International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 6s, 2025
https://www.theaspd.com/ijes.php

Sustainable Optimization of the CI/DC Cycle through Artificial Intelligence: An Efficient and Green Approach to DevOps Practices

William Emmanuel Castillo Ortega
Universidad en Línea de México (UTEL)
Email: wcastilloortega@gmail.com
ORCID: 0009-0002-7012-9208

Summary
This article discusses the impact of artificial intelligence (AI) on optimizing the continuous integration and delivery (CI/CD) cycle, with an emphasis on sustainable and green practices in DevOps environments. As demands for software increase, so does the energy consumption of technology infrastructures. This study proposes an AI-based approach to improve operational efficiency and reduce the carbon footprint in DevOps environments, integrating predictive models, intelligent automation, and sustainability metrics. A quantitative methodology was applied through experimentation in continuous integration environments, evaluating energy consumption, performance and deployment time. The results show that the implementation of AI can reduce the energy consumption of the CI/DC cycle by up to 25%. This research contributes to the development of greener and more resilient software engineering.
Keywords: *DevOps, artificial intelligence, sustainability, CI/CD, energy efficiency, automation.*

INTRODUCTION

The acceleration of software development and delivery processes has been significantly driven by the adoption of DevOps methodologies, which seek to collaboratively integrate development and operations areas. In this context, the Continuous Integration and Continuous Delivery (CI/CD) cycle has become an essential pillar, allowing frequent, automated deployments with less margin for human error (Bass et al., 2022). However, this intensive automation entails extensive use of computational resources, including the constant consumption of CPU, memory, and cloud storage, which generates a significant environmental impact by increasing energy consumption and carbon footprint (Li et al., 2022).Faced with this problem, technological sustainability has become a central concern in contemporary software engineering. The need to adopt greener approaches has motivated the exploration of innovative tools, including artificial intelligence (AI), to mitigate the collateral effects of high dynamism in DevOps (Kumar et al., 2023). AI offers advanced automation, prediction, and decision-making capabilities that allow processes such as automated testing, code error detection, and efficient resource allocation to be optimized (Zhao et al., 2021). Thanks to machine learning algorithms and predictive models, failures can be anticipated before they occur and unnecessary executions that consume resources without providing real value can be reduced.In addition, the growing adoption of cloud environments and orchestrated containers—such as Kubernetes—has created new opportunities to apply AI in dynamic workload management and task scheduling based on energy and operational efficiency criteria (Wang et al., 2020). This convergence between AI and DevOps has given rise to approaches known as *AIOps* or cognitive DevOps, aimed at integrating intelligence into development and deployment flows to make them more resilient, adaptable, and sustainable (Nadeem et al., 2023).

Therefore, the present article aims to analyze how artificial intelligence can contribute to the sustainable optimization of the CI/CD cycle, not only from a technical efficiency perspective, but also from an ecological and responsible approach. This analysis is particularly relevant in the current context of corporate commitment to the Sustainable Development Goals (SDGs), where technology must be aligned with responsible practices in the use of digital resources.

THEORETICAL FRAMEWORK

2.1 DevOps and the CI/CD cycle

The DevOps philosophy promotes integration between software development and operations teams to improve the continuous delivery of value to the end user. This approach is focused on automation, collaboration, and constant monitoring of all phases of software development (Bass et al., 2022). The CI/CD cycle is one of the most representative DevOps practices, and is made up of two key elements:

- **Continuous Integration (CI):** Automates the compilation and testing of every code change.
- **Continuous Delivery (CD):** Automates the deployment of production-ready versions.

While this automation has improved the speed and quality of the software, it has also increased the consumption of computing resources, as many pipelines run multiple times a day, even for minor changes (Khan et al., 2021).

Table 1. Characteristics of the traditional CI/CD cycle in DevOps environments

Element	*Description*	*Energy Involvement*
Compilation	Repetitive and automated	High CPU usage
Unit Tests	Constantly executed with every change	Intensive memory usage
Deployment	It can occur multiple times per day	Power consumption in servers
Continuous monitoring	Active monitoring of logs and metrics	Network traffic and continuous disk usage

Source: Authors' elaboration based on Khan et al. (2021).

2.2 Artificial intelligence applied to software engineering

Artificial intelligence has shown a significant ability to transform software development processes, especially when combined with automation and adaptive learning. Some relevant applications in the context of DevOps include:

- **Log analysis and error diagnosis** (Zhao et al., 2021)
- **Prediction of failures in integration tests** (Wang et al., 2020)
- **Resource orchestration based on energy demand** (Kumar et al., 2023)

AI models can analyze large volumes of data generated during pipelines, recognize patterns, and make optimization decisions in real time. This capability allows you to reduce unnecessary executions, prioritize tasks, and automatically adjust the use of computational resources.

Table 2. AI Applications in DevOps Environments

Application of AI	**Associated technology**	**Key benefit**
Anomaly detection	Neural networks	Reduction of failures in production
Deployment optimization	Reinforcement learning	Lower energy consumption
Predictive Failure Analytics	Decision Trees, SVM	Increased quality in IQ tests
Decision automation	Heuristic algorithms	Dynamic Resource Orchestration

Source: Adapted from Zhao et al. (2021); Kumar et al. (2023).

2.3 Green DevOps: Sustainability and Efficiency

Digital sustainability has become a priority for organizations seeking to align their technological infrastructure with global environmental commitments, such as the SDGs. The discipline known as *Green DevOps* promotes the rational use of digital resources, minimizing the ecological footprint without compromising the agility of development (Nadeem et al., 2023).

Recent studies point out that a single pipeline executed unnecessarily can consume up to 0.5 kWh of energy, especially when it involves complex integration tests or multiple active containers (Li et al., 2022). Thus, the combination of DevOps practices with AI allows a balance to be achieved between speed, quality and sustainability.

Table 3. Comparison between traditional DevOps and sustainable DevOps with AI

FEATURE	TRADITIONAL DEVOPS	SUSTAINABLE DEVOPS WITH AI
NUMBER OF DAILY RUNS	10–20	6–10 (with smart prediction)
AVERAGE EXECUTION TIME	20–30 minutes	12–18 minutes
ESTIMATED ENERGY CONSUMPTION	Alto (>5 kWh/día por pipeline)	Bajo (<4 kWh/día por pipeline)
LEVEL OF AUTOMATION	Middle	High (data-driven decisions)
CARBON FOOTPRINT	High	Reduced

Source: Authors' elaboration with data from Li et al. (2022); Kumar et al. (2023).

METHODOLOGY

This study is part of a **quantitative-experimental applied research**, focused on evaluating the impact of the implementation of artificial intelligence on the energy and operational efficiency of the CI/DC cycle in DevOps environments. To this end, two parallel test environments were designed: one with a standard configuration and the other with AI integration at critical stages of the pipeline.

3.1 Experiment design

Two experimental groups were established:

- **Control group**: conventional CI/DC environment without AI intervention.
- **Experimental group**: CI/CD environment with AI applied to log analysis, failure prediction and dynamic resource management.

Both pipelines were built using tools widely adopted in the DevOps ecosystem: Jenkins for orchestration, Docker for containerization, and Kubernetes for cluster management. The infrastructure was deployed in a hybrid cloud based on Amazon Web Services (AWS) EC2 instances, with energy consumption monitoring using **the Cloud Carbon Footprint API.**

Table 4. Technical configuration of test environments

COMPONENT	CONTROL GROUP	EXPERIMENTAL GROUP (WITH AI)
ORCHESTRATOR	Jenkins	Jenkins + plugins de IA (DeepChecks)
CONTAINERS	Docker	Docker
CLUSTER MANAGEMENT	Kubernetes	Kubernetes with load prediction
MONITORING	Prometheus	Prometheus + detection algorithm
INFRASTRUCTURE	AWS EC2 t3.medium	AWS EC2 t3.medium

Source: Authors, adapted from Wang et al. (2020).

3.2 Implementing AI in CI/CD

In the experimental environment, three types of artificial intelligence solutions were applied:

1. **Log analysis with supervised AI**: classification models were trained to detect failure patterns in previous stages of the pipeline, avoiding unnecessary executions (Zhao et al., 2021).
2. **Intelligent resource allocation**: Through reinforcement learning, the system dynamically adjusted the CPU/RAM allocation according to the complexity of each build (Kumar et al., 2023).
3. **Test automation based on historical relevance**: The algorithm prioritized test cases with a higher probability of failure, thus reducing total execution.

Table 5. AI algorithms used in the experimental environment

DevOps task	*Implemented algorithm*	*Main objective*
Failure prediction	Random *Forest*	Avoid repetitive executions
Resource orchestration	Deep reinforcement learning	Minimize CPU/RAM consumption
Test prioritization	k-NN (Nearest Neighbors)	Reduce the duration of the testing stage

Source: Adapted from Zhao et al. (2021); Kumar et al. (2023).

3.3 Observed variables

The following dependent variables and evaluation metrics were defined for each CI/CD cycle executed:

- **Total execution time (min).**
- **Energy consumption (kWh).**
- **Execution failures**.
- **Average CPU and RAM usage** (%).
- **Number of skipped steps thanks to prediction**.

These metrics were collected over a period of 21 consecutive days, executing equivalent integration and deployment tasks in both environments.

Table 6. Study-dependent variables

Variable	**Guy**	**Harvesting tool**
Runtime	Continuous quantitative	Jenkins Logs
Energy consumption	Continuous quantitative	Cloud Carbon Footprint API
CPU/RAM Usage Percentage	Continuous quantitative	Prometheus Metrics
Number of faults detected	Discrete quantitative	Reportes de testing y builds

3.4 Experimental procedure

1. Two equivalent pipelines were configured in terms of codebase, libraries and test environment.
2. Integration and deployment tasks were executed synchronously in both groups (minimum 3 daily executions).
3. All variables were measured at the end of each cycle.
4. A descriptive and comparative statistical analysis was used to interpret the data.
5. A Student's **t-test was used** to verify whether the differences between the two groups were statistically significant ($p < 0.05$).

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 6s, 2025
https://www.theaspd.com/ijes.php

RESULTS

After an experimental period of 21 days, with more than 120 executions per environment, quantitative data was collected and analyzed that allowed the performance of the traditional CI/CD pipeline to be compared to its AI-optimized version. The results are grouped into three key dimensions: time efficiency, energy efficiency, and operational performance.

4.1 Time efficiency

The average time per integration and deployment cycle was significantly lower in the experimental group. While the traditional pipeline had an average of 22.8 minutes per execution, the AI pipeline managed to reduce that time to 16.4 minutes, which represents a **28.1**% improvement in time efficiency.

Table 7. Average Execution Time Comparison

Group	Minimum (min)	Maximum (min)	Average (min)	Standard deviation
Traditional CI/CD	18.7	27.4	22.8	2.89
AI-powered CI/CD	13.2	19.3	16.4	1.88

Source: Authors' elaboration based on Jenkins logs and Prometheus chronometers.

This improvement is mainly attributed to the intelligent prioritization of tests and the automatic omission of redundant stages through historical analysis (Zhao et al., 2021).

4.2 Energy efficiency

Energy consumption was recorded using the Cloud Carbon Footprint API, which translates CPU, memory, and storage usage into kilowatt-hours (kWh). The data shows that the AI pipeline consumed on average **3.9 kWh per d**
compared to **5.6 kWh** in the traditional pipeline, showing a **30.3% reduction in daily energy consumption**.

Table 8. Cumulative energy consumption in 21 days

Group	*Total consumption (kWh)*	*Average daily consumption (kWh)*
Traditional CI/CD	117.6	5.6
AI-powered CI/CD	81.9	3.9

Source: Authors' elaboration based on Cloud Carbon Footprint metrics.

These results coincide with previous research showing how artificial intelligence can be an effective tool to reduce the ecological footprint in technological environments (Kumar et al., 2023; Li et al., 2022).

4.3 Reducing errors and compilation failures

A significant decrease in the number of failed executions was also observed. The traditional group had an average of 6.4 weekly failures, compared to only 2.3 weekly failures in the optimized pipeline. The implementation of predictive models made it possible to anticipate frequent integration errors and avoid failed builds (Wang et al., 2020).

Table 9. Average number of errors detected per week

Error Type	*Traditional CI/CD*	*AI-powered CI/CD*
Unit Test Failures	3.1	1.2
Integration failures	2.0	0.6
Deployment failures	1.3	0.5
Weekly total	**6.4**	**2.3**

Source: Jenkins Bug Log and GitHub Actions.

4.4 Use of computational resources

Average CPU and RAM usage levels during executions were also measured. The pipeline with AI demonstrated a more efficient use of resources, due to dynamic orchestration based on demand and priority.

Table 10. Comparing Average Resource Usage

Resource	*Traditional CI/CD*	*AI-powered CI/CD*
CPU (%)	74.2	58.7
RAM (%)	81.9	65.4

Source: Metrics collected from Prometheus and Grafana dashboards.

4.5 Statistical analysis

To validate the significance of the results, a Student's t-test was applied for independent samples. The results showed statistically significant differences ($p < 0.01$) in the three main variables:

- Runtime
- Energy consumption
- Number of errors

These findings support the hypothesis that integrating AI into DevOps pipelines not only improves operational efficiency, but also has a measurable impact on process sustainability (Nadeem et al., 2023).

CONCLUSIONS

The results obtained in this research clearly and quantifiably show that the integration of artificial intelligence techniques in CI/CD cycles within DevOps environments represents an effective strategy not only to optimize technical processes, but also to move towards a more sustainable software development model. This multi-dimensional optimization simultaneously addresses aspects of performance, energy efficiency, and failure reduction, which is critical in modern organizations oriented towards continuous value delivery.

First, it was found that the use of AI algorithms, such as reinforcement learning, random forests, and k-NN, significantly reduces the average execution time of pipelines, by up to 28.1%. This directly translates into higher productivity and more agile delivery cycles, which is consistent with the findings of Wang et al. (2020), who state that artificial intelligence improves the adaptive and reactive capacity of CI/CD architectures.

Second, reducing energy consumption by more than 30% demonstrates that it is possible to implement environmentally responsible DevOps practices. This result supports the idea of a "green DevOps", where automation not only seeks speed, but also sustainability. Studies such as those by Li et al. (2022) and Kumar et al. (2023) have pointed out that the efficient use of digital infrastructure must be part of an organizational strategy oriented towards the Sustainable Development Goals (SDGs), especially with regard to responsible energy consumption and climate action.Another notable

finding was the decrease in errors in the test, integration and deployment stages. AI's ability to anticipate and prevent errors by analyzing historical logs and predicting failure patterns is invaluable in reducing rework, latency, and wear and tear on computational resources (Zhao et al., 2021). This automated prevention improves the quality of the software delivered, while reducing operational costs.Additionally, the intelligent orchestration of computational resources (CPU, RAM) contributes to minimizing excessive or poorly distributed use of infrastructure, promoting the elasticity and scalability of the DevOps environment, without increasing the ecological impact (Nadeem et al., 2023). In contexts where companies use cloud platforms with consumption-based billing, this optimization also translates into tangible economic benefits.

From an organizational perspective, the findings of this study suggest that the adoption of AI in CI/CD pipelines should be accompanied by a technology governance strategy that considers not only performance indicators, but also sustainability metrics. This implies that technology leaders must begin to measure, report and manage the environmental impact of their continuous development processes.In short, artificial intelligence not only transforms DevOps flows from an operational perspective, but also allows building a greener, more **efficient, and more resilient** software development ecosystem. Its implementation represents a strategic opportunity for organizations seeking competitiveness and environmental responsibility in the digital age.

References

- Bass, L., Weber, I., & Zhu, L. (2022). *DevOps: A Software Architect's Perspective* (2nd ed.). Addison-Wesley.
- Khan, M. A., Aslam, S., & Qamar, U. (2021). An energy-aware DevOps framework using software-defined infrastructure. *Journal of Systems and Software*, 178, 110974. https://doi.org/10.1016/j.jss.2021.110974
- Kumar, A., Singh, V., & Sharma, M. (2023). Energy-Aware DevOps Pipelines: AI-Driven Optimization Strategies. *Journal of Sustainable Computing*, 17, 102045. https://doi.org/10.1016/j.suscom.2023.102045
- Li, Y., Chen, Z., & Huang, J. (2022). Green IT and Sustainable Software Engineering Practices. *IEEE Access*, 10, 46789–46801. https://doi.org/10.1109/ACCESS.2022.3165221
- Nadeem, A., Malik, B., & Asif, M. (2023). Sustainable Software Development Using Artificial Intelligence Techniques. *Journal of Cleaner Production*, 414, 137651. https://doi.org/10.1016/j.jclepro.2023.137651
- Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3645–3650. https://doi.org/10.18653/v1/P19-1355
- Wang, Y., Wang, Y., & Song, Y. (2020). Smart Orchestration for CI/CD using Reinforcement Learning. *ACM Transactions on Software Engineering and Methodology*, 29(4), 1–23. https://doi.org/10.1145/3385734
- Zhao, Q., Yu, H., & Tang, X. (2021). AI-Driven DevOps: An Intelligent Framework for Agile Software Delivery. *Future Generation Computer Systems*, 115, 385–398. https://doi.org/10.1016/j.future.2020.09.001