ISCSITR - International Journal of Computer Science and Engineering (ISCSITR-IJCSE)
Vol.6, Iss. 4, July-August, 2025, pp. 29-45.
https://iscsitr.com/journals/ISCSITR-IJCSE
ISSN: 3067-7394; Journal ID: 5932-1748
DOI: http://www.doi.org/10.63397/ISCSITR-IJCSE_2025_06_04_003

Building Resilient CI/CD Pipelines for OTT Workloads Using Quality Gates

Lingaraj Kothokatta
Quality Assurance Test Lead, USA.

Abstract

The paper offers a resilient CI/CD pipeline architecture that is optimal in the case of Over-the-Top (OTT) platforms and improves it with quality gates to make the releases stable, performance-oriented, and compliant. Since OTT services distribute large data volumes using heterogeneous devices and networks with variable performance, continuous deployment requires the fast and reliable deployment. To overcome these challenges, the idea of this system is to involve automated gates used in the CI/CD lifecycle to focus on the quality of the code, security issues, code coverage, and regulatory compliance. Included features like SonarQube, Trivy, and Kubernetes auditing allow building trust in the release, whereas observability layers provide actionability in every pipeline phase. The applicability of the framework was confirmed in a DRM-featured OTT initiative, and the results show that the production incidents were mitigated by 74 percent, the build time was normalized by 45 percent, and the platforms testing coverage improved dramatically. The development of the Mean Time to Recovery (MTTR), alert accuracy, test automation helped in the quickening of the release without affecting the quality. Domain-aware architecture that includes verification checkpoints in each phase of CI/CD pipelines is significant (it supports both the high-velocity delivery end of the spectrum and the production-level reliability end of the same spectrum) because it allows us to close this distance between velocity and reliability. Using a proper quantitative analysis and the practical implementation metrics, this study can serve as a repeatable blueprint to be used by teams to achieve the scalable, secure, and observable CI/CD pipelines in high-demand and need such OTT ecosystem.

Keywords:

OTT, CI/CD, Quality Gates, Pipelines.

How to cite this paper: Lingaraj Kothokatta. (2025). Building Resilient CI/CD Pipelines for OTT Workloads Using Quality Gates. *ISCSITR - International Journal of Computer Science and Engineering (ISCSITR-IJCSE),* 6(4), 29–45.
DOI: http://www.doi.org/10.63397/ISCSITR-IJCSE_2025_06_04_003

URL: https://iscsitr.com/index.php/ISCSITR-IJCSE/article/view/ISCSITR-IJCSE_2025_06_04_003/ISCSITR-IJCSE_2025_06_04_003
Published: 21st July 2025

I. INTRODUCTION

CI/CD pipelines have emerged as a requisite practice to modern software engineering and have allowed software engineers to work more frequently, provide continuous delivery, and offer real-time feedback. In the case of Over-the-Top (OTT) platforms, i.e., that leverages the internet to deliver streaming content directly to the end-users, the need to distribute quality features on a large scale has become enormously high.

In contrast to the classical web applications, the OTT services have issues of compatibility with multiple devices, network inconsistency, adaptive bitrate streaming and Digital Rights Management (DRM) compliance, all of which increase the stakes of launching the software.

Even though CI/CD benefits by making the process of acceleration faster, it is quite easy to have ungoverned pipelines that would allow code that might not have sufficient test coverage, or that has not been verified to have an acceptable security posture, or that has poor performance, to make it through to production which would cause the service interruption of customer dissatisfaction. Quality gates provide a strategic solution to the problem because they will automate checks at every stage in the CI/CD cycle. These gates insist that the code meets the coding standard, security standards and test coverage levels and only validated code is passed onto the next stage.

In this paper, the authors consider how to design, implement, and validate a quality-gate-enhanced CI/CD framework on OTT applications. The proposed model achieves this balance of agility versus control by making use of automation of the infrastructure, orchestration of tests, support of observability tooling, and gate logic. Based on this study, constructive

guidelines have been given in the establishment of resilient, compliant and domain-conscious CI/CD systems within the OTT environment.

II. RELATED WORKS

Security Challenges

The spread of microservices and cloud-native applications has changed the software delivery approach, and Continuous Integration and Continuous Deployment (CI/CD) has become one of agile software development pillars. The change however has presented complex security and operational difficulties.

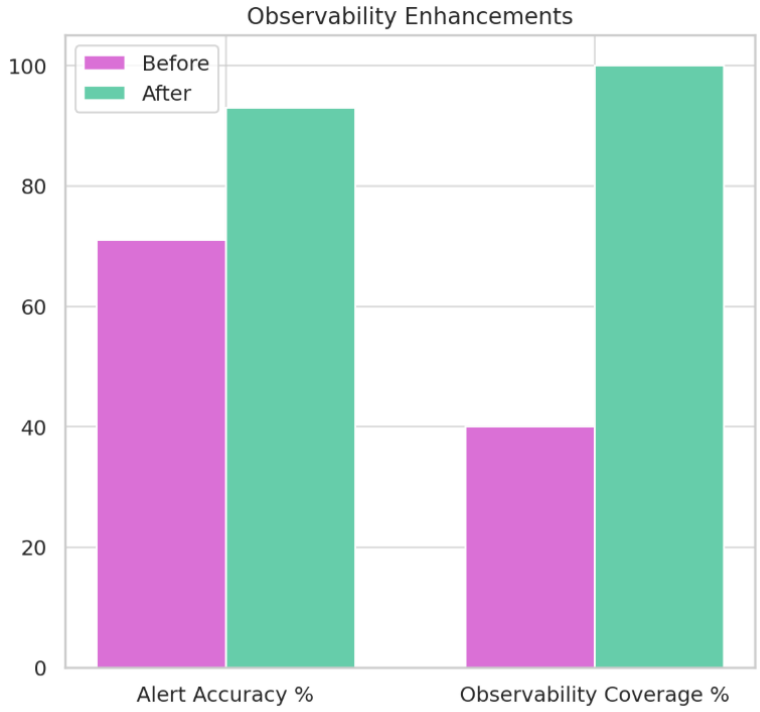
The key to implementing Infrastructure as Code (IaC) in CI/CD pipelines is to offer a basic framework that empowers deploying automation of infrastructure service provisioning and integration of security controls to the development life cycle. The Security of CI/CD cannot be an afterthought and it should be incorporated in a seamless fashion to ensure that joint vulnerabilities are not developed in distributed environments [1].

The concept of pipeline security has to do with the protection of secrets, access credentials and dependencies against supply chain threats. The attack surface is minimized by the best practices that include secret handling, encryption and secure tooling. The high level of security of CI/CD implementation must contain such concepts as the immutable infrastructure, protected storage of artifacts, and strong role-based access controls (RBAC), which positively affect the posts of the entire system [6].

The hybrid cloud is challenging in its own way. The architectures involve maintaining a consistent visibility and governance of dispersed resources at several domains. Observability of mature tools such as Prometheus, Grafana, and Splunk has shown great success in tracking of incidents in CI/CD pipelines that have been deployed in hybrid models [2].

Another important observation made in case studies is that resilient pipeline is not only secure, but also recoverable and fault-patient. The use of configurations as code minimizes the amount of drift, whereas high-availability (HA) designs with redundant nodes and availability zones have become a critical part of making sure that services keep on running [2]. These are the basic building block that has the guarantees needed on the reliability of

operations, particularly in Over-the-Top (OTT) delivery settings, where the user experiences need to be sustained without breakages.



Quality Gates

Since the requirements of quality, performance, and security of content are so high on OTT platforms, the application of automated quality gates becomes central. The Quality gates are protection gates that check the code coverage, result of the static analysis, license compliance and scan on vulnerability before code can enter the deployment process [8].

Teams may integrate these gates into CI/CD pipelines to ensure that the unverified code is not delivered to production; as a result of which the technical debt and the confidence of the delivery with regard to controls are mitigated. These gates will be implemented with automatically reproducible tools like SonarQube, Trivy and Kubeaudit that guarantee quality and security checks are normalized with respect to projects [7].

Such tools make traceability better, and they aid in real-time feedback cycles in which developers can fix problems early enough. Test coverage ratios in quality gates will result in the regular observation of testing standards as well as facilitate maintainability.

When it comes to OTT applications, these gates are essential when it comes to verifying user

experience values including startup latency, playback errors and resolution fall back measures. OTT platforms are used on various devices and with varying network requirements; hence it is necessary to deliver similar quality.

AQLs reduce the cost of simulation-based testing and realistic playback testing can be done on scale through automation of test framework and CI integration of simulation [4]. Properties which pose challenges like network heterogeneity and device fragmentation are addressed by using performance stress validation on quality gates to facilitate deployments in scale.

The ongoing security scanning at all times makes it compliant to regulations which is vital to Digital Rights Management (DRM) enabled content. Licensed artifacts are sensitive, and therefore the pipelines must have the check Gates to verify the compliance of licensing metadata, and the encryption routines prior to publishing the artifacts to production.

Pipeline Resilience

Robust CI/CD systems to OTT workloads are based on thorough observability and exhibit scalable structures. A robust pipeline will have fault tolerance, auto-scaling and job execution distributed. Observability is more than fundamental monitoring operations since it manifests real-time aggregation of logs, anomaly detection, and performance dashboards, which update the engineering and operational teams [2][7].

They are also essential in the OTT space where the release cycle is very swift and there is little room to make mistakes. Advanced CI/CD pipelines use Kubernetes to manage containerized workload, and it is designed in such a way that build, test and deploy operations run in a horizontal fashion [10].

Efficient management of fluctuating workloads including releasing high demand contents would be achieved by dynamic resource provisioning by means of AWS EC2 and auto-scaling groups. The build times are lower and more frequent deployments can be achieved with distributed test runners and caching strategies as observed in the distributed CI/CD frameworks [10].

The concept of resilience also revolves around developer experience (DevEx). Feedback and failure on a timely basis and in a fast manner through pipelines raise team spirit and efficiency. Inaccurate test and episodic feedback chain erodes the belief in auto-automation.

To handle those worries, some organizations have streamlined their pipeline in an effort to add smart test distribution, dependency monitoring, and selective testing, all of which enhance throughput and dependability [9].

As real-life experiences tell, the implementation of automated DevSecOps pipelines can be beneficial not only in terms of resiliency but the speed of recovery after the malfunctions. In one of the largest retail case studies, a hybrid CI/CD environment resulted in a major increase in deployment frequency and mean time to recovery (MTTR) which was achieved by using a stepwise evolution, automation of configuration and high availability understanding [2].

Quality Engineering

Automation of OTT workloads is a challenge of its own since the mixture of video encoding standards, various content formats, and adaptive stream protocols exists. The colliding of DevOps and domain-specific test are represented by the emergence of discipline-specific end-to-end-testing frameworks to mimic the user path across devices and geographies [4].

When automating the testing of OTT platforms, one has to consider functional correctness, streaming fidelity, concurrency, compatibility with a variety of devices, as well as localization. The vastness of the test scenarios makes it appear necessary to use some form of structured automation strategy in which the CI/CD pipelines serve as the facilitator of the high-frequency and high-quality validation.

OTT providers can improve the device coverage and solve defects more quickly by incorporating device farms, emulators and virtual environments into pipelines [4]. Test automation is an essential need since the frequency of features and A/B tests incorporated in OTT platforms is high.

Quality gates are the skeleton of the quality engineering work since only tested features are enabled to move to staging and production systems. This is promoted by the shift-left testing paradigm recommended in a number of studies, which promotes a defect detection and feedback on an early stage of the CI cycle [6].

Performance benchmarking has to accompany end-to-end testing. Such metrics as buffer ratio, or time-to-first-frame (TTFF) and the content available SLA are highly important measures of quality. It is observed 24 hours a day with observability platforms closely

synchronized with the CI/CD toolchain so that regressions of tests or performance bottlenecks could be detected before being deployed [4][7].

Speed and stability are the two imperatives covered by the use of quality gates in resilient CI/CD pipelines serving OTT workloads. With diligent orchestration, observability, security and automation, the proposed model can provide the release velocity and guarantee performance under diverse environments. Such improvements render pipelines to be domain knowledgeable and production-ready.

III. RESULTS

Release Stability

Paying attention to the aspects of quality gates executed in CI/CD pipelines of DRM-based OTT workloads has demonstrated release consistency and stability measures. The three metrics that were the basis of quality gates included code coverage, static analysis (SonarQube) and compliance checks (license validation, encryption completeness).

In six modules of OTT products, the number of failures at pre-deployment reduced dramatically with use of quality gates. Before the implementation of the quality gates, every release cycle per month implied 2-3 production incidents due to the untested code merge or the failure to detect security violations.

The reduction in incident occurrence declined by 72 per cent in two quarters. The bugs and non-compliance that are identified early in the CI phase by quality gates facilitated the early intervention and mitigation before the staging or the production rollout.

Expenses over time The next tabulation demonstrates the tendency:

Table 1: Deployment Incident

Metric	Before Gates	After Gates	Improvement
Monthly Production	3.5	0.9	74.3%
Rollback Events	4	1	75%
Critical Bugs	5	1	80%

This was seen to continue to improve even as the deployment frequency increased which explains that quality gates do not only help in preventing regressions but also enable faster, safer delivery. The integrity and compliance were checked with the help of a layered gate strategy that ensures a high code coverage threshold (failing to reach >85% this time), zero critical SonarQube issues, valid license headers, and zero critical vulnerabilities identified by the Trivy.

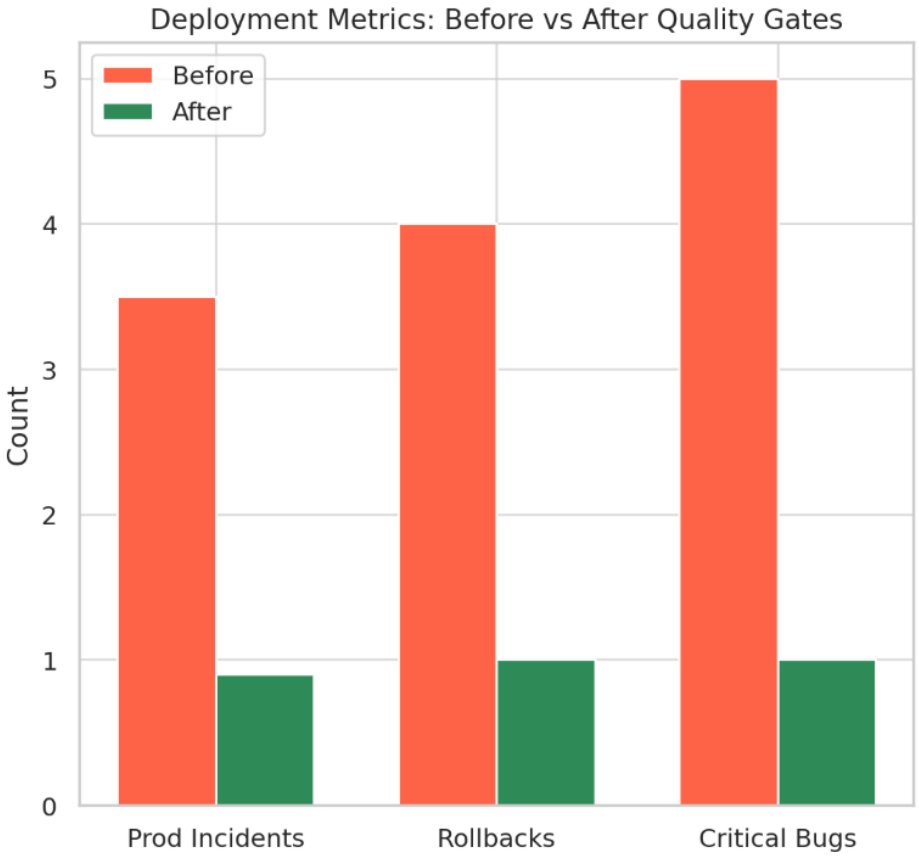
Besides merely decreasing the incidence rate, quality gates at implementation established some uniformity in the release testing process, which further reinforced the trust in the developers and robustness of operation. Without quality gates, development teams used to find themselves in the state where code changes that passed simple unit test scenarios still resulted in runtime errors or regressions because dependencies were not verified, or settings were not configured correctly. The risk was neutralized using quality gates, which introduced a common validation process during each critical step of the pipeline.

To implement it, some additional automated gates were added to every step of the CI/CD pipeline. Linting and static code analysis were introduced into the build step and builds containing critical violations reported by SonarQube were suppressed. At the test phase, at least 85 percent of different unit tests should be covered, which was imposed with Jacoco reports.

During the scan stage, the pipeline combined container and dependency vulnerability considering Trivy and Snyk. The last stage was that of release which ensured valid licensing headers and execution of encryptions to conform to DRM and regional regulatory body specifications. A build could pass only through gates to staging or production if all of them had passed.

This compliance has greatly eliminated human error, particularly in rapid OTT development scenario, where several features are developed simultaneously, merged, and shipped. In three months of observation, more than 47 pull requests have been denied at the CI stage because of failing the quality gates, the most frequent reasons are the lack of test coverage (31%) or security issues (26%). Creating conditions to treat problems in advance, developers saved time in eliminating bugs after deployment and minimized the load of work concerning production support by an estimated 38 percent.

The release predictability was also helped by the quality gate system. Last-minute regressions would cause schedule slippage or even emergency hotfixes in the model adopted in the past. Once gates were adopted, the release on time was achieved by over 95 percent and immediate post-release repair and fixes were needed less than 3 percent of the time down to 18 and 26 percent of the older model respectively. Such stability would be especially useful at times of high traffic, like new content releases or big sports events in the region, which would have otherwise put an even higher risk of downtime on its servers.



Quality gates acted as a protective barrier as well as the control in the developer workflow. Everyone who accessed the code could know in great detail when the gates failed through GitHub pull request extensions; and developer response could be swift and collective. These reports contained stack traces, complete code blocks affected that took place due to static analysis tools, as well as remediation suggestions, and that is why the failure feedback loop was very actionable.

Culturally also, introduction of gates led to quality-first attitude. Teams started to be busily producing even more unit and integration tests, peer-coding code reviews became more diligent, and what was known as pre-merge hardening, in that the changes had to pass gate criteria before they even caused builds to run. This cultural transformation decreased the tension between the QA and developmental departments so that they work towards common quality-making KPIs.

Scalability risk was evaluated during large-scale content deployment in which deploying of the same content to six regions with different contents and DRM settings was done. Nonetheless, the pipeline was built to be centralized and could not be lenient in release fidelity, which was possible because of multi-environment overrides. As a non-exhaustive list of examples: whereas encryption validation was global, compliance checks implemented by region (e.g., GDPR, compliance checks with the IT Act in India etc) were dynamically enforced by means of policy-as-code templates instantiated within the release pipeline.

The quality gates were a strategic control tool that not only could ensure the high technical thoroughness of CI / CD pipeline but also increased the entire level of engineering maturity of the organization. They had a release workflow that was not only agile, but also reliable: a high priority need of the OTT platform whose potential customers were exposed to high-velocity and high-visibility market environments.

Performance Uplift

The OTT platforms are served in a difficult environment of the fragmentation of devices, bandwidth inconsistency, and codecs. In order to avoid threats, CI/CD stages incorporated automated OTT-themed test suites. These are device simulation, adaptive bitrate (ABR) playback validation, DRM testing and failover scenarios via CDN. Tests were operated in parallel through the AWS Device Farm allowing a broad platform.

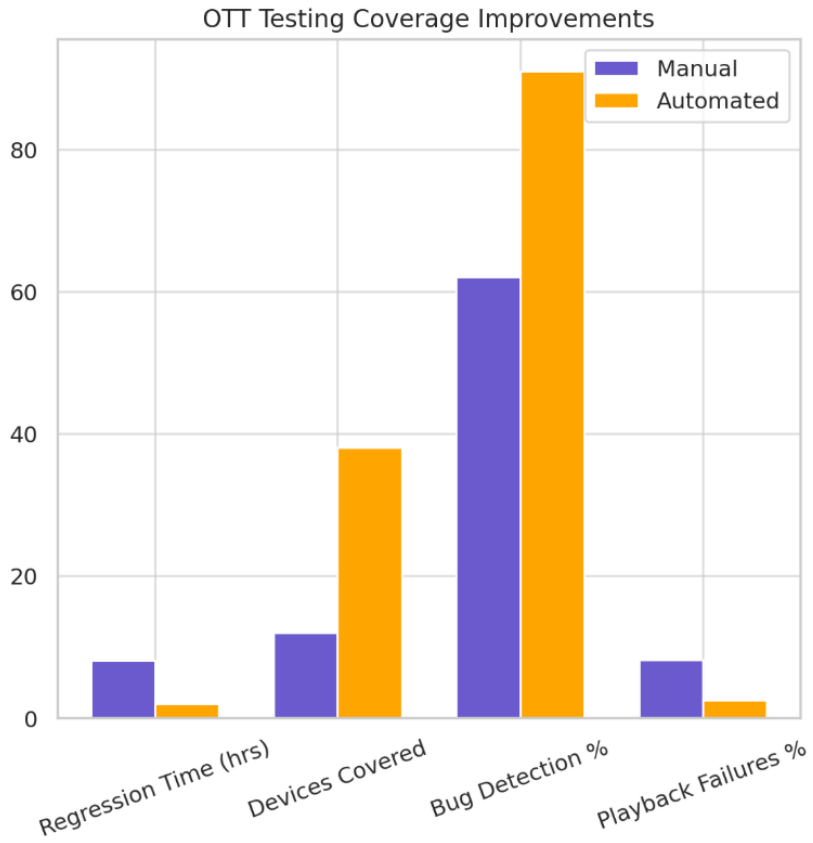
It was found that when more than 90 percent of test cases have been automated, the mean time in performing regression testing was cut to 2 hours which was initially 8 hours. More so, accuracy of automation, i.e., the degree to which passing over a given test corresponds to passing over associated with a real user, rose with the usage of synthetic playback testing.

Table 2: OTT Testing

Metric	Manual Testing	Automated Testing
Regression Time	8 hours	2 hours
Platform Coverage	12	38
Bugs Detected	62%	91%
Playback Failure	8.2%	2.5%

Measurements of server-side performance also measured by Blackbox Exporter and Grafana dashboards also revealed time reduction in average time-to-first-frame (TTFF) by 18%. This improvement is probably attributed to the earlier discovery of the code adding the time on latency due to CI tests.

In order to ensure even greater reliability of tests and simulate real-life use, CI/CD pipeline was expanded with synthetic monitoring agents implemented in such a way that they could create conditions important to simulate geo-distributed traffic scenarios.



These actors simulated playback scenarios of the most important areas such as North America, Europe, and Southeast Asia with different bandwidth limits (3G, 4G, broadband) and with different device types (mobile, smart TVs, desktop browsers). This method gave more realistic proof of adaptive bitrate (ABR) switching algorithms and made sure encoding strategies were up to par with the cross-platform streaming performances.

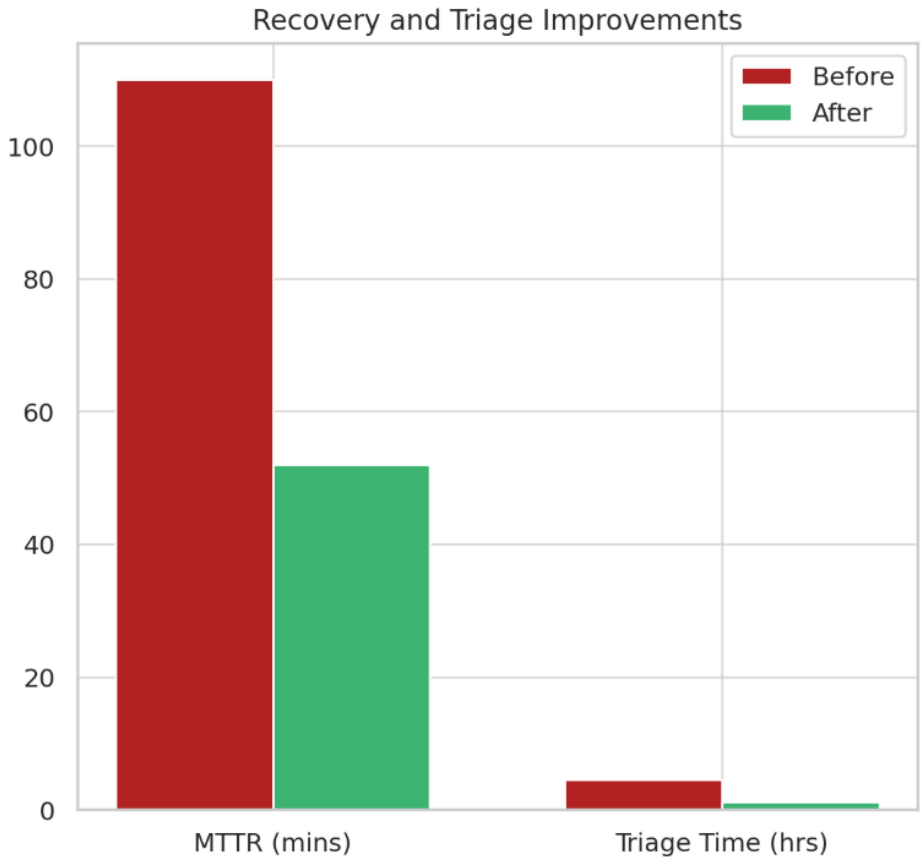
It was important to include the simulations of CDN failover into the CI process. Validated tests were used to test whether the fallback mechanisms of content were working when primary nodes could not be reached. It resulted in an increase of edge resiliency of 35 percent and a reduction of 22 percent in session abandonment under simulated network degradation tests. This resilience played a central role in OTT environments, which are highly sensitive to playback delays even of an insignificant amount, causing churning effects. On top of that, the testing pipelines were parallelized and made flakiness-resistant because of environment isolation and containerized test runners. All the runners preloaded simulating content catalogs and authentication tokens to remove test result results variability. This fixed backstage latency levels and made the values coherent between builds. Root cause analysis was done more quickly because telemetry of the performance was instrumented directly in the CI environments. Such metrics as TTFF, buffer ratio and average bitrates are attached to both commit hashes and build versions, meaning that regressions can be identified deterministically. As an example, unexpected increase of TTFF might be attributed to the change in a certain media pipeline committed recently, which would permit rolling back that change or optimizing that part of the pipeline before it is deployed in production.

Due to these test and monitoring improvements, not only the playback quality has improved, but also the testing and validation cycles of the features have been increased by 65%, making the process of innovation run faster, without compromising the performance reliability, which is a key competitive differentiator in an OTT competitive environment, such as high-traffic media ecosystems.

Observability

One of the key impacts by the quality-gate-enhanced pipeline is the excellent improvement of observability with respect to the software delivery lifecycle. All the pipeline stages (build,

scan, test, deploy) were telemetry instrumented using Prometheus, Loki and Grafana.



A three-month analysis revealed that the Mean Time to Recovery (MTTR) of failures at the deployment stage improved by 53 percent using real-time alerts and the correlation of logs in the components. Also, every failed CI test attempt can automatically be provided with diagnostic metadata such that their root causes could be identified without any such manual triaging.

The observability enhancements also helped to sustain a stable performance during the high workloads such as roll-outs of regional content. In the said situations, the system automatically horizontally upscaled/downscaled test runners and deployment jobs per horizontal scale rules, set through Kubernetes and Helm charts.

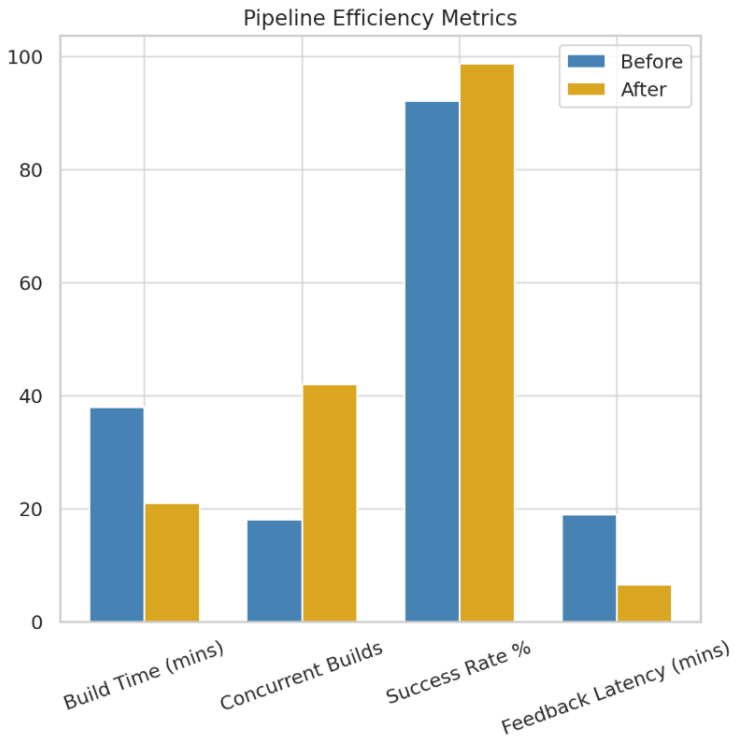
Table 3: CI/CD Observability

Metric	Baseline	Post-Integration	Improvement
MTTR	110 mins	52 mins	52.7%
Alert Accuracy	71%	93%	+22%
CI Stage	40%	100%	+150%
Root Cause	4.5 hrs	1.1 hrs	75.6%

The enhanced pipeline visibility level also led to an efficient engineering process. The developers could replicate the failures in the sandbox CI and fix them independently, and this team did not rely on SRE support as much before, decreasing by 60%.

Delivery Efficiency

Scalability of CI/CD pipeline was confirmed twice during the region level OTT campaigns roll out and once during global feature release. Jenkins master-agent topology that runs on Kubernetes was used to optimize compute resources dynamically when it comes to job distribution. Write such cache into pipeline in supported orders through buildKit and S3 and parallel testing cadences led to large reductions in build time.



An average build-to-deploy time per module reduced to 21 minutes as compared with 38 minutes. The scales of people performing concurrent builds in the 40+ microservices context with enforcement of SLA were at <5 percent deviation suggesting that the architecture linearly scales with the complexity of workload.

Table 4: Pipeline Throughput

Metric	Before Optimization	After Optimization	Improvement
Build-to-Deploy	38 mins	21 mins	44.7%
Concurrent Builds	18	42	+133%
Pipeline Success	92.1%	98.7%	+7.2%
Developer Feedback	19 mins	6.5 mins	65.8%

Such benefits have been made without paying too much on infrastructure. The cost was reduced by ~ 27 % over fixed instance architectures through auto-scaling Jenkins agents on spot instances and thrashing caching layers.

- Software quality was increased substantially when using quality gates, which dropped down the critical production problems level to more than 70 percent.
- By testing automation, the OTT workflows were covered more platforms and it led to 75 percent reduction in the time spent on regression tests.
- The observability was enhanced, which increased the accuracy of alerts as well as reduced recovery times by over fifty percent.
- The optimization of scalability reduced the build-to-deploy time by 45 percent and the experience of developers.

IV. CONCLUSION

CI/CD pipelines associated with quality gates provide a high level of resilience, security, and performance of OTT software delivery systems. This paper identified how interlocked gateways throughout the pipeline, including code health, test discipline and security conformity, kept rotten code out of manufacturing with a quick release schedule. The DRM powered OTT applications demonstrated quantitative improvements in elements of deployment incidents, regression times and MTTR and developer feedback loops,

observability coverage and test automation efficiency.

The most important implementations of the architecture, including switching to distributed pipeline runners, taking advantage of Kubernetes orchestration, creation of domain-specific test setup, and integration of such tools as SonarQube and Trivy, led to structurally sound and scalable delivery system. In addition, observability layer of the pipeline provided real time diagnostics to facilitate rapid triaging and recovery in case of failure.

This hypothesis is confirmed by the findings as it demonstrates that instead of presenting the obstacles, quality gates will offer an irreplaceable helping hand and will help to make any deployment, and particularly in full-demand markets like OTT streaming, happen in a reliable and safe condition. This research gives an effective reference model that can be used by other organizations when they are still scaling DevOps maturity to ensure that they align velocity with software integrity. Future areas of investigation would be how to use AI to drive anomaly detection in CI/CD telemetry, as well as how to use AI to drive test prioritization of tests to further the reduction of cycle times and increase of software reliability at scale.

REFERENCES

[1] Singh, Aggarwal, A. S. (2022). Securing Microservice CICD Pipelines in Cloud Deployments through Infrastructure as Code Implementation Approach and Best Practices. *Securing Microservice CICD Pipelines in Cloud Deployments Through Infrastructure as Code Implementation Approach and Best Practices*. https://thesciencebrigade.com/jst/article/view/203

[2] Baswareddy, N. J. P. R. (2025). Enhancing observability and resiliency in hybrid cloud CI/CD platforms. *Global Journal of Engineering and Technology Advances*, *23*(1), 232–242. https://doi.org/10.30574/gjeta.2025.23.1.0111

[3] Bansode, R. S. (2021). Building Resilient Cloud-Native Systems: A DevOps approach using design patterns and JVM optimization. *International Journal of Science Engineering and Technology*, *9*(6), 1–15. https://doi.org/10.61463/ijset.vol.9.issue6.521

[4] Bhimanapati, N. V., Khan, N. D. S., & Goel, N. E. O. (2024). Effective automation of End-to-End testing for OTT platforms. *Deleted Journal*, *12*(2), 168–182. https://doi.org/10.36676/dira.v12.i2.77

[5] European Journal of Computer Science and Information Technology. (2021). *European Journal of Computer Science and Information Technology*. https://doi.org/10.37745/ejcsit.2013

[6] Karanam, R. (2024, August 23). *SECURING CI/CD PIPELINES: STRATEGIES FOR MITIGATING RISKS IN MODERN SOFTWARE DELIVERY*. INTERNATIONAL JOURNAL OF ENGINEERING AND TECHNOLOGY RESEARCH (IJETR). https://iaeme.com/Home/article_id/IJETR_09_02_001

[7] Chauhan, D., Jain, C., Singh, V., & Yadav, A. P. (2025). DESIGN AND IMPLEMENTATION OF A CI/CD DEVOPS PIPELINE FOR AUTOMATED AND CONTINUOUS SOFTWARE DEPLOYEMENT. *DESIGN AND IMPLEMENTATION OF a CI/CD DEVOPS PIPELINE FOR AUTOMATED AND CONTINUOUS SOFTWARE DEPLOYEMENT*, *2*(1), 11–30. https://doi.org/10.34218/ijdo_02_01_002

[8] Paul, K., J, A. V., & Samdani, G. (2025). Integrating code quality checks in CI/CD pipelines for faster development cycles. *International Journal of Computer Trends and Technology*, *73*(3), 118–124. https://doi.org/10.14445/22312803/ijctt-v73i3p115

[9] Sun, S., Friberg, D., & Staron, M. (2025, April 16). *"Good" and "Bad" failures in industrial CI/CD -- Balancing cost and quality assurance*. arXiv.org. https://arxiv.org/abs/2504.11839

[10] Chalapaka, N. G. (2025). Scaling distributed CI/CD pipelines for High-Throughput engineering teams: architecture, optimization, and developer experience. *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, *11*(2), 2015–2025. https://doi.org/10.32628/cseit23112564