International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 2, April 2023, pp. 1817~1826
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i2.pp1817-1826

Performance assessment and analysis of development and operations based automation tools for source code management

Pooja Mittal, Poonam Narang
Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak, India

Article Info

Article history:

Received Mar 26, 2022
Revised Sep 29, 2022
Accepted Oct 24, 2022

Keywords:

Automation
Automation tools
DevOps
Software development
Source code management

ABSTRACT

Development and operations (DevOps), an accretion of automation tools, efficiently reaches the goals of software development, test, release, and delivery in terms of optimization, speed and quality. Diverse set of alternative automation tools exist for different phases of software development, for which DevOps adopts several selection criteria to choose the best tool. This research paper represents the performance evaluation and analysis of automation tools employed in the coding phase of DevOps culture. We have taken most commonly followed source code management tools-BitBucket, GitHub actions, and GitLab into consideration. Current work assesses and analyzes their performance based on DevOps evaluation criteria that too are categorized into different dimensions. For the purpose of performance evaluation, weightage and overall score is assigned to these criteria based on existing renowned literature and industrial case study of TekMentors Pvt Ltd. On the ground of performance outcome, the tool with the highest overall score is realized as the best source code automation tool. This performance analysis or measure will be a great benefit to our young researchers/students to gain an understanding of the modus operandi of DevOps culture, particularly source code automation tools. As a part of future research, other dimensions of selection criteria can also be considered for evaluation purposes.

Corresponding Author:

Poonam Narang
Department of Computer Science and Applications, Maharshi Dayanand University (MDU)
Rohtak, Haryana-124001, India
Email: poonam.mehta20@gmail.com

1. INTRODUCTION

Development and operations (DevOps), a fine fusion of development and operations teams together for speedy, reliable and quality delivery of software. For this purpose, DevOps employs a set of different automation tools at each stage of software development. These tools help in automating different activities or tasks of teams from the development of software till production. This research work revolves distinctively around the coding phase of DevOps culture. Different automation tools that exist for the purpose of source code management (SCM) include Gerrit, BitBucket Cloud, GitLab, Mercurial, GitHub Actions, and Team Foundation Server (TFS). Management goes around with several selection criteria or accustomed activities viz. code security, vulnerability management, internet protocol (IP) whitelisting, in-built continuous integration and continuous delivery (CI/CD), and commit hooks (pre and post). to ease out the critical task of best performer tool selection out of alternative automation tools list. To peculiarly focus on the challenges of Information technology (IT) management or software development teams, present analysis considers categorization of selection criteria into three dimensions viz. Development and architecture, code security and maintenance and support. To get a crisp clear understanding of these criteria we have focused on one dimension of selection criteria. These evaluation or selection criteria are compared for three representative SCM automation tools-BitBucket cloud, GitHub Actions and GitLab, taken from alternative sets of automation tools based on current market demand and their utility scenario. These representatives are studied in-depth to find out their weightage and evaluation score as per their requirement in the real world. For the fulfillment of which, case study of TekMentors Pvt Ltd as software development industry has also been taken into consideration. These computed figures are considered to evaluate their performance on the basis of final outcome after analysis. As the presence of these criteria in the tool, escalate the software development along with its delivery process up to much greater extent, so the research here involves the quantification of SCM tools in terms of overall or final score based on these several selection criteria. The score computed can be used in the selection of the best performer tool. The structure of the research work in the form of schematic diagram is described in Figure 1.

Figure 1 clearly lists the steps that are to be followed for the completion of underlying research work. The analysis performed can be used by researchers/students to realize best automation tool in source code management phase of DevOps culture. Research can be further extended to cover either remaining tools or other dimensions of selection criteria for source code management phase in DevOps culture.

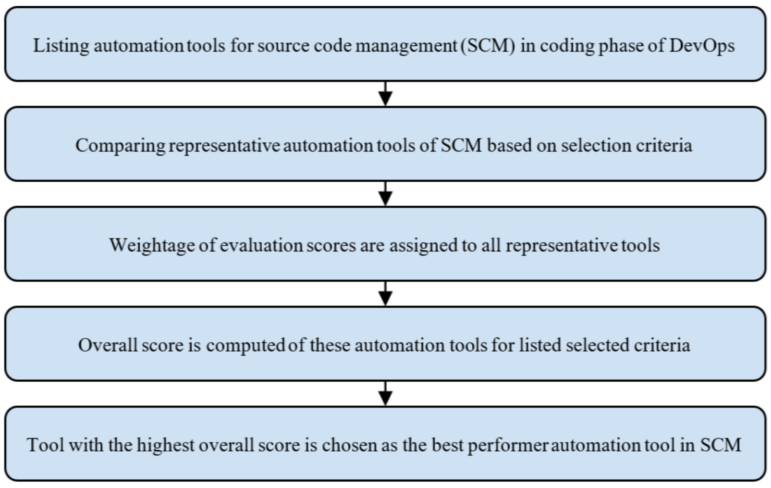

Listing automation tools for source code management (SCM) in coding phase of DevOps
Comparing representative automation tools of SCM based on selection criteria
Weightage of evaluation scores are assigned to all representative tools
Overall score is computed of these automation tools for listed selected criteria
Tool with the highest overall score is chosen as the best performer automation tool in SCM

Figure 1. Schematic flow of tasks performed in comparing and evaluating SCM automation tools

2. RELATED WORK

The selection/evaluation criteria, tasks, activities or practices considered by DevOps for the selection of appropriate source code automation tools have been reviewed with respect to different existing research papers. As Erich *et al.* [1], in their paper “A qualitative study of DevOps usage in practice,” highlighted that software industries are focusing on speeding up the development process and also improving the quality of their software by introducing the term DevOps as a portmanteau term to describe their efforts. The report details the DevOps implementation practices used by enterprises and the results they see as a result. In their article, Perera *et al.* [2] and Erich *et al.* [3] conducted research to determine how DevOps practices enhance software quality. They show through the use of multiple regression analysis that culture, automation, measurement, and sharing are crucial elements in raising the caliber of software. DevOps based software development was recommended to achieve high quality software.

DevOps has been recognized by Lwakatare *et al.* [4] as a crucial component of the continuous deployment paradigm in academic research circles and practitioner communities. It was also characterized by collaboration, automation, and measurement and monitoring as four major contributors to the DevOps movement. Riungu-Kalliosaari *et al.* [5] collectively in their paper on DevOps confirms benefits of adopting DevOps by conducting qualitative multiple-case study. This study acknowledges that DevOps practitioners foster stronger departmental collaboration, which improves communication and worker well-being. Dwivedi *et al.* [6] and taking into account additional DevOps research that addresses software quality concurrently and examines the effects of DevOps features on software quality. This study provided a methodical mapping of the effect of DevOps on software quality. The primary research areas were DevOps automation, culture, continuous delivery, and quick feedback. In their works on DevOps, Olguin [7] and Jabbari *et al.* [8]

emphasize that the goal of the DevOps movement is to automate the processes of continuous delivery of new software updates while simultaneously ensuring their accuracy and dependability. The definition of DevOps has also been thoroughly reviewed in the literature, and Jabbari *et al.* [8] concur that DevOps expands the agility element in the software development paradigm.

The literature also shows the development of several models for the guidance of industry or software practitioners for fruitful and successful implementation of DevOps in practice. Many models have been proposed in this context like the unicorn framework proposed by Trihinas and others [9], to overcome different challenges of DevOps through continuous releases. In a similar vein, the DevOps Research and Assessment model proposed by Forsgren *et al.* [10] discusses successful product delivery, and Hussaini [11] accepts the emerging DevOps paradigm as a response to the growing understanding of the gap between the development and operation teams functions of an organization, which he refers to as the "4 Cs" (communications, cooperation, culture, and collaboration). The "Wall of confusion" between these teams is also accepted by authors. Conflicting motives among people, processes, and technology/tools all contribute to this "Wall". Thus, there is a need to improve the coordination between the Dev and Ops teams. For improving the effectiveness and efficiency of DevOps stakeholder's interest, the concept was also described in [11]. As many renowned researchers restrict the adoption of DevOps in practice though there are multiples of theories that are against DevOps application and talk about its challenges and lack of performance measure [12]. For example, Leite *et al.* conducted a survey in their paper [13] and discussed different challenges in DevOps adoption. Other researches also force the compulsion of DevOps practices for the organization to move towards delivering higher performance and quality software [14].

The individual automation tool discussion that is conducted as the core component of this study work was missing from the literature review studied, mentioned above, or discussed thus far. At the source code management stage of software development, we also conducted a comparative analysis of automation tools to get the best out of many alternatives present in current market scenario. Underlying research has selected BitBucket, GitHub Actions; GitLab has been taken into consideration as per the tool market demand and also based on most commonly followed tools.

3. ALTERNATIVE AUTOMATION TOOLS EXISTENCE FOR DIFFERENT PHASES OF SOFTWRE DEVELOPMENT IN DevOps CULTURE

Different set of tools exists to render automation in all software development phases of DevOps culture. These automation tools help development and operations teams to speed up the tasks of software release. Existence of different automation tools in DevOps is shown in Table 1. Table 1 includes alternative and most commonly followed automation tools at different stages of software development. These automation tools ease the tasks of development and operations teams but challenges exist to choose the best tool from the corresponding list of alternative tools. Tools chosen prior to development not only save development time but also reduce the cost of rework at the same time. Current research works around the coding phase of development to comparatively analyze the best SCM automation tool for the underlying phase. To conduct this work, three most commonly followed SCM tools have been taken are BitBucket, GitHub Actions, GitLab. These tools are compared to one another based on their needful characteristics and requirements of the business market. Results of the comparison helps to choose the best out of many available alternatives.

Table 1. Alternative automation tools for different phases of software development in DevOps culture

DevOps Phase	Automation Tools
Requirement management	Confluence, BookStack, Notion
Project planning	AgileCraft, Trello, Jira
Coding	BitBucket, GitHub Actions, GitLab
Build	Teamcity, Jenkins, Bamboo
Testing	UFT, TestComplete, Selenium

4. SCM TOOLS FOR CODING PHASE OF DEVOPS SOFTWARE DEVELOPMENT CULTURE

Primary aim of DevOps is to optimize the end-to-end delivery process of the software. For this purpose, DevOps employs different automation tools at each and every phase of development life cycle with the objective to cut down the manual efforts and pace the cycle time from requirement to software release. This optimization process starts from the requirement gathering/management phase where it is recommended to use agile practices and continuous prioritization of requirements. Once the development team has prioritized a subset of requirements in the form of stories, the development team picks the stories in their priority order and starts working on the software development.

After the completion of the continuous prioritization phase, the next critical phase is coding. There are N numbers of developers involved in the coding phase where they all work on different requirements viz. new feature development, and enhancement of existing features or bug fixes. All these works go in parallel and code of all these developers bundled together to create artifacts like *.jar, .war, .ear, .egg* that can be deployed to the production environment. In coding or development phase, the most critical aspect is to resolve the issue of SCM. One needs to have an efficient SCM tool that can not only handle versioning of source code but is also enriched in features that make developers life easy along with handling complex code merge issues, well structuring of code, security, reporting and analytics capability, code scalability, and availability. For this purpose of best performer SCM tool selection step by step procedure followed by coding team is shown diagrammatically in Figure 2. As depicted in Figure 2, automation tool selection process for a project includes requirement identification of tool for the project, selection of the vendor for tool followed by computing weightage or evaluation scores for all alternative tools and finally choosing highest scorer or performer automation tool as the best suitable tool for the project. This paper works on the third step of the selection process that includes assigning weightage and computation of evaluation and overall scores afterwards selection of best performer tool based on several selection criteria. For this analysis purpose, we have taken three market leader tools as the representatives of SCM tools-BitBucket, GitHub Actions and GitLab. A tabular comparative analysis is performed for these source code management tools based on several selection criteria covered in the following section.

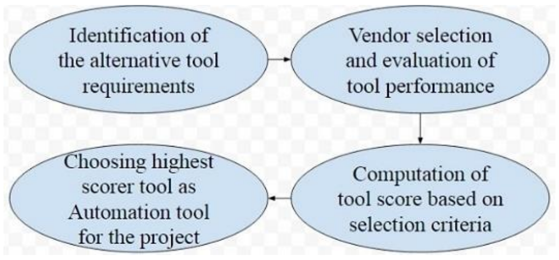

Identification of the alternative tool requirements
Vendor selection and evaluation of tool performance
Computation of tool score based on selection criteria
Choosing highest scorer tool as Automation tool for the project

Figure 2. Steps for project tool selection process from the set of available automation tools

5. SELECTION CRITERIA FOR THE ACQUISITION OF OPTIMAL SCM TOOLS

The acquisition of the best SCM automation tool in coding phase of DevOps depends on several selection criteria that are taken into consideration. Based on these criteria or evaluation parameters, automation tools are compared to one another and the tool getting the highest score for those generic features/criteria is chosen for the particular project. For the better understanding of these criteria and comparative analysis, these selection criteria have been categorized into three different dimensions-development and architecture, code security and maintenance and support. These different selection criteria that are considered under these dimensions are also been tabulated in Table 2.

Table 2. Categorization and description of different dimensions of selection criteria for automation tools

Selection Criteria	Description
Development and Architecture	
Support for multiple Repo in Project	Maximum number of repositories
Access management	Managing permissions at project level
In built CI/CD support	In built feature to build test, deploy code
Feature request	Addition of new features in SCM tool itself
Issue board	Single tool option for all manageable
In-built code review	Allow to contain multiple observable
Multiple reviewers	Authorization to multiple reviewers for same repository
Code snippet	Shareable and reusable piece of code
Hooks	Mechanism to get integration information
Rest APIs	GUI interface for interaction purpose
Code Security	
Code encryption (In-Transit/At Rest)	Code migrated to vendor cloud is encrypted
Scanning repo for Password/Keys	Scanning of repositories for password/keys
Vulnerability scanning	Ability to highlight vulnerability in the code
Maintenance and Support	
Import code from other SCM codes	Import code from other hosted repositories
Multiple LDAP integration	Integration with active directory
In-premise tool integration	Integration with other tools

There exist many selection criteria for tool excerption and for their crisp understanding we have categorized them into three pillars or dimensions as shown in Table 2. For this current research, we are targeting only development and architecture dimension of selection criteria that covers multiple of parameters like support for different repositories, access management, and issue board. All these parameters for selected criteria are explained beneath.

5.1. Support for multiple repositories in project

Source code of any application can be organized in many different ways. Legacy way of managing the code is to logically divide the code based on application functionality, where all the code is packaged together in the form of artifacts and deployed at once. In this form of delivery, complete code will be deployed at once even if there is a change expected in any single file or feature. After the deployment of code, need arises for stringent management of deployed code to maintain the integrity, consistency, uniformity along with unambiguous features. This need advances towards the SCM Automation tool, a much more optimized way of managing the code, employed by DevOps culture in which application is divided into multiple components and each component is independent of each other and can even be deployed to production individually. Architecturally each component can be developed as a micro service where all micro service code is separate from each other and each micro service can be deployed to production independently without any impact on the availability of other services.

In this type of model, during the deployment or release process, all the application would be available to users except only one feature of the application. In order to support this type of flexible and scalable architecture, each micro service code should be stored separately in their respective repository and each repo has its own CI/CD pipeline to support deployment of individual micro service as assigned in the Figure 3 that explains the steps to create a new repository in GitHub. Hence it becomes most important that SCM tools support multiple repositories within a single project where each repository can store and manage the code of each application component i.e., micro service.

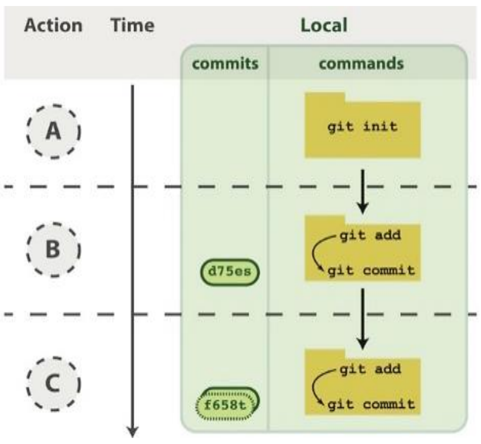


Figure 3. Creating new repository in GitHub source code automation tool [15]

5.2. Access management-project level/repository level/both

With the increase in microservices culture, there is always a need for a separate code repository for each micro service. Each micro service has a predefined set of objectives and functionalities for delivery to different stakeholders out of which only a very small number of stakeholders need access to all microservices e.g. the solution architect of the complete software may require all Microservices. In order to manage these selection criteria well, there is a need for provision of both project level code and repository level access with different sets of roles viz. admin, read-only, read-write, and reviewer.

5.3. In-built CI/CD support

While tools like BitBucket, GitLab and GitHub Actions are primarily used for source code management; there is a huge demand for additional features that can ease out many tasks of developers along with less maintenance overhead and requirement of low learning curve. One of these most demanding features in the list is in-built support for build and deployment, which can optimize the end-to-end delivery process up to much greater extent. After the code commit statement by the developer, this in-built support feature of the tool automatically kicks the build pipeline integrated with required repository to bundle the code into deployable artifacts, runs the unit test, followed by the execution of quality checks, automated functional test, performance test, and security test. Another CI/CD feature is also made available in the SCM tool itself that makes the overall DevOps tool stack simpler to use, maintain and scale. BitBucket pipeline, GitHub actions and GitLab CI are examples of SCM Tools having such features.

5.4. Feature request

Few organizations take feature requests attributes available in the source code management tool itself to post the release date, backlog management, and prioritized list of features on the tool to make their product open source. These features also help in source code management. Figure 4 mention different steps to add backlog feature in automation tools as creation of issue, development and quality review. Most of organizations also use likes, votes, watchers attribute of issue to analyze which feature is awaited by most of the users. Usually, the community picks the most sought-after feature and fixes that for all. Thus, the organization looks for in-built support for feature request in the source code management tool.

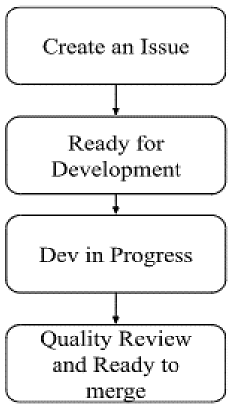


Figure 4. Steps for adding or managing backlog feature with ease in automation tools

5.5. Issue board

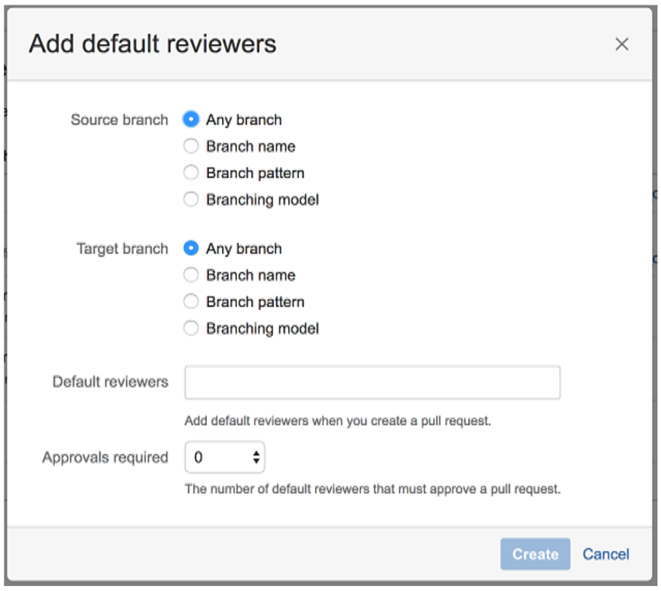
Issue board is one of the critical features that companies are looking for in the source code management tool. Issues could be anything like a new feature request, and enhancement of an existing feature or a defect in the application. This feature becomes more important for smaller companies or startups where they can use a single tool for source code management as well as for taking issues regarding their requirement backlog management. Having an issue board within the source code management tool has multiple advantages like cost-saving, automatic traceability between source code and requirements including less training needs for the whole team.

5.6. In-built code review

Another critical aspect is the code review that contains multiple observables to consider regarding product design. Examples of code review could be checking of code similarity with the design, correspondence of design with issued guidelines, confirming of the code optimization with application demands of performance, and response time. Modern SCM tools like BitBucket, GitHub Actions and GitLab provide code review features within their tool itself. In most of the systems it is called a pull request. Instead of directly committing the code to the main branch, developers raise a pull request which goes to one of the senior developers or architects of the team who reviews the code and approves the merge request of the code from feature or story branch to the main branch from where CI/CD pipeline will be kicked off. Having this feature already available with the source code management system saves developer time as they can see the review comments in the same tool. They can also fix the code directly and again raise the pull request if required. Now no need to run two different tools, no switching between tools required that again optimize the hand holding process.

5.7. Multiple reviewers

In organizations there is usually more than one person who is authorized to review the code so that if one person is not available it should not delay the commit or the CI/CD process. Snapshot of authorizing multiple reviewers in one repository is represented in Figure 5. Ideally code review process should have the flexibility to assign a review to multiple reviewers like senior developers or the architect, so that anyone who has bandwidth to do the code review can pick that review request and give the feedback with no dependency on a single reviewer.


Add default reviewers

Source branch: [✓ Any branch] / Branch name / Branch pattern / Branching model

Target branch: [✓ Any branch] / Branch name / Branch pattern / Branching model

Default reviewers:

Add default reviewers when you create a pull request.

Approvals required: 0

The number of default reviewers that must approve a pull request.

Figure 5. Adding of multiple reviewers for different branches of source code in management tools [16]

5.8. Code snippet

Code snippet is a small piece of code that can easily be shared with anyone in the team. It is a reusable piece of code that we can make public so that other people can copy it or refer to it for common functionality. Generally, this code snippet feature is more important for companies who are doing open source projects or for a community that is supporting any open source project. This snippet of code can be stored in separate files from where it can be referred to. It also has the advantage of adding other people to put comments to allow them make any changes in the code to make it more improved and optimized. These Snippets could be a single file or a collection of files.

5.9. Hooks (pre-commit/post-commit/WebHooks)

Hooks are the mechanism to create and validate information from other tools that are integrated with source code management tools. Generally, event-based, specific event triggers hooks do this work of validation for other tools. Hooks are of different types viz. pre-commit hook, post-commit or WebHooks. Consider a good example of a pre-commit hook that validates ticket ID in the commit message. Now if a developer is working on a story or defect, it is mandatory for the developer to mention the story/defect ID from the task management tool like Jira or Trello but if that issue ID is not mentioned in the commit message, then the source code management tool can block that commit. This block will not only help to maintain the automatic traceability from the requirement to source code but also help the developers to maintain the code of the live application. These hook features prevent the developers from introducing the regression defect or breaking the existing functionality of the application.

In a similar way, post-commit hook helps to trigger any action in other tools after the commit. Examples could be resolving tickets in task management systems like Jira, and Trello. after the developer commits the code for that particular ticket or triggers the build in tools like Jenkins, Bamboo, and Teamcity. Tools from the same vendor are integrated implicitly by application links like BitBucket, JIRA, and Bamboo. Hence implementation of pre/post commit hooks is through native integration but for cross vendor tool integration like BitBucket-Teamcity, BitBucket-Jenkins needs WebHooks.

5.10. Rest application programming interface (APIs)

Most of the DevOps tools expose both graphical user interface and APIs for interaction. Rest API is important to communicate the tool programmatically. Organizations create their own tool to extract data for analytical purposes. Not all the reports required by the organizations are available off the shelf by the tool. Thus, they need to write custom solutions to extract the data using rest APIs. This section clearly describes the need or usage of different selection criteria for development and architecture dimension. These criteria are taken into consideration before the actual selection of the automation tools. Other criteria given in Table 2 are also checked for select the best automation tool, but currently this paper works on parameters under development criteria.

6. PERFORMANCE ANALYSIS OF SOURCE CODE AUTOMATION TOOLS-A TABULAR COMPARISON BASED ON SELECTION CRITERIA

Many alternative source code automation tools exist for managing code in DevOps culture. This research has taken BitBucket Cloud, GitHub Actions, and GitLab as representative tools for source code management based on their market analysis and common usage. These tools are assigned scores in 3-Steps: Firstly, a numerical value called weightage/weight is assigned to each selection criteria. Weights are assigned as per the merits or requirements of criteria to their target customers. Next step includes assigning an evaluation score to the SCM automation tool as per the selection criteria or corresponding feature availability in the tool and the last step multiplies the weightage with the corresponding evaluation score and finally adds them for each tool to get an overall score for that tool. These assigned and evaluated scores for all the representative tools are mentioned in a comparative tabular format as shown in Table 3.

Table 3. Performance analysis table containing weightage assigned along with evaluation and overall score assigned to the representative SCM automation tool [17]–[25]

Selection Criteria	Description	Weigh-tage	Evaluation Score (out of 10)			Overall Score=Weightage * Evaluation Score		
			Bit-Bucket Cloud	Git-Hub Actions	Git-Lab	Bit-Bucket Cloud	Git-Hub Actions	Git-Lab
Dimension-Development and Architecture								
Support for multiple Repo in Project	Maximum number of repositories which can be created in a single project/group	10	9	9	9	90	90	90
Access management-project level/ Repository Level/Both	Managing permissions at project level as well as at repository level	10	7.5	7 [18]	8.5	75	70	85
In built CI/CD Support	In built feature to build test, deploy code	8	6.5 [20]	8	8	52	64	64
Feature Request	Ability to build features requests within SCM tools	8	7	8	8	56	64	64
Issue Board	In built features to manage tasks with help of Kanban and scrum boards	6	5	8	7.5	30	48	45
In-Built Code Review	Ability to perform code review within the tool	9	8.5	8.5	9	76.5	76.5	81
Multiple Reviewers	Ability to add multiple reviewers for a single review/pull request	9	8	8.5	9	72	76.5	81
Code Snippet	Snippets allow to share files with yourself, members of your workplace	7	7.8	9	7.5	54.6	63	52.5
Hooks [Pre-Commit/Post-Commit/WebHooks]	Ability to enforce hooks/merge check functionalities	10	7.5	7.5	8.5	75	75	85
Rest API	usage matrix and other data available as Rest API call	9	7.5	8.5	9 [18]	67.5	76.5	81
Best In-Class reporting and analytics	In built report for all below mentioned items					0	0	0
Most commit by persons	For a specific repository who is the active contributor	7	7	8	7.5	49	56	52.5
Most active Project/Repo	Most active repository of a project based on commits	7	0	0	0	0	0	0
Inactive Repo/Branch	Inactive Repositories and Branch	7	0	0	0	0	0	0
Commit Frequency	Frequency of commits for a repository	7	7	7.5	8.5	49	52.5	59.5
Merge/Pull request Pending/Overdue	Pull request dashboard for repository/Project	9	7.5	9	8.5	67.5	81	76.5
						814.1	**893**	**917**

Analysis Table 3 is the result of implementation and evaluation with respect to all mentioned features with SCM tools-BitBucket, GitLab, and GitHub for TekMentors Consulting Pvt Ltd. Software company and renowned researchers already referred to in the paper. Tool recommended is also being used in companies for the efficient management of source code. Based on the overall score of all the representative SCM tools, GitLab can be concluded as the best performer tool among others.

Several different functionalities or features available in GitLab, as seen in the performance analysis table, makes it the most commonly observed automation tool for source code management. This can further be analyzed from interest over time report for previous 6-7 years according to Stack Overflow report. Among these representative SCM tools, again GitLab shows the highest usage trend in last years based on stack overflow report as depicted in the Figure 6.

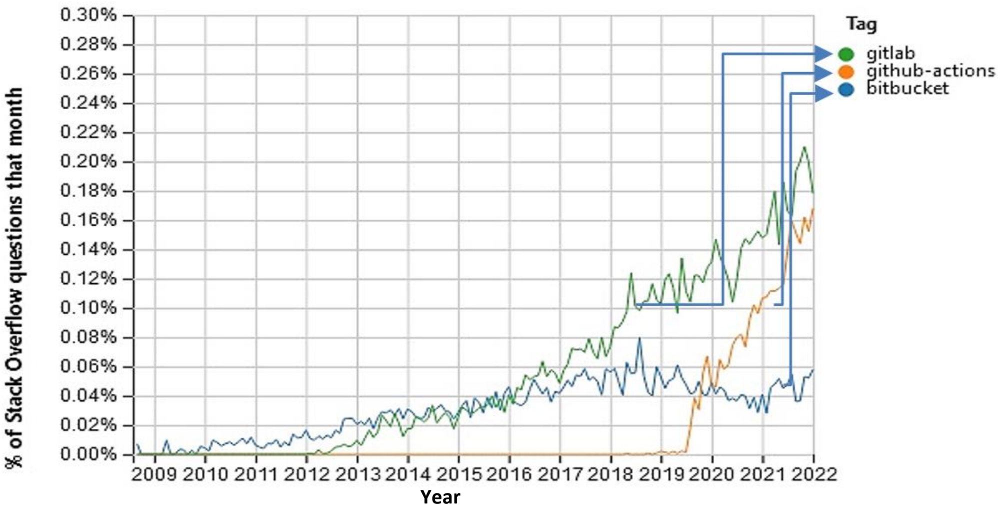


Figure 6. SCM Automation Tools interest over time depicted by stack overflow latest trends report over past 7-8 years [26]

7. CONCLUSION AND DISCUSSION

This paper discussed many tasks or activities as the selection criteria to choose the best performing source code automation tool for the coding phase of DevOps software development culture. As there are a number of selection criteria in the market and these were not narrowed till, so this was our idea to first categorize them into three dimensions in order to smoothen out the selection process of automation tools. For this work, we have only focused on the development and architecture dimension of selection criteria. Opting on these criteria not only helps to manage the source code with the selection of the right set of automation tools but at the same time it also escalates the development process up to a much greater extent. This research computes the performance of the representative tools on the basis of weightage assigned, evaluation and overall scores along with the case study of TekMentors Pvt Ltd. The analysis performed reveals GitLab as the best source code management tool depending upon the outcome of overall score. GitLab is selected as the best performer tool on the basis of several selection criteria that are generic in nature. Other alternative automation tools for source code management can also be chosen as per the requirements of project-specific criteria. This performance or comparative analysis of source code automation tools will be useful for students/researchers to learn in-depth about DevOps culture and usage of alternative automation tools at different stages of software development. A performance evaluation or comparative analysis of other automation tools may also be carried as a part of further research. For future scope, our work can also be extended to carry out research on other two dimensions of selection criteria to complete the process of code management automation tool selection in the software development process.

REFERENCES

[1] F. M. A. Erich, C. Amrit, and M. Daneva, "A qualitative study of DevOps usage in practice," *Journal of Software: Evolution and Process*, vol. 29, no. 6, Jun. 2017, doi: 10.1002/smr.1885.

[2] P. Perera, R. Silva, and I. Perera, "Improve software quality through practicing DevOps," in *2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, Sep. 2017, pp. 1–6, doi: 10.1109/ICTER.2017.8257807.

[3] F. Erich, C. Amrit, and M. Daneva, "A mapping study on cooperation between information system development and operations," in *Product-Focused Software Process Improvement*, Springer International Publishing, 2014, pp. 277–280.

[4] L. E. Lwakatare, P. Kuvaja, and M. Oivo, "Dimensions of DevOps," in *Lecture Notes in Business Information Processing*, Springer International Publishing, 2015, pp. 212–217.

[5] L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen, and T. Männistö, "DevOps adoption benefits and challenges in practice: a case study," in *Product-Focused Software Process Improvement*, Springer International Publishing, 2016, pp. 590–597.

[6] A. Dwivedi, G. Srivastava, S. Dhar, and R. Singh, “A decentralized privacy-preserving healthcare blockchain for IoT,” *Sensors*, vol. 19, no. 2, Jan. 2019, doi: 10.3390/s19020326.
[7] R. J. Olguín, “DevOps challenges and implications,” University of Murcia, Murcia, Spain, pp. 1–5, 2019.
[8] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer, “What is DevOps?,” in *Proceedings of the Scientific Workshop Proceedings of XP2016*, May 2016, pp. 1–11, doi: 10.1145/2962695.2962707.
[9] D. Trihinas, A. Tryfonos, M. D. Dikaiakos, and G. Pallis, “DevOps as a service: pushing the boundaries of microservice adoption,” *IEEE Internet Computing*, vol. 22, no. 3, pp. 65–71, May 2018, doi: 10.1109/MIC.2018.032501519.
[10] N. Forsgren, M. C. Tremblay, D. VanderMeer, and J. Humble, “DORA platform: DevOps assessment and benchmarking,” in *Lecture Notes in Computer Science*, Springer International Publishing, 2017, pp. 436–440.
[11] S. W. Hussaini, “Strengthening harmonization of development (Dev) and operations (Ops) silos in IT environment through systems approach,” in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2014, pp. 178–183, doi: 10.1109/ITSC.2014.6957687.
[12] A. A. Khan and M. Shameem, “Multicriteria decision-making taxonomy for DevOps challenging factors using analytical hierarchy process,” *Journal of Software: Evolution and Process*, vol. 32, no. 10, Oct. 2020, doi: 10.1002/smr.2263.
[13] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, “A survey of DevOps concepts and challenges,” *ACM Computing Surveys*, vol. 52, no. 6, pp. 1–35, Nov. 2020, doi: 10.1145/3359981.
[14] R. Bolscher and M. Daneva, “Designing software architecture to support continuous delivery and DevOps: a systematic literature review,” in *Proceedings of the 14th International Conference on Software Technologies*, 2019, pp. 27–39, doi: 10.5220/0007837000270039.
[15] “Documentation for BitBucket server 7.5,” *Atlassian*. 2020, Accessed: Sep. 15, 2022. [Online]. Available: https://confluence.atlassian.com/alldoc/files/278071997/1019389387/1/1597212077409/BitbucketServer-075.pdf.
[16] R. Banerjee, “Backup blog, using the BitBucket API,” *Rewind*. 2021, Accessed: Sep. 15, 2022. [Online]. Available: https://www.backhub.co/blog/using-bitbucket-api.
[17] A. Q. Gill, A. Loumish, I. Riyat, and S. Han, “DevOps for information management systems,” *VINE Journal of Information and Knowledge Management Systems*, vol. 48, no. 1, pp. 122–139, Feb. 2018, doi: 10.1108/VJIKMS-02-2017-0007.
[18] S. Uzunbayir and K. Kurtel, “A review of source code management tools for continuous software development,” in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, Sep. 2018, pp. 414–419, doi: 10.1109/UBMK.2018.8566644.
[19] S. Bokefode, R. N. Tendulkar, and R. R. Nakade, “Project management system like BitBucket (App),” *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 6, pp. 5240–5243, 2020.
[20] S. M. Mohammad, “Continuous integration and automation,” *International Journal of Creative Research Thoughts (IJCRT)*, pp. 2320–2882, 2016.
[21] B. Vasilescu, S. van Schuylenburg, J. Wulms, A. Serebrenik, and M. G. J. van den Brand, “Continuous integration in a social-coding world: empirical evidence from GitHub,” in *2014 IEEE International Conference on Software Maintenance and Evolution*, Sep. 2014, pp. 401–405, doi: 10.1109/ICSME.2014.62.
[22] L. E. Lwakatare, P. Kuvaja, and M. Oivo, “An exploratory study of DevOps extending the dimensions of DevOps with practices,” *ICSEA*, vol. 104, 2016.
[23] H. L. Akshaya, J. Vidya, and K. Veena, “A basic introduction to DevOps tools,” *International Journal of Computer Science and Information Technologies*, vol. 6, no. 3, pp. 5–6, 2015.
[24] M. Shahin, M. Ali Babar, and L. Zhu, “Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices,” *IEEE Access*, vol. 5, pp. 3909–3943, 2017, doi: 10.1109/ACCESS.2017.2685629.
[25] I. Sommerville, *Software engineering*, 9th Editio. Pearson, 2010.
[26] “Stack overflow trends,” *Stack Overflow*. https://insights.stackoverflow.com/trends?tags=bitbucket%2Cgithub-actions%2Cgitlab (accessed Sep. 17, 2022).

BIOGRAPHIES OF AUTHORS

Pooja Mittal obtained her Ph.D. degree from Maharshi Dayanand University. Her area of research and specialization include data mining, data warehousing, and computer science. She had published more than 50 research papers in renowned International and National Journals and attended more than 30 Conferences. Currently she is working as Assistant Professor in the Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak (Haryana). She can be contacted at mpoojamdu@gmail.com.

Poonam Narang Research Scholar, pursuing Ph.D. from the Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak, Haryana under the supervision of Respected Dr. Pooja Mittal (Research Guide and Second Author). Author’s Qualification is M.Phil. (CS), MCA. She had attended many National and International Conferences including Springer and IEEE and also published many research papers. She can be contacted at email: poonam.mehta20@gmail.com.