Continuous Resilience: DevSecOps Strategies for Cloud and Quantum Platforms

Robert TICU-JIANU
Bucharest University of Economic Studies, Romania
ticujianurobert19@stud.ase.ro

The state of the technological landscape in the software development field has been in a continuous change and evolution at a fast pace ever since its beginnings. Only a few decades ago, the waterfall model was the best-in-class approach to the Software Development Lifecycle (SDLC) with most projects and teams being on the smaller side and the industry still making its baby steps. Nowadays, this approach is mostly seen as inefficient and adopted with disdain, since many projects use some form of an agile methodology. By synthesizing theoretical concepts with empirical evidence, this paper aims not only to investigate the benefits and drawbacks of implementing a DevOps methodology in software products, but also to act as a stepping stone for practitioners that want to accelerate the velocity with which a product goes from requirements to the production environment, regardless of the nature of the product: classic or quantum.

Keywords: *DevOps, DevSecOps, Continuous Integration, Continuous Deployment, Resilience, security, Quantum computing, Cloud computing*
DOI: 10.24818/issn14531305/28.4.2024.05

1 Introduction

Ever since its apparition, the term DevOps has always found itself under the aegis of the infinity sign since it is a beacon of continuous improvement of processes within the Software Development Lifecycle (SDLC). It is a methodology which brings together multiple philosophies, processes and practices, that when implemented within a project, it offers a tremendous improvement in terms of velocity and reliability of the software creation, quality assurance and delivery [1]. The term DevOps appeared as a fusion between the terms “Development” and “Operations”. It is sometimes used synonymously with the term CI/CD, since these are its 2 most important components: Continuous Integration and Continuous Delivery/Deployment. [2] The former denotes the approach in which developers make small changes to their code which are easier to test, integrate and build, therefore streamlining code changes, while the latter presents an automated delivery of the product to testing and deployment environments. [3]

Although considerable research has been devoted to providing characteristics and detailing the elements that constitute the DevOps culture, rather less attention has been paid to exactly makes it tick and how can a newcomer be aided in his path of adopting the culture. [1] While there is not a definitive answer to “What is the best approach to DevOps?” there are some milestones that can be followed to achieve this culture’s objective, like introducing the well-defined stages of SDLC, disseminating a collaborative culture between Developers, DevOps and Testing teams, automating certain time-consuming actions, monitoring the implemented systems, collecting feedback about the outcomes of the processes applied and then using it to improve the approaches followed. [3] [4]

The new DevOps approach was warmly welcomed into many projects and companies because it substantially improved the velocity with which the teams were moving through the SDLC, but usually moving at a faster pace comes with a price: attention to detail. The details that were omitted in this case mostly belonged to the security field. Developers embraced the quick pace, but have left themselves completely out in the open to bad actors while doing so. This is the moment in which the new DevSecOps approach emerged

and fixed the shortcomings of the previous ways. Although the DevOps and Security principles might seem contradictory to one another at first, in reality, they complement each other perfectly [5].

DevOps is focused on the bigger picture, accelerating the time to market of the product and the agility with which the products go through the SDLC, Security is observing the minute details and making sure that there are no cracks through which bad actors can squeeze. The obvious downside of checking all the steps is the amount of resources that needs to be spent while doing so, but this is exactly the biggest strength that the DevOps principles bring: speed and efficient use of resources. [6]

While these methodologies have seen the spotlight for classical computing, in the recent years a completely new perspective has surfaced: quantum computing. It is a transformative technology with huge potential to revolutionize entire industries, representing a pivotal moment in the history of computer science. It reconstructs the fundamental element of software development: the bit. In quantum computing, it is replaced by the qubit, which keeps the 2 possible states that the regular bit had (0 and 1) and adds a new third state known as a superposition [7] [8], which represents 0,1 and all the other positions that can be found in between. It also introduces a new concept of bit entanglement, in which 2 entangled qubits will always share the same state. [9]

Since it is still in its infancy, the integration of the quantum software development into the mainstream development practices, particularly the DevOps methodologies, has been somewhat overlooked by developers and organizations, since they have yet to fully grasp the implications that the quantum computing brings to the table. This paper aims to address this oversight by offering a platform that can address the scalability, reliability and efficiency needs of quantum software as well as classical one.

2 Literature Review

During the past two decades, a lot of information has become available on the practices of IT Operations and Development. A longitudinal study of the DevOps methodologies and the benefits offered by them was conducted by Gene Kim, the co-author of the book “The DevOps principles” and a distinguished within the IT Industry, renowned for his contributions to shaping the discourse and best practices around software development and IT operations. This study is drawing inspiration from his available literature as well as other seminal works from prominent figures in the DevOps world, to propose a strategy for developers seeking to improve the velocity of their projects.

One of the main benefits highlighted in the literature is the emphasis that DevOps puts on collaboration and communication between the development and operations teams. The research of *Kim, Behr and Spafford* (2016) [10] points out how removing the barriers and breaking down the traditional silos that formed between the two sides, help the entire project in its development. By sharing goals and responsibilities, DevOps encourages frequent communication and feedback [11] which directly translates to improved efficiency and product quality [12] [13]

Another benefit identified is the significant reduction of time that it takes for software product to arrive on the market. According to the State of DevOps Report [11], projects that embrace these methodologies have shorter development cycles, lower MTTR [14] (mean time to recovery) and faster deployment frequency. The feedback that the teams give and receive is also more qualitative [15] and it reaches each side faster thanks to the improved communication. [11] This also leads do a significant improvement of the code quality [16], since most bugs or smelly code are snuffed out by the quality assurance stage in which peer reviewing and static code analysis play a major role. [17]

Ultimately, all these benefits lead to an increase in team performance [1] [18] and costs reduction [19] [20] , since a lot of the repetitive and synchronous tasks are now either automated or broken down into smaller steps with improved velocity, allowing the

developers to work with increased efficiency and improving the value of the time spent on a task. [21]

Table 1 - The evolution of the agility, cost and risk of software delivery [21]

	1970s-1980s	1990s	2000s - Present
Era	Mainframes	Client/Server	Commodization and Cloud
Representative technology of the era	COBOL, DB2 on MVS, etc	C++, Oracle, Solaris, etc.	Java, MySQL, Red Hat, Python, etc
Cycle time	1-5 years	3-12 months	2-12 weeks
Cost	$1M-$100M	$100k-$10M	$10k-$1M
What is at risk	The whole company	A product line or division	A product feature
Cost of failure	Bankruptcy, sell the company, massive layoffs	Revenue miss, CIO/CTO's job	Negligible

Looking back on software products during the years, we can observe how their costs and that in the 1970s-1980s the SDLC could span across 1 to 5 years just to release a product to the public [21], the cost of such project could be anywhere between $1M and $100M dollars and the entire company was at stake if the product did not succeed. In the uneventful case in which the product flopped, most of the times the companies went bankrupt, had to be bought by another one or have massive layoffs. Just a decade later, the stakes have already lowered by a moderate amount: a cycle would take somewhere between 3 to 12 months, the cost of such project would start from $100k up to $10M, the risk was mainly around the product line or the division that was working on the product and the price of failure mostly consisted in revenue missed or it would cost the CIO/CTO their job. Nowadays, thanks to the multitude of new methodologies and technologies, the SDLC time was massively cut down to 2 to 12 weeks, the cost has shrunk to a range of 10k$-1M$ and the risk has been driven down to just discarding a product feature. In case of failure, the losses incurred by the company are negligible. These are just some of the benefits that came along with the Agile and DevOps methodologies.

In spite of the widespread recognition of the penalties went down significantly, while their agility only evolved. We can observe in DevOps principles and their benefits, companies are still slow to adopt these transformations into their work. A quick glance at the "Testing in DevOps 2022" survey ran by mabl.com, which can be seen in **Fig.** 1 - DevOps Transformation by Company Size, shows us that only about 40% of large enterprises (2000+ employees) have mostly or fully adopted DevOps principles. In addition, in small businesses or startups (1-100 employees) 22% of the employees were not sure of the progress the company made with adopting the methodology, while a staggering 32% only aspire to integrate it into their workflows.

The slow pace at which companies are adopting the DevOps methodology differs based on the size of the company. Large enterprises tend to have very complex organizational structures that is usually paired up with extensive bureaucracy and intricate established processes. These structures often cripple the swift acceptance of changes, therefore, the biggest hurdle for them, unsurprisingly is the slow rate at which changes are accepted due to the internal processes that create bottlenecks at multiple points within the flow. This is then followed by lack of internal expertise and technology

limitations.

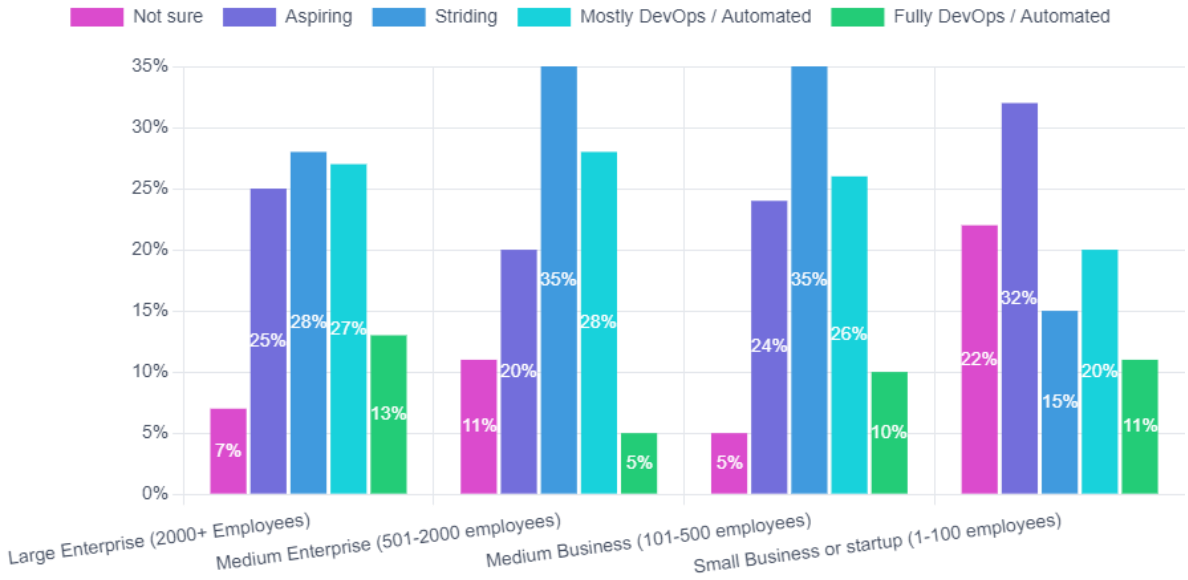

Fig. 1 - DevOps Transformation by Company Size [32]

According to **Fig.** 2 - DevOps Obstacles by Company *Size*, the challenges that small businesses and startups face reflect the unique dynamics and constraints that exist in such organizations. In this context, resource constraints and competing priorities are common, therefore organizational inertia appears and they tend to maintain status quo until a compelling reason appears to change directions. [10] This is also felt in the budgetary constraints, as disposable resources are scarce and they cannot afford to waste or distribute unwisely the already limited budget. Hence, for small businesses and startups, the first spot is taken by the lack of prioritization from the company to adopt changes, then next spots are occupied by budget and technological limitations and slow change adoption. [19]

DevSecOps is a sophisticated and comprehensive method of developing software that integrates security protocols into the DevOps workflow, guaranteeing that security is ensured within all crucial components of each stage of the software development lifecycle. This shift in perspective tackles the increasing demand for secure agile development methods from the beginning, as opposed to adding security features later on. Through the integration of security procedures into pipelines for continuous integration and deployment (CI/CD), DevSecOps fosters a mentality that views "security as code," promoting the early detection and correction of vulnerabilities. The development, security, and operations teams working together enhances the ability to rapidly produce secured software, lowers the risk of security breaches, and close-to ensures regulatory compliance. As a result, DevSecOps is an essential part of contemporary software development approaches since it enhances an organization's overall security posture while streamlining processes and speeding up delivery timelines. The DevSecOps methodologies are innate to the solution presented by this paper, by having incorporated checks from the coding stage, up to the deployment of the application. Automated security scans such as static application security testing and software composition analysis are present within the CI/CD and they offer the possibility to identify vulnerabilities so early into the code, that the technical debt become non-existent. The deployed applications also benefit from continuous monitoring and feedback, which would detect and alert on security issues or breaches.

Considering the context that is offered by the

previously presented state of the DevOps methodologies in the IT industry, this study proposes a strategy to fill the gap for software teams which are lacking agility in their SDLC. The literature reviewed also demonstrates the numerous benefits that the DevOps methodologies bring, ranging from: the culture of collaboration [22] and communication, encouraging experimentation and innovation with low or negligible costs, accelerated time to market, enhanced quality and stability, and improved flexibility and scalability of software products.

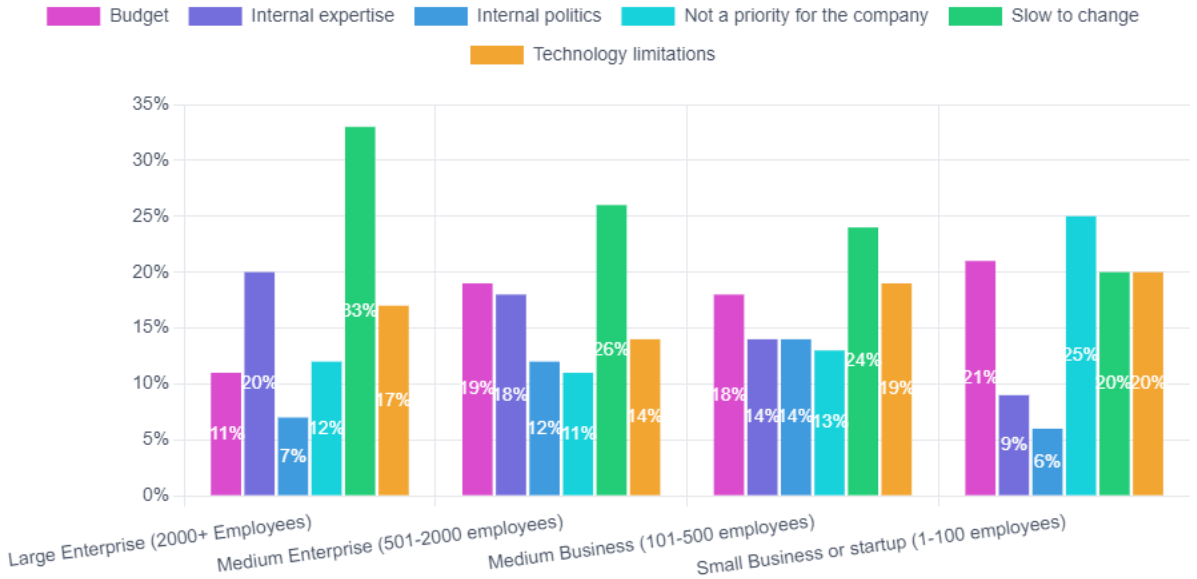

Fig. 3 - DevOps Obstacles by Company Size *[32]*

Due to its still nascent stage, the quantum software development is prone to exhibit noise-induced errors [23]. Factors like the quantum decoherence (the systems are still unstable and can be influenced by external disturbances like electromagnetic radiation, interaction between neighbouring particles, temperature fluctuations etc), imperfections in the fabrication of the qubits, gate operations errors (the manipulation of the qubits can exhibit inaccuracies in the control signals and timings) or simply by the limited quantum resources that are available at the moment. [24]

Furthermore, the quantum Software Development Life Cycle is more intricate than those of regular software due to a handful of factors. The quantum applications are heavily reliant on the scarce hardware, therefore it is coupled with the constraints that come with it: limited qubits, high error rates, short coherence times etc. The systems are noisy and prone to errors, hence error correction schemes are a vital part of the development, adding more complexity to the process. Before learning quantum algorithms, a strong foundation of linear algebra, quantum mechanics and optimization theory is required, which makes the learning curve relatively steep for newcomers. Because the technology did not have time yet to mature, the ecosystem around it is still in its beginning phases. Development tools, simulators, libraries, testing tools are all scarce resources that developers can hardly grasp in the current stage. [23]

Through factoring the nature of quantum software development and its prerequisites, this paper aims to demonstrate that quantum applications could highly benefit from the same benefits that DevOps methodologies bring to classical computing: strict, reproducible testing/vast validation procedures (to combat noisiness), continuous integration (reduces cycle length and risk of integration bugs), continuous deployment (scalability, deploy time, etc) and efficiency.

Currently some proposals for the integration of the DevOps methodologies already exist, an eloquent example being the work of *I.D.*

Gheorghe-Pop et al. [25], in which the "Quantum DevOps" concept is presented along with its value and benefits. The paper reminds the audience that while "quantum advantage" is a palpable achievement, it requires hefty resources and complex topologies, hurdles that could be more easily passed with the help of "Quantum DevOps".

3 Practical Analysis

Software projects, like any other project that aims to generate value, are usually complex and require multiple teams to work together to bring it to an end. In recent times, in IT projects a new kind of team was added to the landscape: The DevOps/DevSecOps team. It is a team that kicks in before the developers start their work. It starts by creating and configuring all the environments that the teams will be working in: from a repository in the version control system to the biome in which the solution will run. By having a distributed SVC repo, the DevOps team ensures that the codebase is properly set up for version control, enabling efficient tracking of changes and collaboration among developers. It breaks down and eliminates the hurdles faced by teams that are working asynchronously and remotely, since there is no longer a dependency on the other team member's work. Code can be changed, versioned, review and merged whenever the team deems appropriate. Next, the DevOps team sets up the Continuous Integration and Continuous Deployment (CI/CD) pipelines depending on the needs of the developers. This involves configuring automated processes for building, testing, and deploying the application, ensuring that code changes are continuously integrated and deployed. These steps require good communication between developers and DevOps since the scenarios can differ wildly depending on the plans of the project. Infrastructure needs to be deployed for both the testing environment as well as the production one. The unit, smoke, sanity and regression tests also need to be available to the DevOps team so that the automatization process can go smoothly.

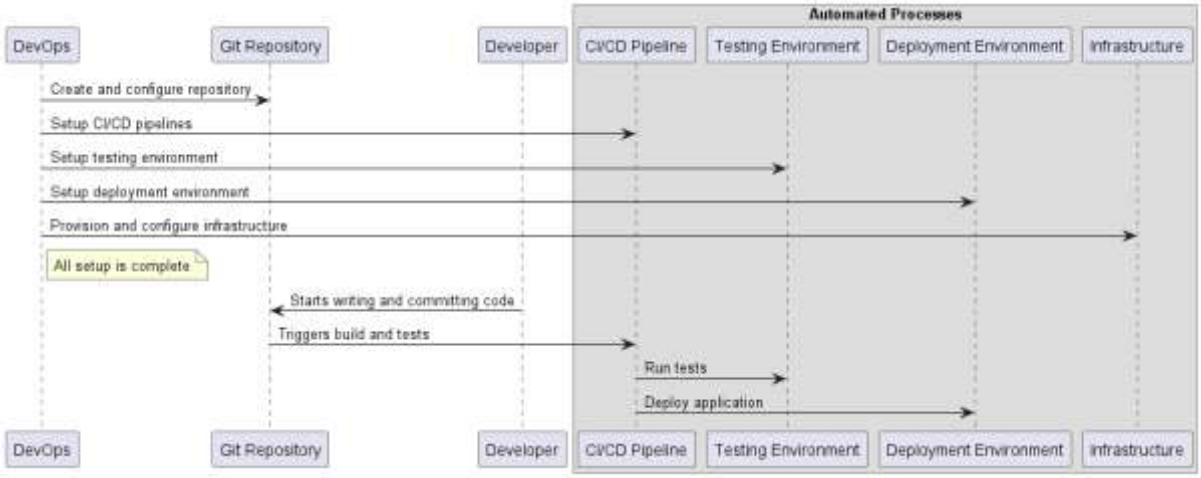

Fig. 5 - Classic DevOps Approach

This approach creates a bottleneck on the DevOps team, since the developers are not able to automatically build, test and deploy their code unless they:

1. Have a team of DevOps engineers
2. The DevOps team has done their due diligence and set up all the processes of automation
3. The pipelines are tested, available and resilient.

Fig. 5 - Classic DevOps Approach reveals why this approach inherently creates a significant bottleneck within the DevOps team, primarily because developers lack the capability to autonomously build, test, and deploy their code. This dependency on the DevOps team manifests in several critical ways:

- Developers are fundamentally reliant on having access to a dedicated team of DevOps engineers. These engineers are essential for establishing and maintaining the infrastructure and processes required for automated operations. Without a specialized DevOps team, developers cannot proceed with automated tasks, leading to delays and inefficiencies.
- The DevOps engineers ensure a very high quality of the pipelines and the set-up of all necessary automation processes. This means ensuring that each step from code integration to deployment is seamlessly automated. Any errors in these processes will lead to stalling the development lifecycle and deadlocks where neither team knows exactly which one caused the issue
- The project is also required to pay a very steep initial price in the kick off phase compared to the pay as you go service offered by the web platform. Establishing a team of DevOps engineers requires not only substantial financial investment in terms of salaries and benefits but also incurs costs related to recruitment, onboarding, and ongoing training.

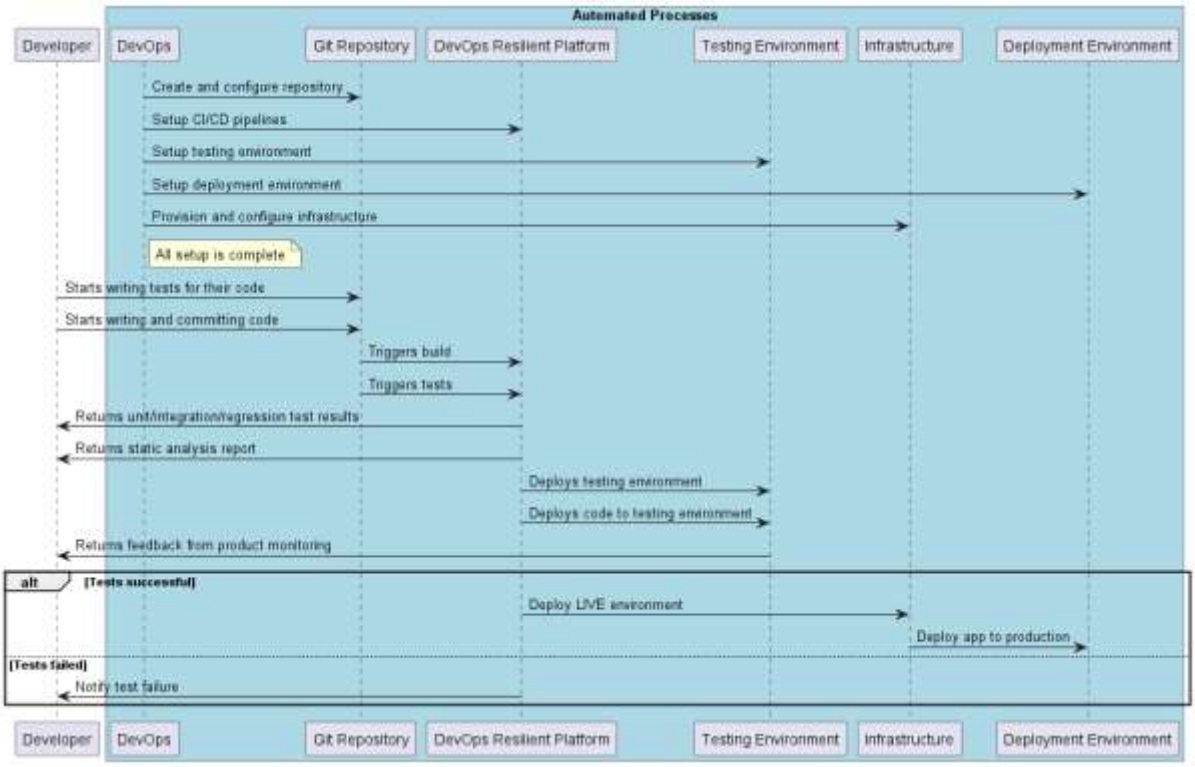

Fig. 6 - Resilient DevOps Solution Workflow

With the help of the presented DevOps Resilient Solution for Cloud and Quantum Computing, we can tackle all the aforementioned drawbacks of the classical DevOps approach with a regular CI/CD Software and DevOps engineers' team. The approach highlights the collaboration between developers and the provided DevOps platform without the upfront effort at the beginning and the costs of maintaining such mechanism, similar to **Fig.** 6 - Resilient DevOps Solution Workflow.

Since the platform is offered as a SaaS, the project could start building, testing, analysing and deploying their code since the first commit of the developers. The only prerequisites are the code written by the developers and the tests they want to run on their code. This completely removes the need of a DevOps team since the process has been automated by the platform. The web application offers flexibility to the project of switching technologies and the reliability of the pipelines and the infrastructure with no

time wasted upon waiting for another team to finish up their tasks. Every step is also transparent to the developer since the platforms offers feedback to the developer at all times. The components that contribute to the efficient running of the platform are analysed in this paper.

3.1 Automation server

The backbone of this solution consists of a Jenkins server, an open-source automation server that allows for automatization of stages from the SDLC such as building, testing, deployment etc. In this scenario, it is used for running the predefined pipelines by applying the data from the user (code, build parameters, tests, etc) to the existing templates for the supported technologies (Open QASM, React, .Net, etc.). The developer does not have direct access to this server. The technologies supported by the platform need to have a specific pipeline template in the Jenkins server which can be triggered remotely via the Jenkins API. More supported platforms can be added at a later time depending on the demand for them.The Automation server is configured for 2 main platforms: Web and Quantum. Each of them has a few technologies supported, each having stages that can be run: build, analysis, test, and deploy.

3.2 Front-end portal

The automation server is interfaced to the user through a ReactJS single web page application. The developer can log into the platform with his credentials or through different social platforms like Google, Facebook, GitHub, etc. Multiple options are available to the user: build, test or deploy your app. These choices require a link to a hosted git repository that can be accessed and downloaded by the server so that it can apply the specified operations further. The user can also provide build parameters, testing scripts or deployment preferences. When the portal has all the requested information, a pipeline is started via the REST API of Jenkins for the specified platform. The website asynchronously checks the server for data and prints in real-time the logs that are offered by the job as well as the artifacts that are produced: binaries, test reports, logs etc. In the meantime, the monetization of the platform is done through sheer use time of the computing resources, meaning that the time of usage of the runners is monitored. Upon the completion or cancellation of the pipeline, the duration in milliseconds is sent to a backend that keeps track of the costs that a user incurs, generating invoices that a user can afterwards pay through Stripe. The platform incorporates the best practices and architectural patterns for resilient and scalable cloud solutions presented by *Cornelia Davis* [26] such as CI, CD, microservices, orchestration etc.

3.3 REST API

Everything is glued together by an ExpressJS REST API that feeds data from the SQL database to the web portal. The backend also plays the role as a middleman between the Auth0 user database and the SQL database which holds the information about the users. The usage time, costs incurred, invoices etc are stored in this database correlated with Auth0. The API is secured via JSON Web Tokens provided by the Auth0 service based on the user's credentials/social accounts. To further increase the security of the system, the backend shall also act as a proxy between the outside medium and all the internals of the solution. For example, all requests that are issued by the user from the web portal shall be interfaced through the backend that will filter out the malicious requests and will sanitize the input that the automation server receives.

By providing a standalone DevOps platform tailored specifically for developers, the paper aims to alleviate the mentioned pain points by empowering developers to independently create, customize, and manage their CI/CD pipelines without the need for extensive DevOps expertise or external assistance. This solution not only accelerates the software delivery process but also promotes autonomy, innovation, and efficiency within development teams, ultimately driving organizational success in today's dynamic digital environment.

3.4 Auth0

It is a very popular IAM (Identity Access management) platform that is used to provide seamless signing in to the users with the help of features like SSO (Single Sign-On), MFA (multi-factor authentication), Social Identity Providers Sign In (Google, Facebook, GitHub, etc). It offers developers the possibility to securely integrate a module for user management, authorization, role management, user grouping and many more features, with ease and minimal impact on the whole application. It also supports a multitude of authentication protocols such as OAuth 2.0, OpenID Connect, SAML etc and provides developers with SDKs for most popular web platforms as well as an extensive documentation for them. Additionally, it alleviates the developers of plenty of security risks, such as: API Security, compliance with security regulations such as GDPR, HIPAA etc, Token Forgery, CSRF (Cross-site request forgery) etc.

3.5 Stripe

Stripe is a widely used payment processing platforms that helps businesses around the world to accept payments over the internet. It has an extensive suite of APIs, tools and SDKs that allow for easy integration of billing, revenue management and payment processing for online businesses. It offers a wide range of payment methods including digital wallets like Google Pay, Apple Pay, credit and debit cards, bank transfers and more. The developers also benefit from other vital features like fraud prevention, dispute resolution, support for international currencies and live support for both the business as well as the customers regarding their payments. It is a great bridge between the developers that get to use a very accessible technology and the project itself that benefits oof a platform to monetise the work done.

4 Conclusion

This paper has explored the DevOps methodologies, its benefits, prerequisites and its applicability in the quantum and classical software development, both domains which still have a lot of room for more extensive usage the aforementioned principles. The agility, culture of collaboration, efficiency, decreased time to market are just a few of entries off the long list of improvements [27] that could be brought into a software project by just implementing a CI/CD platform and following the guidelines provided not just by DevOps principles, but DevSecOps. As long as the field of quantum continues to evolve, the assimilation of classical software methodologies will be inevitable and it will help in accelerating the exit of the quantum computing from its nascent stage as well as achieving its full potential. This is only the first step towards a future where industries will be revolutionised by the quantum computing, accelerating innovation and reshaping the possibilities of the computer science and engineering worlds.

No matter the nature of the software (classical or quantum), nor the organization (or its size) that is developing it, by embracing the DevOps culture and adapting its principles, a project is positioned for better chances of success in the current competitive and rapidly changing environment. This paper leaves as a proposal a comprehensive and noteworthy architecture and implementation of a CI/CD platform that serves both cloud and quantum environments. By also including security aspects within it, this solution aims to not only improve process efficiency, but to also increase reliability in such system and ultimately opening the door for a wider use of quantum technology. Finally, the current research demonstrates that a well-executed CI/CD platform may act as a spark for innovation within the computer science industry, which will allow developers to fully utilize their resources in their development lifecycles.

References

[1] W. P. Luz, G. Pinto and R. Bonifácio, “Adopting DevOps in the real world: A theory, a model, and a case study,” Journal of Systems and Software, vol. 157, pp. 1-16, 2019.

[2] R. Jabbari, N. bin Ali, K. Petersen and B. Tanveer, "What is DevOps?: A Systematic Mapping Study on Definitions and Practices," in Proceedings of the Scientific Workshop Proceedings of XP2016, Edinburgh, 2016.
[3] L. E. Lwakatare, P. Kuvaja and M. Oivo, "Dimensions of DevOps," in Agile Processes in Software Engineering and Extreme Programming, Cham, 2015.
[4] F. Erich, C. Amrit and M. Daneva, "Report: DevOps Literature Review," University of Twente, Twente, 2014.
[5] H. Myrbakken and R. Colomo-Palacios, "DevSecOps: A Multivocal Literature Review," in International Conference on Software Process Improvement and Capability Determination, Cham, 2017.
[6] T. A. Chick, A. Reffett, N. Shevchenko and J. Yankel, "Modeling DevSecOps to Reduce the Time-to-Deploy and Increase Resiliency," Carnegie Mellon University, 2021.
[7] R. S. Sutor, Dancing with Qubits: How quantum computing works and how it can change the world, Packt Publishing Ltd, 2019.
[8] M. Hirvensalo, Quantum computing, Springer Berlin, Heidelberg, 2010.
[9] J. Vos, Quantum Computing in Action, Simon and Schuster, 2022.
[10] G. Kim, K. Behr and G. Spafford, The Pheonix Project, Portland: IT Revolution Press, 2013.
[11] N. Forsgren, J. Humble and G. Kim, Accelerate: The Science of Lean Software and Devops, IT Revolution Press, 2018.
[12] B. Fitzgerald and K. J. Stol, "Continuous software engineering: A roadmap and agenda," Journal of Systems and Software, no. 123, pp. 176-189, 2017.
[13] A. Katal, V. Bajoria and S. Dahiya, "DevOps: Bridging the gap between Development and Operations," in International Conference on Computing Methodologies and Communication, 2019.
[14] F. Joao, D. Adriano, R. Amaro, R. Pereira and M. M. da Silva, "DevOps benefits: A systematic literature review," Software: Practice and Experience, no. 9, pp. 1905-1926, 2022.
[15] A. Mishra and Z. Otaiwi, "DevOps and software quality: A systematic mapping," Computer Science Review, no. 38, 2020.
[16] M. Erder and P. Pureur, "Continuous Architecture: Sustainable Architecture in an Agile and Cloud-Centric World," Morgan Kaufmann, pp. 1-303, 2015.
[17] J. Humble and D. Farley, Continuous Delivery, Addison-Wesley Professional, 2010.
[18] R. Jabbari, N. Bin Ali, B. Tanveer and K. Petersen, "Towards a benefits dependency network for DevOps based on a systematic literature review," Journal of Software: Evolution and Process, pp. 1-26, 2018.
[19] G. Kim, The Devops Handbook, Trade Select, 2016.
[20] M. Hering and B. Ghosh, Devops for the Modern Enterprise, Trade Select, 2018.
[21] A. Cockcroft, Velocity and Volume (or Speed Wins), San Francisco: FlowCon, 2013.
[22] J. Davis and R. Daniels, Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale, O'reilly, 2016.
[23] J. Romero-Álvarez, J. Alvarado-Valiente, E. Moguel, J. Garcia-Alonso and J. M. Murillo, "Enabling continuous deployment techniques for quantum services," Software: Practice and Experience, pp. 1-25, 2024.
[24] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information, Cambridge, 2012.
[25] I. -D. Gheorghe-Pop, N. Tcholtchev, T. Ritter and M. Hauswirth, "Quantum DevOps: Towards Reliable and Applicable NISQ Quantum Computing," in IEEE, Taipei, 2020.
[26] C. Davis, Cloud Native Patterns: Designing change-tolerant software, Manning, 2019.
[27] A. Hemon, B. Lyonnet and F. Rowe, "From Agile to DevOps: Smart Skills and

Collaborations," Information Systems Frontiers, pp. 927-945, 2020.
[28] C. Dotson, Practical Cloud Security, O'reilly, 2019.
[29] B. Beyer, J. Petoff , C. Jones and N. R. Murphy, Site Reliability Engineering, O′Reilly, 2016.
[30] R. Kumar and R. Goyal, "On cloud security requirements threats vulnerabilities and countermeasures: A survey," Computer Science Review, pp. 1-48, 2019.
[31] M. Daneva and R. Bolscher, "What We Know About Software Architecture Styles in Continuous Delivery and DevOps?," in Springer, 2020.
[32] mabl.com, "Testing in DevOps 2022," mabl, 2022. [Online]. Available: https://www.mabl.com/reports/testing-in-devops-2022. [Accessed 25 April 2024].

Robert TICU-JIANU is a graduate of the IT&C Security Master of the Faculty of Cybernetics, Statistics and Economic Informatics within the Bucharest University of Economic Studies. His research aims to further accelerate the process of automating processes within the SDLC, regardless of whether the projects are of classical or quantum nature. While doing so, he also puts a lot of emphasis on the security aspects that need to be carefully addressed since his signature of choice is defined by resiliency and robustness in both his solutions and thoughts.