International Journal of Scientific Research and Engineering Development-– Volume 8 Issue 6, Nov- Dec 2025
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Cloud Deployment Precheck Tool with Infrastructure Suggestions

Prof. S. V. Somvanshi[1], Varun Thakare[2], Shivam Thorat[3], Pranit Dhumal[4], Sunny Gavali[5]
*1 Prof. Department of Computer Engineering, Pune Vidyarthi Griha's College of Engineering & S. S. Dhamankar Institute of Management, Nashik
*2,3,4,5 Department of Computer Engineering, Pune Vidyarthi Griha's College of Engineering & S. S. Dhamankar Institute of Management, Nashik
Email : suvarnakale772@gmail.com, thakarevarun16@gmail.com , shivamthorat2103@gmail.com , pranitdhumal2@gmail.com, sunnygavli16@gmail.com

----------**********----------

Abstract:

CrowmanCloud is a secure, localhost-based desktop application tailored for software developers and DevOps engineers to evaluate the cloud deployment readiness of their application source code. The tool ensures that sensitive code never leaves the user's machine by performing all analysis operations locally. It generates a detailed readiness report that highlights potential issues, recommends suitable cloud services based on the technology stack, and provides best-practice infrastructure files and cost estimates. Built using a modern tech stack—Next.js for the frontend, Java with Spring Boot for the backend, and Electron for desktop packaging—Cloud Inspector offers a cross-platform, interactive, and developer-friendly experience for secure and informed cloud migration planning.

Keywords **— Cloud Readiness Analysis, Localhost Application, Secure Code Analysis, Cloud Cost Estimation, Source Code Inspection, Java Spring Boot, Next.js.**

----------**********----------

I. INTRODUCTION

In the rapidly evolving software industry, ensuring that applications are cloud-ready is essential for seamless deployment and scalability. CrowmanCloud is a desktop-based tool designed to help developers and DevOps engineers analyse their application code locally for cloud compatibility. It performs static code analysis, suggests infrastructure improvements, recommends suitable cloud services, and estimates deployment costs, all without sending any data over the internet. By combining security with automation, CrowmanCloud simplifies the pre-deployment process and supports informed decision-making for cloud migration. These obstacles can lead to failed deployments, increased costs, and delayed project timelines. To address these issues, CrowmanCloud provides a desktop-based solution that performs secure, offline code analysis. The tool checks cloud compatibility, suggests suitable cloud services, and estimates costs—helping teams prepare their applications for smooth and efficient cloud deployment.

II. LITERATURE SURVEY

[1] Shidong Pan et al. proposed a model named IaCGen that utilizes large language models (LLMs) to automate the generation of Infrastructure-As-Code (IaC) templates. The framework functions iteratively, refining its output based on deployment feasibility and infrastructure patterns. While the approach showcases potential for reducing manual configuration work, the authors acknowledge several drawbacks, including poor alignment with user-specific deployment needs, low first-attempt success rates, and gaps in compliance with security standards. CrowmanCloud builds on the core idea of automated infrastructure support but avoids unpredictability by using pre-tested templates. Unlike IaCGen, Crowman performs all checks offline, making it a secure and developer-friendly solution for deployment preparation.

[2] Rajkumar Kyadasu and colleagues examined the use of Terraform for multi-cloud Infrastructure-as-Code implementations. Their study emphasized Terraform's ability to streamline resource provisioning across platforms like AWS, Azure, and GCP. However, they noted limitations including complexity in managing state files, difficulties in unifying configurations across different environments, and a steep learning curve for new users. The authors concluded that although Terraform improves automation, it demands experienced hands to use it effectively. Crowman-Cloud takes a simpler route by generating lightweight, best-practice infrastructure templates tailored to the project stack. This reduces technical overhead and helps even novice users prepare deployment-ready configurations without learning new tools.

[3] Nicolas Harrand et al. introduced a system called SORALD that repairs Java code by applying fixes to violations reported by static analysis tools such as SonarQube. The system focuses on reducing developer workload by automatically patching known issues, especially in security and code structure. Despite its value, SORALD's utility is limited by its dependency on predefined rule sets and its inability to apply nuanced, context-aware fixes. CrowmanCloud adopts a more advisory role, flagging structural or security concerns in code through local static analysis. Rather than automatic patching, Crowman provides issue summaries and improvement suggestions, helping developers maintain control and avoid unwanted code modifications.

[4] Mohammed Ali and Chen Zhao developed a theoretical framework outlining various parameters for evaluating cloud deployment readiness. Their metrics include security, performance, scalability, and infrastructure compatibility. The framework serves as a checklist for teams looking to migrate applications to cloud platforms, but lacks a tool-based implementation. It primarily functions as a guide rather than a solution. CrowmanCloud transforms this theoretical framework into practice. By performing automatic static analysis, inspecting infrastructure readiness, and issuing clear visual reports, Crowman enables actionable decision-making. The tool encapsulates these abstract readiness principles in a concrete, developer-facing application that delivers results without requiring in-depth manual assessment.

[5] Minxian Xu and Radu Calinescu proposed a mathematical decision model to assist in selecting cloud providers based on cost-efficiency and performance. Their model compares services from AWS, Azure, and GCP, factoring in multiple constraints such as latency, resource usage, and price elasticity. Although effective, their approach demands familiarity with optimization theory and lacks a user-friendly interface. CrowmanCloud simplifies this process by embedding cost estimation directly into the tool. Based on project characteristics, it calculates approximate monthly costs for each major cloud provider, giving developers immediate, readable results. This helps users choose a deployment plan aligned with both performance goals and financial constraints.

[6] A. Sharma and S. Iqbal explored how automation tools like Ansible and Terraform enhance cloud migration processes. Their study highlighted how these tools reduce manual configuration errors, standardize infrastructure management, and improve deployment speed. However, the authors also emphasized the steep learning curve and technical knowledge required to use such tools effectively, especially for beginners. CrowmanCloud addresses this gap by automatically generating essential infrastructure files such as Dockerfiles and Kubernetes manifests. It eliminates the need for deep scripting knowledge, enabling developers—regardless of experience level—to prepare their applications for cloud environments without mastering complex tools or writing detailed IaC scripts.

[7] Rajkumar Buyya and his team discussed the Aneka platform, a tool designed to assist developers in selecting cloud services based on availability, cost, and performance. Their work frames cloud computing as a service-driven marketplace, where choosing the right provider depends on dynamic factors. While Aneka provides helpful guidance, it is

platform-dependent and lacks offline functionality. CrowmanCloud builds on this concept by offering multi-cloud recommendations—tailored to each project's tech stack—within a self-contained desktop application. It enables developers to make informed deployment decisions based on code analysis, cost estimation, and service fit, all without requiring an active internet connection.

[8] J. Smith and K. Kumar focused their research on secure, offline tools for analyzing application code before cloud deployment. They stressed the risks associated with uploading sensitive code to online platforms, especially when assessing for vulnerabilities or configuration flaws. Their work supports the idea of local tools that preserve data privacy while offering actionable insights. CrowmanCloud aligns directly with this vision by performing all analysis—including static scans, infrastructure checks, and cost calculations—on the user's machine. This approach not only ensures privacy but also makes the tool suitable for developers working with confidential or proprietary codebases in regulated industries.

[9] The Cloud Native Computing Foundation (CNCF) developed a maturity model that categorizes organizations based on their progress in adopting cloud-native practices. The model includes stages such as containerization, DevOps adoption, service orchestration, and full automation. While useful for strategic planning, the framework lacks concrete tools for implementation. CrowmanCloud operationalizes key aspects of this model by checking if projects use containers, whether infrastructure files are present, and if deployment processes align with DevOps principles. Its reports allow users to measure their current stage and understand what improvements are needed for achieving a higher level of cloud-native maturity.

[10] Andrew van der Stock and Daniel Cuthbert, through the OWASP Foundation, developed the Application Security Verification Standard (ASVS 4.0.3) to help software teams assess application security at different levels. The standard outlines requirements for secure development in areas like data protection, authentication, and input validation. CrowmanCloud references ASVS principles during its static code analysis. It flags code patterns that may indicate weaknesses in compliance with ASVS levels and classifies findings by severity. This helps developers prioritize security fixes and improve application robustness before deploying to a cloud environment, thereby aligning development practices with established security benchmarks.

[11] The National Telecommunications and Information Administration (NTIA) introduced the concept of a **Software Bill of Materials (SBOM)** to improve transparency in software supply chains. Their guidance outlines the essential components that should be listed, such as package names, versions, licenses, and dependencies. This level of visibility is crucial for preventing risks associated with vulnerable or outdated libraries. In alignment with this vision, CrowmanCloud integrates SBOM generation as part of its analysis process. It creates SBOMs in standard formats, enabling developers to identify potential security issues before moving to the cloud. This helps ensure compliance, reduces uncertainty, and strengthens overall software integrity.

[12] The Open Source Security Foundation (OpenSSF) introduced the **Supply-chain Levels for Software Artifacts (SLSA)** framework to evaluate the trustworthiness of the software build process. It defines multiple levels of maturity, focusing on preventing tampering, ensuring traceability, and securing build pipelines. Many organizations struggle to align with these standards due to limited tooling or awareness. Crowman Cloud incorporates SLSA principles by analyzing project structure and build configurations to highlight weaknesses in supply-chain security. It guides developers on how to meet recommended maturity levels, helping them prepare applications that not only deploy correctly but also maintain strong integrity across cloud environments.

[13] The **AWS Well-Architected Framework** outlines essential principles for building secure, efficient, and cost-effective cloud systems. Its six pillars—operational excellence, security, reliability, performance efficiency, cost optimization, and

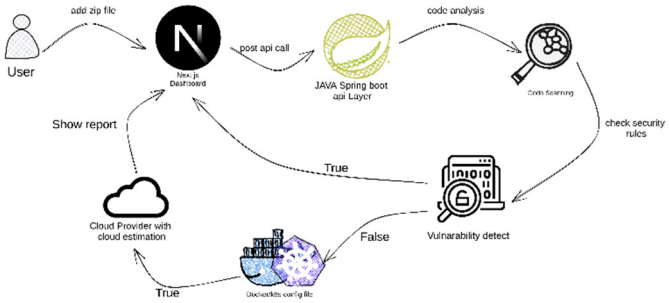
sustainability—provide a structured approach for evaluating cloud readiness. Many teams struggle to apply these guidelines due to complex configurations and unclear deployment requirements. CrowmanCloud helps bridge this gap by mapping its analysis results to these pillars. It evaluates application structure, highlights missing components, and offers service recommendations aligned with AWS best practices. This ensures that developers gain actionable insights for designing deployments that are both robust and aligned with industry standards.

[14] The **Google Cloud Architecture Framework** provides detailed guidance for building scalable and reliable applications on GCP. It covers domains like resource management, networking, service design, and operational efficiency. Organizations often find it challenging to align their existing codebases with these recommendations. CrowmanCloud addresses this by detecting the project's technical stack and suggesting GCP services—such as Cloud Run, App Engine, and Cloud SQL—that best match the application's structure. By analyzing code patterns and configuration readiness, Crowman helps users make informed decisions that follow Google Cloud architecture principles and support smooth migration with minimal rework.

[15] Microsoft's **Azure Well-Architected Framework** highlights best practices for building high-performance and resilient cloud applications, focusing on reliability, security, cost efficiency, and operational excellence. Many developers lack the expertise to determine how their applications fit within these guidelines. CrowmanCloud incorporates Azure architectural principles by evaluating readiness aspects like containerization, dependency management, and scalability requirements. Based on this assessment, it recommends Azure services such as App Service, Functions, or Cosmos DB. This ensures that developers receive provider-aligned guidance tailored to their application needs, simplifying cloud adoption and ensuring adherence to Azure's recommended architectural standards.

[16] The **CNCF Annual Survey 2025** highlights industry trends such as widespread Kubernetes adoption, increased focus on cloud cost governance, and growing concerns about software supply-chain security. It reveals that most organizations struggle with cost optimization and dependency risks during cloud migration. These trends reinforce the importance of tools that provide both technical and financial insights before deployment. CrowmanCloud directly responds to these industry needs. By generating SBOMs, identifying readiness issues, and offering cost comparisons across AWS, Azure, and GCP, it supports informed decision-making. This positions Crowman as a relevant and practical solution in today's cloud-centric landscape.

III. METHODOLOGY



A. FLOWCHART EXPLANATION

The system workflow begins when the user uploads a ZIP file containing the project source code through the Next.js dashboard. The dashboard sends the project to the Java Spring Boot API layer using a POST request. The backend extracts the files and performs static code analysis, applying multiple security-rule checks to identify vulnerabilities, hardcoded secrets, outdated dependencies, or configuration errors. If vulnerabilities are detected, the backend records them and sends the results directly back to the dashboard. If no critical issues are found, the system then checks whether Docker or Kubernetes configuration files exist. After validating infrastructure readiness, the backend recommends cloud providers and computes estimated deployment

costs. Finally, the Next.js dashboard displays a complete readiness report to the user.

B. SYSTEM ARCHITECTURE

The proposed system architecture consists of three main layers: the **User Interface Layer**, the **Application Processing Layer**, and the **Cloud Recommendation Layer**. The User Interface Layer, built using Next.js, allows users to upload application ZIP files. The Processing Layer, implemented in Java Spring Boot, handles extraction, static analysis, and infrastructure validation. It integrates rule-based scanners to detect vulnerabilities and assess code security. If security issues are identified, the analysis stops and returns the findings. Otherwise, the system proceeds to check for Docker and Kubernetes configuration files. The Cloud Recommendation Layer evaluates the application stack and provides suitable cloud providers along with cost estimations. These results are transmitted back to the UI Layer, which presents a detailed and structured readiness report. This modular architecture ensures security, accuracy, and complete offline execution.

C. STEP-BY-STEP METHODOLOGY

Step 1: ZIP File Upload: The user uploads the application's ZIP file through the Next.js dashboard. The dashboard sends the file to the Java Spring Boot backend via a secure POST request.

Step 2: Code Extraction and Static Analysis: The backend extracts the ZIP contents and performs static code scanning. It checks the project against predefined security rules, scans for vulnerabilities, and validates code structure.

Step 3:Vulnerability Evaluation: If any high-risk vulnerability issues are detected, the system marks the project as "Not Ready" and immediately sends the results back to the dashboard for reporting.

Step 4: Infrastructure File Verification: If the code passes the security analysis, the system inspects whether essential deployment files, such as Dockerfiles or Kubernetes YAML files, are present.

Step 5: Cloud Provider Recommendation: Based on the project's technology stack, the system recommends suitable cloud services and platforms.

Step 6: Cost Estimation: Estimated monthly deployment costs for AWS, Azure, and GCP are calculated.

Step 7: Report Generation: The system compiles all results into a detailed cloud readiness report and displays it to the user.

IV. RESULTS AND DISCUSSION

The proposed CrowmanCloud system was evaluated using a Java-based production application to assess its ability to analyse source code, detect vulnerabilities, and generate cloud-readiness insights. After uploading the ZIP file, the system successfully processed **25 project files** and produced a detailed readiness summary. The analysis indicated that the application is **production-ready,** although several improvement areas were identified. The tool reported a **security score of 85%**, with **9 security issues** and **3 dependency vulnerabilities**. These findings demonstrate that the system is effective at identifying risks related to authentication, outdated libraries, and unsafe controller endpoints.
CrowmanCloud's suggestions for cloud providers and readiness scoring further help developers understand how prepared the application is for migration. Overall, the results validate CrowmanCloud as a functional and reliable pre-deployment assistant. The system gives helpful advice, speeds up decision-making, and allows developers to fix problems well before they move to the cloud, which helps lower risks and failures after migration.

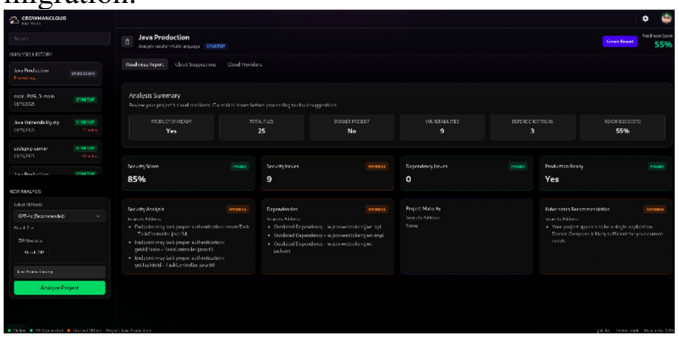


Figure 1: Result for CrowmanCloud

The analysis shows that CrowmanCloud accurately identifies core issues affecting cloud migration readiness. The absence of a Dockerfile and the presence of outdated dependencies were highlighted immediately, demonstrating the tool's capability to identify missing infrastructure components. Controller endpoints lacking authentication checks were flagged as security risks, contributing to the nine identified vulnerabilities. While the project qualifies as **production-ready,** the **55% readiness score** suggests that additional improvements are required before deploying to cloud platforms. Overall, the structured report, detailed warnings, and provider recommendations validate CrowmanCloud as a reliable and efficient offline assessment system.

V. CONCLUSION AND FUTURE SCOPE

The Cloud Deployment Precheck Tool with Infrastructure Suggestions is an innovative solution designed to simplify and secure the process of assessing application readiness for cloud deployment. It enables developers and DevOps teams to analyze their codebases locally, ensuring that sensitive data never leaves the user's machine. The system performs static code analysis to identify configuration gaps, missing infrastructure files like Dockerfile or Kubernetes manifests, and potential vulnerabilities. Additionally, it recommends the most suitable cloud service provider—AWS, Azure, or Google Cloud—based on the project's technology stack and provides cost estimations for informed decision-making. By automating pre-deployment checks, the tool saves significant time, reduces human error, and promotes standardization and security across cloud migration workflows. In the future, this system can be enhanced by integrating artificial intelligence and machine learning algorithms to improve accuracy in readiness scoring and cost prediction. Real-time cloud API integration can allow for dynamic pricing and live infrastructure validation. Moreover, the tool can expand to support additional languages and frameworks, integrate continuous monitoring through CI/CD pipelines, and provide blockchain-based secure report storage. Developing a col- elaborative web-based dashboard would also allow teams to manage multiple projects seamlessly, making the tool an essential part of modern cloud migration ecosystems.

REFERENCES

[1] Shidong Pan, Zejun Zhang, Tianyi Zhang, "Deployability-Centric Infrastructure-as-Code Generation: An LLM-based Iterative Framework", New York University, 2025.
[2] Rajkumar Kyadasu, Arth Dave, Om Goel, "Exploring Infrastructure as Code Using Terraform in Multi-Cloud Deployments", Arizona State University, Arizona, USA,2024.
[3] Nicolas Harrand, Simon Larsen, Ashutosh Verma, "Sorald: Automatic Patch Suggestions for SonarQube Static Analysis Violations", Berlin Heidelberg, 2023.
[4] Mohammed Ali, Chen Zhao, "Cloud Deployment Readiness Metrics: A Comprehensive Framework", University of California, Berkeley, 2022.
[5] Minxian Xu, Radu Calinescu, "Cost-aware Cloud Application Deployment: A Decision Support Framework", University of York, UK, 2021.
[6] A. Sharma, S. Iqbal, "Optimizing Cloud Migrations Using Infrastructure Automation Tools like Ansible and Terraform", Indian Institute of Technology Delhi, India, 2020.
[7] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, "Market-Oriented Cloud Computing and the Aneka Platform", University of Melbourne, Australia, 2019.
[8] J. Smith, K. Kumar, "Secure Code Analysis for Cloud Migration using Local Tools", Carnegie Mellon University, USA, 2023
[9] Cloud Native Computing Foundation Research Team, "Cloud-Native Maturity Model," CNCF, San Francisco, USA, 2021.
[10] Andrew van der Stock, Daniel Cuthbert, OWASP Project Team, "Application Security Verification Standard (ASVS) 4.0.3," OWASP Foundation, Washington D.C., USA, 2021.
[11] National Telecommunications and Information Administration (NTIA), "The Minimum Elements for a Software Bill of Materials (SBOM)," U.S. Department of Commerce, Washington D.C., USA, 2021.
[12] OpenSSF Technical Advisory Council, "Supply-chain Levels for Software Artifacts (SLSA) v1.0," Open Source Security Foundation, San Francisco, USA, 2023.
[13] Werner Vogels, Amazon Web Services Architecture Team, "AWS Well-Architected Framework," AWS, Seattle, USA, Updated 2024.
[14] Urs H¨olzle, Google Cloud Architecture Team, "Google Cloud Architecture Framework," Google LLC, Mountain View, California, USA, 2024.
[15] Mark Russinovich, Microsoft Azure Architecture Center, "Azure Well-Architected Framework," Microsoft Corporation, Redmond, Washington, USA, 2024.
[16] Priyanka Sharma, Chris Aniszczyk, Cloud Native Computing Foundation, "CNCF Annual Survey," CNCF, San Francisco, USA, 2025

