International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 05 | May 2021 www.irjet.net p-ISSN: 2395-0072

GERRIT MIGRATION

Suresh[1], Dr. Anitha G S[2], Shruti L[3]

[1]PG Student, Department of Electrical and Electronics Engineering, RV College of Engineering, Bengaluru
[2]Associate Professor, Department of Electrical and Electronics Engineering, RV College of Engineering, Bengaluru
[3]R and D Engineer, Nokia Networks and Solutions, Bengaluru

Abstract - *Version Control Systems are the most important things in software development life cycle . Which keep monitors and tracks about addition and deletion activities that are made to system. Developers creating a new source code and modifying the existing source code for a software component. In this paper, we have described about distributed version control System(DVCS) tool like Git. Git is a free and open source system developed to manage everything from small to very large projects with speed and efficiency. Git can copy or clone the repository to local repository, Developer can commit and update their local repository without any interference, to update their local repositories with new data from the central server by an command called "pull" and update the main repository with the local repository commits by an operation called "push" Git tracks the changes made by the developer to the files , so developer have a record of what has been done. If at any point of time central server gets crashed, the lost data can be easily recovered from any of the local repositories. SVN(Subversion) is not locally available, to perform any action always need to be connected to a network. As SVN is a centralized version control system(CVCS) in case if the server getting crashed or corrupted will result in losing the entire data of the project, switching to the feature branch in developing a script is not happen in SVN. To overcome this problem, Migration from SVN to Git is carried out in an organization with the help of shell scripts and groovy scripts.*

Key Words: **Version control, Git, SVN, Gerrit, Continuous Integration, Continuous Deployment, Jenkins.**

1.INTRODUCTION

Currently, the Central build Version Control system SVN(Subversion) is used, where simultaneously the repository cannot be accessed by different developers to write and read from the repository, to overcome this the distributed version control system has been used that is Gerrit. By using Gerrit, more than one developer can connect to the repository simultaneously. This project is about Gerrit migration where the svn repository is migrated to Gerrit to take the advantage of a distributed version control system The methodology that has been adopted in this project helps the Developers to access repositories simultaneously from a different location, different server and developer can simultaneously commit the latest updates to the repository that developers are working on. Version-control benefits groups to illuminate these sorts of issues, taking after each individualistic misshape by each engenderer and profiting ordering cognate work from getting clashed. Adjustment wiped out to one portion of the program can cause struggle with the variation done by another engineer who is contributing concurrently. The forerunner complication must be recognized and derived in a cautious way without decelerating or ceasing the work of other designers within the group.

2. RELATED WORK

The new project has to be created for a new repository to migrate to a Gerrit. In Gerrit Login page create a new project. Every organization has its requirement for executing a new repository, first, developer need to access the self-service portal followed by a particular team name, respective team having the requirement to create and has to give the project name, description and if owner want initial commit then can proceed with that else it is fine without initial commit also once the program is executed the repository will automatically create if any issues while creating a repository there must be access problem to the developer, need to contact to the service portal wait for the access to be granted then proceed with all the requirements to create a new project. Creating a project user needs to enter a respective team name, project name (repository name), and description for the project(testing purpose or production environment). On the server-side, a computer program store is regularly overseen by source control or store supervisors. A few of the store directors permit to total other store areas into one URL(Uniform Asset Locator) and give a caching intermediary. When doing nonstop builds numerous artifacts are delivered and frequently centrally put away, so naturally erasing the ones those are not discharged is critical. Once the store is effectively made. At that point continue with the movement portion.

3. SYSTEM ARCHITECTURE

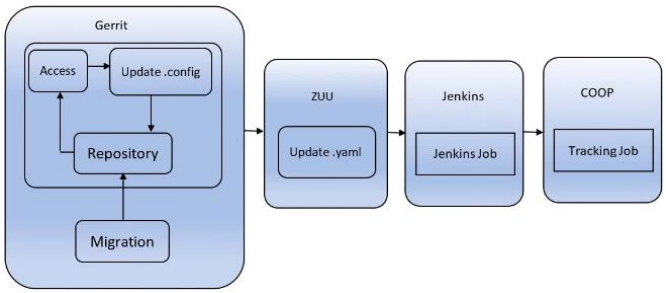

Fig -1: Block Diagram

It has following sections 1. Create a new repository in gerrit. 2. Giving access to the repository. 3. Updating the Project.config file. 4. Zuul .yaml file updating. and 5. Migrating the contents from svn to Gerrit.

3.1 Gerrit

Gerrit is an open source tool and web-based collaboration tool, it is also used for code review purpose and to track the commits to the repo. Gerrit includes git enabled ssh and https server compactable with all git clients.

3.1.1 Create a new repository in gerrit

Every organization has its requirement for executing a new repository, first, developer need to access the self-service portal followed by a particular team name, respective team having the requirement to create and has to give the project name, description and if developer wants initial commit then developer can proceed with that, else it is fine without initial commit also. Once the program is executed, the repository will automatically create.

3.1.2 Giving access to the repository

The Code-Review name is arranged upon the creation of a Gerrit occasion. The code esteem ranges from -2 to +2. Directors have all scores (-2 to +2) accessible, -3 is set as the default. Venture Proprietors have constrained scores (-2 to +2) accessible, -2 is set as the default. Enrolled Clients have restricted scores (-1 to +1) accessible, -1 is set as the default. The code -2 is so unpleasantly incorrect/buggy/broken that it must not be submitted to this venture or this department. This esteem is substantial over all fix sets within the same alter, i.e. the analyst must effectively alter his/her survey to something else some time recently the alter is submittable. This might not be consolidated. The code -1 doesn't see right or might be done in an unexpected way, but the analyst is willing to live with it as-is in case another commentator acknowledges it, maybe because it is way better than what is as of now within the project. Regularly typically moreover utilized by contributors who do not just like the alter but too aren't dependable for the extend long-term and hence do not have the ultimate say on alter accommodation. The code says Didn't attempt to perform the code audit errand or looked over it but don't have an educated conclusion, however. The code +1 looks right to this analyst, but the commentator doesn't have get to the +2value for this category. Regularly usually utilized by supporters to a extend who were able to review the alter and like what it is doing, but don't have last endorsement over what gets submitted. The code +2 Looks great, affirmed. Essentially, the same as +1, but for those who have the ultimate say over how the extend will create.

3.1.3 Updating the Project.config file

Clone the required repo to the local repository by using a command "git clone <git server>", Go to that folder where the repository is cloned. In the current directory update the config file by using any editor like either in visual code or VI editor. Then check the modified file by command "git status", update the file by a command "git add ." , make a new developing branch and switch to that branch. Commit the file with a command -- git commit -m--added the FILE. Yaml file". This is the staging area after this stage Push the file to main repository by the command --git push origin master--with the developing branch.

3.1.4 Migrating the contents from svn to Gerrit

Creating a authors list for the respective repository from the SVN server. Once the author list is created then need to create temporary folder where svn contents are cloned. Change the current directory to that temporary folder, with in the temporary folder Clone the SVN repo into as a Git. If repository is having large number of xml files which requires more storage then it needs artifactory support, it means for cloning and pushing the contents which consists of more storage like in terms of GB. Like 50GB for these types of repositories it is time consuming and data also required more and requires more storage capacity. So, to overcome this git lfs support is needed. Follow these commands to migrate the SVN contents to Gerrit with the help of git lfs support.

Install the git support in the GitHub server "Install the git lfs". After installation it will pop up with the message "git LFS initialized". To track the XML files "git lfs track "*.xml" ". After tracking the files, it will pop up the message -- Tracking "*.xml" –. If folder is different then it will pop up the message "Not a git repository". To Config the git lfs file "Config the lfs", Adding .gitattributes to git lfs support. and to "cat .gitattributes", It will pop up with the message -- *.xml filter=lfs diff=lfs merge=lfs -text. --. To track the modified files. "git status", Add the files to move to new repository by moving to staging area use the command "git add" . Commit the changes that have made to push to "git commit -m "<message>", Add the Git Hub repository origin where files have to move "git remote origin <gerrit server>". To check the server is added or not "git remote – v", if it is existing It will pop up with the message: "fatal: remote origin already exists". If it is newly added it will give newly added repository server name like. Origin <server name > (fetch) and origin <server name > (push). Push the contents to repository to remote GitHub "git push origin master". Moving SVN branches and tags to the Git repository. List the tags and branches in the SVN repo, "git tag <tag name>" and "git branch < branch name>" , Push the tags and branches to local Git by a command "git push -all". SVN has trunk,

branches and tags in similar way GitHub also having Master, branches, and tags. The difference is the code which is migrated in trunk is goes to master in GitHub. Here master is the main branch where the SVN contents in the trunk get moved to master in Gerrit server repository.

3.2 Zuul

Zuul may be a CI gating system. It may be a program that's utilized to door the source code store of a extend so that changes are as it were combined on the off chance that commit pass tests. The most component of Zuul is the scheduler. It gets occasions related to proposed changes, triggers tests based on those occasions, and reports back. ZUUL configuration is written in yaml file. File extension should be in .yaml only.

3.2.1 Zuul .yaml file updating

Clone the required repo to the local repository by using a command “git clone <git server>”. Go to that folder where the repository is cloned. In the current directory update the .yaml file in any editor like either in visual code or VI editor. Check the modified file by command “git status”, if no changes added to commit it will pop up the message use "git add" and/or "git commit -a", then update the file by a command “git add .” , make a new branch and switch to the new branch. Utilize the command 'git push' to send all the neighbourhood commits to the inaccessible stores by the command, “git thrust beginning HEAD:refs/for/master” with the creating branch.

3.3 Jenkins CI Build

Jenkins is an open-source computerization apparatus composed in Java programming dialect that permits nonstop integration. Jenkins builds and tests the computer program ventures and persistently making it simpler for engineers to coordinated changes to the venture and making it less demanding for clients to get a new construct.

3.3.1 Jenkins Job

The scripts which are in Domain specific language(DSL) are written in .groovy and Cb_funtions, which are written in shell scripts. These are adapted to git version control system replacing with svn. To add the details like getting variable from svn to git. Getting the tags from the svn to git , updating the internal ECL(Environmental code lines) file and header files, release recipes this information should be pushed to Gerrit by running a Jenkins jobs through a continuous integration. In Jenkins it has pre-CI and Post CI jobs. During the pre-CI it will validate the code, whether all the indentation is correct or not and programme structure. It is an independent job. During the pre-CI one more stage where manual intervention takes place to give code review +2 to -2 that how code is efficient to move to post stage, which is a dependent job which checks the jobs in parallel. During a post check It will Post the modified content to GitHub. Once the Jenkins job is success code gets merge to master.

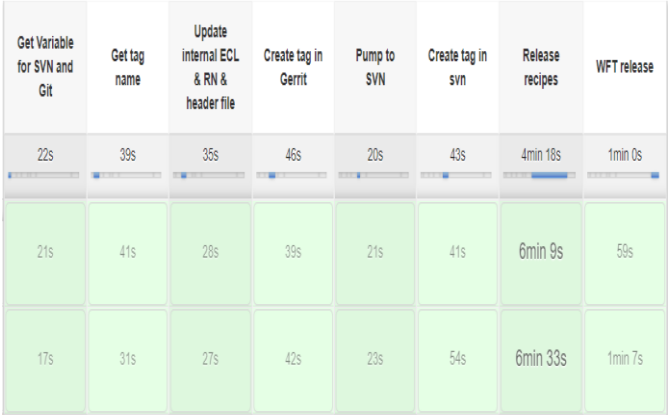

Fig -2: Jenkins Job

3.4 Coop

COOP is developed for the participation of two or more substances or other specialists to deliver a combined impact more noteworthy than the whole of their isolated impacts. During the software continuous integration process, huge amounts of data will be generated, but those data spread across too many places to synergistic this, Developers are in a process of creating an end-to-end view when it comes to the connections between CI systems and to show how the content propagates from the commit to system tests of a build that contains that commit, to have a Continuous Integration data visualization purpose.

3.4.1 Tracking job

Repo configuration is done with Project name and gerrit server name. Internally it connects to the gerrit server and track the recent commits updated by the developer to the respective repo by system component provided during the repo configuration. Python scripts is implemented to post repo configuration to coop by json template.

4. CONCLUSIONS

Source code management(SCM) is an integral part of software development for all organizations. Continuous Integration, deployment and Continuous delivery are of utmost importance because tasks like builds and testing needs to execute regularly on a daily basis in continuity. Jenkins, Gerrit, Git, MOBXterm and other tools help in the SVN(subversion) to Git migration process and making the task easier. Jenkins tools and plugins, are used to carry out automations and static code analysis, reduce the burden of manual execution and repeated procedures. The use of migration is to get the latest commit and logs, developer can switch to any feature branch by keeping the present work in staging area, developer can work offline, only while pushing commits to repository developer need an internet connection. In any case data in the central server is lost can be backup by any of the developer’s local machine.

REFERENCES

[1] M. Mukadam, C. Bird and P. C. Rigby, "Gerrit software code review data from Android," 2013 10th Working Conference on Mining Software Repositories (MSR), 2013, pp. 45-48, doi: 10.1109/MSR.2013.6624002.

[2] W. Yiran, Z. Tongyang and G. Yidong, "Design and implementation of continuous integration scheme based on Jenkins and Ansible," 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), 2018, pp. 245-249, doi: 10.1109/ICAIBD.2018.8396203.

[3] S. Elsen, "VisGi: Visualizing Git branches," 2013 First IEEE Working Conference on Software Visualization (VISSOFT), 2013, pp. 1-4, doi: 10.1109/VISSOFT.2013.6650522.

[4] Karthik Pai B H, Vasudeva Pai, Devidas, Deeksh S N, Rahul Rao, "A Prologue of Git and SVN" International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249 – 8958, Volume-9 Issue-1, October 2019.

[5] https://nokia.sharepoint.com.

[6] S. O. Dmitriev, D. A. Valter and A. M. Kontsov, "System for Efficient Storage and Version Control of Arbitrary File Collections," 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2020, pp. 295-298, doi: 10.1109/EIConRus49466.2020.9038922.

[7] S. Mysari and V. Bejgam, "Continuous Integration and Continuous Deployment pipeline Automation Using Jenkins Ansible," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020, pp. 1-4, doi: 10.1109/ic-ETITE47903.2020.239.

[8] A. Irfan and A. U. Rahman, "Tracking the Ripple Impact of Code Changes through Version Control Systems," 2019 International Conference on Frontiers of Information Technology (FIT), 2019, pp. 262-2623, doi: 10.1109/FIT47737.2019.00056.

BIOGRAPHIES

Suresh, PG Student, Department of Electrical and Electronics Engineering, RV College of Engineering, Bengaluru.