Volume: 8 Issue: XI Month of publication: November 2020
DOI: https://doi.org/10.22214/ijraset.2020.32105

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue XI Nov 2020- Available at www.ijraset.com

Automation Pipeline and Build Infrastructure using DevOps

P. Maragathavalli[1], M. Seshankkumar[2]
[1, 2]Department of Information Technology, Pondicherry Engineering College, Puducherry, India.

Abstract: A complex project involves complex development effort and it involves more manual work and more time. It is very difficult to reduce the time involved. In the development cycle the use of the concurrent engineering concept only increases the problem by increasing the number of people and department involved at any point in the development effort. DevOps techniques can be used in these projects to control the use of project elements. Using DevOps Technique many companies involved in complex projects are able to reduce both development time and manual work involved in projects.
This project solves issues that arise due to manual work on a project that involve more than one member in a team. The usual way of building a project in which code are distributed among team members leads to many issues like integration and compatible problems. The existing version control system gives us integration but that need to be tuned for more effective way of using that tool. This project builds an automation system that handles all the integration and deployment phrases automatically. This also reduces the time of compilation of huge codes. Manager need not wait for the results from team members since the results are stored automatically in Git.
Keywords: DevOps, Git Hub, CICD pipeline, YML.

I. INTRODUCTION

The main purpose of this project is to automate the development process of a project that includes the phrases of building the code, testing and deployment of the project which need more time when doing these tasks manually. Eight builds are involved which would take a huge time if done manually. The existing version control system does the three phrases separately which takes more time and work. It is a lengthy process that needs more space and time. The continuous integration and deployment were required that will be presented by this project. This project uses DevOps which helps in Continuous Integration and Continuous Deployment of the distributed code that are required at the single phrase for building, testing and deploying the project. The main of this project is designing of an application automation system using the technologies DevOps and CICD pipeline through GitLab. The CICD means continuous integration and continuous deployment.

A. Objective

Main objective of the proposed work is,

1) To provides an application automation (issues tracking, fixing, regression) system.
2) To provide a system does the process of building, testing and deploying in the continuous pipeline manner.
3) To build an application that handles complete control over a centralized code with many developers.
4) To reduce the time of builds since many changes will come in a day in organization all these changes need separate builds.
5) To spot the error in a particular build and find the developer responsible for the error.
6) To sport the warnings count in each build.

II. LITERATURE SURVEY

This section discusses the methodologies learnt from previous works for our system.

A. Concurrent versions system (CVS)

CONCURRENT VERSIONS SYSTEM also known as Computer Vision Syndrome is a version control system, a vital part of Source Configuration Management (SCM). Utilizing it, developer can record the historical backdrop of sources documents, and reports. It fills a comparable job to the free programming RCS, PRCS, and Aegis bundles. CVS is a generation quality framework in wide use the world over, including many free programming tasks. It additionally lets to share various adaptations of records in a typical vault between group of engineers. CVS monitors numerous duplicates of source code documents and furthermore keep up a solitary duplicate and all the progressions are recorded.

Every designer's work independently in a different working index and CVS monitors all engineer's work. Crafted by a group of designers can be converged in a typical vault when wanted. Submit order is utilized to blend the changes.

CVS utilizes Client–Server engineering: a worker stores the current version(s) of an undertaking and its set of experiences, and customers interface with the worker so as to "look at" a total duplicate of the task, chip away at this duplicate and afterward "check in" their changes. CVS is arranged towards text records.CVS is oriented towards text files. In cross-platform development, It is very easy to mess up the management of binary files (e.g., graphics such as icons). It is a bit difficult to learn about the myriad of configuration files that control CVS's behaviour. The official manual (by Per Cederqvist) is a bit out of date. It can run scripts which you can supply to log CVS operations or enforce site-specific polices.

B. *Highlights of CVS*

1) Client/worker CVS empowers engineers dispersed by geology or moderate modems to work as a solitary team.The adaptation history is put away on a solitary focal worker and the customer machines have a duplicate of the apparent multitude of documents that the designers are working on. Therefore, the organization between the customer and the worker must be up to perform CVS tasks, (for example, registration or updates) however need not be up to alter or control the current renditions of the files. Clients can play out in no way different activities which are accessible locally.
2) In situations where a few engineers or groups need to each keep up their own form of the records, on account of topography and additionally strategy, CVS's seller branches can import a variant from another group (regardless of whether they don't utilize CVS), and afterward CVS can mergethe changes from the merchant branch with the most recent records if that is what is wanted.
3) Open checkouts, permitting more than one engineer to chip away at similar records at a similar time.CVS gives an adaptable modules information base that gives an emblematic planning of names to segments of a bigger programming dispersion. It applies names to accumulations of indexes and documents. A solitary order can control the whole collection.
4) CVS workers run on most Unix variations, and customers for Windows NT/95, OS/2 and VMS are additionally accessible. CVS will likewise work in what is here and there called worker mode against nearby archives on Windows 95/NT

III. PROPOSED SYSTEM

In request to beat downsides, for example, time, manual work and space gives needs to change to the advanced most recent philosophy like DevOps and Agile in computerization measure so as to get a proficient framework. 9DevOps is a product improvement approach which includes ceaseless turn of events, constant testing, persistent mix, consistent organization, and nonstop checking of the product all through its advancement lifecycle.
A persistent incorporation and sending pipeline (CD/CI) are such a significant part of a product venture. It spares a huge load of manual, blunder inclined arrangement work. It results in higher quality software for continuous integration, automated tests, and code metrics.
Auto DevOps plans to improve the arrangement and execution of a mature&modern programming advancement lifecycle. Highlight rich: Git archive the board, code audits, issue following, action takes care of and wikis. Relapse testing is the way toward testing changes to PC projects to ensure that the more seasoned programming actually works with the new changes. Regression testing is re-running practical and non-utilitarian tests to guarantee that recently created and tried programming actually performs after a change. To lessen the exertion needed to finish the test robotization measure.

A. *Steps involved in proposed system:*

The system that is proposed has following steps for automation :
1) Configure the yml in GitLab according to developers need.
2) Make the GitLab for storing code and make it a centralized repository.
3) Construct the CICD pipeline for building, testing and deployment of the code across the developers.
4) Push the modified code to the pipeline.
5) Build the test suite for regression testing.
6) Let’s initiate the system to do different phrases.
7) As per the report do the improvement process

IV. MODULES DESCRIPTION

The various modules involved in the work are

1) *CICD Pipeline Configuration*
a) Build Phase
b) Test Phase
c) Deploy Phase
2) *Regression Testing Automation CICD Pipeline Configuration:* The pipeline needs to be configured as per the requirement of the developer using the yml configuration file. Then the complete flow of the phases needs to be designed and configured. Complete automation process is depending upon the CICD pipeline construction.

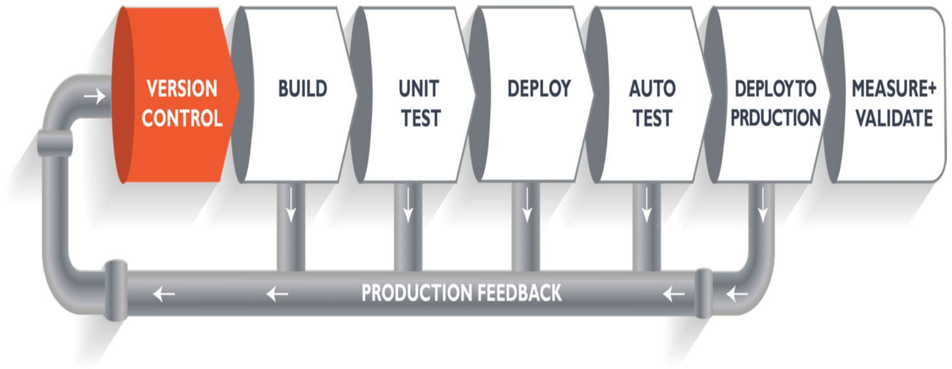


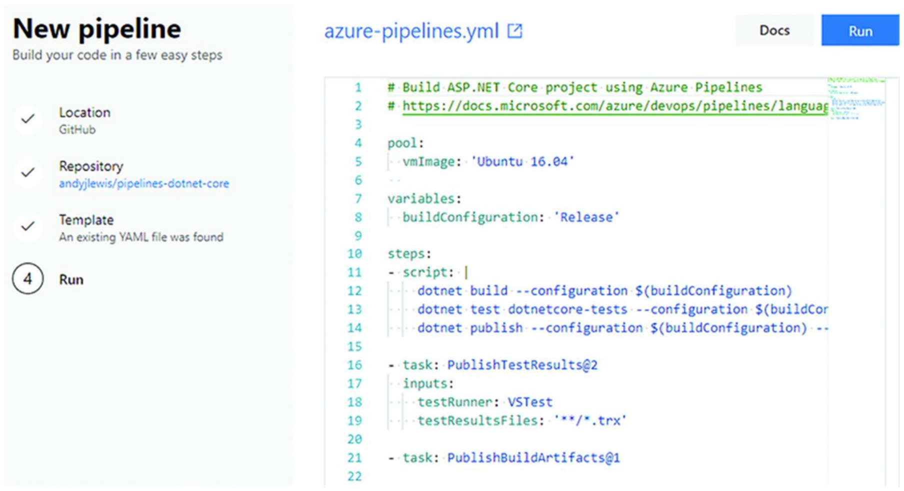
Figure 4.1 CICD pipeline

a) *Version Control:* Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. Client will utilize programming source code as the records being variant controlled, however truly it can do this with almost any kind of document on a PC.
b) *Build Phase:* The IDE compiles the source files and generates the packaged build output, such as a JAR file or WAR file. User can build a project and all of its required projects, or build any project individually. User does not need to build the project or compile individual classes to run the project in the IDE.
c) *Test Phase:* Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test.
d) *Deploy Phase:* Software deployment is all of the activities that make a software system available for use. The general deployment process consists of several interrelated activities with possible transitions between them. These activities can occur at the producer side or at the consumer side or both.

A. *Regression Test Automation*

The Regression tool uses the test Suite for each feature in the code. Each Suite consists of many test cases which will check the different functionalities. Each test case consists of input, output, expected output and test case order file. Testing starts by test Suite Init file which make use of test case order file for order of execution. Regression testing is re-running useful and non-utilitarian tests to guarantee that recently created and tried programming still performs after a change. If not, that would be known as a relapse. Changes that may require relapse testing incorporate bug fixes, programming improvements, setup changes, and even substitution of electronic segments. As relapse test suites will in general develop with each discovered deformity, test robotization is much of the time included. Once in a while a change sway examination is performed to decide a proper subset of tests. As programming is refreshed or changed, or reused on an adjusted target, rise of new blames as well as re-development of old flaws is very normal. Some of the time re-rise happens on the grounds that a fix gets lost through poor amendment control practices (or basic human mistake in modification control). Regularly, a fix for an issue will be "delicate" in that it fixes the issue in the tight situation where it was first watched however not in increasingly broad cases which may emerge over the lifetime of the product. Much of the time, a fix for an issue in one zone accidentally causes a product bug in another region. At long last, it might happen that, when some element is overhauled, a portion of similar slip-ups that were made in the first execution of the component are made in the update.

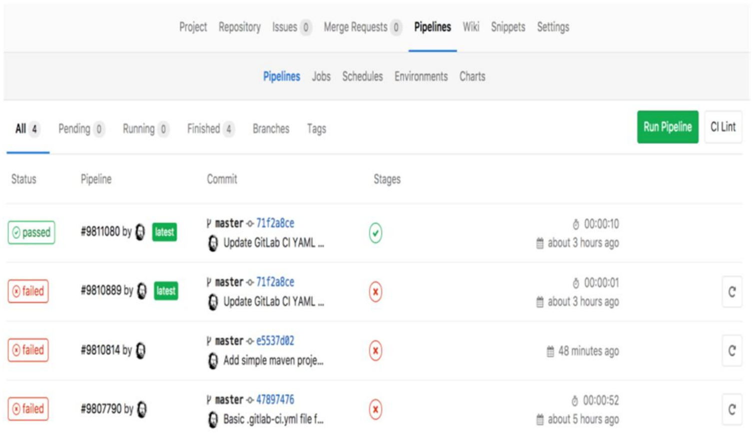
V. RESULTS


New pipeline
Build your code in a few easy steps
Location
GitHub
Repository
andyjlewis/pipelines-dotnet-core
Template
An existing YAML file was found
4
Run
azure-pipelines.yml
Docs
Run

```yaml
# Build ASP.NET Core project using Azure Pipelines
# https://docs.microsoft.com/azure/devops/pipelines/languag

pool:
  vmImage: 'Ubuntu 16.04'

variables:
  buildConfiguration: 'Release'

steps:
- script: |
    dotnet build --configuration $(buildConfiguration)
    dotnet test dotnetcore-tests --configuration $(buildCor
    dotnet publish --configuration $(buildConfiguration) --

- task: PublishTestResults@2
  inputs:
    testRunner: VSTest
    testResultsFiles: '**/*.trx'

- task: PublishBuildArtifacts@1

```

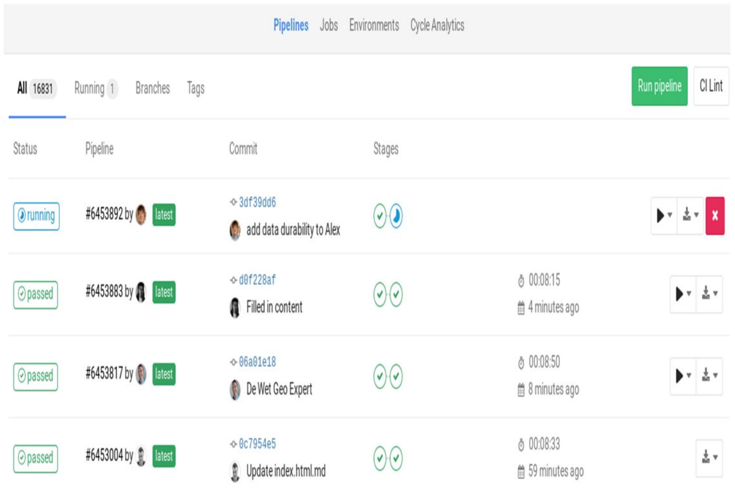
Figure 5.1 YML configuration file

The Figure 5.1 shows the YML configuration file which has the order of build and test to be executed and the script to be executed for a particular build or test.



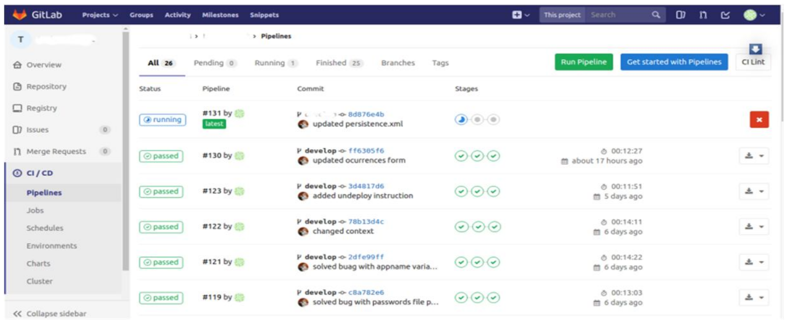
Status	Pipeline	Commit	Stages	
passed	#9811080 by latest	master 71f2a8ce Update GitLab CI YAML ...	✓	00:00:10 about 3 hours ago
failed	#9810889 by latest	master 71f2a8ce Update GitLab CI YAML ...	✗	00:00:01 about 3 hours ago
failed	#9810814 by	master e5537d02 Add simple maven proje...	✗	48 minutes ago
failed	#9807790 by	master 47897476 Basic .gitlab-ci.yml file f...	✗	00:00:52 about 5 hours ago

Figure 5.2 Single phrase build



Status	Pipeline	Commit	Stages	
running	#6453892 by latest	3df39dd6 add data durability to Alex	✓ ●	
passed	#6453883 by latest	d0f228af Filled in content	✓ ✓	00:08:15 4 minutes ago
passed	#6453817 by latest	06a01e18 De Wet Geo Expert	✓ ✓	00:08:50 8 minutes ago
passed	#6453004 by latest	0c7954e5 Update index.html.md	✓ ✓	00:08:33 59 minutes ago

Figure 5.3 Double phrase build and testing



Status	Pipeline	Commit	Stages		
running	#131 by latest	8d876e4b updated persistence.xml			✕
passed	#130 by	develop ff6305f6 updated ocurrences form	✓ ✓ ✓	00:12:27 about 17 hours ago	
passed	#123 by	develop 3d4817d6 added undeploy instruction	✓ ✓ ✓	00:11:51 5 days ago	
passed	#122 by	develop 78b13d4c changed context	✓ ✓ ✓	00:14:11 6 days ago	
passed	#121 by	develop 2dfe99ff solved buag with appname varia...	✓ ✓ ✓	00:14:22 6 days ago	
passed	#119 by	develop c8a782e6 solved bug with passwords file p...	✓ ✓ ✓	00:13:03 6 days ago	

Figure 6.4 shows the Three phrases build, test and deploy

VI. CONCLUSION

In the existing system, the developer need to manually do the process of building, testing and deployment which consumes more time and even if user use scripts to do these process user can't stop while execution but if user use these automation system, the developer do these process in a pipeline manner and get the report periodically even if any phase got error the system don't move to next phase. It helps the developer to track the issues and fix it easier. In future this work can be further improved with the help of different phrases to be added to the existing system that can further improve the time and space issues. This work can be further extended for list of projects also. Along with this project some special phrases like system testing also can be added.

REFERENCES

[1] Vidroha Debroy, Senecca Miller, "Overcoming Challenges with Continuous Integration and Deployment Pipelines When Moving from Monolithic Apps to Microservices", IEEE Software, IEEE, ISSN: 2169-3536, Vol. 37, Issue: 3, Feb 2020, pp.21-29.
[2] Keheliya Gallaba, "Improving the Robustness and Efficiency of Continuous Integration and Deployment", 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, ISSN: 2576-3148, Vol. 10, Dec 2019, pp.619 - 623.
[3] Ana Filipa Nogueira, Jos´e C. B. Ribeiro, M´ario A. Zenha-Rela, Antoine Craske, "Improving La Redoute's CI/CD pipeline and DevOps processes by applying Machine Learning techniques", 2018 International Conference on the Quality of Information and Communications Technology, IEEE, ISBN: 978-1-5386-5841-3, Vol. 96, Dec 2018, pp.282 - 286.
[4] Mohammed Shamsul Arefeen, Michael Schiller, "Continuous Integration Using Gitlab", URNCST Journal, Vol. 3, Issue: 8, Nov 2019, pp.40-45.
[5] Alexander Poth, Mark Werner, and Xinyan Lei, "How to Deliver Faster with CI/CD IntegratedTesting Services?", European Conference on Software Process Improvement, Springer, ISBN: 978-3-319-97924-0, Vol. 896, Aug 2018, pp.401 – 409.

Author Biography

Dr. P. Maragathavalli

She received her B.E degree in CSE from Bharathidasan University, M.Tech. degree in CSE from Pondicherry University and PhD degree in CSE from Pondicherry University. She is working as Assistant Professor in the Department of Information Technology; Pondicherry Engineering College. She is a Life member of ISTE.

M. Seshank kumar

He is pursuing his B.Tech degree in the Department of Information Technology, Pondicherry Engineering College from Pondicherry University.