Cognizant

Adopting DevOps: Overcoming Three Common Stumbling Blocks

By taking a pragmatic approach to DevOps, IT organizations can more effectively address operational, structural and technological challenges to efficiently deliver high-quality applications and services.

Executive Summary

DevOps is a set of operating practices supplemented by tool kits that promise to transform the speed and quality with which IT organizations deliver applications and services to businesses.

However, the euphoria associated with the concept and its benefits has not been matched by its actual adoption. This is largely due to three major stumbling blocks: **the state of organizational preparedness; the number of extant heritage applications and systems in an IT landscape; and concerns about reliability, security and compliance.**

This white paper lays out practical solutions that IT can apply to overcome each of these obstacles and apply the principles of DevOps to swiftly and continuously innovate and iterate, making the business more competitive.

TRIPPING OVER DEVOPS

The digital business era demands that organizations rapidly innovate and quickly bring their best ideas to market – and then continuously improve on them. DevOps encompasses concepts, practices and tools designed to bring IT development and operations capabilities together to achieve these goals.

While every organization may define and implement DevOps differently, a core DevOps tenet is the enablement of development and operations to improve their agility and collaborate more effectively while continuously building, testing, deploying and maintaining.

Most organizations favor DevOps because of the business advantages it offers. Yet many IT organizations struggle to varying degrees in fully practicing DevOps. These difficulties are not due to DevOps' conceptual complexity or dilemmas about which technologies to choose. Instead, the following three challenges are typically what cause IT organizations to stumble when attempting to evolve toward DevOps:

- **Organizational preparedness for enabling the three practices of DevOps – agility, collaboration and orchestration:** Technology support in enabling agility is relatively easy. The more challenging aspect is creating the collaborative organizational operating model to support the "continuous everything" mindset and the competency orchestration needed during the ever-shrinking release cycle.
- **Scalability of heritage infrastructure and software platforms in a DevOps-paced digital world:** The challenges – and myths – associated with legacy modernization often become a technology and financial deterrent to DevOps adoption.
- **Reliability, security and regulatory compliance:** Concerns about securing the DevOps-enabled continuous development (CD) ecosystem and ensuring that the myriad of interconnected open source technologies perform reliably can cause some organizations to hit the pause button on DevOps.

Addressing these three obstacles is critical for good development hygiene, though solving the underlying issues will be complex. Those undertaking the effort, however, will generate substantial efficiency gains for the IT organization and greater competitive flexibility for the business.

ENABLING ORGANIZATIONAL PREPAREDNESS

Readiness implies the organization is geared up with technology-enabled agility, a collaborative operating model with a continuous everything mindset and competency orchestration to make a successful transition. Drilling into the three dimensions in readiness reveals several potential issues.

Technology-Enabled Agility

Technology-enabled agility should be relatively easy to achieve, provided application development and operations maturity are in place. In reality, however, Dev and Ops have conflicting priorities, e.g., rapid change vs. stability. The two areas must define "maturity" in the same language. Service operations discipline (ITIL, COBIT in action) is a prerequisite for operations to claim its side of maturity and robust engineering and quality practices (CMMI in action) is a prerequisite for development to fulfill its side of the promise. Put together, these mean maturity for:

- Understanding and controlling the flow of work relevant to the business context,

reducing cycle time, managing constraints and preventing handoff of defects downstream.

- Eliminating wait times and enabling the feedback loop to return to the earliest part of definition, design and development.
- Resilience engineering by continuously injecting tension into the system to reinforce habits and improve performance.

Throwing technology tools into lifecycle automation will not be fruitful without an appreciation of each type of maturity. Technology injection in a black box development and operations environment will lead only to "blackouts," however effective the integration and deployment automation technology chosen.

In short, technology adoption for agility does not mean eliminating total quality management (TQM) practices. Rather, it is the other way around: TQM maturity is a prerequisite for technology-driven agility adoption in a DevOps ecosystem. ITIL disciplines become integral to continuous delivery much in the same way as CMMI disciplines into continuous integration (CI) are the foundation of the broader DevOps practices.

Collaborative Operating Models

Developing collaborative operating models for "continuous everything" is the most difficult readiness dimension to address. Almost all organizations falter on their first attempts to address the cultural changes associated with the DevOps operating model and its consequent need for organizational silos to be broken down.

Common Stumbling Blocks to Adopting DevOps

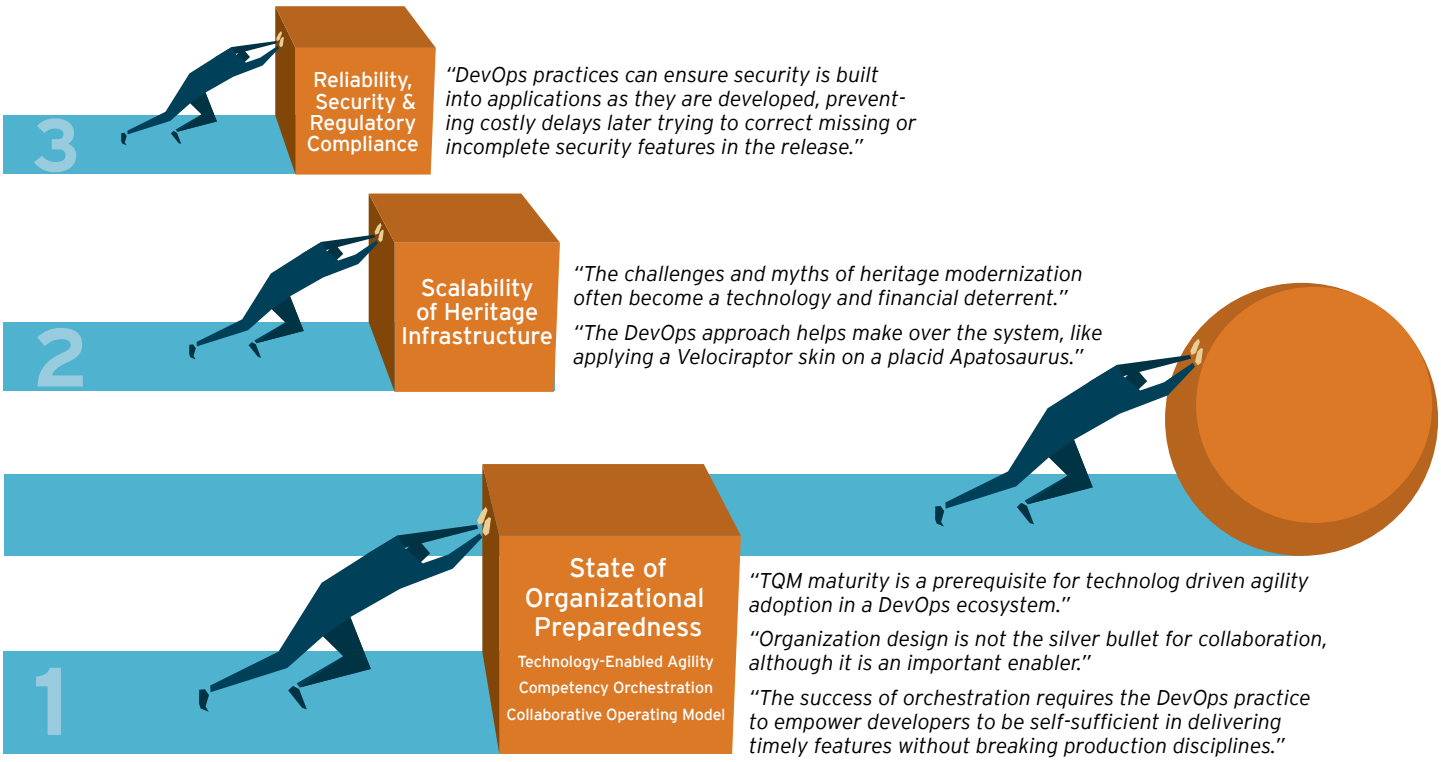


Figure 1

Organization design is not the silver bullet for collaboration, although it is an important enabler.

Third-party IT service delivery has made this a larger challenge because the sourced and/or offshore elements increase compartmentalization and accountability demarcations within the organization and providers. For collaboration to truly work in an Agile environment with a continuous everything mindset, the following steps are necessary:

- Breaking down silos and establishing a multidisciplinary group with a shared vision across internal and supplier staff guided by a common understanding of the business value stream enabled by IT.
- Extending development practices into operations and operations disciplines into development. Multiple iterative Scrums should have a single repository of truth with trail-based quality gates, standardized build and deploy processes, regimented operations discipline, simulated production readiness and assurance in development. The development Scrums should include operational simulations to break things early and often with real-time feedback.
- Establishing a single point of accountability and cohesive operating model across a meshed organization design.

Cooperation must give way to true collaboration. With expertise that is an inch wide and a mile deep required, no functional group or supplier can expect to make the transition to a DevOps practice alone because solutions exist within functional and technology intersections that span multiple groups. Organizations and functional groups need to adjust their mindsets and operating models to allow them to borrow with pride, build further and collaborate on a revised offering.

It is only natural that success in the continuous-everything DevOps model needs continuous working collaboration. There is no room for siloed outcomes; each work center or specialization must have a bidirectional collaboration with its predecessor and successor.

Organization design is not the silver bullet for collaboration, although it is an important enabler. Collaboration must happen without the artificiality of management leading the "trust-fall" sessions with the team. Some practical enablers include:

- A behavioral orientation that any information or work is collective, not singular, and success is interdependent. The efficiency of collaboration becomes a leading indicator. Look out for contracted work that is not interlocked with this behavioral orientation.
- Componentized knowledge organized as a central knowledge bank with a visual "work map" of associated work centers and interdependencies. This is akin to a map of a mall or park, on which the viewer's location is marked "You are here" for orientation and quick identification of the trails to a desired destination.
- Strong culture of inclusion, trust, empowerment and feedback. Development needs operations' inclusion to enable "design for operations" and operations needs development's inclusion to ensure "operations work; don't go backward"; that is, no rework or unplanned work.

Most organizations stumble at this block as they struggle to manage culture, cooperation, compartmentalization and contracts to enable collaboration for continuous everything.

Competency Orchestration

Orchestrating the competencies necessary to transition to DevOps is often misinterpreted as platform enablement, and therefore it becomes a pure technology conversation exercise. It should instead be seen as the seamless bidirectional integration of the system of engagement to the system of records through a fully meshed competency orchestration involving process, technology and service solution tenants addressing the business need. For orchestration to make a successful transition, this would require:

- A competency framework that is not purely technology-centric but service-aligned, incubating DevOps practices into the technology and process elements of service provision. It identifies the DevOps roles and associated technology, process and business service tracks.
- A unified DevOps practice that is capable of dealing with a large and diverse application portfolio in a consistent manner.
- An easy-to-use integration and measurement model across the infrastructure and services stack — and no islands of automation. Version everything, track and plan everything, automate everything, audit and monitor everything.
- A service catalog, with customizable service offerings aligned to the development, integration and deployment technology platform. This is an important prerequisite and an enabler for the integration of DevOps tools.
- A workable minimum viable product (MVP) model for each feature introduction.
- A full stack continuous monitoring and management model with communication and notification becomes as important as execution across the release lifecycle.
- Insights leveraging feedback to enhance user experience.

The success of orchestration — and ultimately of DevOps — requires the DevOps practice to empower developers to be self-sufficient in delivering timely features at desired quality without breaking any of the production disciplines. The qualities outlined above are fair indicators of such self-sufficiency.

SCALABILITY OF HERITAGE INFRASTRUCTURE

Heritage infrastructure and software platforms frequently seem like plodding dinosaurs in today's fast-paced digital world. Often, the DevOps implementation conversation falters when confronted with the hard reality in most businesses that their mission-critical systems are running on those infrastructure and software platforms. Fortunately, it's equally true that DevOps can enable organizations to more efficiently modernize these heritage systems, which should spark lively discussions about the following:

- **Adapt the dinosaur to the modern world.** DevOps practice adoption for heritage infrastructure may mean enabling plug-ins to tap the core system rather than reengineering or retiring the heritage platform. In these cases, the DevOps approach helps make over the system, like applying a Velociraptor skin to a placid Apatosaurus. There are many compelling reasons to take this approach, including:

- The world's most valuable code still runs on heritage platforms (Cobol, PL/I, etc.) whether for ordering pizzas via smartphones or global banking platforms. Reengineering this code would be a nightmare.
- DevOps practices absolutely apply to heritage (e.g., mainframe) development and maintenance with minimal modification. While there are conditional differences based on heritage software coding patterns, there is enough tooling to circumvent these idiosyncrasies.
- In light of mainframes' cost-effectiveness, scalability and reliability, a DevOps-practice-enabled mainframe becomes a significant business advantage that should not be left untapped.

- **Make over dinosaur sub-services.** Cosmetics also must be applied to those discrete sub-services that constitute the DevOps ecosystem. It is important to ascertain what elements of these discrete sub-services need a makeover and how to approach these as abstraction models tightly coupled to the heritage infrastructure platform. Approaches include:
  - **Agile programming platform:** A pragmatic approach to applying DevOps practices is to adopt a hybrid model with the combination of Agile practices on the heritage platform and integration of modern platforms with legacy systems. Product vendors have accelerators and integrated development environment (IDE) kits that accommodate DevOps practices. These enable integration of heritage with modern software platforms with a single interface for developing, debugging, testing and deploying code.

    Development tool kits enable developers to easily understand heritage application interdependencies and abstract heritage code to modern programming concepts. MVP-based Agile feature development is an equally acceptable concept on the heritage platform. Leveraging the reliability and efficiency of Cobol in a new development and delivery framework is the epitome of heritage modernization.
  - **Enterprise continuous delivery:** Continuous integration and delivery is as much a reality on a heritage platform as it is with a modern digital platform when applying DevOps practices and associated tool suites. Developers adopt DevOps practices in creating RESTful APIs[1] from traditional z/OS[2] assets, enabling the back-end heritage business-critical transactions to be conducted on cloud platforms and mobile channels.

    Automated continuous integration is enabled through an integrated pipeline[3] for code/test/platform configuration and self-validating builds. Automated continuous delivery is enabled through continuous testing across the delivery pipeline (functional, API, UI, performance data) powered with X86[4] emulation capabilities to optimize mainframe MIPS[5] and on-demand capabilities for environment provisioning, operations and retirement, whether deployed as physical, cloud or virtualized (stubs).

» **Enterprise release orchestration:** Zero touch deployments and Agile release orchestration exist on heritage and mainframe platforms through DevOps practices adoption. A standardized source-to-image framework[6] eases deployment, with control gates managing audit trails, versioning and approvals. Heritage tool suites can enable elasticity by automating the application environment on the cloud. Native mainframe deployment support there is complemented by cross-platform deployment capabilities and extensive third-party integration accomplished through open software lifecycle collaboration. Automated process enforcement reduces risks and rollbacks to meet audit and compliance mandates.

» **Insights and amplification:** Although still evolving, there are tool sets and practices for the heritage platform to enable lifecycle feedback amplification of application and infrastructure insights. Application performance management (APM) and DevOps practices provide monitoring and management solutions ranging from synthetic monitoring, alert management and delivery pipeline through CI/CD, deployment automation, release orchestration and log management/analytics. The insights drive the feedback amplification process to enrich the user experience in a continuous iterative cycle.

The pragmatic approach is to leverage an API to integrate heritage monitoring and management information with modern digital capabilities for lifecycle feedback amplification.

» **Fabric (infrastructure and device) as code:** The relevancy of fabric-as-a-code[7] on heritage platforms relates to infrastructure provisioning (on

Adopting DevOps: Heritage Environment

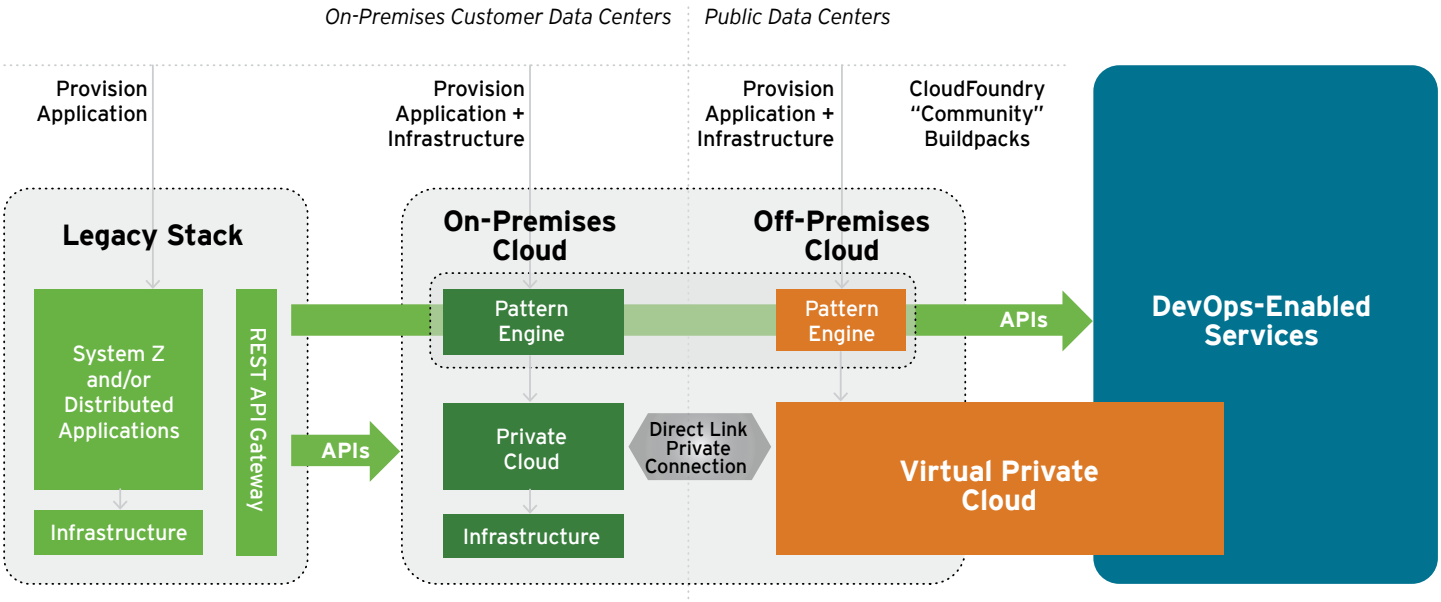


Figure 2

premises, cloud or hybrid). There is surprising maturity in the APIs on heritage platforms; in fact, oftentimes integration is based on a set of well understood and easily accessible connectors based on REST/JSON[8] patterns, allowing cross-platform leverage. This allows only the relevant microservices that need agility to be exposed in an "as a code" construct. RESTful APIs are available from one end point to existing heritage platforms' (e.g., mainframes) subsystems and data.

RELIABILITY, SECURITY AND REGULATORY COMPLIANCE

While the DevOps methodology is known for accelerating product and service development, its speed raises a stumbling block. Some practitioners believe that DevOps' speed and continuous development, delivery and integration can somehow compromise system reliability, security and compliance. Yet there are answers to each of these concerns, including:

- **Adopting resilience vs. reliability:** In DevOps, reliability must give way to a richer concept, that of resilience. Yet resilience is a paradox with regard to TQM. In an attempt to stabilize a complex system by reducing variation, the system becomes less resilient to unexpected events. In DevOps, the attempt is to create an adaptive system with highly independent agents and a dynamic structure, where the focus is on resilience and variation. Stability should be engineered into the variance (rather than forcing a trade-off between the two). The approaches to creating resilience include:
  - » **Fail early, fail often:** Simulated failures early in the lifecycle can provide insights to help develop more resilient processes, tools and capabilities.
  - » **Data management strategies:** These should be distributed, fault tolerant — and in some cases, even self-healing, spawning nodes as needed.
- **Extending DevOps practices to security operations:** Security practitioners are a frustrated lot in many organizations, often being branded "speed breakers" by the development community. Information security professionals often think the DevOps paradigm of "continuous everything" will lead to overlooked security issues. The answer to these concerns is to extend DevOps practices to security operations to maintain development momentum while also addressing security issues. Some tactics to consider:
  - » **Make security operations "code-able."** Inject code analysis tools into the development process, automate attacks against preproduction code and environments, and conduct continuous penetration testing.
  - » **Turn the adversary to an ally.** Security professionals with the right security automation and operational tools expertise may become an integral part of the developer community rather than just an audit entity.
  - » **Foster the mindset that security and DevOps are complementary.** Businesses want to accelerate time to market while maintaining resilience and security. DevOps practices can ensure security is built into applications as they are developed, preventing costly delays later trying to correct missing or incomplete security features in a new release.

- **Coding the compliance question:** Regulatory requirements are so far-reaching that even without DevOps, full compliance is a myth. It is normally a tradeoff between "it's good to go" vs. "we don't go until we have crossed every t and dotted every i." Compliance concerns can effectively reduce development velocity, which goes against the DevOps practice of agility. The way to shift the compliance paradigm is to think of compliance as being code-able. Consider the following:
  - » **Extend the developer mindset into the audit, compliance and risk function.** Developers should start seeing these controls as work packages to be "codified" at the appropriate phases of the lifecycle.
  - » **Ensure only relevant compliance suites are added bases for the change scope of the Scrum.**
  - » **Simplify compliance governance and reporting.** Build control into the developer work practice and avoid proving compliance just because it's a row item on a checklist.

LOOKING AHEAD: NEXT STEPS

We recommend the following to overcome the three main stumbling blocks organizations encounter as they embark on DevOps adoption.

- **Identifying which obstacles your organization may encounter is akin to developing a "You are here" map.** The map should clearly delineate the aspirational level of DevOps adoption in relation to the current state of maturity, thereby building a contextual roadmap for DevOps adoption and a clearly defined point of arrival. Mapping the route should include regular assessments of organizational preparedness and cultural orientation, to enable changing mindsets and breaking down silos; the extent of heritage systems; and an evaluation of reliability, compliance and security issues likely to arise in the target state. This exercise will help the organization appreciate and set appropriate priorities for overcoming those obstacles and creating a smoother path for broader DevOps implementation.
- **It is prudent to also apply the Scrum-based DevOps development principles to DevOps adoption.** Identify small, iterative "quick win" bundles among platforms and applications that have smaller maturity gaps, or new applications that have limited dependencies, and then extend and repeat the experience in quick succession. Further, DevOps adoption implies an enterprise-wide change in working models. This includes IT partners and contracted functions that should not be missed in the roadmap or the associated adoption Scrums.

With a roadmap delineating the most likely DevOps obstacles, the organization can begin the adoption journey well prepared to overcome them, implement DevOps more widely, and begin transforming the speed and quality of its applications and services delivery.

FOOTNOTES

[1] Web services that use REST architecture are called REST APIs (Application Programming Interfaces). REST stands for representational state transfer, an architectural style used to build lightweight, maintainable and scalable services.

[2] z/OS is a 64-bit operating system for IBM mainframes.

[3] Integrated pipeline mode enables requests to be handled through a unified pipeline. This is enabled by the integration of development platform runtime with the web server.

[4] x86 is a family of backward compatible instruction set architectures based on the Intel 8086 CPU.

[5] For mainframes, million instructions per second (MIPS) is a way to measure the cost of computing: the more MIPS delivered for the money, the better the value.

[6] Source-to-image (S2I) is a framework that makes it easy to write images that take application source code as an input and produce a new image that runs the assembled application as output.

[7] Fabric (infrastructure and device) as code is the process of managing and provisioning computing infrastructure (processes, bare-metal servers, virtual servers, etc.), devices and their configuration through machine-processable definition files, rather than physical hardware configuration or the use of interactive configuration tools.

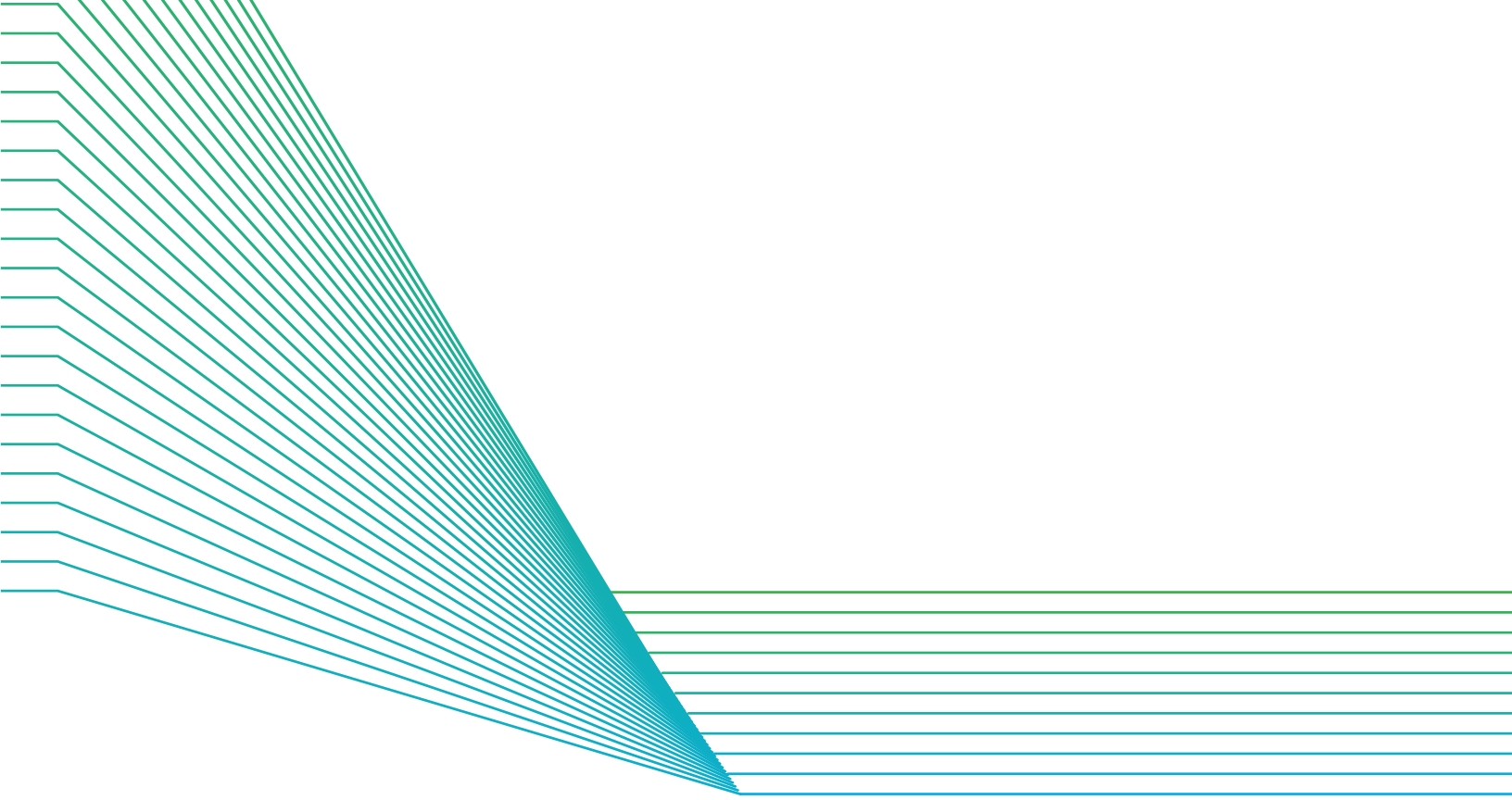
[8] JSON (JavaScript Object Notation) is a lightweight data-interchange format utilizing REST architecture.

ABOUT THE AUTHOR

Krishna Kumar Kezhakkevetil

Senior Director of Delivery, Cognizant Infrastructure Services

Krishna Kumar Kezhakkevetil is a Senior Director of Delivery within Cognizant Infrastructure Services. He has over 24 years of experience in IT across infrastructure and application services in senior leadership roles for service organizations, captives and front offices. In his current role, Krishna leads the DevOps practice for CIS in Cognizant. He has a bachelor's degree in electronics and an M.B.A. in Operations Management. Krishna can be reached at Krishnakumar.Kezhakkevetil@cognizant.com | LinkedIn: www.linkedin.com/in/krishna-kumar-7893484/.



ABOUT COGNIZANT

Cognizant (NASDAQ-100: CTSH) is one of the world's leading professionalservices companies, transforming clients' business, operating and technology models for the digital era. Our unique industry-based, consultative approach helps clients envision, build and run more innovative and efficient businesses. Headquartered in the U.S., Cognizant is ranked 230 on the Fortune 500 and is consistently listed among the most admired companies in the world. Learn how Cognizant helps clients lead with digital at www.cognizant.com or follow us @Cognizant.

Cognizant

World Headquarters
500 Frank W. Burr Blvd.
Teaneck, NJ 07666 USA
Phone: +1 201 801 0233
Fax: +1 201 801 0243
Toll Free: +1 888 937 3277

European Headquarters
1 Kingdom Street
Paddington Central
London W2 6BD England
Phone: +44 (0) 20 7297 7600
Fax: +44 (0) 20 7121 0102

India Operations Headquarters
#5/535 Old Mahabalipuram Road
Okkiyam Pettai, Thoraipakkam
Chennai, 600 096 India
Phone: +91 (0) 44 4209 6000
Fax: +91 (0) 44 4209 6060

TL Codex 2630