Data608 - Assignment 2

Author: Andrii Voitkiv

INTRODUCTION PART

For this assignment we are going to investigate containers by examining pre-built Docker containers maintained by the Jupyter project. You will also be asked to deploy a container on a cloud instance.
It is not required that you install Docker Desktop, but if you have the time and inclination, you are welcome to try it out.

These containers are available online via DockerHub:
https://hub.docker.com/r/jupyter/datascience-notebook
They are maintained via github:
https://github.com/jupyter/docker-stacks
Documentation is maintained via readthedocs.io: https://jupyter-docker-stacks.readthedocs.io/en/latest/index.html

⚠ **Submit**

You can submit to D2L a Word or PDF document with your responses to the following questions.

PART A: EXPLORING CONTAINERS USING JUPYTER'S DOCKER STACKS

In this section, we will use **jupyter/scipy-notebook** and one **other container image created by Jupyter (of your choice)** to see how a container is structured.

Q1

Assuming that you had Docker installed, how would you download and install these onto your own machine?

I would use the `docker pull` command followed by the image name from Docker Hub.

```
docker pull jupyter/scipy-notebook
```

Another Jupyter image from the Jupyter Docker Stacks available on Docker Hub of my choice:

```
docker pull jupyter/datascience-notebook
```

Q2

Once downloaded, how do you run them?

```
docker run -p 8888:8888 jupyter/scipy-notebook
```

⚠ **Open url on local machine**

Try changing the port configuration in optional setting when running the new container to port 80 and call with http://127.0.0.1/?token=2e31ab3088fcf6a707480d456152c993606f26164fcc99ab (eliminating the port 8888 in the given address, as 80 is the default).

Q3

Pick one package which jupyter/scipy-notebook has installed which you're not previously familiar with and describe what it does in a sentence or two.

ⓘ **Get list of packages**

Go to container's terminal:

```
docker exec -it container-id /bin/sh
```

and then

```
pip freeze
```

`dask`: Dask is a parallel computing library that allows you to parallelize and distribute computations with ease. It provides dynamic task scheduling, which is optimized for interactive computational workloads and can scale from a single computer to a cluster of machines.

Q4

Choose a container image also created by Jupyter, and clearly indicate what it is. Now find a package contained in jupyter/scipy-notebook but not your other container image. What is it, what does it do, and why is it in scipy-notebook, but not your other container image?

I'll choose the `jupyter/minimal-notebook` image, which is a minimal Jupyter Notebook environment with only the basic packages required to run Jupyter Notebooks.

One package that is included in the `jupyter/scipy-notebook` image but not in the `jupyter/minimal-notebook` image is `scikit-learn`. Scikit-learn is a popular machine learning library. It includes various classification, regression, and clustering algorithms, as well as utilities for preprocessing, model selection, and evaluation.

The reason `scikit-learn` is included in the `jupyter/scipy-notebook` image but not in the `jupyter/minimal-notebook` image is because the `scipy-notebook` image is designed to provide a more comprehensive environment for scientific computing and data analysis, while the `minimal-notebook` image is meant to be a lightweight, bare-bones environment with only the essential packages required to run Jupyter Notebooks.

Next, let’s look at the Dockerfiles for both (for instance, https://github.com/jupyter/docker-stacks/blob/master/scipy-notebook/Dockerfile).

Q5

What container image are jupyter/scipy-notebook and your chosen container image based upon? Which variable in the DockerFile defines these?

Both the `jupyter/scipy-notebook` and the `jupyter/minimal-notebook` container images are based on the `jupyter/minimal-notebook` image. In the Dockerfile for `jupyter/scipy-notebook`, this is defined by the following ARG and FROM lines:

```
ARG OWNER=jupyter
ARG BASE_CONTAINER=$OWNER/minimal-notebook
FROM $BASE_CONTAINER
```

Q6

What other commands in the Dockerfile can you spot? List at least two more.

`LABEL`: This command adds metadata to the container image. In this case, it assigns a maintainer for the image:

```
LABEL maintainer="Jupyter Project <jupyter@googlegroups.com>"
```

`RUN`: This command executes commands during the image build process. In this Dockerfile, there are multiple RUN commands. One example is the command that updates the package list, installs several packages, and cleans up the package cache

```
RUN apt-get update --yes && \
    apt-get install --yes --no-install-recommends \
    build-essential \
    cm-super \
    dvipng \
    ffmpeg && \
    apt-get clean && rm -rf /var/lib/apt/lists/*
```

Another example is the `RUN` command that installs various Python packages using the mamba package manager.

Q7

What are the similarities between the parent container image and jupyter/scipy-notebook? Discuss one or two major similarities you can identify

Both `jupyter/scipy-notebook` and `jupyter/minimal-notebook` images are created and maintained by the Jupyter project and serve as a basis for creating customized Jupyter environments with pre-installed packages.

Q8

What are the differences between these two container images? Talk about one or two major ones.

The main difference is the set of pre-installed packages, with `jupyter/scipy-notebook` containing additional scientific computing libraries and tools compared to `jupyter/minimal-notebook`.

PART B: COMPARE WITH ANOTHER CONTAINER IMAGE

Now let’s look at the Dockerfile for Neo4j (click here).

Q1

What Dockerfile commands do you see in this file that are different than the ones for Jupyter’s Docker stacks? Are there any similarities?

The Neo4j Dockerfile has a different base image, environment variables, additional tools, and container entry points compared to Jupyter's Docker stacks. However, they share similarities in package installation, file and directory permission handling, and cleanup operations.

Q2

Discuss the differences you observe between the Neo4j container image and the container images maintained by Jupyter (including but not limited to the Dockerfiles).

Differences:

1. Base image: The base image for Neo4j is `openjdk:11-jdk-slim`, which is an OpenJDK-based image optimized for smaller size. In contrast, Jupyter's Docker stacks are based on `jupyter/minimal-notebook`, which is tailored for Jupyter environments.
2. Environment variables: The Neo4j Dockerfile has specific environment variables related to Neo4j, like `NEO4J_SHA256`, `NEO4J_TARBALL`, and `NEO4J_EDITION`. These are not present in Jupyter's Docker stacks.
3. Additional tools: The Neo4j Dockerfile installs `curl`, `wget`, `gosu`, and `jq` using the `apt install` command. Jupyter's Docker stacks do not install these specific tools.
4. Volumes: The Neo4j Dockerfile defines volumes for data and logs using the `VOLUME` command. Jupyter's Docker stacks do not define volumes.
5. Entrypoint and CMD: Neo4j's Dockerfile sets an entrypoint using the `ENTRYPOINT` command, and specifies a command with `CMD`. Jupyter's Docker stacks set these differently.

Similarities:

1. Package installation: Both Dockerfiles use package manager `apt` to install necessary packages and dependencies.
2. File and directory permissions: Both Dockerfiles set permissions for specific directories and files using `chown` and `chmod` commands to ensure proper access.

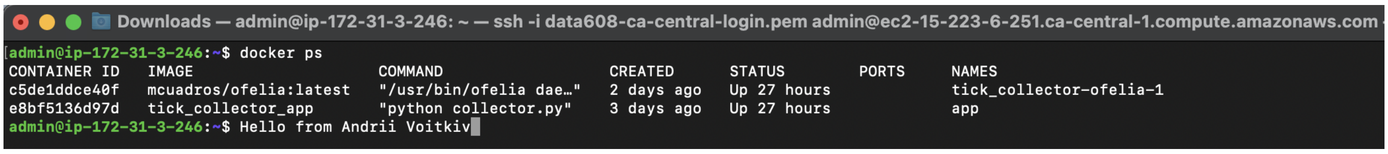
3. Cleanup: Both Dockerfiles perform cleanup operations, such as removing temporary files, cleaning package manager caches, and uninstalling unnecessary packages to reduce the final image size.

PART C: DEPLOYMENT TO CLOUD

Choose a Docker container image of your choice, and deploy this to a cloud platform of your choosing. You may take advantage of the AWS Learning Lab space (remember to find the "Learner Lab" link, agree to the conditions, start the lab, then click on the link which says "AWS"), use the provided class credit for Google Cloud Platforms, or use another cloud platform of your own choice.

Tutorials for both AWS and GCP can be found at the following links:
• https://aws.amazon.com/getting-started/hands-on/deploy-docker-containers/
• https://cloud.google.com/run/docs/deploying

Include as part of your response for the assignment a screenshot which demonstrates that you have successfully deployed the container to a cloud instance.



Downloads — admin@ip-172-31-3-246: ~ — ssh -i data608-ca-central-login.pem admin@ec2-15-223-6-251.ca-central-1.compute.amazonaws.com

```
admin@ip-172-31-3-246:~$ docker ps
CONTAINER ID   IMAGE                   COMMAND                  CREATED       STATUS         PORTS     NAMES
c5de1ddce40f   mcuadros/ofelia:latest  "/usr/bin/ofelia dae…"   2 days ago    Up 27 hours              tick_collector-ofelia-1
e8bf5136d97d   tick_collector_app      "python collector.py"    3 days ago    Up 27 hours              app
admin@ip-172-31-3-246:~$ Hello from Andrii Voitkiv
```

Two containers are deployed by me on AWS EC2 instance directly for the project.

Also include a list of references for any other websites or resources you may have used to supplement your learning in this part.

Q1

What was the most surprising part of this process? For example, was it simple, or was there some aspect which was confusing?

In the beginning I was a bit confused about the distinction between a Docker image and a Docker container. I understood that both are related to Docker, but I was not quite sure how they differ in terms of functionality and usage. After this exercise I clarified the distinction between both.

Q2

What was the most difficult part of this process?

I struggled with configuring the port for a container that runs a Jupyter notebook. When I tried to open the notebook link locally, I couldn't access it, and I wasn't sure why. Upon further investigation,

Check if port is free

`netstat -an | grep 8888` command in terminal

I discovered that there were already some listeners on the port I was trying to use. After changing the port to one that wasn't occupied, everything worked smoothly, and I could access the notebook without any issues.

Q3

Name one additional service supported by your cloud platform which this exercise has prompted you to investigate further, either as part of the assignment, your project, or for your own future development.

One additional service supported by AWS that I might investigate further is AWS SageMaker for machine learning. It provides an integrated Jupyter notebook interface. AWS SageMaker provides a feature called "SageMaker Notebook Instances" which allows to run Jupyter notebooks on the AWS infrastructure. I can create a custom Docker container that includes desired Jupyter notebook stack for machine learning (`jupyter/scipy-notebook` or `jupyter/tensorflow-notebook`).

Note

Use container registry service like Amazon ECR to store my custom Docker image

REFERENCES

1. Stack Overflow. (2019). Can't access jupyter notebook from docker. Available at: https://stackoverflow.com/questions/57064541/cant-access-jupyter-notebook-from-docker (Accessed: [2023-03-21]).
2. Amazon Web Services. (n.d.). AWS Sage Maker Documentation. Available at: https://docs.aws.amazon.com/sagemaker/index.html (Accessed: [2023-03-21]).
3. Saka, G. (n.d.). Deploying docker container on EC2 tutorial. Available at: https://gokturksaka.medium.com/deploy-docker-container-to-aws-ec2-with-bitbucket-pipeline-a25a9a1e28a2 (Accessed: [2023-03-14]).