

THE JUPYTER DOCKER STACK

Need Conda, notebooks and scientific Python ecosystem?

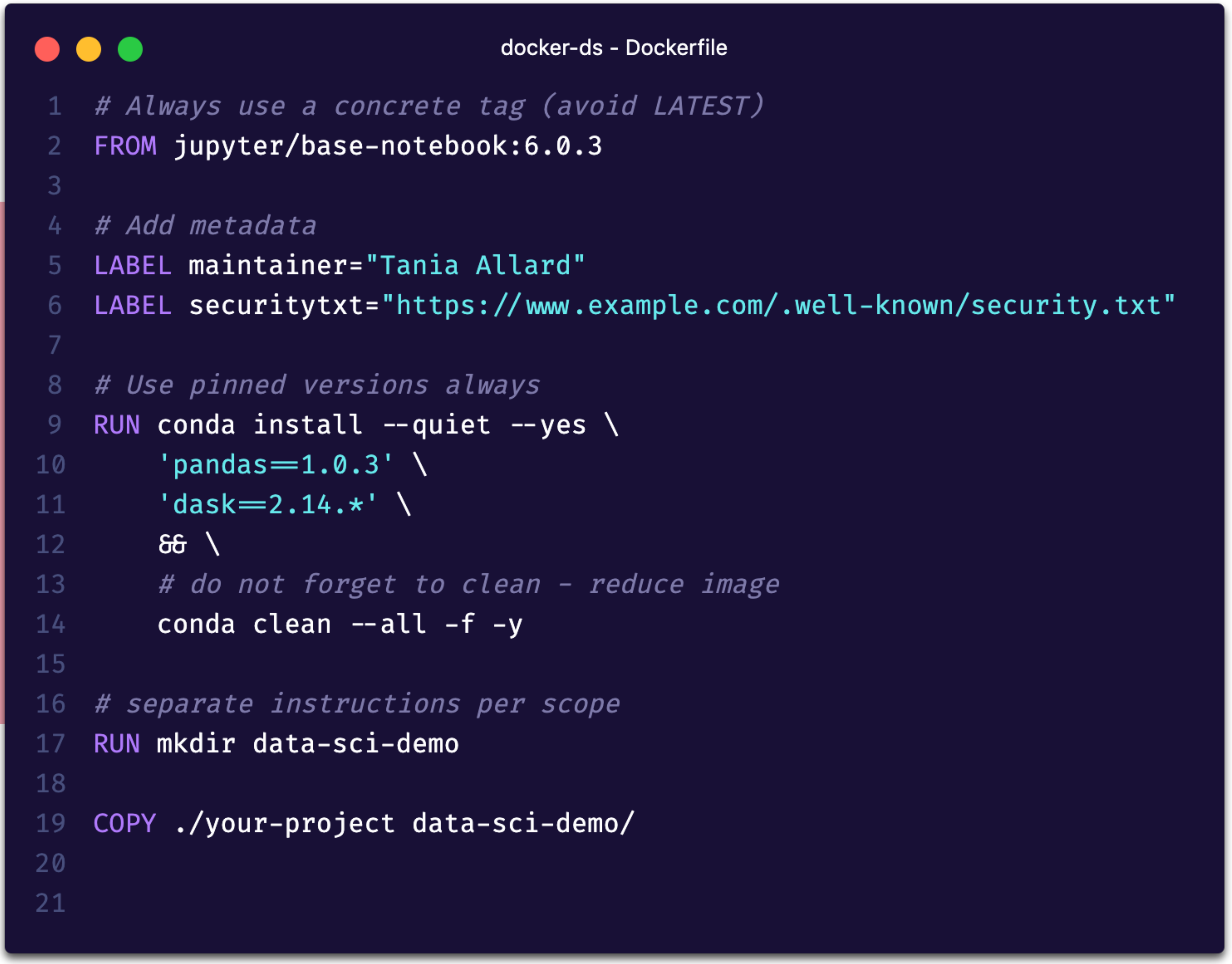
Try Jupyter Docker stacks

https://jupyter-docker-stacks.readthedocs.io/



BEST PRACTICES

- Always know what you are expecting
- Provide context with LABELS
- Split complex RUN statements and sort them
- Prefer COPY to add files



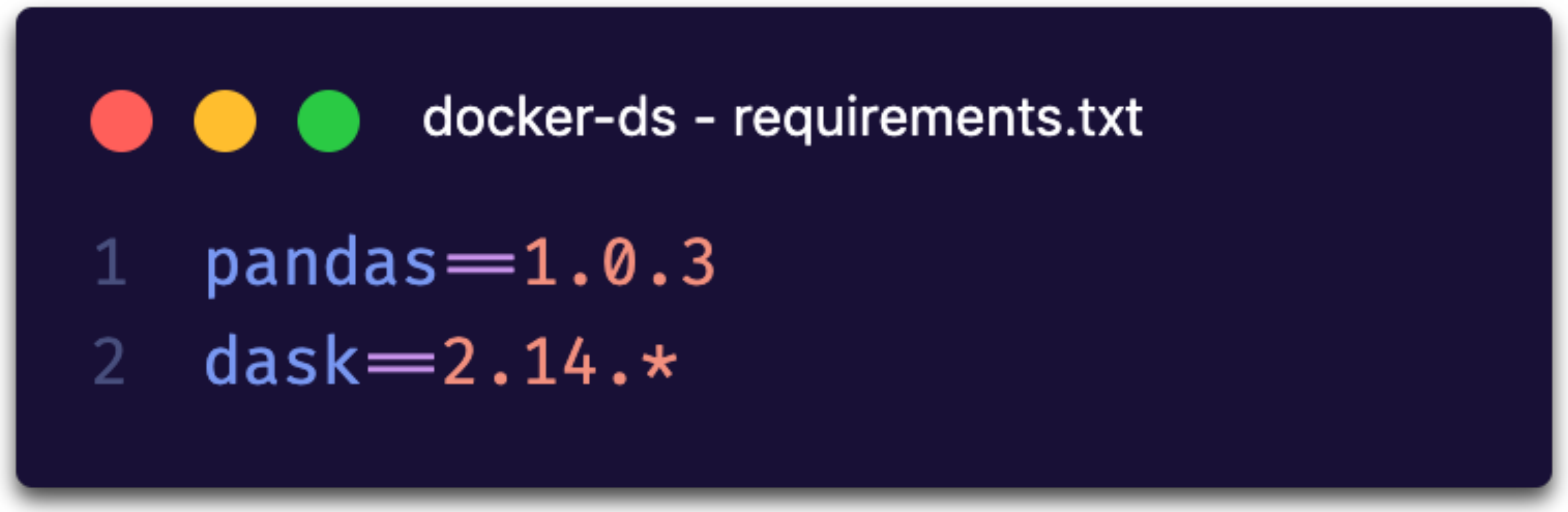
docker-ds - Dockerfile

```
# Always use a concrete tag (avoid LATEST)
FROM jupyter/base-notebook:6.0.3

# Add metadata
LABEL maintainer="Tania Allard"
LABEL securitytxt="https://www.example.com/.well-known/security.txt"

# Use pinned versions always
RUN conda install --quiet --yes \
    'pandas==1.0.3' \
    'dask==2.14.*' \
    && \
    # do not forget to clean - reduce image
    conda clean --all -f -y

# separate instructions per scope
RUN mkdir data-sci-demo

COPY ./your-project data-sci-demo/
```

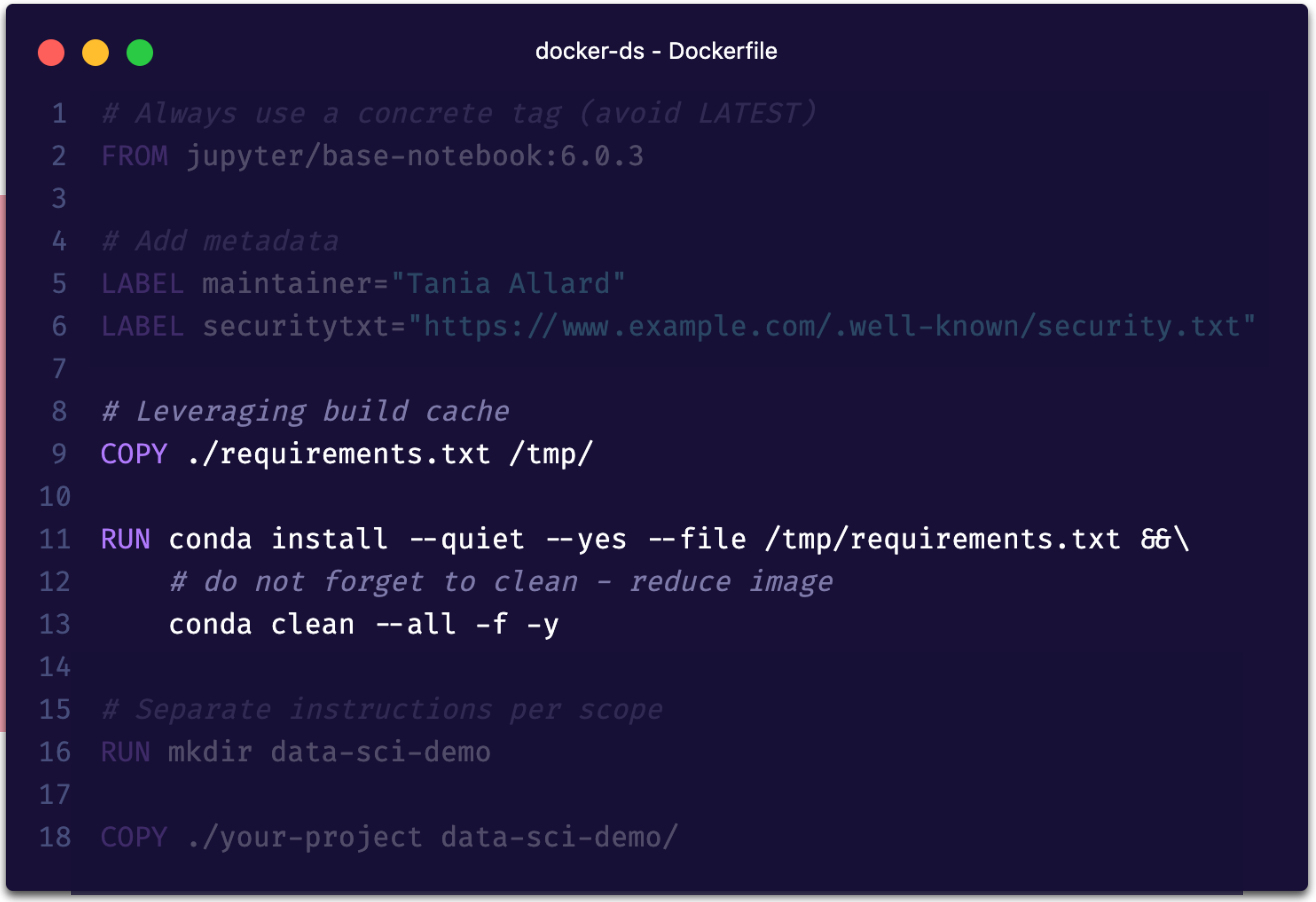
https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

SPEED UP YOUR BUILD

- Leverage build cache
- Install only necessary packages


docker-ds - requirements.txt

```
pandas==1.0.3
dask==2.14.*
```

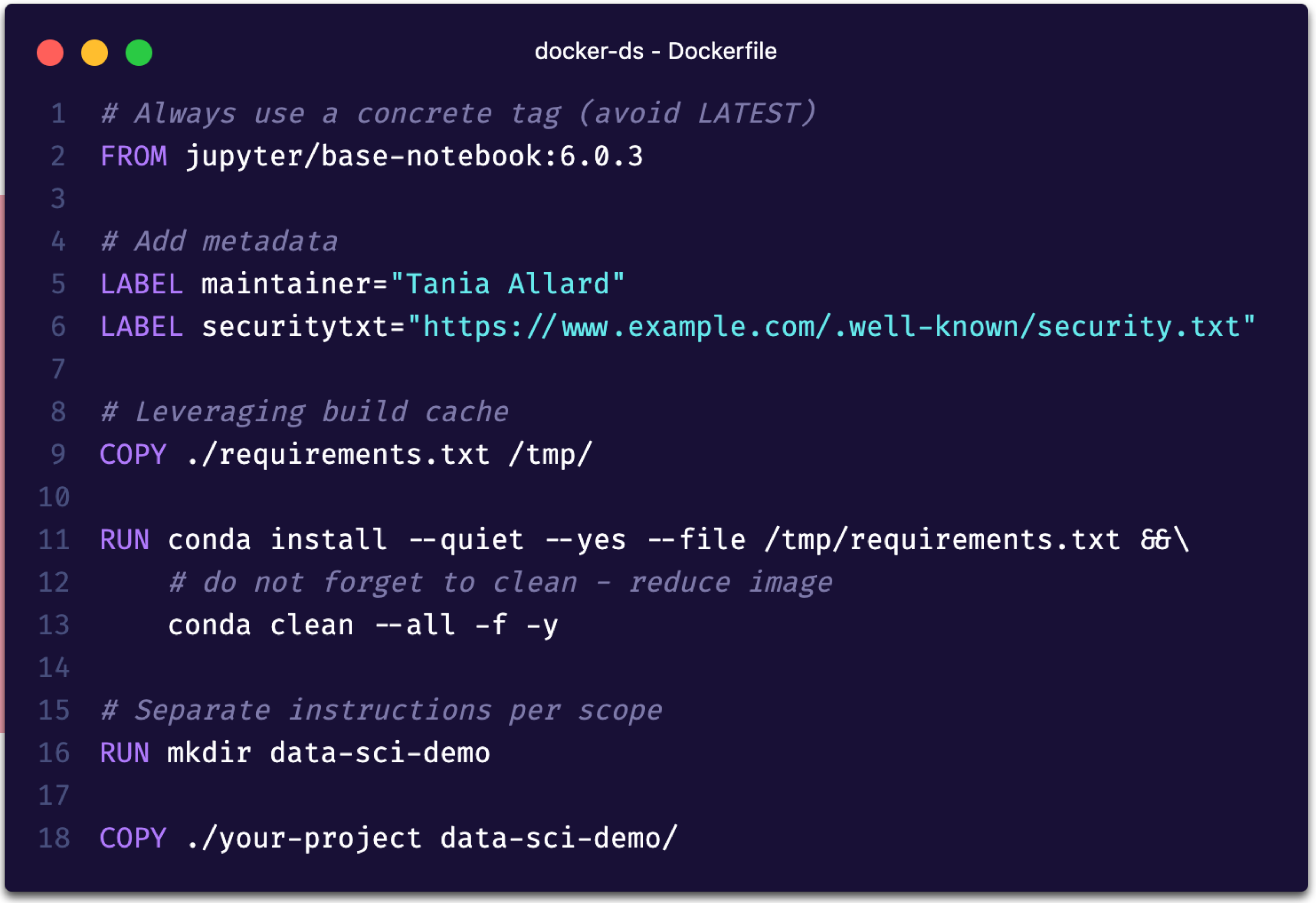

docker-ds - Dockerfile

```
# Always use a concrete tag (avoid LATEST)
FROM jupyter/base-notebook:6.0.3

# Add metadata
LABEL maintainer="Tania Allard"
LABEL securitytxt="https://www.example.com/.well-known/security.txt"

# Leveraging build cache
COPY ./requirements.txt /tmp/

RUN conda install --quiet --yes --file /tmp/requirements.txt &&\
    # do not forget to clean - reduce image
    conda clean --all -f -y

# Separate instructions per scope
RUN mkdir data-sci-demo

COPY ./your-project data-sci-demo/
```

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

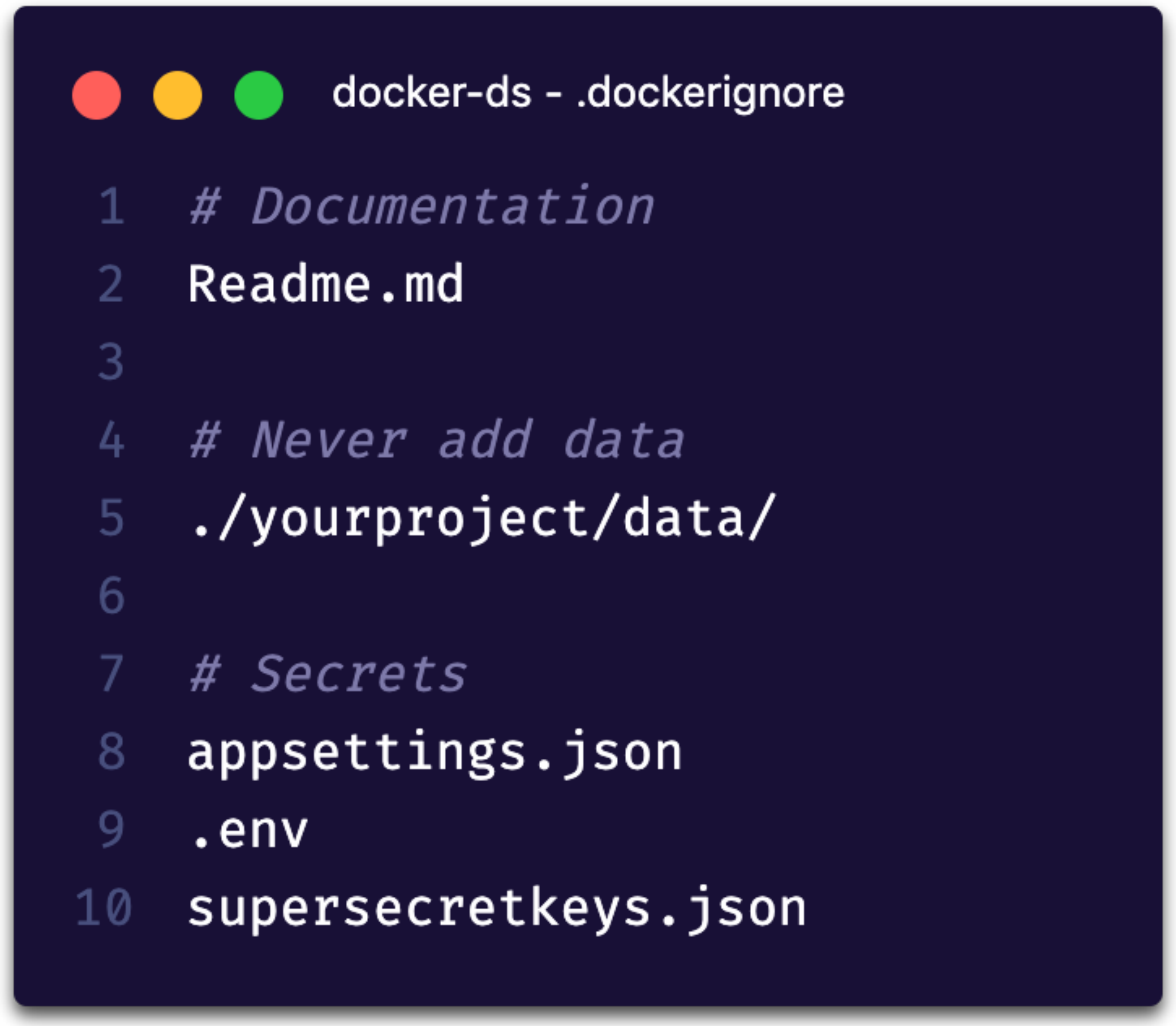
SPEED UP YOUR BUILD AND PROOF

- Leverage build cache
- Install only necessary packages
- Explicitly ignore files



docker-ds - Dockerfile

```dockerfile
# Always use a concrete tag (avoid LATEST)
FROM jupyter/base-notebook:6.0.3

# Add metadata
LABEL maintainer="Tania Allard"
LABEL securitytxt="https://www.example.com/.well-known/security.txt"

# Leveraging build cache
COPY ./requirements.txt /tmp/

RUN conda install --quiet --yes --file /tmp/requirements.txt &&\
    # do not forget to clean - reduce image
    conda clean --all -f -y

# Separate instructions per scope
RUN mkdir data-sci-demo

COPY ./your-project data-sci-demo/
```



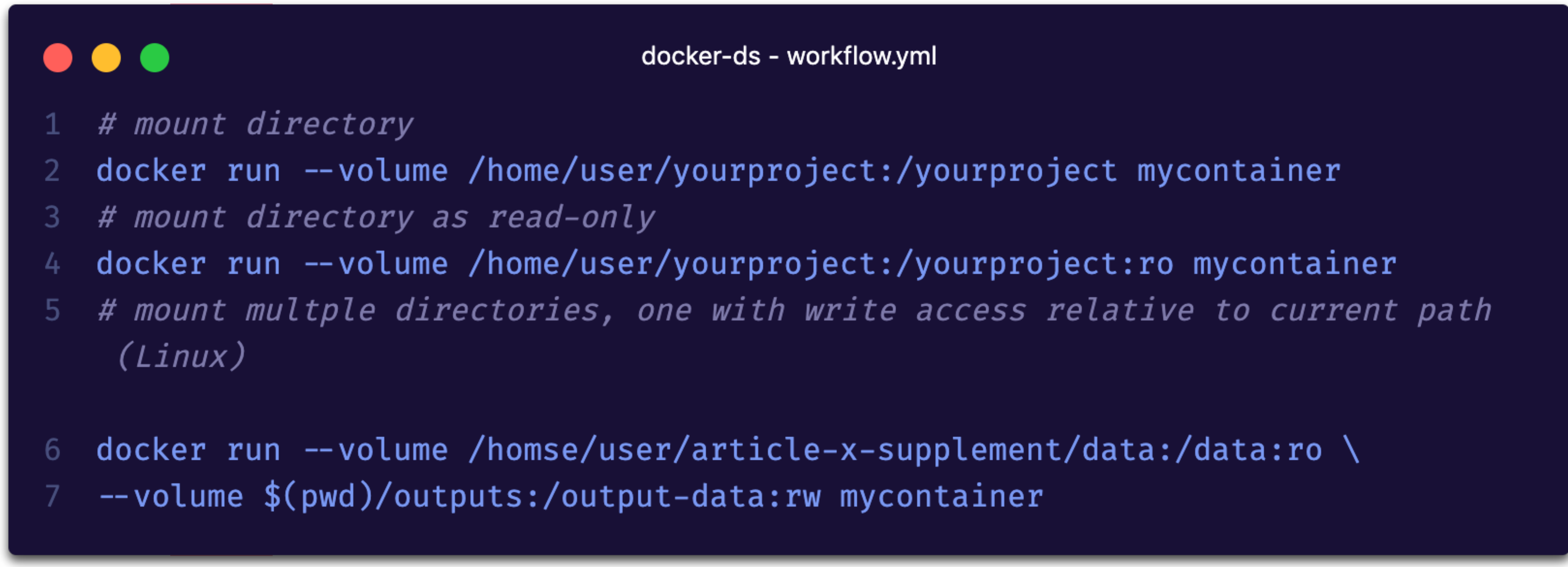
docker-ds - .dockerignore

```
# Documentation
Readme.md

# Never add data
./yourproject/data/

# Secrets
appsettings.json
.env
supersecretkeys.json
```

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

MOUNT VOLUMES TO ACCESS DATA

- You can use bind mounts to directories (unless you are using a database)
- Avoid issues by creating a non-root user



docker-ds - workflow.yml

```
# mount directory
docker run --volume /home/user/yourproject:/yourproject mycontainer
# mount directory as read-only
docker run --volume /home/user/yourproject:/yourproject:ro mycontainer
# mount multple directories, one with write access relative to current path
 (Linux)

docker run --volume /homse/user/article-x-supplement/data:/data:ro \
--volume $(pwd)/outputs:/output-data:rw mycontainer
```

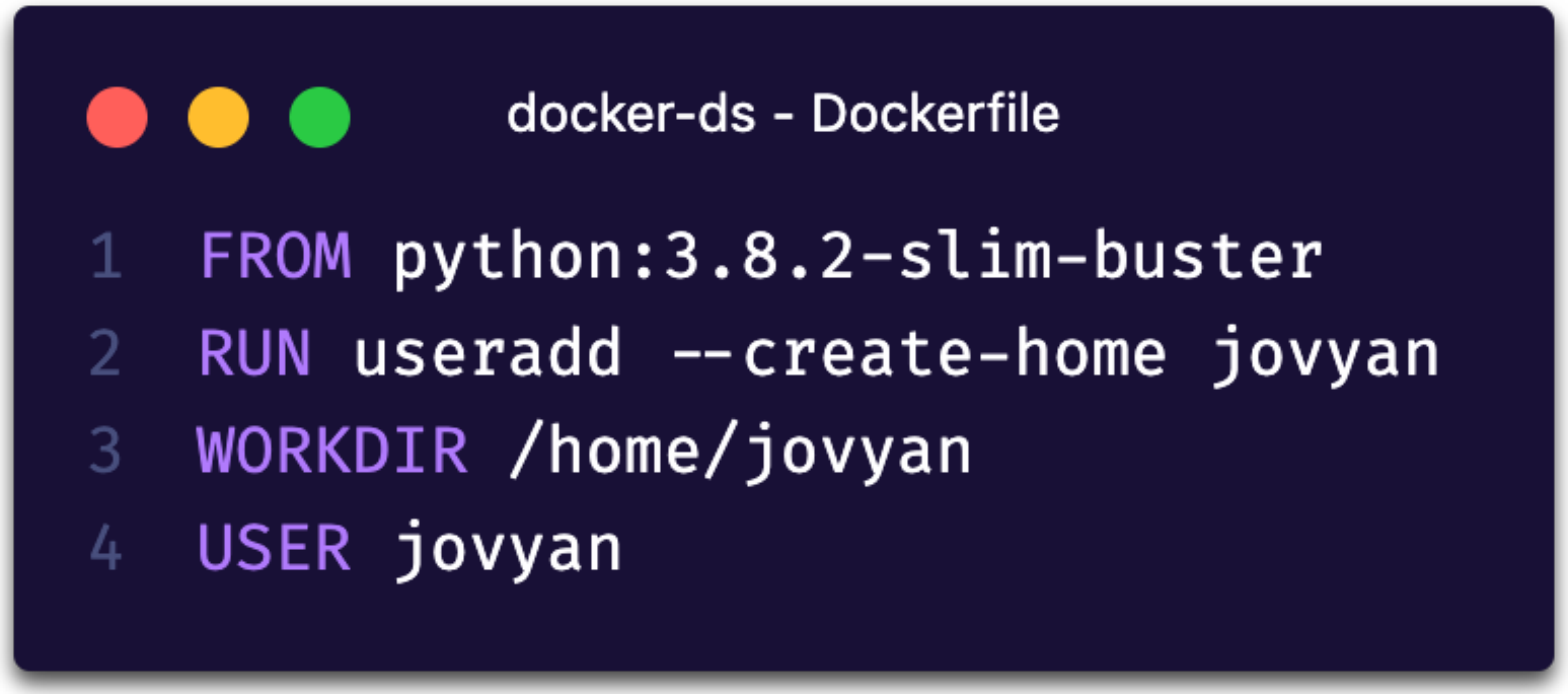
https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

SECURITY AND PERFORMANCE

MINIMISE PRIVILEGE - FAVOUR LESS PRIVILEGED USER

Lock down your container:

- Run as non-root user (Docker runs as root by default)
- Minimise capabilities



docker-ds - Dockerfile

```dockerfile
FROM python:3.8.2-slim-buster
RUN useradd --create-home jovyan
WORKDIR /home/jovyan
USER jovyan
```

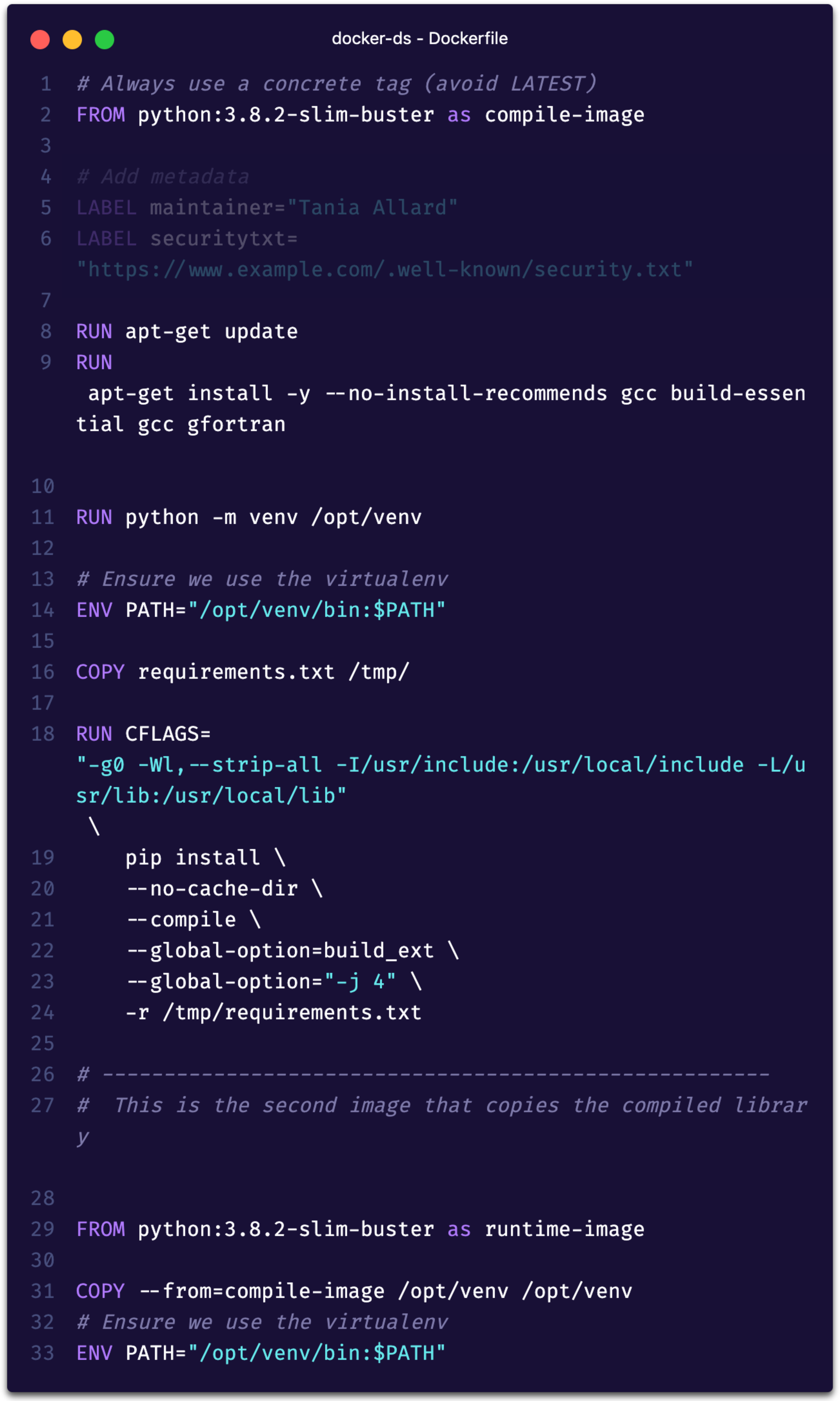
DON'T LEAK SENSITIVE INFORMATION

Remember Docker images are like onions. If you copy keys in an intermediate layer they are cached.

Keep them out of your Dockerfile.

USE MULTI STAGE BUILDS

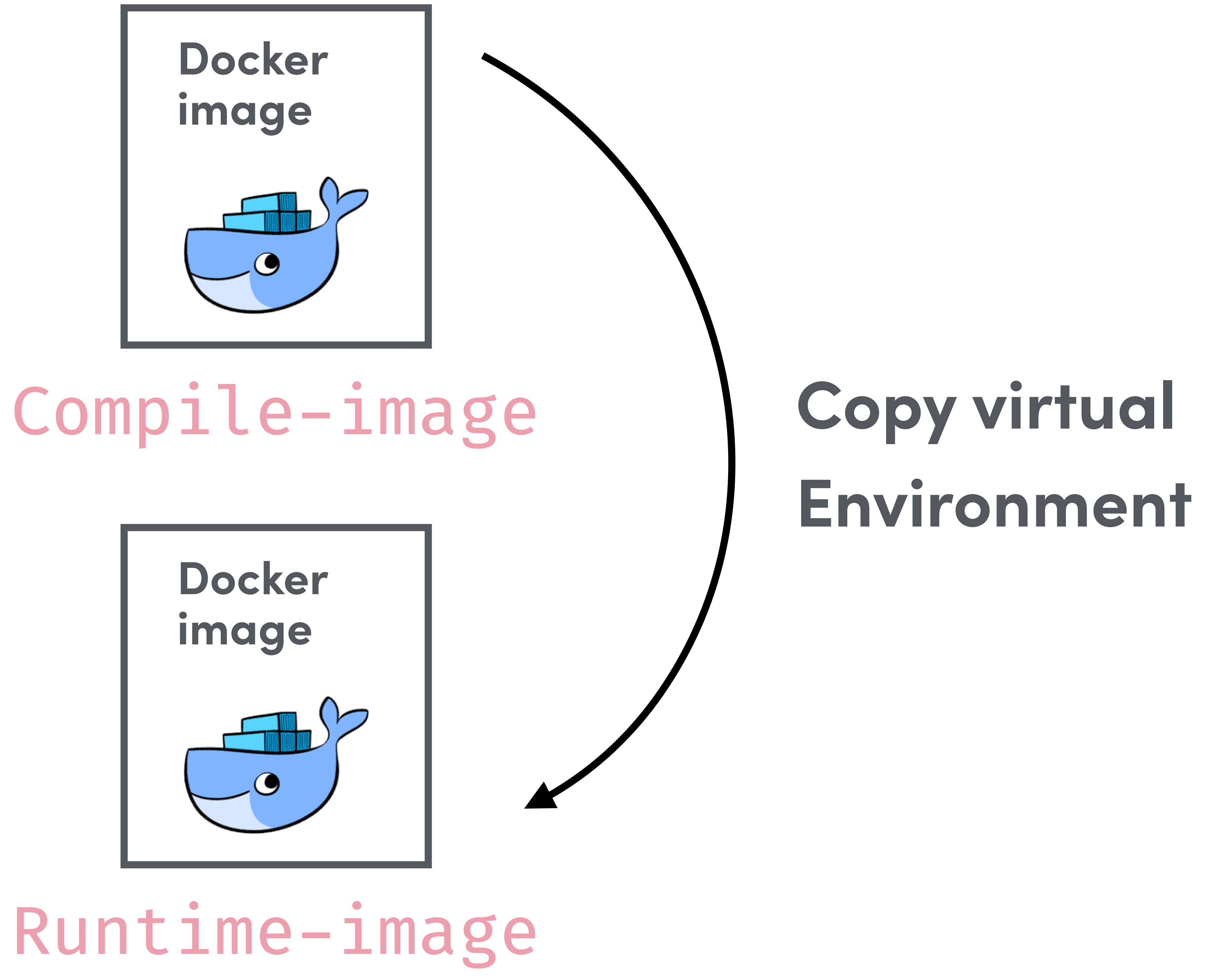
- Fetch and manage secrets in an intermediate layer
- Not all your dependencies will have been packed as wheels so you might need a compiler - build a compile and a runtime image
- Smaller images overall

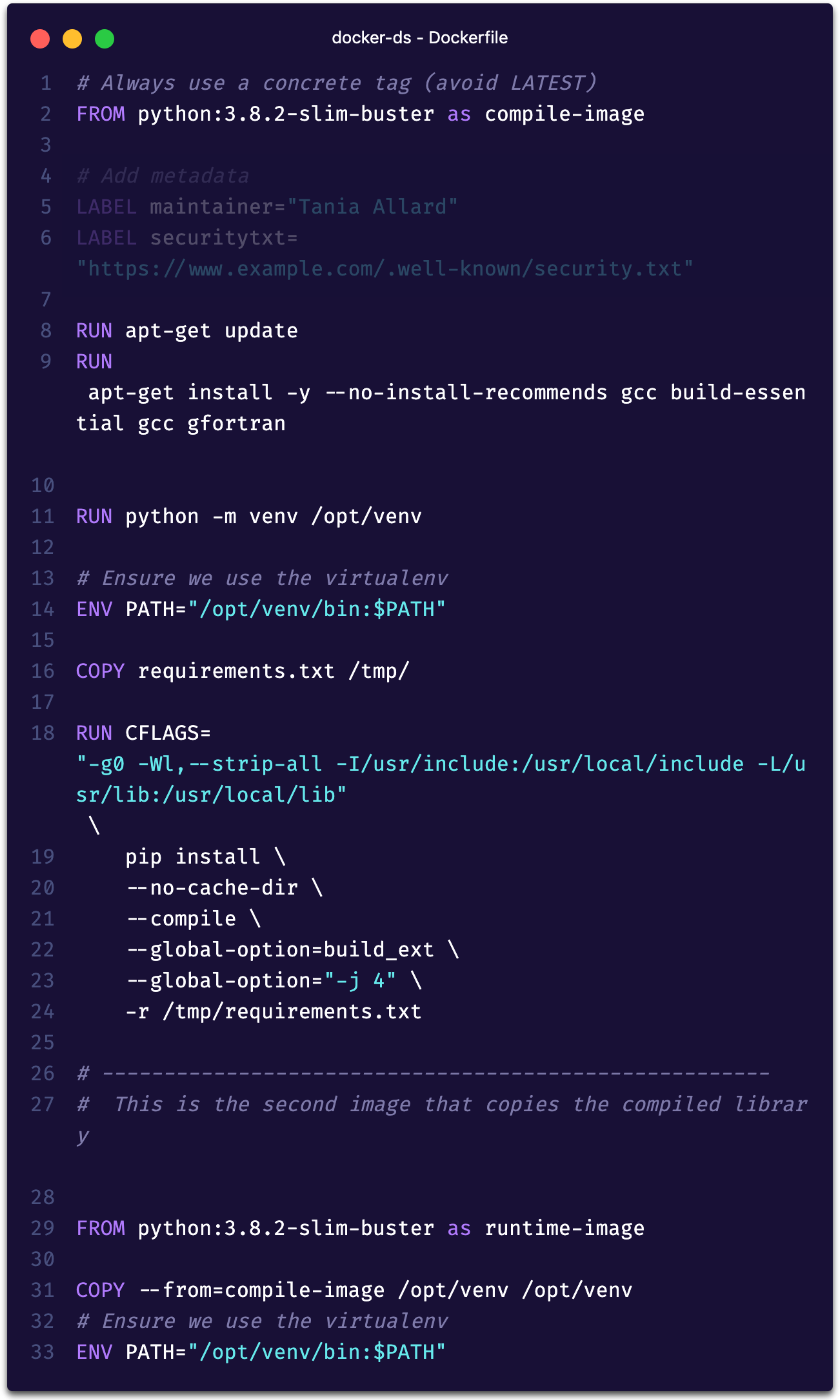

docker-ds - Dockerfile

```dockerfile
# Always use a concrete tag (avoid LATEST)
FROM python:3.8.2-slim-buster as compile-image

# Add metadata
LABEL maintainer="Tania Allard"
LABEL securitytxt="https://www.example.com/.well-known/security.txt"

RUN apt-get update
RUN apt-get install -y --no-install-recommends gcc build-essential gcc gfortran


RUN python -m venv /opt/venv

# Ensure we use the virtualenv
ENV PATH="/opt/venv/bin:$PATH"

COPY requirements.txt /tmp/

RUN CFLAGS="-g0 -Wl,--strip-all -I/usr/include:/usr/local/include -L/usr/lib:/usr/local/lib" \
    pip install \
    --no-cache-dir \
    --compile \
    --global-option=build_ext \
    --global-option="-j 4" \
    -r /tmp/requirements.txt

# ------------------------------------------------------
#  This is the second image that copies the compiled library


FROM python:3.8.2-slim-buster as runtime-image

COPY --from=compile-image /opt/venv /opt/venv
# Ensure we use the virtualenv
ENV PATH="/opt/venv/bin:$PATH"
```

USE MULTI STAGE BUILDS

```
$ docker build --pull --rm -f “Dockerfile"\
 -t trallard:data-scratch-1.0 "."
```

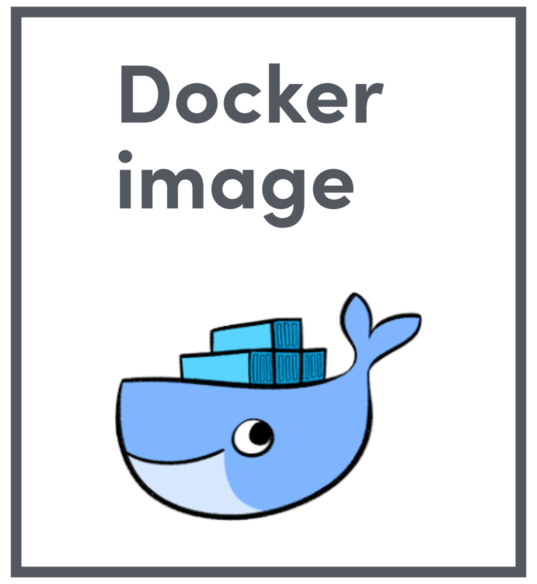




```
# Always use a concrete tag (avoid LATEST)
FROM python:3.8.2-slim-buster as compile-image

# Add metadata
LABEL maintainer="Tania Allard"
LABEL securitytxt=
"https://www.example.com/.well-known/security.txt"

RUN apt-get update
RUN
 apt-get install -y --no-install-recommends gcc build-essen
tial gcc gfortran


RUN python -m venv /opt/venv

# Ensure we use the virtualenv
ENV PATH="/opt/venv/bin:$PATH"

COPY requirements.txt /tmp/

RUN CFLAGS=
"-g0 -Wl,--strip-all -I/usr/include:/usr/local/include -L/u
sr/lib:/usr/local/lib"
 \
    pip install \
    --no-cache-dir \
    --compile \
    --global-option=build_ext \
    --global-option="-j 4" \
    -r /tmp/requirements.txt

# -------------------------------------------------------
#  This is the second image that copies the compiled librar
y


FROM python:3.8.2-slim-buster as runtime-image

COPY --from=compile-image /opt/venv /opt/venv
# Ensure we use the virtualenv
ENV PATH="/opt/venv/bin:$PATH"
```

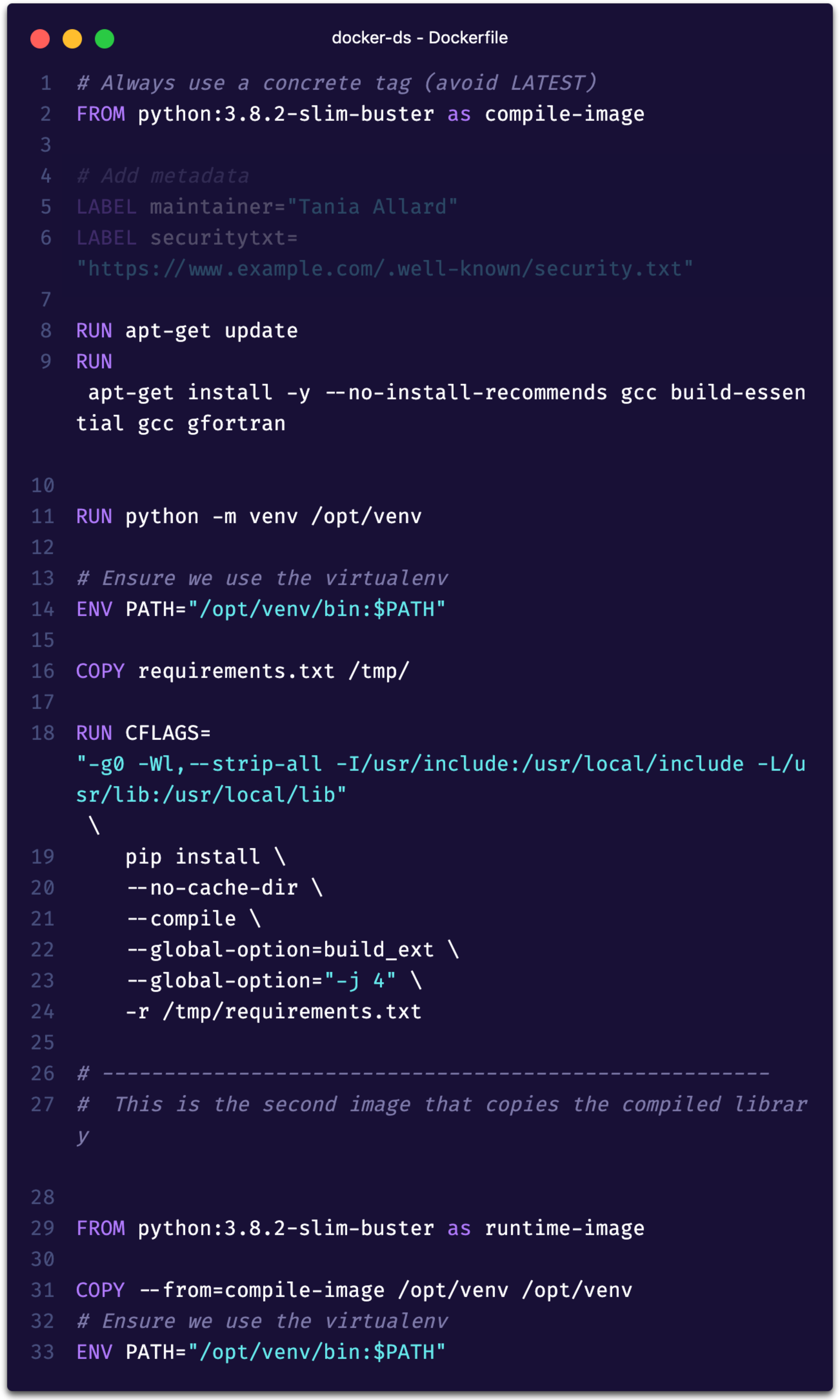
USE MULTI STAGE BUILDS

FINAL IMAGE



Runtime-image

trallard:data-scratch-1.0



```
# Always use a concrete tag (avoid LATEST)
FROM python:3.8.2-slim-buster as compile-image

# Add metadata
LABEL maintainer="Tania Allard"
LABEL securitytxt=
"https://www.example.com/.well-known/security.txt"

RUN apt-get update
RUN
 apt-get install -y --no-install-recommends gcc build-essential gcc gfortran


RUN python -m venv /opt/venv

# Ensure we use the virtualenv
ENV PATH="/opt/venv/bin:$PATH"

COPY requirements.txt /tmp/

RUN CFLAGS=
"-g0 -Wl,--strip-all -I/usr/include:/usr/local/include -L/usr/lib:/usr/local/lib"
 \
    pip install \
    --no-cache-dir \
    --compile \
    --global-option=build_ext \
    --global-option="-j 4" \
    -r /tmp/requirements.txt

# ------------------------------------------------------
#  This is the second image that copies the compiled library


FROM python:3.8.2-slim-buster as runtime-image

COPY --from=compile-image /opt/venv /opt/venv
# Ensure we use the virtualenv
ENV PATH="/opt/venv/bin:$PATH"
```

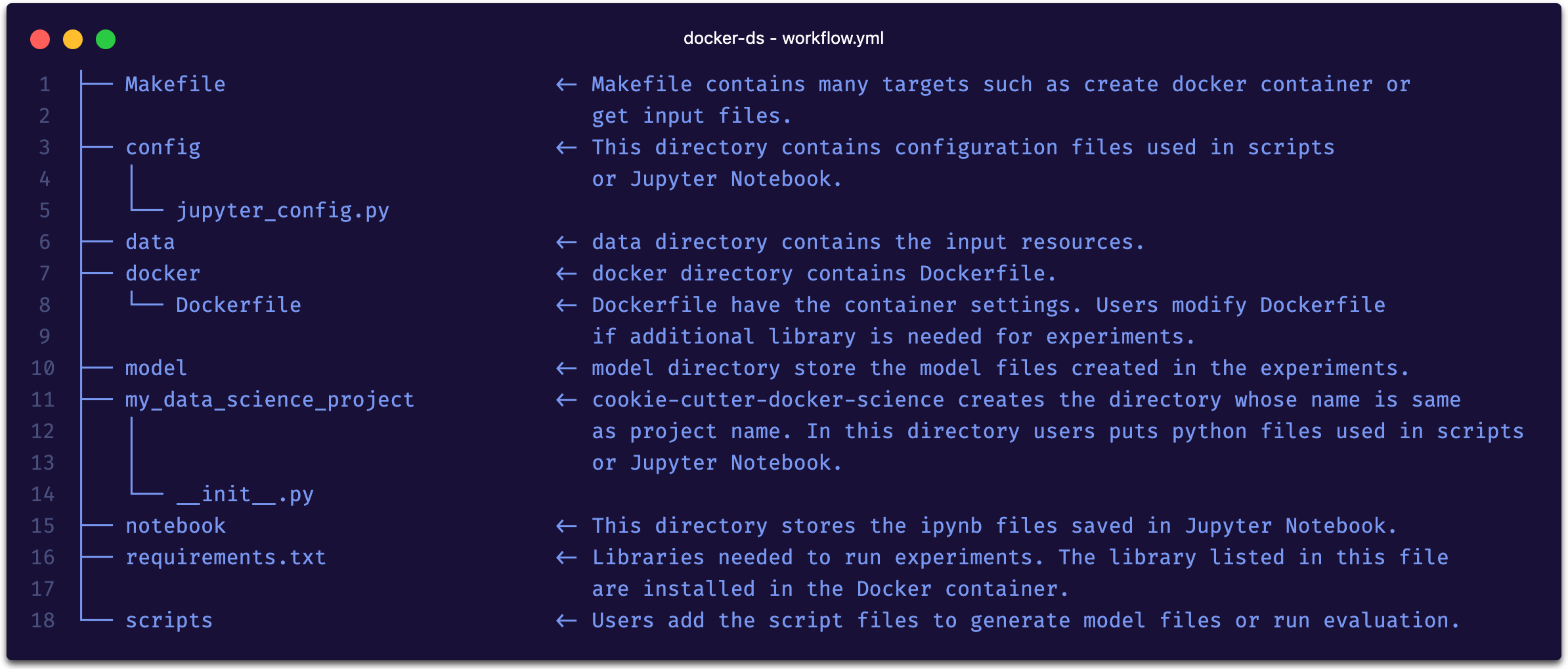
AUTOMATE

PROJECT TEMPLATES

Need a standard project template?

Use `cookie cutter data science`

Or `cookie cutter docker science`



docker-ds - workflow.yml

```
├── Makefile                       ← Makefile contains many targets such as create docker container or
│                                    get input files.
├── config                         ← This directory contains configuration files used in scripts
│   │                                or Jupyter Notebook.
│   └── jupyter_config.py
├── data                           ← data directory contains the input resources.
├── docker                         ← docker directory contains Dockerfile.
│   └── Dockerfile                 ← Dockerfile have the container settings. Users modify Dockerfile
│                                    if additional library is needed for experiments.
├── model                          ← model directory store the model files created in the experiments.
├── my_data_science_project        ← cookie-cutter-docker-science creates the directory whose name is same
│   │                                as project name. In this directory users puts python files used in scripts
│   │                                or Jupyter Notebook.
│   └── __init__.py
├── notebook                       ← This directory stores the ipynb files saved in Jupyter Notebook.
├── requirements.txt               ← Libraries needed to run experiments. The library listed in this file
│                                    are installed in the Docker container.
└── scripts                        ← Users add the script files to generate model files or run evaluation.
```

https://github.com/docker-science/cookiecutter-docker-science

https://drivendata.github.io/cookiecutter-data-science/

```
$ conda install jupyter repo2docker
$ jupyter-repo2docker “.”
```

tania@Mictlan: ~/Documents/github/trallard/docker-ds

```
…/trallard/docker-ds  master ● ?                                      anaconda3
$ jupyter-repo2docker .
```

```
100%    10 GB    18%    04/5, 3:58 pm
```

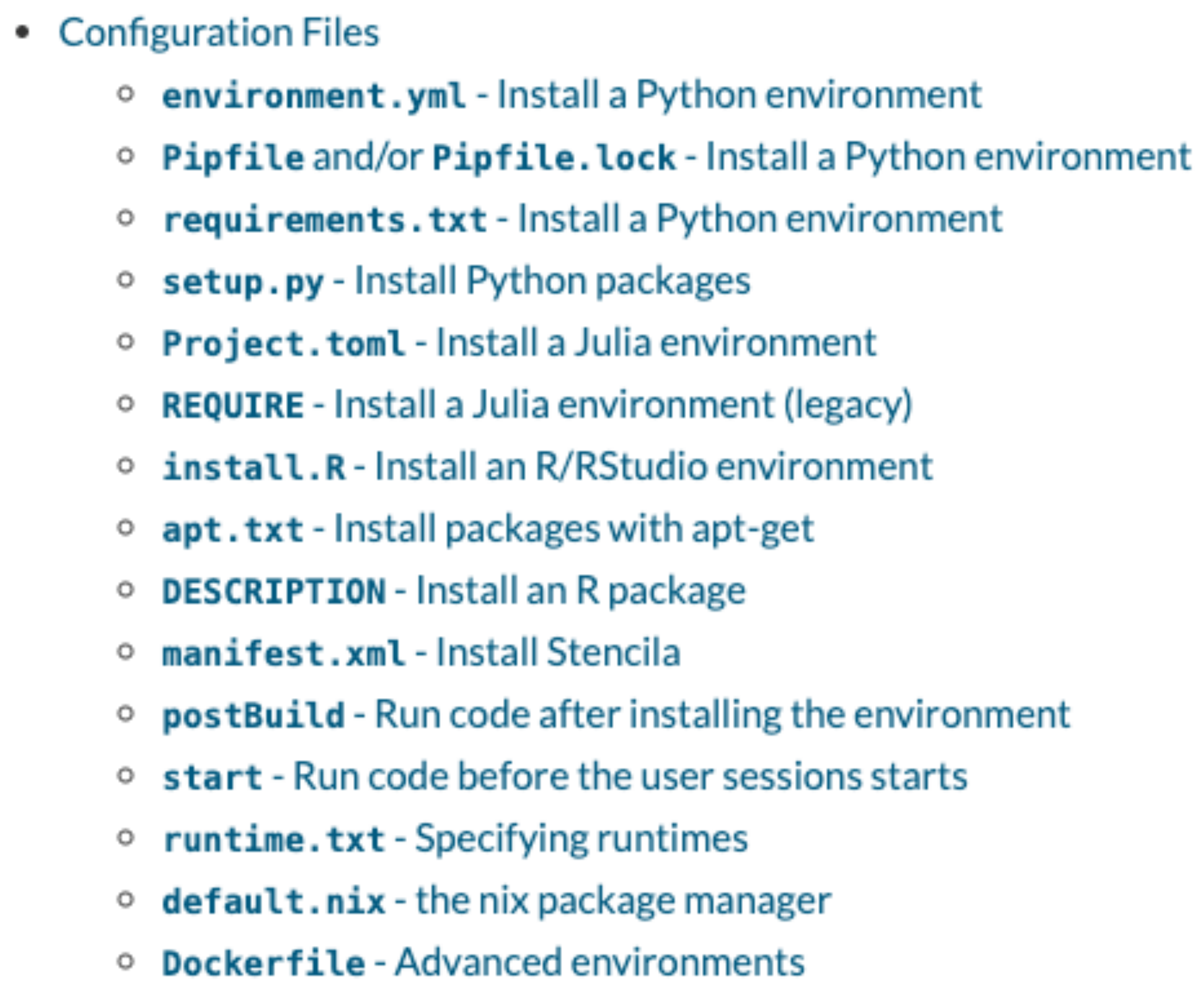
https://repo2docker.readthedocs.io/en/latest

DO NOT REINVENT THE WHEEL

Leverage the existence and usage of tools like `repo2docker`.

Already configured and optimised for Data Science / Scientific computing.

DO NOT REINVENT THE WHEEL

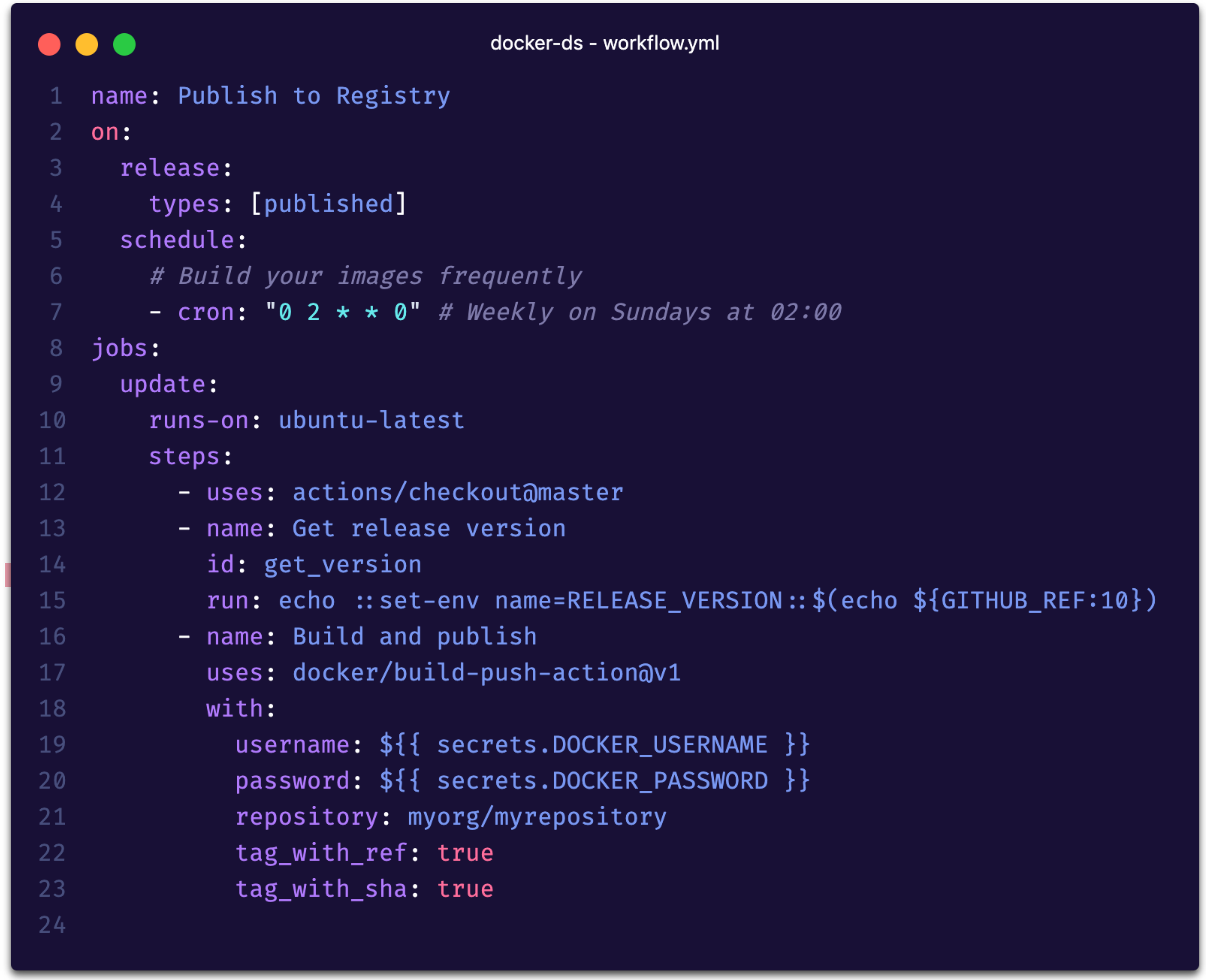


- Configuration Files
    - `environment.yml` - Install a Python environment
    - `Pipfile` and/or `Pipfile.lock` - Install a Python environment
    - `requirements.txt` - Install a Python environment
    - `setup.py` - Install Python packages
    - `Project.toml` - Install a Julia environment
    - `REQUIRE` - Install a Julia environment (legacy)
    - `install.R` - Install an R/RStudio environment
    - `apt.txt` - Install packages with apt-get
    - `DESCRIPTION` - Install an R package
    - `manifest.xml` - Install Stencila
    - `postBuild` - Run code after installing the environment
    - `start` - Run code before the user sessions starts
    - `runtime.txt` - Specifying runtimes
    - `default.nix` - the nix package manager
    - `Dockerfile` - Advanced environments

Leverage the existence and usage of tools like `repo2docker`.

Already configured and optimised for Data Science / Scientific computing.

https://repo2docker.readthedocs.io/en/latest


docker-ds - workflow.yml

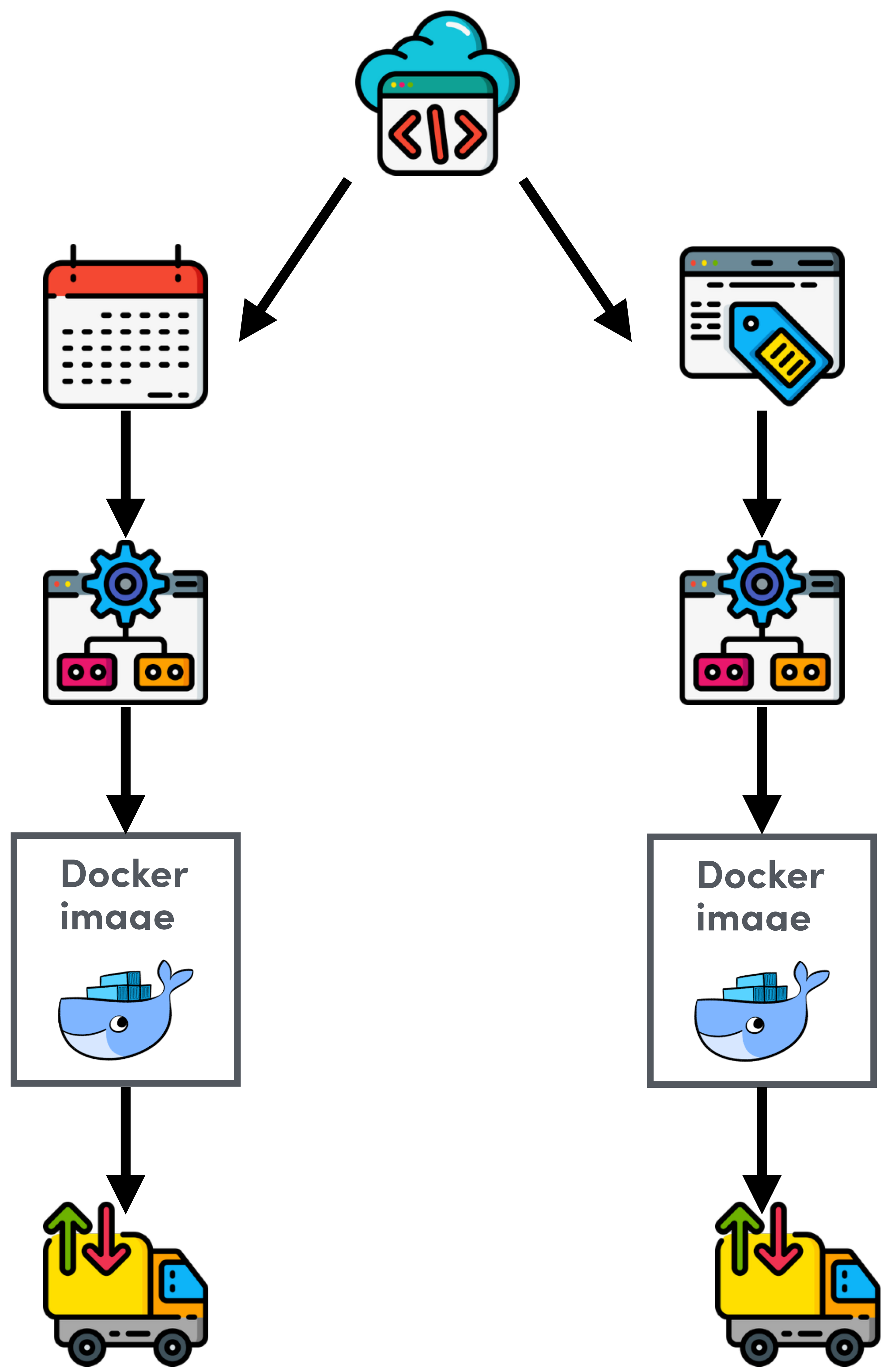
```yaml
name: Publish to Registry
on:
  release:
    types: [published]
  schedule:
    # Build your images frequently
    - cron: "0 2 * * 0" # Weekly on Sundays at 02:00
jobs:
  update:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@master
      - name: Get release version
        id: get_version
        run: echo ::set-env name=RELEASE_VERSION::$(echo ${GITHUB_REF:10})
      - name: Build and publish
        uses: docker/build-push-action@v1
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_PASSWORD }}
          repository: myorg/myrepository
          tag_with_ref: true
          tag_with_sha: true

```

DELEGATE TO YOUR CONTINUOUS INTEGRATION TOOL

Set Continuous integration (Travis, GitHub Actions, whatever you prefer).

And delegate your build - also build often.

https://repo2docker.readthedocs.io/en/latest



THIS WORKFLOW

- Code in version control
- Trigger on tag / Also scheduled trigger
- Build image
- Push image

TOP TIPS

TOP TIPS

1. Rebuild your images frequently - get security updates for system packages
2. Never work as root / minimise the privileges
3. You do not want to use Alpine Linux (go for buster, stretch or the Jupyter stack)
4. Always know what you are expecting: pin / version EVERYTHING (use pip-tools, conda, poetry or pipenv)
5. Leverage build cache

TOP TIPS

6. Use one Dockerfile per project
7. Use multi-stage builds - need to compile code? Need to reduce your image size?
8. Make your images identifiable (test, production, R&D) - also be careful when accessing databases and using ENV variables / build variables
9. Do not reinvent the wheel! Use repo2docker
10. Automate - no need to build and push manually
11. Use a linter

THANK YOU

@ixek

@trallard

trallard.dev