Software application to manage the most frequent CI/CD issues in a DevOps environment

Ivar Ekeland[1], Roger Temam[2] Jeffrey Dean, David Grove, Craig Chambers, Kim B. Bruce, and Elsa Bertino

[1] Princeton University, Princeton NJ 08544, USA,
I.Ekeland@princeton.edu,
WWW home page: http://users/~iekeland/web/welcome.html
[2] Université de Paris-Sud, Laboratoire d'Analyse Numérique, Bâtiment 425,
F-91405 Orsay Cedex, France

Abstract. Currently, some software development companies work with the integration of tools that use the Continuous Quality concept, which are synchronized with DevOps to verify and monitor at each step the non-existence of any quality violation. However, when implementing CI/CD tools, many unplanned incidents and time expenditure usually occur when correcting these types of errors, since there is no central place where the solutions to these errors can be consulted. The present work proposes the development of a tool that can categorize and store the most frequent errors, to later create libraries that help quality, and these also help reduce the factors already mentioned that are lost with the lack of unforeseen quality. To develop the proposal, the prototype methodology is used, which revolves around continuous iterations in order to achieve the desired product.

Keywords: DevOps, Continuous Quality, software development, incident report

1 Introduction

When building any type of software, the use of methodologies and tools is essential to keep track of time, money and effort. When developing software with agile methodologies, companies have more rigorous control of the products they develop, since this has proven to save time in development and improve communication between clients and developers, unlike traditional methodologies. Since in these cases, if an error is made at one stage of the software construction, it will not be resolved until reaching advanced stages and therefore can delay the software construction time; these errors, also known as incidents, can affect long-term development. Based on agile methodologies, and due to the lack of co-operation and the problems that arise between the operations and development teams for the projects, the Development and Operations discipline (DevOps) was born, which is a culture based on the conjunction of the two departments to have better communication, reduce downtime, streamline processes and also release software products quickly and frequently to clients [27].

On the other hand, Continuous Integration (CI) is a phase of DevOps, which controls the source code that is integrated frequently and is automatically verified to enter clean source code into the existing one, this also helping to deliver a faster and more reliable software product [24]. In this same sense, there is Continuous Delivery (CD), which is another phase of DevOps, which focuses on the operations team making deliveries iteratively in a short cycle to guarantee that the software is always in a state release without affecting the changes that the development team continues to make. Implementing CD allows the development and operations team to continue working on the iteration in parallel [6], [16].

Studies carried out, to understand the behavior when implementing DevOps practices [27], report that the most discussed problems among their respondents are the lack of culture in the team, this being the most prominent; the control and knowledge of the tools and hardware, the control of resources, and finally, the non-existence of a clean process to implement DevOps or a specific guide. Therefore, it can be deduced that the implementation of DevOps in a company or organization is not an easy task, since many internal changes have to be made, such as technological changes, the structure and process that is currently in place. When implementing DevOps, it is observed that the majority of respondents agree that this discipline stands out mainly for its most important advantages [27], which are: work teams have more communication and quality when a project is carried out, quality of software increases and the development cost and time are decreased. The advantages most highlighted by respondents imply that there are more features implemented and more frequent releases [17]. It is also mentioned that DevOps helps in automation, thereby reducing the effort required for manual configurations, helping companies and organizations to release software as frequently as possible.

When managing quality in DevOps, there is something known as Continuous Quality (CQ), by which we refer to the CI/CD quality control approach; thus putting quality on par with CI/CD, helping to solve problems earlier, therefore it will help prevent technical debt from increasing [1]. By improving software quality, companies lose less development time and less money for each project they develop. Quality is one of the most relevant factors, therefore, if it is not treated properly, it can affect companies in each of their projects. To solve the problem of the lack of quality that is not yet taken into account, it is proposed to collect the most frequent errors in CI/CD with a software tool to later store and consult these when necessary. This will verify quality and constantly review what was previously not taken into account for quality. By creating this software, the aim is to increase quality in companies that use CI/CD in a DevOps environment and thereby reduce rework, reduction of time when finding incidents, the cost of their projects and effort, in addition to helping developers here.

There are some ways to achieve good quality in CI/CD, one of them is how it is managed by ”Quality Clouds”, which is a quality control software manager. Continuous Quality (CQ) is integrated with tools and with the support of a ”Quality Manager” who stores and verifies violations of best practices at each stage. These can be selected either from a pre-existing library or customized to your individual needs. At the time of developing the software, each step is being monitored to provide a real-time response to the changes that are being made, thus detecting early errors so that they are corrected as soon as possible before moving to any other stage of the development process; whether they are newly coded or legacy. These incremental fixes will strengthen the quality of the software. When performing quality checks, all phases belonging to CI/CD use various tools during the development, build, and release phases to perform quality checks. There are many tools that integrate well into this process, including Jenkins, Bamboo, BitBucket, GitLab, Copado, and CircleCI [1].

In this sense, a member of the Quality Clouds team mentions that one of their clients, with a mature DevOps configuration, began to implement a continuous quality approach. In the first 6 months, they have already seen a 15% increase in their development speed. With less technical debt accumulated, early indications show that this will have an even greater impact on platform development in the future [1]. When developing software, companies and organizations that implement agile methodologies and work with an environment such as DevOps within them considerably speed up the construction work of their projects through automation tools; therefore, the development cost is lower. Currently, some companies are working with the integration of tools that use the Continuous Quality concept, which are synchronized with DevOps to verify and monitor at each step the non-existence of any quality violation. However, when implementing CI/CD tools, many unplanned incidents and time expenditure usually occur when correcting these types of errors, since there is no central place where the solutions to these errors can be consulted, and it is spent a lot of time searching on different internet pages.

To detect incidents and their possible solution, there are several software tools that help visualize and monitor services and servers, such as: Grafana, New Relic, DataDog, Sentry, among others. Likewise, there are forums, websites and communities, as well as StackOverFlow, where incidents and their possible solutions are discussed and shared. As far as we know, there is no software for managing and storing incidents with their possible solutions, where all incidents are grouped in one place and it is easy to perform a search or filter. That is why the purpose of this work is to propose the development of a software application to store, consult and resolve the most frequent incidents in CI/CD in a DevOps environment in order to implement an improvement in the development of software systems. based on CQ.

The rest of the document is organized as follows, section 2 includes a descriptive review of related approaches, while section 3 includes the proposed application developed. While section 4 shows the results obtained. Finally, section 5 comments on final considerations for conclusions and future work.

2 Background

The concept of DevOps arose with the need to unite the development and operations departments (see Fig. 1), in order to have better communication and better performance in software products. DevOps emphasizes continuous learning and improvement by passing feedback from production to development and improving cycle time [10]. DevOps involves aspects of culture, automation, measurement and sharing [20], [16].

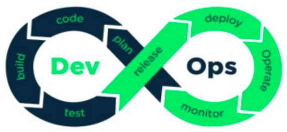


Fig. 1: DevOps methodology workflow

2.1 Relevance of quality in software

When talking about quality in software, it is normal to mention about standards and norms that help improve it. One of the most common and well-known standards is ISO 9001, which focuses on all the quality management elements that an organization must have in place to have an effective system to manage and improve its products and services generally or constantly. The aforementioned standard helps companies to have a certification where it is demonstrated that they formally have quality in their software products, these being verified and keeping track of them in the life cycles of each software; requirements analysis, software design, software development, testing and validation, and finally, testing and maintenance [15].

2.2 Continuous quality

Continuous quality (CQ) is a systematic approach to finding and fixing software defects during all phases of the software development cycle. CQ reduces the risk of security vulnerabilities and software defects (bugs) by helping developers find and fix problems as early as possible in the development cycle [8]. Along with new agile methodologies in software development, DevOps and CI/CD are used to ensure that quality is constant and can operate at an adequate pace to keep up with the demands of software consumers. There are several tools for software quality control, one of them called ConQAT (Continuous Quality Assessment Toolkit), which can be used to monitor several projects at the same time, it can analyze various artifacts such as the source code of programming languages (Java, C, C++, PL/1, Cobol and VB) [7].

2.3 Software quality models

In the software field, there are different models to allow a process of continuous improvement with the implementations that are carried out. Each of the models has characteristics and a structure. There are companies that implement these models in order to certify and guarantee their products and processes. It is mentioned [19] that quality models are those documents that integrate most of the best practices, they also propose management topics that each organization must emphasize, and integrate different practices aimed at key processes and allow progress in quality to be measured. The structure of software quality models is generally made up of criteria that are evaluated metrically and have various quality factors, thus obtaining an evaluation of the software from a general view to something very specific about it, thus allowing a clear observation of the affecting factors, both quantitatively and qualitatively.

Quality and models at the process level. When software is developed, quality must be present at all times, being programmed from the beginning of the project. Subsequently, control and supervision of each phase of the product life cycle must continue so that quality is inherent to the product, thus mitigating risks and offering continuous monitoring. If certain factors and rules are followed, the level can be optimal to meet quality factors. Likewise, if some of these factors are left in the life stages of the software, it may not materialize or cause failures in the fully developed product [5].

In this sense, we have Bootstrap, which focuses on improving processes through six key activities that are implemented: first, needs are reviewed to identify areas for improvement; then, the improvement process begins with concrete actions. The evaluation is then prepared and conducted to collect relevant data, which is analyzed in detail to understand the results obtained. Subsequently, the identified improvements are implemented and the process is concluded ensuring that the improvement objectives have been achieved [9], [5]. One can also talk about Personal Software Process (PSP), which is designed to promote the professional development of the engineer, focusing on effective quality management in development projects, seeking to reduce defects in products through accurate estimation and efficient planning of the work [25]. In conjunction with PSP is the Team Software Process (TSP), designed for self-directed software development teams, focusing on creating products with minimal defects within estimated times and costs. This phase, which follows the PSP, includes detailed plans and processes such as personal reviews, inspections, quality performance indices, and promotes team integration [14]. We can also mention ISO/IEC 15504, which promotes helping organizations achieve maturity, which implies having well-defined processes and responsibilities, the ability to foresee results, deliver quality products within agreed deadlines, increase productivity and achieve satisfaction of both customers and employees

Quality and models at the product level. The classification of software quality models is based on the evaluation approach, whether at the process, product or quality of use level. The product quality model focuses on specifying and evaluating compliance with product criteria as requested, applying internal and, if necessary, external measures [4]. In this context, norms and standards have defined product quality in three types: internal, external and in use [5], [18]. The main objective of this approach is to verify that the features defined by the client during the analysis are completely fulfilled and satisfy their requirements.

Under this approach, we can mention FURPS, which implies a model for improving evaluation [23], which is based on five fundamental criteria that define its name: functionality, usability, reliability, performance and supportability. You can also talk about Boehm [26], which is a model that is responsible for organizing work into task regions, which are subdivided into sets defined by iterations. The team decides the number of iterations, and each one is structured in four different phases: planning, risk analysis, engineering and evaluation [5]. Relating it in this same area, McCall focuses on the evaluation of software quality, which structures his work in three defined stages: factors, criteria and metrics. Each of these stages plays a crucial role in ensuring that the software meets established quality standards [11]. Furthermore, ISO 9126 is destined for the same objective, which is aimed at developers, quality assurers, testers, analysts and all those involved in the software construction process. This model is made up of four essential parts: the quality model, external metrics, internal metrics and the quality of the metrics in use. Each of these sections focuses on fundamental elements grouped into six key characteristics: functionality, reliability, usability, efficiency, maintainability and portability, in addition to their associated subcharacteristics [3].

On the other hand, GILB is a quality model that guides the evaluation of software based on four main attributes: workability, adaptability, availability and usability. These attributes and their sub-attributes serve to support software project management, providing clear guidance for troubleshooting and detecting risks. By addressing these critical aspects, the model helps ensure that the software meets the necessary quality standards and meets the expectations of users and stakeholders [11]. Finally, ISO 25000 carries out structured control of development, ensuring that requirements are met and quality attributes are evaluated systematically. This model focuses on the following quality attributes: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability [2].

2.4 Quality control in CI/CD using DevOps

Quality is one of the factors that affect software development processes in their different stages. DevOps is one of the proposed solutions to such pressure. The primary focus of DevOps is to increase the speed, frequency, and quality of deployment. An organization can implement automation in several ways. DevOps automation is supported by different designs. Some of which include using the cloud for large data storage, using cloud-based email and logging services, using a real-time monitoring tool. The use of SaaS (Software as a Service) and IaaS (Infrastructure as a Service) is capable of supporting DevOps automation [13].

In the field of DevOps culture, companies and organizations often use tools for automation, thus saving time and money on possible errors made if these jobs were carried out manually. Quality management is present at every stage of this culture, where SonarQube is used for quality control. SonarQube is a static code analysis platform, which analyzes both the quality and possible vulnerabilities that may exist in our developments. As with Jenkins, Sonar is the tool that is consulted on a daily basis, so it is interesting to know it in more depth [22].

Likewise, there are monitoring applications for quality control in DevOps, one of them is Omnia, which helps with automation and speed, where it alerts users and their movements to ensure that everything is in the correct state. [12]. Test automation is related to the continuous delivery process and therefore helps ensure the reliability, maintainability, and security attributes of software quality.

3 Proposed scheme

For the development of the proposal, the prototype methodology was used (see Fig. 2), which is a system development method in which a prototype is built, tested and then reworked as necessary until an acceptable result is obtained from which the complete system can be developed [21], [**?**]. The phases of this methodology include: i) definition of objectives, ii) definition of functionality, iii) design and development of the prototype, and iv) evaluation of the prototype. Within this context, a prototype is a sample implementation, which is limited only to the main functionalities of the system. It works on a trial and error basis (rotating in stages) between developers and users, so that the latter test the functionalities before they are fully implemented.

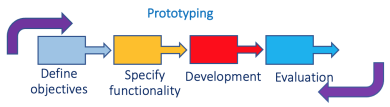


Fig. 2: Prototyping methodology phases

3.1 Prototype objectives

The main objective is to design, develop and implement a software system where it will be possible to collect the most frequent issues committed by developers in CI/CD in a DevOps environment. Later they can be given a solution, and they will be visible to other developers who can add comments and contributions, being able to have a graphic visualization of the stages and tools that tend to have the most incidents. The objectives of the prototype are the following:

a) The system must be entered using a username and password.
b) The software system can be accessed from anywhere.
c) It should be able to allow recording, modifying, deleting and displaying errors made in development.
d) The system will allow comments to be added to reported incidents/issues.
e) The system will show graphs with the reported incidents.
f) Reports of registered errors can be generated.

3.2 Prototype functionality

For the development of this system, the expected functionalities to be implemented are the following: i) manage errors: collect, modify, delete and display error documentation; ii) generate error reports. Administrator will be able to carry out all the developer's activities, and he will also be able to eliminate or modify any incident in the system. Meanwhile developer user will be able to capture new issues and be able to modify them. This user can also add comments to incidents and generate reports. An then, guest user will be able to see all the registered issues and their solutions. Likewise, this user can see the graphs and generate incident reports.

3.3 Prototype development

Architecture. The following technologies were selected for the system development: i) Python (back-end), ii) Django (back and front), iii) Mysql (database), Bootstrap (front-end). Fig. 3 diagrams the architecture of the created system which uses various GCP services and features to provide a scalable and secure application. Using a load balancer ensures that the workload is distributed evenly, while firewall and IAM authentication policies ensure that only authorized traffic and users can interact with the application and its resources.

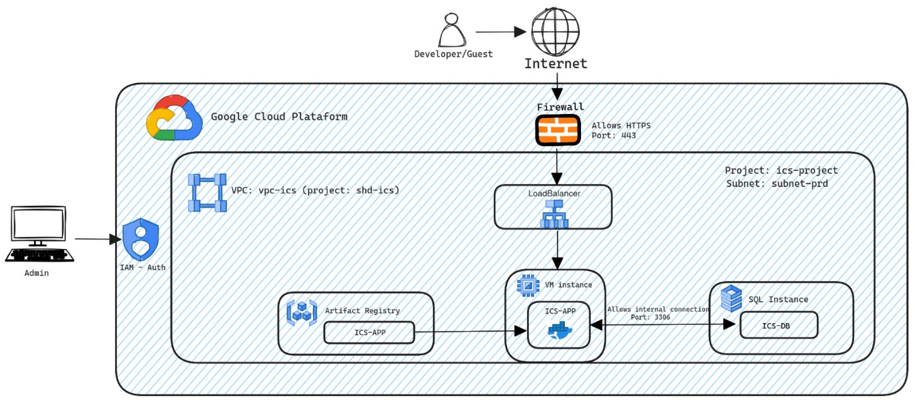


Fig. 3: Proposed system architecture diagram

Architecture components

a) User: a developer who accesses the application over the internet.
b) Internet: means through which the developer connects to the application.
c) Firewall: configured to allow HTTPS traffic on port 443.
d) Load balancer: LB distributes incoming traffic between application instances.

e) VM instance: a virtual machine instance where the application is executed (ICS-APP). This instance has Docker installed to contain the application.
f) SQL Instance: a MySQL database instance (ICS-DB) that stores the application data. Allows internal connections on port 3306.
g) Artifact registry: a repository where application artifacts (for example, Docker images) are stored. Contains the application image (ICS-APP).
h) IAM - Auth: the administrator uses IAM (Identity and Access Management) authentication to manage and deploy the application and its components.
i) VPC (vpc-ics): project-specific virtual private network (project: shd-ics). Subnet: subnet-prod.

Security in the developed software

a) CSRF Protection: Cross-Site Request Forgery is a process where an attacker tricks an authenticated user into executing unwanted actions in a web application. This validation against CSRF was added to forms and class-based views, which generates a token that is invited through a POST method. This protection was implemented as it is crucial to maintain the safety of users.
b) SQL Injection Protection: SQL injection is a process where an attacker inserts or manipulates malicious SQL queries to execute unauthorized actions on an application's database. The Django ORM is used to interact with the database instead of writing SQL queries manually. Django's ORM automatically escapes values, preventing SQL injections. Regarding the principles of least privilege, database accounts are configured with the lowest privileges necessary for the application. Implementing these practices is crucial to prevent SQL injections and ensure the integrity and security of your database. Django offers built-in tools that make secure development easy.
c) Protecting against XSS: Cross-site Scripting (XSS) is an attack where an attacker inserts malicious scripts into web pages viewed by other users. These scripts can steal sensitive data, manipulate page content, or perform unauthorized actions on the user's behalf. Implementing these practices is crucial to prevent XSS attacks and ensure the integrity and security of your web application. Django provides built-in tools that make secure development easy.
   i) Data Escaping: escapes all user-supplied data before displaying it on the web page. Django automatically escapes data in templates from it, but it is important to make sure this is applied at all times.
   ii) Use safe templates: use Django's templating system, which automatically escapes data to prevent malicious script execution.
   iii) Input Validation: validates and sanitizes all user input before processing or displaying it. This includes removing or encoding special characters that can be used in XSS attacks.
   iv) HTTPOnly and Secure Cookies: set cookies with the HttpOnly and Secure options to prevent them from being accessible from JavaScript and ensure that they are only sent over HTTPS connections.

Architecture flow

1) User access: the application is accessed from a device over the Internet. The request reaches the Firewall, which allows HTTPS traffic on port 443.
2) Application distribution: the request is received by the Load Balancer, which distributes the workload among the application instances.
3) Running the application: the Load Balancer redirects the request to a specific instance of the VM where the application (ICS-APP) is running in a Docker container. The application may need to access the database to retrieve or store information.
4) Database access: the application (ICS-APP) connects internally to the SQL Instance on port 3306 to perform database operations.
5) Administration and deployment: the administrator manages and deploys the application and its components using IAM credentials. Application artifacts, such as Docker images, are stored in the Artifact Registry and deployed to VM instances as needed.

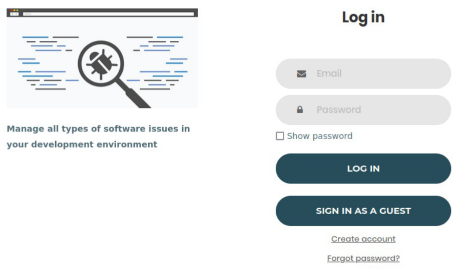
3.4 Prototype evaluation

For the monitoring part of the Prototype methodology phases, the evaluation part was defined with the fulfillment and validation of the objectives set in various iterations, for iteration 1 the entry process was completed through user and password. In iteration 2, functional remote access functionality was developed via web link; while in iteration 3 the module for recording, modifying, deleting and displaying errors made in development was generated. While in iteration 4 the functionality of adding comments to reported incidents was defined; continuing in iteration 5 with the procedure of showing graphs with the reported incidents. Finally, in iteration 6, the reports of the recorded errors were generated.

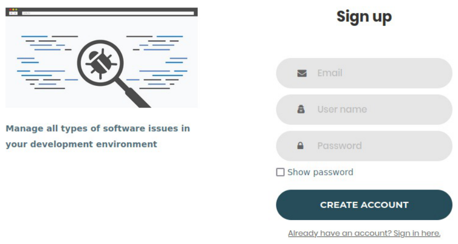
4 Results and discussion

Under the quality regime at the process level and at the product level, in addition to being guided by the DevOps Continuous Quality guidelines, the result of the proposal is generated through an expandable, scalable web application that encourages the accumulation of incidents/issues resolved through different stages and participants involved in software development.

In Fig. 4a you can see the login page, where developers can enter the system using their email and password; likewise, you can enter the system as a guest without the need to have an account. While in Fig. 4b you can see the registration page where a user can be created by email to be able to interact with the system. After this process, you can verify your home page (see Fig. 5), where you can navigate to the different system options. Once you have access to the system, a new incident can be generated (see Fig. 6a), where the registered user can add an incident and its possible solution. Once the incident has been registered, you can access the specification of the provided record (see Fig. 6b), where there

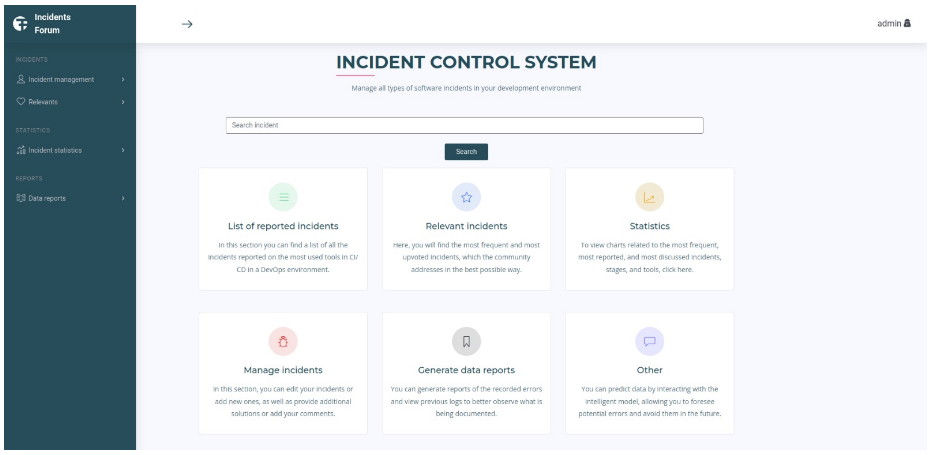


(a) Login page


Sign up
Manage all types of software issues in your development environment
Email
User name
Password
Show password
CREATE ACCOUNT
Already have an account? Sign in here.

(b) User registration page

Fig. 4: Initial user windows



INCIDENT CONTROL SYSTEM

Manage all types of software incidents in your development environment

List of reported incidents

In this section you can find a list of all the incidents reported on the most used tools in CI/CD in a DevOps environment.

Relevant incidents

Here, you will find the most frequent and most upvoted incidents, which the community addresses in the best possible way.

Statistics

To view charts related to the most frequent, most reported, and most discussed incidents, stages, and tools, click here.

Manage incidents

In this section, you can edit your incidents or add new ones, as well as provide additional solutions or add your comments.

Generate data reports

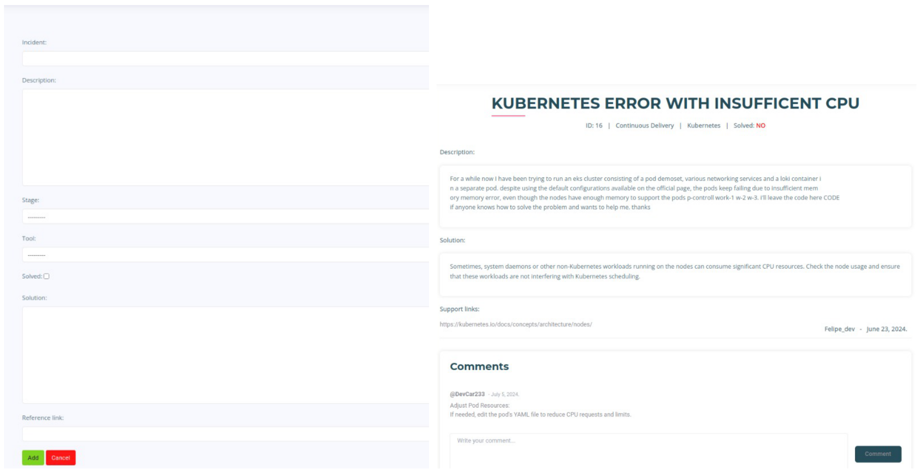
You can generate reports of the recorded errors and view previous logs to better observe what is being documented.

Other

You can predict data by interacting with the intelligent model, allowing you to foresee potential errors and avoid them in the future.

Fig. 5: System home page

is an incident, its details and comments that can be made by other registered users. In Fig. 7 you can see that three types of graphs are provided to determine the tools and stages where more incidents are recorded, and therefore where it is more likely to spend more development time. These graphs include bar graphs, for incidents recorded in specific CI/CD tools; cake graph, for incident stages of the development process; while the radar one is also shown, for stage and tool. Fig. 8a and 8b provide tools for listing and searching the most relevant incidents reported in the system. Page where you can create incident reports using filters and combinations of these for more efficient and exact searches. In the end, like any development process, it can be subject to improvements, to more iterations that generate more robust feedback from software industry participants.



Incident:

Description:

Stage:

Tool:

Solved: ☐

Solution:

Reference link:

KUBERNETES ERROR WITH INSUFFICENT CPU

ID: 16 | Continuous Delivery | Kubernetes | Solved: NO

Description:

For a while now I have been trying to run an eks cluster consisting of a pod demoset, various networking services and a loki container i
n a separate pod. despite using the default configurations available on the official page, the pods keep failing due to insufficient mem
ory memory error, even though the nodes have enough memory to support the pods p-controll work-1 w-2 w-3. I'll leave the code here CODE
if anyone knows how to solve the problem and wants to help me. thanks

Solution:

Sometimes, system daemons or other non-Kubernetes workloads running on the nodes can consume significant CPU resources. Check the node usage and ensure that these workloads are not interfering with Kubernetes scheduling.

Support links:

https://kubernetes.io/docs/concepts/architecture/nodes/

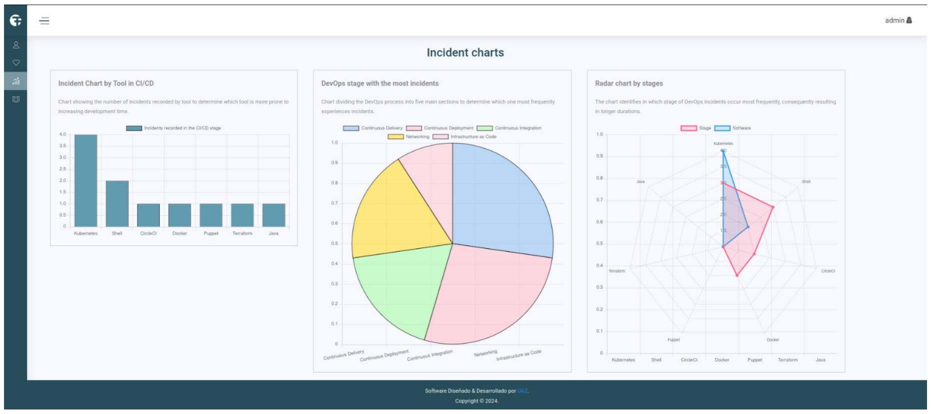
Felipe_dev - June 23, 2024.

Comments

@DevCar233 - July 5, 2024.
Adjust Pod Resources:
If needed, edit the pod's YAML file to reduce CPU requests and limits.

(a) Add new issue (b) Details of a new issue

Fig. 6: New issue registration page and its features



Incident charts

Incident Chart by Tool in CI/CD

Chart showing the number of incidents recorded by tool to determine which tool is more prone to increasing development time.

DevOps stage with the most incidents

Chart dividing the DevOps process into five main sections to determine which one most frequently experiences incidents.

Radar chart by stages

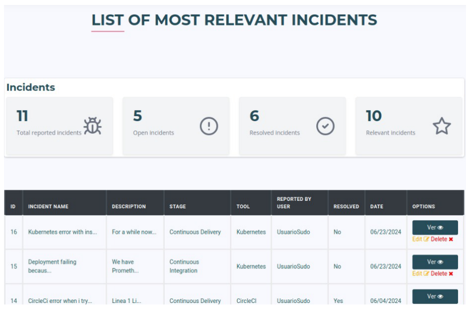
The chart identifies in which stage of DevOps incidents occur most frequently, consequently resulting in longer durations.

Software Diseñado & Desarrollado por UAZ.
Copyright © 2024.

Fig. 7: Incident graph page

5 Conclusions and future work

It is well known that there are currently various software development companies that work with an integrated workflow of various tools to apply the Continuous Quality concept paradigm, which go hand in hand with the DevOps process. This flow of data and actions is used to verify and monitor adherence to product quality at all times. However, the moment CI/CD tools begin to be used, errors/incidents and unplanned investments of time usually occur to correct these types of situations, since there is no central repository to store and consult the solutions to these facts. In this work, a proposal was defined, under the need to generate a tool that can categorize and store the most frequent errors (is-

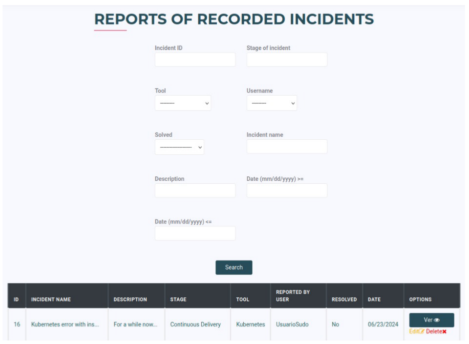

LIST OF MOST RELEVANT INCIDENTS

Incidents

11 Total reported incidents
5 Open incidents
6 Resolved incidents
10 Relevant incidents

ID	INCIDENT NAME	DESCRIPTION	STAGE	TOOL	REPORTED BY USER	RESOLVED	DATE	OPTIONS
16	Kubernetes error with ins...	For a while now...	Continuous Delivery	Kubernetes	UsuarioSudo	No	06/23/2024	Ver Edit Delete
15	Deployment failing becaus...	We have Prometh...	Continuous Integration	Kubernetes	UsuarioSudo	No	06/23/2024	Ver Edit Delete
14	CircleCi error when i try...	Linea 1 Li...	Continuous Delivery	CircleCI	UsuarioSudo	Yes	06/04/2024	Ver

(a) List of relevant issues


REPORTS OF RECORDED INCIDENTS

Incident ID
Stage of incident
Tool: ---------
Username: ---------
Solved: ---------
Incident name
Description
Date (mm/dd/yyyy) >=
Date (mm/dd/yyyy) <=

Search

ID	INCIDENT NAME	DESCRIPTION	STAGE	TOOL	REPORTED BY USER	RESOLVED	DATE	OPTIONS
16	Kubernetes error with ins...	For a while now...	Continuous Delivery	Kubernetes	UsuarioSudo	No	06/23/2024	Ver Edit Delete

(b) Issue search page

Fig. 8: Lists of issues and their search

sues/incidents), to later create libraries that help quality and also reduce the aforementioned factors that are lost with the unforeseen lack of quality. In the development process, the prototype methodology was used, which is directed under a flow of continuous iterations in order to achieve the desired product. As future work, it is planned to generate a supervised learning module that allows predicting future incidents, as well as their types and possible solutions.

References

1. Albert, F.: Continuous quality – the missing ci/cd ingredient. https://www.qualityclouds.com/continuous-quality-the-missing-ci-cd-ingredient/ (2020)
2. Alfonso, P.L.: Revisión de modelos para evaluar la calidad de productos web. experimentación en portales bancarios del nea. Master's thesis, Universidad Nacional de La Plata, Argentina (2012)
3. Ango Herrera, L.F.: Evaluación de sistemas. Master's thesis, Pontificia Universidad Católica del Ecuador, Ibarra (2014) 19p.
4. Bevan, N.: Los nuevos modelos de iso para la calidad y la calidad en uso del software. In: Calidad del producto y proceso software. Editorial Ra-Ma, España (2010) 5–75
5. Callejas-Cuervo, M., Alarcón-Aldana, A.C., Álvarez Carreño, A.M.: Modelos de calidad del software, un estado del arte. Entramado **13**(1) (2017) 236–250
6. Chen, L.: Continuous delivery: Huge benefits, but challenges too. IEEE Software **32**(2) (2015) 50–54
7. Deissenboeck, Florian Juergens, E..H.B..W.S..B.M..P..P.M.: Tool Support for Continuous Quality Control (2008)
8. Dowd, R.: Continuous quality. https://searchsoftwarequality.techtarget.com/definition/continuous-quality (2018)
9. Herrera H, A.: Bootstrap. Procesos Software (2012)
10. Hüttermann, M.: DevOps for Developers. Apress, Berkeley, CA (2012)
11. Khosravi, Khashayar Guéhéneuc, Y.G.: A quality model for design patterns. In: German Industry Standard. (2004)

12. Miglierina, M., Tamburri, D.A.: Towards omnia: A monitoring factory for quality-aware devops. In: Proceedings of the 8th ACM/SPEC International Conference on Performance Engineering Companion. (April 2017) 145–150
13. Mishra, A. Otaiwi, Z.: Devops and software quality: A systematic mapping. Computer Science Review **38** (2020) 100308
14. Mondragón Campos, O.: Integrando tsp y cmmi: Lo mejor de dos mundos. Software Guru **50** (2011) [Links].
15. Natarajan, D.: Implementing qms with erp software. (2017)
16. Purohi, K.: Executing devops ci/cd, reduce in manual dependency. International Journal of Scientific Development and Research (IJSDR) **5** (2020) 511–515
17. Riungu-Kalliosaari Leah, Mäkinen S., L.L.T.J.M.T.: Devops adoption benefits and challenges in practice: A case study. (2016)
18. Rodríguez, M., et al.: Evaluation of software product functional suitability: A case study. Software Quality Professional Magazine **18**(3) (2016)
19. Scalone, F.: Estudio comparativo de los modelos y estándares de calidad del software. Trabajo de maestría, Universidad Tecnológica Nacional, Facultad Regional Buenos Aires (2006)
20. Smeds, Jens Nybom, K..P.I.: Devops: A definition and perceived adoption impediments (May 2015)
21. Sommerville, I.: Software Engineering. 8 edn. Addison-Wesley, England (2010)
22. SonarSource: SonarQube 10.6 Documentation. https://docs.sonarsource.com/sonarqube/latest/ (2024) Accessed: July 8, 2024.
23. Soto Peña, J.R., et al.: Actividad 2.2: Cuadro comparativo de modelos para evaluar la calidad del software (módulo: evaluación de la calidad de la tecnología educativa). En: ISO 69 (2015)
24. Ståhl, D., Bosch, J.: Modeling continuous integration practice differences in industry software development. Journal of Systems and Software **87** (2014) 48–59
25. Vargas, Fabio Soto Duran, D.: Introduciendo psp (proceso personal de software) en el aula. Revista Colombiana de Tecnologías de Avanzada **2**(16) (2010)
26. Velazco, A.: Modelo en espiral introducción boehm (2016)
27. Zulfahmi, Toh Sahibuddin, S..M.M.N.: Adoption issues in devops from the perspective of continuous delivery pipeline. Proceedings of the 2019 8th International Conference on Software and Computer Applications (2019) 173–177