Automated Testing with Jenkins for Continuous Integration in Software Development

Hemanth Kumar B V[1], Sudha R Karbari[2]
Department of Electronics and Communication Engineering, RV College of Engineering, Bengaluru -59
Email: sudhark@rvce.edu.in

Abstract

Continuous Integration (CI) is a practice in software program development process where software program builders combine code into a shared repository frequently, more than one instances through the day. Jenkins is a continuous integration tool which assist developer and testers by using automating the entire test, on the way to reduce their work with the aid of tracking the development at each and every stage in software development, each integration push is then tested by means of automated test cases, and an easy way to make CI quicker and accelerate CI procedure is to automate the testing of recent build. In this paper a real scenario is taken into consideration, how the software program trying out is performed in corporate sectors and how Jenkins can save developers/testers important valuable hours by automating the whole software development system.

Keywords: *Jenkins, Python, Automated Testing, Cloud Integration, Software Development.*

I Introduction

In software development the developers working in a team, have a shared repository to which continuous integration new features or build that is developed will be integrated frequently. , each and every push can then be checked with aid of an automated testing. Quality assurance team can go through the latest build merged and detect faults. The faults can be the build not satisfying the requirements, the build cannot be adaptable, build can affect the earlier functionalities. Recent survey shows that companies are using continuous integration in their software development routine because of its highlighting features and testing through automation. Continuous deployment and continuous delivery is found to be the best practices for keeping application ready at any point of time and also making new changes available to production at the earliest possible. This helps team to be quick and make company to be engage with the customers by providing them the high-quality standards that can be automatically tested. Jenkins is an open-source tool written in java programming language with built in plugins for continuous integration. Jenkins is widely used in software development life cycle, in building and testing projects, helping new changes are integrated into main code base thus providing customers with the fresh build.

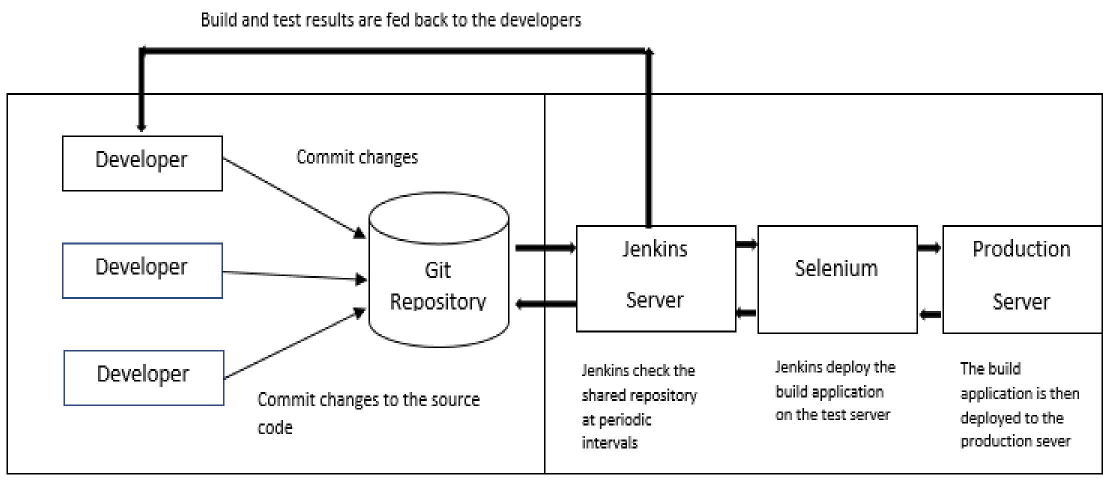

Build and test results are fed back to the developers
Developer
Developer
Developer
Commit changes
Git Repository
Commit changes to the source code
Jenkins Server
Selenium
Production Server
Jenkins check the shared repository at periodic intervals
Jenkins deploy the build application on the test server
The build application is then deployed to the production sever

Fig 1. Jenkins Single server Architecture

1.1 Jenkins Single Server Architecture

Jenkins single server architecture is shown in

Fig 1. **Jenkins Single server Architecture**This architecture is best suited for smaller applications dedicated to single platforms but fails when the project build is huge and to be run on multiple platforms, since server cannot handle the entire load. These drawbacks are overcome by the new master-slave architecture.

1.2 Jenkins Master-Slave architecture

Jenkins Master-Slave architecture is used to manage distributed build. Communication between master and slave is through standard TCP/IP protocol. The above **Error! Reference source not found.**shows the Jenkins Master-Slave architecture. Jenkins main server is the master, and it performs the below mentioned tasks:

- Executing builds over slaves.
- Maintaining slaves (software packages).
- Providing a stable build for production.
- Scheduling jobs.

Jenkins slaves are the remote machine that are tagged in the respective master. Main functionalities of Jenkins slaves are:

- Get the requests sent from master.
- Slaves can be run on various operating systems.
- Slaves provide a flexibility to run

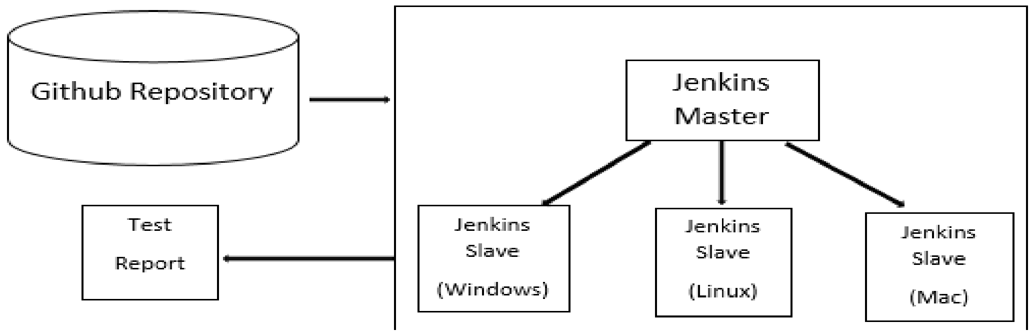


Fig 2. Jenkins Master-Slave Architecture

specific tasks on them periodically.

- These slaves are usually Virtual machine deployed on Vmware.

1.3 Proposed Approach

This paper proposes the need for automation and its implementation in the Jenkins tool and master/slave architecture is proposed where a Jenkins server is able to execute certain jobs parallel on multiple clients parallel. Implementation of Jenkins for automating test execution to handle day by day integration venture, is proposed in this paper. The remaining section are section 2 abut the software testing techniques and advantage of automated testing. Section 3 is discussion of the proposed work. Section 5 shows the final testing reports in Jenkins. Finally section 5 gives the conclusion of this paper.

II Software Testing

Software testing is an activity to verify the functionalities of the software are working as per the requirements and to guarantee the software is free of imperfections so that it can be delivered to next stages say production. Software testing is very important in order to figure out the defects and errors that were made during the development phases. Software testing ensures that quality item is provided to the customer, Fig *2*. Software testing flow 3 shows the flow of software testing. There are 2 major software testing types:

1. **Manual Testing:**

Manual testing is a testing technique to test the product in presence of a dedicated skilled testing team. Testing team should have the viewpoint od the end user, and they should assure that all the functionalities of the software product are working same as that mentioned in the requirements document.

2. **Automated Testing:**

Automation testing is the way of testing software with the help of test cases. Test cases are the code that is been written in specific programming language, where the expected and actual results are compared and this test cases can be preserved to run in the next stages. Any new feature when added this test cases are made to run so that it will assure that new feature is not affecting the existing one. This kind of testing requires initial investment in the early stages.

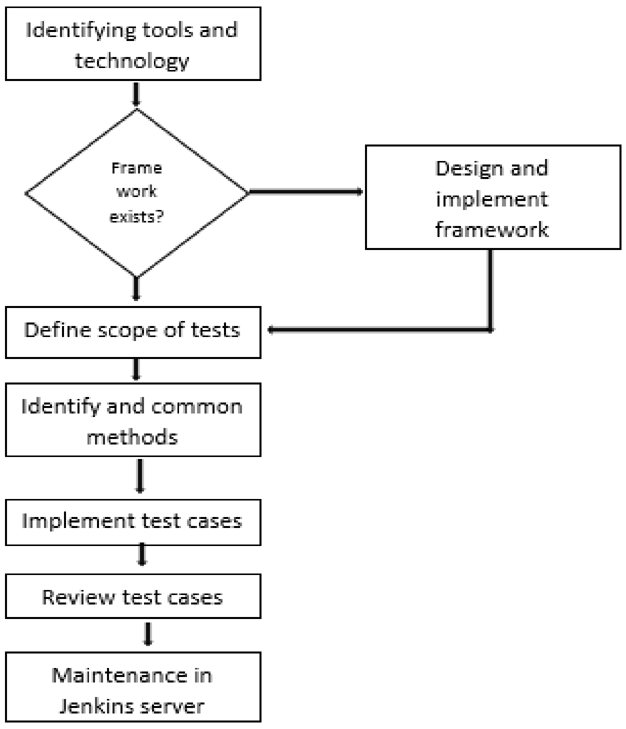

Identifying tools and technology
Frame work exists?
Design and implement framework
Define scope of tests
Identify and common methods
Implement test cases
Review test cases
Maintenance in Jenkins server

Fig 2. Software testing flow

Maintenance of a software is an activity which includes frequent development, error corrections, optimization and deletion of existing feature, these changes may affect the system corrupted, working not properly so regression testing becomes necessary. Regression testing can be defined as the software testing technique that guarantees that recent updates made to the code is not affecting the earlier features. There can be full or partial selection of existing and executed test cases which are runt repeatedly to assure the software stability.

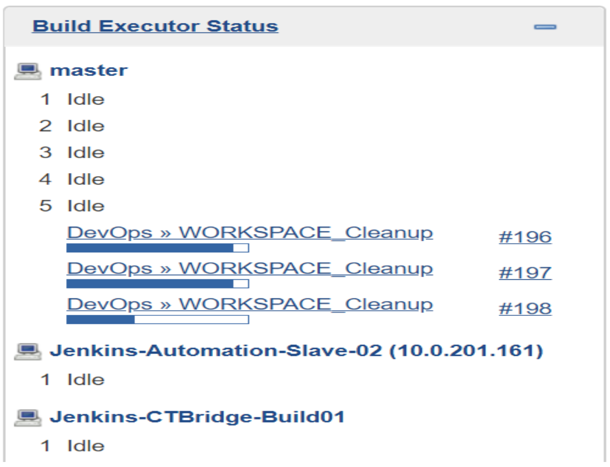
In the considered scenario the test cases are written in the python language, 6 unit test cases are written for execution purpose. The main purpose of software testing is to find out the faults and defects so that they can be covered before reaching it to customers. Software reliability matters in case of continuous integration, it is the probability of the being fault-free operating as desired for specific period of time under a specified environment.

2.1 Difficulties in software testing

1. Testing team is responsible for communicating with customers in order to understand the requirements. Lack of communication, and improper documentation results in inefficient testing.
2. Deadline for testing makes the testers to complete testing without covering most of the functionalities, so testers should be enough skilled to increase test coverage and quality of work in specified time.
3. All the features may not be possible to test, so a detailed report to be made before testing including the test scenarios and functionalities that to tested and this report should be share with the development team to ensure the test coverage.

III Testing using Jenkins

The Jenkins master is installed on a windows machine and slaves are the virtual machines deployed with the help of vm ware. The testing software is a multiplatform software, therefore different platform (OS) resources are deployed say windows and ubuntu. The slaves are added under the master using manage nodes option. The below **Error! Reference source not found.**, shows the master-slaves status, here the master is in idle state present, Jenkins-Automation Slave02 , Jenkins-CTBridge-Build01 are the slave nodes deployed over vm ware and Workspace_cleanup are the last builds made by master.



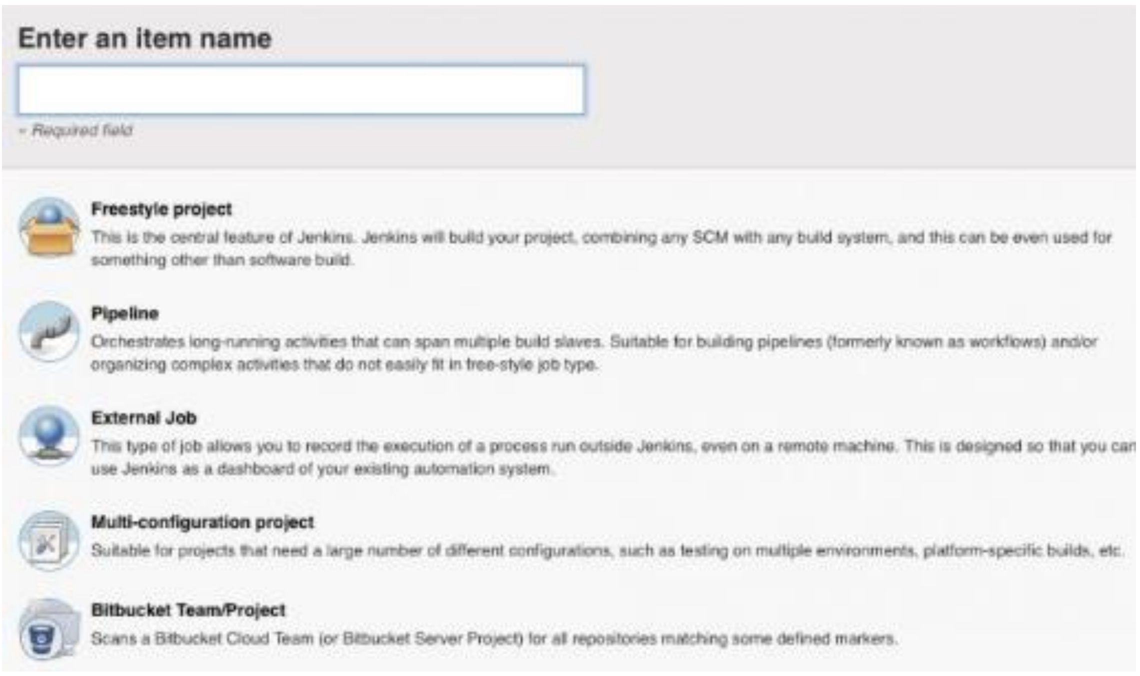
Build Executor Status

master
1 Idle
2 Idle
3 Idle
4 Idle
5 Idle
DevOps » WORKSPACE_Cleanup #196
DevOps » WORKSPACE_Cleanup #197
DevOps » WORKSPACE_Cleanup #198

Jenkins-Automation-Slave-02 (10.0.201.161)
1 Idle

Jenkins-CTBridge-Build01
1 Idle

Fig 3. Jenkins master-slave status



Enter an item name

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Bitbucket Team/Project
Scans a Bitbucket Cloud Team (or Bitbucket Server Project) for all repositories matching some defined markers.

Fig 4. New Project creation

In order to create new project, Jenkins provide various plugin options for project as shown in Fig 4. New Project creation, among which free style technique is used in this project execution, the advantage of free-style over others is it enables customization like adding mail recipients for receiving test report, build report etc.

After creating a project the test cases to be run on the slave nodes are fetched from the remote repository, the remote repository here holds the test suites, which include test cases validating the functionalities of the product. The packages to be installed on the slave nodes prior to excution of the test suite are entered in requirements list ,so that Jenkins take care of installing all these packages, a image file is created for this, **Error! Reference source not found.** shows the flow of testing process in Jenkins. After creating the build the testing team can verify the run, in the build status widget shown in Fig 5. Build status, the color of the icon represent the build stability, blue denotes the stable build while red denotes the unstable, ex: #494 is one of

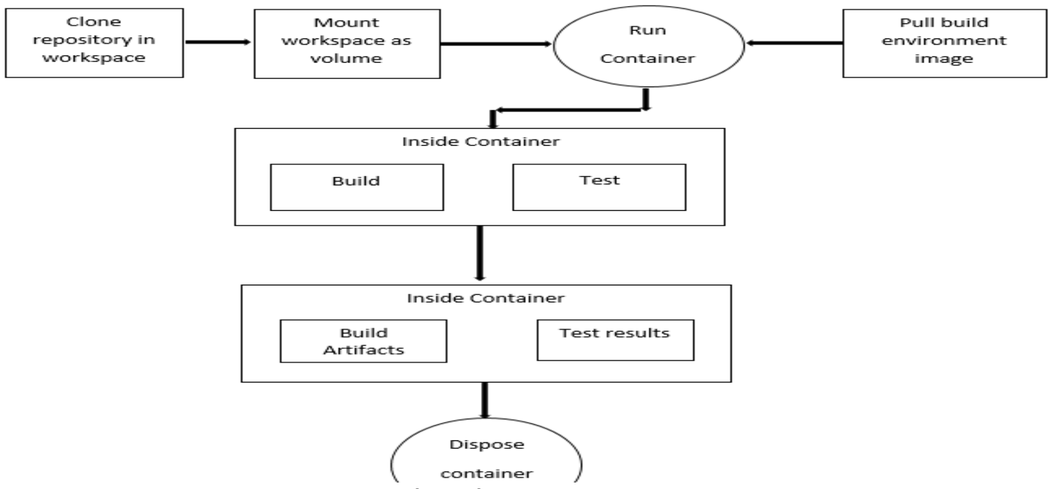
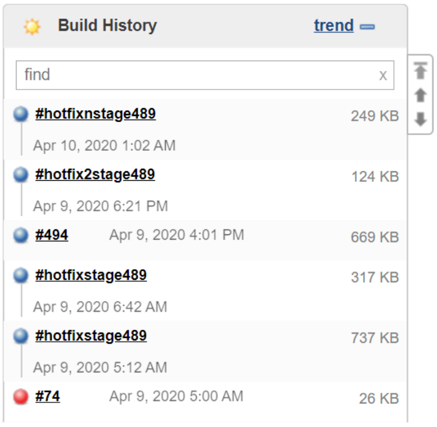


Fig 6 shows the flow of testing process in Jenkins the stable build and #74 build is the unstable build.


Build History trend

find

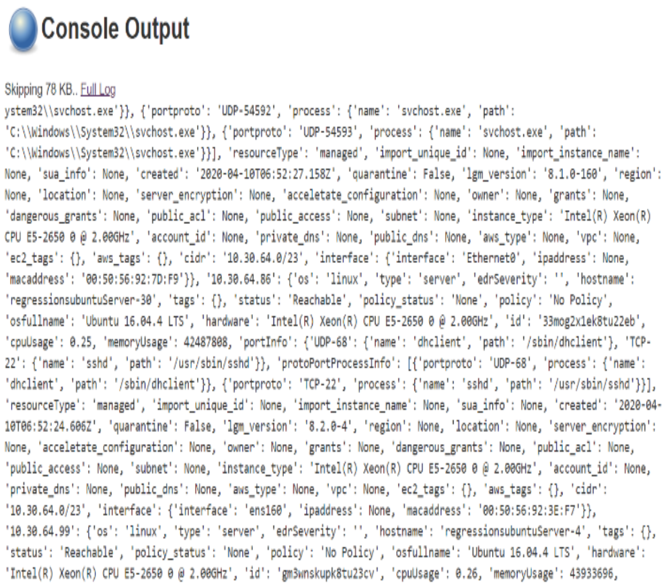
#hotfixnstage489 Apr 10, 2020 1:02 AM	249 KB
#hotfix2stage489 Apr 9, 2020 6:21 PM	124 KB
#494 Apr 9, 2020 4:01 PM	669 KB
#hotfixstage489 Apr 9, 2020 6:42 AM	317 KB
#hotfixstage489 Apr 9, 2020 5:12 AM	737 KB
#74 Apr 9, 2020 5:00 AM	26 KB

Fig 5. Build status

For the testing purupose 6 test cases are written in the test suite. These test cases mainly include python script using the pytest and selenium packages, to verify the GUI and functionality of the software. Only the stable builds are added for regression, where any new features added go through this testing everytime, this is to certify that the existing features are not affected due to the recent updations. Testing team will trigger this build i.e., rebuild this periodically to verify the reliability of the software.

IV Results

Fig 6. Test run logs shows the logs displayed during execution of automation test scripts, these logs are used during debugging, Bug fixing. The time consumption for test run shown in the above Fig 7. Test run duration can be reduced by optimizing the script length and less usage of loops, with specialized coding techniques.


Console Output

```
Skipping 78 KB.. Full Log
ystem32\\svchost.exe'}}, {'portproto': 'UDP-54592', 'process': {'name': 'svchost.exe', 'path':
'C:\\Windows\\System32\\svchost.exe'}}, {'portproto': 'UDP-54593', 'process': {'name': 'svchost.exe', 'path':
'C:\\Windows\\System32\\svchost.exe'}}], 'resourceType': 'managed', 'import_unique_id': None, 'import_instance_name':
None, 'sua_info': None, 'created': '2020-04-10T06:52:27.158Z', 'quarantine': False, 'lgm_version': '8.1.0-160', 'region':
None, 'location': None, 'server_encryption': None, 'acceletate_configuration': None, 'owner': None, 'grants': None,
'dangerous_grants': None, 'public_acl': None, 'public_access': None, 'subnet': None, 'instance_type': 'Intel(R) Xeon(R)
CPU E5-2650 0 @ 2.00GHz', 'account_id': None, 'private_dns': None, 'public_dns': None, 'aws_type': None, 'vpc': None,
'ec2_tags': {}, 'aws_tags': {}, 'cidr': '10.30.64.0/23', 'interface': {'interface': 'Ethernet0', 'ipaddress': None,
'macaddress': '00:50:56:92:7D:F9'}}, '10.30.64.86': {'os': 'linux', 'type': 'server', 'edrSeverity': '', 'hostname':
'regressionsubuntuServer-30', 'tags': {}, 'status': 'Reachable', 'policy_status': 'None', 'policy': 'No Policy',
'osfullname': 'Ubuntu 16.04.4 LTS', 'hardware': 'Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz', 'id': '33mog2x1ek8tu22eb',
'cpuUsage': 0.25, 'memoryUsage': 42487808, 'portInfo': {'UDP-68': {'name': 'dhclient', 'path': '/sbin/dhclient'}, 'TCP-
22': {'name': 'sshd', 'path': '/usr/sbin/sshd'}}, 'protoPortProcessInfo': [{'portproto': 'UDP-68', 'process': {'name':
'dhclient', 'path': '/sbin/dhclient'}}, {'portproto': 'TCP-22', 'process': {'name': 'sshd', 'path': '/usr/sbin/sshd'}}],
'resourceType': 'managed', 'import_unique_id': None, 'import_instance_name': None, 'sua_info': None, 'created': '2020-04-
10T06:52:24.606Z', 'quarantine': False, 'lgm_version': '8.2.0-4', 'region': None, 'location': None, 'server_encryption':
None, 'acceletate_configuration': None, 'owner': None, 'grants': None, 'dangerous_grants': None, 'public_acl': None,
'public_access': None, 'subnet': None, 'instance_type': 'Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz', 'account_id': None,
'private_dns': None, 'public_dns': None, 'aws_type': None, 'vpc': None, 'ec2_tags': {}, 'aws_tags': {}, 'cidr':
'10.30.64.0/23', 'interface': {'interface': 'ens160', 'ipaddress': None, 'macaddress': '00:50:56:92:3E:F7'}},
'10.30.64.99': {'os': 'linux', 'type': 'server', 'edrSeverity': '', 'hostname': 'regressionsubuntuServer-4', 'tags': {},
'status': 'Reachable', 'policy_status': 'None', 'policy': 'No Policy', 'osfullname': 'Ubuntu 16.04.4 LTS', 'hardware':
'Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz', 'id': 'gm3wnskupk8tu23cv', 'cpuUsage': 0.26, 'memoryUsage': 43933696,
```

Fig 6. Test run logs

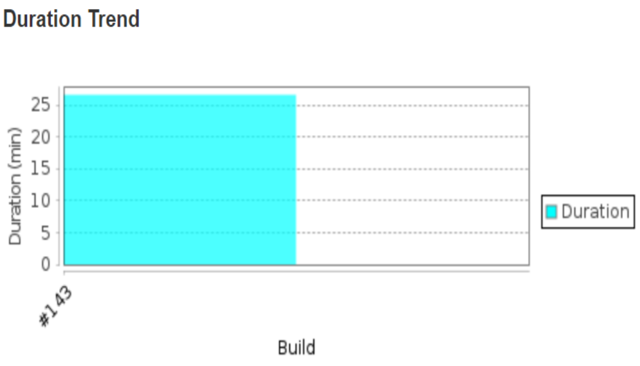
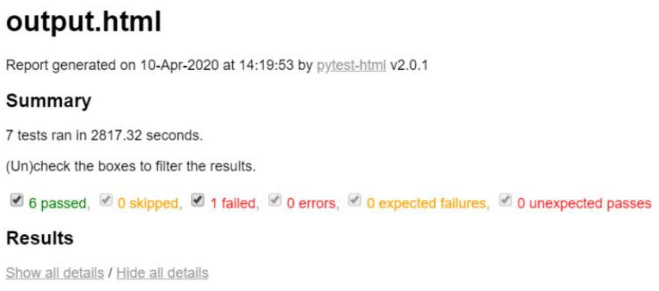


Fig 7. Test run duration


output.html

Report generated on 10-Apr-2020 at 14:19:53 by pytest-html v2.0.1

Summary

7 tests ran in 2817.32 seconds.

(Un)check the boxes to filter the results.

✓ 6 passed, ✓ 0 skipped, ✓ 1 failed, ✓ 0 errors, ✓ 0 expected failures, ✓ 0 unexpected passes

Results

Show all details / Hide all details

Fig 8. Test report

The Fig 8. Test reportshows the test case report, where it can be seen 6 test cases are passed and 1 test case is failing, the console output gives the detailed log during execution as show in Fig 8, which helps in debugging the errors.

V Conclusion

Continuous integration is of much importance in software development process. Automation of the integrated tasks is of high priority since these task are made to execute periodically. The stable build is pipelined into regression, this is to assure that recent changes is not affecting the existing features. Jenkins is a better solution for continuous integration in software development. This paper gives the overview of how to automate the test cases in Jenkins, how to trigger regression and how to add test cases in to pipeline. The detailed discussion and review of Jenkins and challenges in automation is done in this paper.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] N. Seth and R. Khare, "ACI (automated Continuous Integration) using Jenkins: Key for successful embedded Software development," 2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS), Chandigarh, 2015, pp. 1-6.

[3] S. Sivanandan and Yogeesha C. B, "Agile development cycle: Approach to design an effective Model Based Testing with Behaviour driven automation framework," 20th Annual International Conference on Advanced Computing and Communications (ADCOM), Bangalore, 2014, pp. 22-25.

[4] L. Crispin, J. Gregory, Agile Testing: A Practical Guide for Testers and Agile Teams, Addison-Wesley, 2011.

[5] A. Askarunisa, K. A. J. Punitha, A. M. Abirami, "Black Box Test Case Prioritization Techniques for Semantic Based Composite Web Services Using OWL-S", Proceedings of the IEEE-International Conference on Recent Trends in Information Technology ICRTIT, 2011.

[6] Y. Liu, Y. Lu, Y. Li, An Android-Based Approach for automatic unit test, 2015.

[7] Online Resource, [online] Available: https://jenkins-ci.org/.

[8] Online Resource, [online] Available: https://wiki.jenkins-ci.org/display/JENKINS/Home.

[9] E. M. A. Rauf, E. M. Reddy, "Software Test Automation: An Algorithm for Solving System Management Automation Problems", Proceedings of the International Conference on Information and Communication Technologies ICICT, 2014.

[10] D. Jenkins, J. Arnaud, S. Thompson, M. Yau, J. Wright, Version Control and Patch Management of Protection and Automation Systems, 2013.

[11] IEEE Draft International Standard for Software and Systems Engineering--Software testing--Part 4: Test Techniques," in ISO/IEC/IEEE P29119-4-DISMay2013 , vol., no., pp.1-132, 21 Feb. 2014.

[12] K. Jambunatha, "Design and implement Automated Procedure to upgrade remote network devices using Python," 2015 IEEE International Advance Computing Conference (IACC), Banglore, 2015, pp. 217-221.

[13] Victor E. L. Valenzuela, Vicente F. Lucena, Nasser Jazdi, Peter Göhner, "Reusable hardware and software model for remote supervision of Industrial Automation Systems using Web technologies" in , IEEE, ISBN 978-1-4799-0864-6/13/\$31.00 ©2013.

[14] Fernando Pianegiani, David Macii, "An open distributed measurement system based on abstract Client-Server Architecture", IEEE transactions on instrumentation and measurement, vol. 52, no. 3, JUNE 2003.

ACRONYMS LIST

SW	Software
OS	Operating System
CI	Continuous Integration
MID	Model-Implementation Description
MISTRA	Model Based Integration and System Test Automation
RST	Regression Test Selection
SD	Software Development