applied sciences

MDPI

Article

A Pilot Experience with Software Programming Environments as a Service for Teaching Activities

Amanda Calatrava Arroyo *,†, Marcos Ramos Montes † and J. Damian Segrelles Quilis †

Instituto de Instrumentación para Imagen Molecular (I3M), Universitat Politècnica de València (UPV), 46022 Valencia, Spain; marramon@inf.upv.es (M.R.M.); dquilis@dsic.upv.es (J.D.S.Q.)
* Correspondence: amcaar@i3m.upv.es
† These authors contributed equally to this work.

Abstract: Software programming is one of the key abilities for the development of *Computational Thinking* (CT) skills in Science, Technology, Engineering and Mathematics (STEM). However, specific software tools to emulate realistic scenarios are required for effective teaching. Unfortunately, these tools have some limitations in educational environments due to the need of an adequate configuration and orchestration, which usually assumes an unaffordable work overload for teachers and is inaccessible for students outside the laboratories. To mitigate the aforementioned limitations, we rely on cloud solutions that automate the process of orchestration and configuration of software tools on top of cloud computing infrastructures. This way, the paper presents ACTaaS as a cloud-based educational resource that deploys and orchestrates a whole realistic software programming environment. ACTaaS provides a simple, fast and automatic way to set up a professional integrated environment without involving an overload to the teacher, and it provides an ubiquitous access to the environment. The solution has been tested in a pilot group of 28 students. Currently, there is no tool like ACTaaS that allows such a high grade of automation for the deployment of software production environments focused on educational activities supporting a wide range of cloud providers. Preliminary results through a pilot group predict its effectiveness due to the efficiency to set up a class environment in minutes without overloading the teachers, and providing ubiquitous access to students. In addition, the first student opinions about the experience were greatly positive.

Keywords: cloud computing; software programming; STEM education; learning environments; software as a service

Citation: Calatrava Arroyo, A.; Ramos Montes, M.; Segrelles Quilis, J.D. A Pilot Experience with Software Programming Environments as a Service for Teaching Activities. *Appl. Sci.* **2021**, *11*, 341. https://doi.org/10.3390/app11010341

Received: 26 November 2020
Accepted: 22 December 2020
Published: 31 December 2020

1. Introduction

Nowadays, some fundamental skills to develop in STEM (Science, Technology, Engineering and Mathematics) are those associated with the **Computational Thinking (CT)** [1,2]. These skills underlie from computer science [3] and allow students to develop abstract and lateral thinking, capable of generating multiple, creative and ingenious solutions to the same problem.

Developing and effectively evaluating CT skills is a challenging task due to the fact that, ideally, instruments (software tools) should be the same as in professional atmospheres, bringing students closer to realistic environments and enhancing the teaching-learning process [4–6]. Although there are other solutions in the literature such as the work in [7], that apply a specific developed software in the class to evaluate CT skills, they do not cope with the usage of a professional environment in the class. These instruments are focused on industry software production and not for teaching environments; hence, their configuration and orchestration are barriers for both instructors and students, even more if we are talking about individualised environments focused on students. Teachers have to assume a work overload that in many cases is unaffordable, and in many times it is difficult for students to access these configured and integrated tools outside of class laboratories due to their specific settings, especially in on-line scenarios.

The instruments commented about above are mainly version control repositories (e.g., Git, a free and open source distributed version control system. (https://git-scm.com/), Apache Subversion (https://subversion.apache.org/)), testing and continuous integration tools (CI/CD *Continuous Integration and Continuous Delivery*) (e.g., Jenkins (https://jenkins.io/), GitLab CI (https://about.gitlab.com/stages-devops-lifecycle/continuous-integration/)) and IDE (*Integrated Deployment Environments*) (e.g., DevC ++ (https://sourceforge.net/projects/orwelldevcpp/), code:Blocks (http://www.codeblocks.org/)) among others. The use of all these specific tools allow the teaching community to implement learning methodologies in which students are forced to immerse themselves in real software development environments.

The motivation of this work is to develop and provide an educational tool to the educative community which allows to create environments that simulate professional software production infrastructures for didactic purposes, without involving a work overload for teachers. In addition, we want this environment to be accessible in a ubiquitous way and with minimal requirements, allowing students to use it not only in the class but in any place at any moment.

Cloud technologies have already demonstrated the benefits of their use in higher education [8], both for teachers and students. These benefits include: (i) the ability of the cloud to provide automated deployments of specific instruments, software and hardware orchestrated and configured ad-hoc for an educational activity [9]; and (ii) the possibility for students to access the infrastructure and software in an ubiquitous way with 24/7 accessibility from their own devices, since the cloud provides the computing and storage capacity, and not the devices themselves [10]. Therefore, our hypothesis is that cloud technology is appropriate to develop the aforementioned educational tool.

The main contribution of this paper is **ACTaaS (Assessment of Computational Thinking as a Service)**, a cloud-based educational tool that eases and reduces the workload to deploy and configure a set of professional instruments for software programming on the cloud (private, public or federated) in an economic, simply and quickly way. These instruments are automatically configured and orchestrated to be used in teaching environments, and students can access them in a ubiquitous way. The paper presents the ACTaaS architecture, how all its functionalities have been implemented and what cloud-based tools have been employed to develop the educational tool. Moreover, we include the results of an initial experience in a pilot group of 28 students where all the functionalities of ACTaaS have been tested. Currently, as far as the authors know, there is no similar tool like ACTaaS that allows such a high grade of automation for the deployment of software production environments focused on educational activities supporting a wide range of cloud providers. The fact that the academic community has openly accessible an educational tool like ACTaaS will allow an improvement in the learning processes related to the acquisition of CT skills, since it will let students to use professional software production environments identical to the ones used in the industry. Numerous studies, such as fluid mechanics [6], electrical engineering [5] or higher education in general [11] have demonstrated that combining real scenarios and appropriate teaching methods improves the learning process.

The structure of this paper starts with this introduction. Next, we review the related works in the area of education, and also analyze the state of the art related to cloud orchestration, because tools in the area will be of the key components in ACTaaS. Then, we present and analyze the architecture and utilities offered inside the ACTaaS solution. We pay special attention to the actions that both the teacher and the students have to do to integrate ACTaaS in the class. Finally, we present the preliminary results of its adoption in a real class environment with a pilot use case, to finish the paper with the conclusions and future plans of this work.

2. Related Work

In this section, we analyze previous works in the literature that promote the usage of DevOps, CI/CD tools and techniques in the educational environment, that is of special interest for the industry [12]. To start with an example, the work in [13] presents a survey in the integration of Test-driven development at university education, considering that testing is a fundamental practice. The work [14] exposes their experience in the usage of automated unit testing in practical sessions of programming courses. Finally, the work [15] presents a methodology based on the agile test-driven approach, that was introduced in the class to establish a industrial environment to the students. Some works ([14,16,17]) have developed their own solutions to incorporate automated testing tools in class, but this has the disadvantage that students are not familiar with professional tools.

There are also more recent works about the usage of CI/CD tools in the class, like [16]. It shows their experience in the usage of Github in MOOCs, or [18] that it illustrates their experience with Github and Github Classroom in programming courses of a software engineering program. The work in [19] studies how to teach an Agile Development Methodology using a DevOps approach in a software engineering degree. Finally, the work in [20] proposes a solution based in Jenkins and GitHub to automatically assess practical sessions. This has been successfully adopted by more than 3.000 students, which reinforces the viability of the solution proposed in this article, that also uses both well-known tools.

However, the main problem to adapt such solutions to educational scenarios is the technical complexity they introduce, together with the limited time and resources that teachers usually have. This can be mitigated with solutions that automate the process of configuring the environment required to deploy the tools, on top of a cloud computing infrastructure. As far as the authors know, most of the works in the literature lack the automation of the deployment and configuration of the proposed solutions, although some of them already rely on a cloud computing environment [8]. ACTaaS brings students closer to professional environments by including specific instruments in the classroom workflow. It automates the process of deploying and configuring the environment in the cloud, reducing the effort of the instructors considerably and allowing an easy access to the students from their own devices.

3. State of the Art in Cloud Orchestration

To successfully apply ACTaaS in a real class environment, we need a tool to mitigate the teacher overload and produce the deployment and configuration of the environment in the cloud. The concept **cloud orchestration** refers to the process to automate the entire life-cycle of an application in a cloud infrastructure. It involves the deployment of all the computing resources, the installation and configuration of the different components of the application and its correct interconnection. To describe cloud applications, the OASIS consortium defined the Open Standard of the Topology and Orchestration Specification for Cloud Applications (from its acronym in English, TOSCA) [21], thus allowing portability and automated management between cloud providers, regardless the underlying platform or infrastructure.

There are several open source orchestration tools and services on the market, as reflected in this review [22]. As an example, we can cite tools such as Cloudify [23] Starcluster [24], Elasticluster [25] and Apache ARIA [26], and projects such as OpenTOSCA [27], CELAR [28] and CompatibleOne [29]. However, most of them have limitations regarding the support of the wide range of providers existing in cloud technology and not all of them support the TOSCA standard.

Our previous work in the area, the Infrastructure Manager (IM) [30] is a tool that facilitates the deployment and configuration of complex infrastructures on cloud platforms. The IM is compatible with most cloud providers, both public (i.e., Amazon Web Services (AWS) or Microsoft Azure), and private (i.e., Openstack or Opennebula), among others, and provides its own description language for users to build recipes in order to describe the infrastructure and configuration required. IM supports TOSCA for the definition of the topology, and integrates Ansible [31], to achieve the installation and configuration of all applications required by the user, providing a fully functional infrastructure. Because of its support of a wide range of cloud providers, the simplicity of its usage, and the compilation with the standards, IM is the tool we adopted for ACTaaS to deploy the required infrastructure in the cloud.

4. ACTaaS

As we mentioned previously, ACTaaS is the solution we propose in this work. ACTaaS facilitates the deployment and configuration of a CI/CD environment in the classroom workflow by relying on cloud computing resources and techniques. The main goal is to help teachers to bring closer the professional software developing environment to the students with a reasonable effort, and to have it accessible for all of them from their own devices. ACTaaS is supported by several well-known open source tools and services like GitHub, GitHub Classroom and Jenkins. The solution is available as open source-code in Github (https://github.com/grycap/ACTaaS).

The **classroom workflow** for which ACTaaS was conceived is represented in Figure 1. The figure shows the general architecture of the work methodology proposed to be applied when using ACTaaS. It highlights the main actions to be carried out by both the teacher (in orange) and the students (in green). We analyze deeply each of these actions in the subsections below.

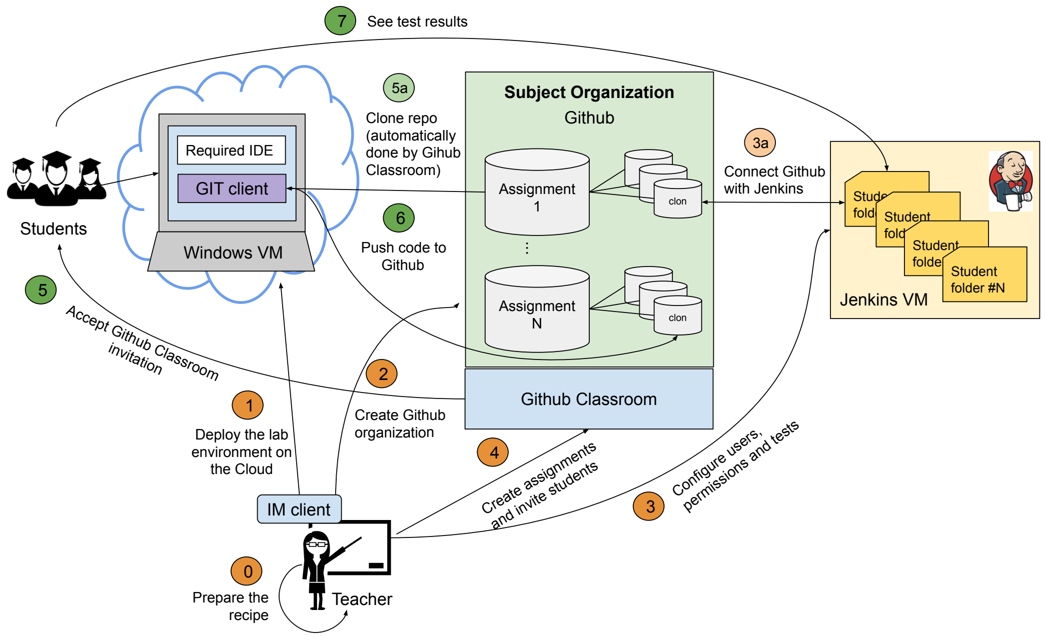

7 See test results
5a Clone repo (automatically done by Gihub Classroom)
Subject Organization
Github
Required IDE
GIT client
Windows VM
Students
Assignment 1
Assignment N
clon
3a Connect Github with Jenkins
Student folder #N
Jenkins VM
6 Push code to Github
5 Accept Github Classroom invitation
Github Classroom
2 Create Github organization
1 Deploy the lab environment on the Cloud
4 Create assignments and invite students
3 Configure users, permissions and tests
IM client
0 Prepare the recipe
Teacher

Figure 1. Architecture of the proposed solution. In orange we represent the actions to be done by the teacher and, in green, the actions to be done by the students.

4.1. Designing the Software Production Environment

The initial step to use ACTaaS is to define the production environment the teacher requires for the lessons. This supposes to determine which tools the students will need to use, like the desired IDE, and the Jenkins server version we want to use. Moreover, the teacher has to determine the hardware and software characteristics of the machines, the network configuration needed to offer connectivity to the Virtual Machines (VMs) and the amount of machines he or she will need for all the students, to then prepare a recipe for the IM. This can be considered as the *step 0*, where the teacher analyses the requirements and builds a recipe that contains them. ACTaaS and IM offer a set of predefined templates (like a Jenkins server instance, or Linux and Windows machines with different software tools) to create this recipe, so the teacher does not need to start from scratch. Figure 2 presents a schema of the typical structure of this recipe.

It is important to notice that, thanks to this flexibility, the proposed solution is fully customizable. Although we present a possible implementation in the next section for our pilot use case, involving a specific IDE and specific software tools, ACTaaS can be easily adapted to other class environments and needs. Teachers just need to prepare a customized recipe for their specific subject to start using ACTaaS in their class.

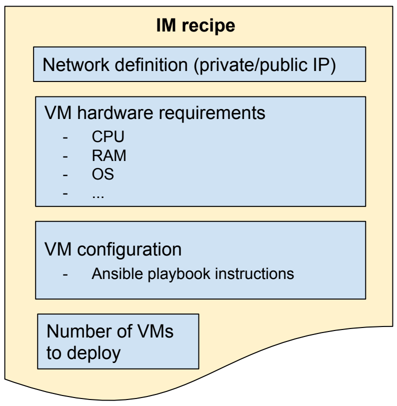


Figure 2. Schema of the structure of a recipe for the Infrastructure Manager.

4.2. Deployment of Specific Software Instruments in the Cloud

Now, the teacher has to deploy in the cloud the specific software instruments required for the educational activities (*step 1*). This includes the deployment of a VM per set of students configured with the necessary software tools to directly work on their assignments; and a VM that acts as the Jenkins server, to run the tests on the delivers of the students.

Thanks to the IM tool, ACTaaS offers a solution to set the environment in the cloud with just one command. As we explained previously, the teacher just needs to customize the predefinded recipes and use the IM client to deploy and configure the environment in the cloud. IM also needs the specific credentials to access the cloud provider preferred by the teacher. On-premises clouds such as OpenStack and OpenNebula, and public clouds such as AWS or Microsoft Azure, among others, are supported.

4.3. Creating the GitHub Organization

Once the instruments are deployed and configured, it is highly recommended to configure the space that will be used for the academic course on GitHub. In addition to user accounts, GitHub offers organization accounts, which represent a group of users who share projects, and manage them as work teams. Thus, the teacher can create a specific organization on GitHub for his/her classroom that will serve to collect the practice repositories of all the students in a centralized way (*step 2*). Moreover, we recommend to configure the organization to allow private repositories, so students can only see their own developments.

4.4. Configuring Jenkins

In *step 3*, the teacher has to configure the Jenkins server previously deployed to test the assignments of each student automatically. This involves different actions to be carried out in Jenkins: (i) create a user account in for each student; (ii) create a hierarchy of folders, corresponding with each lab session, per student; (iii) configure a test per exercise assignment linked to the corresponding repository of each student; and (iv) perform an appropriate configuration of permissions so that students only have access to their own assignment tests and not to those of other classmates.

To mitigate the overload that these actions can produce, ACTaaS offers the following utilities:

- A script (*registerStudents.sh*) to create the users, the folder structure and the corresponding role permission settings for a list of students. Thus, the script automates the creation of users in Jenkins and sends each student their login credentials. The management of the permissions of each user is done through roles using the Jenkins

Role Based Authorization Strategy plugin to create and assign roles for each user and folder. This script only needs to be executed once per course, by giving as input the number of assignments of the subject and the list of the students, so it creates the necessary folder structure that will keep the different tests per exercise of each scheduled practical session.

- Utilities in charge of creating, for each practice, the tests (*jobs* in Jenkins) that will evaluate each exercise of the assignments. These are: (i) a script (*createTests.sh*) that receives a file containing the list of GitHub users along with the URL of the repository of the practice for which the tests will be created; (ii) an auxiliary script (*get_repos_users_file.sh*) to generate the above file with just a command, which provides the list of URLs of the repositories within an organization that comply with a given prefix; (iii) a Jenkins Job template in XML format from which the tool will configure the tests for each student; and (iv) the code tests for each exercise. By using these utilities, the connection between Jenkins and each one of the students Github repositories is done (*step 3a*). Notice that each test will be created appropriately within the folder structure created by the previous script, so that each student can only see his/her own results.

Notice that all these scripts only work with Jenkins. In other scenarios where other CI/CD tools might be used, like Travis CI, new scripts with the same functionality will be required.

4.5. Creating Assignments in GitHub Classroom

Next step (*step 4*) is to create assignments for each of the practical sessions of the subject/course using GitHub Classroom. This platform assists both the teacher and the student by automating the creation of repositories and access control to them, thus facilitating the distribution of the initial code of the practices and the collection of their developments. To guide the student in each practice, the teacher can create a "template" repository in GitHub that will contain the initial code of the exercises, serving as a starting point for the students for each one of the assignments. Thus, the teacher will create an assignment for each practice of the subject, indicating the template repository with the initial code of the task. The teacher will obtain a link that must be sent to the students so that they can accept the invitation to the assignment. This assignment can be individual or by groups.

4.6. Accepting the Assignment

In *step 5* the student accepts the assignment invitation (they simply access the link provided by their teacher). The template repository of the practical session corresponding to the assignment is automatically cloned as a private (if activated) repository in the student's GitHub account (*step 5a*), within the organization initially created by the teacher. GitHub Classroom provides a mechanism for the teacher, called roster, that allows students' GitHub accounts to be associated with a list of names or emails to facilitate the identification of the authorship of each repository.

4.7. Developing the Assignments

Finally, students access to their assigned VMs using tools like a remote desktop. Students will be able to develop their practical sessions integrating the GitHub workflow into them, cloning their repository locally, developing the exercises for each assignment, uploading the changes to the repository (*step 6*) and accessing Jenkins to receive automatic feedback and check the test results. These tests in Jenkins are configured to run every time a student push changes to his/her repository, or they can be run on demand when the student connects to the Jenkins interface and explicitly requests it from the Jenkins GUI (*step 7*).

The results of the Jenkins tests will serve the students to know the success in each assignment exercise with a continuous feedback approach. It also facilitates the teacher the evaluation of the proposed tasks. Also, it is important to highlight that the students can use their own devices for accessing the VM where all the hardware, software and configuration of the employed instruments are provided by the cloud.

As summary, Table 1 lists the main actions of the classroom workflow to be performed by the teacher and by the students, as well as the support provided by ACTaaS to both roles.

Table 1. Summary of actions to be carried out by the teacher and the students to implement a CI/CD environment in class and the support provided by the proposed solution.

Actions	Teacher	Students	How It Is Assisted?
Designing the software production environment	X		Partially by ACTaaS
Deployment of specific software instruments in the cloud	X		Completely by ACTaaS
Configure Jenkins	X		Completely by ACTaaS
Creating the GitHub organization	X		By GitHub web service
Creating assignments in GitHub Classroom	X		By GitHub Classroom
Accepting the assignment		X	By a mail client and Github Classroom
Do the assignment on a CI/CD environment		X	By ACTaaS and GitHub Desktop
Auto analyze the progress of the assignments		X	By ACTaaS and Jenkins
Evaluate the assignments	X		Partially by Jenkins test outputs

5. Pilot Use Case

In this section, we present an example use case for ACTaaS, specifying in each of the next subsections all the steps performed to create the software programming environment in the class. These subsections match the ones presented in the section above, to facilitate the comprehension of the steps required to use ACTaaS.

We started to use ACTaaS during the course of 2019–2020, in a Pilot Group composed of 28 students of the first course of the *Electronic and Automatic Engineering Degree (EAED)* at the *Universitat Politècnica de València* (UPV), under the subject "Computer Science". Traditionally, students solved the exercises using DevC++ IDE, that it was installed in all PCs of the laboratory or they could install it in their own devices. In this scenario, the classroom workflow is far from emulating a realistic scenario because they didn't use any kind of version control repositories, testing or CI/CD tools. Therefore, we use ACTaaS to deploy a whole production software environment and change the current classroom workflow to a more realistic approach.

The test has been performed in two practical lessons (lab sessions, each one seen as an 'assignment'), where the students have to design and implement in C standard language a set of programs (structured in 'exercises') both in the classroom and as autonomous work. The rest of the subject assignments have been developed using the traditional approach, in order to be able to compare them.

The objective of this pilot use case is to verify that the deployment of the required environment using ACTaaS is simple and the effort is reasonable for the teacher. We also want to demonstrate that a whole production environment can be configured and be accessible for the students in a easy way, 24/7, to facilitate students the production of software in a CI/CD scenario. Next subsections expose how the identified tasks in Section 4 have been performed to deploy and start using the virtual software programming environment through ACTaaS. Results are presented regarding the analysis of deployment and configuration times, and the students feedback through a short poll.

5.1. Designing the Software Production Environment

In our specific scenario, the instruments that the students need are made up of Windows machines configured with the DevC ++ IDE and a graphical client for GitHub, to facilitate the process of learning Git to the students. For that, we have customized a recipe (*lab-environment.radl*, avaliable at: https://github.com/grycap/ACTaaS/blob/master/im_recipes/lab-environment.radl.) that is responsible for deploying the necessary resources, and automatically installing and configuring the students' lab environment together with the Jenkins instance.

5.2. Deployment of Specific Software Instruments in the Cloud

As described in Section 4.2, the first action is to deploy the required VMs in the cloud using ACTaaS. The process is as easy as executing the next command:

```
$ im_client create lab-environment.radl -a <credentials_file>
```

The command 'create' of the IM client receives two parameters: the 'credentials_file' that must contain valid credentials to access the desired cloud provider and the recipe (or recipes) to configure the VMs. In our case, 'lab-environment.radl' is the deployment recipe we prepared in the first step.

Table 2 presents the average deployment times from our pilot use case. The first column is the average time of deploying just the Jenkins server on an Ubuntu 16.04 image. Column two refers to the same deployment, but here, we have prepared a preconfigured VM image with Jenkins already installed. Column three contains the average time of deploying and configuring a Windows 7 machine with the requested software to carry on the lab sessions. Finally, columns four and five represent the time to deploy a Jenkins server VM plus some Windows VMs (In each Windows VM have assigned 5–6 students). For these tests, we have relied on an on-premises cloud, managed by OpenNebula. The physical infrastructure has two types of nodes: (i) 64GB of RAM memory, two Intel(R) Xeon(R) CPU E5-2683 v3 2.00GHz with 14 cores each, 240GB SSD and (ii) 128GB of RAM memory, two Intel(R) Xeon(R) CPU E5-2660 v4 2.00GHz with 14 cores each and 250GB SSD. The Jenkins VM has been requested with 1 CPU and 4Gb of RAM, while the Windows VMs have 2 CPUs and 8Gb of RAM.

Table 2. Cloud deployment times for the VMs.

Jenkins	Jenkins (preconf.)	Win7	Jenkins + 1 Win7	Jenkins + 5 Win7
380.8 s	250.8 s	766.7 s	936.5 s	1173.1 s

5.3. Creating the GitHub Organization

To manage all GitHub repositories involved in this pilot use case (these are templates for GitHub Classroom assignments and all the student repositories), an organisation for the pilot group named "ARAGroupGIA" was created. In GitHub, the ability to create private repositories imply a cost. To mitigate the involved costs which can suppose a barrier in the adoption of the proposed solution in the academic area, we applied to the 'GitHub Education program' (https://education.github.com/). This program, offered by Github Inc., is conceived to facilitate the access to their services in the educational community. This allowed us to create private repositories without costs, among other benefits.

5.4. Configuring Jenkins

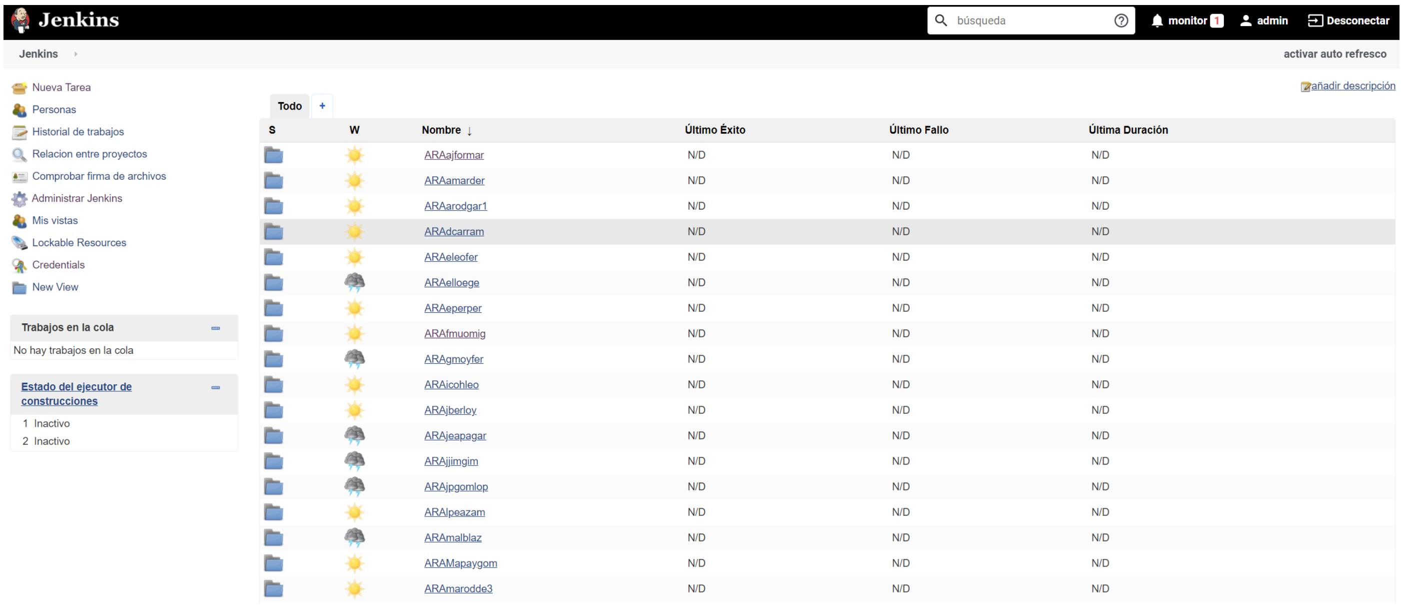
Once the environment has been deployed in the cloud, the next action is to configure Jenkins appropriately. For this purpose, the instructors have to previously create the required unit tests to evaluate the code of each exercise proposed to the students. In our pilot use case, we created bash scripts to verify the outputs of the programs. The scripts for the practice Lesson 1 (assignment 1 with 5 exercises) and practice Lesson 2 (assignment 2 with 5 exercises) are located in the ACTaaS repository (https://github.com/grycap/ACTaaS/tree/master/practices). These unit test can be reused in future courses, thus the overhead of creating them is only considered in the first starting up of the solution.

Another consideration before going on with the next step is that all students need to have an account in Github. They may previously have one, or they must register in Github. Anyway, the teacher will need to know their account names to properly associate them with their Jenkins accounts that we will create later.

Thus, once the whole environment has been deployed, the unit test created and all students are registered in GitHub, we have to configure Jenkins. We will create individual and isolated Jenkins accounts for each student and also create the unit tests in the CI/CD tool for all students accounts. As described in Section 4.4, in this pilot use case we had mainly employed two scripts: (i) one to register the students in Jenkins, together with the initial folder structure; and (ii) another to create the tests for each exercise of the assignments.

```
$./registerStudents.sh -j http://158.42.105.24:8080 -a 2 -u admin -p
 <cred_jenkins>
     -f students.txt
```

When the above command is executed, all student accounts and the folder structure to contain the tests are created in Jenkins, with proper configuration of permissions. The file "students.txt" contains the list of student accounts (GitHub accounts) to add in Jenkins that corresponds to the 28 students of the pilot group. In our pilot use case, we created two folders per student (one per assignment). The Figure 3 shows the Jenkins view from the teacher (or admin) account, where all the created students folders are.



Trabajos en la cola

No hay trabajos en la cola

Estado del ejecutor de construcciones

1 Inactivo
2 Inactivo

S	W	Nombre ↓	Último Éxito	Último Fallo	Última Duración
📁	☀	ARAajformar	N/D	N/D	N/D
📁	☀	ARAamarder	N/D	N/D	N/D
📁	☀	ARAarodgar1	N/D	N/D	N/D
📁	☀	ARAdcarram	N/D	N/D	N/D
📁	☀	ARAeleofer	N/D	N/D	N/D
📁	⛈	ARAelloege	N/D	N/D	N/D
📁	☀	ARAeperper	N/D	N/D	N/D
📁	☀	ARAfmuomig	N/D	N/D	N/D
📁	⛈	ARAgmoyfer	N/D	N/D	N/D
📁	☀	ARAicohleo	N/D	N/D	N/D
📁	☀	ARAjberloy	N/D	N/D	N/D
📁	⛈	ARAjeapagar	N/D	N/D	N/D
📁	⛈	ARAjjimgim	N/D	N/D	N/D
📁	⛈	ARAjpgomlop	N/D	N/D	N/D
📁	☀	ARAlpeazam	N/D	N/D	N/D
📁	⛈	ARAmalblaz	N/D	N/D	N/D
📁	☀	ARAMapaygom	N/D	N/D	N/D
📁	☀	ARAmarodde3	N/D	N/D	N/D

Figure 3. Jenkins view for the teacher with all created students folders.

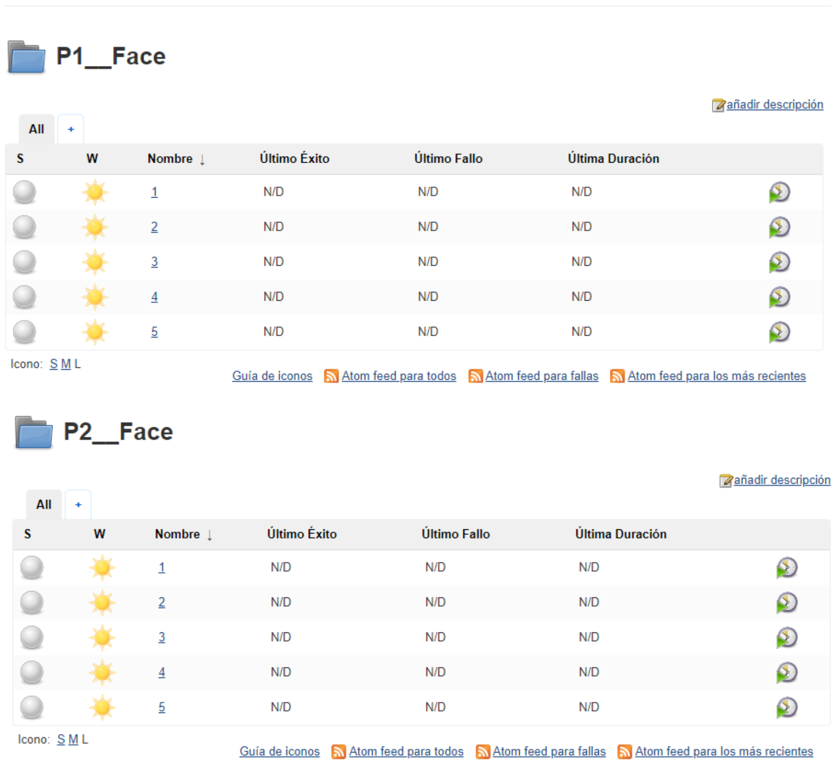
After that, we have executed *createTests.sh* to create the unit test for each exercise. This script has to be executed once per test required, and it creates that test for each student, linked to the proper Github repo in each case. As an example, the next command creates a unit test for exercise '5' in assignment '1'. Hence, for the rest of exercises, as well as with the other assignment, we have executed similar commands indicating the proper exercise number together with the corresponding assignment. The other key parameter is the template of the test to be created. The basic structure of all unit tests is provided by an XML file, *test_template.xml*, where a set of parameters is automatically configured for each exercise. This set of parameters includes: (i) the Github master credentials to access the student repositories where they resolve the exercises; (ii) the student's repository URL, and (iii) the command to execute the test depending on the exercise for launching the unit test. The repositories URL is provided in the file *list_repos.txt*, which can be generated automatically with the auxiliary script (*get_repos_users_file.sh*, available at: https://github.com/grycap/ACTaaS/blob/master/scripts_github/get_repos_users_file.sh) that ACTaaS facilitates to the teacher.

```
$./createTests.sh -j http://158.42.105.24:8080 -a 1 -n 5 -t test_template.
xml -u admin
    -p <cred_jenkins> -f list_repos.txt -i <cred_github>
```

Table 3 presents the configuration times for a variable number of students. We can see that for *registerStudents.sh* the execution time increases linearly with the number of students. Specifically, it takes approximately 0.5 s per student. Regarding the *createTest.sh* script, it requires approximately 0.2 s per student to create one unit test in a given assignment. Thus, to configure Jenkins for our pilot group, we only needed less than five minutes to create the 28 student account and the 280 unit tests required (28 students $\times$ 2 assignments $\times$ 5 exercises). The Figure 4 shows the Jenkins interface from the point of view of a student after creating tests for both Practice Lesson 1 and 2.

Table 3. Configuration times of Jenkins for each script.

N°Students	registerStudents.sh	createTests.sh
10	5531 s	2024 s
20	10,388 s	3096 s
30	15,894 s	4791 s


P1__Face

añadir descripción

All +

S	W	Nombre ↓	Último Éxito	Último Fallo	Última Duración	
●	☀	1	N/D	N/D	N/D	
●	☀	2	N/D	N/D	N/D	
●	☀	3	N/D	N/D	N/D	
●	☀	4	N/D	N/D	N/D	
●	☀	5	N/D	N/D	N/D	

Icono: S M L

Guía de iconos Atom feed para todos Atom feed para fallas Atom feed para los más recientes

P2__Face

añadir descripción

All +

S	W	Nombre ↓	Último Éxito	Último Fallo	Última Duración	
●	☀	1	N/D	N/D	N/D	
●	☀	2	N/D	N/D	N/D	
●	☀	3	N/D	N/D	N/D	
●	☀	4	N/D	N/D	N/D	
●	☀	5	N/D	N/D	N/D	

Icono: S M L

Guía de iconos Atom feed para todos Atom feed para fallas Atom feed para los más recientes

Figure 4. Jenkins view for the practice 1 (**top**) and practice 2 (**down**) which are composed of 5 exercises.

5.5. Creating the Assignments in GitHub Classroom

Github Classroom has been adopted in ACTaaS to facilitate the management of the assignments. For the pilot use case, we created two assignments in GitHub Classroom by using its web interface. We prepared and associated to each assignment two 'template' repositories (available at: https://github.com/grycap/ACTaaS/tree/master/practices), so the students do not start their exercises from scratch. Instead, they receive an initial structure automatically cloned by the service to facilitate their developments.

5.6. Accepting the Assignment

Finally, when all the environment is ready, the students accept the assignments created by GitHub Classroom, causing the creation of private repositories for each student from where they can work solving the exercises. To invite all students to accept the GitHub Classroom assignment, the Learning Management System Sakai (https://www.sakailms.org/) was employed. The teacher uploaded the study guide for the assignment to the platform, where the exercises of each practice lesson are explained together with the URLs to accept the GitHub Classroom assignment.

The development of the exercises is done by using the VM assigned to each student though remote desktop. Students learn how to interact with version control repositories by launching commits and pushes to their own GitHub repository that automatically trigger the execution of unit tests in Jenkins. In the pilot group, students could access the Jenkins instance with their credentials and validate by their own the result of the exercises proposed.

5.7. Feedback from Students

In the first approach of the experience to the students, the general opinion of the group was considered with many doubts when asked about the use of cloud resources and CI/CD tools in their lab environment. However, once we finished the pilot use case, we asked students to fill a brief questionnaire, composed of four questions, to better know their opinion. The questions proposed are statements (shown in Table 4) where the possible answers were related to the level of agreement of the student with each statement. Students could choose one of these options: Strongly Disagree (SD), Rather Disagree (RD), Middle Term (MT), Rather Agree (RA) and Strongly Agree (SA). All the 28 students participating in the pilot group participated in the questionnaire.

Table 4. Questionnaire answered by the pilot use case students after working on the two assignments with ACTaaS. *Strongly Disagree (SD), Rather Disagree (RD), Middle Term (MT), Rather Agree (RA) and Strongly Agree (SA).*

Question	SD	RD	MT	RA	SA
Q1. I have always had available from anywhere the computing resources necessary to develop the practices.	-	-	-	-	-
Q2. In general, solving the proposed exercises through the new workflow seems appropriate and a good method.	-	-	-	-	-
Q3. I think that the use of Virtual Machines on the cloud offered by Remote Desktops can be extended to other subjects.	-	-	-	-	-
Q4. I prefer to use the new workflow in the cloud than the traditional workflow for practical lessons.	-	-	-	-	-

Analysing the obtained results, for the first question (Q1), we can confirm the availability of the VMs at any time during the experience. Almost 80% of the students agreed regarding this fact (RA (57%) or SA (21%)). The students who did not completely agree with this statement said that in some occasions, they have not been able to access the resources, since they did not have good Internet connectivity from outside the university (from external libraries, internet cafes, etc.).

Regarding the second statement about student opinion of the workflow used to develop the assignments (Q2), 92% of them corroborate (26% of RA and 46% of SA) that the use of the new workflow is appropriate and a good method. Thus, we can extract from this question that the proposed solution to approximate real environments to the class is well-received by students who also felt comfortable working with it.

The results of Q3 statement confirm that the use of cloud computing techniques in the educational environment can be applied to a wider scenario. Students felt that the proposed solution could be valuable in other subjects of their degree, with a 32% of RA answers and 54% of SA responses.

Finally, in question 4 (Q4), we can see that there is a high percentage of MT (50%) answers, meaning that at least the student perception is that the new workflow is comparable with the traditional one, and they can consider the two methods as equally valid for carrying out the practical lessons. Moreover, there is an acceptance of more than 30% from students who believe that the the new workflow is better (the sum of RA and SA), and they prefer to work with it. On the other hand, 17% of students prefer the traditional workflow over the new one. Nevertheless, we understand that this fact is positive given that it validates the new method, although modifications and improvements are required to increase student's satisfaction with ACTaaS.

6. Discussion

As we have already mentioned, the main problem to adapt professional solutions to educational scenarios is the technical complexity they introduce, together with the limited time and resources that teachers usually have. This can be mitigated relying on cloud computing techniques together with tools that automate the process of configuring the required environment. As far as the authors know, previous works in the literature that aim to create CI/CD environments in academia, such as [20], lack this automation, thus complicating its adoption in a daily educational atmosphere. ACTaaS automates the process of deploying and configuring the environment, reducing the effort of the instructors considerably and allowing an easy access to the students from their own devices.

Moreover, ACTaaS allows adapting the instruments for any scenario where software production environments are necessary. However, to achieve this customization, new IM recipes have to be built. This might be seen as an initial barrier for teachers, but they can find support for that process in the examples provided under the ACTaaS repository plus the official documentation of IM.

Another obstacle that instructors can find when considering using ACTaaS is the development of unit tests for their educational activities. This is the task that will require the biggest effort from teachers. Nevertheless, it can be assisted by several tools and libraries to facilitate the development of such unit tests, like *JUnit* [32] in the case of Java or the *Check* library [33] in the case of C language. Furthermore, this effort can be reused for the subsequent courses.

From an ACTaaS user point of view, instructors employ a command line interface to perform the actions of deploying the software production environment. However, this fact might cause teachers to discard ACTaaS for those not familiar with this type of interface. It would be interesting to provide them with a web-based interface, user-friendly and intuitive to use in order to guide them through the deployment process. Furthermore, despite having a high level of automation, there are still some actions that require the interactive intervention of the teacher. Our target will be to completely automatise all these actions.

7. Conclusions and Future Work

This work has presented ACTaaS as an open-source educational tool to deploy on the cloud. It is a realistic software production environment for educational activities. The solution facilitates the usage of professional tools for software programming in practical laboratory lessons, allowing students to learn good practices on software engineering. In addition, it provides them with continuous feedback about their developments without the need for interaction with the teacher.

The initial resource deployment is done in a simple and easy way (with just a command) in a few minutes. Then, the required configuration and orchestration of Jenkins for a set of students do not suppose a high workload to the teacher, thanks to the utilities ACTaaS offers. The required accounts and tests are easily created by using two easy commands. The pilot use case has demonstrated that the effort of the instructors is concentrated mainly in the development of assignments (identification of the required resources, description of exercises, etc...) and the creation of unit tests to automatically evaluate student development, and not on the deployment and orchestration of the required software instruments. Moreover, these tasks can be reused in future courses, minimising that initial effort.

Regarding cloud technologies, the cloud production environment deployed for the pilot group has demonstrated to be always available to the students during the experiment. They could work on their assignments at any time and anywhere accessin the same production environment in a transparent way. Thus, cloud resources have proven to be effective in educational environments.

As a last remark, the general opinion of the students is highly positive, demonstrating that the proposed solution of this work, ACTaaS, can be integrated in the classroom workflow with a good range of acceptance among students. However, this is an initial experience in a class. The valuable opinions collected will be considered to further improve and develop the solution for next courses. We plan to develop a graphical user interface for the teacher that includes the scripts and utilities shown, to facilitate even more its usage. Furthermore, we will continue working on improving the unit tests to enrich the feedback students receive. Additionally, it would be interesting as future work to provide new recipes to support the most common learning scenarios inside ACTaaS.

Moreover, we plan to extend the usage of ACTaaS to all the assignments of the subject and deploy new infrastructures for all groups of the degree (around 300 students). In a more distant scenario, we also want to apply ACTaaS in other degrees where software programming is part of the curriculum.

Finally, the capacity to automatically recollect software and usage metrics from CI/CD tools by using specific plug-ins opens an opportunity to establish relations between these metrics and the skills related to the Computational Thinking. This is a crucial point to continue with future ACTaaS developments.

Author Contributions: Conceptualization, A.C.A. and J.D.S.Q.; Funding acquisition, J.D.S.Q.; Investigation, A.C.A., M.R.M. and J.D.S.Q.; Resources A.C.A.; Software, M.R.M.; Supervision, J.D.S.Q.; Validation, J.D.S.Q.; Writing—original draft, A.C.A. and J.D.S.Q.; Writing—review & editing, A.C.A. Authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Conselleria d'Innovació, Universitat, Ciència i Societat Digital for the project "CloudSTEM" grant number AICO/2019/313, and the Vicerrectorado de Estudios, Calidad y Acreditación of the Universitat Politècnica de València grant number PIME/19-20/166.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CT	Computational Thinking
STEM	Science, Technology, Engineering and Mathematics
CI/CD	Continuous Integration and Continuous Delivery
IDE	Integrated Deployment Enviroments
ACTaaS	Assessment of Computational Thinking as a Service
TOSCA	Topology and90Orchestration Specification for Cloud Applications
AWS	Amazon Web Services
IM	Infrastructure Manager
VM	Virtual Machine
UPV	Universitat Politècnica de València

References

1. Khine, M.S. *Computational Thinking in the STEM Disciplines: Foundations and Research Highlights*; Springer: Berlin/Heidelberg, Germany, 2018.
2. Velázquez-Iturbied, J.Á. Towards an Analysis of Computational Thinking. In Proceedings of the 2018 International Symposium on Computers in Education (SIIE), Jerez, Spain, 19–21 September 2018; pp. 1–6.
3. Burbaitė, R.; Drąsutė, V.; Štuikys, V. Integration of computational thinking skills in STEM-driven computer science education. In Proceedings of the 2018 IEEE Global Engineering Education Conference (EDUCON), Tenerife, Spain, 17–20 April 2018; pp. 1824–1832.
4. Hu, C.C.; Tseng, H.T.; Chen, M.H.; Alexis, G.P.I.; Chen, N.S. Comparing the effects of robots and IoT objects on STEM learning outcomes and computational thinking skills between programming-experienced learners and programming-novice learners. In Proceedings of the 2020 IEEE 20th International Conference on Advanced Learning Technologies (ICALT), Tartu, Estonia, 6–9 July 2020; pp. 87–89.
5. Campbell, J.O.; Bourne, J.R.; Mosterman, P.J.; Brodersen, A.J. The effectiveness of learning simulations for electronic laboratories. *J. Eng. Educ.* **2002**, *91*, 81–87. [CrossRef]
6. Fraser, D.; Pillay, R.; Tjatindi, L.; Case, J. Enhancing the learning of fluid mechanics using computer simulations. *J. Eng. Educ.* **2007**, *96*, 381–388. [CrossRef]
7. Troussas, C.; Krouska, A.; Sgouropoulou, C. Collaboration and fuzzy-modeled personalization for mobile game-based learning in higher education. *Comput. Educ.* **2020**, *144*, 103698. [CrossRef]
8. González-Martínez, J.A.; Bote-Lorenzo, M.L.; Gómez-Sánchez, E.; Cano-Parra, R. Cloud computing and education: A state-of-the-art survey. *Comput. Educ.* **2015**, *80*, 132–151. [CrossRef]
9. Segrelles, J.D.; Moltó, G.; Caballer, M. Remote Computational Labs for Educational Activities via a Cloud Computing Platform. In Proceedings of the 2015 Information Systems Education Conference (ISECON), Orlando, FL, USA, 5–7 November 2015; pp. 309–321.
10. Segrelles, J.D.; Moltó, G. Assessment of Cloud-based Computational Environments for Higher Education. In Proceedings of the 2016 IEEE Frontiers in Education Conference, Erie, PA, USA, 12–15 October 2016.
11. Varela-Candamio, L.; García-Álvarez, M.T. Analysis of information and communication technologies in higher education: A case study of business degree. *Int. J. Eng. Educ.* **2012**, *28*, 1301–1308.
12. Moreno, A.M.; Sanchez-Segura, M.I.; Medina-Dominguez, F.; Carvajal, L. Balancing software engineering education and industrial needs. *J. Syst. Softw.* **2012**, *85*, 1607–1620. [CrossRef]
13. Desai, C.; Janzen, D.; Savage, K. A survey of evidence for test-driven development in academia. *ACM SIGCSE Bull.* **2008**, *40*, 97–101. [CrossRef]
14. Barriocanal, E.G.; Urbán, M.Á.S.; Cuevas, I.A.; Pérez, P.D. An experience in integrating automated unit testing practices in an introductory programming course. *ACM SIGCSE Bull.* **2002**, *34*, 125–128. [CrossRef]
15. Bowyer, J.; Hughes, J. Assessing undergraduate experience of continuous integration and test-driven development. In Proceedings of the 28th International Conference on Software Engineering, Shanghai, China, 20–28 May 2006; pp. 691–694.
16. Matthies, C.; Treffer, A.; Uflacker, M. Prof. CI: Employing continuous integration services and Github workflows to teach test-driven development. In Proceedings of the 2017 IEEE Frontiers in Education Conference (FIE), Indianapolis, ID, USA, 18–21 October 2017; pp. 1–8.
17. Hill, J.H. CUTS: A system execution modeling tool for realizing continuous system integration testing. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, Cape Town, South Africa, 1–8 May 2010; Volume 2, pp. 309–310.
18. Angulo, M.A.; Aktunc, O. Using GitHub as a teaching tool for programming courses. In *ASEE Gulf-Southwest Section Annual Meeting 2018 Papers*; American Society for Engineering Education: Austin, TX, USA, 2019.
19. Mason, R.T.; Masters, W.; Stark, A. Teaching Agile Development with DevOps in a Software Engineering and Database Technologies Practicum. In Proceedings of the 3rd International Conference on Higher Education Advances, Valencia, Spain, 21–23 June 2017; Volume 17.
20. Heckman, S.; King, J. Developing software engineering skills using real tools for automated grading. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education, Baltimore, MD, USA, 21–24 February 2018; pp. 794–799.
21. OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA). Available online: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca (accessed on 20 December 2020).
22. Tomarchio, O.; Calcaterra, D.; Di Modica, G. Cloud resource orchestration in the multi-cloud landscape: A systematic review of existing frameworks. *J. Cloud Comput.* **2020**, *9*, 49. [CrossRef]
23. Cloudify. Available online: https://cloudify.co (accessed on 20 December 2020).
24. Massachusetts Institute of Technology. StarCluster. Available online: http://web.mit.edu/stardev/cluster/ (accessed on 20 December 2020).
25. Zurich, U. ElastiCluster. Available online: https://elasticluster.github.io/elasticluster/ (accessed on 20 December 2020).
26. The Apache Software Foundation. Apache ARIA TOSCA Orchestration Engine. Available online: http://ariatosca.incubator.apache.org (accessed on 20 December 2020).
27. University of Stuttgart. OpenTOSCA. Available online: http://www.opentosca.org (accessed on 20 December 2020).

28. Giannakopoulos, I.; Papailiou, N.; Mantas, C.; Konstantinou, I.; Tsoumakos, D.; Koziris, N. CELAR: Automated application elasticity platform. In Proceedings of the 2014 IEEE International Conference on Big Data (Big Data), Anchorage, AK, USA, 27 June– 2 July 2014; pp. 23–25. [CrossRef]
29. Yangui, S.; Marshall, I.J.; Laisne, J.P.; Tata, S. CompatibleOne: The Open Source Cloud Broker. *J. Grid Comput.* **2014**, *12*, 93–109. [CrossRef]
30. Caballer, M.; Blanquer, I.; Moltó, G.; de Alfonso, C. Dynamic management of virtual infrastructures. *J. Grid Comput.* **2015**, *13*, 53–70. [CrossRef]
31. Hat, R. Ansible. Available online: https://www.ansible.com/ (accessed on 20 December 2020).
32. JUnit Team. JUnit Framework for Java. Available online: https://junit.org/ (accessed on 20 December 2020).
33. Arien Malec, E.A. Check Unit Testing Framework for C. Available online: https://libcheck.github.io/check/ (accessed on 20 December 2020).