International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 12-No.1 (2026) pp. 1528-1536
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Advancing Cloud Infrastructure Supportability: A Framework for Digital Inclusion and Technical Empowerment

Sampath Rao Madarapu*

Independent Researcher, USA
* **Corresponding Author Email:** madarapu.sampathrao@gmail.com - **ORCID:** 0000-0002-3447-3050

Article Info:

DOI: 10.22399/ijcesen.5088
Received : 02 January 2026
Revised : 10 March 2026
Accepted : 15 March 2026

Keywords

Cloud Infrastructure Supportability,
Guided Troubleshooting Systems,
Self-Service Diagnostics,
Knowledge Management Architecture,
Digital Inclusion Framework

Abstract:

Cloud infrastructure supportability includes the use of knowledge management, diagnostic automation, and guided fault isolation processes at scale to make subject matter expertise consumable across many user segments. Knowledge stores help to achieve consistent regional diagnostics across distributed engineering teams using documented processes for troubleshooting and resolution of issues. Self-service diagnostic systems can provide insight into problems and bottlenecks from telemetry, configurations, and operational behavior. Guided troubleshooters implement diagnostic workflows, possibly spanning more than one domain and following an iterative process, gradually narrowing down the possibilities and helping the user understand how components of the infrastructure are related. Documentation is used for self-service by customers and to ensure a common understanding of the capabilities of support services. The telemetry from each support interaction is used in the iterative improvement of documentation and automated diagnostic logic. These feedback loops between internal knowledge, customer-facing diagnostics, and publicly available documentation increase the dependability of service, even lowering the technical knowledge barrier. This tri-layer supportability model accelerates the adoption of cloud across economies and geographies by making support models from reactive to proactive available in a systematic manner.

1. Introduction

Cloud infrastructure supports globally and socially critical services, such as the healthcare industry, education, finance, and enterprise applications. Mission-critical services require features of the network control layer that are low-latency and highly fault-tolerant in order to minimize disruption to services. Research has shown that to handle latency-sensitive service chains, end-to-end latency bounds should be under 10 ms for real-time applications, and availability should be higher than 99.999% uptime [1]. The technical complexity of a distributed cloud infrastructure can also be a barrier for organizations to deploy and manage technology, especially if specialized knowledge is scarce or distributed [50].

Supportability engineering seeks to fill this gap in knowledge by systematically developing knowledge libraries, automating diagnostics, and establishing troubleshooting procedures. Disruptions to the availability of cloud services have a direct impact on business. It is estimated that, depending on the size and type of business, the cost of service disruption is between US$56,000 and US$90,000 per minute [2]. Self-service diagnostics are designed so that users can solve their own problems without any human interaction by embedding diagnostic reasoning into diagnostics platforms. A diagnosing system processes streams of telemetry data, the operational state of the system components, and configuration data in order to identify performance bottlenecks or connectivity issues and resource contention without further human involvement.

Cloud availability is concerned with the availability of knowledge, diagnostic tools, and infrastructure (technical uptime). High-availability infrastructure solutions approach five-nines availability of 99.999% uptime with less than 52260 seconds, or 5.26 minutes, of downtime per year. Availability-as-a-service is an alternative solution. [2]. Organizations of all sizes and scales have experienced challenges in virtual machine

performance, agent communication, certificate management, and resource allocation. Customary operational support models rely on a small number of subject matter experts who are familiar with these specific areas. This creates slow resolution times and expertise bottlenecks.

Guided troubleshooters are interactive decision support systems that transform complex diagnostic procedures into stepwise procedures. For mission-critical service chains, the control plane must recover from failure conditions within the order of milliseconds since the control plane being unresponsive can cause the unavailability of many other dependent services [1]. However, while static documentation expects users to synthesize information from multiple sources, guided systems can select the next question based on observables and responses to other questions, progressively narrowing the scope of problems. These systems also have the ability to educate users on the underlying concepts and relationships of the infrastructure.

A three-layer supportability ecosystem consists of internal databases of diagnostic rules used by engineering teams, in-product self-service diagnostic tools used by customers to identify problems in real time, and public documentation that is continuously updated. Inter-layer feedback loops include support interactions informing the documentation team of knowledge gaps, feedback from customers using self-service diagnostic tools informing diagnostics and troubleshooting teams about common failure modes, and updated public documentation informing automated troubleshooting. Virtual machine troubleshooting exemplifies supportability system integration across multiple problem domains. Automating the troubleshooting of CPU utilization, I/O subsystem, network path, and agent health reduces the technical skills required to manage cloud-based technology infrastructure and democratizes knowledge to drive a digital transformation strategy irrespective of economic and geographical barriers.

1.1 Defining Cloud Infrastructure Supportability Engineering

Cloud Infrastructure Supportability Engineering may be described as the methodical design of expertise, diagnostics, and directed choice systems that permit dependable operation, fault isolation, and recovery of cloud services at scale. Although supportability engineering may be similar to other related fields, including Site Reliability Engineering (SRE), DevOps, and traditional IT operations, the primary goal of supportability engineering is the democratization and operationalization of expert knowledge instead of the exclusive optimization of system reliability metrics.

Conventional operations models are based on expert-led, reactive incident response. SRE focuses on reliability by automating and having error budgets, whereas DevOps focuses on speeding up delivery and feedback loops between operations and development. Supportability engineering adds to these paradigms by considering the way operational expertise is captured, structured, automated, and delivered to various users, such as non-expert operators and customers [3]. In this regard, supportability engineering considers diagnostic knowledge as a first-class system artifact, which can be versioned, validated, and continually improved. Formalizing supportability as a discipline can help organizations to create systems that can scale not just the capacity of infrastructure but also human knowledge [1]. These reframing roles underpin supportability engineering as a cornerstone capability of globally distributed cloud platforms, where the knowledge of operations needs to be resilient in the face of workforce turnover, geographic dispersion, and the rapid evolution of the platform.

2. Internal Knowledge Management Systems

2.1 Structured Troubleshooting Documentation

Internal knowledge bases are shared by distributed support engineering teams to exchange operational knowledge in managing cloud services at scale. Knowledge articles record troubleshooting methods, incident management, and resolution methods in structured and reusable forms that cut across geographic, organizational, and hierarchical boundaries. Enterprise application resiliency in the cloud relies on systematic documentation of operational behavior, incident patterns, and incident recovery procedures. Organizations that use systematic knowledge management processes can resolve IT incidents 40% faster than those that use ad-hoc knowledge management processes [3]. The systematic documentation of technical knowledge, incidents, and solutions transforms individual knowledge into enterprise knowledge and resources that are no longer dependent on the employee.

A troubleshooting documentation structure could consist of multiple pieces, each covering a functional area, and each piece is dependent on others to address a potential failure. The troubleshooting process to investigate performance problems with virtual machines is an example of such a documentation structure. The process involves using monitoring tools for CPU usage,

workload profiling, and resource contention analysis. To isolate performance bottlenecks, agent troubleshooting documentation describes observing storage I/O throughput, network bandwidth utilization, memory usage, and application resource consumption metrics. To troubleshoot communication errors, the documentation describes detailed steps for testing network connectivity through port availability checks, authentication through credential and permission scope checks, service health status through daemon status checks and log file inspection, and configuration by comparing current parameters to baseline settings.

With high-availability applications, redundancy configurations, and failover, disaster recovery policies must be documented and tested. For example, applications defined to have an uptime of 99.95% in cloud computing systems are expected to have not more than 4 hours and 23 minutes of downtime per month. For five nines of reliability, possible downtime is 26 seconds every month [4]. The certificate management documentation covers expiration checking and trust chain validation, renewing certificates, and the rotation ceremony. The extension certificate issues are the certificate store, private key, binding, cryptographic provider accessibility, setting permissions, and configuration on the client and server.

2.2 Cross-Regional Knowledge Transfer

Infrastructure operations are generally global, with problems and fixes transacted across time zones, languages, and cultures. By publishing knowledge from incident response to a centralized knowledge base, engineering teams can reduce mean time to resolution by eliminating repetitive diagnostic effort. Resilient application design involves not just redundant infrastructure but also redundant knowledge. Documentation should also be replicated in multiple regions to be available during a regional outage. Standardized documentation, including standard terminologies, diagnostics, and procedures for documentation, can be used to reproduce knowledge independent of regional practice. High availability resource management may be across teams and may also be across geolocations. Knowledge-sharing practices can reduce incident service restoration time by up to 35% for multi-region failures [4]. Engineers can use documented solutions from geographically distant engineers to solve previously unknown problems without the need for direct communication or synchronous collaboration. Knowledge base search, semantic understanding of technical jargon, symptom-to-solution matching with variations of problem descriptions, and relevance ranking with previously validated solutions determine the practical utility of documentation repositories.

3. Self-Service Diagnostic Capabilities

3.1 Automated Problem Detection

Self-service diagnostics are a product of telemetry analysis, configuration verification, and automated reasoning engines. Infrastructure users can self-diagnose and repair issues without needing skilled support technicians. Self-service diagnostics systems gather and monitor telemetry data, compare actual behavior against expected behavior, and, where certain thresholds are exceeded, trigger defined workflows to remediate issues. In terms of accessibility for all users and use cases, the diversity of users and their use cases needs to be considered through interface design principles that ensure explanations and advice are at an appropriate level for a diverse group of users. For example, a study found that only 23% of the reviewed AI diagnostic tools in this study adhered to accessibility guidelines for users with disabilities [5]. Automated diagnostic intelligence converts scarce technical support requiring human effort into a utility accessible through infrastructure management interfaces. Diagnostic automation targets multiple use cases. These include resource contention diagnostics—identifying the processes using the most CPU over time and correlating CPU usage spikes with workload events, and recommending actions to reduce the contention, such as terminating the process, allocating the resources, or migrating the workload to another device or server. Diagnostics for memory seek allocation patterns, for memory leak detection via buildup in consumption, for swap usage detection, and for suggesting configuration changes and physical capacity upgrades are also relevant. Locating performance bottlenecks and proving where they are requires multi-factor analysis of the infrastructure stack. Storage diagnostics include monitoring the IOPS and average latencies, the increased queue depth, and the differences between frontend applications and backend storage subsystem delays.

Diagnostics for agent communication: checks for virtual machine and management endpoint connectivity, including port availability checks; authentication checks for expired credentials and sufficient permission scopes; and service and configuration checks on daemon state and logs and known-good configuration values. Diagnostics for extension provisioning: checks for provisioning sequence steps, runtime component dependencies,

state during the installation process, and error points during the provisioning sequence. Certificate diagnostics directly address cryptographic infrastructure failures by automating checks for certificate expiration, trust chain validation, accessibility, and permission settings for private keys, as well as the correctness of certificate bindings and cryptographic providers.

3.2 Sustainability Integration and Adoption Impact

Sustainability impacts exist in relation to diagnostic automation, such as informing recommendations on resource use to minimize economic and environmental costs of operating diagnostic automation. Cloud computing technologies can also lead to positive sustainability impacts through improving the efficiency of resource use. Optimizing cloud resources can lead to energy savings of 30% to 40% [6]. Identifying underutilized resources provides right-sizing recommendations that reduce hardware requirements while meeting capacity needs. Identifying inefficiencies in distributing workload leads to consolidation, which results in better hardware utilization. Cloud sustainability frameworks also indicate that generally, every 1% improvement in server utilization efficiency translates to 0.7% less carbon emissions [6].

Self-service diagnostics reduce the number of manual support interactions by either resolving common issues or preventing the need to escalate issues to level-2 support. Professional support metrics for self-service diagnostics include case deflection and reduced resolution time. Deflection is a decrease in the number of support cases created. Reduced mean time to resolution (MTTR) indicates faster remediation due to faster detection of the problem. Broader availability of diagnostic tools enables organizations without important technical resources to manage their cloud infrastructure. Cloud services democratize the use of high-end diagnostic tools previously available only to enterprises with dedicated support teams.

3.3 Lowering Technical Barriers to Cloud Adoption

The limitation of digital inclusion in cloud computing is not always a matter of access to infrastructure, but rather access to operational expertise. Emerging economies often have small enterprises, public sector organizations, and institutions that do not have specialized cloud engineers who can diagnose complex failures in compute, networking, storage, and security domains. Cloud architectures that are based on supportability deal with this imbalance by putting operational intelligence into the platform.

Guided troubleshooting systems and self-service diagnostics Minimize the need to have specialized staff by converting expert thinking into a repeatable process [3]. This allows organizations with less technical capability to deliver operational results similar to those of large organizations. Consequently, cloud adoption is no longer tied to geographic location, the maturity of the labor market, or the size of an organization.

Supportability engineering directly leads to technical empowerment by minimizing the cognitive and technical barriers necessary to use cloud infrastructure. Users do not just receive answers but are taken through diagnostic reasoning that gradually develops knowledge about system behavior [5]. This learning impact converts the support interactions into learning experiences, enhancing long-term operational capacity and building sustainable digital participation.

3.4 Supportability Performance Metrics

In order to measure the success of supportability engineering efforts, organizations need metrics that go beyond the conventional availability and uptime metrics. Although infrastructure reliability is a vital factor [1][2], supportability performance is a measure of how well knowledge and diagnostics are provided to the user.

The most important supportability effectiveness indicators are Mean Time to Diagnose (MTTD), which is used to measure the speed at which a root cause can be identified, and Mean Time to Resolution (MTTR), which is used to measure the speed of remediation. Case deflection rate is a measure of the percentage of problems that are solved by self-service diagnostics. Knowledge reuse rate is the frequency of application of documented solutions across incidents, which is a measure of scalability of operational knowledge [3].

Other indicators are guided completion of troubleshooting success, which assesses the ability of users to arrive at a solution by using structured workflows, and documentation freshness, which measures the correspondence between documented procedures and the behavior of the live system [9]. Together, these measures allow organizations to measure the maturity of their supportability ecosystem and to invest in automation, documentation, and user experience.

4. Guided Troubleshooting Architecture

4.1 Step-by-Step Resolution Workflows

Guided troubleshooters are decision support systems that transform complex problem resolution dilemmas into stepwise, situation-adequate procedures. Rather than being given a series of steps that the user cross-references with disparate documentation, a guided troubleshooter prompts the user to take steps based on attributes of the symptoms, infrastructure under observation, and telemetry data over time. At each diagnostic step, candidates for the root cause are eliminated, while providing the user with information about dependencies in the infrastructure and operations. It is structured in several layers of intelligence, including symptom diagnosis engines, decider modules, and telemetry integration layers that validate user-reported symptoms with the current state of the infrastructure. An example of guided architecture complexity concerns virtual machine performance troubleshooting workflows, especially CPU utilization. Research on CPU utilization prediction modeling shows that VM workloads have special conceptual properties. CPU utilization prediction models that account for time series patterns and workload classes can achieve an accuracy ranging from 85% to 92% [7]. To investigate high CPU utilization, the first step is to confirm the reported high utilization through threshold checks. High CPU utilization may be further analyzed at the process level. Workload characterization studies have shown that CPU-bound workloads have a utilization variance coefficient between 0.15 and 0.35, and batch workloads have a coefficient higher than 0.50 [7]. Workflows for performance bottleneck isolation guide users in a multi-layer analysis where the I/O latency distributions of the storage subsystem, the bandwidth consumption of the network paths, memory allocations, and application execution are analyzed.

Kernel-level performance isolation is the mechanism responsible for the behavior of virtual machines. Kernel lock contention shows that locking leads to performance isolation violations. It was observed that extreme kernel contentions lead to performance deviations of over 200%. The median deviation of the moderate contention experiment conducted here was 15-30% [8]. Agent troubleshooting workflows resolve communication failures with verification sequences. Network connectivity tests are the primary check. The port accessibility, routing validation, and authentication verification steps check whether a service can be contacted on the specified port and whether authentication credentials are valid. The service health check checks if the daemon is running, checks log files for errors, and verifies the config file. Extension troubleshooting workflows trace the provision process to identify failure points of a multi-stage deployment. This is done by verifying dependency validation, state transitions, and the triggering error codes for rollbacks.

4.2 Human-Centered Design Principles

Good guided troubleshooters enable users to balance exhaustiveness with cognitive load through progressive disclosure, with low-level questions in plain language with minimal jargon appearing before high-level questions, and only high-level questions appearing after low-level lines of investigation have been exhausted. In an analysis of kernel lock performance isolation, the challenge of abstraction is that lock contention metrics (e.g., spinlock acquire times, mutex wait times, and read-write semaphore blocking patterns) need to be translatable into actionable tips for the average user [8]. Feedback was needed to indicate the progress of the investigation, which diagnostic actions have already been taken, and which paths remain to be pursued. Contextual help provided just-in-time education integrated into the diagnostics workflow. For example, it includes inline definitions, linking the appearance of technical terms to when they are first used within a workflow, and links for explaining concepts to related sections of documentation.

4.3 Feedback Loops as First-Class Architectural Components

In a fully developed supportability architecture, feedback loops between telemetry, diagnostics, guided troubleshooting, and documentation are first-class system components and not byproducts of support activity. The trends in directed troubleshooting processes, including common abandonment points or recurring symptom patterns can be used to point to areas in which diagnostic reasoning or explanatory material needs to be sharpened.

The interactions between support that result in human engineering produce high-value signals regarding the emergent failure modes, undocumented behaviors, or unclear diagnostic routes. These signals, when systematically obtained, are fed back into knowledge bases within the system, revise automated diagnostic rules, and prompt revisions in publicly available documentation. The closed-loop architecture allows the ongoing learning of the supportability

ecosystem so that operational intelligence is updated as the infrastructure that it supports is updated [3].

4.4 Governance and Risk in Automated Diagnostics

Governance and trust are becoming critical design aspects as guided troubleshooters and diagnostic systems are becoming more and more a part of machine learning and AI-based reasoning. Automated diagnostics should strike a balance between autonomy and mechanisms that ensure wrong or unsafe remediation measures are not taken. False positives will cause unnecessary configuration changes, and false negatives may slow down the resolution and undermine user confidence.

Good governance practices involve scoring the confidence of diagnostic conclusions, clear descriptions of the reasoning paths, and clear escalation limits that direct uncertain cases to human experts. Explainability is especially important in guided troubleshooting, where users need to know not only what action is suggested but also why it is suitable based on observed system behavior [5].

Human-in-the-loop design ensures that automation augments rather than replaces expert judgment [8]. By incorporating rollback mechanisms, auditability of diagnostic decisions, and continuous validation of automated logic against real-world outcomes, supportability systems can maintain trust while scaling operational intelligence.

5. Public Documentation and Continuous Improvement

5.1 Documentation Quality and Maintenance Practices

Publicly available documentation for cloud infrastructure ecosystems comes in two flavors: the first is the full technical documentation for customers to self-serve, and the second is for transparency of service capabilities and limitations. The quality, willingness, and currency of published information have a direct effect on adoption, as completeness and ease of finding information can be an important part of evaluation and continuing operational awareness for organizations considering what cloud environments to move their infrastructure into. Infrastructure services follow a never-ending cycle of adding and deprecating features, improving security, and improving performance, and documentation has to keep up with these changes. The term "knowledge decay" is the growing disparity between documented procedures and actual system behavior. The resulting discrepancy erodes user confidence and leads to unnecessary support contacts. Documentation should be reviewed and kept up to date against the service implementations. However, recent progress in automated bug localization techniques, for example, has shown promising results in the context of localization of kernel bugs, with LLM agents achieving localization rates of 28-42% on large and complex system-level bugs, given the documentation context [9]. This suggests that the quality of documentation is important for humans as well as for the performance of automated kernel bug localization tools.

Agent troubleshooting documentation describes the communication infrastructure between VMs and management endpoints. The Linux agent documentation describes this systematically, providing resolutions for connectivity failures. Authentication validation techniques stress credential configurations, and log analysis techniques stress error patterns, while configuration troubleshooting validates each parameter by comparing it against the expected schema. Provisioning workflows, dependency analysis, state transition monitoring, and failure remediation techniques are documented as troubleshooting extensions. Documenting security vulnerabilities adds pressure to get them right and release them promptly. One study of virtualization security vulnerabilities found security boundary violations that went unregulated for years. As an example, in studies on secure encrypted virtualization, researchers have also discovered that interrupt-based vulnerabilities can lead to privilege escalation in almost every case if countermeasures are not implemented [10]. This shows the importance of updated security documentation.

5.2 Feedback-Driven Continuous Improvement

Fostering a feedback loop with support operations provides telemetry from these interactions, and documentation provides business intelligence analysis. Commonly escalated issues may indicate deficiencies and weaknesses in self-service documentation. Repeatedly questioned problems indicate a need for added clarity and coverage, while long resolution times indicate inaccuracies or deprecated instructions in the documentation. By using telemetry to determine the content of their documentation, they are able to make just-in-time content that accounts for what the user is trying to do. The quality of their automated diagnostic mechanisms is directly related to the quality of their documentation. Research on bug localization

(finding bugs using tests) has shown that documentation context can improve automatic defect localization by 14-20 percentage points, compared to systems that do not provide structured documentation [9]. Accessibility includes aspects such as the availability, readability, structural layout, and navigability of documentation. The information architecture is hierarchical, with conceptual overview sections at the top, followed by procedural sections, which give instructions, and reference sections, which list parameters and options. Troubleshooting sections cover common failure scenarios.

Documentation systems and guided troubleshooters reinforce each other: the documentation system fills gaps in troublemaker support knowledge, which in turn drives the creation of documentation; documentation changes may lead to changes in the automated troubleshooting algorithm and vice versa, with changes to troubleshooting procedures appearing as new decision tree algorithms. Patterns like poorly configured systems can be used to create rules for self-service diagnostic tools. More detailed security documentation is also important to ensure patterns of vulnerabilities and their exploitation are rapidly documented, and defensive measures and risk assessments can be implemented [10]. The versioning support keeps track of the versions of the documentation that were released with the various releases of the service, allowing users to get the right version. An ongoing improvement process will make sure that the documentation is continuously updated with the implementation and contributions of support and users.

5.3 Traditional Support Models vs. Tri-layer Supportability Ecosystems

Conventional cloud support models are largely reactive and expert-based and are based on escalation chains that are concentrated in a small number of experts. Although they work well in the case of single events, these models do not scale well with the complexity of infrastructure and the diversity of customers. The time to resolution increases, the cost of operation increases, and knowledge is still isolated in teams [2]. In contrast, the tri-layer supportability ecosystem distributes expertise across internal knowledge systems, automated diagnostics, and guided troubleshooting interfaces [3]. This model scales horizontally by transforming expert reasoning into reusable assets that can be consumed repeatedly without proportional increases in staffing. Knowledge becomes resilient to personnel changes [4], and operational capability extends beyond centralized support organizations to end users themselves.

Table 1: *Critical Requirements and Business Impact of Cloud Service Reliability [1, 2]*

Infrastructure Characteristic	Description
Latency requirement for real-time applications	End-to-end latency bounds below 10 milliseconds
Enterprise cost impact of service interruptions	$5,600 to $9,000 per minute
Annual downtime allowance (at 99.999% reliability)	Approximately 5.26 minutes
Control plane failure consequence	Cascading effects on multiple dependent services
Common infrastructure challenges faced	Virtual machine performance degradation
	Agent communication failures
	Certificate management complexities
	Resource allocation inefficiencies

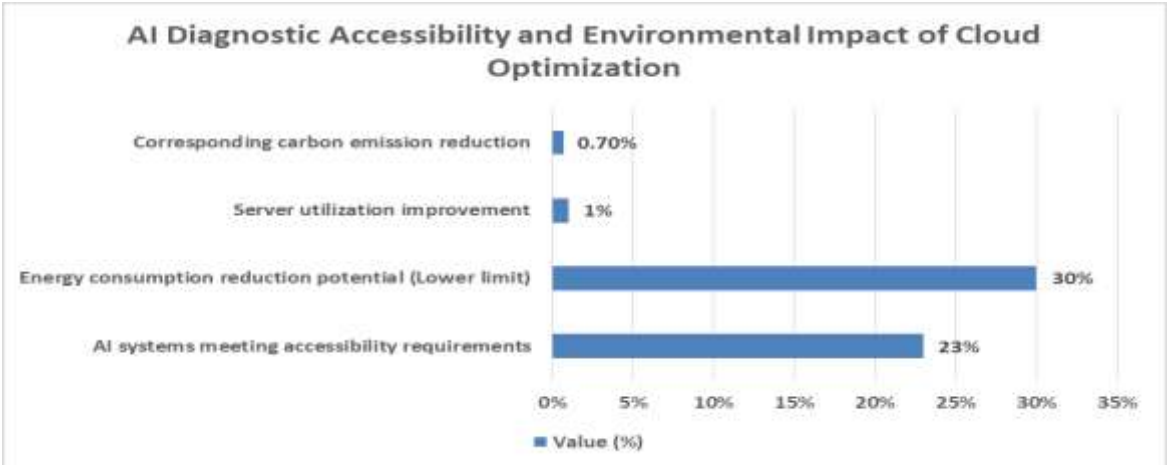

Figure 1: *AI Diagnostic Accessibility and Environmental Impact of Cloud Optimization [5, 6]*

Table 2: *CPU Utilization Modeling and Kernel-Level Performance Degradation Patterns [7, 8]*

Performance Analysis Element	Characteristic Value
CPU utilization prediction accuracy level (for temporal patterns)	85-92%
CPU-intensive application variance coefficient	0.15 to 0.35
Batch processing workload variance coefficient	> 0.50
Median performance degradation under moderate contention	15-30%

Table 3*: Documentation Quality Impact and Security Vulnerability Characteristics [9, 10]*

Characteristic	Performance or Risk Level
Automated bug localization accuracy range	28-42% (for complex system-level issues)
Knowledge decay definition	Gradual divergence between documented and actual behaviors
Agent troubleshooting documentation coverage	Connectivity failures, authentication validation, and log analysis
Extension troubleshooting documentation scope	Provisioning workflows, dependency validation, and state monitoring
Recurring question pattern indication	Clarity problems or information gaps
Bidirectional knowledge flow benefits	Documentation gaps inform content priorities
Version control mechanism function	Tracks documentation evolution across service releases

6. Conclusions

Knowledge management structures, automated diagnostics, and guided troubleshooting converge in a unified model to run inclusive cloud infrastructure operations, enabling reliable diagnostics across distributed engineering teams and preserving organizational knowledge within knowledge documentation systems despite employee turnover. Self-service diagnostic capabilities integrate intelligence into infrastructure platforms and transform technical support from a rare human skill to a ubiquitous automated utility. Guided troubleshooters combine the accessibility of manuals with the power of automated diagnostics in an interactive, conversational process that trains users in the process. Public documentation increases transparency, enables self-service, and ensures that technical documentation continues to improve with the services it documents. The flow of knowledge in this process creates a synergetic and compound improvement loop, where support knowledge influences documentation priorities and then automated diagnostics, and then back again. The resulting tri-layered supportability ecosystem will reduce the burden of technical knowledge required to operate cloud resources. This democratization opens up the opportunity for a larger number of stakeholders to participate in the digital transformation, regardless of location or industry. Helped by machine learning and increased telemetry, technical capability will also be democratized.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

References

[1] Chin-Wei Huang et al., "Resilient and Reliable Cloud Network Control for Mission-Critical Latency-Sensitive Service Chains," arXiv, 26th Nov. 2025. Available: https://arxiv.org/html/2511.21960v1

[2] Pengfei Chen et al., "Making Availability as a Service in the Clouds," arXiv, Mar. 2025. Available: https://arxiv.org/pdf/1503.04422

[3] Gireesh Kambala, "Designing resilient enterprise applications in the cloud: Strategies and best practices," WJARR, 2023. Available: https://wjarr.com/sites/default/files/WJARR-2023-0303.pdf

[4] Mr. Bibhu Kalyan Mishra and Prof. (Dr.) S.K. Yadav, "High Availability Of Resource In Cloud Computing Technology: Review, Issues And Challenges," IJETT, 2020. Available: https://ijettjournal.org/assets/Volume-68/Issue-1/IJETT-V68I1P205.pdf

[5] Chukwunonso Henry Nwokoye et al., "A Survey Of Accessible Explainable Artificial Intelligence Research," arXiv, 2024. Available: https://arxiv.org/pdf/2407.17484

[6] Paula Bajdor, "Cloud Computing in Terms of Sustainable Development – Evaluation and Mutual Relations," ScienceDirect. 2023. Available: https://www.sciencedirect.com/science/article/pii/S1877050923011778

[7] Shengwei Chen et al., "Modeling Conceptual Characteristics of Virtual Machines for CPU Utilization Prediction," arXiv, 2018. Available: https://arxiv.org/pdf/1811.04731

[8] Anjali and Michael M. Swift, "Locked In, Leaked Out: Measuring Isolation via Kernel Locks," arXiv, Jul. 2025. Available: https://arxiv.org/html/2507.21248v1

[9] Zhenhao Zhou et al., "Benchmarking and Enhancing LLM Agents in Localizing Linux Kernel Bugs," arXiv, May 2025. Available: https://arxiv.org/pdf/2505.19489

[10] Benedict Schlüter et al., "WeSee: Using Malicious #VC Interrupts to Break AMD SEV-SNP," arXiv, 2024. Available: https://arxiv.org/html/2404.03526v1