PYTHON IMAGES

Docker setup

MAC: https://docs.docker.com/desktop/install/mac-install/
LINUX: https://docs.docker.com/desktop/install/linux-install/

WINDOWS standard: https://docs.docker.com/desktop/install/windows-install/

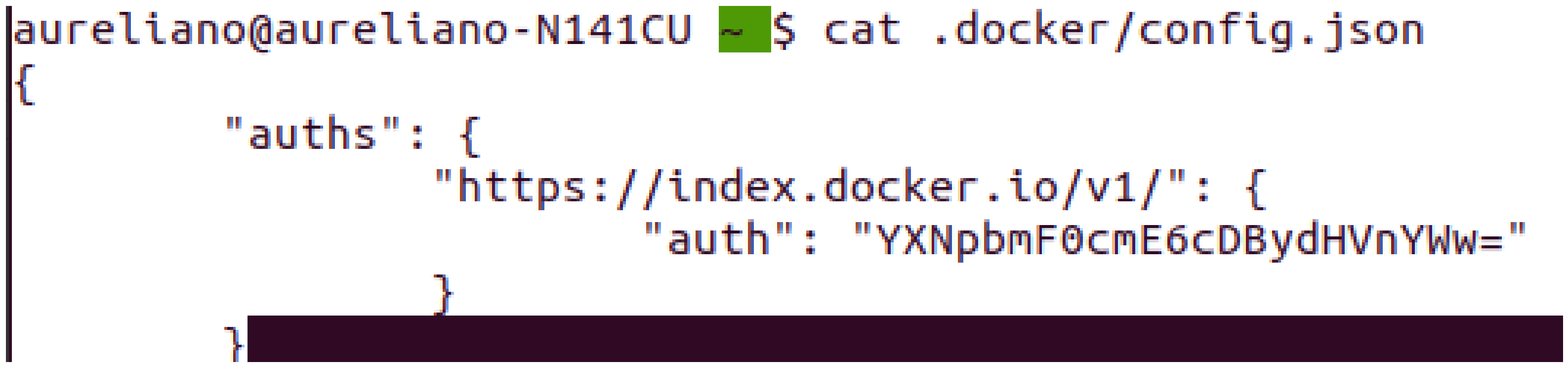
WINDOWS alternative1:
https://armstar.medium.com/install-virtualbox-ubuntu-docker-on-windows-10-a16765a09b

WINDOWS alternative2:
https://medium.com/@peorth/using-docker-with-virtualbox-and-windows-10-b351e7a34adc

Be sure to follow post install phase to avoid permissions issues

REGISTRY LOGIN

- CREATE an ACCOUNT ON https://hub.docker.com/
- DOCKER login to a registry : docker login registry.hub.docker.com -u user -p password
- DOCKER config file: auth stored in base64 encoding



```
aureliano@aureliano-N141CU ~ $ cat .docker/config.json
{
        "auths": {
                "https://index.docker.io/v1/": {
                        "auth": "YXNpbmF0cmE6cDBydHVnYWw="
                }
        }
```

DOCKER REGISTRY TEST

CHECKUP:

```
docker system info
```

BUILD:

create Dockerfile with content:

```
FROM alpine:3.15
```

```
docker build . -t myimage
docker image ls
```

RUN:

```
docker run -it myimage
docker ps
```

PUSH:

```
docker tag myimage mydockerhubaccount/myimage:v1.0.0
docker push mydockerhubaccount/myimage:v1.0.0
```

BASIC

```
Given a Dockerfile with an env var build

FROM python
ENV MY_VAR=test
ADD . .
CMD python run.py
```

- Build the image
- Push to private dockerhub registry
- Run the container overriding the default env var

https://github.com/ynov-campus-sophia/devops-B3-2023/tree/main/docker/basic

PYTHON PACKAGE MANAGERS

The **Python Package Index** (PyPI) is a repository of software that hosts an extensive collection of Python packages, development frameworks, tools, and libraries.

Pip is the “original” python package manager that others have attempted to improve upon. Pipenv & Poetry are two package managers that have done this with great success.

Virtualenv is a tool that allows the creation of named virtual environments where you can install packages in an isolated manner. Each environment has its own installation directories and doesn’t share libraries with other virtual environments (including globally installed libraries). A virtual environment is a folder containing packages and other dependencies that a Python project needs. The purpose of these environments is to keep projects separate and prevent dependency, version, and permission conflicts. Imagine a script that relies on 1.10 of the package NumPy, but a different script requires version 1.20.

Pipenv is a package management tool that “aims to bring the best of all packaging worlds” to Python. Pipenv is similar in spirit to Node.js’s npm and Ruby’s bundler. It’s popular among the Python community because it merges virtual environments and package management into a single tool. While pip is sufficient for personal use, Pipenv is recommended for collaborative projects as it’s a higher-level tool that simplifies dependency management for common use cases and can create virtual environments.

PYTHON PACKAGE MANAGERS

Poetry prides itself on making Python packaging and dependency management “easy”. Besides package management, it can help build distributions for applications and deploy them to PyPI. It also allows the declaration of the libraries a project depends on and installs/updates them avoiding any conflicting package requirements. Furthermore, Poetry isolates development versus production dependencies into separate virtual environments.

Conda is a multi-purpose package management tool. It manages package dependencies, can create virtual environments for applications, installs compatible Python distributions, and packages applications for deployment to production. It originated from Anaconda, which started as a data science package for Python. Conda installs packages from Anaconda rather than PyPI and can be used with multiple programming languages.

PIP

Given a Dockerfile with requirements txt file defining packages to be installed

```
FROM ubuntu:20.04

LABEL maintainer="aureliano.buendias@outlook.com"
RUN apt-get update -y && apt-get install -y python3 python3-dev python3-pip curl pipenv

WORKDIR /app

ADD requirements.txt /app/

RUN pip install -r requirements.txt
COPY src .

CMD ["/usr/bin/python3", "run.py"]
```

- Build the image
  docker build docker/pip pipimage
- Run the container exposing the guest port to host port
  docker run -p 5000:5000 pipimage

https://github.com/ynov-campus-sophia/devops-B3-2023/tree/main/docker/pip

READY TO USE IMAGES

No need to use ubuntu for a python deployment. Use prepackaged debian based python image

```
FROM python:3.10-slim
ENV PYTHONUNBUFFERED 1

WORKDIR /app/

ADD requirements.txt .
RUN pip install -r requirements.txt

COPY src .

CMD ["python", "run.py"]
```

- Check image size