International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 06 | June - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

A Practical Evaluation of Self-Hosted n8n for Secure and Scalable Workflow Automation

Mr. Saurabh Pawar[1], Mr. Shrish Pattewar[2], Ms. Mukta G. Shelke [3]

[1]*CSE Department, MGM's College Of Engineering, Nanded.*
[2] *CSE Department, MGM's College Of Engineering, Nanded.*
[3] *CSE Department, MGM's College Of Engineering, Nanded.*

Abstract - This study explores the implementation and performance of the n8n automation tool in a self-hosted environment. The primary objective is to determine whether deploying n8n locally can offer operational benefits over cloud-based CI/CD platforms. We hypothesize that local deployment of n8n provides enhanced control, improved performance, and cost savings, particularly for teams with strict data security or infrastructure customization requirements. The evaluation was conducted by setting up n8n on a virtual server using Docker, integrating it with essential services, and running test workflows on example projects. Results showed measurable gains in task execution speed and reliability, along with predictable resource usage and minimal external dependencies. These outcomes suggest that self-hosting n8n is a viable strategy for teams aiming to streamline development pipelines while maintaining full ownership of their automation environment. The findings contribute to the growing interest in open-source, self-managed DevOps solutions for modern software teams.

Key Words: automation, AI workflow, n8n, open source

1. INTRODUCTION

Automation is a cornerstone of modern software engineering, particularly in the realm of DevOps, where development and operations converge to create streamlined and resilient software delivery pipelines. The use of Continuous Integration and Continuous Deployment (CI/CD) has become essential to support agile methodologies and ensure that code changes are tested, built, and deployed rapidly and reliably. Numerous cloud-based tools have emerged to meet this demand— GitHub Actions, GitLab CI/CD, Jenkins, and CircleCI being among the most widely used. These platforms abstract much of the complexity involved in building automation pipelines, offering integration, pre-configured environments, and scalable execution infrastructure.

Despite their popularity, cloud-hosted automation platforms come with inherent limitations. Concerns around data privacy, cost at scale, and the inability to fully customize execution environments often arise in enterprise or compliance-heavy contexts. For instance, companies' operating in sectors like finance, healthcare, or defense may require more granular control over their data and systems than public platforms can guarantee. Moreover, cloud CI/CD tools typically impose limits on usage or restrict functionality behind premium tiers, creating scalability issues for growing teams or resource intensive projects. These challenges highlight the need for lightweight, self-managed alternatives that allow teams to maintain full ownership of their automation stack.

n8n is an emerging open-source automation tool designed to run pipelines and orchestrate workflows in self-hosted environments. Built with a container-first philosophy, n8n supports Docker based deployments, RESTful APIs, and GitOps-style integration, making it both flexible and easy to adopt. Unlike monolithic solutions such as Jenkins, which require extensive configuration and plugin management, n8n offers a simplified, modular approach that caters to smaller teams or developers seeking an efficient, minimal CI/CD engine. However, as it is relatively new in the automation landscape, there has been limited empirical research into its operational capabilities and comparative advantages over established cloud CI/CD services.

This study seeks to evaluate the viability of using n8n as a self-hosted automation solution for real world DevOps pipelines. Specifically, the research addresses the question : **Can self-hosting the n8n tool provide an effective, secure, and resource-efficient alternative to cloud-based CI/CD platforms?** We hypothesize that deploying n8n in a controlled server environment enables faster task execution, reduced latency, and increased configurability while maintaining reasonable system resource usage. To investigate this, we set up n8n on a virtual server using Docker, configured it with NGINX and PostgreSQL, and executed representative build-test-deploy workflows. The results of this research aim to guide practitioners and researchers in choosing or designing CI/CD solutions that align with security, performance, and cost objectives in modern software delivery environments.

2. METHODOLOGY

1. Materials

The n8n automation tool (version 1.3.1) was obtained from its official GitHub repository (https://github.com/n8n/n8n). The tool is distributed under the MIT License and publicly available. A virtual private server (VPS) with Ubuntu 22.04 LTS was provisioned from DigitalOcean (https://www.digitalocean.com), equipped with 2 vCPUs, 4 GB RAM, and 80 GB SSD storage. Docker Engine (version 24.0) and Docker Compose (version 2.20) were installed to support container-based deployment. PostgreSQL (version 15) was used for persistent job data storage, installed as a Docker container from the official PostgreSQL image. NGINX (version 1.24) was deployed as a reverse proxy with SSL secured via Let's Encrypt (https://letsencrypt.org). GitHub was used as the version control system, and webhook integration was configured to trigger workflows automatically.

2. Environment Setup and Deployment

All components were installed and configured manually through shell scripting to ensure replicability. The n8n container was launched using Docker Compose, with environment variables defined in an external .env file. A volume was mounted to persist

logs and configurations. PostgreSQL was linked to the n8n service to manage job states and workflow metadata. NGINX was configured to route external HTTPS traffic to the internal n8n service port, using TLS certificates issued by Certbot. The server firewall was managed via UFW to restrict access to only necessary ports (80, 443, and SSH).

3. Workflow Design and Execution

Two representative software projects were created: one in Python (Flask-based web app) and one in Node.js (Express-based API). For each project, a CI/CD pipeline was defined using n8n's declarative YAML syntax. The pipeline included build, test, and deploy stages. Each step executed in isolated Docker containers to ensure clean environments. Webhooks were configured to trigger the pipeline upon push events on the main branch. Logging was enabled for all jobs and stored in mounted volumes for analysis.

Each workflow was executed 20 times to account for variability in performance and to simulate real world usage. The start and end times of each stage were recorded, along with the status (success/failure), resource consumption (CPU/RAM), and any errors encountered.

4. Experimental Design

This study adopted a comparative benchmarking approach. Workflow latency, system load, and recovery time were measured for n8n and compared against GitHub Actions, using identical repositories and scripts. The primary goal was to assess the effectiveness of n8n in terms of speed, reliability, and resource usage in a self-hosted configuration. For GitHub Actions, the free tier was used, with default runners located in the U.S. East region.

5. Statistical Analysis

Basic statistical analysis was applied to compare workflow performance between n8n and GitHub Actions. Mean execution times, standard deviations, and 95% confidence intervals were calculated. Paired t-tests were used to assess the significance of differences in task duration and resource usage across platforms. Data was visualized using Python's matplotlib and pandas libraries to assist in interpretation.

All experiments were conducted within a controlled network environment, and no background processes were allowed to interfere with server performance during test runs.

3. RESULTS AND DISCUSSION

A total of 20 workflow executions (n = 20) were carried out for both the self-hosted n8n environment and the GitHub Actions platform. The collected metrics include average execution latency, system resource usage, and failure recovery time. Standard deviation (SD) and p-values from paired t-tests are provided where comparisons were made.

1. Workflow Execution Latency

Table 1 presents the mean execution time (in seconds) for the complete build-test-deploy pipeline. The self-hosted n8n setup demonstrated a lower average latency (M = 9.3 s, SD = 0.61 s) compared to GitHub Actions (M = 14.8 s, SD = 0.77 s). A paired t-test revealed that this difference was statistically significant ($p < 0.001$).

Table -1: Mean Workflow Execution Time (n = 20)

Platform	Mean Latency	SD	SEM	p-value
n8n (Self-hosted)	9.3	0.61	0.14	-
GitHub Actions	14.8	0.77	0.17	< 0.001

- **System Resource Usage**

CPU and memory consumption were tracked during all test runs using Docker statistics. Table 2 summarizes the average CPU and RAM utilization observed during pipeline execution in the self-hosted environment. CPU usage averaged 41% (SD = 3.4%), and memory usage averaged 520 MB (SD = 42 MB). No significant resource spikes were recorded.

Table -2: Average System Resource Utilization(n8n Only)

Metric	Mean Value	SD	SEM
CPU Usage (%)	41	3.4	0.76
RAM Usage (MB)	520	42	9.4

- **Failure Recovery Time**

Recovery time was measured from the moment of failure to the successful restart of a pipeline. The n8n environment demonstrated a significantly faster recovery rate (M = 12.1 s, SD = 1.5 s) in comparison to GitHub Actions (M = 28.2 s, SD = 2.1 s). As shown in Table 3, the difference in recovery times was also statistically significant ($p < 0.001$).

Table -3: Pipeline Failure Recovery Time (n = 20)

Platform	Mean Latency	SD	SEM	p-value
n8n (Self-hosted)	12.1	1.5	0.33	-
GitHub Actions	28.2	2.1	0.47	< 0.001

- **Discussion**

This study examined the performance, efficiency, and operational characteristics of the n8n automation tool deployed in a self-hosted environment. The objective was to determine whether a privately managed instance of n8n could serve as a viable alternative to popular cloud-based CI/CD services. The hypothesis posited that self-hosting n8n would offer greater control, reduced execution latency, and cost-effectiveness without sacrificing reliability or scalability. The results presented in the previous section support this hypothesis.

The first key objective was to evaluate the execution performance of n8n compared to GitHub Actions. The data showed that workflows executed in the self-hosted n8n environment consistently ran faster, with an average latency reduction of over 37%. This improvement can be attributed to reduced network overhead, absence of queueing delays, and full control over pipeline concurrency. These results align with prior studies suggesting that on-premise or local automation solutions can outperform cloud-based options in controlled environments.

The second objective was to assess system resource utilization. The n8n deployment maintained stable CPU and memory usage during all workflow executions, averaging 41% CPU load and 520 MB RAM usage. This level of efficiency highlights the lightweight nature of n8n and its suitability for smaller virtual servers. In comparison, cloud-based systems abstract resource consumption, making them harder to predict and optimize. The predictability in n8n's performance adds value for infrastructure-conscious teams.

The third objective was to measure reliability and recovery performance. Self-hosted n8n demonstrated faster failure recovery, taking less than half the time required by GitHub Actions. The ability to restart jobs locally without dependency on remote job queues or usage limits contributes to this performance advantage. These findings are particularly significant for teams operating under strict uptime or compliance requirements, where delays in recovery can affect critical services.

Despite its strengths, several limitations must be noted. First, setting up and maintaining a self-hosted solution requires a moderate level of system administration expertise. Misconfigurations, outdated images, or weak security practices may expose the system to vulnerabilities. Furthermore, the study did not simulate distributed workload conditions (e.g., across multiple nodes or containers), which could influence n8n's performance under scale.

In comparison to existing literature, few open-source CI/CD systems are as modular and minimal as n8n. Tools like Jenkins, while powerful, introduce complexity due to plugin management and maintenance. Drone CI offers a similar containerized model, but lacks the simplified declarative syntax that n8n provides. By providing empirical performance data, this study contributes practical insights into the operational advantages of self-hosted automation, especially for DevOps teams seeking lean, reliable systems.

4. CONCLUSIONS

This study explored the potential of self-hosting the n8n automation tool as a lightweight, secure, and efficient solution for managing CI/CD pipelines. By deploying the tool on a virtual server and comparing it against GitHub Actions, the results revealed significant improvements in workflow latency, resource predictability, and recovery performance. The experimental evidence supports the hypothesis that self-hosting n8n can serve as a cost-effective and controllable alternative to cloud native platforms, especially in contexts where security and customization are priorities.

The major discussion points highlight n8n's low system overhead, quick recovery time, and deterministic performance. While some operational effort is required for configuration and maintenance, the trade-off in control and transparency is considerable. Moreover, the ability to fully define and monitor automation in isolated environments makes it well-suited for research institutions, startups, and regulated industries.

This work contributes to the broader field of DevOps automation by providing empirical benchmarks and deployment guidance for an emerging open-source tool. It encourages further exploration into self-hosted CI/CD models and fosters innovation beyond traditional cloud services.

5. FUTURE SCOPE

Future research can extend this work by:

- Testing n8n under distributed and clustered environments using Kubernetes.
- Integrating n8n with service meshes and observability platforms (e.g., Prometheus, Grafana).
- Benchmarking against other self-hosted tools like Drone CI, GoCD, and Jenkins-X.
- Exploring security hardening techniques for production-level deployments.
- Evaluating the energy and cost efficiency of running n8n in hybrid cloud environments.

REFERENCES

1. n8n GitHub Repository. (2024). n8n automation tool. GitHub. https://github.com/n8n/n8n
2. Humble, J., & Farley, D. (2010). Continuous delivery: Reliable software releases through build, test, and deployment automation. Addison-Wesley.
3. Docker. (2024). Docker Engine and Docker Compose. https://docs.docker.com
4. NGINX. (2023). NGINX reverse proxy configuration guide. https://nginx.org/en/docs/
5. GitOps Working Group. (2022). GitOps principles and best practices (CNCF Whitepaper). Cloud Native Computing Foundation.
6. Sharma, P., Rane, A., & Chavan, S. (2021). Evaluating CI/CD performance in self-hosted environments. Journal of DevOps Practices, 12(3), 45–52.