# Grader: An LTI app for automatic, secure, program validation using the Docker sandbox

Petrović Gajo, Nikolić Aleksandar, Segedinac Milan, Kovačević Aleksandar, Konjović Zora
University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia
{gajop, anikolic, milansegedinac, kocha78, ftn_zora}@uns.ac.rs

***Abstract*— In this paper we present a software framework for automatic validation of code submitted for programming assignments. The framework consists of 1) a website interface which can be used by teachers to add new assignments and by students to submit solutions, 2) a REST API interface used to submit solutions programmatically, 3) test environment which invokes assignment-specific tests for the supported programming languages, 4) a sandbox environment, created using the Docker containers, which allows for secure execution of unsafe code and 5) LMS integration by implementing the LTI specification. We have added initial support for writing simple tests in Matlab, Python and Java, as well as the ability to extend it via plugins to other programming languages.**

## I. INTRODUCTION

In recent years we are witnessing an explosion of e-learning systems and online courses. More and more students are taking online courses, which can be seen from the rise of sites offering vast varieties of courses, (examples include Coursera, Udacity, Edx and similar [1-3]). The term MOOC [4] (Massive Open Online Course) has also been coined recently to denote the ever increasing presence of online courses that have large amount of enrolled students. In these courses, with student numbers sometimes reaching tens and hundreds of thousands, it's unfeasible to manually inspect submitted work and automatic grading is therefore necessary. For online programming courses it would therefore be useful to have a way to create and publish code assignments which can be automatically validated. Traditional university courses also often have an online component, usually in the form of a site where students can obtain course information, lecture notes, assignments and similar. As the components of these sites tend to be rather similar even in different areas of science, many universities have adopted an LMS (Learning Management System) [5] designed to provide common functionalities. As the usage of these LMSs has grown, so has the demand for additional features, and in recent years we have seen a push towards the creation of LTI apps [6] (Learning tools interoperability applications, i.e. content providers) that can be added to specific courses (content consumers). In order to reduce teacher workload when grading exams, as well as to create better learning material, we propose a framework for automatic validation. Our framework consists of an extendable validation module that tests submitted solutions, a Docker-based sandboxing framework and a web application that implements the LTI interface, allowing it to be integrated within an LMS. The web application has been created using the Django framework [7]. The implemented system is available on our BitBucket repository[1] . The system modules are presented on Figure 1, also showing the control flow within a system, and each module is described in detail in the corresponding section of the paper.

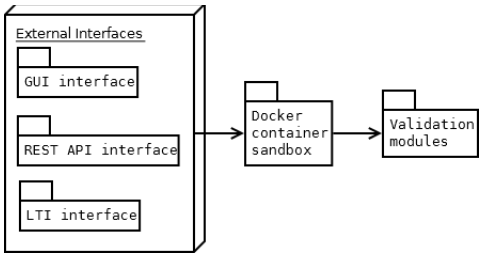


Figure 1. System modules

This paper is organized as follows: section 1 is the introduction, section 2 presents the website GUI and API interfaces, section 3 describes the validation module, section 4 introduces the Docker sandboxing environment, section 5 describes the LTI specification and LMS integration and lastly, section 6 is the conclusion in which we summarize what was done.

## II. WEBSITE INTERFACE

The course instructor submits tasks on the web site by uploading files and selecting their type, possible options include:

- Implementation - file containing a correct solution, supplied by the instructor , optional
- Test - a file containing a collection of tests used when validating code correctness, at least one is required
- Unspecified - any additional file not belonging to the first two categories (e.g. PDF files that give detailed assignment instructions, or input files needed to be read by the program).

Once submitted, the task will become available to students who can then attempt to solve them by submitting their solutions to the site. This is done either by using the web interface or programmatically, using the REST API. To use the REST API, the client must do the following: 1) create a zip archive from all solution files, 2) create a base64 encoded string from the created archive and 3) do a POST request with the new string as one of the parameters. The server will then decode the string, unzip the files and then proceed normally as if the files were submitted using the web site interface.

[1] Repository of the implemented system: https://bitbucket.org/gajop/automatic-grading-ftn.

The submitted code is copied to a testing folder and then, using a new Docker container, validation is performed for each test, after which test results are returned and then displayed to the user.

## III. Validation Module

The validation module provides an extendable platform for executing various user-defined tests. These tests, often created by teachers along with the correct solution, are used to verify program correctness. We have implemented support for the creation of simple tests in Matlab, Python and Java programming languages. Validation is performed by invoking functions from the code being tested and verifying if the results match an expected outcome.

However, the key feature in the validation module is that it can be easily extended with new test types, by writing modules that comply with a simple interface.

These modules (from hereon called plugins) are added to the application by modifying the main Django configuration file. They can be added by assigning a list of viable testers for each programming language, so new plugins could just be appended to the list. Support for new programming languages can also be added, which implies that test writing for arbitrary programming language could be supported by creating plugins.

As mentioned before, plugins are required to implement a certain interface. In this case, this includes creating a function which takes submitted code, tests and additional data (usually correct code if any) as input and provides test results as output. Test results include a Boolean value denoting each test's successfulness, along with an optional message, and another Boolean value denoting the successfulness of the entire test suite (this will most often be true only if all tests are successful, and otherwise false).

As Docker provides a way to sandbox code execution, the plugins don't need to be written by taking malicious code into consideration. That said, they still need to use whatever constructs of the language are available to check for certain types of erroneous code such as the case of: infinite loops, exception throwing, incorrect function interfaces (e.g. wrong type or number of function arguments), and so on. This requires the plugins to be non-trivial, but the hope is that they will usually be written once per programming language, and that the most users will be writing simple tests for specific tasks.

## IV. Docker

Docker [8] is an open source project created by Docker Inc. (formerly dotCloud) capable of running programs inside LXC (Linux Containers) [9]. LXC allow users to execute code in an environment isolated from the host operating system, and Docker wraps around LXC and provides a higher level API that allows easier creation and management of such containers.

This method of isolating executed code from the host operating system, also called sandboxing, is often used 1) to ease portability ? as software only needs to work within a predefined container, and that container can then be distributed, and 2) to safely run code ? by separating the host operating system from the container, the programs within the container are unable to easily damage the system (they would need to break containment, which while sometimes possible with kernel exploits, usually isn't trivial). In the case of Docker, it's also possible to add extra constraints that increase the security and stability of untrusted code execution. In this case, we limited maximum RAM usage by supplying the option ?m=MAX_RAM (in megabytes) on each execution. This prevents the tested program to exhaust the system memory and therefore crash various other OS programs when they try to allocate memory for themselves.

Docker containers are created by making a configuration file which is done by taking a base image (such as stock Ubuntu), and then listing commands that initialize it to a desired state. This usually involves making some system-wide settings such as obtaining software packages or changing configuration files, as well as copying some custom files one would need.

In our case, we created two Docker images. The first image, grading-base, starts from the official Ubuntu base, updates the system packages and installs a few additional ones (python-pip and octave). Detailed information on how to setup a configuration script is available[2], and we have also released our grading-base script at the GitHub repository [3]. The second image, grading, uses the previously created grading-base as its foundation and adds a few additional files used to run tests (most notable being the validation plugins)[4]. The reason we opted to create two containers instead of one was because this way we have one image (grading-base) that takes a while to build, but will remain mostly unchanged through time. This allows us to add and modify plugins fast, as the other container can be rebuilt quickly, having to only copy a few of the plugin files, a process far faster than doing system-wide updates.

An alternative to Docker containers (more specifically LXC) would be to use full blown VMs such as VirtualBox or VMware that partially emulate the hardware to increase the level of virtualization. While these alternatives tend to be more secure than Docker as it's usually harder to break through the hypervisor that's used by VMs, they tend to have poor performances [10], seeing how it can take minutes for VMs to start cleanly and often 100s of MBs of RAM to start a new OS instance. In comparison to VMs, we have found that Docker container startup measures in a few seconds, which makes it fast enough.

Multipliers can be especially confusing. Write "Magnetization (kA/m)" or "Magnetization ($10^3$ A/m)." Figure labels should be legible, about 10-point type.

## V. LTI – Learning Tools Interoperability

LTI apps are educational tools which implement the LTI specification as described by IMS Global. These tools provide integration with a tool consumer, usually an LMS. Creating separate LTI apps instead of bundling their functionality within an LMS allows tool reuse in multiple LMS systems (e.g. Canvas, Moodle, Blackboard [12-14], etc.) without any modifications. This also allows for the creation of LMS systems with just the core functionality that is a necessity for most users, while the specific requirements of a learning institution (e.g. high school or university) or course are met by adding custom LTI apps.

[2] Docker configuration script instructions: http://docs.docker.io/en/latest/use/builder/
[3] Grading-base configuration script https://github.com/gajop/grading-base
[4] Grading configuration scirpt: http://bit.ly/MhOAok

LTI apps are web applications running on arbitrary servers that implement a simple interface by handling a certain POST request, named app launch [15]. In app launch, which happens automatically when the user first tries to use the tool, the LMS sends the POST request comprised of useful information, largely to identify the user, course and usage context, for which the most important parameters are:

- user_id: Unique id of the user.
- roles: Roles that the user has in this context (most common ones being Learner and Instructor)
- lis_person_name_full: Full name of the user. This won't be sent if the app is configured to launch users in anonymous mode.
- context_id: Context (course) unique id.
- context_title: Name of the context (course).

After the initial request, the tool is started and will be rendered within an LMS. The rendering will usually be done in an iframe as displayed in Figure 2. If implemented correctly, by rendering it within an iframe it can seem as if the LTI app is a part of a single LMS, which users tend to prefer. If applicable, the tool may provide configuration files for a specific LMS which defines how the tool can be accessed within the LMS, usually by putting links in the LMS to certain parts of the tool to better integrate each functionality. Example Canvas configuration files are also provided[5].

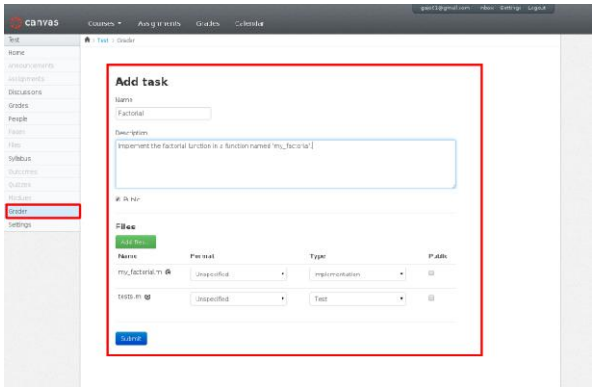


Figure 2. LTI integration within the Canvas LMS (red outline denotes the LTI components.)

As the LTI app receives parameters defining the active user and their permissions in the request, there needs to be a way that the authenticity of the request sender (LMS) can be verified. To do this, the LTI specification defines the use of the OAuth protocol, which requires a key and a shared secret [16]. The key is transmitted with each message along with an OAuth signature generated based on the key. The consumer key is used by both parties to identify who they are talking to, while the secret is used to digitally sign messages in both directions. Both the tool and the consumer need to agree on a key and secret, and after those are set, a higher level of security will be achieved.

[5] Canvas configuration file can be found at the repository URL: http://bit.ly/LqlNOq

## VI. Conclusion

In this paper we presented a framework for creating and executing automatic tests that validate user submitted programs. Programs can be submitted through the website either by uploading files manually using the GUI, or programmatically using a REST API.

The system is capable of executing untrusted code securely by sandboxing it within a Docker container. Along with the added security, this also frees resources that would otherwise need to go into considering the security aspects when building new validation plugins.

We have also implemented the LTI specification and thus our web application can be used as an LTI app, and can be embedded in various LMS. This can speed up the deployment time greatly, and it gives us an ability to create tools that focus on doing one job really well, without the need to add common functionality that can be found in an LMS (e.g. no need to create interfaces for authorization, account creation and similar).

As future work we would like to implement the ability to store and display code analysis other than correctness validation, such as code complexity analysis or plagiarism detection.

## Acknowledgment

Research presented in this paper is partly funded by the Ministry of Education, Science and Technological Development of the Republic of Serbia, Grant No. III 47003.

## References

[1] "Coursera online course platform" [Online], Available: https://www.coursera.org/, [Accessed 30 1 2014].

[2] "Udacity online course platform" [Online], https://www.udacity.com/ [Accessed 30 1 2014]

[3] "Edx online course platform" [Online], https://www.edx.org/ [Accessed 30 1 2014]

[4] S. Kolukuluri, "Massive Open Online Courses, Enhancement to edX-platform", master thesis, Indian Institute of Technology, Bombay Mumbai, 10.2013

[5] W. R. Watson, S. L. Watson, "An Argument for Clarity: What are Learning Management Systems, What are They Not, and What Should They Become? ", pp. 28-34, TechTrends, 2007

[6] S. Booth, S. Peacock, S. P. Vickers, "Plug and play learning application integration using IMS Learning Tools Interoperability", ascilite, pp. 143-147, 2011

[7] "Django framework" [Online], https://djangoproject.com, [Accessed 30 1 2014]

[8] "Docker container manager" [Online], http://www.docker.io/ [Accessed 30 1 2014]

[9] "LXC – Linux Containers" [Online], http://linuxcontainers.org/ [Accessed 30 1 2014]

[10] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, C. A. F. De Rose, "Performance Evaluation of Container-based Virtualization for High Performance Computing Environments", Parallel, Distributed and Network-Based Processing, pp. 233-240, 2013

[11] "IMS Global Learning Consortium" [Online], http://www.imsglobal.org/ [Accessed 30 1 2014]

[12] "Canvas LMS by Instructure", [Online] http://www.instructure.com/ [Accessed 30 1 2014]

[13] "Moodle LMS" [Online] https://moodle.org/ [Accessed 30 1 2014]

[14] "Blackboard LMS" [Online] http://www.blackboard.com/ [Accessed 30 1 2014]

[15] "Writing LTI Stuff" [Online] https://lti-examples.heroku.com/code.html [Accessed 30 1 2014]

[16] "IMS GLC Learning Tools Interoperability Basic LTI Implementation Guide" http://bit.ly/1fgmOmn [Online] [Accessed 30 1 2014]