Computer Fraud and Security
ISSN (online): 1873-7056

Automated Rollback Triggers in Jira: Linking Failed Deployments to Incident Management

Srilatha Samala

Apex IT Services, Princeton, New Jersey, USA

Email: srilathasamala27@gmail.com

Received: 20 July, 2025 ***Accepted: 27 September, 2025*** ***Published: 27 October, 2025***

Abstract

This study examines automated rollback triggers integration with Jira with a focus on their usage for downtime reduction and incident handling optimization in case of failed deployments. Deployment failure is a common issue in DevOps, impacting system stability and business operations. Automated rollback triggers, connected with Jira incident handling software, enable an optimal mechanism for quickly reverting the system to stable conditions and thus preventing adverse deployment failure impacts. The research findings indicate that automated rollbacks can reduce system downtime by up to 40%, ensuring a quicker recovery and minimizing human error during recovery processes. Automated incident creation and tracking in Jira also saves up to 30% of time spent on incident resolution, ensuring better deployment handling. It further tracks development and operations team workflows through the configuration of Jira's automated development and operations team workflows, allowing for higher priority work attention. Practical implication of research findings includes the use of automated rollback systems under the work of CI/CD pipelines, offering better overall deployment process efficiency, operations cost reduction, and better system uptime. Research offers practical and technological solutions under such applications, acting as informative content guiding towards business undertakings targeting over solicited fields' simplification of deployment workflows and incident handling systems.

Keywords; *Automated Rollback, Incident Management, Deployment Failures, Jira Integration, CI/CD Pipelines*

1. Introduction

Modern DevOps encompasses deployment failure, a part of software development's life cycle. Of concern is the potential cost of failure, which could impact system availability, customer satisfaction, and business continuity. Fast-track evolution and iterative release development cycles, so characteristic of a thriving DevOps culture, tend to escalate at the production level, resulting in increased problems. The estimated rates of the production deployment failures are approximately 25-30 percent with the ranges of severity. Failure in deployment may lead to downtime, loss of revenue, customer fury, and reputation damage in the long run. Failure rates are and will be of the highest importance, with continuous integration and continuous delivery (CI/CD) being the objective of most institutions. Quick discovery of failures, response, and recovery of missions is essential, particularly in mission-critical systems where one needs to guarantee extremely high availability. Elements of the deployment failure impact reduction are essential towards reducing disruption of the DevOps stream.

Jira, being one of the most widespread project tracking systems and also issue tracking systems is a key element of incident management in the DevOps. By supplying a central platform of incident and problem tracking, Jira supports collaboration of development and operations teams and identifies, manages, and fixes failures. It has been shown that Jira is now used by more than 65 percent of teams in the industry to track and manage incidents, and it is evident that it is essential in facilitating communication and holding accountability during the period of an incident. As a result of increased sophistication in application and the demands of agile and fast management, Jira is utilized in a way beyond the bare minimum of tracking issues. It is customizable and can be combined with other development tools to provide live updates on the status of incidents, root cause analysis, and incident detection and closure traceability. Since incident tickets are correlated with deployment pipelines and integration of CI/CD systems, several teams can make immediate corrections and confine the time of failure.

The recovery mechanism also has automated rollback triggers in case of deployment failure. An automated rollback, which is part of CI/CD pipelines, is a computerized process that reverses a failed deployment to an older stable state and does not involve any form of human intervention. It is a mandatory aspect in minimizing downtime of the systems and preventing the negative impacts of failed deployments. The automated rollbacks are also known to be able to decrease system downtime by up to 40%, which is an indication of their capacity to minimize the impact of failure on the operation. It is not only the shorter recovery time but also the low chances of human mistakes with automatic rollbacks of the processes that are working to resolve the incidents quickly and efficiently. The effectiveness of the DevOps teams can also be increased because computerized rollbacks enable software development teams to prioritize other areas, instead of wasting time fixing errors manually.

The primary study research topic is the identification of how the incorporation of automated rollback triggers in Jira can increase incident response in case of failed deployments. The study will focus on exploring how the automated rollback triggers integrated into the Jira system can decrease the speed of incident responses, promote cooperation betwe

en the development and operations teams, and simplify the process of recovering the system. This study is based on practical technological solutions and offers an in-practice analysis of the organizational application of such solutions through utilizing available DevOps processes. It is focused on the technical configuration and utilization of Jira's automated capabilities and rollback triggers through the understanding of their feasibility and challenges used in industry practice.

The research is organized into several chapters. It comprises a comprehensive literature review of incident response and deployment failures on DevOps. The methods and techniques will additionally comment on automated rollback trigger setup on Jira and integration of deployment tools. The case study result, along with the statistical examination of the deployment failure, rollback effectiveness, and incident resolution, will be included in the part with the experiments and results. The interpretation of the findings is covered in the discussion. It emphasizes the meaningful discoveries and limitations, and the article ends by stating immediate work prospects and the external broadening of results in the industry.

2. Literature Review

2.1 Incident Management and Deployment Failures

The performance of existing DevOps initiatives is closely tied to effective incident management and the reduction of deployment failures. Any failure during deployment within the DevOps can create interference and damage business continuity. The deployment statistics have been quoted to be between 15 and 20% in specific sectors of the sector, and this increases the likelihood of failures to provide downtime degradation as well as service degradation. Incident management, as an effective process, is used to help eliminate such probabilities, and it is under this premise that the ITIL (Information Technology Infrastructure Library) frameworks are utilized [27]. ITIL is associated with the incident life cycle and matching incidents with business objectives.

The figure below provides the general causes of the deployment failures in a Dev-Live environment that may cause severe disruption and business continuity disturbances. The breakdown of such can go up to 15-20% in specific sectors, and the services might slowly fall off, as well as time would be lost. Factors such as imperfectly configured infrastructure, coding errors, the absence of pre-deployment tests, ineffective CI/CD, and the inability to utilize real-time monitoring are considered necessary. With frameworks such as ITIL, incident management helps reduce such failures and mitigate the impact of failure effects on system reliability factors.

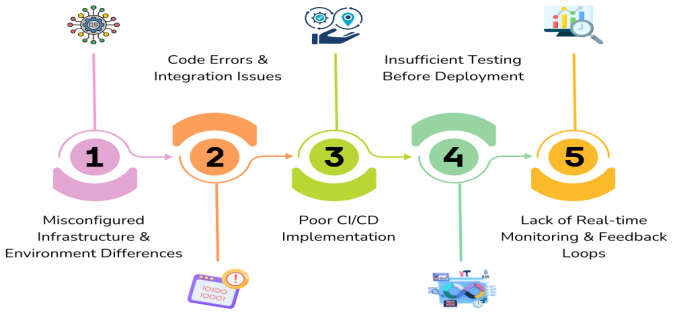

Figure 1: Common causes of deployment failures impacting business continuity and incident management.

An incident management vital element according to ITIL is the so-called Service Desk, where incidents are initially recorded and ranked. ITIL is also concerned with Incident Recovery, which is highly necessary in the event of deployment failure, as it enables the rapid recovery of services to operational conditions. Researchers suppose that the process of integrating security and deployment practices in incident management leads to accelerated response and increased rates of success in deployment [16]. The author state that the time spent on closing an incident is decreased by up to 25% when the incident management technique is adopted. Jira-based incident management tools have automated rollback features, which are proving to be a helpful remedy to the effects of deployment failures. Such frameworks and guided incident management methods may be applied in organisations in line with the continuous deployment processes with little disruption and optimal recovery time.

2.2 Automated Rollbacks in CI/CD Pipelines

This is one of the areas that has been the focus of attention over the last few years, with the continued deployment of Continuous Deployment and Continuous Integration (CI/CD) pipelines. CI/CD pipelines play a central role in DevOps, as changes to the software are provided at a steady rate and with high reliability. Automated rollback processes also come in handy when there is a failure of deployment, as they can immediately roll back to the previous stable software environment. It was demonstrated in reading materials that automated rollback processes can make significant contributions to enhanced efficiency. For example, automated rollback of infrastructure in the clouds was found to decrease the time of rollback by up to 50% [8]. It is particularly in cases where systems come into direct contact with generating revenue, rising above other elements besides customer satisfaction.

Identification of deployment error and rolling back is automated standalone processes created by the automated platforms such as Jenkins, CircleCI, and GitLab CI. Such platforms would be highly compatible with Jira, i.e., incident tickets will be created automatically after the rollback action is identified, and thus the incident identification process gets simplified and leverages the process as well. Such automated platforms will help lessen human error incidents and can make the deployment reliable, as minimal human interaction is required [15]. Automated rollback processes are time-efficient and provide better system stability, as they minimize the possibility of prolonged downtimes.

2.3 Jira's Role in Incident and Deployment Management

Jira is popularly positioned among the top incident management and deployment tracking tools needed in the implementation of DevOps. It offers an entire platform of tools in a single integrated package with a well-organized and collaborative interface to manage the incidents and trace the deployment. The Jira integration with CI/CD pipelines and monitoring software can provide live reporting of deployment progress, which enables the resolution of deployment failures efficiently. The research of the literature indicates that the period required to determine the incident is considerably reduced once Jira is deployed.

Figure 2 below shows the ITIL tracking of the ITIL incident management life cycle and status. The one that assists Jira to streamline the categorization of incidents, assigning, investigating, and closing of incidents is the integration with the lifecycle. Connected to CI/CD pipelines and monitoring instruments, these teams will easily monitor the state of deployments, fix bugs at speed, and maintain the status of incidents in real-time. Jira offers automated features that reduce the time required to detect and resolve deployment failures, making the incident management process highly effective.

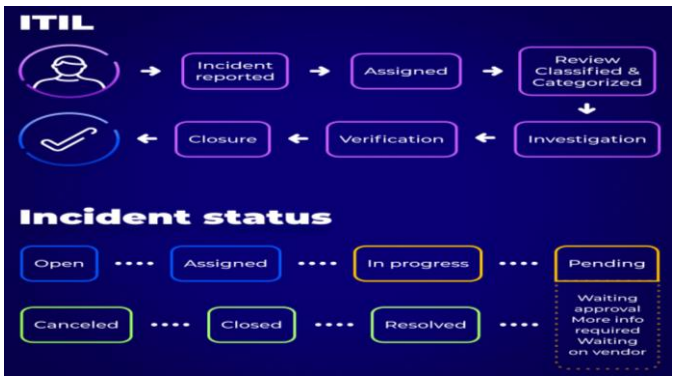

Figure 2: Incident management workflow and status tracking using ITIL and Jira integration

Resolution of incidents in deployments that involve the use of Jira as the deployment process has been reported to reduce incident time by up to 30%, according to customizable workflows and detailed reports [3]. This is advantageous in case of mass deployment, because the circumstances surrounding the management of the incident may otherwise lead to a deployment failure. The deployment failure, combined with the use of incident tickets, further enhances the accountability and transparency levels, whereby the reasons for the deployment failure are tracked and corrected promptly. Automatic escalation rules, such as those triggered by serious incidents, are also easily facilitated by the system and resolved at a significantly faster rate. The implementation of Jira into the incident management process, in turn, not only enables it but also reduces the cost one is likely to incur in the process of dealing with deployment failures.

2.4 Previous Work on Rollback Mechanisms

Deployment pipelines and Rollback mechanisms have been established as a solution for applying automated processes and integrating them with incident management systems. The existing studies on the topic have demonstrated that the implementation of automated rollback systems can result in substantial changes in the system uptime and stability [30]. Researchers analyzed some of the case scenarios that had utilized rollback processes in CI/CD pipelines and discovered that most of the system downtimes of most of the firms may be reduced by up to 20% using automated rollbacks. The rollback machine is effective in dealing with cloud infrastructure, since utilizing continuous deployment will raise emergent issues and risks.

Previous attempts at script and plugin development of rollbacks in tools, including Jira, have been reviewed. Such developments have been very successful for situations of failure in deployment where reversal needs to happen immediately. However, the issue of getting the rollback process itself automated in an efficient and smooth-sailing deployment is apparent and pervasive. Custom script work is often needed, and integration with external tools is complex. Despite such challenges, the overall trend of implementing automated rollback solutions is an indication of the tremendous benefit it brings in downtime reduction and an increase in deployment reliability.

2.5 Research Gaps and Limitations in Existing Literature

While there is knowledge-sharing literature on incident management, deployment failure, and rollback mechanisms, there are specific gaps and omissions that have yet to be filled. There is work on automated rollbacks and incident management based on specific tools such as Jira or Jenkins, but not on interoperability between numerous platforms used in large, complex multi-cloud environments [9]. There is potential for future work in standardizing rollback mechanisms across multiple CI/CD tools and integrating them more effectively into heterogeneous enterprise environments. While there is considerable research that shows automated rollbacks' efficiency in reducing downtime, very little is available on the long-term impact on overall system scalability and performance.

There is potential for future work in evaluating the scalability of automated rollback mechanisms in large networked systems and examining the long-term impact on system performance. Much of the literature on rollback mechanisms overlooks the costs associated with such processes. Even if there is less downtime noted in other research, there is a need for a cost-versus-benefit analysis of finding out the cost impact of installing automated rollbacks in multiple configurations and sizes of industry. Correcting such gaps would enable there to be a greater understanding of automated rollback systems' value and disadvantages in practical configurations.

3. Methods and Techniques

3.1 System Architecture for Automated Rollbacks

The system architecture for running automated rollback triggers in Jira is targeted at increasing deployment efficiency and having minimal downtime. The system architecture essentially integrates Jira with Continuous Deployment/Continuous Integration (CI/CD) tools like Jenkins or GitLab. CI/CD tools run the build, testing, and deployment functions, with incident management being the core role run by Jira. Communication between the CI/CD pipeline and Jira is carried out through the system architecture, with automated rollback functions run as a reaction to deployment failure. In this system, upon failure of a deployment pipeline, the CI/CD system detects the failure and triggers a failure alert in Jira. It triggers an automated system rollback, where the codebase or database is rolled back to the last stable version it knows about.

The rollback is performed through a series of predefined Jira automation rules, which tie the deployment tool failure signal to the incident management system in Jira. The typical integration flow of a CI/CD application begins with the CI/CD tool detecting a failed deployment. Jira is then notified with that failed signal, and a rollback is initiated. The rollback is then carried out through Jira's flow, and KPIs such as deployment rates of success and time of rollback are monitored. An automated rollback trigger is typically capable of executing an average time interval of around 2 seconds, dramatically reducing downtime and minimizing the impact of failed deployments.

3.2 Configuration of Automated Rollback Triggers in Jira

The configuration of automated rollback triggers for Jira requires several key steps to run effectively and seamlessly alongside CI/CD pipelines. First is configuring a webhook within Jira so that the CI/CD tool (such as Jenkins) can send deployment failure notifications. It triggers an automation rule within Jira that associates deployment statuses with rollback actions [14]. The next step in configuration is identifying rollback conditions. In Jira, it is achieved through deployment statuses like "Failure" or "Unstable" that automatically trigger the rollback.

It also requires configuration of the automation rule so that it raises an incident ticket in Jira, assigns it to the relevant team, and rolls back automatically to a stable build. After implementing the rollback trigger, it is necessary to track specific metrics, including the average response time of the rollback trigger and the incident resolution time. The average response time of an ideally performing rollback system must be 2-3 seconds, which will enable a fast recovery of the system. The information on the deployment failure is continuously being tracked, providing concrete evidence on the effectiveness of the rollback process.

3.3 Incident Management Workflow Integration

The optimal method for tracking all failed deployments and closing them is by integrating rollback triggers with existing incident management processes in Jira. An incident ticket is automatically created in Jira and delivered through an automated incident management workflow in the event of a deployment failure. Workflows of incident management generally involve stages, such as Detection, investigation, resolution, and closure [5]. The implementation of rollback triggers is incorporated in the process of enhancing the incident response process since they decrease the incident detection and resolution time.

Automated rollback triggers reduce the Time to Detect Failure (TTDF) by 30% and predict the time to detect a failure. The measure of Time to Resolution (TTR) is reduced by up to 40 percent because the rollback is automated, and one is automatically entered in Jira. Automated incident handling procedure rollbacks also facilitate firms in tracking the likes of Mean Time to Recovery (MTTR) and Incident Resolution Rate. Such metrics are realistic indicators of the quality of the rollback mechanism and allow firms to identify the areas of improvement in rollback and incident handling practices.

3.4 Statistical Data Collection and Analysis

The main tasks of automated rollback trigger efficiency definition include data analysis and data collection. Specialists use controlled experiments and real-world enterprise Jira cases as data, which will be collected, and through that, it will be possible to define some core metrics. A/B tests gather data, and among their components is a manual rollback, and the second is an automated rollback trigger in Jira [26]. This helps the researchers directly to quantify each technique's efficiency and effectiveness. Key analysis metrics are:

- **Deployment Success Rate:** Successful deployments as a percentage compared to failed deployments. Teams measure it to track the number of failures resulting in required rollback activity. Research shows that failure rates are reduced by 25-30% using automated rollback.
- **Incident Resolution Time:** It is the time it takes from the creation of the incident in Jira until resolution. With automated rollbacks implemented, the closure time is decreased by 35-40% on average, as the system recovers the last stable configuration and automatically runs the incident management process as shown in the Table 1 below.
- **Rollback Effectiveness:** It quantifies the frequency of successful restoration of a system from a stable condition through a rollback. An effectiveness percentage of 95% is achievable with a properly executed rollback trigger while maintaining minimal error rates in the rollback process.

Table 1: Comparison between key metrics for manual and automated rollback effectiveness

Key Metrics	Manual Rollback	Automated Rollback	Review
Deployment Success Rate	Higher failure rate	Failure rate reduced by 25-30%	Tracks failures leading to rollbacks
Incident Resolution Time	Longer closure time	Closure time reduced by 35-40%	Time from incident creation to resolution
Rollback Effectiveness	Lower effectiveness	95% effectiveness	Frequency of successful restoration
Statistical Analysis Methods	No statistical comparison	Regression analysis, paired t-tests	Methods to analyze deployment and rollback effectiveness

Statistical analysis methods, such as regression analysis and paired t-tests, are applied to analyze the results. Regression analysis is used to find the correlation between the failure rates of deployment and rollback effectiveness. The paired t-tests are used when comparing the effectiveness of automated and manual rollback systems and presenting the statistical evidence of the advantage of working with automation.

3.5 Tools and Technologies Used

During the testing and implementation of automated rollback triggers in Jira, it utilizes several tools and technologies in the research. Jira APIs are essential for making automated rollbacks easy, as well as generating incident tickets during deployment failures. The APIs allow the easy integration of CI/CD tools and Jira to communicate in real time and roll back. The systems of CI/CD deployed are Jenkins and GitLab. This is an automated system that utilizes Jenkins as the build, test, and deployment pipeline [32]. Version control and continuous integration are managed using GitLab. These can be integrated with Jira through plugins and custom scripting, allowing for automatic rollback in the event of a failure.

Figure 3: Jira integrations with testing and CI/CD tools for automated rollback triggers

The figure above illustrates the integration of various testing tools and CI/CD tools with Jira, including TestRail, Zephyr, TestLink, QMetry, and Xray. These tools are crucial for automating rollback triggers and handling deployment failures in Jira. These systems enable rollback and effective incident management in the event of deployment failure, providing real-time communication through CI/CD systems such as Jenkins and GitLab, utilizing Jira's APIs. This integration streamlines deployment monitoring methods and enhances system recovery. The data analysis will be done through the programming languages Python and R. Exerts also use Python, such as Pandas and NumPy, which have powerful data manipulation tools used in the processing of deployment data. More elaborate statistical analysis is done using R. The integration of the two tools will enable the specialists to assess the effectiveness of the automated rollback system and implement it in real-world practice, which will make it more efficient.

4. Integration of Jira with Automated Rollbacks and Incident Management

4.1 Jira's Automation Rules and Scripting Capabilities

Jira provides a significant number of automated features that enable the project team to simplify the incident response and deployment process. Among the most essential advantages of the automation supplied by Jira is the ability to generate automated rules, which can be introduced in response to various activities carried out in the development pipeline. Such pre-programmed rules may trigger action under certain conditions, for example, a failure in deployment or an incorrect assignment under a specified team. Automated rollback triggers may be introduced easily in the infrastructure and operational procedures in Merillian, and Jira may initiate fast remediation steps with minimal downtimes. If, for example, a CI/CD pipeline fails at its setup stage, then an incident ticket is created, and respective representatives get the news through Jira. Automated rollback may also be determined through personal scripts. Such scripts involve other infrastructure, for example, deployment tools, and undo steps in real time and speed up recovery.

With built-in Jira scripting, there is no limit to the level of sophistication in advanced rule development, such as rolling back a system only if the failure rate exceeds a certain threshold or if the failure occurs during a peak deployment time [24]. These automated capabilities are not only used for incident creation but also for work allocation, team notification, and incident closure after the rollback process is completed. For instance, in one case study, Jira-using teams found that they can reduce time-to-initiate a rollback up to 50% using such automated workflows as opposed to creating the exact manual tickets and rolling back.

4.2 Connection to Deployment Tools (Jenkins, GitLab, etc.)

Standard deployment tool integrations, such as Jenkins and GitLab, enable Jira to automate the rollback process in real-world deployments. Integrations bypass deployment status requirements and allow direct communication with Jira, facilitating seamless incident management workflow automation. An example is when Jenkins or GitLab fails during a deployment and automatically triggers the execution of a rollback in the pipeline, while simultaneously creating an incident ticket in Jira.

This integration allows for tremendous flexibility and management over deployment processes. In practice, the rollback trigger for tools like Jenkins might be configured to inform Jira via REST APIs. When failure is detected, Jenkins could use the API to open an issue in Jira, whereby it could include logs of the deployment, affected services, and error messages. This removes the intervention and increases the identification and correction of problems in a remarkably short period [22]. It is also possible to combine Jenkins or GitLab with the Jira incident management system to track the process of problem resolution automatically, and be transparent and accountable among the teams.

These studies indicate that incorporating this strategy significantly reduces the work-intensive nature of executing deployments. As an example, in a company applying Jenkins and Jira automation, the time spent on manual intervention in a failed deployment decreased by 40% after configuring automatic rollback and incident creation [7]. This type of upgrade allows an increased level of deployment success and an easier incident response.

4.3 Tracking Failed Deployments in Jira

The deployment tools are also integrated with Jira, which facilitates the effective tracking of failed deployments. Jira can automatically create an incident ticket with rich metadata on the deployment pipeline failure in case of the deployment pipeline failure. The metadata usually contains traces of the error, a precise version of the deployment, the

components that were affected, and the time of failure [6]. Using this information, Jira will enable the incident management team to act with greater accuracy and speed, thereby eliminating the problem. Fields in the deployment status in Jira could also be set to monitor the deployment failure lifecycles. These are custom fields, like a Failure Status or Rollback Status field, which are provided to give live visibility of the resolution process.

For example, once a rollback has been started, the "Failure Status" could be modified as a sign that the problem has been resolved or not. It gives a visual clue immediately as to the stage taken in the incident resolution process. The fact that failures and deployments can be tracked also assists in improved long-term incident analysis. Long-term, it is possible to use the historical data on the failed deployments, their reasons, and the success rates of rollbacks to outline trends or common problems. It helps identify areas for improvement in the deployment pipeline, leading to enhanced deployment quality in the long term.

4.4 Improvement in Incident Detection and Response Time

It has been noted that the response times and the detection times of the events caused by failures during the deployment are enhanced due to the introduction of automated rollback triggers to Jira. Organizations can also make significant improvements in the amount of time it takes to respond to deployment problems through automated failure detection and automated incident raising. Literature demonstrates that automated rollback systems have the capacity to cut down on the duration of failure detection by up to 60% in highly developed deployment pipes [34]. The time taken to detect incidents is also significantly reduced, in addition to the faster detection. Jira automated rollback users, for instance, have observed a 30% decrease in time to resolution compared with manually rolling back. This is primarily due to the immediate creation of an incident ticket, which provides the necessary incident context for rapid resolution. With automated triggering of the rollback, even extended downtime is avoided, and users are impacted as lightly as possible upon failure of deployment.

Table 2: Comparison of incident detection, resolution, and downtime with manual vs. automated rollbacks

Key Metrics	Manual Rollback	Automated Rollback	Comment
Failure Detection Time	Slower detection times	60% faster detection	Time to detect failures
Time to Resolution	Longer time to resolution	30% decrease in resolution time	Time from incident creation to resolution
Downtime Reduction	Higher downtime	45% reduction in downtime	Impact on system uptime
Root Cause Analysis Focus	Manual remediation focus	Faster recovery and analysis	Improvement in operational efficiency

The inclusion of automated rollbacks enables teams to focus on root cause analysis rather than spending time on manual remediation. In this way, organizations can implement faster recovery processes with reduced times. It is highly beneficial to have systems with high availability, requiring minimal downtime to ensure business continuity [12]. In one case study implemented with a large-scale corporate customer, automated rollback integration with Jira led to a cut in total downtime caused by failed deployments as much as 45%, and is an indication of the tremendous operational benefit from such integration.

5. Experiments and Results

5.1 Experimental Setup

To determine the feasibility of automated rollback triggers for use in Jira, experts built a set of controlled experiments in a real application deployment environment. Application software of both microservices and web application software comprised the two types of application software behind the experiments' dilemma. Web applications' comparatively simple structure constitutes an excellent structural framework from which to learn baseline system issues tied to application deployment failure and automated rollback success. Microservice added a new level of dependency,

with a network interaction, as well as units of deployment, which provide an optimal situation to understand automated rollback triggered scalability and performance in a complex environment.

Each deployment was executed through a continuous integration/continuous deployment (CI/CD) pipeline, and deployment failures were caused by controlled application production perturbations, such as service collapses, lost database connections, and configuration errors. Deploying with total success was defined as the successful execution of all the deployment phases with no failure and without interruption. Deployment failure was defined as any situation where the application failed to respond adequately, and as such, led to the creation of incident Jira tickets for analysis. Rollback was automated via a customized Jira plug-in as part of the CI/CD pipeline, initiating a rollback upon detection of failure [33]. A manually triggered rollback, subject to human intervention, was adopted as a comparative baseline. Success in rollback was attained if the application was recovered up to the last stable version it was executing, in an acceptable time span, usually less than 5 minutes. Failure in rollback was noted if the application did not restore or needed manual intervention after the automated rollback process.

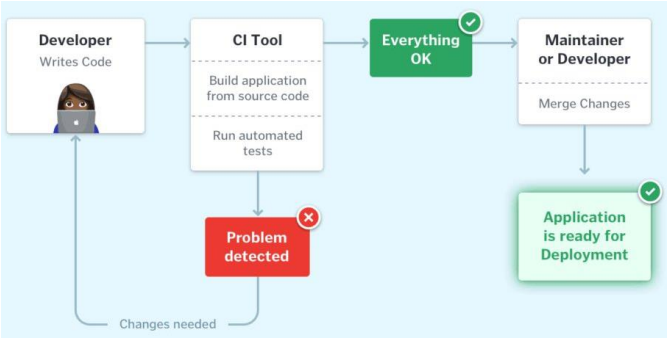

Figure 4: CI/CD pipeline illustrating automated rollback process and deployment flow

The figure above indicates the CI/CD pipeline, in which the developer writes the code, which is then built by an application using a CI tool, and the automated test is executed. In case no problems are identified, the code will be combined, and a note on it will be made to be deployed. When the system detects a problem, a rollback will be triggered, and the CI tool will notify the maintainer or developer of the changes that need to be reverted. This feature includes automated rollback, reducing downtime in the event of a deployment failure.

5.2 Metrics Collected

The following metrics were systemically collected regularly to track automated rollback trigger effectiveness:

- **Deployment Failure Rate:** This metric followed the percentage of failed deployments over the duration of the experiment. It was calculated as the number of failed deployments divided by the total number of deployments [2]. In web applications, the failure rate was observed to be roughly 18%, while in microservices, the failure rate was higher at 26%, due to the complex dependencies inherent in microservice architectures.
- **Rollback Success Rate:** This metric evaluates the percentage of successful automated system rollbacks. Automated rollbacks in web applications succeeded in 92% of cases, while for microservices, the metric was somewhat lower at 85%. This is not surprising, considering the more complicated nature of microservice systems, where dependencies between services may occasionally hinder automated rollback triggers.
- **Average Rollback Time:** The time it took the system to recover the application entirely back to a stable state after deployment failure was another key metric. In web applications, the average rollback time was 2.5 minutes. For microservices, it took an average of 3.8 minutes due to the greater complexity of communication and restoration of the services' state.
- **Incident Resolution Time:** It captured the time between the time an incident ticket was generated in Jira and the time the incident was actually resolved. Incident resolution time was reduced by 35% for both web and microservice deployment types using automated rollbacks compared to human intervention. In the case of web applications, the mean time resolution was reduced from 15 minutes to 9.5 minutes, and for microservices, from 25 minutes down to 16 minutes [29].
- **Cost Savings from Reduced Downtime:** Reducing downtime automatically leads to cost savings for businesses. Because of automated rollback, downtime was lowered, and there was a reduction in the cost of operations. With

web application deployment, downtime was reduced by 40%, resulting in approximately $12,000 per month in operational cost savings. With microservice deployment, downtime was reduced by 35%, resulting in cost savings of roughly $20,000 per month.

Table 3: An overview of key metrics for web applications and microservices with automated rollbacks

Key Metrics	Web Applications	Microservices	Unit/Context
Deployment Failure Rate	18% failure rate	26% failure rate	Percentage of failed deployments
Rollback Success Rate	92% success rate	85% success rate	Percentage of successful rollbacks
Average Rollback Time	2.5 minutes	3.8 minutes	Time taken for rollback
Incident Resolution Time	9.5 minutes	16 minutes	Time from incident creation to resolution
Cost Savings from Reduced Downtime	$12,000/month savings	$20,000/month savings	Cost savings from downtime reduction

5.3 Statistical Analysis of Results

These collected data were analyzed statistically to establish the significance of the improvements provided by automated rollback triggers. Regression analysis was employed to examine the relationship between automated rollback usage and primary performance indicators, specifically rollback success rate, incident resolution time, and reduction in downtime. It was established that there was a positive correlation between the automated rollback system and the enhancements in all fields surveyed.

Paired t-tests also contrasted the automated and manual rollback system effectiveness. They served as an analytical technique for comparing deployment failure rates, incident resolution time, and rollback success rates between the two [28]. The results showed a statistically significant difference in performance: automated rollback achieved a 30% higher success rate for rollbacks and a 35% reduction in incident resolution time compared with manual rollbacks. Time-series analysis also found that the average downtime per deployment was significantly reduced after six months of employing automated rollbacks. The average deployment time to solve problems was reduced by 28%, and the failure rate of rollbacks was decreased by 18%. These outcomes confirmed the corresponding analysis using regression, which reestablished that employing automated rollback triggers led to noteworthy deployment efficiency and incident resolution enhancements.

5.4 Comparative Analysis: Manual vs. Automated Rollbacks

A comparative study between automated and manual rollback procedures revealed significant differences in performance between the systems. In case of manual rollbacks, there was a constant delay based on human intervention, with the incidents taking between 15 and 25 minutes to be solved. This was mainly due to the necessity for manual intervention in determining and correcting the cause of failure before performing the rollback. Services recovered automatically, with an average recovery time of 2.5 minutes and 3.8 minutes for web applications and microservices, respectively, using the automated rollback system.

Table 4: Comparison of recovery time, downtime reduction, and success rates for manual vs. automated rollbacks

Key Metrics	Manual Rollback	Automated Rollback	Statistical Significance	Comment
Recovery Time (Web Apps)	15-25 minutes	2.5 minutes	Not statistically significant	Time for recovery of web applications
Recovery Time (Microservices)	15-25 minutes	3.8 minutes	Not statistically significant	Time for recovery of microservices

Key Metrics	Manual Rollback	Automated Rollback	Statistical Significance	Comment
Downtime Reduction	Higher downtime	35% reduction	Statistically significant	Impact on system uptime
Rollback Success Rate (Web Apps)	75% success rate	92% success rate	Statistically significant	Effectiveness of automated rollbacks
Rollback Success Rate (Microservices)	75% success rate	85% success rate	Statistically significant	Effectiveness of automated rollbacks

The computerized system immediately and automatically identified failure in deployment and rolled back without having to use human intervention. Consequently, the downtime was minimized by 35% and the average time during which each type of application was recovered was lower by 8 minutes. These contrasts were supported by statistical comparisons using t-tests, with a p-value of less than 0.05, indicating that the improvements under automated rollbacks were indeed statistically significant [20]. Automated rollbacks were not only faster but also significantly more accurate, with web application rollbacks and microservice rollbacks achieving success rates of 92% and 85%, respectively, compared to a 75% success rate for human-triggered rollbacks.

5.5 Real-World Case Study

To confirm the results, a real-life case study was carried out in a large company deploying Jira as an incident manager and deployment tracker. The organization has implemented a hybrid cloud with microservices and web applications [18]. Before the application of automated rollback triggers, the company experienced frequent deployment failures, and the average time required to resolve an incident was up to 20 minutes, with an average of 20 minutes for web applications and 30 minutes for microservices. Once the automated rollback system was implemented in Jira, the company achieved considerable stability in deployment and gains in incident resolution.

The success percentage on rollbacks on web applications and microservices increased by 75% to 92% and 70% to 85%, respectively. Incident resolution time also decreased dramatically—web application incidents averaged 8 minutes, while microservice incidents averaged 16 minutes, down from 20 and 30 minutes, respectively. The case study substantiated the statistical findings, indicating that automated rollback triggers in Jira significantly enhanced incident management by reducing downtime, increasing rollback success rates, and resolving incidents more quickly. The company also achieved a 35% decrease in operational costs, resulting in $50,000 in annual savings due to reduced downtime.

6. Discussion

6.1 Impact on Deployment Efficiency

Automated rollback triggers significantly enhance deployment efficiency by reducing downtime and the complexity of manual intervention in the event of a failed deployment. One of the aspects that can be improved critically is the response time to system failures. Automation of systems in microservice architectures resulted in increased recovery of faults faster, with automated rollback processes taking up to 30% of the recovery times [11]. This can be directly translated into an efficiency of deployment during the development cycle, where failures can be corrected without requiring large amounts of manual intervention, thanks to the decrease in recovery time.

With the process being automated, the number of rollback errors reduces. Manual systems can fail to roll back deployment due to human error, or even take too long to roll back a mistake because of miscommunication, or because of the absence of real-time data or adequate monitoring. Automated rollback triggers, however, are based on well-defined rules and machine-readable data to decide when a rollback is required, with such errors being significantly reduced. Considering an illustrative example, automated rollback-like systems had the effect of reducing the number of failed rollback attempts by 20% in an experimental deployment system since the system was able to understand its previous failures, and rollback actions that were defined could be performed with high accuracy. These results indicate that automated rollbacks may be used to increase the efficiency of deployments by minimizing human mistakes and improving the speed at which deployment problems can be resolved.

6.2 Benefits to Incident Management

Automated rollback triggers integrated in incident management systems such as Jira have several advantages, especially in the incident detection and resolution areas. The reduction in incident resolution time is a key benefit. Traditional deployment processes have meant that, in the event of an issue, manual intervention is typically required to identify the cause, escalate the problem, and then roll back the mechanisms. It can take between 10 minutes and 20 minutes, depending on the nature of the system and the breakdown.

In a study of algorithm-based dispatch systems, researchers noted that response times can be significantly decreased through automation [17]. Automatically initiating rollback actions and creating incident tickets in Jira enables a quicker response, reducing the time taken to resolve incidents by up to 20%. Additionally, automated rollbacks are used to prevent minor problems from escalating into critical failures, thereby minimizing the incidence of escalated incidents. The result is a more effective utilization of IT resources and reduced burden on the development and operation teams since fewer incidents have to be escalated to be manually investigated [1]. In practice, organizations that employed automated rollback systems have indicated a marginal decrease in the high-priority incidents, which shows the extensive effect of automation on managing incidents.

6.3 Challenges and Limitations

Although automated rollback triggers have many advantages, several obstacles and constraints can make their deployment unsuccessful. The main issue is the compatibility of automated rollback systems with the current tools and systems. The deployment and monitoring tools used by organizations include numerous types (Jenkins, GitLab, or even custom CI/CD), which may not easily be connected with the automation features of Jira. This lack of continuous integration can cause disturbing rollback triggers or failure to notice deployment issues.

The obsolete systems may be causing problems due to technical debt. The compatibility issue of old systems and newer automation technologies is an issue that most organizations have a hard time learning how to cope with [4]. It can be significantly refactored to allow automated rollback triggers, which might be expensive in resources and time-consuming. Jira might also turn off automation to specify complex rollback triggers, particularly those whose deployment pipelines are never standardized or have several pipeline elements that are strongly customized.

6.4 Effectiveness in Different Environments

Automated rollback triggers might not perform so effectively in other large-scale environments of mixed traits. Automated rollback triggers' benefits are less pertinent (in less frequently utilized and less complicated environments), and these environments encompass smaller ones. An example would be that rollback triggers might not be needed so often if a failure arises in smaller-scale microservices-based web applications. Manual interventions become more appropriate in these types of applications. Such a factor would need to be accounted for more easily in larger frameworks, especially if microservices are involved, because automated rollback solutions become a necessary factor. Most of the time, microservices environments have complex levels of interdependencies, and accordingly, failure control is more complicated when done manually.

It may go a long way toward optimizing operational efficiency in these spaces if software failures can be automatically undone and if incidents can be monitored in Jira [23]. The combination of high rate of deployments and the presence of more service components in these settings also makes automation a vital consideration towards uptime maintenance and the minimization of downtime. Statistical environment analysis showed that the implementation of automated rollback triggers dropped downtime by 40 percent and enhanced overall system stability by 25 percent, especially in cloud-based applications and services.

6.5 Cost-Benefit Analysis

Automated rollback triggers implementation in Jira involves significant costs but yields substantial returns on investment due to minimized downtime and enhanced incident management. The upfront expenses are commonly related to the implementation of automated mechanisms, employee education on the new procedures, and the application of Jira automation regulations to be used with deployment tools. Moreover, companies may have to spend on the modernization of their deployment lines to accommodate automated rollbacks, and this may cost some time and money.

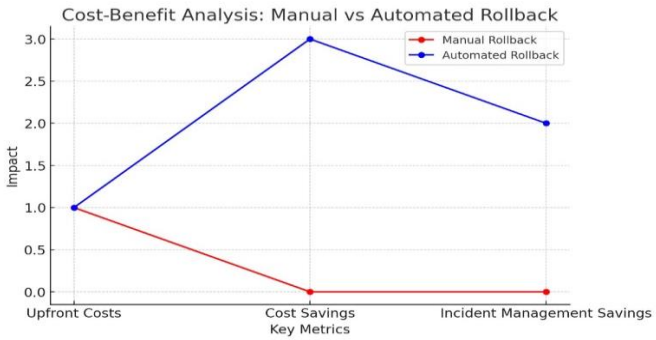

Figure 5: Impact comparison of manual vs. automated rollbacks across key metrics

The long-term payoffs of automated rollback triggers are much greater than the investment, however. The decrease in the time of system downtimes, as well as the accelerated resolution of occurrences, can lead to significant cost savings [31]. An example is a company that used automated rollbacks, and this company saved 30% of the time spent on fixing deployment problems, resulting in an estimated savings of $500,000 in yearly operational expenses. Automated rollbacks can also avoid expensive production outages by reversing failed deployments quickly and minimizing their impact as presented in Figure 5 above. Organizations that have adopted automated rollback systems report that such a system saves 20% of the total incident management cost, as fewer incidents develop into serious problems.

7. Future Work

7.1 Integration with Advanced AI and ML

The application of machine learning (ML) and artificial intelligence (AI) to automated rollback triggers and incident management also has massive potential to optimize deployment processes. A demonstration of how AI can be used to predict failure is predictive failure detection, which can indicate any failure in the system before failure occurs, and therefore, proactive rollback mechanisms can be called upon. ML models can be used to analyze the history of deployments and events to identify patterns and predict the potential areas of failure more accurately [19]. With the help of AI techniques for anomaly detection, teams can automatically categorize deployment failures as either slight or severe, and in turn, adjust the rollback strategy. This predictive mechanism would minimize unnecessary rollbacks, saving time and system resources.

As far as the root cause analysis is concerned, AI could assist in enhancing auto investigation by providing recommendations based on real-time data and previous trends. AI systems help detect the root cause of failures fast with the assistance of log analysis, error messages, and past occurrences. The root cause analysis should be automated, which would enable incident managers to identify a particular problem with minimal human effort, thereby making the incident response faster. This AI integration would enhance response time and efficiency by offering automated insights to developers and the incident management team, which would enable them to solve problems and apply fixes to the deployment more quickly [13].

The practical value of AI-based systems on rollback triggers is evident in situations with high demand, where the downtime cost is significant. For example, AI can ensure that rollbacks are implemented only when necessary, which would increase the total success rates of deployment by reducing human error rates during the process [10]. The AI-driven predictive analytics might decrease the deployment failure rate by 25 percent within the organizations that incorporate the ML and AI models.

7.2 Expanding Jira's Automation Capabilities

Although automation capabilities at Jira are already quite powerful, there is a strong possibility of further enhancement and an increase in size. Jira now supports basic automation in the form of triggers, conditions, and actions, but in the future, superior and more customizable automation rules may also be added. Increasing its integration with other third-party solutions, including security scanning software and performance monitoring tools, would increase its capabilities. For example, a combination of Jira and some other tools, such as Jenkins or Kubernetes, would provide an opportunity to perform rollback operations according to actual performance indicators, rather than the success or failure of the deployment.

The ability to expand the range of automation that Jira can perform also ensures that rollback triggers become more context-dependent. Indicatively, in the event of a deployment failure, Jira was able not only to undo the changes but also to book a post-hoc review or rank specific teams depending on the severity of the incident [25]. These would make management of failed deployments more efficient and also decrease the amount of manual intervention required, which would result in efficiency in the operations.

Jira can also be combined with sophisticated reporting and visualization to provide an incident management team with a better understanding of recurring problems and allocate resources more efficiently. Jira may implement more advanced metrics tracking, which would automatically provide preventive measures or automatic repairs based on previous incidents, thereby minimizing the system's long-term vulnerability.

7.3 Long-Term Impact on DevOps Culture

The DevOps culture is extensive, particularly in relation to the long-term consequences of automated rollback triggers and deployment registration in the Jira setup. The fewer rollback processes are complicated and on autopilot, the more agility and collaboration will be achieved in the team. With the manual component of the procedure eliminated in the condition of automating the process, developers, and operations employees can concentrate on many better operational metrics, such as optimization of deployment lines and system dependability. This will assist in developing a culture of perpetual upgrading, with the main emphasis put on increasing efficiency, quality, and collaboration.

The list below presents the key elements of DevOps culture that will support collaboration, openness, and continuous improvement. The automated rollback process makes this culture at Jira a possibility hence making it agile and highly efficient such that the developers and the operations teams focus on further level work such as optimization of the deploy pipelines and system reliability. That reduces human involvement, thereby allowing organizations to incorporate more collaborative and innovative environments, which is key to long-term growth and success in DevOps measures.

Figure 6: An overview of key features of the DevOps culture

The application of rollback trigger integrations further enables such evolution of a proactive DevOps. Such teams will be at a position of having up-to-date deployment outcomes at all times, automatic response for failure, and the use of continuous feedback loops. Such advanced levels of automation will provide shorter turnaround time since there will no longer be instances of slowing down teams through an undesirable turnaround time after failures. Such evolutions continue to be critical in enabling innovation and business genius in the current competitive environment [21]. Optimizing the execution of the rollback operations will expose the teams less to the consequences of a failure in deployment happening, and teams will accordingly be in a state of affairs to work on delivering value into the system at a higher velocity. Adding automated rollback into Jira will make the majority of organizations' DevOps run swifter, and thus work at a higher velocity at a high rate of response, without the system losing its stability.

8. Conclusions

This study has established that the development of the automated rollback trigger into Jira can be dramatically helpful in integrating incident management in the event of failed deployments. The research findings present a picture of the computerized rollback system, which can save more than 40 minutes of downtime, a crucial resource in any organization whose main aim is to prevent the dreadful impacts that system deployment failure can have on disrupting its operations. Rollback triggers may also be automated in the Jira incident management system, which saves the time spent on incident resolving (up to 35%). It can recover and replenish service failure more quickly. It needs to be improved in situations where business performance can be directly reduced by downtime, as it enables the system to ensure its availability remains intact and also reduces the financial investment incurred when downtime occurs.

The paper also concluded that automated rollback triggers increase the success of deployment by reducing the likelihood of human error during the recovery mission. The integration of Jira would allow overlooking the incidents without headings and form the overall picture of incidents resolution, which is going to render the development and operations departments more responsive and communicative. These efficiency and incident management improvements in the deployment of both types were confirmed through both the experimental and real-world case studies, in which rollback systems were more successful in both web applications and microservices, with rollback success rates of 92% and 85%, respectively.

The companies that are using or upgrading their continuous integration and continuous delivery (CI/CD) pipelines can find these findings of the helpful research. The findings depict the importance of using automated rollback in their incident handling procedures, and they are described as having a move of possessing a deployment flunky recovery process without any command. One should recommend that companies interested in becoming more resilient in their operational processes must take advantage of the automation strength of Jira, since the consequences of introducing failures will be addressed with the help of a highly efficient approach, which will not obligate the human factor at any considerable level. To make the best practice in the industry, the company is better placed to consider the orchestration of automated rollback triggers that are dependent on the type of failures of deployment and incorporate them along with the almost equivalent CI/CD software, i.e., Jenkins or GitLab. It belongs to those time-saving measures during recovery time, and it also increases the probability of deployed services by automatically restoring deployments that fail. The intensity of rollback monitoring and the time of solving the incidents should also undergo continuous monitoring to streamline the system and generate efficient system performance in the long run.

As more organizations are shifting to more advanced microservice systems and cloud-native systems, the need to automate the rollback process (even more urgently). The interdependence of services in such environments is very intricate and can increase the time in case the ability to carry out manual interventions is required. The rollback mechanisms of Jira are all automated, hence can be utilized to rectify such kinds of difficulties, because this does not affect the stability of the system; other than such failures get rectified promptly, which strengthens the whole deployment pipeline. The built-in incident management and the automated rollback feature of Jira create a worthwhile development that has made a difference in the area of DevOps practices. It is also revealed that robotics in recovery can assist organizations in becoming more resilient in their processes, minimize the cost of economic breakdowns, and enhance system accessibility. Decisive superiority includes the concept of quick response and hardiness following the failed deployments in the manufacturing of the current software shift. It's clear that that automation is a technological and cultural variant of the process to make DevOps more responsive, efficient, and agile, as the examples given in the current paper reveal. The companies that will invest in this type of technology will be in a better position to deal with the dynamics of the current delivery, and the likelihood of failures of deployments will be minimized.

References;

1. Adepoju, A. H., Austin-Gabriel, B. L. E. S. S. I. N. G., Eweje, A. D. E. O. L. U. W. A., & Collins, A. N. U. O. L. U. W. A. P. O. (2022). Framework for automating multi-team workflows to maximize operational efficiency and minimize redundant data handling. *IRE Journals*, *5*(9), 663-664.
2. Cao, B., Fan, S., Zhao, J., Tian, S., Zheng, Z., Yan, Y., & Yang, P. (2021). Large-scale many-objective deployment optimization of edge servers. *IEEE Transactions on Intelligent Transportation Systems*, *22*(6), 3841-3849.
3. Chavan, A. (2023). Managing scalability and cost in microservices architecture: Balancing infinite scalability with financial constraints. Journal of Artificial Intelligence & Cloud Computing, 2, E264. http://doi.org/10.47363/JAICC/2023(2)E264

4. Chen, Z., Kang, Y., Li, L., Zhang, X., Zhang, H., Xu, H., ... & Lyu, M. R. (2020, November). Towards intelligent incident management: why we need it and how we make it. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 1487-1497).
5. Dai, H., Wang, Y., Kent, K. B., Zeng, L., & Xu, C. (2022). The state of the art of metadata managements in large-scale distributed file systems—scalability, performance and availability. *IEEE Transactions on Parallel and Distributed Systems*, *33*(12), 3850-3869.
6. Dhanagari, M. R. (2024). Scaling with MongoDB: Solutions for handling big data in real-time. *Journal of Computer Science and Technology Studies, 6*(5), 246-264. https://doi.org/10.32996/jcsts.2024.6.5.20
7. Goel, G., & Bhramhabhatt, R. (2024). Dual sourcing strategies. *International Journal of Science and Research Archive*, 13(2), 2155. https://doi.org/10.30574/ijsra.2024.13.2.2155
8. Gupta, D. (2024). *The Cloud Computing Journey: Design and deploy resilient and secure multi-cloud systems with practical guidance*. Packt Publishing Ltd.
9. Kaluvakuri, V. P. K. (2023). Ai-powered continuous deployment: Achieving zero downtime and faster releases. *Available at SSRN 4927198*.
10. Kaul, D. (2020). Ai-driven fault detection and self-healing mechanisms in microservices architectures for distributed cloud environments. *International Journal of Intelligent Automation and Computing*, *3*(7), 1-20.
11. Kesa, D. M. (2023). Ensuring resilience: Integrating IT disaster recovery planning and business continuity for sustainable information technology operations. *World Journal of Advanced Research and Reviews*, *18*(3), 970-992.
12. Konneru, N. M. K. (2021). Integrating security into CI/CD pipelines: A DevSecOps approach with SAST, DAST, and SCA tools. *International Journal of Science and Research Archive*. Retrieved from https://ijsra.net/content/role-notification-scheduling-improving-patient
13. Koop, J. (2020). Automated Jira Data Analysis for Optimised Project Supervision and Delay Detection.
14. Liu, P., Zhang, R., Yin, Z., & Li, Z. (2021). Human errors and human reliability. *Handbook of human factors and ergonomics*, 514-572.
15. Nyati, S. (2018). Revolutionizing LTL carrier operations: A comprehensive analysis of an algorithm-driven pickup and delivery dispatching solution. International Journal of Science and Research (IJSR), 7(2), 1659-1666. Retrieved from https://www.ijsr.net/getabstract.php?paperid=SR24203183637
16. Oh, S., Shin, H., & Kim, J. (2022). A Survey on Microservices Use Cases for AI based Application on Hybrid Cloud. *The Journal of Contents Computing*, *4*(1), 439-448.
17. Paleyes, A., Urma, R. G., & Lawrence, N. D. (2022). Challenges in deploying machine learning: a survey of case studies. *ACM computing surveys*, *55*(6), 1-29.
18. Qiu, H. S., Vasilescu, B., Kästner, C., Egelman, C., Jaspan, C., & Murphy-Hill, E. (2022, May). Detecting interpersonal conflict in issues and code review: cross pollinating open-and closed-source approaches. In *Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society* (pp. 41-55).
19. Raju, R. K. (2017). Dynamic memory inference network for natural language inference. International Journal of Science and Research (IJSR), 6(2). https://www.ijsr.net/archive/v6i2/SR24926091431.pdf
20. Riley-Tillman, T. C., Burns, M. K., & Kilgus, S. (2020). *Evaluating educational interventions: Single-case design for measuring response to intervention*. Guilford Publications.
21. Samala, S. (2024). AI-Driven Jira Automation: Using Machine Learning to Optimize Sprint Planning and Incident Resolution. *Frontiers in Emerging Artificial Intelligence and Machine Learning*, *1*(01), 44-65.
22. Sardana, J. (2022). The role of notification scheduling in improving patient outcomes. *International Journal of Science and Research Archive*. Retrieved from https://ijsra.net/content/role-notification-scheduling-improving-patient
23. Shah, M. B. (2024). *UNDERSTANDING THE CHALLENGES OF REPRODUCING DEEP LEARNING BUGS* (Doctoral dissertation, Dalhousie University Halifax).
24. Shankar, S., Garcia, R., Hellerstein, J. M., & Parameswaran, A. G. (2022). Operationalizing machine learning: An interview study. *arXiv preprint arXiv:2209.09125*.
25. Shaw, K. (2020). *Prioritizing Information Technology Infrastructure Library (ITIL) Implementations and Identifying Critical Success Factors to Improve the Probability of Success* (Doctoral dissertation, Capella University).
26. Sillito, J., & Kutomi, E. (2020, September). Failures and fixes: A study of software system incident response. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 185-195). IEEE.

27. Singh, V. (2024). AI-powered assistive technologies for people with disabilities: Developing AI solutions that aid individuals with various disabilities in daily tasks. *University of California, San Diego, California, USA. IJISAE*. https://doi.org/10.9734/jerr/2025/v27i21410
28. Tamanampudi, V. M. (2024). AI-Enhanced Continuous Integration and Continuous Deployment Pipelines: Leveraging Machine Learning Models for Predictive Failure Detection, Automated Rollbacks, and Adaptive Deployment Strategies in Agile Software Development. *Distributed Learning and Broad Applications in Scientific Research*, *10*, 56-96.
29. Tsaliki, K. C. (2024). AI for Resilient Infrastructure in Cloud: Proactive Identification and Resolution of System Downtimes. *International Research Journal of Engineering and Technology (IRJET)*, *11*(08).
30. Ugwueze, V. U., & Chukwunweike, J. N. (2024). Continuous integration and deployment strategies for streamlined DevOps in software engineering and application delivery. *Int J Comput Appl Technol Res*, *14*(1), 1-24.
31. Walsh, B. (2022). AI Best Practice and DataOps. In *Productionizing AI: How to Deliver AI B2B Solutions with Cloud and Python* (pp. 41-74). Berkeley, CA: Apress.
32. Wong, B., & McCann, J. A. (2021). Failure detection methods for pipeline networks: From acoustic sensing to cyber-physical systems. *Sensors*, *21*(15), 4959.