A+ will have a maintenance break for a version upgrade on Monday January 3rd 2022, starting at 9:00 A.M. The site will not be available during the break. The break is expected to end around 3:00PM, but the site will be available as soon as the upgrade is complete. More information (https://wiki.aalto.fi/pages/viewpage.action?pageId=197422875).

Assignment: create a python build environment

This assignment involves building a Docker image that can be used for developing python software. The goal is to create a container image that can be used as build (test) environment for python, for instance, as typically used in the context of continuous integration and deployment.

Warning

The activities in this course are individual work. **Do not read or copy solutions from other students**. **Do not share solutions**. Remember that episodes of plagiarism and collusion are fraudulent means in studying according to the Aalto University Code of Academic Integrity (https://into.aalto.fi/display/enregulations/Aalto+University+Code+of+Academic+Integrity+and+Handling+Violations+Thereof) which may result in caution or suspension. See also the code of conduct (https://mycourses.aalto.fi/mod/page/view.php?id=774501) of the course.

Tip

Carefully read the related instructions (https://mycourses.aalto.fi/mod/page/view.php?id=774516) before submitting the assignment(s).

Task

Your task is to create a Docker image that can run, test, and package a python application.

Python versions

The image should support different versions of python: 3.8 and 3.9. They could be either pre-installed or selected through a `PYTHON_VERSION` build-time variable (https://docs.docker.com/engine/reference/commandline/build/#set-build-time-variables---build-arg).

Note

The image is built with:

```
docker build --build-arg PYTHON_VERSION=<NUMBER> .
```

Where `<NUMBER>` is one of 3.8 and 3.9.

Python 3.8 can be installed using offical repositories while Python 3.9 needs to be compiled and installed using the source found here (https://www.python.org/ftp/python/3.9.0/Python-3.9.0.tgz). Installing Python 3.9 requires the following dependencies:

- build-essential, libssl-dev, zlib1g-dev, libncurses5-dev, libncursesw5-dev, libreadline-dev, libsqlite3-dev,libgdbm-dev, libdb5.3-dev, libbz2-dev, libexpat1-dev, liblzma-dev, libffi-dev, and uuid-dev

Attention

Avoid compiling Python 3.8 especially any Python version less than 3.8.5. Compiling Python versions less than 3.8.5 would result in the submission getting stuck with `In grading` status. This is due to incompatibility with the specific python version and requirements installed by pip.

Tip

Try to pre-install all versions first and look at the build-time variables only later, once you have done a successful submission.

Build steps

The grader runs different build steps that rely on the tools detailed below.

Step	Software or command used
Install dependencies	`pip3 install -r requirements.txt`
Syntax check	`python3 -m compileall .`
Linting	pyLint (https://www.pylint.org/)
Unit testing	pytest (https://docs.pytest.org/) , nbmake (https://pypi.org/project/nbmake/)
Build wheel (https://pythonwheels.com/) package	pip3, C / C++ development tools, python-dev

Make sure that you have all necessary software installed in the container. The step that builds the python package relies on a C / C++ compiler (with `gcc` recommended) to build native libraries that need to be included in the wheel.

You can download a sample application to test your Docker container **`here`** (https://grader.cs.hut.fi/static/CS-E4190_2021Autumn/_downloads/sample_python.zip). Your container should be able to build and install the application successfully when invoked as follows:

- For both versions of python (3.8 and 3.9)

```
docker run --rm -it -v <application_path/sample_python>:/application <docker_container_image>
```

- For python 3.8 only

```
docker run --rm -it -v <application_path/sample_python>:/application <docker_container_image>
```

- For python 3.9 only

```
docker run --rm -it -v <application_path/sample_python>:/application <docker_container_image>
```

Grading

Submission are evaluated by means of an automated system based on the Container Structure Tests (https://github.com/GoogleContainerTools/container-structure-test) tool. You only need to submit a single file, namely, the Dockerfile for the container image that satisfies the requirements above.

Attention

Make sure to test your Dockerfile locally before submitting it. This is especially useful to obtain detailed error messages in case of issues. Please note that the grader will take some time to evaluate the submission, the usual evaluation time is 2 to 10 minutes but this might take longer than that. You can speed up the Python compilation by **not** using the `--enable-optimizations` flag in the configure script and running make with four to eight threads.

Make sure that your Dockerfile fulfills the following requirements.

- The base image **must** be ubuntu:latest (https://github.com/tianon/docker-brew-ubuntu-core/blob/1b67caf335fdf475610a883e53c686ce3b48d7a5/focal/Dockerfile).

- It specifies `/application` as the working directory.
- It defines the `PYTHON_VERSION` argument **only if** it supports build-time arguments.
- It does not contain any command (CMD) or entrypoint.

Note

The assignment runs multiple unit tests which give fractional points based on how the requirements in task are fulfilled, according to the table below.

Test	Points
Image is correctly built and working dir exists	12
Python exists	12
Install dependencies successfully	12
Syntax check is successful	12
Linting is successful	12
Unit testing is successful (Non-Jupyter)	12
Unit testing is successful (Jupyter notebooks)	12
Wheel package is built successfully	12
Image is correctly built for specific python version with build argument	12
Image size is below 1024 MB	12

⚠ The deadline for the assignment has passed (Friday, 29 October 2021, 23:30).

Docker Assignment: dockerize build environment

Upload the Dockerfile

Upload your Dockerfile as the solution.

Dockerfile

选择文件 未选择任何文件

Submit

« 3.3 Assignment: Hello Docker! (/css/2021/container...

4. Container orchestration and microservices » (/css/...