Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 100 (2015) 1665 – 1671

Procedia Engineering

www.elsevier.com/locate/procedia

25th DAAAM International Symposium on Intelligent Manufacturing and Automation, DAAAM 2014

# Docker as Platform for Assignments Evaluation

František Špaček*, Radomír Sohlich, Tomáš Dulík

*Tomas Bata University in Zlín, Faculty of Applied Informatics, Nad Stráněmi 4511, 760 05 Zlín, Czech Republic*

**Abstract**

Programming courses are significant part of IT experts' education process. To being able to provide adequate teaching quality in such courses, lecturers should be exempted from routine tasks like source code compilation, testing and grading. Current computers are equipped with enough computational power to automate these routine tasks. This paper discusses the analysis and realization of such a system for user submitted automatic source code evaluation. The main system requirement was the safe runtime environment (sandbox) for executing potentially dangerous programs. Container based platform Docker was selected after research of ready to use sandbox technologies. This platform simplifies access to isolation mechanism which are implemented in the current Linux kernel and provides API for system integration. The implemented system around Docker platform is named APAC (Automatic Programming Assignment Checker). In the paper APAC's architecture and implementation are described and discussed.


Peer-review under responsibility of DAAAM International Vienna

*Keywords:* Linux; Docker; Containers; Sandbox; Assignments evaluation

## 1. Introduction

Computer science is a cutting-edge and progressive field of study. To maintain suitable teaching quality, especially in programming courses, it is important to reduce the time that lecturers spend for routine tasks such as students' submissions evaluation [1]. An automatic evaluation system provides a great relief for the teachers, and at the same time, it improves the objectivity, quality and speed of the assessment and gives the feedback to the students in almost real-time. The first mention of automatic grading of source code submission is dated to 1965. [2] Since

* Corresponding author
*E-mail address:* spacek@fai.utb.cz


Peer-review under responsibility of DAAAM International Vienna
doi:10.1016/j.proeng.2015.01.541

then, number of IT students has rapidly increased and automatic evaluation systems are integral part of programming courses. There are multiple ready to use solutions for automatic assignment evaluation [3, 4], unfortunately many of them are no longer actively developed (based on activity in source code repositories). One of the widely spread system is the web based system Mooshak [5], which was designed to support programming contests. This system is mature and it can be recognized as pioneer in the field of web based automatic source code evaluation. Submissions are tested inside Apache *safeexec* module. This module can be considered as primitive sandbox environment. Another more current evaluation system is the CS50. This web-based solution was developed at Harvard University as part of online course system called EdX. It uses modern approach for testing assignments in sandbox environment. CS50 provides HTTP API for upload and run of the assignments. Server backend is built on Node.js. Run-time sandbox is created via SELinux [6] and PAM limits [7]. CS50 is available under Creative Commons license. [8]

The reviewed systems give us extended view on the problems related to the development of automatic source code evaluation system. From our point of view, many of these systems are not properly focused on importance of safe runtime testing environment. There are a few mechanisms for establishing sandbox environment usable in the automatic submission evaluation system. [9] The aim of our work is to develop a new system providing safe testing environment, without compromising host system. Based on our experimental implementation a solution built on kernel namespace isolation was chosen. Kernel namespaces are implemented by means of LXC tools [10] and Docker platform [11]. Docker platform was selected because its readiness for system integration.

As part of our work, we have developed a system named Automatic Programming Assignment Checker (APAC) which was implemented around isolated sandbox containers provided by Docker.

This paper starts with the description of previous work on our solution. Next section of paper describes Docker platform and isolation mechanisms, which are used to create isolated containers. Following section is an architecture and implementation overview for APAC. In the conclusion, the benefits of the proposed approach are summarized.

## 2. Previous work

The system described within this paper represents the second iteration, which is built and based on the foundations and feedback from first version. The first version [12] of our system was successfully run during summer term in 2013/14 academic year within course ”Algorithms and data structures” at Faculty of Applied Informatics, Thomas Bata University in Zlin. Feedback from the course was subsequently used to improve list of requirements and functionalities for next version of the system. Resulting list of new most required features as follows:

1. assignment configuration management,
2. management of submitted programs,
3. test vectors definition administration,
4. evaluation of execution output,
5. isolation of submission testing from host system,
6. sandbox management and monitoring,
7. XEN/KVM machine compatibility,
8. expandable programming language and analysis tools support.

The initial version can be perceived as proof-of-concept of our proposed approach. This version is also based on Docker platform, although support for programming languages is hard coded into web application. It also implements only simple API for start processing submission files from Moodle. [12]

## 3. Docker

Docker is a container virtualization platform. It is built on publicly available open source technologies. Overview of Docker architecture is depicted in Fig. 1. Container virtualization is in its core lightweight process groups' isolation. Its implementation is based on set of Linux kernel mechanisms. Process resources separation is provided by kernel namespaces. From the Linux kernel version 3.8, full set of namespaces is implemented: *PID, IPC,*

*Network, Mount, UTS,* and *User*. This set of namespaces provides complete segregation for standalone containers creation. *PID* namespace isolates process ID numberspace. Each *PID* namespace has its own init (PID 1) process. *IPC* namespace ensures interprocess communication isolation. This is useful for running for e.g. PostgreSQL in multiple containers. *Network* namespace provides isolation for network resources. Each instance of namespace can have different network interfaces. *Mount* namespace allows to a process to isolate its mount points. *UTS* namespace allows each container to have its own hostname and NIS domain. *User* namespace isolates user and group ID number spaces. The most interesting use-case is the possibility to run privileged process inside namespace without root rights outside the namespace. [13]

Resource constraints can be configured with control groups (cgroups). *Cgroups* provide ways to measure and limit resources for groups of processes. The group of isolated processes creates so called container. With *cgroups*, developer is able to limit number of resources such as RAM, CPU, I/O operations and other process resources. [14]

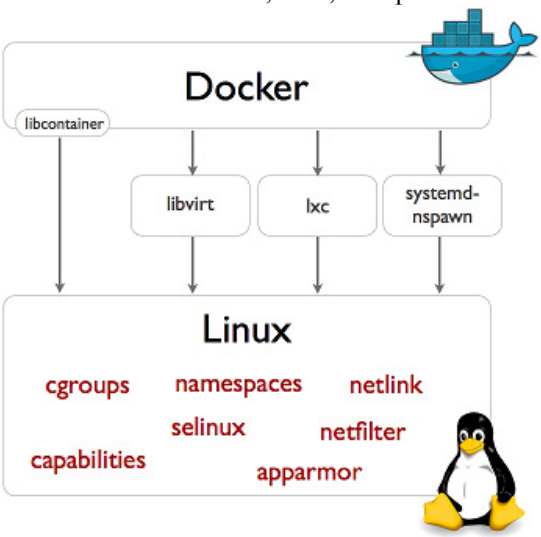


Fig. 1. Docker architecture [15].

A security of containers created with Docker can be improved with MAC mechanisms like SELinux or AppArmor. [16] Interaction with Docker containers is possible via CLI or REST API [17]. By default, Docker daemon is listening on Unix socket. Unix socket is a file, accessible only by privileged user. To make daemon REST API reachable for unprivileged user, it is necessary to configure Docker daemon to listen on TCP. This can be done with command below:

```
sudo docker -H=tcp://127.0.0.1:4243 -d
```

The access to TCP is required for straightforward implementation in Java owing to the fact that native Unix socket support is unavailable in Java platform.

Another proficient feature of Docker platform is the Dockerfile. Dockerfile is script file which describes how a container image will be built. Dockerfile provides set of instructions to simplify the build process. All available commands are conceived to create image for standalone containers. Docker uses advanced versioning filesystem AUFS [18] as the image store. The original image is stored only once; all containers which use this image, are only represented as diff files to the base image. The container image can be built from Dockerfile by following command:

```
sudo docker build -t images/apac .
```

Complete documentation for Dockerfile and Docker itself is available at [11].

## 4. APAC

APAC is acronym for Automatic Programming Assignment Checker. Main focus of this system is to create stand-alone and flexible system for automatic grading in programming courses with isolated testing environment. E-learning systems have significant impact to today's university education. [19] To extend these systems with automatic submissions evaluation, the current version of APAC was designed to provide API for simple integration. APAC is developed under GPLv3 license. APAC is Java EE application built around Docker platform. Architecture overview is given in Fig.2.

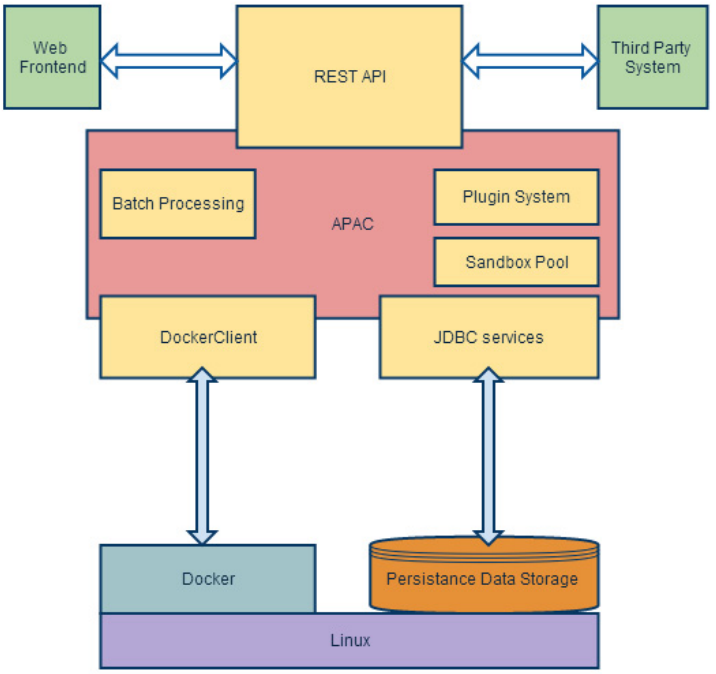


Fig. 2. APAC architecture.

The current version of APAC is developed in Java EE full profile without any other framework. Interaction with Docker containers is possible via CLI or REST API. This functionality is implemented as DockerClient component. Communication with REST API of Docker is encapsulated with Apache HttpClient library [20].

The persistence storage is represented by MySQL relational database. Operations with database are implemented using JDBC. Database provides storage for assignments configuration, test vectors and information about students' submissions. Submitted tasks are processed sequentially from input queue. Queue items processing was implemented with help of JSR 352 [21], which describes batch processing for Java EE 7. For submissions execution, a batch job was created. The job is defined by XML description file and contains all positive and negative cases of process flow. Structure of XML job description is provided by JSR 352.

APAC contains a plugin subsystem, which provides SPI (Service Provider Interface) for custom submission processor implementations. Implementation classes packaged as jar archive can be auto-deployed on to running APAC instance.

Data flow between Docker container and APAC application is realized through SSH connection. Inside each container instance, the SSH daemon is running. Docker creates private LAN between container instances. When container is successfully created, a SSH session to the sandbox is established. This session is held in the sandbox pool. This pool is implemented using Apache Commons Pool library [22]. A status of sandbox SSH session is

checked during borrow phase from the pool. If SSH connection is down, session is removed from the pool and container is deleted. The pool is configured to hold minimum count of valid SSH sessions. If current instance count is lower, new container and new SSH session is created. Compiled submission files are sent to a container through SCP protocol. Execution command is submitted via exec channel of the SSH connection. SSH functionality in Java is provided by JSch library [23]. Submission is processed in following (simplified) steps:

1. submission files normalization,
2. copy normalized submission files into working directory,
3. choose appropriate processor plugin,
4. compile source code files on host,
5. acquire idle sandbox instance,
6. copy compiled files to sandbox container,
7. execute program inside sandbox container,
8. read STDIN and STDOUT from executed program,
9. calculate total points for submission.

Sandbox container image is built from custom Dockerfile. This image contains everything required to evaluate the compiled submission. Container is started with limited resources. By default, APAC sandbox is restricted to 100 megabytes of RAM and maximum 50 percent of CPU usage. These limits are achieved by Docker REST API. Sandbox instance is started with following JSON:

```
{
 "Image": "frantiseks/apac",
 "Memory": 104857600,
 "CpuShares": 50,
 "Tty": true
}
```

Students' programs are tested against defined set of test vectors. APAC supports four types of vector: *input file, string, output file, STDIN file*. The execution command is built from defined vectors; additional files are also selected based on this definition. The test vectors are stored in database as JSON string. Sample JSON is below:

```
{
 "id":1,
 "output":"...",
 "points":10,
 "arguments": [
  { "name":"<",
    "type":"STDINFILE",
    "value":"input1.txt" }
 ]
}
```

Types of vector are determined by lecturers' requirements and courses specifications.

### *4.1. Programming language support*

Reference plugin implementations are provided for C, C++ and Java. Each implementation has to override method for compilation, execution and score calculation. In reference plugins, Levenshtein distance algorithm is used [24] to evaluate differences between submission output and reference output. In case of C and C++, Valgrind

[25] output is included when calculating the score. Score calculation is dependent on plugin's author. It is possible to implement advanced static source code analysis [26] or dynamic runtime analysis with any Linux compatible debugger tools. The results from this analysis can provide better grading algorithm.

## 5. Conclusion

Automatic evaluation systems gain more significance thanks to the increasing industry demand for skilled programmers. An evaluation system can considerably improve quality of course participants. Students are forced to produce error-free and efficient solutions for every assignment. The main requirement for automatic programming submission evaluation should be the secure and isolate testing environment from another parts of system, because students' source code can be malicious and dangerous.

This feature is not implemented by the many systems available for use. APAC was designed to address these demands. Docker platform ensures isolation of testing sandbox and also provides flexibility of sandbox environment. It is possible to create almost every runtime environment for submission compilation and execution. Docker platform provides mechanism to share host devices with container environment. This feature can be used for testing parallel programming assignments based on Nvidia CUDA platform [27]. Only Linux itself limits the sandbox. Another limitation can be implementation of submission processor plugin. It is entirely on plugin's author how robust submission processing will be. The desired runtime environment has to be compatible with Linux and has to be runnable inside isolated container. The sandbox image is built by custom Dockerfile. On top of the sandbox platform, APAC web application provides straightforward REST API for third party integration and plugin API for adding of new submission processors.

The proposed system has considerable potential for future development. Submission evaluation is only the core part of possible complex system for the usage in programming courses. Significant component of such system is the source code originality control. In the first version [12] of our system, YAP [28] tool was used; unfortunately it is too outdate to meet today's demands. For a plagiarism module more research is needed, however based on our preliminary analysis we are planning to implement this module as a multilayered detection mechanism [29], to improve pattern matching results. Another proficient feature can be intelligent automatic feedback module. This module should automatically shows to students, which part of their source code can be improved.

Currently the project is actively developed. The current release of APAC depends on application server Wildlfy [30] developed by RedHat, however this can be changed in future. Project source code can be found at [31]. Binary builds are currently not provided.

## Acknowledgements

This project was supported by the Ministry of Education of the Czech Republic under a FRVŠ project 1542/2013/G1.

## References

[1] Douce, Christopher; Livingstone, David; Orwell, James. Automatic test-based assessment of programming: A review. Journal on Educational Resources in Computing (JERIC), 2005, 5.3: 4.

[2] Forsythe, George E.; Wirth, Niklaus. Automatic grading programs. Communications of the ACM, 1965, 8.5: 275-278.

[3] Queirós, Ricardo; LEAL, José Paulo. Programming Exercises Evaluation Systems-An Interoperability Survey. In: CSEDU (1). 2012. p. 83-90.

[4] Caiza, J. C.; Del Alamo, J. M. Programming Assignments Automatic Grading: Review of Tools and Implementations. Inted 2013 Proceedings, 2013, 5691-5700.

[5] Leal, José Paulo a Fernando Silva. Mooshak: a Web-based multi-site programming contest system. Software: Practice and Experience. 2003, vol. 33, issue 6, s. 567-581. DOI: 10.1002/spe.522. Available from: http://doi.wiley.com/10.1002/spe.522

[6] Security-Enhanced Linux: Chapter 2. Introduction. In: Red Hat Customer Portal [online]. 2014 [accessed 2014-11-15]. Available from: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/chap-Security-Enhanced_Linux-Introduction.html

[7] User Limits Made Easy In: ServerWatch. [online]. 2011 [accessed 2014-11-15]. Available from: http://www.serverwatch.com/server-tutorials/set-user-limits-with-pamlimits-and-limits.conf.html
[8] Malan, David J. Cs50 sandbox: secure execution of untrusted code. In: Proceeding of the 44th ACM technical symposium on Computer science education. ACM, 2013. p.141-146.
[9] MAREŠ, Martin; Blackham, Bernard. A new contest sandbox. Olympiads in Informatics, 2012, 6: 100-109.
[10] LXC: Linux Containers [online]. 2014 [accessed 2014-11-15]. Available from: https://linuxcontainers.org/
[11] Dotcloud INC. Docker Documentation [online]. 2014 [accessed 2014-11-15]. Available from: https://docs.docker.com/
[12] Pohuba, D.; Dulik, T.; Janku, P., "Automatic evaluation of correctness and originality of source codes," Microelectronics Education (EWME), 10th European Workshop on , vol., no., pp.49,52, 14-16 May 2014
[13] Namespaces in operation, part 1: namespaces overview. In: LWN [online]. 2014 [acessed 2014-11-15]. Available from: http://lwn.net/Articles/531114/
[14] Chapter 1. Introduction to Control Groups (Cgroups). In: RedHat Customer Portal [online]. 2014 [accessed 2014-11-15]. Available from: https://access.redhat.com/site/documentat-ion/en-US/Red_Hat_Enterprise_Linux/6/ html/Resource_Management_Guide/ch01.html
[15] Docker 0.9: Introducing Execution Drivers and Libcontainer. In: Docker Blog [online]. 2014 [accessed 2014-11-15]. Available from: http://blog.docker.io/2014/03/docker-0-9-introducing-execution-drivers-and-libcontainer/
[16] Overview. AppArmor Linux Application Security [online]. 2014 [accesed 2014-11-15]. Available from: https://www.suse.com/support/security/apparmor/
[17] Richardson, Leonard, Mike Amundsen and Sam Ruby.Rest fulWebAPIs.S.l.: O'Reilly Media, 2013. ISBN 978-144-9358-068.
[18] Petazzoni, Jérôme A Sam Alba. Dotcloud INC. PaaS Under The Hood [online]. 2013. [accessed. 2014-11-15]. Available from: https://www.dotcloud.com/static/paas_under_the_hood_printversion.pdf
[19] Fenollera, Maria; Goicoechea, Itziar. E-Learning Applied To The University Education In Engineering. Annals of DAAAM & Proceedings, 2011.
[20] HttpClient Overview. Apache HttpComponents [online]. 2014 [accessed 2014-11-15]. Available from: https://hc.apache.org/httpcomponents-client-ga/index.html
[21] JSR 352. Batch Applications for the Java Platform. 1. ed. USA: Oracle America, Inc. 2014. Available from: http://download.oracle.com/otndocs/jcp/batch-1_0-fr-eval-spec/
[22] Overview. Apache Commons Pool [online]. 2014 [accessed 2014-11-15]. Available from : https://commons.apache.org/proper/commons-pool/
[23] JSch - Java Secure Channel. JCraft [online]. 2014 [accessed 2014-11-15]. Available from: http://www.jcraft.com/jsch/
[24] Algorithms and Theory of Computation Handbook, CRC Press LLC, 1999, "Levenshtein distance", in Dictionary of Algorithms and Data Structures [online], Vreda Pieterse and Paul E. Black, eds. 22 August 2013. [accessed 2014-11-15] Available from: http://www.nist.gov/dads/HTML/Levenshtein.html
[25] Valgrind Home [online]. 2014 [accessed 2014-11-15]. Available from: http://valgrind.org/
[26] Striewe, Michael; Goedicke, Michael. A Review of Static Analysis Approaches for Programming Exercises. In: Computer Assisted Assessment. Research into E-Assessment. Springer International Publishing, 2014. p. 100-113.
[27] About Cuda. Nvidia Cuda Zone [online]. 2014 [accessed 2014-11-15]. Available from: https://developer.nvidia.com/about-cuda
[28] WISE, Michael J. YAP3. ACM SIGCSE Bulletin. 1996-03-01, vol. 28, issue 1, s. 130-134. DOI: 10.1145/236462.236525. Available from: http://portal.acm.org/citation.cfm?doid=236462.236525
[29] POON, Jonathan YH, et al. Instructor-centric source code plagiarism detection and plagiarism corpus. In: Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education. ACM, 2012. p. 122-127.
[30] About Wildfly. REDHAT, Inc. Wildfly [online]. 2014 [accessed 2014-11-15]. Available from: http://wildfly.org/about/
[31] APAC Repository. Bitbucket [online]. 2014 [accessed 2014-11-15]. Available from: https://bitbucket.org/frantiseks/apac