Red Hat
Ansible Automation
Platform

Observability and Event-Driven Ansible

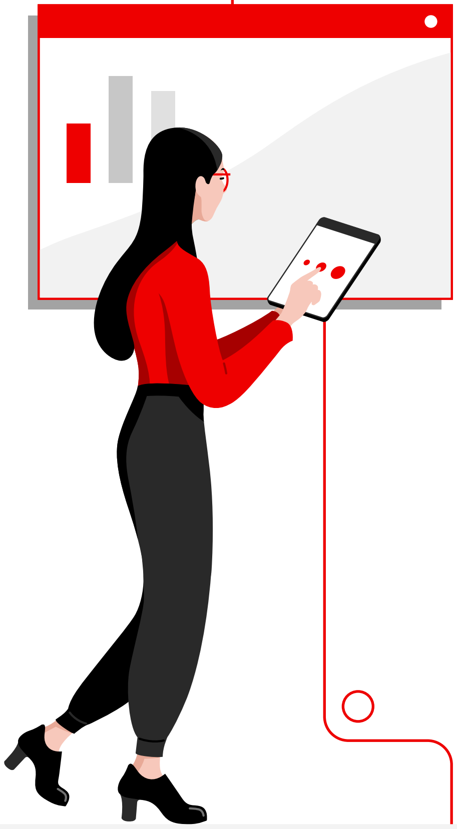
Observability meets automated action for efficient, resilient SRE practices

Contents

Introduction

While organizations have unique system, network, and application requirements, they share a common goal: ensuring that key applications and IT services are running smoothly by reducing downtime and proactively responding to changing IT conditions.

In a digital world, applications must run reliably to serve the needs of customers, employees, and business partners. When these applications are not running in top operational form, the implications can be massive.



Ineffective applications can cause outages, security holes, and downtime.
The results are lost productivity and revenue.

Site reliability engineering (SRE) and platform engineering have become essential approaches for enterprise organizations to ensure the reliability, scalability, and efficiency of IT services and innovations, which are the foundation for delivering a consistent customer experience and maintaining competitive advantage in the digital economy.

This e-book takes a closer look at both disciplines, and how enterprise organizations can use automation to accelerate reliable software delivery and robust operational practices to keep their organizations' key IT solutions running smoothly in today's competitive market.

[1] "Average cost of downtime per industry." Pingdom, accessed 7 Feb. 2023.

Chapter 1

Why site reliability engineering and platform engineering are essential

SRE fills the gap between software engineering and operations to deliver and operate scalable and highly-reliable applications and IT operational services.

The SRE concept has been adopted by organizations around the world, often in the form of customized approaches designed to meet an organization's unique needs. SRE emphasizes the use of DevOps approaches to tackle operational issues, such as system scalability, reliability, and efficiency.

What does a site reliability engineer do?

They create a bridge between development and operations by applying a DevOps software engineering mindset to applications and underlying infrastructure administration. This involves setting clear reliability goals and measuring performance against these goals, factoring in scalability and reliability from the start.

While SRE goals vary depending on the organization and its specific needs, there are some common reliability goals that many organizations share.

Typical SRE objectives

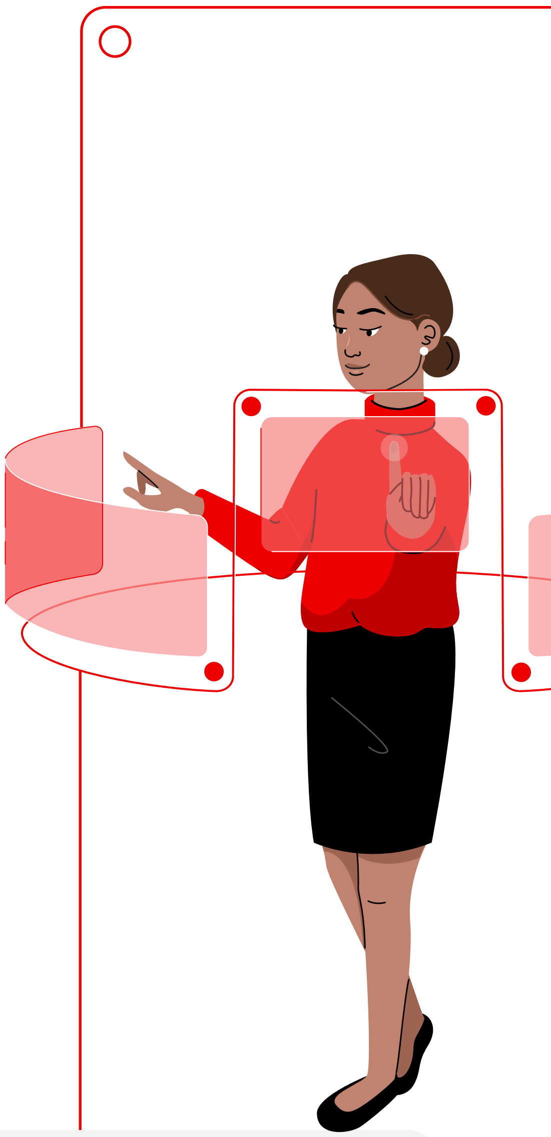
1. **Service availability.** Setting a target for uptime, aiming for high availability percentages, and ensuring services are accessible.
2. **Mean time to resolution (MTTR).** Reducing the time it takes to recover from incidents or outages. This metric measures the average time it takes to restore service functionality after an incident occurs.
3. **Monitoring and alerting.** Establishing robust monitoring and alerting systems to detect and respond to issues promptly. This includes setting up alerts for key performance indicators (KPIs) and implementing proactive monitoring practices.
4. **Backup and recovery.** Developing and implementing plans to recover from failures or disruptions. This includes backups, procedures to restore service and more.
5. **Post-incident analysis.** Establishing processes for responding to incidents promptly and conducting thorough post-incident analyses to identify a root cause and prevent similar issues from occurring in the future.

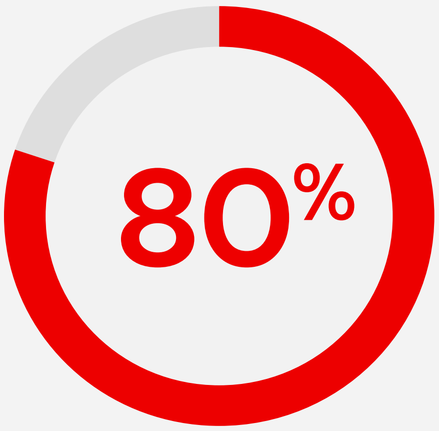
SRE teams codify operational work into software, for example, automating routine tasks, and building resilience into systems by anticipating and planning for failures. Through these practices, an SRE professional aims to improve the reliability, availability, and performance of systems, while also promoting efficient development and release of software.

What does a platform engineer do?

Platform engineering focuses on building and maintaining the development environment and processes that support application development and deployment. This can include the unique needs of their technology stack, application requirements and processes, such as specific cloud environments, development and deployment pipeline processes, specific databases and data sources, and networking systems. A key goal is to abstract complexity for development teams and help them to deploy software innovations more efficiently while taking DevOps principles into consideration.

Platform engineers also aim to reduce friction and toil for the developer by providing a self-service platform with the capabilities they need to produce valuable software with as little overhead as possible. The platform is usually designed to include everything a development team needs, presented in a manner that reflects the team's preferred workflow, in order to increase developer productivity and reduce the cognitive load. For example, an organization deploying to the cloud can abstract this complexity for a software developer who may be new to cloud deployment, but has specific expertise such as coding a financial services application.





By 2026, 80% of large software engineering organizations ***will establish platform engineering teams*** *as internal providers of reusable services, components, and tools for application delivery.*[1]

[1] "Gartner. What is platform engineering?" 26 Oct. 2023.

By building and managing these platforms, organizations can:

- **Streamline their development processes.**
- **Reduce time to market.**
- **Enhance overall operational efficiency.**
- **Provide a more sustainable and scalable approach to software delivery.**

While they focus on different aspects of IT operations, both SRE and platform engineering teams work toward the common goal of making IT systems more resilient, efficient, and supportive of business objectives.

Why SRE and platform engineering roles are so popular today

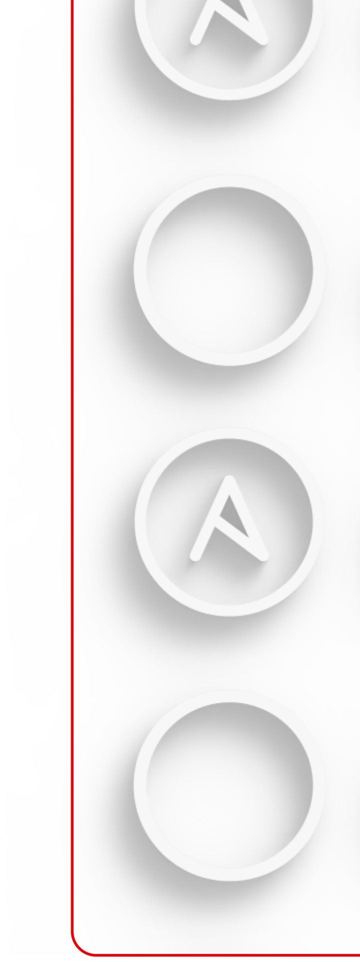
According to Gartner research, organizations are under pressure to accelerate innovation and are relying heavily on digital channels to reach their customers.[2]

This means that organizations are dependent on digital products and services now more than ever, which is pushing them to explore modern approaches such as SRE and platform engineering to balance reliability and the rapid pace of change.

SREs keep mission-critical applications scalable and resilient while platform engineers streamline the development platform so new applications can be created more quickly.

*By 2027, **75% of enterprises will use site reliability engineering practices across their organizations** to optimize product design, cost, and operations to meet customer expectations, up from 10% in 2022.*[2]

[2] Gartner. "2023 Hype cycles: Deglobalization, AI at the cusp and operational sustainability." 17 July 2023.

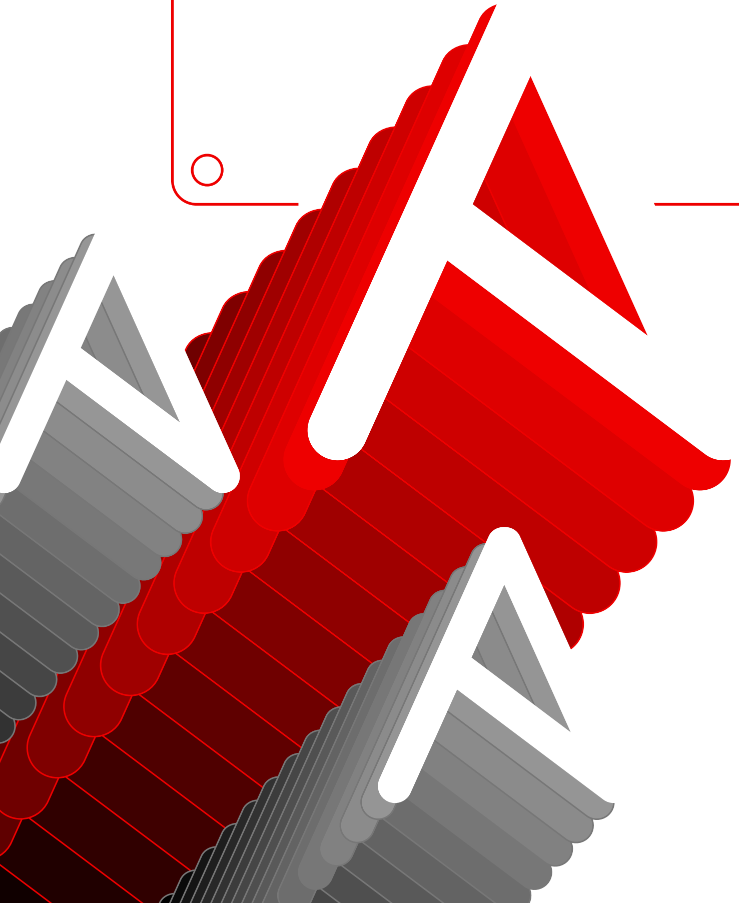


Top challenges make event-driven automation mission critical

As systems and processes evolve, teams face the challenge of balancing the need for innovation with the imperative of system reliability.

In a digital world, applications must stay up and running because they drive revenue, productivity and much more. They must remain resilient and continually manage risk. New applications and innovations, such as edge and generative AI applications, add to existing computing demands yet are often strategic because they are delivering the next level of digital capabilities. It all becomes too much to manage manually, through singular task-based automation or via scripts.

This is where event-driven automation comes in to keep all of these key systems in top operational form. What's more, it works with your observability tools to go from knowledge of a problem to actions you design in an automated way.



What is Event-Driven Ansible?

As a part of Red Hat® Ansible® Automation Platform, Event-Driven Ansible provides the event-handling capability needed to automate critical tasks across any IT domain. This helps teams determine the appropriate response to a particular event, then executes automated actions to address or remediate that event. It connects operational intelligence such as observability data to automated actions, improving the resilience and responsiveness of IT while freeing teams to focus on innovation and key priorities.

How Event-Driven Ansible works

Event-Driven Ansible processes events containing discrete intelligence about conditions in your IT environment, determines the appropriate response to the event, then executes automated actions to address or remediate the event. These 3 components are the building blocks of Event-Driven Ansible: event sources, rules, and actions.

Event sources are sent to Event-Driven Ansible via an event source plug-in or webhook. Event-Driven Ansible receives the event source and processes each event against the ruleset in the Ansible Rulebook.

Rulebooks contain rulesets and conditions that need to be met in order to trigger an action. Conditional statements are used to filter events and determine the desired action. These actions could include responding to an event with an Ansible Playbook, module, workflow, or job templates from Event-Driven Ansible controller.

Actions are the result of the automation. Once a rule has been matched, the action associated with that rule is triggered.

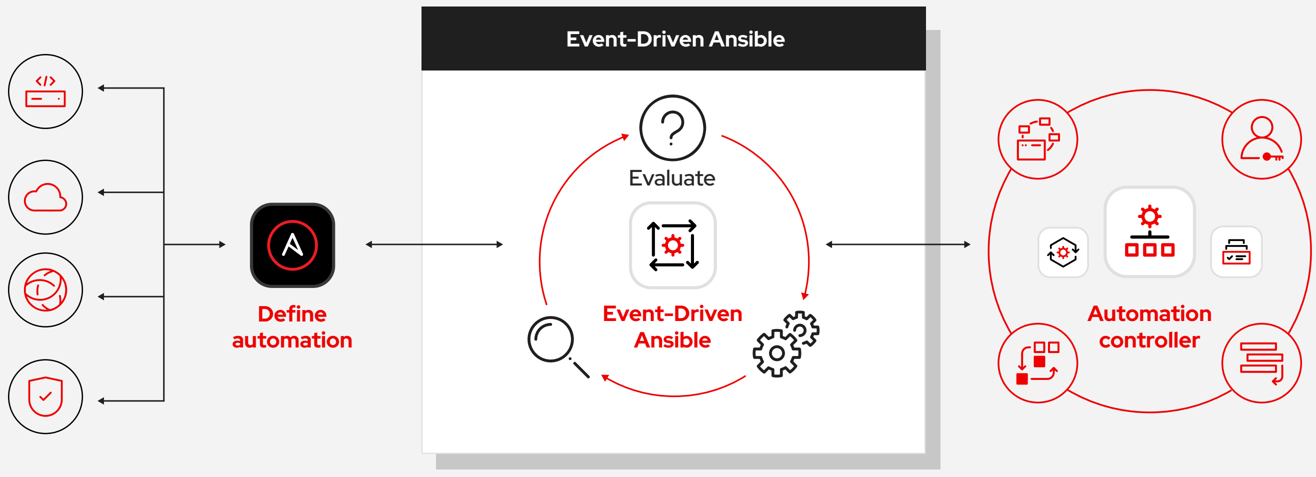


Figure 1. Event source, rules and decisioning, and the automated action.

Event-driven automation can help teams address common organizational challenges including:

Complexity

As organizations expand their IT environments, the number of components and interactions within their systems increases exponentially. This makes it difficult to predict how changes in 1 part of the system will affect others, complicating the task of maintaining reliability.

Disconnected teams

Another significant challenge is the absence of direct ownership of systems, making remediation highly dependent on collaboration. Inaccessible and/or disparate data, as well as the need for data to flow through many systems to be analyzed are among the top challenges facing organizations today.

Pace of change

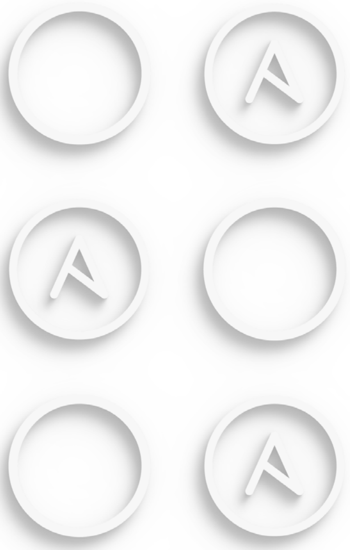
SRE teams must stay abreast of the latest practices, tools, and technologies to effectively manage system reliability. This requires ongoing education and sometimes significant shifts in tools or practices, which can be resource-intensive to implement.

Legacy technology and tool sprawl

As organizations grow over time, legacy applications and infrastructures are inevitable, which can create gaps in efficiency, skills, and security. Without standardization of tools and their usage across the organization, effective collaboration becomes difficult. Significant skills gaps may also be present for legacy technologies.

Chapter 2

Observability paired with event-driven automation can help manage complexity



As systems architectures increase in complexity and scale, IT teams face growing pressure to track and respond to conditions and issues across their multicloud environments. As a result, IT operations and SRE teams are looking for greater observability of these increasingly diverse and complex computing environments.

What is observability and how is it different from monitoring?

Observability is traditionally defined as using logs, metrics, and traces to understand the state of a system or application in order to quickly determine the root cause of unanticipated issues.

In more complex cloud environments, observability encompasses additional sources of information, including metadata, user behavior, topology and network mapping, and access to code-level details. The importance of observability has grown due to the increasing complexity of software systems, the widespread adoption of microservices, and the growing reliance on distributed architectures.

Today, observability tools provide more of a unified interface designed to help you manage across your estate, and these solutions are adding additional capabilities to help teams identify key issues, for example by applying AI to distill a high volume of alerts, or integrating with related capabilities such as Event-Driven Ansible, ticketing and notification systems and more.

In a monitoring scenario, teams typically preconfigure dashboards to send alerts about anticipated future performance issues. These dashboards rely on the assumption that teams will be able to either predict problems or identify and resolve them early to minimize their impacts.

Observability to action: adding event-driven automation

Event-driven automation is the process of responding automatically to changing conditions in an IT environment to help resolve issues more quickly, minimize service interruptions and latency, and reduce the prevalence of routine, repetitive tasks. It connects data, analytics, and service requests to automated actions so that activities—such as responding to an outage or adjusting some aspect of an IT system—can take place in a single, agile motion.

Using event-driven automation helps IT teams address how and when to target specific actions. It also helps manage the complexity of hybrid cloud, edge and complex AI environments, while freeing teams to focus on other priorities.

Combining observability with self-healing has improved resolution times and reduced services downtime. We saw a 50% reduction in service tickets.[3]

[3] Red Hat case study. "Mutua Madrileña adopts automation as standard with Red Hat," 15 Nov. 2023.

What does it mean to be event-driven?

In an IT environment, being event-driven means connecting intelligence about what is happening in the IT environment and/or service requests to automated actions so that manual steps typically taken by IT teams can happen in a single automated workflow. Event-Driven Ansible, part of Ansible Automation Platform, works with a range of observability and other management tools. It minimizes the time required for communication, coordination and collaboration across a team and as a result can reduce MTTR, increase efficiency, and ensure consistent and accurate responses to common IT responses.

Event-driven automation receives an alert and can employ a predefined automated response when an event occurs. For example, a system outage can trigger an event that automatically executes a specific action—such as logging a trouble ticket, gathering facts needed for troubleshooting, or performing a reboot. Since these actions are predefined and automated, they can be performed more quickly than if the required steps were done manually. Event-Driven Ansible is flexible from event source, to rule, to action, meaning you define the exact response you desire when a specific alert or condition is detected.

Event-driven automation helps teams perform a variety of additional Day 2 operations, such as:

- Configuration management and resolution of drift issues.
- Provisioning according to defined criteria.
- Edge device management.
- Tuning and scalability across storage, databases, and applications.
- User management.
- Adhering to compliance standards.

Challenges of using observability tools

While observability tools and logs will record events and other issues, sorting through the high volume of alerts to respond to the right need is a challenge. Manual coding, scripting, and application programming interfaces (APIs) can be used, but these processes are time consuming and can contribute to technical debt. As a result, they are usually reserved for 1 or 2 key use cases.

By using a solution like Event-Driven Ansible, which is flexible from source to rule to action, teams can expand the use of event-driven automation, accelerating their MTTR when issues arise, automating management actions, and making the most of their observability intelligence tooling.

Event-Driven Ansible can also help platform engineers get more out of their observability tools and support teams as they:

- Automate response to key alerts so they do not get missed.
- Provision resources for development and ensure adherence to compliance standards.
- Access self-service solutions for efficiency across their extended team.
- Streamline application testing and deployment.
- Ensure appropriate application security checks are carried out.

Chapter 3

Go from observability to action for smoother business-critical operations

Adding Event-Driven Ansible to observability practices can help advance an organization's event-handling capabilities by automating time-consuming tasks, responding to changing conditions in any IT domain, and reducing MTTR.

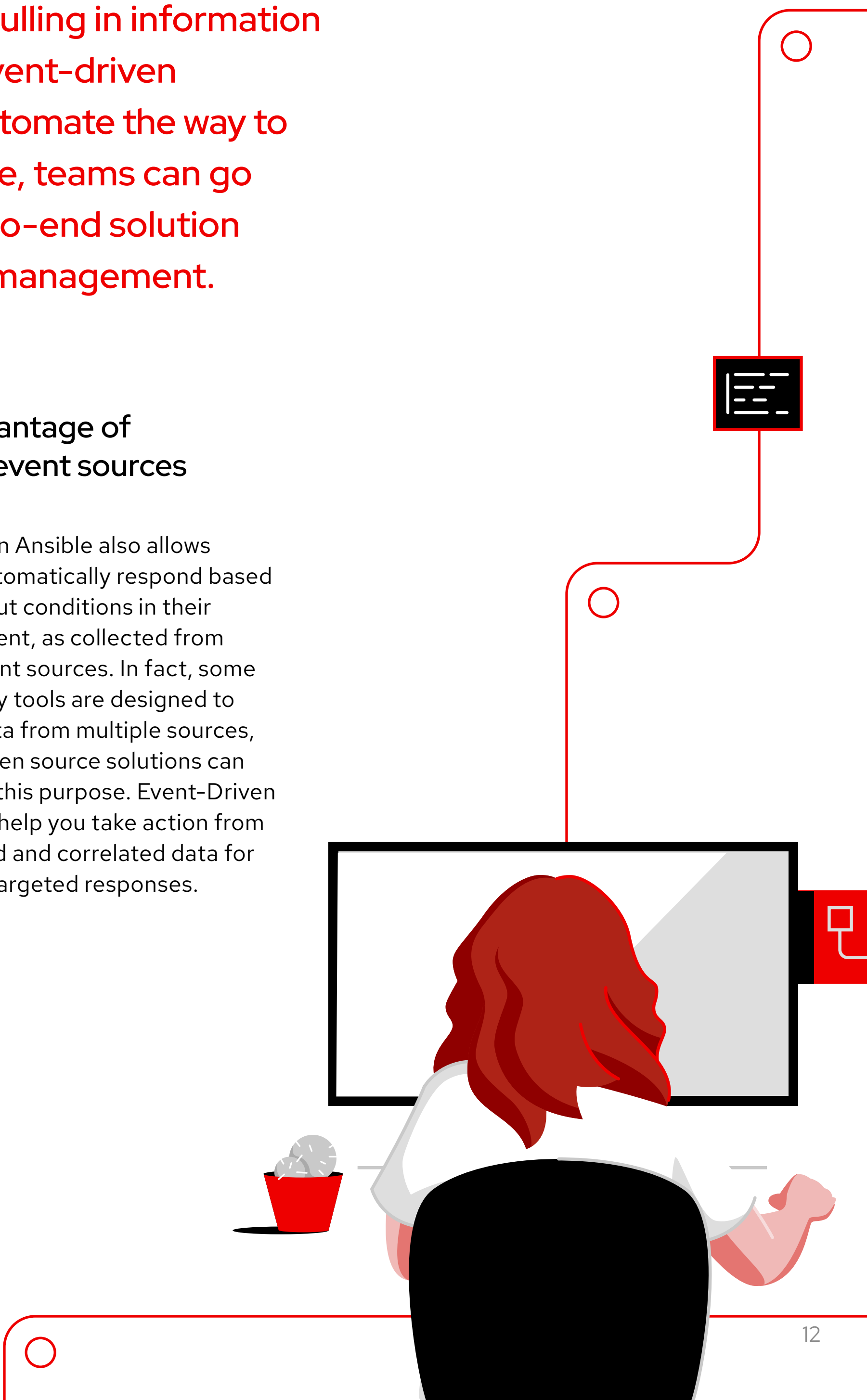
Observability tools are excellent at pulling in information about what needs to be done, and event-driven automation provides the means to automate the way to act on that information. So in essence, teams can go from insights to action as a full end-to-end solution as part of modern Day 2 operations management.

This automated approach can help organizations:

- Get more value from observability data.
- Improve response times.
- Reduce the number of service tickets received.
- Apply comprehensive automation, which in an end-to-end scenario, allows for a more advanced and modern approach to Day 2 operations.
- Create self-service scenarios for any task.
- Automate responses to security alerts.
- Dramatically increase agility and efficiency.

Take advantage of multiple event sources

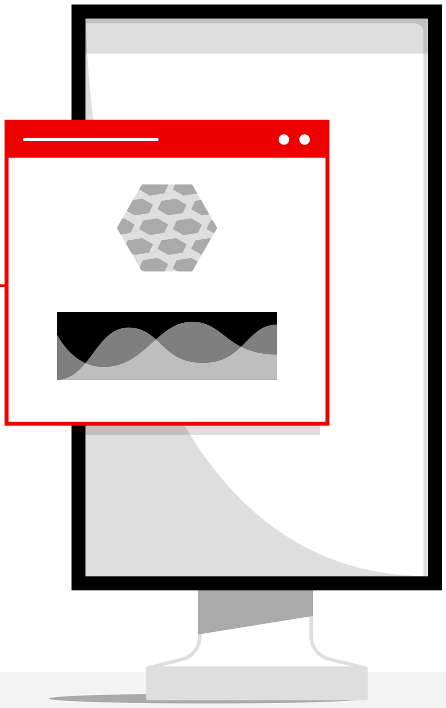
Event-Driven Ansible also allows teams to automatically respond based on data about conditions in their IT environment, as collected from multiple event sources. In fact, some observability tools are designed to combine data from multiple sources, and even open source solutions can be used for this purpose. Event-Driven Ansible can help you take action from consolidated and correlated data for even more targeted responses.



The advantage of event-driven automation for SRE and platform engineers

Both SRE and platform engineers build and deploy automation for different use cases and needs. Event-driven automation helps both roles respond to changing IT conditions.

For example, an SRE might use event-driven automation to automatically respond to a spike in application traffic by scaling up storage or memory resources to meet the demand. The platform engineering role may use event-driven automation to automatically reapply a standard configuration when drift is detected in the development or test environments.



Did you know?

Event-Driven Ansible works with a wide variety of event sources. This means you can work with open source aggregation tools such as Prometheus Alertmanager or Kafka topics.

There are also a series of observability partners who have created certified or validated Event-Driven Ansible Content Collections to make it faster and easier to stand up these end-to-end automation scenarios.

Observability, monitoring, and platform engineering tools

There are different types of observability, monitoring, and platform engineering tools. These tools have integrations and other content that can help you jumpstart your event-driven automation scenarios.

Red Hat and our ecosystem of expert partners create and support the collections that are validated or certified by Red Hat. As a general rule, certified content includes the “how to” for specific platforms, while validated collections are best practices of “what to do” to manage operations.

Chapter 4

Business value of event-driven automation with observability tools

Both the SRE and platform engineering roles focus on reducing downtime and disruption, while improving operational efficiency, scalability, and responsiveness.

By automating reactions to real-time events, organizations can streamline workflows, reduce manual intervention, and minimize the latency in their decision-making processes. This automation not only accelerates the pace at which businesses can respond to changes and opportunities, but also enhances reliability and consistency in IT operations. By decoupling services and employing an event-driven model, Event-Driven Ansible helps organizations to better manage complex environments, leading to improved service quality and customer satisfaction.

Especially when combined with observability tools, event-driven automation is a foundation for operating agile, scalable, and resilient IT environments that adapt to evolving business needs.

To understand the potential of event-driven automation in your organization, consider the following observability to action scenarios:

Deliver business value from IT for a competitive advantage.

What if you could perform key functions more efficiently and more cost-effectively than the competition? For example, an e-commerce retailer could scale up operations automatically to meet a sudden surge in demand during sales promotions such as Black Friday.

Respond to observability data automatically.

Automating a response to drift or a security issue can allow you to limit the impact of issues and avoid full-blown outages. Additionally, you can streamline certificate management by allowing for an automated response to certificate renewal and rotation needs. Automatically create a ticket and update the configuration management database.

Reduce downtime and keep business solutions running optimally.

Consider the business benefits of being able to limit the scope of an outage, where downtime often translates to productivity and revenue loss. This could include being able to respond automatically to service requests, resulting in better overall efficiency, or being able to shut down or reroute network traffic when there is an alert, giving you time to investigate and improving security and compliance.

Automate a software release for consistency.

This capability applies to virtually any industry, where new applications or enhancements to existing applications are needed to provide business value.

Work from a "single source of truth" for operational consistency.

Manage from a single source of truth to enforce configurations, policies and other criteria as soon as a solution goes out of compliance, immediately close a security risk or stay aligned to your operational requirements.

How to get started with Event-Driven Ansible

With an understanding of how event-driven automation can work together with your observability tools, you may be wondering how to get started. Red Hat recommends a “start small, think big” approach to adopting any type of automation, including Event-Driven Ansible. First master the simpler use cases that don’t directly change your solutions and then gradually grow scope and sophistication of the event-driven automation you are using–in line with your expanding first-hand knowledge about how the solution works.

A great place to begin is by exploring a few simple use cases to gradually implement Event-Driven Ansible and grow your automation approach from there.

1 Gather facts

When a ticket comes in, the Event-Driven Ansible rulebook may define an action to reach out to the affected device and gather and add the configuration information to a trouble ticket. So, when the support team responds to the ticket, this information they require is already present and they are able to address the issue faster. This simple step saves time, reduces toil and churn, and is a great low-impact use case.

2 Generate a service ticket

When a condition is identified by your observability or monitoring tool, Event-Driven Ansible can automatically generate a ticket in an IT service management (ITSM) solution, or post a notification to an internal messaging system, such as Slack or a packaged application. For example, a security certificate is about to expire, your rulebook can create an alert and automatically generate a service ticket.

3 Send a notification

Take the automatic ticket generation a step further so that you can also send a notification to the right person on your team to address the event. For example, if a network or edge device is unresponsive, Event-Driven Ansible can automatically create a ticket and notify the designated team member, which can accelerate response time.

4 Do basic remediation

The next step is basic remediation, which might include resetting or rebooting a system and/or sending a notification if necessary. Using the example of an unresponsive network or edge device, Event-Driven Ansible can automatically create a ticket along with performing a basic reboot. If the basic reboot does not work, a team member can be notified as part of the automation sequence.

5 Perform advanced remediation

With the previous steps mastered, you are ready to bring in multiple event sources and correlate them to orchestrate the response that best suits your needs. For example, in the event that a basic reboot does not work, Event-Driven Ansible can, based on your prewritten Ansible rulebook, read a second event, find out which neighboring device is available, and reroute the network traffic. An important thing to remember is that Event-Drive Ansible can execute predetermined automation as an action according to rulebook conditions. This allows you to supplement your automation by integrating Event-Driven Ansible with existing playbooks.

Learn more

Ready to take the next step in understanding how Event-Driven Ansible works?

Read how to use event-driven automation with ITOps. Download our in-depth e-book.

Learn how Spanish insurance company Mutua Madrileña adopted automation as standard. Download the case study or watch the video.

Discover the power of Event-Driven Ansible with Red Hat's expert partners in this webinar series.

Try event-driven automation for yourself. Join a self-paced lab.

North America	Europe, Middle East, and Africa	Asia Pacific	Latin America
1 888 REDHAT1	00800 7334 2835	+65 6490 4200	+54 11 4329 7300
www.redhat.com	europe@redhat.com	apac@redhat.com	info-latam@redhat.com

facebook.com/redhatinc
@RedHat
linkedin.com/company/red-hat

redhat.com