Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 9, No. 6, 2815-2826
2025
Publisher: Learning Gate
DOI: 10.55214/25768484.v9i6.8494

Development of internal developer platform for software development lifecycle optimization

Ahmet Vehbi Olgaç[1], Erkan Zileli[2], Gökhan Karadaş[3], Ceren Ulus[4], M. Fatih Akay[5*]
[1,2,3]Department of Technology - App. Delivery Platform, Trendyol, Türkiye; ahmet.olgac@trendyol.com (A.V.O.) erkan.zileli@trendyol.com (E.Z.) gokhan.karadas@trendyol.com (G.K.)
[4,5]Department of Computer Engineering, Çukurova University, Türkiye; f.cerenulus@gmail.com (C.U.) mfakay@cu.edu.tr (M.F.A.).

Abstract: New technologies are constantly developing, processes are progressing rapidly, and many items that need to be followed and managed are emerging. In this context, the need for new approaches to increase efficiency, collaboration, and innovation for software development and technology teams is constantly increasing. The Internal Developer Platform (IDP) is a solution to meet these needs. This study aims to develop an IDP that supports development activities and application needs at Trendyol. A platform has been developed that provides a unified and integrated environment, empowering developers, accelerating development cycles, and ensuring the delivery of high-quality software products. Kubernetes, KubeVela, and GitLab have been used to develop the IDP. Golang and JavaScript have been used as software development languages, and Couchbase and Elasticsearch have been used as databases. As a result, with the developed IDP, more than 3,000 applications within Trendyol have been integrated into the system. The results highlight that the entire deployment process of an application is now completely under the control of developers, eliminating the need to wait for other teams to perform manual operations. The findings of this study may endorse saving time spent on manual operational processes, allowing developers to focus directly on the development process and maximize their individual productivity.

Keywords: *Internal developer Platform, Kubernetes, Software development life cycle.*

1. Introduction

The software development is a dynamic and multifaceted process that meets user needs, involving the design of various solutions, coding, testing, and maintaining the resulting output. The software process, which has gained increasing importance with technological advancements, is undergoing dynamic change. Consequently, the user expectation is continually evolving, highlighting the need for constant innovation and adaptation in the software development.

Regular updates and optimizations play a critical role in ensuring that the software remains up to date in terms of security and performance. In particular, e-commerce platforms and software development teams are constantly adopting new strategies and approaches to increase efficiency, strengthen collaboration, and encourage innovation. However, each team creating their own standards makes it difficult to maintain consistent processes and delays updates that need to be made within a certain period of time. In addition, in cases where problems occur during application deployment or during operation, teams may need to check multiple platforms to resolve the problem. This is often time-consuming and requires a lot of effort. In this context, IDPs are gaining importance today as internal products consisting of tools, services, and information that enable software teams to deliver software autonomously and faster.

History: Received: 21 April 2025; Revised: 9 June 2025; Accepted: 12 June 2025; Published: 1 July 2025
* Correspondence: mfakay@cu.edu.tr

An IDP is a centralized ecosystem designed to organize, optimize, and monitor the software development lifecycle within an organization. An IDP, which is very important for software developers, provides a platform where they can access all the tools and services they need to build, test, and deploy applications. The platform is usually tailored to the specific needs of a company and integrates various tools, services, and workflows into a consistent and user-friendly system. In this context, these platforms offer a variety of standardized self-service tools that developers may need during the code development process, and when designed according to the needs of businesses, they provide significant gains in productivity and work speed for both developers and operational teams. IDP, offering a dynamic, flexible, and continuously evolving framework for modern software teams, play a pivotal role in accelerating business processes, reducing costs, and enhancing software quality.

In this study, it is aimed to develop a IDP that meets the development activities and application needs at Trendyol, and to encourage a culture of experimentation and innovation by providing an environment suitable for rapid prototyping and testing processes.

This study is organized as follows: Relevant literature is presented in Section 2. Section 3 explains the details of the platform. The results of the study presented in Section 4. Section 5 concludes the paper.

2. Literature Review

Saeed, et al. [1] aimed to review the recent developments in the field of security integration in the Software Development Life Cycle (SDLC) and suggest a way forward by analyzing articles published in the last twenty years and followed Kitchenham's review protocol. The study has been divided into three main phases: planning, execution and analysis. 100 articles have been selected and from these articles, the need for a collaborative approach to address critical software security risks through effective risk management/prediction techniques has been highlighted.

Khalid, et al. [2] the present study conducted a literature review on security metrics and measures in the field of secure software development. 61 research publications have been selected to extract data based on inclusion and exclusion criteria. 215 software security metrics have been identified and these metrics have been classified according to different stages of the SDLC. The aim has been to evaluate the most cited metrics in each stage of the SDLC and for this purpose, a Strengths, Weaknesses, Opportunities, and Threats (SWOT) analysis has been performed.

Srinivasan, et al. [3] aimed to reduce the time and effort required to execute the above tasks which increased software developer productivity including software development workflow automation. PlatFab, a Platform Engineering service implemented in Industrial Budgeting System, has been proposed. Custom developer portal with Continuous Integration and Continuous Delivery/Continuous Deployment (CI/CD) pipeline has been developed to automate financial workflows and facilitate collaborative development. The results showed that the compilation time has been reduced by one minute for each service and 60 MB of storage has been saved for each service.

Aslina and Nugraha [4] proposed an IDP that supports Scrum, has open source and self-hosting capabilities. They used a Compositional Agile System (CAS), an IDP based on an extensible microservices architecture. Several architectures and example scenarios focusing on Public Administration where CAS could be useful are presented.

Ciancarini, et al. [5] aimed to support the developer experience by proposing the design of an internal developer portal integrated with Artificial Intelligence (AI), using the Lean User Experience approach. This approach addressed challenges such as the steep learning curve associated with new technologies, the need for standardization, and the effective management of infrastructure. In this context, cognitive load has been reduced and usability improved through AI integration. A prototype has been developed and subsequently evaluated. The results indicated that AI enhanced learnability by effectively meeting the needs of the user base.

Aune [6] aimed to investigate the factors that ensure successful adoption of internal developer platforms in software development organizations. A qualitative approach has been adopted using the

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 9, No. 6: 2815-2826, 2025
DOI: 10.55214/25768484.v9i6.8494

stepwise deductive inductive method. The results showed that successful internal developer platforms facilitate processes with clear guidelines and comprehensive documentation. It has been also noted that the flexibility of the platforms allowed developers to adapt the platform to their needs. In addition, the study emphasized that IDPs are more effective when offered as an option than when they are mandatory, and that building a strong community around the platform is important for platform adoption.

Bayer [7] explained the basic components of an IDP. It has been emphasized that it is more important to establish a conceptual basis before organizing the services provided by the IDP.

Chandrasekaran [8] investigated methodologies for improving software effectiveness in addition to the use of IDPs. The success of IDPs in improving software effectiveness has been focused in terms of efficient use of resources, facilitation of standardized processes, and promotion of improved communication among development teams. The success of IDPs on software development cycles, quality assurance practices, and overall software performance has been analyzed. In addition, the impact of these IDPs on continuous improvement and delivery of higher quality software products has been emphasized.

Kunchenapalli [9] has been investigated and analyzed how Development and Operations (DevOps) teams can effectively set up and maintain IDPs to enhance the developer experience. The analysis included the automated provisioning of GitHub repositories, which contained code templates for the supporting infrastructure of a developer portal tailored for Java applications.

Moriconi [10] aimed to explore data-driven approaches for the automated root cause analysis of CI/CD build errors, utilizing both public and industrial datasets, with the objectives of identifying errors, locating root cause messages in build logs, and enhancing the performance and security of CI/CD systems. To this end, it has been presented the ChangeMyMind, a novel method based on Recurrent Neural Networks (RNNs), designed to accurately detect root cause messages in build logs. The integration of Natural Language Processing (NLP) and Knowledge Graph Embedding (KGE) techniques enabled classification of build failured with an accuracy of 94%. Furthermore, it has been proposed X-Ray- Transport Layer Security (TLS), a general and transparent approach for inspecting TLS-encrypted network traffic within CI/CD environments. And finally, it has been emphasized vulnerabilities in CI/CD systems, drawing attention to the potential risks posed by undetected, long-term security threats.

Shropshire and van Devender [11] examined the risk management practices of organizations that underwent significant changes in their software development and delivery architectures following the adoption of internal developer platforms, aiming to enhance productivity and the developer experience. A semi-quantitative risk analysis has been conducted on both the legacy and the newly adopted architectures. These architectures have been compared across 74 specific threat vectors. The findings indicated that, while the internal developer platform offered the security team a centralized control point for managing network and authentication parameters, it also expanded the software attack surface and introduced a high-value target for potential attackers.

Ali [12] provided a comprehensive review of the fundamental principles underlying DevOps and CI/CD. The study emphasized the collaborative advantages of integrating DevOps and CI/CD, while also addressing the challenges associated with their implementation. Furthermore, the role of CI/CD oriented practices on software effectiveness has been analyzed. Additionally, practices with high success rate and potential of future trends in DevOps oriented software engineering have been discussed.

Gokarna [13] aimed to include the stages in the software development lifecycle in the DevOps implementation strategy. An extensive compilation of industry-leading solutions for automating and integrating various phases of software delivery has been presented. The study also examined how DevOps implementation strategies can be incorporated into different stages of the software development lifecycle, as well as the technologies that can be used to achieve these integrations.

Gomes [14] examined the foundational concepts necessary to understand and develop an IDP. Also, solutions for IDP components, IDP implementation approaches and tools for them have been

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 9, No. 6: 2815-2826, 2025
DOI: 10.55214/25768484.v9i6.8494

analyzed. Additionally, a case study has been conducted to establish a suitable implementation standard for DevScope. Within this study, two tools have been developed: the first compiles and distributes application definition files, while the second, the Canaveral Command Line Interface (CLI) tool, generates definition files that comply with the selected standard. The evaluation of the Canaveral CLI tool's usability revealed both areas for improvement and positive findings.

Jahıć and Buzađıja [15] explained the primary responsibilities of DevOps engineers in the context of IDP. Key concepts in the field and fundamental approaches to software design, implementation, and deployment have been outlined. The concepts of cloud infrastructure, virtual machines, containers, infrastructure automation, and process flows have been handled and the development of IDP has been explained.

Jani [16] presented a comprehensive assessment of the working principles, benefits, and technical strategies of CI/CD, supported by real-world examples and case studies. The study evaluated the tools and technologies that make CI/CD possible, identifies the challenges that may be encountered during implementation, and evaluates practices that may provide alternative solutions. Additionally, future trends and the effective role of CI/CD in modern software development have been examined.

van de Kamp, et al. [17] aimed to enhance the effectiveness of platform engineering and accelerate its adoption in enterprises, addressing the rising popularity of platform engineering and the lack of consensus in the field. The study highlighted the productivity and improvement of developer experience offered by engineering platforms like IDP and introduced the Platform Engineering Reference Model, based on the Reference Model for Open Distributed Processing. This model provided software companies with a framework for developing customized platform engineering strategies.

Soares, et al. [18] aimed to investigate and interpret empirical evidence on the impact of the concept of Continuous Integration (CI) on software development. To achieve this, a literature search has been conducted from six digital libraries. A total of 479 studies have been reviewed, from which 101 empirical studies have been selected. Data have been extracted regarding the CI environment, the reported effects of CI, and the methodologies employed in the selected studies. A Thematic Synthesis (TS) has been applied to categorize and summarize the findings. The results indicated that the studies investigated both the positive effects of CI—such as improved collaboration—and the negative effects—such as increased technical and process-related challenges. Six distinct themes emerged from the TS: development activities, software processes, quality assurance, integration patterns, issues and defects, and build patterns. Furthermore, it has been observed that empirical research on CI has significantly increased in recent years.

Leite, et al. [19] focused on how software organizations form development and infrastructure teams and how to effectively distribute work between these groups. The research, which included 37 interviews, resulted in four main organizational structures: silo departments, classic DevOps, cross-functional teams, and platform teams. In addition, the transitions that companies made between these structures over time have been observed.

Leite, et al. [20] investigated how businesses aiming for continuous delivery can properly organize their development and operations teams. In this context, Grounded Theory has been used and data has been collected from 27 Information Technology (IT) professionals. Extensive analysis has been performed and a taxonomy has been created with four organizational structure patterns: siloed departments, classic DevOps, cross-functional teams and platform teams. Among all of them, platform teams have been found to be the most effective, as they increased delivery success by alleviating product teams from handling non-functional issues.

Chen [21] proposed the microservices approach to address the architectural challenges. It also noted the new challenges associated with the increasing number of services, changing contracts between services, technology diversity, and testing, and shared the strategies that can be used to address these challenges.

Hasselbring and Steinacker [22] evaluated how microservices architectures facilitate concepts such as scalability, agility, and reliability. Vertical decomposition and appropriate granularity of

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 9, No. 6: 2815-2826, 2025
DOI: 10.55214/25768484.v9i6.8494

microservices for self-contained systems are explained. Experimental results have shown that with agility, more than 500 live deployments per week can be achieved, and automatic quality assurance is provided with continuous integration and deployment. In this context, it has been observed that high reliability is achieved with these concepts.

Shahin, et al. [23] aimed to review the current state of continuous practices in software development, with the objectives of classifying existing approaches and tools, identifying associated challenges and practices, and revealing gaps for future research. To achieve this, a literature review has been conducted, focusing on peer-reviewed articles related to continuous practices published between 2004 and June 1, 2016. A total of 69 articles have been selected based on predefined inclusion criteria. Thematic analysis has been employed to analyze the collected data. The review evaluated 30 different approaches and related tools that support the implementation of continuous practices. These approaches addressed various aspects such as reducing build and test time in CI, increasing visibility and awareness of build and test results, supporting (semi-)automated continuous testing, detecting violations, defects and bugs in CI, managing security and scalability concerns within deployment pipelines and improving reliability and dependability of deployment processes. In addition, the study identified some critical factors that should be given special attention when introducing continuous practices in organizations. Testing effort and duration, team awareness and transparency, adherence to good design principles, customer involvement, presence of a highly skilled and motivated team, application domain and availability of appropriate infrastructure have been among these highlighted critical factors.

Shahin, et al. [24] investigated how development (Dev) and operations (Ops) teams adopt continuous delivery (CD) practices and the potential impact of these practices on cross-team collaboration and team member accountability. They conducted 21 in-depth, semi-structured interviews and a survey of 93 software practitioners across 19 different organizations. It has been observed that Dev and Ops teams have been divided into 4 groups to effectively initiate and adopt CD practices; separate Dev and Ops teams with middle facilitator; small Ops team with more responsibility for the Dev team; no visible Ops team.

Misra and Singh [25] aimed to analyze the most effective practices for the SDLC approach. In addition, the Open Agile Software Development Life Cycle conceptual model has been proposed. This approach is derived from agile methodologies and open source software. The developed model addressed limitations of existing SDLC approaches. It has been tested to ensure cost efficiency and effort optimization, with survey-based research validating its effectiveness. Findings demonstrated that the conceptual model significantly optimized implementation efforts.

3. Details of the Platform

Kubernetes, Kubevela, and GitLab have been used to develop the IDP. Golang and JavaScript have been chosen as the software development languages, while Couchbase and Elasticsearch have been used as databases. During the platform's development process, integration points with GitLab and other existing tools have been identified and designed. The system architecture is presented in Figure 1.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 9, No. 6: 2815-2826, 2025
DOI: 10.55214/25768484.v9i6.8494

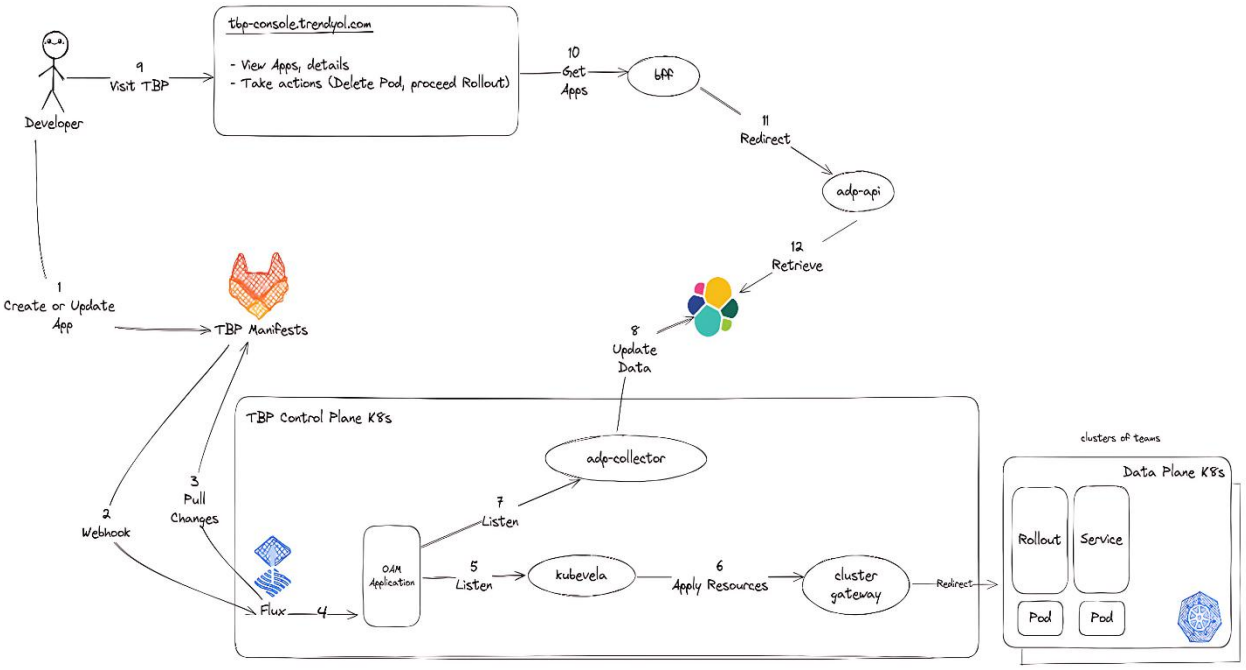

Figure 1.
System architecture.

The developer first creates or updates an application via the Trendyol Build Platform (TBP). This process is forwarded to the TBP manifests component. In the control plane phase, the TBP manifests component triggers changes via the webhook mechanism. These changes are received and managed by Flux and monitored by the application running within TBP. Kubevela applies these changes to the relevant resources and directs them to the appropriate clusters via the cluster gateway. As part of the data update process, the adp-collector component continuously monitors changes within TBP and updates the data accordingly. The updated data is stored and made accessible via ElasticSearch. The developer can access their applications and related details through the top-console.trendyol.com interface. Various operations can be performed through this interface, such as deleting a Pod or initiating the Rollout process. Requests made by the developer via TBP are directed to adp-api through the bff component, where adp-api processes the necessary data and presents it to the developer. In the final stage, the updated applications are deployed to the Data Plane Kubernetes cluster. Through the Rollout mechanism, services and Pods are executed within this data plane, completing the deployment process.

On the homepage, all applications, team metrics, alarms, and recent activities belonging to the user's team are presented at a glance. A screenshot of this screen is shown in Figure 2.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 9, No. 6: 2815-2826, 2025
DOI: 10.55214/25768484.v9i6.8494

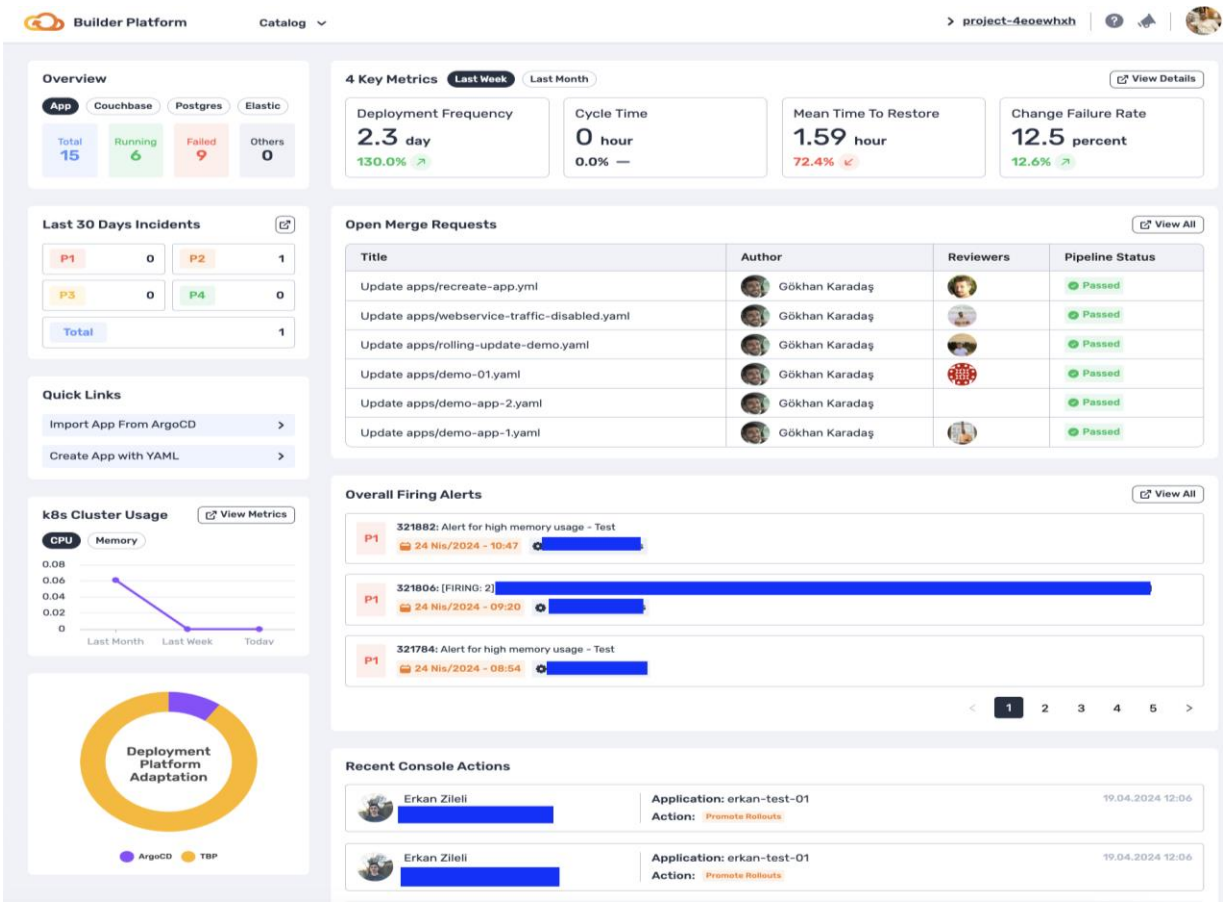

Figure 2.
Home page.

The application detail page is shown in Figure 3. The application detail page serves as the main control screen, where the application status, the latest operations, fast Scale-Up and Rollback actions, as well as all required data and actions, are consolidated. In the workflow status section, past versions can be viewed and rolled back if necessary.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 9, No. 6: 2815-2826, 2025
DOI: 10.55214/25768484.v9i6.8494

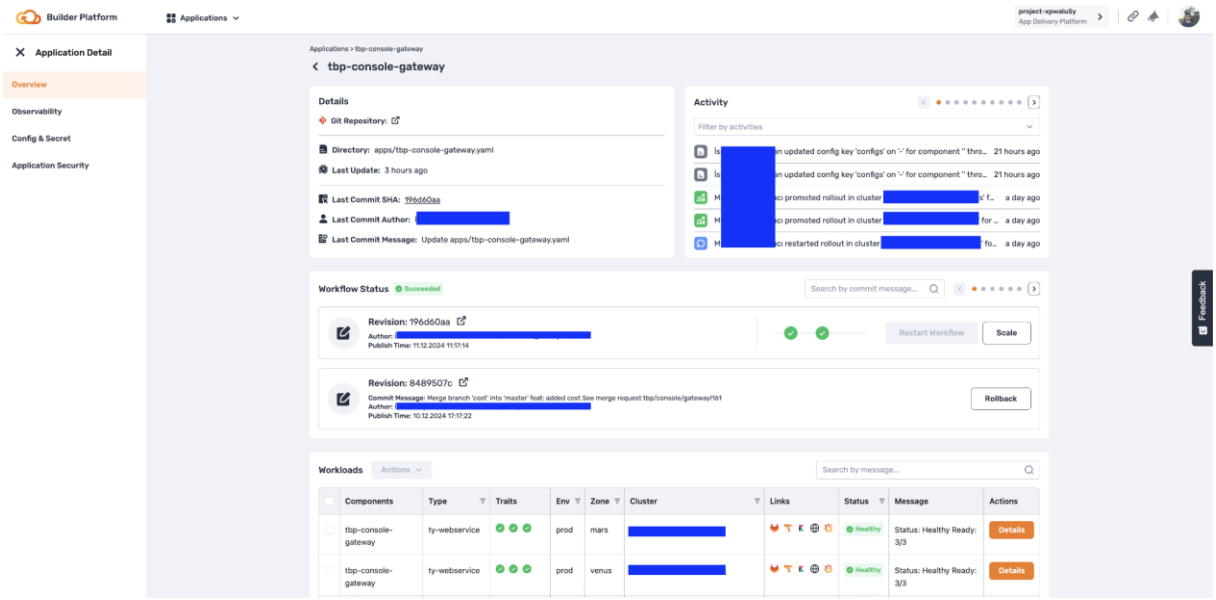

Figure 3.
Application detail.

The workload detail page is shown in Figure 4. The application management page provides the ability to view the application's replicas, as well as the actions, metrics, logs, and traces that can be performed for each replica. Additionally, the Kubernetes manifests created for the application can be examined from this screen.

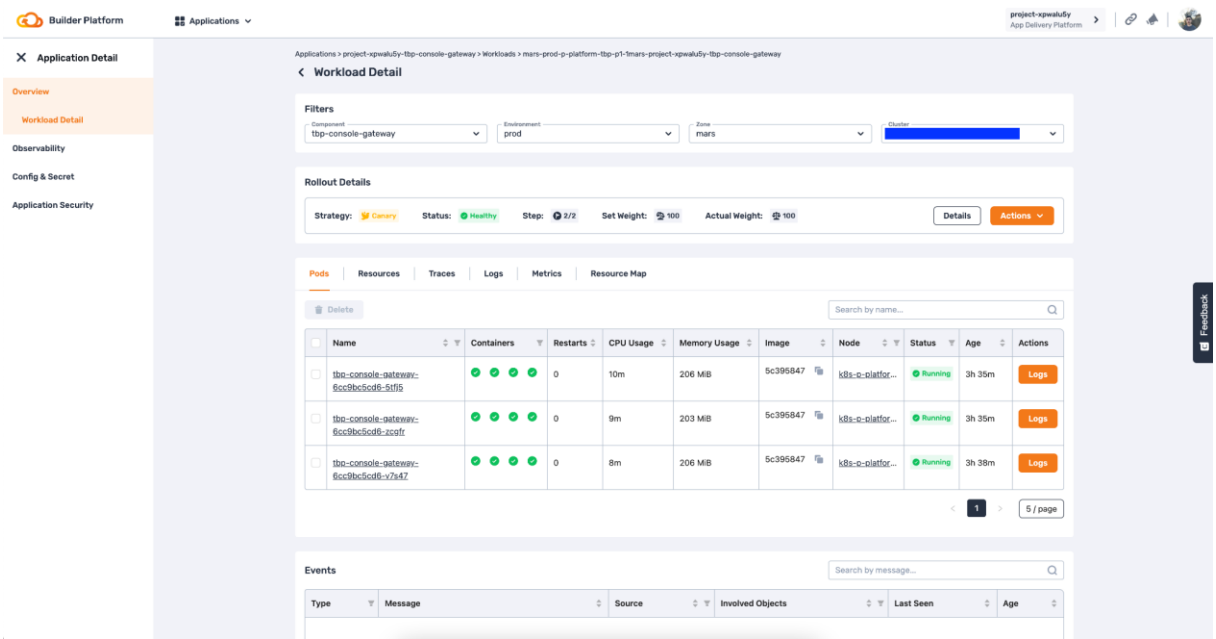

Figure 4.
Workload detail.

The resources created for deploying the application are presented in the screen shown in Figure 5.

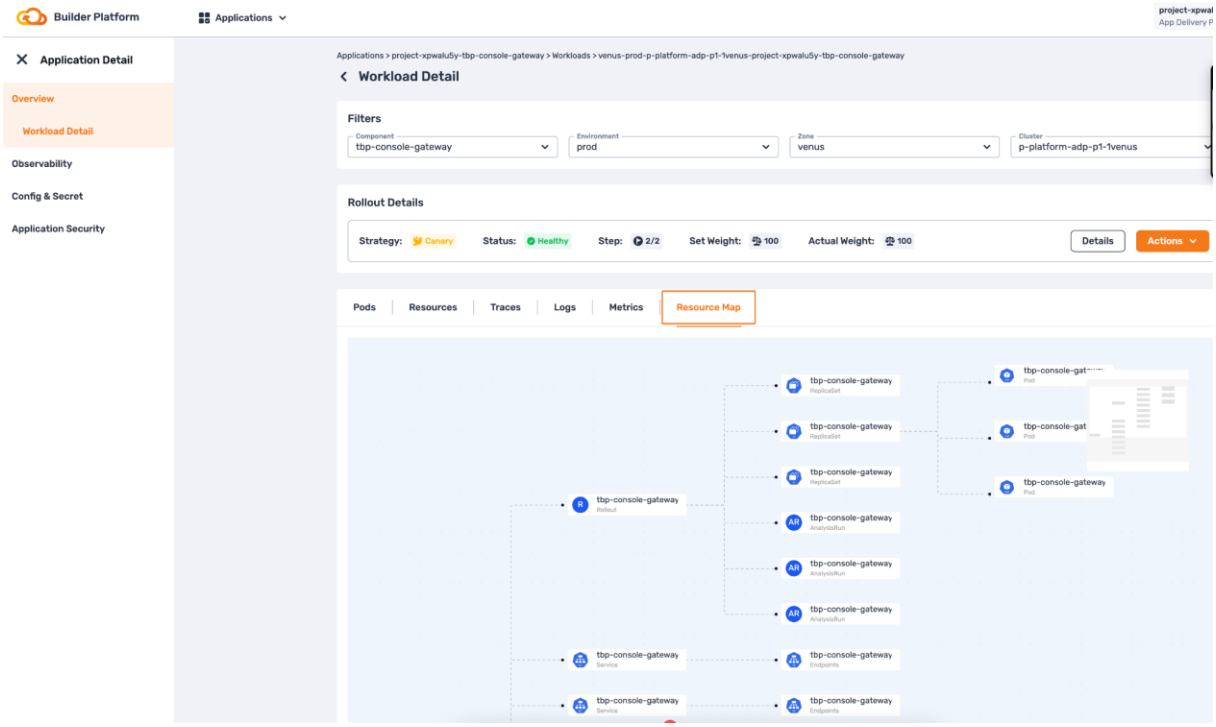

Figure 5.
Resource map screen.

The applications metrics screen is shown in Figure 6. This screen, designed to display the application's metrics, also presents the most viewed metrics of the applications in addition to the 4 Golden Signals data. The main purpose of this screen is to provide developers with all the information they need about the operational metrics of their applications.

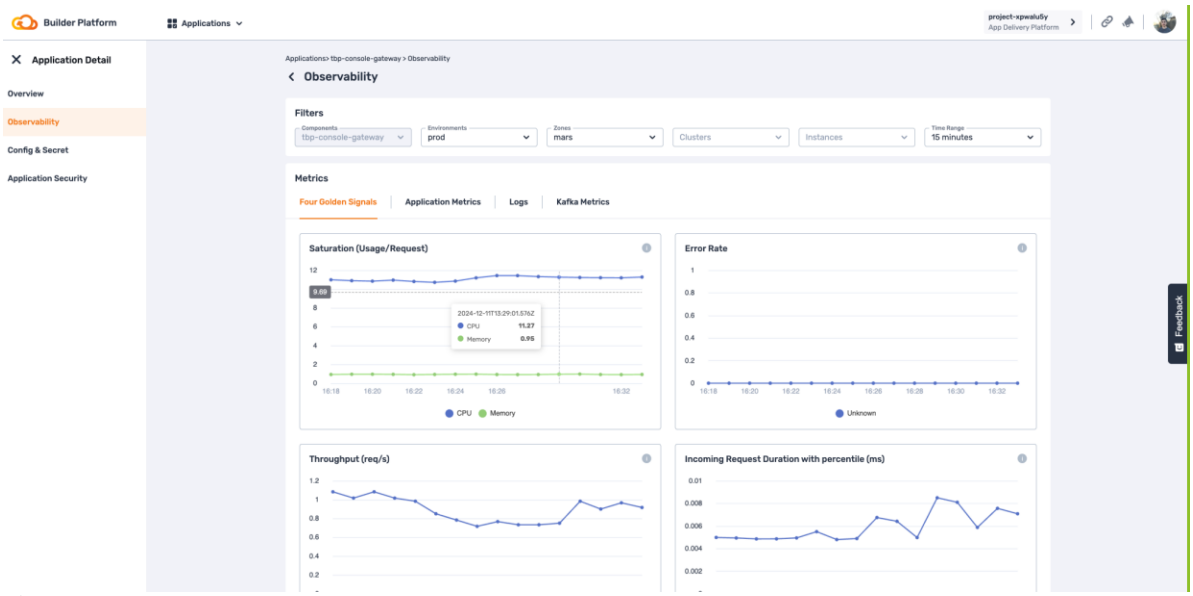

Figure 6.
The application metric screen.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 9, No. 6: 2815-2826, 2025
DOI: 10.55214/25768484.v9i6.8494

The config and secret screen is shown in Figure 7. The screen used for managing application configurations and sensitive password information allows developers to make different configuration settings for each environment in which their applications are deployed.

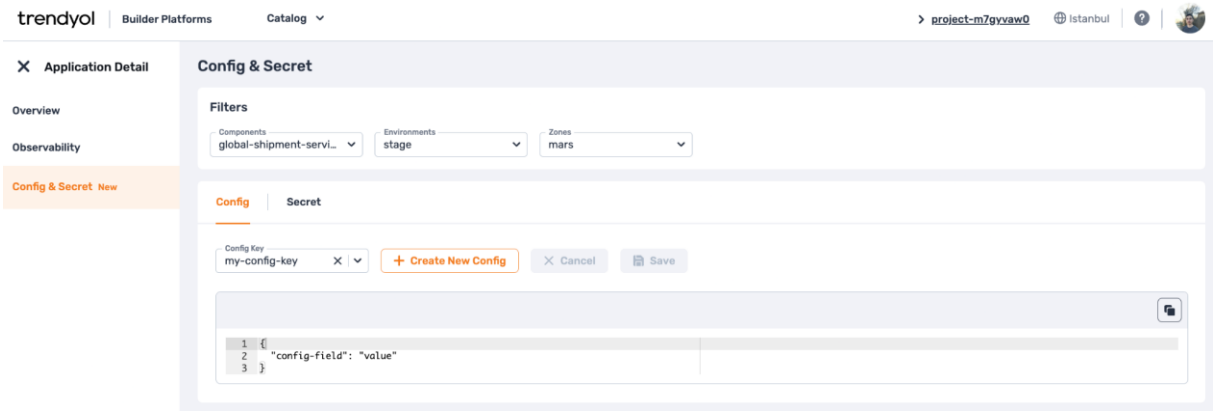

Figure 7.
Config and secret new.

4. Results of the Study

With the platform,

- Teams have been able to focus on standardization through design, infrastructure, Service Level Objectives, and workflow optimization. Repetitive tasks, such as preparing resources or environments for application developers, have been automated through the IDP.
- It has been provided that the entire deployment process of an application is now fully under the control of developers, and eliminating the need to wait for other teams to perform manual operations.
- The cognitive load of developers has been reduced.
- At least 30% of the applications within Trendyol (approximately 6.500) have been deployed and managed through IDP.
- The cycle time of development processes has been shortened.
- A significant increase in daily deployment frequency has been achieved.
- The application deployment time has been reduced from 7 minutes to 2 seconds.
- More than 3.000 applications have been integrated into the system.

5. Conclusion

The ever-evolving dynamics of software development are driving organizations to seek new approaches to enhance the productivity, collaboration, and innovation capacity of their software development and technology teams. In this context, the concept of the IDP has emerged as a powerful and innovative solution to address these needs. An IDP is a centralized ecosystem designed to organize, optimize, and monitor the software development lifecycle within an organization. These platforms include a range of standardized self-service tools that developers may require during the code development process. When tailored to the unique requirements of an organization, they significantly enhance the productivity and efficiency of both developers and operational teams. In this study, an IDP has been developed. This platform provided a unified and integrated environment that empowered developers, accelerated software development cycles, and facilitated the delivery of high-quality software products. Additionally, the IDP fostered a culture of experimentation and innovation within the organization by offering an infrastructure well-suited for rapid prototyping and testing processes.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 9, No. 6: 2815-2826, 2025
DOI: 10.55214/25768484.v9i6.8494

Transparency:

The authors confirm that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.

References

[1] H. Saeed, I. Shafi, J. Ahmad, A.-A. Khan, T. Khurshaid, and I. Ashraf, "Review of techniques for integrating security in software development lifecycle," *Computers, Materials \& Continua*, vol. 82, no. 1, pp. 139-172, 2025. https://doi.org/10.32604/cmc.2024.057587

[2] A. Khalid, M. Raza, P. Afsar, Rafiq A. Khan, Muhammad I. Mohmand, and Hanif U. Rahman, "A SWOT analysis of software development life cycle security metrics," *Journal of Software: Evolution and Process*, vol. 37, no. 1, p. e2744, 2025. https://doi.org/10.1002/smr.2744

[3] V. Srinivasan, M. Rajkumar, S. Santhanam, and A. Garg, "PlatFab: A platform engineering approach to improve developer productivity," *Journal of Information Systems Engineering and Business Intelligence*, vol. 11, no. 1, pp. 79-90, 2025. https://doi.org/10.20473/jisebi.11.1.79-90

[4] Y. R. Aslina and I. G. B. B. Nugraha, "Exploring potential AI use cases in ınternal developer portals: A path to enhanced developer experience," presented at the 2024 IEEE International Conference on Data and Software Engineering (ICoDSE), pp. 143-148, IEEE, 2024, 2024.

[5] P. Ciancarini, R. Giancarlo, G. Grimaudo, M. Missiroli, and T. Cheng Xia, "The design and realization of a self-hosted and open-source agile internal development platform," *IEEE Access*, vol. 13, pp. 79516-79533, 2025. https://doi.org/10.1109/ACCESS.2025.3564141

[6] A. A. W. Aune, "Towards enhanced developer experience: An empirical study on successful adoption of ınternal developer platforms," Master's Thesis, NTNU, 2024.

[7] F. Bayer, *How metamodeling concepts improve internal developer platforms and cloud platforms to foster business agility. In Metamodeling: Applications and Trajectories to the Future: Essays in Honor of Dimitris Karagiannis*. Cham: Springer Nature Switzerland, 2024.

[8] S. Chandrasekaran, "Optimizing software quality through internal developer portals," *International Journal of Science and Research*, vol. 13, no. 1, pp. 696–699, 2024.

[9] V. Kunchenapalli, "Good developer experience with platform engineering and DevOps," *International Journal for Research in Applied Science and Engineering Technology*, vol. 12, no. 3, pp. 2240–2244, 2024.

[10] F. Moriconi, "Improving software development life cycle using data-driven approaches," Doctoral Dissertation, Sorbonne University, 2024.

[11] J. Shropshire and M. S. van Devender, "Analyzing risks to internal developer platforms," 2024.

[12] J. M. Ali, "DevOps and continuous integration/continuous deployment (CI/CD) automation," *Advances in Engineering Innovation*, vol. 4, pp. 38-42, 2023. https://doi.org/10.54254/2977-3903/4/2023031

[13] M. Gokarna, "DevOps phases across software development lifecycle," *Authorea Preprints*, 2023.

[14] A. Gomes, "Deploy-oriented specification of cloud native applications," Master's Thesis, Universidade do Porto (Portugal), 2023.

[15] A. Jahić and N. Buzađija, "DevOps methodology in modern software development," *Quantum Journal of Engineering, Science and Technology*, vol. 4, no. 1, pp. 1-11, 2023.

[16] Y. Jani, "Implementing continuous integration and continuous deployment (CI/CD) in modern software development," *International Journal of Science and Research*, vol. 12, no. 6, pp. 2984–2987, 2023.

[17] R. van de Kamp, K. Bakker, and Z. Zhao, "Paving the path towards platform engineering using a comprehensive reference model," in *International Conference on Enterprise Design, Operations, and Computing (pp. 177-193). Cham: Springer Nature Switzerland*, 2023.

[18] E. Soares, G. Sizilio, J. Santos, D. A. da Costa, and U. Kulesza, "The effects of continuous integration on software development: A systematic literature review," *Empirical Software Engineering*, vol. 27, no. 3, p. 78, 2022. https://doi.org/10.1007/s10664-021-10114-1

[19] L. Leite, G. Pinto, F. Kon, and P. Meirelles, "The organization of software teams in the quest for continuous delivery: A grounded theory approach," *Information and Software Technology*, vol. 139, p. 106672, 2021. https://doi.org/10.1016/j.infsof.2021.106672

[20] L. Leite, F. Kon, G. Pinto, and P. Meirelles, "Platform teams: An organizational structure for continuous delivery," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 505-511.

[21] L. Chen, "Microservices: architecting for continuous delivery and DevOps," in *2018 IEEE International Conference on Software Architecture (ICSA)*, 2018: IEEE, pp. 39-397.

[22] W. Hasselbring and G. Steinacker, "Microservice architectures for scalability, agility and reliability in e-commerce," in *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, 2017: IEEE, pp. 243-246.

[23] 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. https://doi.org/10.1109/ACCESS.2017.2685629

[24] M. Shahin, M. Zahedi, M. A. Babar, and L. Zhu, "Adopting continuous delivery and deployment: Impacts on team structures, collaboration and responsibilities," in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, 2017, pp. 384-393.

[25] S. C. Misra and V. Singh, "Conceptualizing open agile software development life cycle (OASDLC) model," *International Journal of Quality & Reliability Management*, vol. 32, no. 3, pp. 214-235, 2015. https://doi.org/10.1108/IJQRM-08-2013-0127

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 9, No. 6: 2815-2826, 2025
DOI: 10.55214/25768484.v9i6.8494