A Broadcom Company

technologies

Site Reliability Automation

Empowering Site Reliability Engineering Teams to Reduce MTTR and Eliminate Operations Toil

Executive Summary

"It's like being part of the world's most intense pit crew...changing the tires of a race car as it's going 100 mph." That's how Andrew Widdowson of Google describes his role as a site reliability engineer.

Site reliability engineering (SRE) is emerging as one of the most critical roles in IT operations. The demand for mobile experiences and the advent of complex cloud architectures has shifted the operational focus. It's no longer about keeping the lights on. It's about performance. Apps have to work well, the experience has to be great, and the infrastructure needs continual monitoring.

SRE is at the heart of this vortex. In the SRE model, teams take a software engineering approach to IT operations. The goal is to manage a larger volume of changes, faster, and accept the risk of doing more changes into the infrastructure. That is why DevOps and SRE approaches work well together, and why SRE is often considered an extension to DevOps.

"DevOps teams must use site reliability engineering to maximize customer value."

— *Gartner*[1]

If you haven't yet deployed SRE, you are behind the curve. According to Gartner, "by 2023, 40% of DevOps teams will application and infrastructure monitoring tools with AIOps platform capabilities."

— *Gartner*[2]

1,2 Gartner, DevOps Teams Must Use Site Reliability Engineering to Maximize Customer Value, George Spafford, Manjunath Bhat, January 10, 2020

This eBook explores the background to SRE. This includes its definition and the main principles behind SRE: embracing the risk of change, understanding what is broken and why, and eliminating 'toil'. It also explores the major challenges to SRE, including the risks arising from the volume of changes, the difficulty of managing the noise generated by systems sprawl, and keeping mean time to repair (MTTR) in check.

The eBook moves on to reveal how site reliability automation (SRA) tackles these challenges, increasing the operational effectiveness of SRE teams. SRA is an integral part of the AIOps approach, which leverages analytics and artificial intelligence to ingest data and convert it into insight. SRA makes that insight actionable, removing manual work for the operations teams, minimizing their workload, and enabling them to concentrate on more value-added activities.

"In our 2019 DevOps survey, 41% of respondents have already adopted certain elements of SRE, and an additional 42% plan to implement SRE practices by YE20."

— *Gartner*[3]

The eBook closes with a series of use cases for SRA, including infrastructure healing, service restoration, integration with the enterprise toolchain, and compliance. SRA is powered by a recommendation engine, by contextual automation, and out-of-the-box workflows for remediation. Moreover, machine learning (ML) helps teams to choose the most effective workflow for issue remediation.

The key take-away? SRA efficiently reduces operations 'toil' and shrinks MTTR. Ultimately, SRA empowers DevOps initiatives by aligning infrastructure management with the pace of modern continuous delivery. SRE is becoming mainstream. In time, automation will make decisions and ensure consistent operations. Now is the time to review your automation strategy.

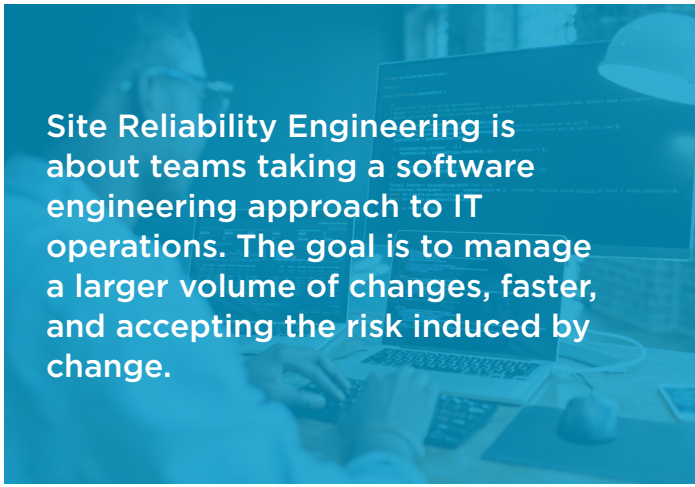
3 Gartner, DevOps Teams Must Use Site Reliability Engineering to Maximize Customer Value, George Spafford, Manjunath Bhat, January 10, 2020

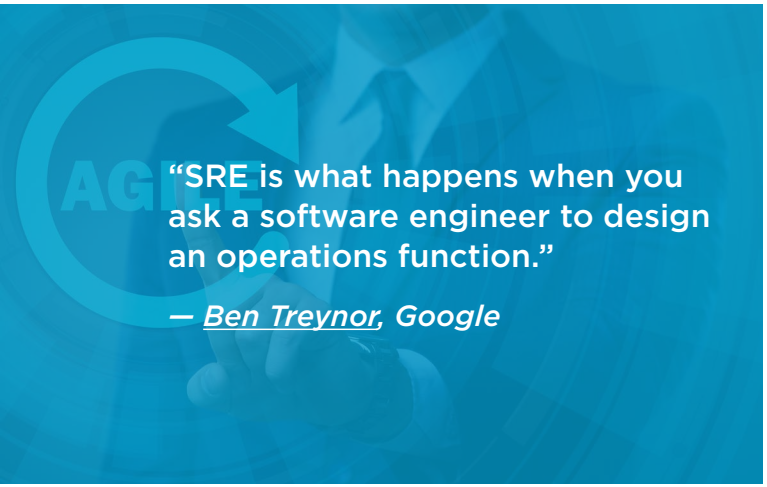
What is Site Reliability Engineering?

For every organization on a digital transformation journey, SRE models have an increasingly strategic significance. When employing SRE, teams take a software engineering approach to IT operations, developing service level objectives that are closely aligned with the business' most important key performance indicators (KPIs). This way, teams can establish the metrics, processes, and capabilities needed to improve service levels and business results.

However, SRE isn't a hotshot new technology model. It has been around since 2003, and is based on practices that Google implemented before DevOps became mainstream. The term SRA was coined by Ben Treynor, who founded Google's Site Reliability Team. According to Treynor, "SRE is what happens when you ask a software engineer to design an operations team. In general, an SRE team is responsible for the availability, latency, performance, efficiency, change management, monitoring, emergency response, and capacity planning of their service(s)."

Unpacking the term a little, people engaged in SRE are primarily engineers. They apply computer science and engineering principles to the development of large distributed computing systems. That may involve the writing of software, building the additional pieces those systems need, or working out how to apply existing solutions to new problems.





SRE also focuses on system reliability, finding ways to improve the design and operation of systems to make them more scalable, more reliable, and more efficient. Finally, SRE is focused on operating services built on top of distributed computing systems, whatever their size.

In summary, SRE is aimed at handling bigger volume of changes faster, and accepting the risk induced by change. That explains why DevOps and SRE approaches work well together, and why SRE is sometimes considered as an extension to DevOps.

SRE versus DevOps

SRE and DevOps co-exist as an essential element of the development team; so how can you distinguish one from the other?

- Like DevOps, SRE combines development and operation teams, helping them see the other side of the process, while introducing visibility to the complete application lifecycle.
- While DevOps is about the 'what' needs to be done, SRE is concerned with 'how' this can be done. It's about expanding the theory into an efficient workflow, with the appropriate work methods and tools. It's also about sharing responsibility and a common vision.
- DevOps and SRE teams are not so different. Both blend developer and operation teams, while sharing similar responsibilities and focusing on automation and reliability.

DevOps or SRE?

The core principles of DevOps – involving the IT function in each phase of system development, reliance on automation versus human effort, the application of engineering practices and tools to operations tasks – are consistent with many of SRE's principles.

DevOps	VS	SRE
Focus on continuous delivery		Focus on service management
Bridge organizational silos		Leverage tooling across teams
Accept failure as 'normal'		Accept risk on service levels
Implement iterative changes		Implement 'atomic' changes
Automate delivery toolchain		Automate standard operating procedures
Optimize time to market		Optimize mean time to repair (MTTR)

Figure 1: The convergence of DevOps and SRE

The main principles governing SRE

1. Embrace the risk of change

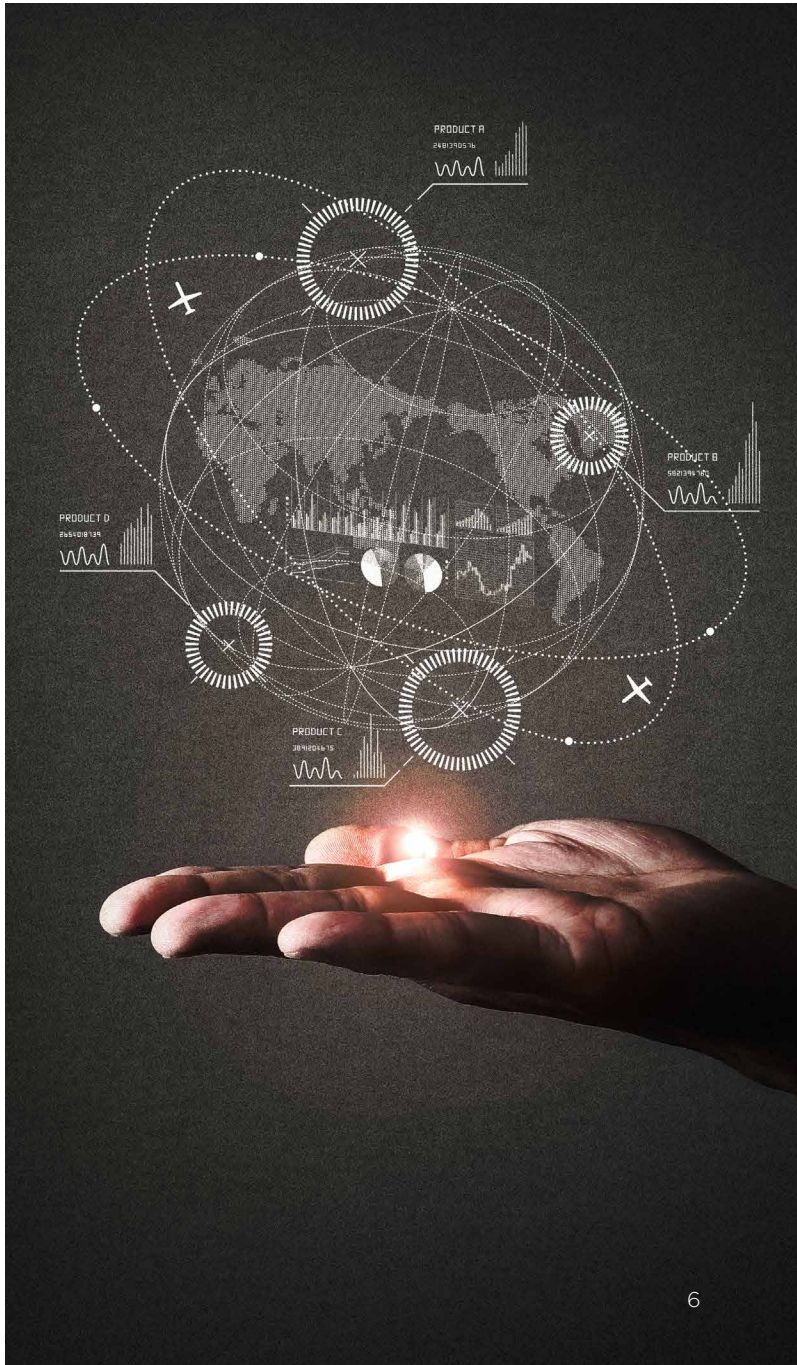
Fact: Failure happens. DevOps accepts failure, seeing it as an opportunity to learn and grow. SREs, meanwhile, have a formula to balance failures against new releases, ensuring the failures don't overwhelm the successes. This formula is measured with two key identifiers: service level indicators (SLIs) which measure the failures per request, and service level objectives (SLOs) which represent the success of SLIs over a period of time.

2. Understand what is broken and why

When failure does occur, you need timely, reliable data to understand where the issue occurred, what caused it, and how to fix it. For immediate remediation, that data also needs to be reliable and actionable. This can be done, for example, by setting up alerts for different scenarios, embracing a method of peer code review, and unit tests.

3. Eliminate manual work

As shown in Figure 1, one of the main focal points for both DevOps and SREs is automation. Both models encourage adding as much automation and tools as possible, as long as they provide value to developers and operations by removing manual tasks.



Toil defined

The term ‘operational work’ is easily misinterpreted, which is why Google uses an alternative phrase: toil. This is not administrative chores, basic tasks, or ‘work I don’t like to do’. After all, some people get satisfaction from manual, repetitive work. There are also administrative chores that have to get done, but should not be categorized as toil: this is overhead. Overhead is often work not directly tied to running a production service, for example team meetings or HR paperwork.

Toil is the kind of work tied to running a production service that tends to be manual, repetitive, automatable, tactical, devoid of enduring value, and that scales linearly as a service grows. Not every task deemed toil has all these attributes, but the more closely work matches one or more of the following descriptions, the more likely it is to be toil. The following defines toil in more detail:

Manual:

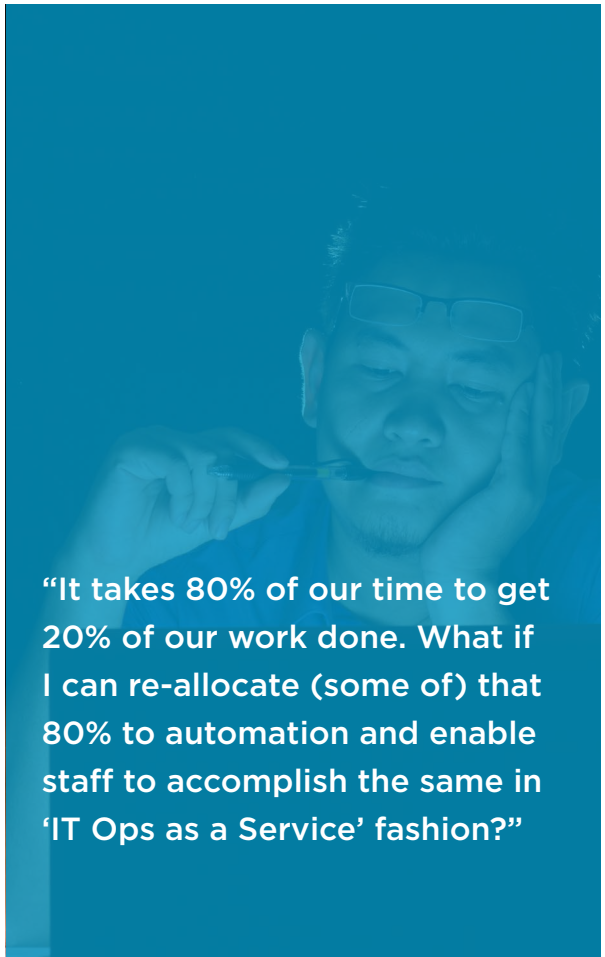
This includes work such as manually running a script that automates some task. Running a script may be quicker than manually executing each step in the script, but the time a human spends running that script is still toil time.

Repetitive:

If you’re performing a task for the first or second time, this work is not toil. Toil is work you do over and over.

Automatable:

If a machine could accomplish the task as well as a human, or the need for the task could be designed away, that task is toil.



“It takes 80% of our time to get 20% of our work done. What if I can re-allocate (some of) that 80% to automation and enable staff to accomplish the same in ‘IT Ops as a Service’ fashion?”

Tactical:

Toil is interrupt-driven and reactive, rather than strategy-driven and proactive. Handling pager alerts is toil.

No enduring value:

If your service remains in the same state after you have finished a task, the task was probably toil. If the task produced a permanent improvement in your service, it probably wasn’t toil, even if some amount of grunt work, like straightening configurations, was involved.

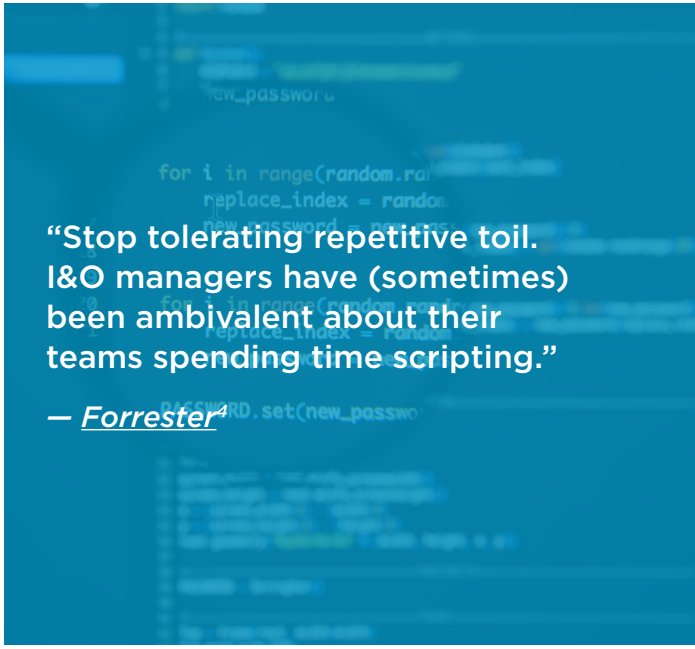
On with service growth:

If the work involved in a task scales up linearly with service size, traffic volume, or user count, that task is probably toil.

Major challenges of your SRE initiatives

Organizations looking to adopt SRE models are either employing in-house developed open source tools or loosely connected toolchains. This can quickly result in siloed tool sprawl and, owing to the ensuing heterogeneity, makes it harder for staff to find a single source of truth. Moreover, a multitude of tools, integrations, and administrative privileges can be difficult to manage – posing a significant security risk.

To add to this complexity, application releases have grown exponentially, impacting reliability and scalability. Some organizations release code daily or weekly, nearly 70% of DevOps users push new code to production monthly, while almost 80% of issues are the result of changes. This and the lack of tools introduce risk with the lack of visibility into release quality, limited to no visibility into the actual changes being delivered, and cumbersome manual processes that waste resources.



As a consequence, SRE adopters face two notable challenges:

- **Eliminating operations toil.**

  It is essential to reduce toil if you ever think about dealing with faster pace and a greater volume of changes.

- **Reducing MTTR.**

  The MTTR measures how long it takes operational teams to fix a problem, either through a workaround, a rollback, or another action. Reducing MTTR has a significant impact on the overall customer experience. It is also critical for SRE teams to minimize MTTR, because as they accept the risk of change and they need to recover fast to protect the service levels demanded by the business.

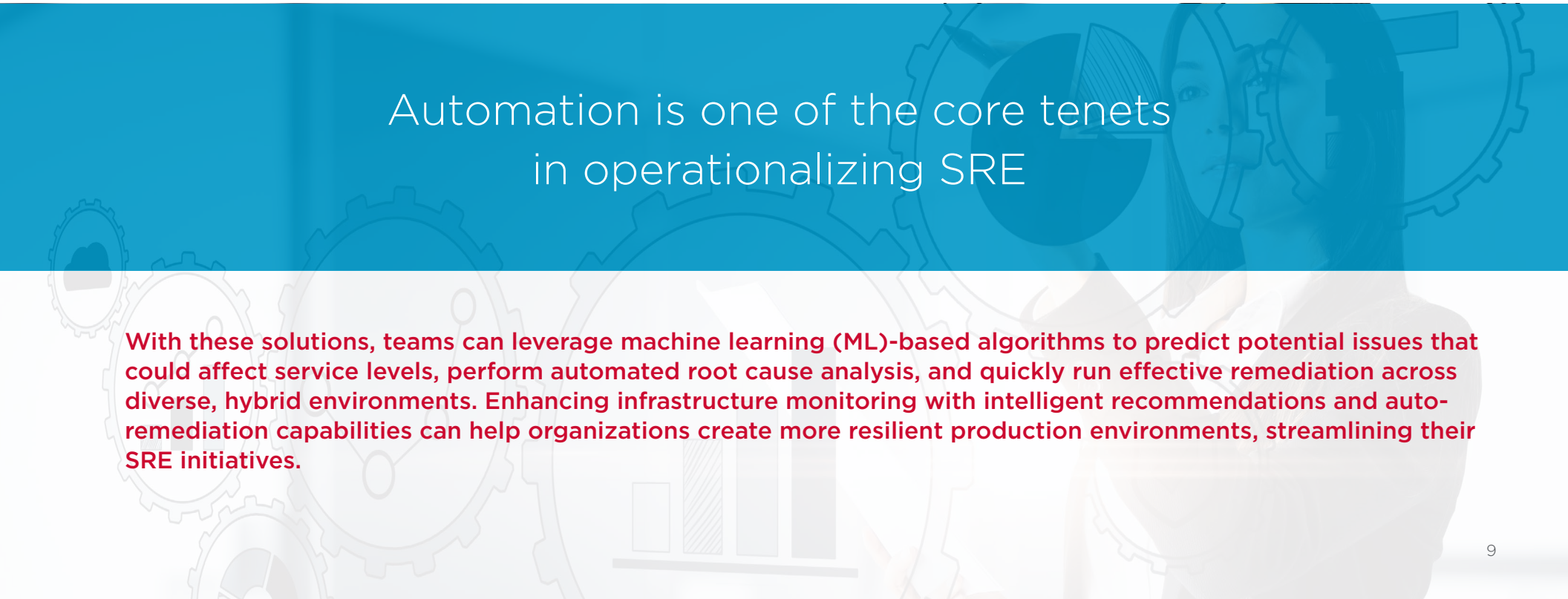
Owing to the business demand to manage more changes, more quickly, while protecting service levels, many organizations view SRE as a monumental undertaking. As a result, they either stall initiatives or implement too many changes too quickly, often failing to deliver enough value to justify sustaining the effort.

4 Forrester Research, Top 10 Trends That Will Shape Modern Infrastructure And Operations In 2020, Charles Betz, December 6, 2019

Site Reliability Automation

Being a site reliability engineer is not an easy job. You have to manage code deployment, configuration, monitoring, and more, so that everything works effortlessly in production. Triage, troubleshooting, remediation, and support are, for the most part, undertaken manually. No matter how good you are, these processes are error-prone and require significant effort.

To ensure their complex, hybrid, interrelated, and dynamic environments deliver the best possible user experience, SRE teams have to achieve fundamental breakthroughs in scale and efficiency. It's no longer sufficient simply to react a little faster when issues arise. Teams must gain the visibility needed to identify potential issues – and address them before they affect service levels. To contend with the explosive growth in data, complexity, and user demands, IT teams need to adopt an artificial intelligence for IT operations (AIOps) solution that provides service-driven, autonomous remediation.



Automation is one of the core tenets in operationalizing SRE

With these solutions, teams can leverage machine learning (ML)-based algorithms to predict potential issues that could affect service levels, perform automated root cause analysis, and quickly run effective remediation across diverse, hybrid environments. Enhancing infrastructure monitoring with intelligent recommendations and auto-remediation capabilities can help organizations create more resilient production environments, streamlining their SRE initiatives.

An integral part of an AIOps approach, SRA enables contextual automation that provides seamless integration of root cause alarms with remedial workflows. This contextual awareness lets SRE teams easily automate standard operating procedures that can be reused across environments. While contextual automation contributes to reducing operations toil, it also enables teams to deal with a bigger volume of events.

In parallel, the use of a recommendation engine leverages cross-domain insight to assist staff in choosing the most effective course of action for issue remediation. ML algorithms are used to rank the most successful remediation workflows in regard to the context. That continuous learning helps resolving more issues faster and reduces the MTTR.

SRA addresses two major challenges of SRE teams by efficiently reducing operations toil and improving MTTR. Ultimately, SRA empowers DevOps initiatives by aligning infrastructure management with the pace of modern continuous delivery. In the very near future, SRE will become mainstream; automation will make the decisions and ensure consistent operations.

KNOWING

...Knowing the root cause of a problem by pulling data together from disparate sources and applying AI and ML to find the source of the problem – fast.

DOING

...Doing an action that seeks to solve the problem, using out-of-the-box remediation and diagnostic workflows.

LEARNING

...Learning what works and what doesn't, so that the optimal response can be made when similar problems occur in the future.

Figure 2: Benefits of SRA

SRA in practice: real-world use cases

1. System / application restarts

Systems restart are frequently the fastest way to fix issues and restore customer experience – but it requires careful orchestration. Imagine, for example, a service hosted in a clustered environment. Once the culprit server is identified (root cause), we need to stop this server accepting new user sessions. This requires reconfiguration of the load balancing service, temporarily removing the affected server. Then we need to wait for sessions to end, or to end then gracefully. Log files then need to be collected for forensics and flush temporary files to ensure a clean system. Only then can the restart be initiated.

How can we be certain the system comes back to full operation? We need to perform tests to ensure the system is back in a working state, following which the system can be returned to the load balancing pool. Even minor tasks may require precise orchestration spanning infrastructure on the periphery of the affected service.

2. Disk space / storage remediation

This remains the most common infrastructure-related remediation use case for IT operations staff. Typically, a clean-up routine is a common redress for 'classic' Windows and *nix-based servers. These routines may already exist in the form of prepared scripts specific to an operating system or application. Cleanup frequently involves identifying the location taking up disk space, and then archiving/deleting it. This may demand a restart. The space should then be confirmed prior to commencing the cleanup routine.

Even in a cloud environment, running out of storage remains a problem. For example, with multiple storage strategies available for container-based deployments, applications may require persistent storage. A variation of clean-up routine would be to permanently or temporarily extend the available storage. Orchestration of storage would be needed here.

3. NetOps remediation

Application accessibility or experience can be impeded anywhere within the 'mainframe-to-mobile' stack, including network. An example of remediation is that a service is seeing peak load, but the VLAN(s) associated with accessing backend services might have a bandwidth policy which cannot cope with traffic volume – impacting user experience. Automation can orchestrate the physical virtual network layer by making a config change to increase the maximum throughput for the given interface/VLAN.

4. SecOps remediation

Most enterprises are overwhelmed both by the magnitude of security-related alerts and the manual, error-prone processes used to address the threats. Solutions are typically built from scripts or open source tools that require significant care and which lack enterprise-wide coverage. An example of automated remediation could be the detection of multiple root logins on single / similar system in a short time span. This could indicate an account was compromised. Further logins should be blocked (or password changed) and logs collected for triage.

5. Continuous delivery / deployment remediation

If a deployment fails, your ability to restore business services hinges on the speed and success of your recovery options. The simplest route to recovery is to revert to the previous version of the application/service. Sometimes however, it's just not as simple as reverting a change. A secondary environment enables you to redirect application traffic away from the problem. However, such a change may involve significant orchestration work in your application, such as promoting a new primary service and rerouting traffic at the load balancers.

Whether you're rolling your deployment back or routing operations to an alternate infrastructure, time matters. With automation, changes can be reverted quickly and safely, with little or no impact on service delivery. It also reduces the stress on personnel performing the service restoration.

6. Alarm / ticket enrichment

Automation provides rich orchestration capabilities across the DevOps tool chain, including common monitoring and ITSM systems. Bi-directional integration enables updating alarms with additional diagnostics evidence, as well as automatically creating and updating incident or change tickets without the need for manual hand-offs between different organizational silos. The rich orchestration provides a consistent way to exchange data between systems in order to correlate alarms to incidents, and change requests needed for restoring production outages.

Automated workflows are ideally suited to collecting the additional information, and enriching the incident data with the information IT operations staff need to take corrective action. This provides full transparency and compliance as to the actions taken and their results.

7. Security and compliance

All too often, enterprises rely on manual, disconnected, and undocumented processes to resolve issues and meet SLAs. This lack of visibility and consistency directly undermines compliance. Automation provides documented, automated IT operational processes, eliminating tribal knowledge. Workflows cannot be deviated from (consistency). Every step is captured in audit logs (who, what, when). Remediation workflows are exposed to SRE staff as, 'self-service standard operating procedures' – which means they don't have privileged access and can only execute selected workflows using predefined credentials.

Drawing it all together

The reality is that many organizations looking to adopt SRE models are employing loosely connected toolchains. After introducing a multitude of tools, staff typically find it harder to manage the infrastructure efficiently and to expedite problem solving. As a consequence, SRE adopters are facing two notable challenges:

- First, reducing operations 'toil'. In Google's definition, toil is not just 'work I don't like to do'. It is the kind of manual, repetitive, and mundane work that provides little value to operations. It is essential to reduce toil if you ever think about dealing with faster pace and greater volume of changes.
- Second, reducing MTTR – the measure of how long it takes operational teams to fix a problem, either through a workaround, a rollback, or another action. Moreover, MTTR has a significant impact on the customer experience. It is also critical for SRE teams to minimize MTTR, because as they accept the risk of change, they need to recover fast to protect the service levels the business expects.

Because of the need to handle more changes, faster, while protecting service levels, many organizations view SRE as a monumental undertaking. As a result, they either stall initiatives or try to implement too many changes too quickly, often failing to deliver enough value to justify sustaining the effort.

This is where Site Reliability Automation (SRA) comes in.

Enhancing infrastructure monitoring with intelligent recommendations and automated remediation capabilities can help organizations create more resilient production environments, streamlining their SRE initiatives. By leveraging site reliability automation capabilities, teams can seamlessly integrate root cause alarms with remedial workflows. This contextual awareness enables SRE teams to easily automate standard operating procedures that can be reused across environments. While contextual automation contributes to reducing operations toil, it also enables teams to deal with a bigger volume of events.

In parallel, advanced SRA solutions – such as from Broadcom – can provide a recommendation engine that leverages cross-domain insight to assist staff in choosing the most effective course of action for issue remediation. ML algorithms can be used to rank the most successful remediation workflows in regard to the context. That continuous learning helps teams resolve more issues faster and reduces MTTR.

The bottom line? SRA enhances infrastructure monitoring with intelligent recommendations and auto-remediation capabilities to help SRE engineers create more resilient production environments.

Broadcom Inc. is a global infrastructure technology leader built on 50 years of innovation, collaboration and engineering excellence.

Broadcom Inc. (NASDAQ: AVGO) is a global technology leader that designs, develops and supplies a broad range of semiconductor and infrastructure software solutions.

Broadcom's category-leading product portfolio serves critical markets including data center, networking, enterprise software, broadband, wireless, storage and industrial. Our solutions include data center networking and storage, enterprise and mainframe software focused on automation, monitoring and security, smartphone components, telecoms and factory automation. For more information, go to www.broadcom.com.

Learn more at:

broadcom.com/enterprise-automation

A Broadcom Company

technologies

For product information please visit our website at: broadcom.com

August 25, 2020