gathr

DORA Metrics and Beyond

Continuously Monitor and Optimize DevOps Performance

Table of Contents

Introduction

Today, any organization developing software is seeking higher agility, efficiency, stability, and reliability in its operations. Teams need continuous improvements in software development practices and require accurate data and insights for monitoring the health and performance of their software development practices.

However, for a long time, organizations lacked a clear direction for measuring software development. Initially, teams attempted to measure the lines of code to assess developer productivity. Metrics related to developer productivity, such as function points, lines of debugged code, number of command-line arguments, etc., propped up. However, people soon realized that such metrics offered little help. For instance, tracking the lines of code improved performance on the metric but also led to a dramatic fall in the quality of the code. Similarly, tracking the number of bug fixes forced the teams to log every minor and trivial bug creating a significant slowdown in the shipping of code. Often, older, trivial bugs with no major impact took precedence over urgent, hard-to-fix bugs, as teams wanted to score high on their bug fixes and reduce the average age of open issues.

This is when DORA (DevOps Research and Assessment) developed its research program. DORA has been publishing annual reports for the last eight years with inputs from software development professionals, making it the longest-running academic research of its kind. The research uses behavioral science to identify better ways to develop and deliver software. DORA's metrics are now seen as an industry standard for measuring DevOps success and also for benchmarking. In this guide, we will explore these metrics and understand how to use them to improve software development. We will also explore some additional metrics and evolving practices to improve DevOps performance.

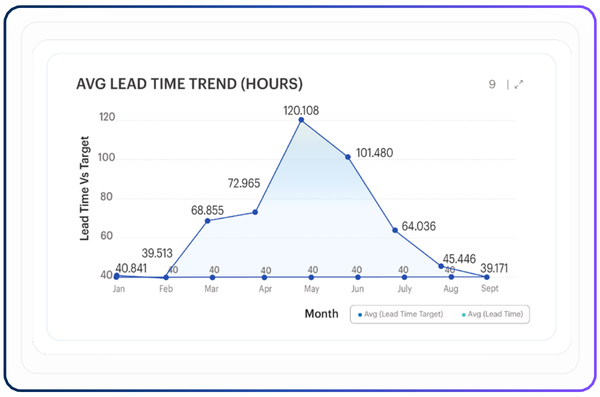
What are DORA Metrics?

DORA surveys thousands (to be precise, 33000 in its 2022 report) of DevOps engineers and leaders every year, evaluating their performance over the four key metrics that are now considered the holy grail for measuring the success of software development. DORA classifies the participating organizations into four distinct categories based on the survey responses as elite performers, high performers, medium performers, and low performers. Any organization tracking these metrics can compare its current state to its peers, identify areas for improvement, and take steps to become an elite performer. We have given a brief description of the metrics below:

Mean Lead Time for Changes

The mean lead time for changes is the average time code changes take to progress from the commit stage to production. While the definition is straightforward, organizations often need help to track this metric as they have to keep track of multiple tools. Any deviation from the normal or an increasing lead time can indicate potential issues with the pipeline and would require deeper root cause analysis. Teams should be able to identify which tasks (stories, bugs, sub-tasks, etc.) are taking a longer time and drill down to resolve bottlenecks.

Why is the mean lead time for changes metric important?



The lead time is a direct indicator of an organization's CI/CD efficiency. It indicates how quickly a team can deliver software or meet end users' requirements. Shorter lead times means teams are able to deploy faster, with faster feedback and quicker course correction. For reference, high-performing teams or elite performers in DORA's survey report the mean lead times between one day and a week and often measure it in hours. Longer lead times may indicate bottlenecks in the pipeline.

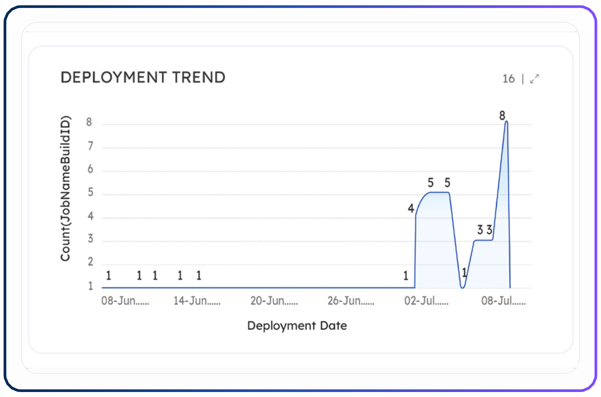
How to reduce the mean lead time for changes?

Teams can reduce their mean lead time for changes by following some of the common development best practices, such as breaking down a project into smaller tasks or batches and implementing trunk-based development. Manual handoffs and inefficiencies also lead to longer lead times. Therefore, automated testing and deployments should be implemented.

Deployment Frequency

Deployment frequency is the number of times a team deploys a code into the production environment over a period. Most high-performing teams can deploy multiple times in a day, on demand. One should note that deployment usually means releasing the changes into production, and it's different from a delivery, which is releasing the changes to a staging environment. The distinction is crucial as, many times, code changes remain stuck in the staging environment due to a lack of a release window or business go-ahead. In such cases, caution is required to view Deployment Frequency as a productivity indicator.

Why is the deployment frequency metric important?



A higher deploymentfrequency indicates higher process efficiency. It means that the work items are moving through the pipeline smoothly. By monitoring historical trends, teams can detect any potential issues whenever there is a sharp fall from the normal. Tracking the average build duration along with deployment frequency can provide a better picture and help in finding bottlenecks.

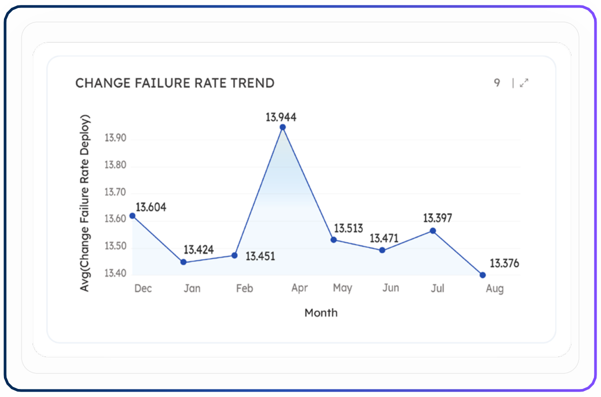
How to increase the deployment frequency??

Again, higher automation at various stages of CI/CD can help teams improve their deployment frequency. It requires a higher integration across CI/CD tools, automated testing, and continuous compliance and governance for quicker release go/no-go decisions.

Change Failure Rate

Change failure rate helps you track how many times code changes lead to a failure in the production environment. It is the percentage of code changes requiring remediation steps, such as rollbacks, patches, or hotfixes in production. Please, note that any issues detected and resolved before reaching production aren't included in measuring the change failure rate.

Why is the change failure rate metric important?



A higher change failure rate not only indicates that the team is spending more time resolving bugs instead of developing new features, but it also means that the application is failing more often and leading to a poor end-user experience. That's why teams need to keep track of this metric and keep it lower. For reference, high-performing teams have a change failure rate of below 15%.

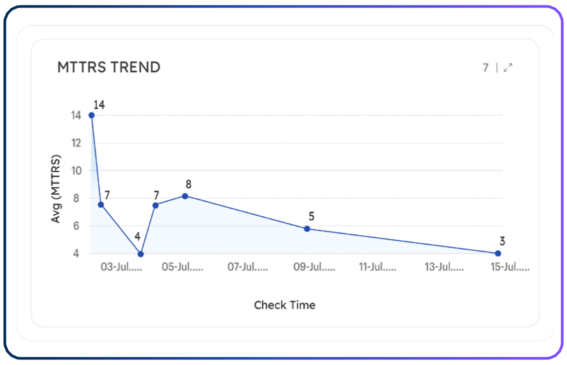
How to reduce the change failure rate?

Teams need to track their test automation using metrics such as test pass rate, test code coverage, and more. They might have to drill down to find potential issues with their testing and code review processes. Establishing quality gates can also help you ensure that only secure, compliant, and high-quality code reaches production.

Mean Time to Restore (MTTR)

The mean time to restore is the average time a team takes to restore service or recover from a system failure. While organizations can track this metric using their service desk or ticketing systems, they need integration across other tools for root cause analysis and troubleshooting. For instance, it can be helpful to track application resource consumption, crash trends, and hosts along with MTTR to draw its correlation with potential issues in RAM and CPU allocation.

Why is the MTTR metric important?

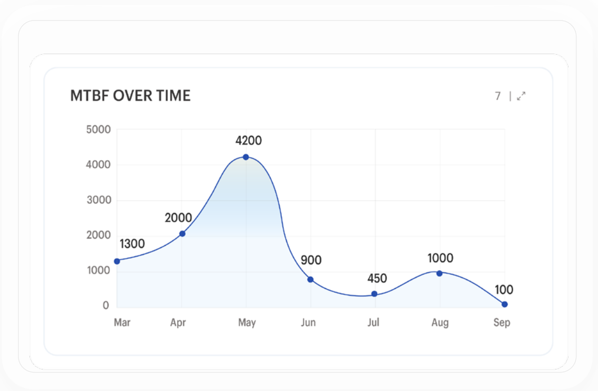


MTTR is a direct indicator of resiliency and tracking the MTTR metric can help teams improve their incident detection and response mechanisms. It can help teams reduce their downtimes and ensure a better experience for end-users. High-performing teams report an MTTR of less than a day and often resolve issues within a few hours.

How to reduce MTTR?

Teams must continuously monitor their system health and improve alert and incident response playbooks with increased automation and integration. In modern hybrid and multi-cloud environments, there are hundreds of alerts and issues to resolve, and teams can get overburdened. As a result, teams need better context, prioritization, and traceability to resolve issues quickly.

Operational Performance – Reliability



Recently, DORA added reliability as the fifth metric to its assessments representing operational performance. The metric helps you measure the success of your operational practices. It is a function of availability, latency, performance, and scalability and can help teams continuously improve their applications to meet end-user expectations. According, to DORA, a higher focus on operational performance can help teams reduce burnout and achieve better outcomes.

Why is the reliability metric important?

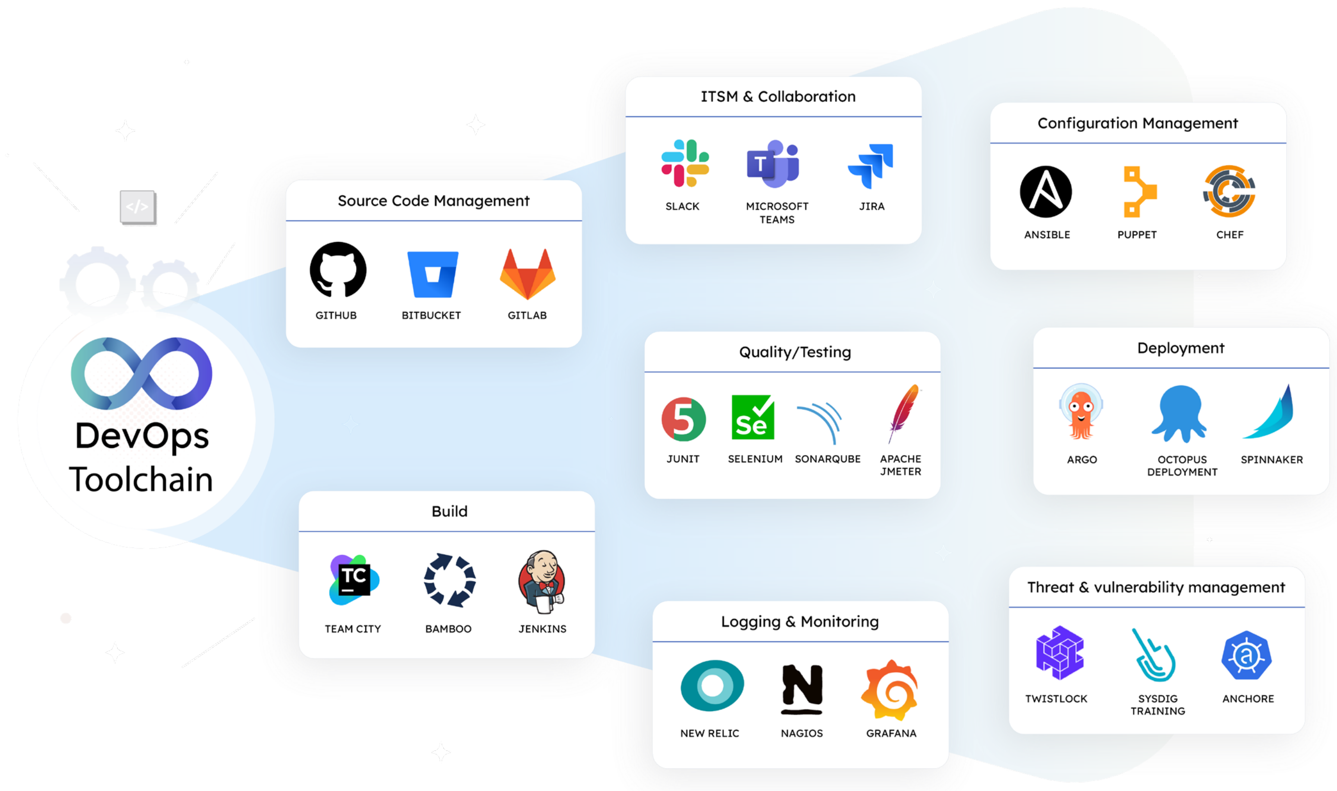
It is seen that even with higher delivery performance, organizations can face challenges in meeting end-users' reliability expectations due to gaps in their operations. While organizations with mature Site Reliability Engineering (SRE) practices are less likely to face such challenges, DORA suggests that it takes time for organizations to achieve their reliability targets. Organizations need to set up clear reliability goals and metrics to make continuous improvements in operational performance.

How to improve reliability?

Implementing SRE best practices can help organizations gradually improve their reliability. It involves end-to-end tracking of error budgets, service level objectives (SLOs), errors, availability, latency, and more. A major goal of SRE practices is to reduce toil, which can be achieved with increased automation across toolchains. Moreover, teams need to accept failures and conduct routine incident postmortems to build more reliable systems.

Challenges with Monitoring DevOps DORA Metrics

While DORA Metrics sound simple in theory, organizations often face significant roadblocks in the initial stages of implementation. Even for those who have implemented the metrics, it is not easy to drill down and across to identify the root cause of issues or find actionable intelligence for optimization and improvements. Organizations often fail to act on these metrics or are uncertain about the next steps to improve performance.



Disparate Toolchain

Organizations need to collect data from multiple tools, such as those for project management, source code management, CI/CD, security scanning, issue tracking, ticketing, and more, to monitor DORA metrics. Teams not only need to centralize this data but also need to transform it into consistent, calculable units, which is a complex task.

Data Analysis

While some open-source project can help teams implement the metrics and dashboards, their monitoring scope is restricted. Teams need to spend significant time and effort in developing solutions that can help them act on these metrics, correlate data across tools, and gather quick insights for troubleshooting.

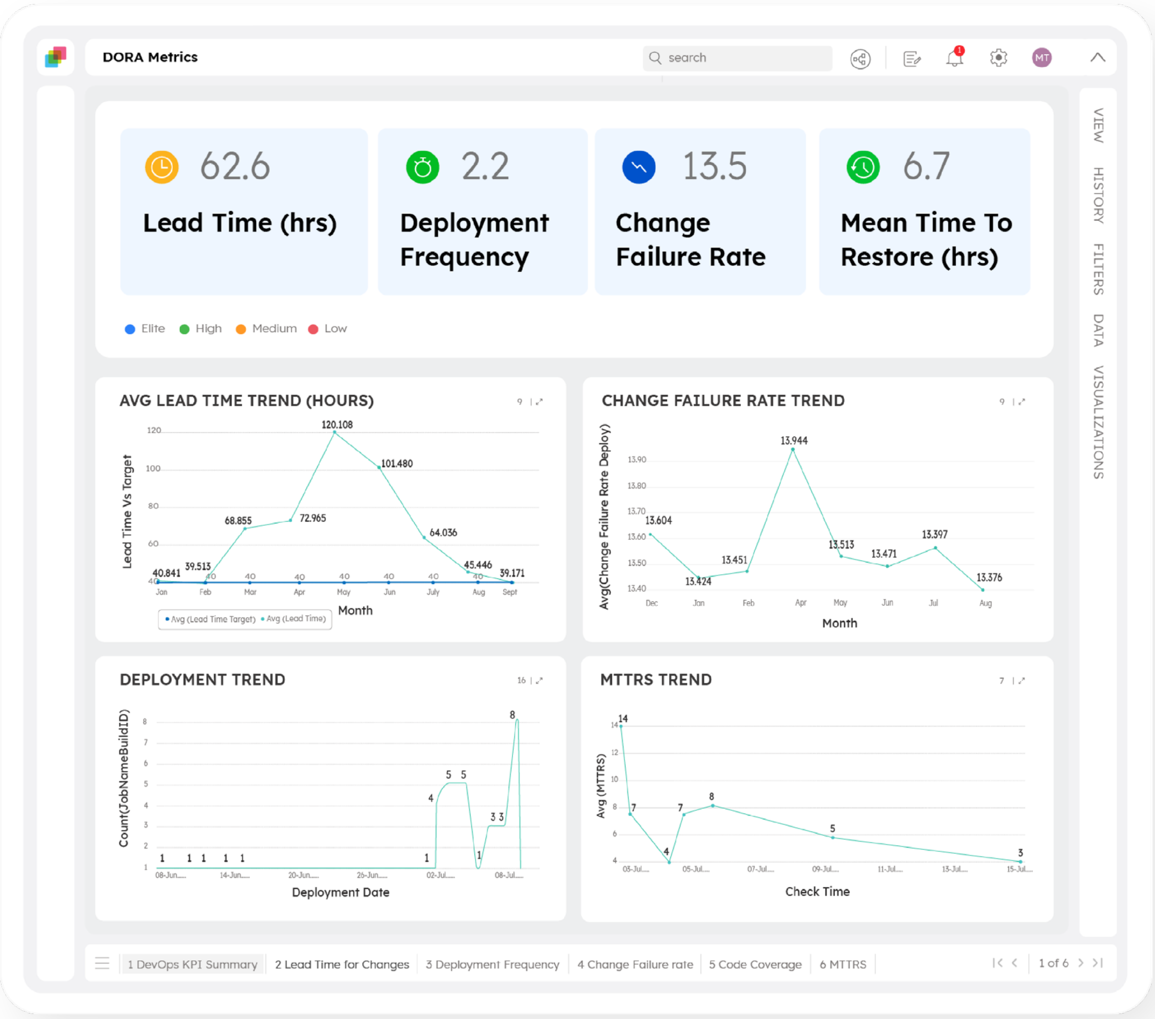
Automation

Lack of automation is another significant challenge that restricts an organization's ability to monitor these metrics accurately. It is possible to manually collect data for certain metrics like MTTR or Deployment Frequency, but it becomes difficult with metrics like Lead Time, which require data analysis from multiple tools.

It is also important to note here that teams shouldn't get blinded by these metrics. As Goodhart's Law states, "When a measure becomes a target, it ceases to be a good measure." Teams should understand that though the DORA metrics are crucial for benchmarking, they shouldn't become the goal. Instead of focusing on individual metrics, teams should prioritize adherence to best practices, increased automation, and observability to achieve better outcomes.

How to Monitor **DORA Metrics?**

As discussed above, organizations can choose open-source or commercial tools to track DORA metrics. Google's Four Keys open-source project can offer a good start to organizations if they have projects in GitHub or GitLab. It automatically sets up a data ingestion pipeline, collecting data from GitHub or Gitlab repos through Google Cloud and Google Data Studio. While it is useful for tracking the DORA metrics, there are obvious restrictions to the type of dashboards and metrics available in the project. It might require significant configuration to adapt the dashboard for your organization's toolchain or get expand monitoring to optimization and troubleshooting.

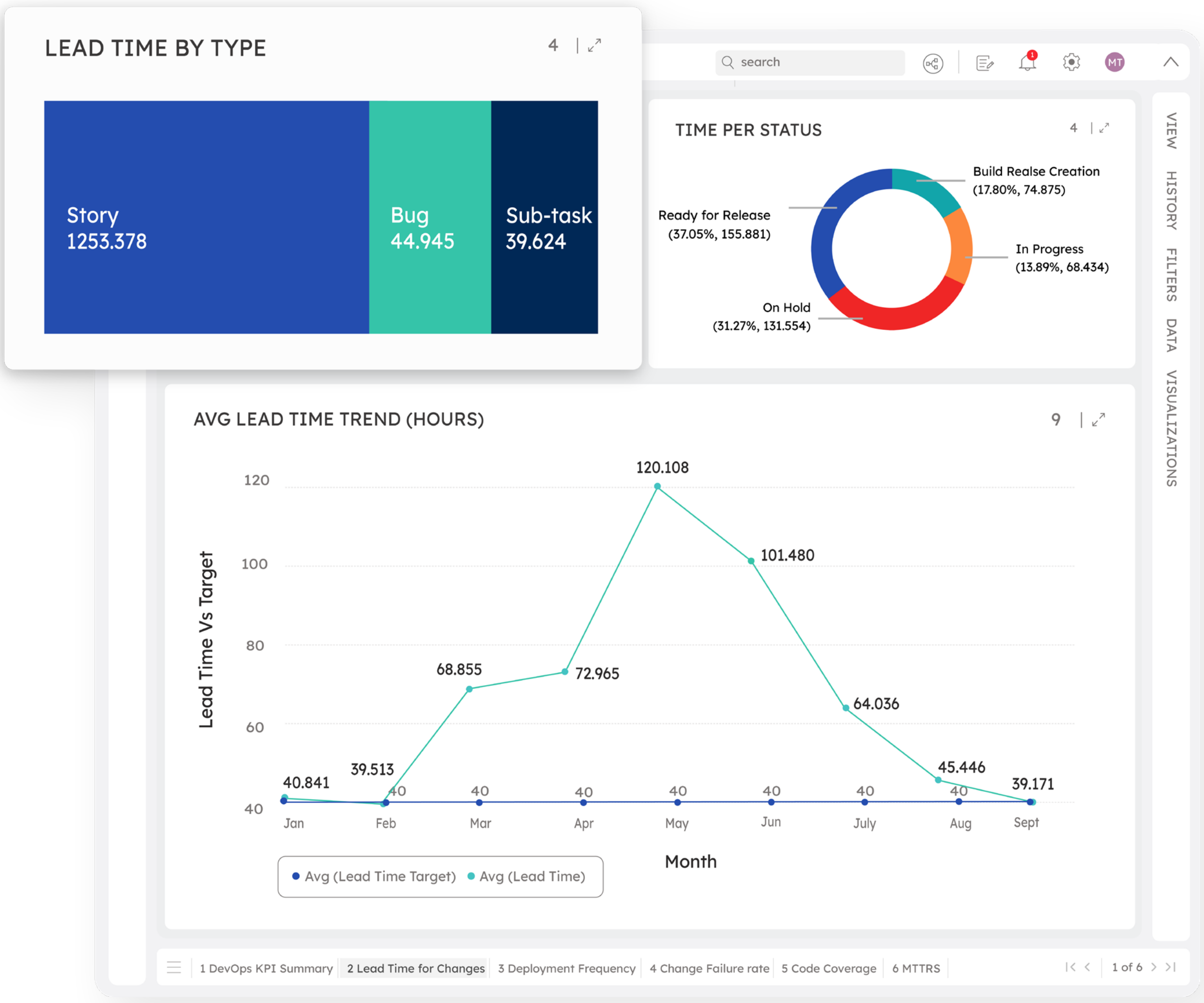

DORA Metrics

62.6	2.2	13.5	6.7
Lead Time (hrs)	Deployment Frequency	Change Failure Rate	Mean Time To Restore (hrs)

● Elite ● High ● Medium ● Low

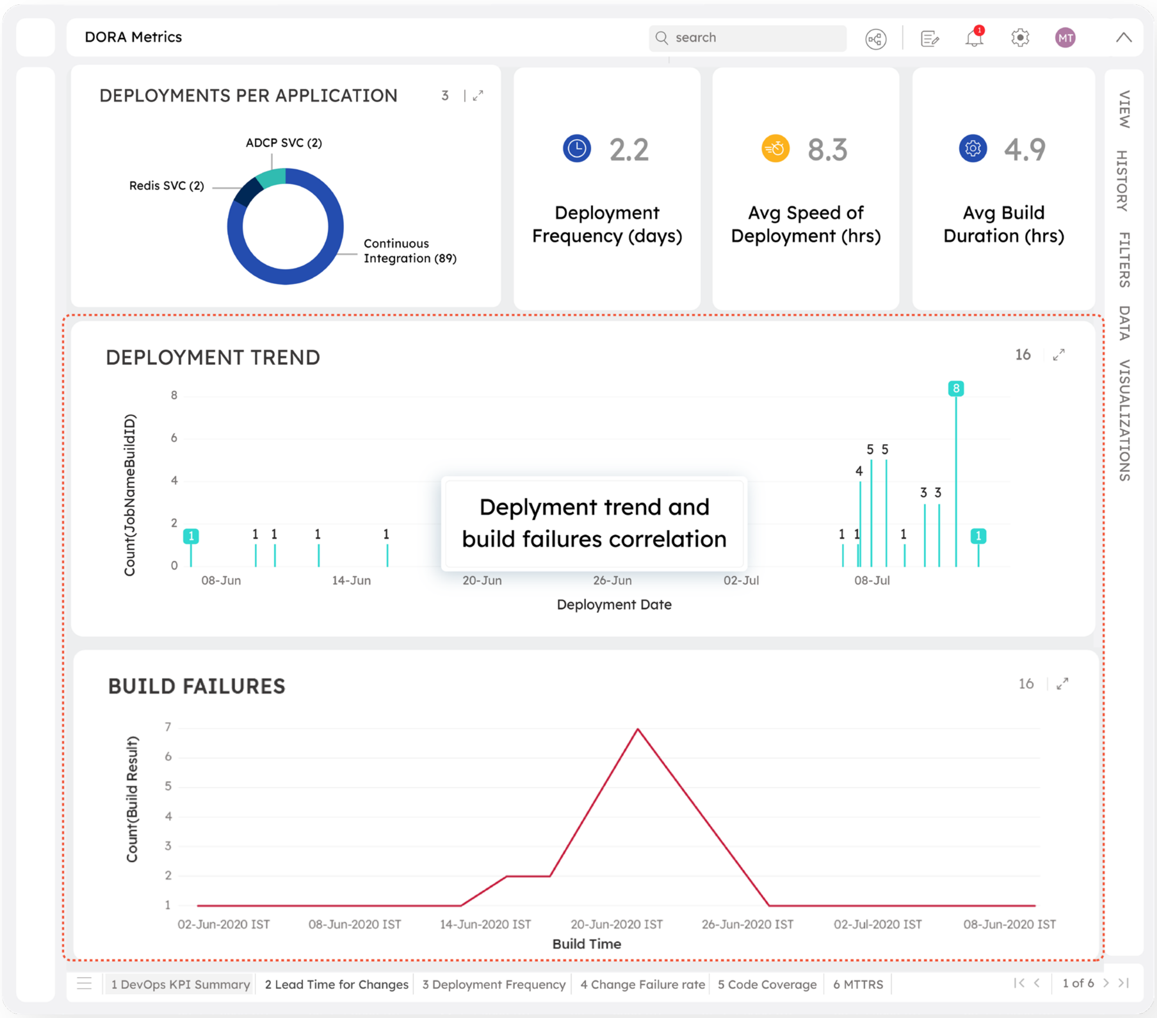
One Size Doesn’t Fit All

Gathr offers an out-of-the-box DORA Metrics solution, which can also be customized to meet your unique needs. For example, an organization developing hardware products might measure deployment frequency differently from someone developing SaaS products. In such cases, Gathr offers increased flexibility to customize metric definitions and get end-to-end visibility into DevOps health and performance.



Drill Down & Across DORA Metrics

You can drill down to each metric; for instance, when you monitor the lead time for changes, you don't only need the historical trends but would also like to see its correlation with the amount of work that the teams are handling and capture the work distribution by its type (bug, story, sub-task, etc.). Such breakdowns are useful if you want to monitor and analyze what kind of work is taking up most of the team's time and get issue details. It is possible to trace the issue details to Jira with a click or add a comment directly from Gathr. If you identify a large amount of work on hold, it might require additional investigations. Gathr can help you in such investigations to understand cross-team dependencies and bottlenecks.



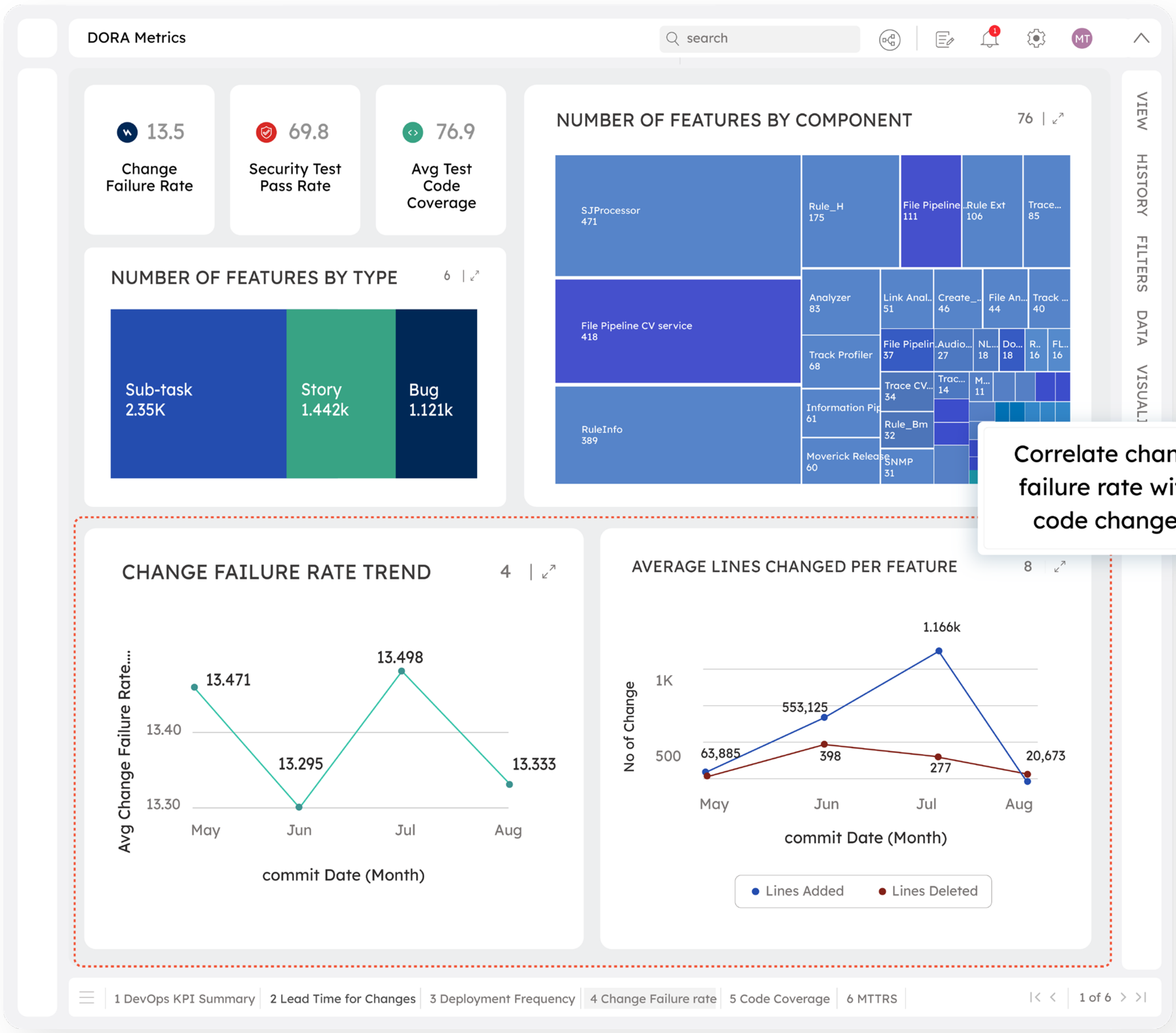
Similarly, you can drill down to the deployment frequency metric. Gathr also tracks the average speed of deployments and average build duration, as these metrics also impact the deployment frequency. The view helps you observe the correlation between deployment trends and build failures. Again, Gathr offers increased flexibility to configure what you need to monitor. For instance, in Jenkins, many times, there are multiple jobs running concurrently, handling different builds and deployments. With Gathr, teams can select the relevant jobs and monitor only them for their work efficiency tracking.

The view helps you observe the correlation between deployment trends and build failures. Again, Gathr offers increased flexibility to configure what you need to monitor. For instance, in Jenkins, many times, there are multiple jobs running concurrently, handling different builds and deployments. With Gathr, teams can select the relevant jobs and monitor only them for their work efficiency tracking.

Get Better Context into Quality

For monitoring the change failure rate, Gathr again offers additional metrics as they offer better context; these are security test pass rate and average test code coverage. The view helps you easily correlate the change failure trends with the code changes (average lines of code added/deleted). You can also get a work breakdown by feature type and component.

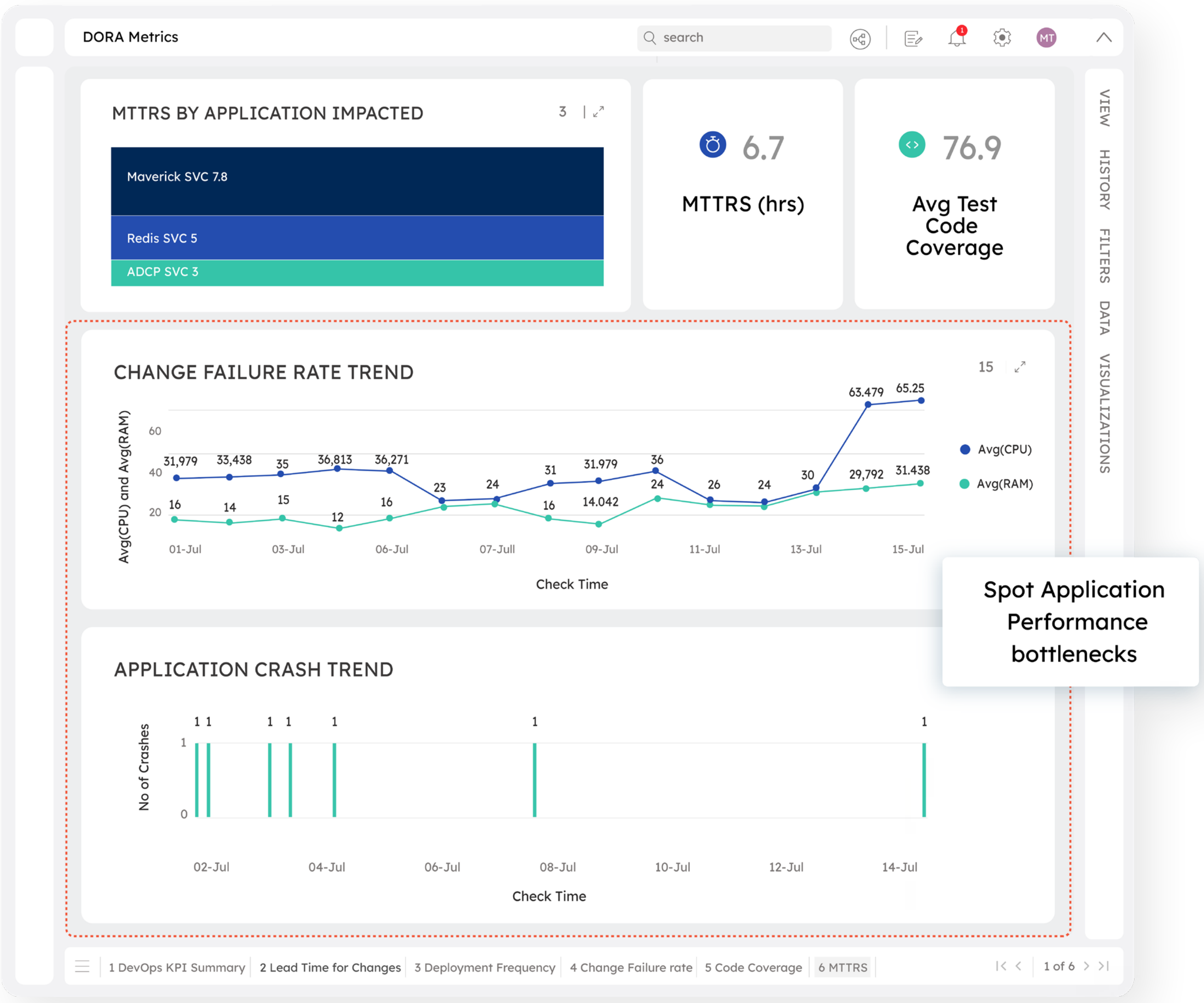
Moreover, it is possible to drill down to metrics, such as code coverage, to identify potential issues with code reviews and change management.



Correlate change failure rate with code changes

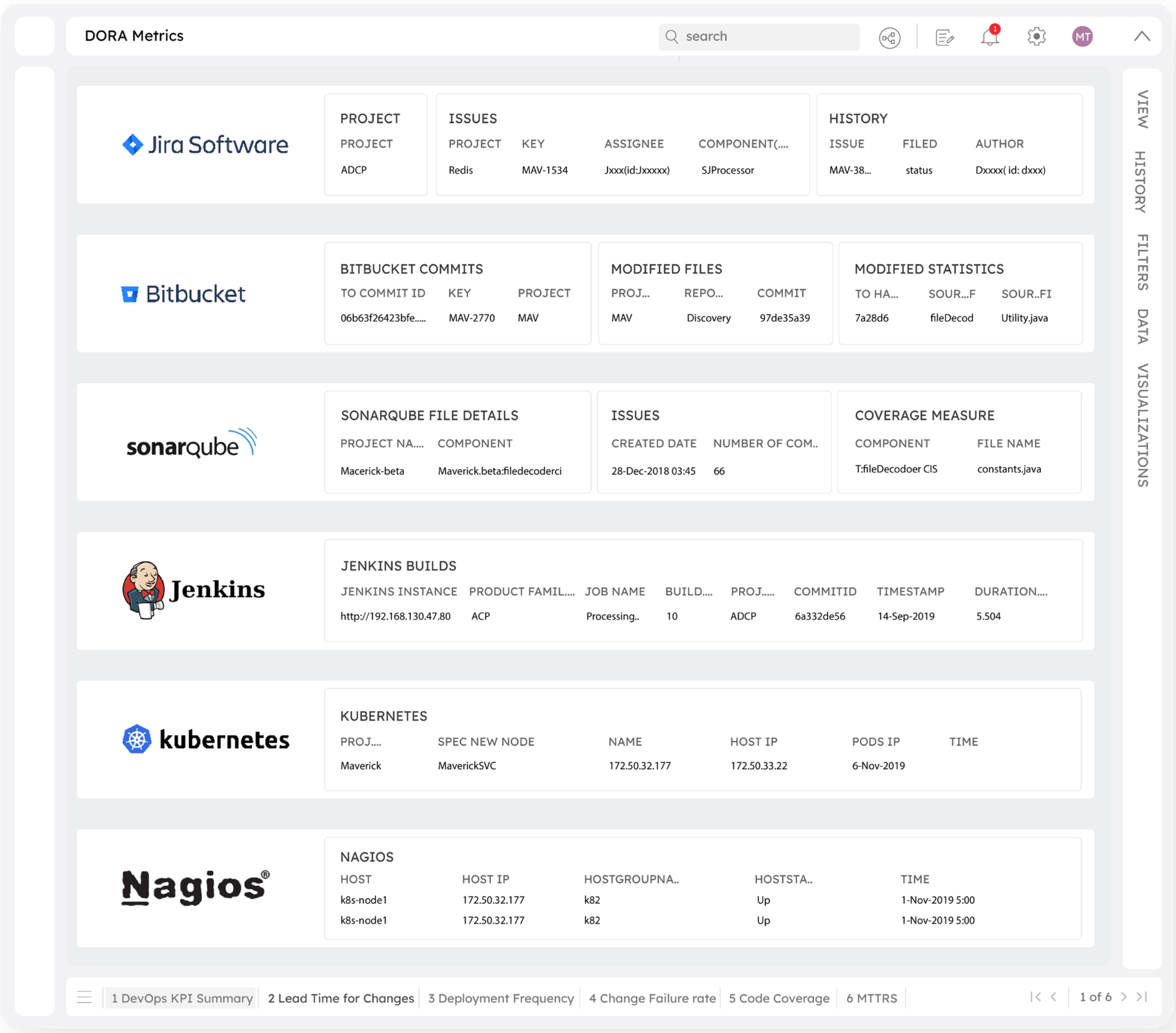
Add Custom Metrics to Gauge Application Performance

You can also configure how you monitor the MTTR metric, adding the details of hosts and resource consumption to understand what's leading to application crashes. Gathr allows you to define your custom metrics using Excel-like formulas and expand your monitoring scope



Onboard Your Tools in Minutes

The solution simplifies data collection from different tools and doesn't require the creation of any data warehouses. It uses smart bi-directional connectors that allow you to collect data from specific instances of your tool within a few simple clicks without requiring any elaborate configuration or coding. This also gives you increased flexibility to onboard new tools at any time to accommodate changes in your toolchain or expand the monitoring scope.



PROJECT

PROJECT
ADCP

ISSUES

PROJECT	KEY	ASSIGNEE	COMPONENT(....
Redis	MAV-1534	Jxxx(id:Jxxxxx)	SJProcessor

HISTORY

ISSUE	FILED	AUTHOR
MAV-38...	status	Dxxxx(id: dxxx)

BITBUCKET COMMITS

TO COMMIT ID	KEY	PROJECT
06b63f26423bfe.....	MAV-2770	MAV

MODIFIED FILES

PROJ...	REPO...	COMMIT
MAV	Discovery	97de35a39

MODIFIED STATISTICS

TO HA...	SOUR...F	SOUR..FI
7a28d6	fileDecod	Utility.java

SONARQUBE FILE DETAILS

PROJECT NA....	COMPONENT
Macerick-beta	Maverick.beta:filedecoderci

ISSUES

CREATED DATE	NUMBER OF COM..
28-Dec-2018 03:45	66

COVERAGE MEASURE

COMPONENT	FILE NAME
T:fileDecodoer CIS	constants.java

JENKINS BUILDS

JENKINS INSTANCE	PRODUCT FAMIL....	JOB NAME	BUILD....	PROJ.....	COMMITID	TIMESTAMP	DURATION....
http://192.168.130.47.80	ACP	Processing..	10	ADCP	6a332de56	14-Sep-2019	5.504

KUBERNETES

PROJ....	SPEC NEW NODE	NAME	HOST IP	PODS IP	TIME
Maverick	MaverickSVC	172.50.32.177	172.50.33.22	6-Nov-2019	

NAGIOS

HOST	HOST IP	HOSTGROUPNA..	HOSTSTA..	TIME
k8s-node1	172.50.32.177	k82	Up	1-Nov-2019 5:00
k8s-node1	172.50.32.177	k82	Up	1-Nov-2019 5:00

Continuous Improvement – **Beyond DORA Metrics**

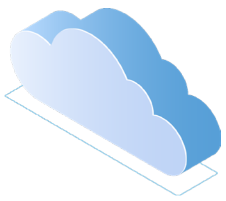
While DORA metrics help organizations quickly assess the level of performance needed to achieve desired business objectives, they can tell only so much . Organizations need a more holistic approach to improve different aspects of software development, including security and cloud operations; DORA has also emphasized these aspects in its recent reports.

Software Supply Chain Security

Software supply chain attacks are now identified as a major risk, as highlighted by the recent SolarWinds exploits, which impacted businesses across all industries. Since the attack, enterprises have become more cautious of the risks in their software development processes, which can allow an easy approach for threat actors to introduce vulnerable code or malware that bypasses network defenders, hiding behind trusted software updates. Security initiatives, such as Supply Chain Levels for Software Artifacts (SLSA) and the NIST Secure Software Development Framework (SSDF) can help organizations better prepare against such attacks. While these frameworks can significantly boost security, there are still gaps in their adoption. Despite awareness, only 18% of the respondents in DORA's survey could confidently claim that the software security protocols like SSDF are seamlessly built into their development process.

It is also seen that organizations fail to implement simple security practices that can prevent such attacks to a great extent. For instance, security scanning as part of CI/CD is now a well-accepted practice, but it's not without its issues. Many times, the scans are performed much later in the process, and developers aren't able to recognize if they are working on a dependency with known vulnerabilities. Similarly, they have to wait for hours to know if their changes have resolved a security issue due to long CI cycles and the inability to run the scans locally. Gathr offers increased flexibility to integrate security tools in CI/CD and improve security posture with continuous compliance and observability.

Cloud Operations



Cloud and DevOps are becoming increasingly interdependent. The advent of practices like Infrastructure as Code has boosted cloud deployments, which are now largely driven by DevOps teams. However, executives have no clear visibility into the efficiency of their infra as code pipelines. Many times, configuration scripts lead to suboptimal resource provisioning. Organizations need a way to implement DORA metrics for monitoring their infra as code pipelines. Gathr addresses this challenge with its out-of-the-box app for IAC monitoring. The app offers an easy approach to adopting DORA metrics for monitoring the IAC pipelines with metrics such as success/failure rates, top errors, failed changes per week, end-to-end time for infra provisioning, and more.

The usage of multi and hybrid clouds has registered high double-digit growths in recent years. Businesses invest in multi-cloud setups for various reasons, including increased availability, the option to leverage the unique benefits of each provider, trust or preference, disaster recovery, regulatory compliance, and more. However, managing and optimizing multi-cloud setups is a complex challenge. Enterprises waste around 30% of the cloud budgets as per their own estimates. Gathr also solves this challenge with its app for multi-cloud cost management. It can offer you quick insights with cloud cost breakdown by region, account, team, services, tags, and more. The app helps you reduce multi-cloud wastage by analyzing spending against budgets and forecasting costs and usage.

Transition from Monitoring to Optimization

Gathr makes it possible to expand the scope of monitoring, move beyond DORA metrics, and create a holistic solution that improves strategic and day-to-day operational decision-making. Gathr's DevOps 360 is purpose- built for such requirements; it combines data, insights, and actions across your infrastructure, application, platform, end-user data, builds and deployments, incidents, and more. With this solution, teams can gauge their application performance trends and make provisioning decisions to maximize the cloud ROI. At the same time, engineering teams can make informed decisions to improve different features and deliver a better end–user experience. You can learn more about DevOps 360 here.

Gathr helps teams develop and operate software with increased visibility and control over data. Please visit our DevOps and Cloud Operations pages to explore an incremental approach for end-to-end observability and workflow automation.