Artifact: Virtual Environment for Evaluating the QoS of Distributed Mobile Applications

Sergio Laso*, Javier Berrocal*, Pablo Fernández†, Antonio Ruiz-Cortés† and Juan M. Murillo*
*University of Extremadura, Cáceres, Spain
Email:{slasom, jberolm, juanmamu}@unex.es
†I3US Institute, SCORE Lab. Universidad de Sevilla, Sevilla, Spain
Email:{pablofm, aruiz}@us.es

I. Introduction

This artifact is a guideline for the use of Perses. Perses is a framework for evaluating the Quality of Service (QoS) of distributed mobile applications. Perses allows developers to define and deploy customized environments with heterogeneous virtual mobile devices in order to simulate the deployment of these applications and analyze the monitored QoS attributes. Perses has been integrated into a DevOps methodology, automating its execution before its deployment in a real environment.

II. Perses Setup

This section explains how to configure Perses to evaluate a distributed mobile application. The main requirements are an account on Github [1] and Amazon Web Service (AWS) [2]. GitHub is required to maintain a repository storing the developed source code of the distributed mobile application. Every new commit launch the virtual environments defined with Perses by means of GitHub Actions [3]. AWS is used by Perses as an infrastructure provider to host all the virtual mobile devices to be deployed. In addition, Perses requires high-capacity AWS *.metal* instances. These enable the nested virtualisation required for Android device virtualisation. This artifact will use C5.metal (Perses takes care of managing it automatically) so you should verify if your account can instantiate that kind of instances before starting the process.[1]

To test Perses, a distributed mobile application is required. Currently, the mobile side is only compatible with Android mobile applications. In order to facilitate the evaluation of Perses, the distributed mobile application of the use case explained in [4] is provided.

Both components are accessible in public repositories [2],[3].

The process that should be followed to configure Perses and evaluate the application's QoS is:

1) **Deployment of the server-side:** To get started, it is necessary to deploy the server-side. In its repository (*Readme.md*) are the instructions for the deployment.
2) **Create the application repository:** It is also necessary to create a personal repository on GitHub to upload the mobile application. Please, note that in order to run gradle-based applications, the gradlew script must have execution rights [4]). In order to ease the testing of Perses, just a fork or a clone of the provided repository can be created that already has the execution rights.
3) **Virtual environment definition:** A configuration file, called *.perses.yml*, must be created in the mobile repository to define: the characteristics of the virtual environment in which the behaviour of the distributed mobile application should be evaluated, the tests to be launched and the desired QoS attributes that have to be checked during the test execution. This file must be created in the root of the mobile application repository.
   For the running example, this file is already provided. In particular, three sets of Android mobile devices have been defined, each one consisting of 8 devices. The difference lies on their hardware, simulating mobile devices with different CPU and RAM. In addition, 3 performance tests have been defined for evaluating the general response time of the distributed mobile application with different number of concurrent users. Finally, the QoS objective of a response time lower that two seconds is also defined, in order to provide an adequate user experience. An interface test with Espresso [5] has also been defined to evaluate the main functionality of the application in which the user clicks on the button to calculate the infection risk percentage.
4) **Automating the execution Workflow:** The execution of Perses is automated with GitHub Actions. For this, the different steps to perform (compilation of the application, installation of the necessary resources, deployment of the virtual mobile devices, execution of the tests, etc.) should be defined in a pipeline. This pipeline is already defined and can be reused to automate the execution of any project.[5]. To do this, it should be copied into the root of the mobile application repos-

[1]https://github.com/perses-org/perses/blob/master/MetalVerification.md
[2]Mobile application: https://doi.org/10.5281/zenodo.4476671
[3]Server: https://doi.org/10.5281/zenodo.4476371
[4]https://stackoverflow.com/questions/17668265/gradlew-permission-denied
[5]https://raw.githubusercontent.com/perses-org/gha/master/workflow/perses-workflow.yml

itory following the structure: *.github/workflows/perses-workflow.yml*. Please, note that this file is already also included in the provided example.

5) **AWS Credentials:** In order for Perses to be able to deploy the virtual devices in AWS, it needs credentials to access and create the necessary infrastructure. To that end, developers have to create an user and obtain the EC2 Key Pair. To create the user, the AWS tutorial should be followed.[6]. Please, note that this user must have *Programmatic* access and *AdministratorAccess* permission. After creating the user, download the provided *.csv* file for later use. To create the EC2 Pair Key, again, the AWS tutorial should be followed.[7].Please note that this key has to be created in the eu-west-1 region (this is to make it easier the artifact guide) otherwise the pipeline execution will not be successful. Finally, the generated key should be downloaded as a *.pem* file.
6) **GitHub Secrets:** to automatically deploy the virtual environment and execute the defined test when a new commit is done, the encrypted environment variables on GitHub, called Secrets, are used to include the AWS credentials in the application repository. To create them, in the application repository, developers have to access to *Settings* and *Secrets*, creating the following variables:
   - **AWS_ACCESS_KEY**. In this variable, developers should add the **Access key ID** provided in the the previously downloaded *.csv* file.
   - **AWS_SECRET_KEY** is used to insert the **Secret access key** provided in the *.csv* file.
   - **KEY_NAME**, which contains the *name* of the EC2 pair key (not the ID).
   - **KEY_PEM**, which stores a copy of the **all** content of the *.pem* file.

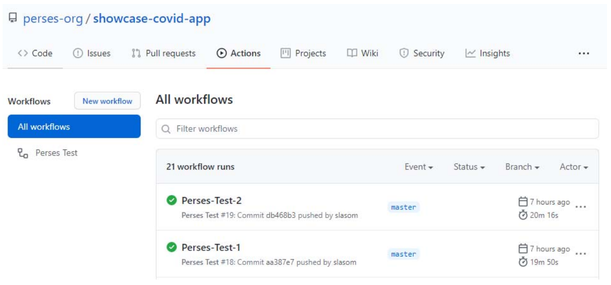
Once these steps are completed, Perses should be ready to deploy the virtual environment to evaluate the QoS.

III. Evaluating the QoS

Perses is integrated in a DevOps and a Continuous Integration process. To launch Perses, it is necessary to make some changes in the repository (e.g. modify the Readme.md file by including a sentence). After saving the change, Github will automatically launch Perses.This launch will execute the steps defined in the above mentioned pipeline. The process of evaluating the QoS can be reviewed accessing to the Action tab in the GitHub repository. Figure 1 shows an example of different executions.

In order to thoroughly analyze the results obtained from each Perses execution, developers can access to each of the listed Perses runs by clicking on *perses-test* in the *Jobs* section. The Github Action console will be opened showing the results of each of the steps defined in the pipeline such as building the project, creation of the infrastructure, deployment of the virtual devices, test execution, etc. Figure 2 shows an excerpt of the content of the step *Check Test Results*, where the results of the performance tests are evaluated to check if they are under the defined response time requirement. This console also shows if the Espresso tests were successfully passed on each device.

[6]https://amzn.to/3nVDoQb
[7]https://amzn.to/3sCr3Ut

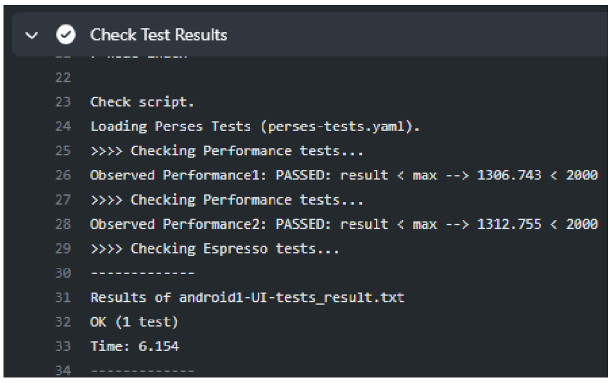


21 workflow runs

Perses-Test-2
Perses Test #19: Commit db468b3 pushed by slasom
master
7 hours ago
20m 16s

Perses-Test-1
Perses Test #18: Commit aa387e7 pushed by slasom
master
7 hours ago
19m 50s

Fig. 1. GitHub Actions Main Page.



```

Check script.
Loading Perses Tests (perses-tests.yaml).
>>>> Checking Performance tests...
Observed Performance1: PASSED: result < max --> 1306.743 < 2000
>>>> Checking Performance tests...
Observed Performance2: PASSED: result < max --> 1312.755 < 2000
>>>> Checking Espresso tests...
------------
Results of android1-UI-tests_result.txt
OK (1 test)
Time: 6.154
------------
```

Fig. 2. Perses workflow.

Acknowledgment

This work has been partially funded by the projects RTI2018-094591-B-I00, RTI2018-101204-B-C21 and TIN2015-70560-R (MCI/AEI/FEDER,UE), the 4IE+ Project (0499-4IE-PLUS-4-E) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the RCIS network (TIN2016-81978-REDT), by the Department of Economy and Infrastructure of the Government of Extremadura (GR18112, IB18030), by the Government of Andalusian (US-1264651) and by the European Regional Development Fund.

References

[1] Github, "Github," https://github.com, (Accessed on 19/01/2021).
[2] A. W. Service, "Amazon web service," https://aws.amazon.com/, (Accessed on 19/01/2021).
[3] Github, "Github actions," https://github.com/features/actions, (Accessed on 28/10/2020).
[4] S. Laso, J. Berrocal, P. Fernández, A. Ruiz-Cortés, and J. M. Murillo, "Virtual environment for evaluating the qos of distributed mobile applications," in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2021, pp. 1–3.
[5] Android, "Espresso - Android Developer," https://developer.android.com/training/testing/espresso, (Accessed on 19/01/2021).