DISTRIBUTED TESTING SYSTEM - Version 1.0

This project aims to build a system to test student programming assignments and give feedback to them.

The users of the systems are students, teachers, and administrators.

The students are supposed to use the system to upload their assignments and visualize the output logs.

The teacher has the task of setting up a programming assignment. For every programming assignment, the teacher should

- provide a docker image. The assumption is that the image contains an executable "/tester/run" that when invoked considers the assignment assignment.zip located in "/assignment", runs the test on the assignments and when finished saves the results in the text file "/output/logs.txt" and in "output/result.txt". The result.txt file should contain only one line of text.
- set up the visibility of the assignment (visible, not visible)
- set up the list of the students that have access to the program assignment (in bulk or one by one)
- set up a deadline for the submissions
- set up a max time and max memory to be used for the test of the assignment
- set up a number of vCPU needed to run the tests
- set up a maximum number of assignment submissions that a student can have open (i.e., not evaluated yet)

The administrator of the system manages the creation, pause, and deletion of teachers/students.

Constraints When a teacher is paused no submissions can be received anymore in the teacher's programming assignments. Moreover, all the submissions for one of his or her assignments should be stopped if running, canceled otherwise. The students when checking the status of a submission should see that the assignment has been paused.

When the teacher is deleted all the teacher programming assignments are deleted. When a programming assignment is deleted, the execution of the submission of the assignments is stopped. The students when checking the status of a submission should see that the assignment has been deleted due to the assignment deletion.

A programming assignment should not be started if the system does not have the necessary vCPU or the memory needed to execute it.

The systems should be fair. Every student has a maximum number of assignments not evaluated and it should not be possible for an assignment to be postponed forever if other students are submitting new assignments. The system is in particular not First Come First Served. Assume indeed that at a certain point, A

has 2 assignments to be tested and B one. When the first test of A is terminated, B should be scheduled even if A has submitted the assignments before B.

Requirements By using a programmatic RESTful API a student should be able to:

- create own profile that can be access with a username and password
- login into the system
- get the list of his or her assignments
- submit a solution to an assignment
- check if the evaluation of an assignment has been done
- if not processed yet, cancel the submission of an assignment
- see the results of the assignment
- GUI support

The teacher of the system should be able to:

- add an assignment
- update the configuration information of the assignment (e.g., change the docker image, the visibility, ...)
- pause an assignment
- delete an assignment
- add or remove individual students from the assignment
- add or remove in bulk students from the assignment
- given an assignment and a student, get the list of student submissions
- given a submission get the status, output, and result of the submission
- delete a programming assignment
- trigger the re-evaluation of a submission
- stop the evaluation of a submission
- given an assignment, stop the evaluation of all submissions
- given an assignment, extract in bulk all the students' submissions logs in a zip file (i.e., the text in logs.txt)
- given an assignment, extract in bulk all the students' submission metadata (e.g., student name, submission id, submission time, result string contained in result.txt) in a CSV file
- given an assignment and a student, extract in bulk all the student's submissions logs in a zip file
- given an assignment and a student, extract in bulk all the student's submission metadata in a CSV file
- GUI support

The administrator of the system should be able to:

- monitoring and logging the platform using a dashboard
- add teachers
- pause a teacher
- delete a student or a teacher

The developer of the systems has to:

- Use Continuous Integration and Deployment
- Infrastructure as a Code with an automatic DevOps pipeline
- scalable, supporting multiple users exploiting if needed more resources in the cloud (note: an assignment validation requires a vCPU and therefore the number of vCPUs limits the number of parallel evaluations possible.)
- have tests to test the system (at least unit test, integration)
- security (proper credential management and common standard security practices enforced). Note that the evaluation of the docker must not tamper with the remaining part of the system since potentially the code of the students is non-trusted
- provide user stories to explain how the system is intended to be used
- provide minimal documentation to deploy and run the system

Notes:

- the possibility to deploy on multi-clouds and avoid vendor lock-in is a plus
- a proper team organization and teamwork management are needed and a plus