Securing Version Control System using DevOps by Key-Vault and Identity Access Management

Vinay Singh*

Department of Computer Science, University of Petroleum and Energy Studies, Dehradun, Uttarakhand, India

Editorial

Received: 24-Apr-2023, Manuscript No. GRCS-23-96853; **Editor assigned:** 27-Apr-2023, Pre QC No. GRCS-23-96853(PQ); **Reviewed:** 12-May-2023, QC No. GRCS-23-96853; **Revised:** 19-May-2023, Manuscript No. GRCS-23-96853 (R); **Published:** 26-May-2023, DOI: 10.4172/2229-371X.14.S1.001

***For Correspondence:**
Vinay Singh, Department of Computer Science, University of Petroleum and Energy Studies, Dehradun, Uttarakhand, India
E-mail: usavinaysingh@yahoo.com

Citation: Singh V. Securing Version Control System using DevOps by Key-Vault and Identity Access Management. J Glob Res Comput Sci.2023; 14:001.

EDITORIAL NOTE

Securing Version Control System (VCS) using DevOps is a critical aspect of software development. Key Vault and Identity Access Management (IAM) are two important tools that can help ensure the security of the VCS.

Key Vault is a cloud-based service that allows us to securely store and manage cryptographic keys, certificates, and other secrets. By using Key Vault, we can keep the sensitive data separate from the application code and ensure that only authorized users have access to it.

IAM, on the other hand, is a set of policies and procedures that control who has access to the VCS and what actions they can perform. By implementing IAM, we can ensure that only authorized users have access to the VCS, and we can also monitor and audit user activity to detect and respond to any security threats.

Together, Key Vault and IAM can help ensure the security of VCS by providing a secure way to store and manage sensitive data, and by controlling who has access to VCS and what they can do with it. By integrating these tools into the DevOps workflow, we can ensure that security is a top priority throughout the software development lifecycle.

Since the last decade the advancement in software, and networks and the migration from virtualization to a niche containerization eco-system made the long-sought vision of cloud computing possible. Cloud providers like Amazon, Google, and Microsoft compete with a wide portfolio of pay-as-you-go services. These services have the potential of sparking new, innovative, and affordable products more than simple IT outsourcing.

Current advancements toward future IOT devices will necessitate the collection and analysis of data from integrated devices such as distributed storage, intelligent loads, and distributed energy resources.

Organizations that want to avoid playing catch-up with threats use threat intelligence to enable proactive defenses. Danger knowledge empowers security crews to anticipate impending dangers and focus on really to confiscate them. Threat intelligence can also be used by security teams to improve their judgments and speed up incident response and remediation [1-3].

Version Control Systems, or VCS for short, are specialized systems that are frequently used to simplify the concept of version control. In recent times, these systems have undergone numerous modifications. Revision Control Systems (RCS) and Source Code Management tools (SCM) are other names for VCSs. With integrations that keep them working in the tools and contexts, key vault and cryptography make operation security nearly invisible to innovators. Solutions that automatically remediate vulnerabilities and help developers across the organization prioritize without false positives will earn developer trust [4-6]. Source code is, obviously, the groundwork of any product advancement practice. In this way, it is important that source code is appropriately overseen without bringing bottlenecks or imperfections into the conveyance pipeline.

The benefits of a version control system

The advantages of a version control system include a thorough long-term change history of every file, which implies that every modification made over the years by numerous people is documented. Teams are now able to examine the addition, deletion, and editing of datasets that have been made since the first copy. Version Control Systems additionally offer a smooth and continuous flow of code changes, avoiding developers from getting lost or missing any updates [6-8]. Teams can collaborate, preserve and restore earlier versions of files, track changes, and back up data with the use of version control systems.

There are several benefits of a version control system

Single source of truth: Version control ensures that every team member is working with the most recent version of a file or project by providing a single source of truth for tracking changes and resolving conflicts. As a result, the code base has fewer errors and inconsistencies, allowing teams to deploy code more safely and confidently [9-11]. Developers may easily track changes and roll back to earlier versions by using a version control system to keep a complete audit record of all code changes.

Manage code conflicts and deployment risk: Code conflicts must be managed and deployment risk must be minimized, and version control systems are essential for this. VCS enables teams to work together seamlessly and continuously without overwriting each other's work by offering a common repository for all code modifications. By doing this, code conflicts are reduced and made easier to resolve [11-13].

Code reviews are another feature of VCS that can help lower deployment risk by identifying errors and inconsistencies early in the development cycle. This results in a more reliable and stable codebase, making deployments smoother and less risky.

Enable parallel development streams: In order to enable simultaneous development streams, version control systems are crucial technologies. VCS enables team members to simultaneously work on various areas of the code base and seamlessly merge their changes [14-16]. With the help of a version control system, each team member can establish their own branch of the code base, work on their modifications independently, and then merge their changes back into the main branch.

Parallel development streams are made possible by this technique, which also guarantees a conflict-free and seamless integration of modifications. Every file in a VCS also has a complete long-term change history, allowing teams to trace changes and, if necessary, roll back to earlier versions.

Maintain a full audit trail: Utilizing a Version Control System has many advantages, one of which is the capacity to keep a complete audit trail of all codebase modifications. Understanding who made what modifications and when is crucial for tracking changes over time. A VCS makes it simple to inspect a file's whole history, including all additions, deletions, and changes. This makes it simple to locate flaws and problems and determine how, when, and where they were first introduced [16]. Additionally, a VCS makes it simple to roll back modifications that cause new defects or problems by allowing us to go back to earlier versions of the code. Overall, maintaining a full audit trail is critical for ensuring the reliability and stability of the code base over time.

Reduce bugs through code review: In order to eliminate problems and raise the quality of the code, code review is a crucial step in the software development process. Code modifications made by developers are reviewed to make sure they adhere to standards and are free of flaws and vulnerabilities. Developers can find flaws and inconsistencies early in the development process and stop them from being deployed to production by evaluating code changes.

Code review assists in enhancing the overall quality of the code base by pointing out areas for development and giving developers input [17]. This criticism may come in the form of recommendations for enhancing code readability, enhancing code performance, or following best practices and coding standards.

Deploy consistently across environments: Maintaining a trustworthy and stable codebase requires consistently deploying across environments. We may make sure that the code is consistent in all settings, from development to production, by using a version control system. This is accomplished by utilizing branching and merging strategies in version control system, which let has make distinct branches for various environments and integrate changes between them as necessary [18]. Additionally, by automating the deployment process and reducing the chance of human mistake, solutions like containerization and infrastructure-as-code can help maintain consistency. We may lower deployment risk and make sure the code base is reliable and stable over time by deploying consistently across environments.

Simplify rollback: Reverting modifications made to a code base to a previous state is known as rolling back. Selecting the version of the code we wish to go back to and restoring it are the simple steps involved in a rollback in version control systems. This is made feasible by VCS, which keeps track of all codebase modifications and enables developers to quickly revert to earlier iterations [19]. When modifications need to be swiftly undone because they introduced faults or errors into the code base, rollback is helpful. VCS gives developers a safety net that enables them to experiment and make changes without worrying that they would irrevocably alter the codebase by simplifying rollback.

Release faster: A strong CI/CD pipeline and a clearly defined development methodology are necessary for speedy product release. While Continuous Deployment (CD) automates the process of deploying code changes to production, Continuous Integration (CI) entails frequently integrating code changes. As a result, iteration and feedback cycles can move more quickly, and errors can be found and fixed sooner in the development cycle.

Version Control Systems like Git can assist manage code conflicts and offer a complete audit trail of changes to enable parallel development streams [20]. Implementing code reviews also contributes to the bug reduction and code quality improvements. It is simpler to deploy consistently across environments and rollback modifications as necessary when a code base is reliable and stable.

This Special Issue aims at publishing high-quality manuscripts covering new research on topics related to the Integration of cloud computing and Big data for better IOT utilization including but not limited to the following:-

- Cloud migration and cloud security
- DevOps security and vulnerabilities
- Version control system and its benefits
- Public, private, and hybrid clouds
- DDOS attacks and how to prevent them.
- Security in micro services and containerization
- Virtualization vs. Containerization
- Internet on Things (IoT)

Topics of interest include, but are not limited to

- Intelligent prediction and recommendation for IoT decision-making.
- Novel big data analytics technology for IoT security.
- Data confidentiality and privacy protection for IoT.
- Lightweight IoT data transmission and communications.
- Authentication and access control for data usage in IoT.
- Security in DevOps.
- Active directory and key vault in the version control system.

CONCLUSION

In version control systems provide many benefits that make it an essential tool for DevOps organizations. By using a version control system, teams can maintain a single source of truth, manage code conflicts, enable parallel development streams, maintain a full audit trail, reduce bugs through code review, deploy consistently across environments, simplify rollback, and release faster. However, despite these benefits, it is important to keep in mind that version control systems also come with security risks. DevOps organizations should prioritize security by implementing key-vault and Identity Access Management, utilizing threat intelligence for proactive defenses, and staying ahead of evolving threats. By prioritizing security and using version control systems effectively, DevOps organizations can streamline their development process while maintaining the highest level of security.

REFERENCES

1. Vinay S, et al. The transition from centralized (subversion) vcs to decentralized (git) vcs: a holistic approach. J Electr Electron Eng.2019; 12:7-15.
2. Sussman BC, et al. Version control with subversion for subversion 1.6. Create Space Independent Publishing Platform. 2010.
3. Nazatul NZ, et al. Version control system: A review. Procedia Comput Sci. 2018; 135: 408–415.

4. Vinay S, et al. A pivot focus on big data impediments & the benefits of using machine learning. Doctoral Colloquium- 2019 in Management, Engineering, Computer Science, Design, Life Sciences & Law.
5. Chamarty S, et al. An authorization scheme for version control systems. SACMAT. 2011; 6:123–132.
6. Singh V, et al. Event driven architecture for message streaming data driven microservices systems residing in distributed version control system. IEEE (ICISTSD). 2022; 8:25-26.
7. Jasmin R, et al. Which change set in git repository is related?. IEEE (QRS). 2016; 8:01-03.
8. Yusuke S, et al. How do GitHub users feel with pull-based development. 7th international workshop on empirical software engineering in practice. IEEE (IWESEP).2016;3:13-13.
9. Hou Q, et al. An empirical study on inter-commit times in SVN. Int Conf on Software Eng and Knowledge Eng. 2014;132–137.
10. Singh V, et al. Performance analysis of middleware distributed and clustered systems (PAMS) concept in mobile communication devices using android operating system. International Conference on Parallel, Distributed and Grid Computing. 2014;2: 11-13.
11. Kolassa C, et al. A model of the commit size distribution of open source. Proc. the 39th Int'l Conf (SOFSEM'13).2013;209:52-66.
12. Singh V, et al. Advantages of using containerization approach for advanced version control system. IEEE (ICERECT). 2022;10:26-27.
13. Hofmann P, et al. Estimating commit sizes efficiently. Proc. the 5th IFIP WG 2.13 Int'l Conf. Open-Source Systems (OSS'09), Sweden.2009;5:105-115.
14. Kolassa C, et al. A Model of the commit size distribution of open source. International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'13).2013; 5266:26-31.
15. Singh V, et al. DevOps based migration aspects from legacy version control system to advanced distributed VCS for deploying micro-services. IEEE (CSITSS).2021; 1-5.
16. Singh V, et al. Advance micro services based approach for distributed version control processing using the sensor-generated data by iot devices. IEEE (ICERECT).2022;12:26-27.
17. Hindle A, et al. What do large commits tell us? A taxonomical study of large commits. Proc. the 5th Int'l Working Conf Mining Softw Repos.2008;5:99-108.
18. Singh V, et al. Improving business deliveries using continuous integration and continuous delivery using jenkins and an advanced version control system for microservices-based system. IEEE IMPACT. 2022; 11 :26-27.
19. Aggarwal A, et al. A Rapid transition from subversion to git: time, space, branching, merging, offline commits & offline builds and repository aspects. Recent Advances in Computer Science and Communications.2022;15:653-660.
20. Singh V, et al. identification of the deployment defects in micro-service hosted in advanced vcs and deployed on containerized cloud environment. Int. Conference on Intelligence Systems ICIS-2022.