International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Managing Legacy Infrastructure with Modern Automation Toolchains

Nadeem Siddiqui

Independent Researcher
New York, USA
nadeem.ahmedk7@gmail.com

Abstract:
Legacy infrastructure is often at the center of critical enterprise systems. It runs core business applications, powers financial systems, and stores years of valuable data. However, these systems are famously hard to manage, secure, and scale. Today, agility and compliance are essential, so many wonder: how can modern automation tools help with legacy systems without causing chaos? This article looks at practical ways to automate aging infrastructure, connect old and new systems, and build a sustainable future for companies that can't simply move everything to the cloud. We'll discuss treating legacy as code, automating without major rewrites, and evolving infrastructure in a realistic, helpful, and even lighthearted way.

Keywords: Legacy Infrastructure, Automation Toolchains, Configuration Management, Infrastructure Modernization, Ansible, Puppet, Terraform, Technical Debt, CI/CD, Infrastructure as Code, Platform Engineering.

Introduction: The Challenge of Legacy Infrastructure

The Pervasive Nature of Legacy Systems

Many modern IT environments still rely on legacy infrastructure, systems, and applications. These were once advanced but are now outdated, hard to maintain, and can create serious operational and security risks. This is a widespread issue for both large companies and government agencies trying to modernize (Christopher Wilson & Ines Mergel, 2022) (Stephane Schlotterbeck & SSchlotterbeck@imf.org, 2017). Technical debt builds up for several reasons, such as the high cost and complexity of migration, a shortage of skilled staff for older technologies, and organizations' reluctance to change (Tariq Masood et al., 2023). As a result, many organizations must balance the push for innovation with the need to keep essential but aging infrastructure running.

Legacy infrastructure brings many challenges. These systems can cause inefficiencies, more downtime, and higher maintenance and support costs (International Monetary Fund, 2013). Security is also a major issue, since older systems often lack up-to-date protections and are more vulnerable to new cyber threats (Haneya Naeem Qureshi et al., 2020). Adding new technologies to legacy systems is often difficult and expensive, which can slow digital transformation and make it harder for organizations to keep up with market changes or use new tools like Artificial Intelligence (Patrick Dunleavy & Helen Margetts, 2023) (Tan Yiğitcanlar et al., 2024).

Bridging the Gap with Modern Automation

Given these challenges, organizations increasingly need effective ways to manage legacy infrastructure and move forward. Modern automation tools, such as Infrastructure as Code (IaC), Configuration Management, and CI/CD pipelines, help manage IT environments more efficiently and securely (Oana-Alexandra Dragomirescu et al., 2025). These tools make it easier to standardize deployments, automate routine tasks, and boost the reliability and flexibility of IT operations.

This paper argues that using modern automation tools in a thoughtful way can reduce the risks and problems linked to legacy infrastructure. By taking a step-by-step approach and using targeted automation strategies, organizations can become more agile, secure, and efficient, even as they keep supporting or slowly updating their old systems. The paper will look at these strategies and focus on practical ways to connect new automation tools with existing legacy environments.

Defining Legacy Infrastructure and Its Characteristics

Core Definition and Components

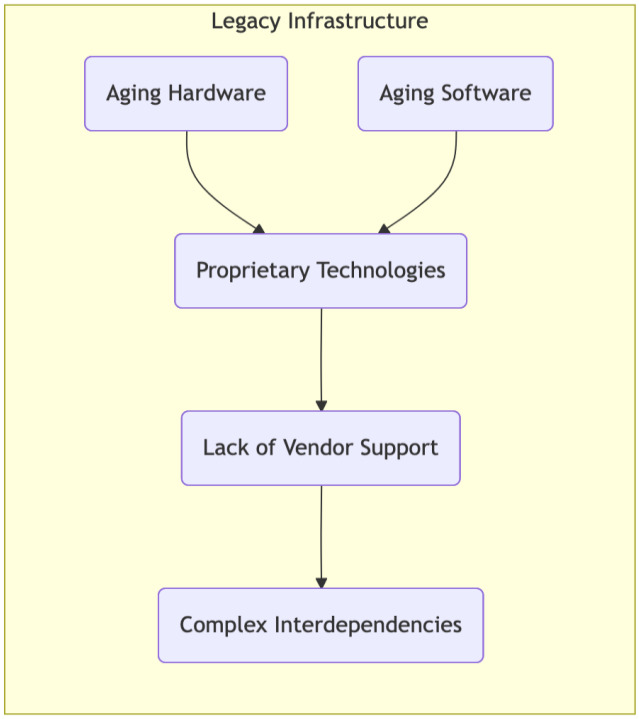
Legacy infrastructure means outdated IT systems, hardware, and software that are still part of an organization's technology stack. These systems, while functional, were typically developed using older technologies and methodologies, making them increasingly challenging to maintain, connect, and secure today. Organizations often keep them because they were expensive and haven't paid off their investment yet. Legacy infrastructure can include things like mainframe computers, old operating systems, or custom applications built without modern design principles like service-orientation (M. Papazoglou & Willem-Jan van den Heuvel, 2007).

The continued reliance on such systems is not necessarily a strategic choice but often a consequence of inertia, cost considerations, or a lack of viable migration paths. Consequently, organizations find themselves managing a heterogeneous environment where newer, agile systems coexist with these older, more rigid infrastructures. This duality presents ongoing operational complexities and can impede the adoption of more modern digital strategies and tools. As highlighted in discussions around evolving network technologies, the integration of new paradigms often requires careful consideration of existing, foundational systems(Rashid Mijumbi et al., 2015).

Common Characteristics and Challenges

Legacy infrastructure has several key features that add to the problems organizations face. Old hardware usually can't keep up with current performance needs and is more likely to break down, with replacement parts hard to find. Older software may not get security updates or support from vendors anymore, making it easier for attackers to exploit them (Rabia Khan et al., 2019). Proprietary technologies from before standardization can also make things worse by locking organizations into certain vendors and making it hard to connect with modern systems.

The interdependencies between legacy components and, increasingly, between legacy and modern systems, create a complex web that is difficult to untangle. Modifying one part of the legacy infrastructure can have unforeseen consequences on other interconnected systems, increasing the risk and cost of any planned changes. This complexity often hinders innovation and slows down the deployment of new services or updates. The lack of comprehensive documentation for older systems can also make troubleshooting and maintenance a time-consuming and knowledge-intensive process, often relying on the expertise of a few long-serving employees.



Defining Legacy Infrastructure and Its Characteristics Diagram

Characteristic	Description	Impact	Example
Aging Hardware	Outdated physical components nearing end-of-life or obsolescence.	Increased failure rates, performance bottlenecks, security vulnerabilities.	Mainframe servers from the 1990s, old network switches.
Outdated Software	Applications built on older programming languages or unsupported versions.	Difficulty in finding skilled personnel, integration challenges, lack of updates.	COBOL applications, Windows Server 2003.
Proprietary Technologies	Systems relying on unique or specialized hardware/software not widely adopted.	Vendor lock-in, high maintenance costs, limited interoperability.	Custom-built database solutions, older ERP systems.
Lack of Vendor Support	When the original vendor no longer provides updates, patches, or technical assistance.	Increased security risks, inability to resolve critical issues, reliance on internal expertise.	Software with discontinued official support, hardware no longer manufactured.
Complex Interdependencies	Tightly coupled systems where changes in one can have unforeseen effects on others.	Risks of system failure, difficulty in upgrades or replacements, slow deployment times.	Multiple applications heavily reliant on each other without clear APIs.

Modern Automation Toolchains: An Overview

Infrastructure as Code and Configuration Management

Modern automation toolchains help solve the challenges of legacy systems. Infrastructure as Code (IaC) leads this shift by treating infrastructure setup and management like software development. Tools such as Terraform and CloudFormation let teams define infrastructure in code, making it easier to use version control, automate deployments, and keep configurations consistent. Alongside IaC, configuration management tools like Ansible, Chef, and Puppet automate software installations and system settings across different environments. These tools help keep systems aligned with set standards, reducing errors and configuration drift that often happen with older methods. Using IaC and configuration management together improves efficiency and security by turning infrastructure and its state into code.

These technologies work best when used in a systematic way. IaC not only makes it easier to create new infrastructure, but also helps manage and remove existing resources, making the process more flexible than manual methods. Configuration management tools are key for keeping systems secure and consistent after they are set up. They make it possible to apply security patches, software updates, and compliance rules across all servers. As Eramo et al. (Romina Eramo et al., 2024) point out, model-based and smart automation are becoming more important for handling complex systems, and IaC with configuration management directly supports this goal.

Continuous Integration, Continuous Deployment, and Orchestration

Modern automation toolchains do more than manage infrastructure—they cover the whole software delivery process. Continuous Integration (CI) and Continuous Deployment (CD), known together as CI/CD, are key parts of this. Tools like Jenkins, GitLab CI, and GitHub Actions automate building, testing, and deploying software, which speeds up release cycles and improves quality (Ahmad Alnafessah et al., 2021)(Baasanjargal Erdenebat et al., 2023). This is especially useful in cloud-native environments where quick changes are needed (Theodoropoulos T, et al., 2023). Orchestration tools like Kubernetes add another layer by handling deployment, scaling, and running containerized apps, making distributed systems easier to manage.

Bringing together CI/CD and orchestration tools changes how software is delivered and managed. This approach encourages teamwork and quick feedback, which helps companies keep up with changing market needs (Rogelio Rodríguez Hernández et al., 2023). Adding AI and Machine Learning (ML) to DevOps, as Oyeniran et al. (Oyekunle Claudius Oyeniran et al., 2023) note, is starting to provide even smarter automation, like predictive maintenance and automatic problem-solving. These improvements help organizations move away from rigid old processes and become more flexible.

Monitoring and Observability

Good automation needs strong monitoring and observability. Modern toolchains use tools like Prometheus, Grafana, and the ELK stack (Elasticsearch, Logstash, Kibana) to give real-time information about system performance, health, and user activity. Observability focuses on understanding a system's internal state by looking at its outputs, which is important for finding problems in complex environments. Unlike traditional monitoring that tracks set metrics, observability helps teams see the bigger picture. Collecting and analyzing lots of data is key to keeping automated systems stable and reliable.

The information from monitoring and observability feeds back into automation, creating a cycle of ongoing improvement. For example, if monitoring tools find performance issues, they can automatically

trigger scaling or send alerts to fix problems. This feedback helps automated systems adjust to changing demands and prevent failures before they happen. As Ruf et al. (Philipp Ruf et al., 2021) explain in MLOps, detailed monitoring is key for managing complex systems, and this idea also applies to infrastructure automation.

Toolchain Category	Purpose	Key Technologies	Example Tools
Infrastructure as Code (IaC)	Automated infrastructure provisioning and management	Declarative configuration files	Terraform, CloudFormation, Pulumi
Configuration Management	Ensuring systems are in a desired state	Idempotent scripting	Ansible, Chef, Puppet
CI/CD Pipelines	Automating software build, test, and deployment	Version control integration, automated testing	Jenkins, GitLab CI, GitHub Actions
Monitoring and Logging	Observing system health and performance	Metrics collection, log aggregation	Prometheus, Grafana, ELK Stack
Orchestration	Automating the deployment, scaling, and management of containers	Container scheduling, service discovery	Kubernetes, Docker Swarm

Bridging the Gap: Automation Strategies for Legacy Systems

The Imperative for Modernization

Legacy infrastructure, with its outdated technology and limited flexibility, creates major challenges for today's operations. These systems often lack vendor support, have security issues, and are hard to connect with newer technologies (Fung Ji Lim & Tan Bee Sian, 2025). Keeping these systems can slow down agility, raise maintenance costs, and increase security risks, which goes against the goal of a flexible and secure IT environment. As a result, organizations need to find ways to add modern automation without completely replacing their existing systems.

Modern automation tools like Infrastructure as Code (IaC), configuration management, and CI/CD pipelines help manage IT environments more efficiently. They make it easier to set up and manage infrastructure consistently and repeatably, reducing mistakes and speeding up deployments (Ammar Salamh Alrawahna et al., 2025). The main challenge is not what these tools can do, but how to use them effectively with older systems that were not built for them.

Gradual Modernization and Re-platforming Approaches

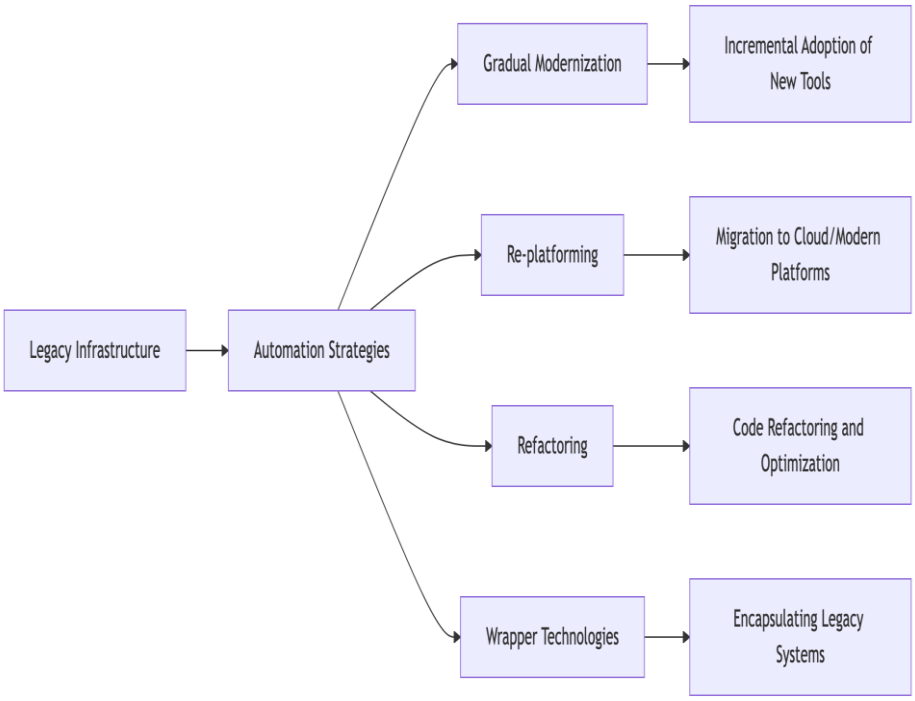
A key way to close the automation gap is through gradual modernization, which means updating or replacing parts of the legacy system incrementally. This method recognizes the risks and costs of making big changes all at once. For example, re-platforming moves an application to a new platform with only minor code changes (Hamdy Michael Ayas et al., 2023). By moving a legacy app to a newer operating system, a containerized setup, or the cloud, organizations can begin to realize automation benefits, such as more consistent deployments and better scalability, without needing to completely rebuild the system.

Refactoring is another important part of gradual modernization. It means changing the internal structure of a legacy application without altering its external behavior to make the code easier to maintain and faster to run. When paired with automated testing, refactoring helps reduce the risk of introducing new problems, making it safer to update and modernize legacy code (Christopher Bogart et al., 2021). By refactoring modules and integrating them into automated CI/CD pipelines, organizations can gradually adopt modern development and deployment practices across older systems.

Wrapper Technologies and Incremental Integration

Wrapper technologies provide a practical way to connect legacy systems to modern automation tools without changing the original code. This usually means building an abstraction layer or API that exposes the legacy system's functions in a standardized, machine-readable way. Modern automation tools, such as IaC scripts or CI/CD pipelines, can then interact with the legacy system via this wrapper (Fung Ji Lim & Tan Bee Sian, 2025). For example, a legacy batch-processing system could be managed via a REST API wrapper, enabling automated scheduling and monitoring.

To make these strategies work, organizations need to take a careful, step-by-step approach that focuses on security, compliance, and readiness. Automating legacy systems takes careful planning, strong testing, and ongoing monitoring to avoid new risks or disruptions. Change management should also be updated to ensure teams have the right skills and knowledge to handle mixed environments. By using gradual modernization, re-platforming, refactoring, and wrapper technologies, organizations can connect their legacy systems to modern automation tools and improve their IT flexibility and strength (Hamdy Michael Ayas et al., 2023).



Bridging the Gap: Automation Strategies for Legacy Systems Diagram

Case Studies on Effective Legacy Infrastructure Automation

Addressing Legacy Infrastructure Challenges through Automation

Legacy infrastructure remains common in organizations and brings major challenges for operations, security, and agility. These systems, often built with monolithic designs and proprietary technology, make it hard to innovate quickly and increase security risks (Andrew Arim & Joseph Wamema, 2022). Because these systems are so deeply embedded, adding new services or adapting to business changes is difficult. Organizations must keep essential functions running while slowly updating their technology. This section examines real examples of how organizations have used modern automation tools to manage and improve legacy infrastructure, demonstrating that full replacement is not always necessary or realistic.

Adopting advanced automation is essential for organizations dealing with complex legacy systems. More companies are using microservices to update old applications, reflecting a move toward more flexible and modular systems (Victor Velepucha & Pamela Flores, 2023) (Idris Oumoussa & Rajaa Saidi, 2025). Still, migration can be difficult. Successful automation usually involves phased steps that connect old, fragile systems with modern, agile methods. This requires a solid understanding of both the legacy setup and the automation tools to reduce risks and boost efficiency. The main goals are to keep operations resilient, improve security, and respond more quickly to market needs without disrupting the business.

Innovative Implementations: Tools and Strategies

Organizations have found success automating legacy infrastructure by combining Infrastructure as Code (IaC), configuration management, and CI/CD pipelines. For instance, a large financial institution facing risks from the manual setup of old mainframe systems chose to modernize in phases. They started by creating virtual versions of their core legacy environments with IaC tools like Terraform, allowing them to deploy infrastructure in test and development environments in a repeatable, controlled way (Olga Starkova & O. O. Андрейчіков, 2024). This lets them try out automation safely, without touching production systems. Next, they used configuration management tools like Ansible to keep operating systems and middleware on legacy servers in the correct state, reducing errors and configuration drift.

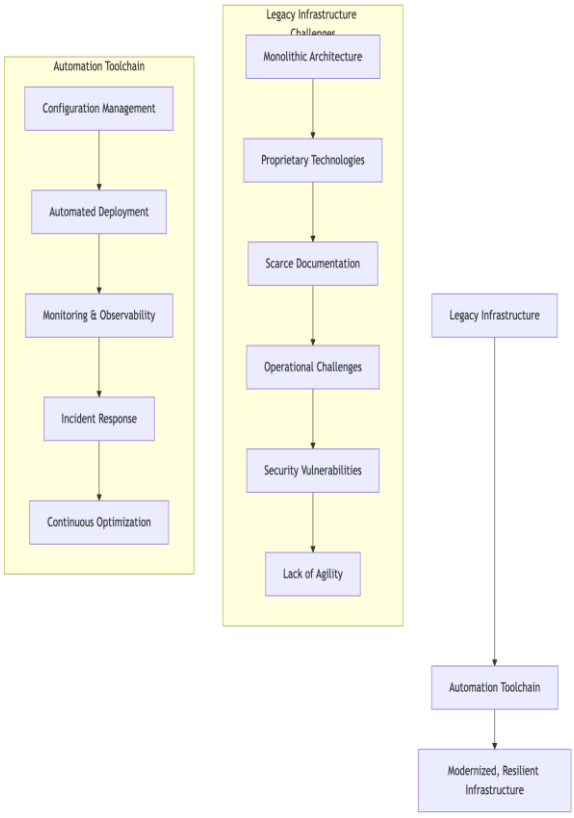
A step-by-step approach to modernization is important. Instead of automating all legacy systems at once, top organizations focus on specific areas that will have the most impact. For example, a telecommunications company tackled the challenge of automating the deployment and management of network services on old hardware. They set up CI/CD pipelines to automate testing and deployment scripts for these network parts. This meant building specialized adapters and wrappers to connect to legacy system command-line interfaces or APIs, which simplified things. As a result, they could deliver new network features faster, cutting lead times from months to weeks, and improved reliability with automated rollbacks when tests failed (Harshad Vijay Pandhare, 2025). Adding intelligent automation, like AI and machine learning for predictive maintenance and advanced testing, is also becoming key to making these complex systems work better (Harshad Vijay Pandhare, 2025).

Results and Future Directions

The results from automating legacy infrastructure show how powerful these strategies can be. The financial institution mentioned earlier cut the time needed to set up new environments from weeks to just a few days and reduced the number of critical incidents caused by configuration errors by over 60% (Olga Starkova & O. O. Андрейчіков, 2024). These gains in stability and efficiency saved money and improved compliance. The telecommunications company also became more agile, responding better to competition

and customer needs. Automating testing and deployment for legacy parts sped up delivery and improved service quality and reliability.

Looking ahead, organizations are moving toward advanced automation frameworks that can handle the mix of legacy and modern cloud-native systems. Studies on moving from monolithic to microservice architectures show that new approaches are needed to manage this complexity (Idris Oumoussa & Rajaa Saidi, 2025). More organizations are using AI and machine learning not just for testing, but also for predictive analytics, smarter incident management, and dynamic resource allocation in legacy systems. In large government IT projects, agile portfolio management is key to setting priorities, meeting policy goals, and staying within budget (Martina Ononiwu et al., 2025). So, automating legacy infrastructure should be seen as a continuous process of adaptation and intelligent management.



Case Studies: Successful Legacy Infrastructure Automation Diagram

Organization	Automation Toolchain	Key Challenges	Outcomes	Year Implemented
Global Bank Corp	Ansible, Jenkins, Docker	Downtime during migration, skill gap	90% reduction in manual deployment, 30% faster release cycles	2019
Manufacturing Solutions Inc.	Terraform, GitLab CI/CD, Kubernetes	Complex dependencies, security compliance	Improved infrastructure stability, 40% cost savings on cloud resources	2020
Retail Chain Operations	Chef, Azure DevOps, PowerShell	Fragmented environments, lack of standardization	15% increase in operational efficiency, enhanced disaster recovery capabilities	2018
Healthcare Provider Network	Puppet, CircleCI, Python scripts	Legacy application compatibility, resistance to change	Reduced configuration errors by 75%, strengthened security posture	2021

Technical Deep Dive: IaC and Configuration Management for Legacy

Adapting IaC to Legacy Environments

Bringing Infrastructure as Code (IaC) and Configuration Management (CM) tools into legacy environments can be challenging, but it is possible. Legacy systems often have monolithic designs and rely on manual processes, making it harder to apply automation directly. Still, with the right approach, these systems can be managed more efficiently. Agentless CM tools like Ansible work well for legacy systems that cannot support permanent agents or have strict network rules. These tools use existing protocols such as SSH or WinRM to run commands and manage settings remotely. By avoiding the need to install agents, this method lowers both the workload and security risks, making automation less disruptive for older systems.

On the other hand, agent-based CM tools like Puppet or Chef usually need more effort to set up. They require installing an agent on each legacy machine. This method provides stronger and ongoing control over configurations, but it only works if you can install and maintain agents on those older systems. Sometimes, organizations have to create specialized network routes or temporary solutions to enable and ensure agent deployment reliability. Deciding between agentless and agent-based tools depends on what your legacy systems can support, so it is important to carefully assess your environment before choosing a tool.

Addressing Idempotency and State Management

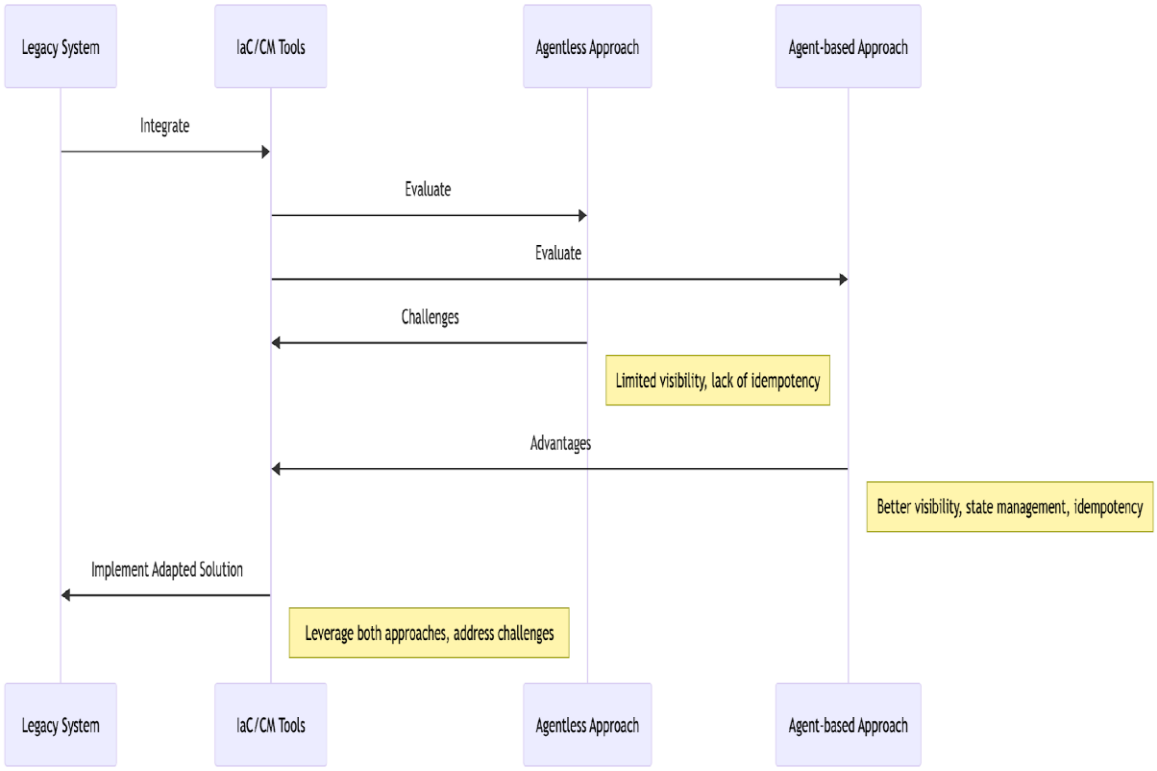
One important technical point when using IaC and CM with legacy systems is idempotency. Idempotency means that running the same configuration multiple times will always yield the same result, helping avoid unwanted changes and keeping things stable. Many legacy applications were not built with this in mind, so they often have complex dependencies and stateful operations that do not align well with declarative IaC approaches. For example, a script that installs a service might fail or even damage the system if the service is already installed. To prevent this, automation scripts should always check the current state before making changes, or use CM tools that have built-in idempotency checks.

Managing the state is also a big challenge. Modern IaC tools maintain a state file that records the current state of resources, so they only make the necessary changes. Legacy systems usually lack a central, machine-readable list of installed components, which makes things harder. Often, organizations have to develop custom approaches to identify and document the current state of their legacy systems before adopting IaC. This could mean writing scripts to check system settings, logs, or registry entries to build an initial picture. Another option is to start small, using IaC for less important tasks or new parts of the legacy system, and gradually take on more as you gain experience. The goal is to reach a clear, predictable state, even if you need to use manual steps or a mix of automation along the way.

Challenges and Mitigation Strategies

Using modern automation tools with legacy infrastructure poses many challenges, including security risks, insufficient documentation, and resistance to change. Security is especially important because legacy systems often use old software with known vulnerabilities, making them easy targets. When applying IaC and CM, it is important to plan carefully so automation does not make these systems more exposed. This could mean implementing strict network rules, using secure communication methods, and leveraging security features in automation tools. Also, because many legacy systems lack adequate documentation, teams often need to spend time investigating and reverse-engineering to understand how they work before they can automate them safely.

To address these challenges, a phased approach often helps, starting with the most important or difficult systems. Gradually modernizing parts of the system, such as moving some services to containers or newer platforms while keeping the main legacy system, can lower risks. For systems that cannot be easily updated, using IaC to regularly apply patches, configure security settings, and add monitoring tools can make them more secure and reliable. Success depends on strong change management, including training staff, encouraging teamwork between operations and development, and setting clear rules for automation. In the end, even though the technical challenges are tough, using IaC and CM thoughtfully can greatly improve how legacy systems are managed and secured.



Technical Deep Dive: IaC and Configuration Management for Legacy Diagram

Security and Compliance in Automated Legacy Environments

Enhancing Security Posture Through Automation

While automating legacy environments comes with its own set of challenges, it also creates valuable opportunities to strengthen an organization's security. Tools like Infrastructure as Code (IaC) and Configuration Management help deploy security controls in a consistent and repeatable way, which lowers the risk of misconfigurations often found in manual legacy systems (ADEDAMOLA ABIODUN SOLANKE, 2022). This approach makes sure security policies are applied the same way throughout the infrastructure, improving resistance to known vulnerabilities and unauthorized access. Adding security testing to Continuous Integration/Continuous Deployment (CI/CD) pipelines, a key part of DevSecOps, also helps find and fix security issues before they reach production (Xiaofan Zhao et al., 2024).

Automation helps organizations manage security more flexibly, making it easier to react quickly to new threats. For example, automated patching and fixing vulnerabilities can be done fast, reducing the time systems are at risk. This is especially important in industries like energy and banking, where digital changes bring new cybersecurity challenges that need quick solutions (Saqib Saeed et al., 2024)(Sonai Singaram Jeyaraj et al., 2024). In cloud environments, security automation can also improve how threats are detected and handled (Harshad Pitkar, 2025).

Risks and Mitigation Strategies

Even though automation brings many benefits, adding it to legacy systems can also create new risks. Because legacy systems are often complex, automation scripts or settings might accidentally open up

security gaps or disrupt important functions (Petter Kyösti & John Lindström, 2022). Since legacy systems often handle sensitive data, especially in regulated fields, strong access controls and audit tools are needed in the automation process (ADEDAMOLA ABIODUN SOLANKE, 2022). Organizations should carefully review how modern automation tools connect with legacy systems to keep sensitive data safe during modernization.

To manage these risks, organizations should automate in stages and make sure they fully understand their legacy systems. Strong testing methods, like security penetration testing of automated setups, are essential. Using a Service-Oriented Architecture (SOA) or a service-based approach can also help align legacy systems with modern digital strategies (Zoran Dragičević & Saša Bošnjak, 2019)(Petter Kyösti & John Lindström, 2022). This way, modernization can happen step by step, and automation can be used where it brings the most benefit.

Maintaining Compliance Standards

To keep automated legacy environments compliant, organizations need a proactive and unified approach. Automation can make compliance easier by creating clear records of all infrastructure changes. Tools that enforce set configurations and policies help meet rules like GDPR or HIPAA, lowering the risk of fines for non-compliance (ADEDAMOLA ABIODUN SOLANKE, 2022). In systems such as smart grids, saving security audits to unchangeable ledgers like blockchain can build trust and make compliance reporting more transparent (Andrés Marín et al., 2020).

It is important to build compliance requirements into automation from the very beginning, not add them later. This means turning regulatory rules into automated checks and controls within IaC and CI/CD pipelines (ADEDAMOLA ABIODUN SOLANKE, 2022). Regularly auditing both the automation processes and the systems they manage helps ensure ongoing compliance with internal and external rules. The flexibility that automation brings also helps organizations quickly adapt their business processes, which is key for staying aligned with digital strategies (Zoran Dragičević & Saša Bošnjak, 2019).

Conclusion and Future Directions

Summary of Findings

This paper has shown that although legacy infrastructure comes with many technical, operational, and security challenges, modern automation tools can help address these issues. Using Infrastructure as Code (IaC), Configuration Management, and CI/CD pipelines can connect legacy systems with current IT practices, making organizations more agile and efficient. As the case studies show, successful integration depends on careful planning and taking care of security and compliance from the start.

Our analysis supports the idea that modern automation, when used carefully, can reduce the risks of aging IT systems. This lets organizations go beyond just maintaining their systems and instead focus on proactive management and improvement, leading to stronger operations.

Future Research Opportunities

There is still much to learn about automating legacy infrastructure. Future research could look at new ways to connect very customized or old systems, possibly using advanced AI or machine learning for automated analysis and fixes (Eklas Hossain et al., 2019). It would also be useful to study the long-term economic benefits and return on investment of these automation projects, and compare them to how new technologies boost economic growth (Ryan A. Decker et al., 2014).

As technologies like 5G (Jeffrey G. Andrews et al., 2014), IoT (S. M. Riazul Islam et al., 2015), and Service-Oriented Architectures (M. Papazoglou & Willem-Jan van den Heuvel, 2007) keep developing and shaping infrastructure, it will be important to study how they can help manage and update legacy systems. Exploring how automation connects with sustainability goals (Jeffrey Rissman et al., 2020) and new ideas like Industry 5.0 (Dimitris Mourtzis et al., 2022) and the Metaverse (Zaheer Allam et al., 2022) could also lead to new innovations.

Acknowledgement

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

While preparing this manuscript, the author used an AI language model (ChatGPT, OpenAI) to help improve the language and readability. The author then reviewed and edited the content to make sure it was accurate and complete, and takes full responsibility for the final version.

REFERENCES:

[1] Christopher Wilson and Ines Mergel, "Overcoming barriers to digital government: mapping the strategies of digital champions," *Elsevier BV*, vol. 39, no. 2, pp. 101681–101681, Feb. 2022, doi: 10.1016/j.giq.2022.101681.

[2] Stephane Schlotterbeck and SSchlotterbeck@imf.org, "Tax Administration Reforms in the Caribbean: Challenges, Achievements, and Next Steps," *International Monetary Fund*, vol. 17, no. 88, pp. 1–1, Jan. 2017, doi: 10.5089/9781475592610.001.

[3] Tariq Masood, Asif Israr, Muhammad Zubair, and Usama Waleed Qazi, "Assessing challenges to sustainability and resilience of energy supply chain in Pakistan: a developing economy from Triple Bottom Line and UN SDGs' perspective," *Taylor & Francis*, vol. 42, no. 1, pp. 268–288, Mar. 2023, doi: 10.1080/14786451.2023.2189489.

[4] International Monetary Fund, "Nigeria: 2012 Article IV Consultation," *International Monetary Fund*, vol. 13, no. 116, pp. 1–1, Jan. 2013, doi: 10.5089/9781484311271.002.

[5] Haneya Naeem Qureshi, Marvin Manalastas, Syed Muhammad Asad Zaidi, Ali Imran, and Mohamad Omar Al Kalaa, "Service Level Agreements for 5G and Beyond: Overview, Challenges and Enablers of 5G-Healthcare Systems," *Institute of Electrical and Electronics Engineers*, vol. 9, pp. 1044–1061, Dec. 2020, doi: 10.1109/access.2020.3046927.

[6] Patrick Dunleavy and Helen Margetts, "Data science, artificial intelligence and the third wave of digital era governance," *SAGE Publishing*, vol. 40, no. 2, pp. 185–214, Sep. 2023, doi: 10.1177/09520767231198737.

[7] Tan Yiğitcanlar, Anne David, Wenda Li, Clinton Fookes, Simon Elias Bibri, and Xinyue Ye, "Unlocking Artificial Intelligence Adoption in Local Governments: Best Practice Lessons from Real-World Implementations," *Multidisciplinary Digital Publishing Institute*, vol. 7, no. 4, pp. 1576–1625, Jun. 2024, doi: 10.3390/smartcities7040064.

[8] Oana-Alexandra Dragomirescu, Pavel-Cristian Crăciun, and Ana-Ramona Bologa, "Enhancing Invoice Processing Automation Through the Integration of DevOps Methodologies and Machine Learning," *Multidisciplinary Digital Publishing Institute*, vol. 13, no. 2, pp. 87–87, Jan. 2025, doi: 10.3390/systems13020087.

[9] M. Papazoglou and Willem-Jan van den Heuvel, "Service oriented architectures: approaches, technologies and research issues," *Springer Science+Business Media*, vol. 16, no. 3, pp. 389–415, Mar. 2007, doi: 10.1007/s00778-007-0044-3.

[10] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," *Institute of Electrical and Electronics Engineers*, vol. 18, no. 1, pp. 236–262, Sep. 2015, doi: 10.1109/comst.2015.2477041.

[11] Rabia Khan, Pardeep Kumar, Dushantha Nalin K. Jayakody, and Madhusanka Liyanage, "A Survey on Security and Privacy of 5G Technologies: Potential Solutions, Recent Advancements, and Future Directions," *Institute of Electrical and Electronics Engineers*, vol. 22, no. 1, pp. 196–248, Aug. 2019, doi: 10.1109/comst.2019.2933899.

[12] Romina Eramo, Bilal Said, Marc Oriol, Hugo Brunelière, and Sergio Morales, "An architecture for model-based and intelligent automation in DevOps," *Elsevier BV*, vol. 217, pp. 112180–112180, Aug. 2024, doi: 10.1016/j.jss.2024.112180.

[13] Ahmad Alnafessah, Alim Ul Gias, Runan Wang, Lulai Zhu, Giuliano Casale, and Antonio Filieri, "Quality-Aware DevOps Research: Where Do We Stand?," *Institute of Electrical and Electronics Engineers*, vol. 9, pp. 44476–44489, Jan. 2021, doi: 10.1109/access.2021.3064867.

[14] Baasanjargal Erdenebat, Bayarjargal Bud, Temuulen Batsuren, and Tamás Kozsik, "Multi-Project Multi-Environment Approach—An Enhancement to Existing DevOps and Continuous Integration and Continuous Deployment Tools," *Multidisciplinary Digital Publishing Institute*, vol. 12, no. 12, pp. 254–254, Dec. 2023, doi: 10.3390/computers12120254.

[15] Theodoropoulos T *et al.*, "Security in Cloud-Native Services: A Survey," *Multidisciplinary Digital Publishing Institute*, vol. 3, no. 4, pp. 758–793, Oct. 2023, doi: 10.3390/jcp3040034.

[16] Rogelio Rodríguez Hernández, Begoña Morós, and Joaquín Nicolás, "Requirements management in DevOps environments: a multivocal mapping study," *Springer Science+Business Media*, vol. 28, no. 3, pp. 317–346, Jan. 2023, doi: 10.1007/s00766-023-00396-w.

[17] Oyekunle Claudius Oyeniran, Adebunmi Okechukwu Adewusi, Adams Gbolahan Adeleke, Lucy Anthony Akwawa, and Chidimma Francisca Azubuko, "AI-driven devops: Leveraging machine learning for automated software deployment and maintenance," *Fair East Publishers*, vol. 4, no. 6, pp. 728–740, Dec. 2023, doi: 10.51594/estj.v4i6.1552.

[18] Philipp Ruf, Manav Madan, Christoph Reich, and Djaffar Ould Abdeslam, "Demystifying MLOps and Presenting a Recipe for the Selection of Open-Source Tools," *Multidisciplinary Digital Publishing Institute*, vol. 11, no. 19, pp. 8861–8861, Sep. 2021, doi: 10.3390/app11198861.

[19] Fung Ji Lim and Tan Bee Sian, "Overview of Software Re-Engineering Concepts, Models and Approaches," *State Polytechnics of Andalas*, vol. 9, no. 1, pp. 46–46, Jan. 2025, doi: 10.62527/joiv.9.1.3034.

[20] Ammar Salamh Alrawahna, Amro Alzghoul, and Hussain Awad, "The Impact of Artificial Intelligence on Public Sector Decision- Making: Benefits, Challenges, and Policy Implications," *EconJournals*, vol. 15, no. 5, pp. 125–138, Aug. 2025, doi: 10.32479/irmm.19419.

[21] Hamdy Michael Ayas, Philipp Leitner, and Regina Hebig, "An empirical study of the systemic and technical migration towards microservices," *Springer Science+Business Media*, vol. 28, no. 4, pp. 85–85, May 2023, doi: 10.1007/s10664-023-10308-9.

[22] Christopher Bogart, Christian Kästner, James D. Herbsleb, and Ferdian Thung, "When and How to Make Breaking Changes," *Association for Computing Machinery*, vol. 30, no. 4, pp. 1–56, Jul. 2021, doi: 10.1145/3447245.

[23] Andrew Arim and Joseph Wamema, "Towards an Improved Framework for E-Risk Management for Digital Financial Services (DFS) in Ugandan Banks," *University of Zagreb, Faculty of organization and informatics*, vol. 46, no. 1, pp. 103–127, Jun. 2022, doi: 10.31341/jios.46.1.6.

[24] Victor Velepucha and Pamela Flores, "A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges," *Institute of Electrical and Electronics Engineers*, vol. 11, pp. 88339–88358, Jan. 2023, doi: 10.1109/access.2023.3305687.

[25] Idris Oumoussa and Rajaa Saidi, "The Ontology-Based Mapping of Microservice Identification Approaches: A Systematic Study of Migration Strategies from Monolithic to Microservice Architectures," *Multidisciplinary Digital Publishing Institute*, vol. 14, no. 4, pp. 133–133, Apr. 2025, doi: 10.3390/computers14040133.

[26] Olga Starkova and O. O. Андрейчиков, "Intellectual capital of IT companies in the development processes of innovative technologies and digital transformations: Historical and genetic analysis," *Business Perspectives*, vol. 23, no. 4, pp. 64–75, Oct. 2024, doi: 10.57111/devt/4.2024.64.

[27] Harshad Vijay Pandhare, "Future of Software Test Automation Using AI/ML," *Engg Journals Publications*, vol. 14, no. 05, pp. 27159–27182, May 2025, doi: 10.18535/ijecs.v14i05.5139.

[28] Martina Ononiwu, Tony Isioma Azonuche, and Joy Onma Enyejo, "Investigating Agile Portfolio Management Techniques for Prioritizing Strategic Initiatives in Large-Scale Government IT Projects," *Fair East Publishers*, vol. 7, no. 6, pp. 464–483, Jun. 2025, doi: 10.51594/ijmer.v7i6.1941.

[29] ADEDAMOLA ABIODUN SOLANKE, "Enterprise DevSecOps: Integrating security into CI/CD pipelines for regulated industries," *GSC Online Press*, vol. 13, no. 2, pp. 633–648, Feb. 2022, doi: 10.30574/wjarr.2022.13.2.0121.

[30] Xiaofan Zhao, Tony Clear, and Ramesh Lal, "Identifying the primary dimensions of DevSecOps: A multi-vocal literature review," *Elsevier BV*, vol. 214, pp. 112063–112063, Apr. 2024, doi: 10.1016/j.jss.2024.112063.

[31] Saqib Saeed *et al.*, "Digital Transformation in Energy Sector: Cybersecurity Challenges and Implications," *Multidisciplinary Digital Publishing Institute*, vol. 15, no. 12, pp. 764–764, Dec. 2024, doi: 10.3390/info15120764.

[32] Sonai Singaram Jeyaraj, Chelliah Paramasivan, Maduraiveeran Sumathi, and Sasankan Silpa, "Navigating Digital Transformation in Banking with Cloud Computing Solutions," *Scientific Research Publishing*, vol. 12, no. 06, pp. 4227–4253, Jan. 2024, doi: 10.4236/ojbm.2024.126212.

[33] Harshad Pitkar, "Cloud Security Automation Through Symmetry: Threat Detection and Response," *Multidisciplinary Digital Publishing Institute*, vol. 17, no. 6, pp. 859–859, Jun. 2025, doi: 10.3390/sym17060859.

[34] Petter Kyösti and John Lindström, "SOA-Based Platform Use in Development and Operation of Automation Solutions: Challenges, Opportunities, and Supporting Pillars towards Emerging Trends," *Multidisciplinary Digital Publishing Institute*, vol. 12, no. 3, pp. 1074–1074, Jan. 2022, doi: 10.3390/app12031074.

[35] Zoran Dragičević and Saša Bošnjak, "Harmonizing business and digital enterprise strategy using SOA middle-out and service-based approach," *University of Novi Sad, Technical faculty Mihajlo Pupin, Zrenjanin*, vol. 9, no. 2, pp. 97–112, Jan. 2019, doi: 10.5937/jemc1902097d.

[36] Andrés Marín, Sergio Chica, David Arroyo, Florina Almenárez, and Daniel Díaz-Sánchez, "Security Information Sharing in Smart Grids: Persisting Security Audits to the Blockchain," *Multidisciplinary Digital Publishing Institute*, vol. 9, no. 11, pp. 1865–1865, Nov. 2020, doi: 10.3390/electronics9111865.

[37] Eklas Hossain, Imtiaj Khan, Fuad Un-Noor, Sarder Shazali Sikander, and Md Samiul Haque Sunny, "Application of Big Data and Machine Learning in Smart Grid, and Associated Security Concerns: A Review," *Institute of Electrical and Electronics Engineers*, vol. 7, pp. 13960–13988, Jan. 2019, doi: 10.1109/access.2019.2894819.

[38] Ryan A. Decker, John Haltiwanger, Ron S. Jarmin, and Javier Miranda, "The Role of Entrepreneurship in US Job Creation and Economic Dynamism," *American Economic Association*, vol. 28, no. 3, pp. 3–24, Jul. 2014, doi: 10.1257/jep.28.3.3.

[39] Jeffrey G. Andrews *et al.*, "What Will 5G Be?," *Institute of Electrical and Electronics Engineers*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014, doi: 10.1109/jsac.2014.2328098.

[40] S. M. Riazul Islam, Daehan Kwak, Md. Humaun Kabir, Mahmud Hossain, and Kyung Sup Kwak, "The Internet of Things for Health Care: A Comprehensive Survey," *Institute of Electrical and Electronics Engineers*, vol. 3, pp. 678–708, Jan. 2015, doi: 10.1109/access.2015.2437951.

[41] Jeffrey Rissman *et al.*, "Technologies and policies to decarbonize global industry: Review and assessment of mitigation drivers through 2070," *Elsevier BV*, vol. 266, pp. 114848–114848, Mar. 2020, doi: 10.1016/j.apenergy.2020.114848.

[42] Dimitris Mourtzis, John Angelopoulos, and Nikos Panopoulos, "A Literature Review of the Challenges and Opportunities of the Transition from Industry 4.0 to Society 5.0," *Multidisciplinary Digital Publishing Institute*, vol. 15, no. 17, pp. 6276–6276, Aug. 2022, doi: 10.3390/en15176276.

[43] Zaheer Allam, Ayyoob Sharifi, Simon Elias Bibri, David S. Jones, and John Krogstie, "The Metaverse as a Virtual Form of Smart Cities: Opportunities and Challenges for Environmental, Economic, and Social Sustainability in Urban Futures," *Multidisciplinary Digital Publishing Institute*, vol. 5, no. 3, pp. 771–801, Jul. 2022, doi: 10.3390/smartcities5030040.