White Paper

BROADCOM
MAINFRAME SOFTWARE

Mainframe Site Reliability Engineering

The SRE Role in a Mainframe Infrastructure Scope

Table of Contents

Abstract

Site Reliability Engineer (SRE) is a job role and a set of principles and practices focused on ensuring the reliability of mission-critical systems. SREs apply engineering and systems thinking skills to make the production environment more robust. Since its inception at Google around 2003, it has become one of the most popular industry standards for managing high-availability environments.

In this white paper, we will examine the role of the SRE as it was originally conceived, its tenets, and the skills and practices that make up the SRE persona. We will then explore its implications for Mainframe services management in counterpart to the traditional implementation for distributed systems. Finally, we will assert the importance of this job role for the Mainframe as the next logical step towards platform modernization and integration into a more hybrid-cloud approach.

Some principles discussed in this document are universally applicable to both mainframe or distributed systems. However, we will explicitly identify where these concepts might be different. It is important to highlight that whenever we refer to the mainframe platform we are primarily focusing on z/OS systems due to the presence of z/OS on the majority of mainframes today. We also recognize that the architecture of z/OS systems differs more significantly from that of distributed systems when compared with other mainframe operating systems like Linux on Z (zLinux).

Dawn of SRE

It is well known that the majority of the cost of a software system is not in its initial development stage but its ongoing maintenance. It is also widely accepted that no matter how automated the system, there will always exist a certain amount of human intervention and maintenance. This is especially true for complex, data-intensive systems that run many critical production services.

In this section, we will explore two approaches for software services management; the traditional approach and the SRE approach as it was originally conceived by Google. In later sections, we will discuss how the latter can be applied to the traditional mainframe service management model, and we will highlight some of the similarities, differences, and ultimately, the benefits of adopting the SRE model.

kyndryl

The Traditional Service Management Approach
Historically, even before the need to manage large and complex applications, humans needed to manage complex systems. For this reason, the discipline of operations was established. This discipline sets rigorous standards and rules of engagement for the management of such systems.

For decades the IT industry has adopted the same approach to infrastructure and service management. In this document, we will refer to this approach as the *sysadmin* approach. It consists of hiring and managing teams of *system administrators* that will be responsible for the IT infrastructure management of a company.

These professionals are responsible for assembling software components and deploying them to work together to produce a service. Along with that, the sysadmins are tasked with running these services and responding to events as they occur.

As the system grows in size and complexity, the amount of required maintenance labor grows, causing the sysadmin team to also grow to absorb the additional work. Moreover, it is not only the amount of work that increases but also the expertise required to do that work. This ultimately leads sysadmin teams to split into more granular teams with more specific areas of expertise such as database administration and storage support, to mention a few.

Since the kind of work that sysadmins and other traditional infrastructure management teams do usually requires a different set of skills than that required to build products and services, developers and sysadmins are divided into two discrete categories: development and operations.

This approach to service management has several advantages. Since it is a familiar industry paradigm, a relevant talent pool is already widely available along with extensive knowledge, tools, practices, and standards that companies can quickly implement.

At the same time, it also has its pitfalls. This hard segmentation between Dev and Ops teams also means that they both have diametrically opposing incentives. On one hand, development teams are encouraged to release new features faster and frequently to attend to new customer demands, while on the other hand, operations are guided by the stability of existing systems, where every new change represents a potential outage risk to production services. This creates a constant conflict between these groups, a lack of clear communication, and consequently an unreliable and often toxic working environment, one in which both teams resort to a form of trench warfare to advance their interests.

Moreover, it's not only the amount of work that increases but also the expertise required to do that work.

The SRE Approach
As a counterpart to this traditional methodology, and to address the challenges of the segmentation between development teams and operations, there is the SRE approach.

On one hand, development teams are encouraged to release new features faster and frequently to attend to new customer demands, while on the other hand, operations are guided by the stability of existing systems, where every new change represents a potential outage risk to production services.

As previously mentioned, the SRE is a concept that originated at Google around 2003. According to the Site Reliability Engineering book, the SRE is a job function where software engineers are tasked with activities that would traditionally fall under the operations umbrella, with a special focus on the reliability of mission-critical applications and services. This approach leads to teams comprised of people who quickly become bored by performing tasks by hand, and have the necessary skill set to automate their previous manual work.

The concept of the SRE originates from the premise that everything is software, not only applications that generate value to customers but everything, including the underlying infrastructure. All applications will eventually run on physical hardware, but until then, everything depends on software. As such, managing the IT infrastructure becomes a software engineering problem. This realization is also one of the key ideas of DevOps as we will see later.

It is important to clarify that although it is common in the industry to treat the software engineer and software developer roles as the same, in the context of this white paper, we are treating them as two separate personas. The reason for this is merely to make explicit the potential differences in the skillsets of both professionals.

On one hand, the software developer is a professional that is responsible for transforming business needs into working software through coding. However, this professional does not necessarily need to have a profound understanding of the underlying systems in which the application will run, nor must it be an expert on computer science concepts. On the other hand, the software engineer is a professional that is also capable of transforming business needs into working software, however in this case this professional has a wider range of skills in computer science and is capable of working at a lower level on system-wide improvements and algorithm optimizations.

For example: While coding a business application, a software developer might need to sort a large list of values. In most cases the developer will not need to implement a sorting algorithm from scratch. Most high-level programming languages include built-in abstractions that can solve problems like sorting very efficiently. However, in some scenarios where performance is absolutely critical, there might be a need to do algorithm optimizations to solve this and many other common computing problems. This is where computer science skills become a requirement.

With this in mind, SREs are, first and foremost, engineers. They are professionals that apply the principles of computer science and engineering to the design and development of computing systems. Next, there is the focus on reliability. Understanding that it is the most fundamental feature of any product, SREs focus on finding ways to improve the design and operations of systems to make them more scalable, reliable, and efficient.

The concept of the SRE originates from the premise that everything is software, not only applications that generate value to customers but everything, including the underlying infrastructure.

The Relation Between SRE and DevOps

DevOps and SRE are sometimes treated as two completely different disciplines, however, they are actually very much related. DevOps principles including breaking silos, heavy reliance on automation versus human effort, and the application of engineering concepts to operational work are all present within many of the SRE principles and practices. In fact, DevOps could be perceived as a wider philosophy and culture for the entire enterprise organization, while SRE being a specific implementation of DevOps with some extensions.

DevOps principles including breaking silos, heavy reliance on automation versus human effort, and the application of engineering concepts to operational work are all present within many of the SRE principles and practices.

In many organizations, there is the role of a DevOps Engineer or simply DevOps. Depending on the size of the organization, this engineer might be solely focused on the software delivery pipelines, a function that is also commonly referred to as release engineering. Meanwhile the SRE focuses on the reliability and continuous improvement of the underlying infrastructure. In other cases, the SRE can absorb the function of the DevOps Engineer as well.

The SRE Tenets

While there are nuances within the priorities and day-to-day activities of different SRE teams in an organization, they all share a common set of core tenets. These tenets can be summarized as follows:

- **Ensure focus on engineering:** A fundamental aspect of the SRE role is that, by definition, they need to focus on engineering work, otherwise they will be swallowed by operations work. As a rule of thumb, SRE teams should cap their operational load by 50%. This means that, ideally, they should spend no more than 50% of their time doing operations labor. This includes incident response, changes, and any traditional system administration and management activities. The other 50% should be spent on engineering work to either automate the operational activities, eliminate toil, or to make the system more stable.

  In practice, this is accomplished by monitoring the operational work of SRE teams and redirecting any excessive manual workload to product development teams, and by integrating developers using on-call rotations. The redirection ends when the operational load drops back to 50% or lower. This shared ownership also creates an effective feedback mechanism, inspiring developers to build systems that do not need manual intervention.

  Blameless postmortems should be written for all significant incidents, even if it did not result in a team or member being paged. In fact, postmortems that did not trigger a page are even more important, as they are likely related to a monitoring gap. The

focus of the postmortems should be to investigate the truth of what happened in detail, and assign actions to correct the problem or improve how it is addressed next time. The blameless aspect fosters a culture of truth discovery rather than blaming individuals, creating incentives for people to come forward whenever they make a mistake instead of creating a punitive environment.

- **Enable maximum change velocity without violating service level objectives:** This tenet aims to eliminate the structural conflict from Dev and Ops and the pace of innovation and product stability by the introduction of the concept of *error budget*.

  The error budget is a principle of applying a consequence for service level objectives (SLOs). Consider that the SLO for a given service is for 99.99% of availability, this means that it can only be 0.01% unavailable. This permitted 0.01% unavailability is the error budget. Development teams can spend this error budget on anything as long as they do not overspend it. It is ideally spent taking risks, implementing new features to products and adding value to customers. This approach enables yet another self-regulation mechanism that allows development teams to ship code fast, without needing to fight with the operations teams.

  The introduction of an error budget concept also promotes clever approaches and optimizations to release management, like phased rollouts, where the changes are applied first to only a small portion of the users, minimizing the impact of any outages and therefore the budget spent.

  The error budget assumes that 100% is the wrong reliability target for almost anything. In general, the users of any software service cannot tell the difference between a system being 100% available and 99.999% available. There usually exist many other systems in the path between the user and the service and those systems collectively are far less available than 99.999%, meaning that the difference between these availability rates is lost in the noise of another unavailability.

- **Monitor:** Monitoring is the basis of reliability. When a rocket is launched telemetry is what enables us to understand if everything is working as expected—the trajectory, the speed, and so on. Similarly, in software, if you are not monitoring a service, you do not know what is happening, and if you are blind to what is happening you cannot be reliable.

  A classic and common approach to monitoring is to watch for a specific value or condition and then trigger some sort of alerting (usually email) when that value is exceeded or the condition occurs. However, this type of alert is not usually an effective monitoring solution. A system that requires a human to read and interpret the alert to decide whether or not an action should be taken is fundamentally flawed. Monitoring should never require a human to interpret any part of the alerting domain. Instead, software should be responsible for interpreting, and humans should only be notified when an action should be taken, or when an automated response has already occurred.

  Another fundamental aspect of monitoring is the understanding that alerts are, in general, closely related to symptoms—not causes. Therefore, a mature monitoring strategy should consider both.

- **The Four Golden Signals:** In general, the four golden signals represent the basic indicators for a decent monitoring of a software service.
  - **Latency:** The time it takes to process a request
  - **Traffic:** A measure of how much demand is being placed on the system.
  - **Errors:** The rate of requests or processes that fail.
  - **Saturation:** Basically, how full the service/system is.

- **Emergency Response:** Reliability can be calculated as the function of mean time to failure (MTTF) and mean time to repair (MTTR), where the most relevant metric in evaluating the effectiveness of the emergency response is how quickly the response team can bring the system back to a state of health.

  Humans add latency to any process. Even if a given system experiences more failures, if it can avoid emergencies that require human intervention it will have a higher availability than systems that require constant hands-on intervention. When humans are ultimately necessary, a well-established framework of knowledge sharing with thoughtful procedures and best practices produces huge improvements in MTTR. In addition to that, constant training and exercises as the “wheel of misfortune” are essential to prepare engineers to react to on-call events. While no playbook, no matter how comprehensive, is a substitute for well-versed engineers able to think on the fly, a prepared on-call engineer strategy works better than having a jack-of-all-trades on-call person.

- **Change Management:** The majority of production outages occur as a consequence of changes. Automations in this domain are a key necessity, especially in the following areas:
  - Progressive rollouts
  - Quickly and accurately detect problems
  - Quickly and safely roll back changes when problems arise

By removing humans from the loop, these practices avoid the normal problems or fatigue, familiarity/contempt, and inattention to highly repetitive tasks.

- **Performance and Capacity Planning:** Understanding current demand, projecting future demand, and ensuring proactively that sufficient capacity exists to support the requirements for availability are essential responsibilities of the SRE.

Final Considerations on the SRE Persona

This section introduced the concept of an SRE, its main responsibilities, practices, and how it represents a significant break from the traditional industry model for managing large data-intensive systems.

Although many of the concepts and technologies are new, for mainframe professionals, a good part of what has been presented is not entirely new. This is because the mainframe industry has been practicing reliability since its origins and continues to do that to this day. In fact, there is a good chance many companies already have professionals that fit this role but are simply not calling it SRE.

The next sections will provide more insights on the role of an SRE, but now for the mainframe perspective.

The Mainframe SRE

Why SRE for Mainframe?

Before we navigate through the possibilities of the SRE role in the mainframe, let's first discuss the reasons why we believe that this concept not only makes sense for the mainframe, but it is also an inevitable part of the modernization of how we approach platform service management.

It is a logical conclusion that the concept of a job role based on the reliability of mission-critical services must also be applicable to the most reliable computing platform on the planet. In the world of computing, no other platform has had the history and impact of the mainframe. Since its introduction, the mainframe has remained the backbone of the most critical and data-intensive applications in retail, finance, insurance, healthcare, aviation, and payment processing industries—just to name a few. The unparalleled reliability, availability, security, performance, and scalability of the platform along with data and software applications are continuing to power that backbone to this day and will do so for the foreseeable future.

The long history of mainframes, however, has contributed to the reputation of mainframes being labeled as *old* or *legacy*. The reality, however, is completely different. With emerging technological trends and related business requirements, the

It i s a logical conclusion that the concept of a job role based on the reliability of mission-critical services must also be applicable to the most reliable computing platform on the planet.

mainframe as a platform continues to evolve and re-invent itself in all aspects including hardware, software, pricing models, and most importantly culture and mindset.

The need for SRE is born out of a need created by a change in the way development is done, the Agile transformation, and the emergence of DevOps, which together enable a constantly increasing velocity of application changes.

Although the mainframe may have lagged behind the rest of the IT world when it comes to Agile and DevOps, it is clearly making huge advancements over the past several years. In addition to making the platform more secure with pervasive encryption, the mainframe has also evolved to support modern languages and frameworks like Node.js, Spark, Go, Python, REST/JSON, and containerization. Enterprises relying on the mainframe are going thru a cultural transformation to make it part of their digital transformation journey by embracing open source and opening up the mainframe applications and data to the new generation of tools, processes, and developers. One clear example of this transformation is Zowe, the first mainframe open-source project.

It is also clear that the requirement to change the way operations are done will reach the mainframe if it has not already. Therefore, it seems reasonable to conclude that the SRE role on the mainframe not only makes sense but is inevitable, and that the specific requirements of the SRE role on the mainframe can be assumed to be similar to the same role in the distributed and cloud world.

Another reason why the SRE concept should be applied to mainframe environments is the set of benefits that it can bring to how we manage its services. SRE teams should significantly contribute to reducing the MTTR of critical services, as well as support a comprehensive monitoring strategy for them. Moreover, SRE teams can also act as a catalyst for the continuous improvement and modernization of the platform.

Mainframe SRE Skills and Persona

While we learn more about the SRE in the IT world we keep asking ourselves how this professional would fit into a mainframe organization, especially in the infrastructure services or the Service Delivery teams. We highlighted some important questions that serve as our case study starting point to understand the role in the Mainframe context.

- What are the background skills of the SRE?
- What are the core skills of the SRE?
- Is it possible to build an SRE career path internally from my organization?
- Should I hire an external professional to be the SRE?

In the following topics we will contextualize these questions.

- **SRE Skill Model:** Let us start with the professional skill model. We can identify three types of professional skills. In the IT world, we usually have the I-shaped professionals, who are the expert in one thing, very present in most of the organizations where we see silos structured by competencies and specific skills. The second professional type is the Generalist. There have been some movements to create the Generalist—professionals capable of doing a lot of things, but not really an expert in any of them. Finally, the last type of professional skill is T-shaped—capable of doing a lot of things and an expert in one of them.

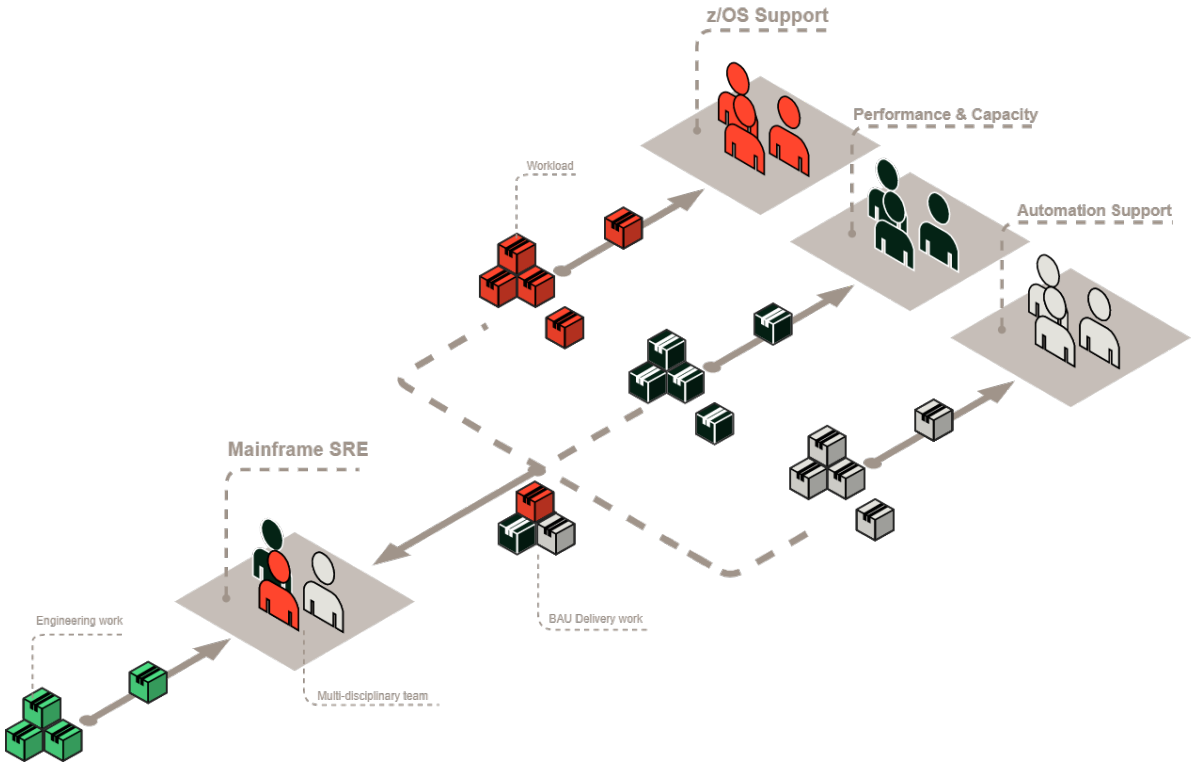
To build an SRE professional, it is essential to be focused on a T-Shaped model. It is crucial to have strong expertise in one critical area, usually the original background skill. Originally in a distributed environment the SRE usually starts with a development/engineering background and then builds the horizontal trait of the T—looking for broader knowledge in the infrastructure operational skills.

However, in a mainframe environment, considering the applicability context of this white paper (MF Infrastructure – Platform – Service Delivery Management) we believe that it should be the other way around. Considering the complexity of mainframe infrastructure skills, we suggest building an SRE starting from professionals with operational background skills.

As you might recall from the Dawn of SRE section the core activities for an SRE would vary from Engineering and Operations, however, it is essential to ensure that an SRE works at least 50% on incidents.

As another suggestion, you can still balance the background skill from your squad by adding fewer professionals with an original background in development/engineering and enabling them in the operational infrastructure side. However, this will require more effort from the professional based on mainframe complexity particularities.

Create successful SRE teams from individuals who possess a broad range of mainframe skills.



Considering the complexity of mainframe infrastructure skills, we suggest building an SRE starting from professionals with operational background skills.

In summary, as per the three skills models listed below, you can think about what is the predominant skill type that your organization is based on.

- I-shaped professional: An expert in one thing.
- Generalist: Capable in a lot of things, but not an expert in any.
- T-shaped practitioners: Capable in a lot of things and expert in one of them.

T-shaped is a new way of learning where you combine a deep understanding of a foundational skill where you are the expert with a broad range of knowledge. T-shaped teams can view problems from various mindsets and competencies and resolve them at a faster speed. That's the core skill model required for a professional looking for SRE skills

- **SRE Core Skills:** There are many skills required to build an SRE professional, this is not specific to mainframe only, and it is not technology wise. Below the core skills are comprised in six main areas:
  - Data analysis: Mathematical and statistical models to assess trends, data-driven/scientific approach to fact-finding, monitoring and event management.
  - Software engineering: Provide solution optimization, innovate breakthrough solutions, and develop strategic plans.
  - Platform skills: System thinking for reliability, troubleshooting, monitoring and event management, security.
  - Tools skill: Deploy and release services, tools, and technology across the SDLC.
  - Process skills: Organizational knowledge, DevOps and Agile principles and practices, and engage thinking at strategic levels.
  - Leadership skills: Strong communication/ collaboration, a work ethic of caring about production systems, availability, and users, communicate on an executive level, employ cross-organizational leadership, employ collaborative influence, and engage in client projects on a leadership/trusted advisory level.

Mainframe SRE Team/Organization

First and foremost, we must make clear that there could be many different implementations of an SRE team for the mainframe environment. They will be heavily dependent on each organization's environment and needs. However, just like the traditional distributed services SRE teams might have different priorities and day-to-day activities, they all should share a common set of core tenets. Based on that, in this white paper we will explore the possibilities and propose a mental exercise on the composition of an SRE team and how it would operate.

That being said, regardless of the organization structure, we believe that any implementation of an SRE team for the mainframe should be composed by professionals from different backgrounds and skillsets, because a team with a diverse background results in higher-quality solutions, as the team will have a holistic view of the environment.

As previously established, one disruptive premise of the traditional approach for distributed services SREs is that only developers/engineers should be hired into an SRE function. We believe that this should be approached differently for mainframe environments. The reason for this is that the premise of a distributed system SRE is that it is easier to teach a software engineer system operations activities due to the proximity of their skillset. For the mainframe we believe that, generally, there is a larger skill gap of traditional application development to infrastructure work, hence it should be easier to teach engineering skills to infrastructure support teams.

For this exercise, from the many different combinations of skillsets that could compose an SRE team for the mainframe, one of them is a team composed majorly of professionals from z/OS System Support, Automation, and Performance & Capacity teams. This choice of team composition comes from the expertise each of these professionals brings to the table and how they relate to the role that an SRE would play in a mainframe environment. For example, the z/OS System support has a deep understanding of the operating system and is capable of troubleshooting problems from a variety of sources, and it is also capable of performing general system administration tasks. The Automation support brings the knowledge of the mainframe automation technology stack, scripting knowledge, and alert management. Finally, the Performance support brings the knowledge of the mainframe performance and capacity management, monitoring, and metrics.

Of course, we must emphasize that this selection of skillset does not mean that other skills and areas of expertise are not valuable to the role of SRE, only

that from the range of skills available, we believe that these three cover a larger area of demands required to fill the SRE role. For example, depending on how an organization chooses to create the team, it could also be composed of experts in other areas like mainframe networking, security, and storage support. However, in some cases, a well-versed z/OS system administrator could also support these areas. In the end, it will all boil down to how the organization is structured.

Another possibility is to have the SRE team mixed with other SRE teams for the distributed platform. Although initially, they will remain siloed, the proximity between both teams under the same management structure means that they will be able to learn and share SRE skills and practices from each other.

What would be the work dynamic of such a team? Based on what we've learned about the site reliability engineering responsibilities and practices we can elaborate on the following scenario:

Just like in the traditional implementation, the SRE team would be responsible for running the mainframe infrastructure, however, we would cap their operational load at, ideally, 50%. The other half of their time would be spent on eliminating toil, making the system more robust, and other modernization projects.

During their operational work they would share responsibilities with the other infrastructure support teams. For example, they might work on tickets and changes from z/OS System Support, Performance, and Automation. However, the exceeding requests would be routed to the support teams themselves. The SRE team would also be involved in critical incidents, however, differently from the other support teams, the SRE would be involved whenever an application/service is impacted. This means that although the SRE team has a holistic view of the entire mainframe infrastructure, they are responsible for the health of the critical applications running on it. Ultimately the mission-critical applications running on the platform are what actually deliver value to the customer.

For this to work, the SRE teams must have a deep understanding of the critical applications running on the mainframe, and a complete monitoring strategy in place for the tasks, jobs, transactions, and any middleware infrastructure that support it.

Meanwhile, as we are managing a multi-disciplinary team, even though each team member might have a specific area of expertise, the team must conduct regular cross-training sessions early on so everyone in the team is ultimately capable of performing any task.

This means that although the SRE team has a holistic view of the entire mainframe infrastructure, they are responsible for the health of the critical applications running on it.

Mainframe SRE Toolchain

With the emergence of a new generation of mainframe SREs, the tooling that enables them to do their job effectively and efficiently is anchored by three key tenets: Open Source, Integrated Environments, and Simplification.

Back in 2013, the Open-Source survey concluded that "Open source is eating the software world". Fast forward to today, it has become a reality that many Enterprises are embracing. In line with that strategy, the new breed of mainframe SREs should leverage the power of open-source tools to modernize and optimize their workflow.

Integrated environments that enable SREs to do more with fewer tools are key to SRE efficiency. As discussed earlier, there are several tasks that an SRE is responsible for including software development, data analysis, monitoring, installation, and configuration, and so on. Having an integrated environment that allows SREs to do all those activities using one solution would improve workflow and efficiency.

Simplification and consolidation of the data and processes to monitor and do root-cause analysis would help an SRE be more effective at their job. With the emergence of AI-driven prediction models, it is important to consolidate alerts and present the relevant information in an easy to consume fashion.

The objective of this section is to provide a list of categories and technology choices that we believe a Mainframe SRE can leverage to perform their regular as well as ad-hoc tasks. However, it is important to highlight that this is not, by any means, an exhaustive list of tools and technologies.

Development Environment and Source Control:
Automation of manual process would entail writing code/scripts and managing them in a version control system for ongoing maintenance and automation. Open source and integrated environments are the tenets driving the tooling considerations. Several open-source tools and technologies are available:

... the new breed of mainframe SREs should leverage the power of open-source tools to modernize and optimize their workflow.

- **IDEs (Visual Studio Code, Eclipse Che):** 3270 terminal emulators have been the most popular IDE for mainframe code development, but with the Zowe-powered open-source revolution, new talent can leverage open-source IDEs like VSCode, Eclipse Che, and so on. These tools provide the flexibility for developers to use the rich client or a web-based interface to write REXX, JavaScript/Typescript, or Python code without having to jump to a different tool. For a modern mainframe SRE, a VSCode like development environment is the best tool to develop and manage the automation code.
- **Git:** Git has emerged as the de-facto standard in software version control. Git enables developers to work in isolation and integrate their changes with others when ready. Enterprise Git repos like GitHub, GitLab, Bitbucket, and so on. use the Git protocol and enable collaboration capabilities like pull requests on top of standard Git. All the popular IDEs provide Git integration for code development, and CI/CD tools like Jenkins and source scanners like Sonar have integrations with Git.

Automation: The daily bread of SRE is building and maintaining any automation that might be useful. Automation for a mainframe SRE could involve CI/CD pipeline automation, software install/maintenance automation, configuration management, auto-recovery, and manual task automation where everything is managed as code. Automation is a category that is driven by several tenets. Automation leverages open-source tools, enabling the integration of off-platform tools with mainframe processes resulting in simplification—a repeatable process with the push of a button. Managing configuration, infrastructure, policy, observability, and so on as code that can be versioned, reused, or reverted enables the SRE teams to achieve repeatable processes. There are several choices available for mainframe SREs to pick the right tool for the right automation job.

- **Zowe:** An open-source initiative under Open Mainframe Project enabling the integration of off-platform tools, open-source or otherwise, to integrate with the mainframe processes and tools. SREs can leverage the power of Zowe CLI, SDKs, and secure API ML to automate CI/CD pipelines using tools like Jenkins, CircleCI, and so on.
- **Ansible:** A framework developed for remote node management primarily with a focus on non-developers. It is particularly strong for inventory, credential, or task management. SREs can leverage Ansible to automate manual IT tasks. Ansible is supported on z/OS and can be the framework for managing hybrid environments that span mainframe and cloud environments.
- **z/OSMF Workflows:** Though not open source, z/OSMF workflows are emerging as part of the new standard for z/OS software management. From the technical perspective, a z/OSMF workflow is an XML file defining a set of steps (JCL, REXX, shell scripts) and variables that are used to guide users through a process. A workflow can be also executed without human interaction to enable automation of software and infrastructure management.
- **z/OS Automation solutions:** Again, not open source, but z/OS traditionally has automation solutions that monitor and react to system events and early problem detection. Broadcom OPS/MVS, BMC AMI Ops Automation for z/OS®, and IBM Z System Automation are a couple of examples of such automation solutions.

Performance Management: Performance Management includes monitoring, alerting, and quick triaging as a part of the SRE responsibility. Though there has been only limited uptake of open-source tools in this space, simplification and artificial intelligence are key drivers.

- **Monitoring:** Traditional resource monitoring tools provide insights into the health of a diverse set of resources, including subsystems, storage devices, and networks.
- **Dashboarding:** To provide better visibility into the overall performance of the system, there is a trend to move towards dashboards presenting relevant information at a higher level and enabling information from multiple sources to be combined to streamline root cause analysis. Technologies used here include Grafana, Prometheus, and Superset.
- **Proactive Issue Detection:** Increasingly, AI engines are used to detect issues before they become problems. SREs should tune these engines where possible to detect more and more issues as they shift from reactive to proactive management of their data center.
- **Alert Management:** The above monitoring and issue detection tools run the risk of overwhelming SREs and Operators with alerts fighting for attention. By centralizing alert management, SREs can define business rules to minimize noise and focus on the alerts that matter.

kyndryl

Final Considerations

Throughout this white paper, we have discussed the role of Site Reliability Engineering in the modern IT industry. We briefly described its origins, the context that led to its creation, the differences with the traditional sysadmin approach for systems management, its intersections and differences with DevOps, and the tenets that guide the SRE profession.

It is not a surprise that many of the concepts presented in this white paper are not entirely new to us mainframers. Reliability runs in our veins. The mainframe is a state-of-the-art technology that has been evolving and adapting to meet our customers' most bold technical demands with the utmost reliability and performance.

In this white paper, we have explored the possibilities around the role of the SRE in a modern mainframe ecosystem. We have laid out our arguments on why we believe that this role is not only a natural next step for our industry, but it is inevitable. We presented some of the possibilities, but at the end of the day, each organization will have to decide which path they will follow.

References

- The Site Reliability Workbook
- Site Reliability Engineering
- On Designing and Deploying Internet-Scale Services
- Tenets of SRE

About the Authors

Writing and other contributions from each of the following individuals were included in the creation of this white paper.

- Venkatauday Balabhadrapatruni, Broadcom
- Guilherme Cartier, Kyndryl
- Michael DuBois, Broadcom
- Per Kroll, Broadcom
- Greg MacKinnon, Broadcom
- Jan Prihoda, Broadcom
- Viviane Sanches, Kyndryl

BROADCOM®
MAINFRAME SOFTWARE

For more product information: broadcom.com

MFD-Site-Reliability-Engineering-WP100 April 27, 2022