OPSERA

The Ultimate

DevOps Platform Buyer's Guide

Introduction

Software development and the IT industry wouldn't be where they are today without the DevOps revolution.

As the tools and best practices evolved, DevOps matured from a set of agile practices that focused on technological work - automated build, test-driven development (TDD), continuous integration/continuous delivery (CI/CD), etc. - to a set of tools and practices that focus on value stream management and delivery.

Gartner identified four drivers that pushed the conversation from isolated DevOps tools to integrated tooling into DevOps platforms:

1. Cloud adoption and container-native architectures. DevOps tools are integrated into the overall cloud infrastructure and questions about elasticity and scaling determine their successful adoption.

2. Need to simplify building and managing DevOps pipelines – minimize the overhead involved in orchestration, integration, and governance.

3. Need for security and compliance automation as part of the DevOps pipeline.

4. Need for end-to-end visibility, traceability, auditing, and observability into the flow of work – the core drivers for value stream management.

This paradigm shift readjusted the focus from technical optimization to a conversation about delivering customer value.

DevOps tools and processes were not viewed as part of Developers' or Operations' workflows, but as integrated into the value streams that produce quality products for the end customers.

You might be wondering: **“What product should I procure to increase the productivity of my software and IT team?”**

Luckily, this buyer guide will help you answer that question.

A sneak-peak of what is coming:

1. The challenges in the current DevOps practices and tools.

2. How a DevOps platform addresses those challenges.

3. The different types of DevOps platforms.

4. Which features should you expect from a DevOps platform?

5. The expected costs of a DevOps platform.

6. Common pitfalls to avoid when procuring a DevOps platform.

And don’t forget to check the checklist at the end of the guide - it will help you evaluate the best DevOps platform for your team.

The challenges of the modern DevOps

With the growth of different DevOps solutions for niche problems and the increasing complexity of modern software systems, DevOps teams face a range of challenges that can make it difficult to achieve their goals.

From managing complex infrastructure and toolchains to ensuring security and compliance, DevOps teams must navigate a wide variety of obstacles to ensure the success of their initiatives.

In this context, understanding the key challenges of modern DevOps is crucial for organizations that want to build and maintain effective DevOps practices and stay competitive in today's fast-paced technology landscape.

Inconsistent tooling

Different teams may use different tools and technologies, leading to inconsistency in processes and workflows. This is not just a debate between GitHub vs GitLab as your preferred product for version control. The inconsistencies make it harder to standardize processes and quality assurance, keep governance over your deployments, and introduce efficiencies to streamline your product delivery systems.

Complex integrations

Integrating different tools and technologies can be complex. From varying levels of know-how to different configuration, communication, and deployment standards between tools, complex integrations lead to delays, potential errors, and increased costs for overhead. The cost of complex integration is increased after particular DevOps tools breaking changes and version releases.

Lack of visibility and transparency

Managing multiple tools and processes across a fragmented DevOps stack can result in a lack of visibility and transparency across the software delivery pipeline. Each tool silos data and makes it harder to understand the KPIs across tools and teams. This can make it difficult to track the progress of software releases, identify bottlenecks, and collaborate effectively across teams.

Limited scalability and flexibility

A fragmented DevOps stack can limit the scalability and flexibility of software delivery pipelines, making it difficult to adapt to changing business requirements and customer needs. Not every tool is ready for cloud integrations, parallelized processing, multithreading, or other technologies that allow the tooling to scale with the company's needs and processes.

Lack of automation and orchestration

Managing multiple DevOps tools and processes in your fragmented DevOps stack can make it difficult to implement automation and orchestration across the software delivery pipeline. This can result in manual errors, decreased efficiency, and increased costs for the organization.

Limited cross-functional collaboration

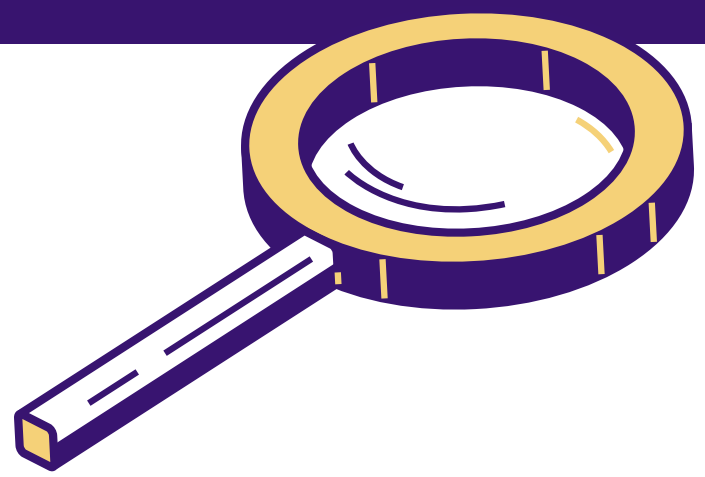
A fragmented DevOps stack can limit cross-functional collaboration between different teams and stakeholders, making it difficult to achieve a shared understanding of software delivery processes and objectives. This can result in misaligned goals, inefficient workflows, duplicated work, underutilized talent, overutilized talent, and decreased productivity.

Increased security risks

Relying on multiple DevOps tools and processes that are not integrated and standardized across your software development processes raises security risks. Each additional tool increases the potential attack surface.

Lower product quality

Lack of cross-stack standardization, automation, and monitoring decreases the overall quality of your software products. Working with a complex DevOps stack increases the chances of manual errors, causes duplicated work, introduces processes of varying quality, and slows down error resolution.



Slower time to market

All of the factors above cause your DevOps initiatives to slow down and delay time-to-market for your software products. Inconsistent tooling standards, complex integrations, and limited collaboration add overhead that delays your product releases. Product quality issues and security threats can result in additional testing and validation processes that further slow down the release process. Limited scalability, flexibility, and automation can make it difficult to adapt to changing business requirements efficiently, causing you to miss opportunities.

If your team is facing some (or all) of these challenges, know that you're not alone.

"Nearly 80% of organizations remain in the middle of their DevOps journey, experiencing varying degrees of success at the team level but not across the entire organization."

- Puppet by Perforce, State of DevOps Report 2023

puppet

The solution: A modern DevOps solution

What is a DevOps platform?

A DevOps platform is a set of tools, processes, and technologies that facilitate the integration and automation of DevOps tools and processes. It provides a unified platform that enables teams to collaborate, streamline workflows, and automate processes across the entire software delivery pipeline.

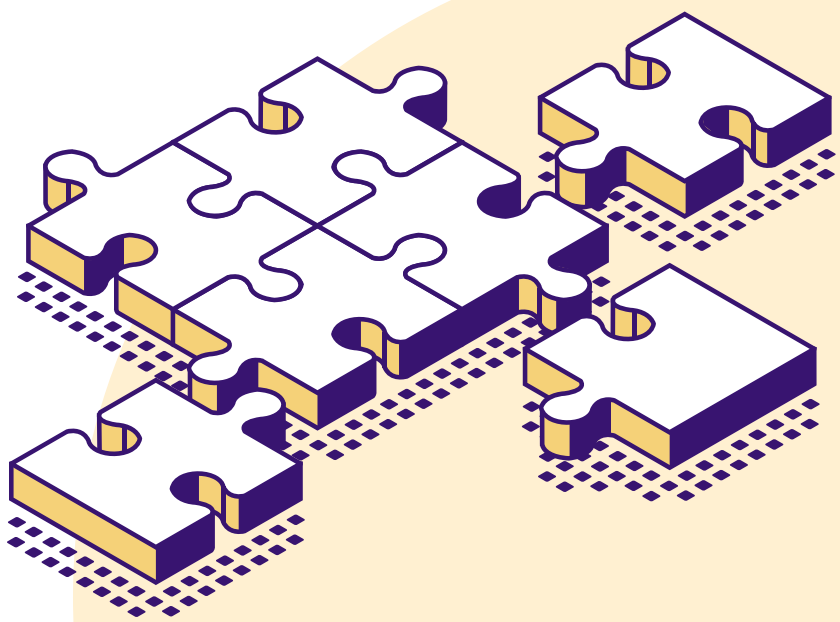
Unlike other DevOps tools, DevOps platforms are tool-agnostic. They connect to existing tools and ingest data from all phases of the software product delivery process into a single engineering platform, making it easier to deploy, test, share, and monitor DevOps pipelines.

Common features found in DevOps platforms

A DevOps platform typically includes tools and technologies for:

Tool integration	Automate the integration of existing tools into a single DevOps platform.
Continuous Integration/ Continuous Delivery (CI/CD)	Automating the build, testing, and deployment of software applications.
Infrastructure as Code (IaC)	Automating the provisioning and management of infrastructure using code.
Configuration management	Automating the configuration of software applications and infrastructure.
Operational insights	Logging, monitoring, collecting metrics, and analyzing data to identify issues, spot opportunity gaps, and improve performance.
Collaboration and communication:	Providing tools for teams to communicate and collaborate effectively throughout the software delivery pipeline.
Secrets management and communication:	The platform needs a secure and scalable way to store and share secrets across your toolchains and pipelines.

The DevOps platform looks like the best invention since sliced bread. How does it achieve this glorious status?



How does a DevOps platform solve today's DevOps challenges?

DevOps platforms were developed as a response to the issues DevOps teams were facing in their software delivery pipelines. As such, the DevOps platform helps to tackle these problems with a combination of features:

DevOps challenge	How does a DevOps platform solve it?
Inconsistent tooling	Standardize tooling configuration, management, access right granting, monitoring, and deployment through the platform-imposed standards.
Complex integrations	Simplified one-click pre-built integrations.
Lack of visibility and transparency	Cross-platform metrics and unified insights in dashboards.
Limited scalability and flexibility	CI/CD pipeline parallelization, reusable pipelines, pipeline templatization.
Lack of automation and orchestration	End-to-end automated CI/CD and IaC platforms that execute via multiple triggers.
Limited cross-functional collaboration	Centralized logic for all your DevOps deployment into a single platform; alerts and notifications for security pipeline, and audit events; integration with collaboration and communication tools.
Increased security risks	Automated security tests with security-as-code, proprietary security solutions, consistent cross-platform role-based access control, and additional features like Bring Your Own Vault and Approval Gates.
Lower product quality	Automated quality tests with quality-as-code, extensive integrations with quality assurance DevOps tools.
Slower time to market	Acceleration via additive effects of automation, streamlining, standardization, and increased scalability.

Don't sweat the details, we'll explore the features provided by DevOps platforms in-depth in a later chapter.

Benefits of using a unified DevOps platform

By integrating and centralizing the entire DevOps stack, DevOps platforms unlock many benefits for their users:

Simplified management

The unified platform streamlines the previously time-consuming management processes without adding value to the DevOps processes. From single-point tool configuration to access control, the centralization of all the DevOps process logic into a single platform releases teams from tedious work.

Better cross-functional collaboration and communication

Teams' work is united into a single platform, which makes it easier to collaborate, even across functional silos and different groups. Talent is focused on end-to-end pipelines instead of spending time tweaking specialized tools.

Improved visibility and transparency

DevOps platforms offer metrics that track performance across the entire software development lifecycle. Making it easier to spot bottlenecks and delays, and recuperating the lost time. Additionally, monitoring based on the entire end-to-end product development lifecycle offers better insights into which levers to pull to increase efficiency. Instead of optimizing individual segments of product development ("Is the Jenkins config written correctly?") companies can focus their efforts on the aspects of the end-to-end pipeline that have the biggest impact on the bottom line.

Increased productivity

By automating CI/CD pipeline orchestration, tests, and security probing, the DevOps teams relying on DevOps platforms can spend less time figuring out which weak link between different tools failed and more time building products.

Increased experimentation

The plug-and-play design of DevOps platforms allows users to test different tools and processes without the integration, management, monitoring, and governance

overhead. The platform takes care of the back office work. DevOps teams can experiment with different tools and find those that increase productivity and have a positive impact on value delivery metrics (time to market, product quality, responsiveness to errors, etc.).

Increased security

By joining all tools into a single platform, DevOps teams can standardize access controls and security tests across all their productization initiatives. Lowering the potential attack surface and mitigating the risks of “missing something”. By automating security tests across the entire DevOps stack, teams can be more confident in their deployments.

Improved product quality

By automating testing and deployment processes, a DevOps platform can help ensure that software is thoroughly tested and deployed consistently and reliably. This can improve the quality of software releases and reduce the risk of defects or downtime.

Faster time to market

All of the benefits above result in faster productization. Companies spend less time worrying about integrations, manual CI/CD steps, error debugging, security tests, traceability, etc., and let the DevOps platform handle the heavy lifting, so your technical talent can spend more time on product development. The best part? Each automation the platform provides is additive, increasing the time savings non-linearly.

“We had built our own pipelines and tooling over a two-year period ... The Opsera team built the integration in two days. I feel like the holistic approach that the Opsera is taking to solve the Orchestrating problems is a key step in the evolution of DevOps.”

- Kevin Railsback, Senior Director of Technical Operations at Reflektion

Must-have features of a DevOps platform

A DevOps platform typically includes tools and technologies for:

1. **Tool integration**
2. **Continuous Integration/Continuous**
3. **Delivery (CI/CD)**
4. **Infrastructure as Code (IaC)**
5. **Configuration management**
6. **Operational insights**
7. **Collaboration and communication**
8. **Security, governance, and compliance**
9. **Secrets management**

But not all platforms are built the same. Some are great for keeping security tight but don't really make your developers more efficient.

Let's figure out how to compare them with a deep dive into the details of the technologies, features, and offerings of the best DevOps platforms.

Keep your eyes open for:

Must-have features	No, these are not negotiable.
Advanced features	Do you *need* this feature? No. Will you be glad to have it? Absolutely!
Pro tips	Not every buying decision needs to be hard. Especially if you get the inside scoop.
Industry case studies and best practices	Get ready to be inspired by industry leaders.

One place for all your integrations

Got a favorite tool? Luckily, you can bring it along to your DevOps platform. One of the most common ways DevOps platforms add value is by **integrating** DevOps tools **out of the box**.

To evaluate a DevOps platform, assess the platform's contribution to the **must-have integration** features:

Ease of integration

Does the platform make integration a one-time effort, or do you need to configure tool integrations with every change? Platforms that simplify integrations are a time saver. You integrate a tool once and let your team spend time using the tool instead of configuring it.

Maintenance

Tools break. And they have this pesky habit of breaking just before a looming deadline. Whether it's an API endpoint change or a version release incompatibility, each massive change inevitably leads to integration maintenance efforts. Look for platforms that actively work on maintenance of their provided integrations and therefore save you time.

What about **advanced integration features**? What should you look for?

Some DevOps platforms offer additional advanced integration features like:

Configuration management

Automate the configuration of an integrated tool once, and re-use the configuration across all your deployments to save time, standardize tool usage across your operations, and increase quality controls. Separate configurations can be used for your development, testing, and production environments, keeping the development branches separated but consistent.

Secrets management

Keep your secrets hidden from prying eyes and let the platform take care of accessing and securing the right user credentials, tokens, and certificates whenever your team uses the DevOps tool.

Toolchain automation

DevOps teams rarely use just one or two tools for all their processes and tasks. Instead, they use multiple tools to achieve continuous integration, continuous delivery, automation, and collaboration. This is when teams rely on toolchains instead of individual tools.

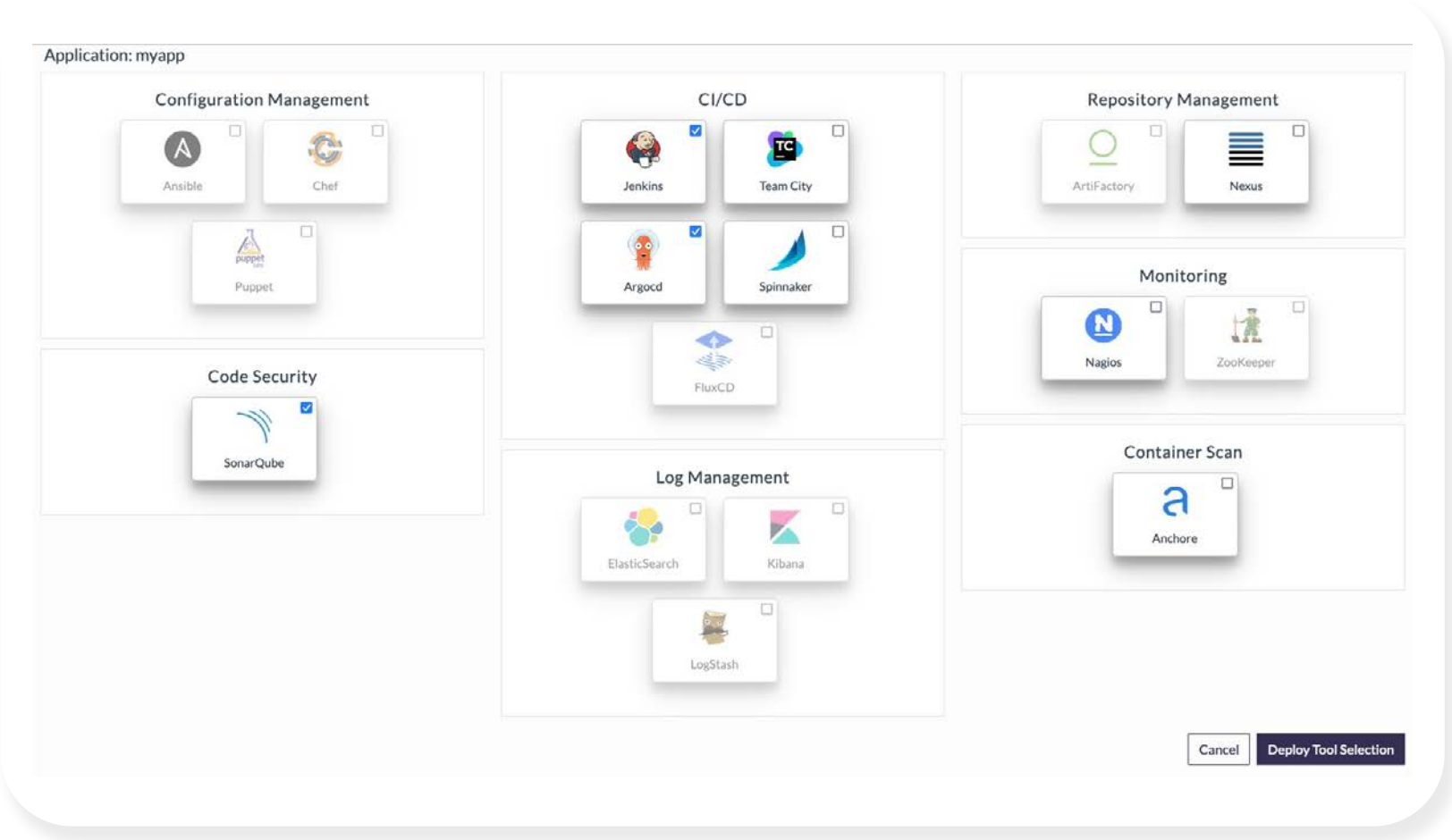
For example, a team can use Jira for planning and workflow tracking, Jenkins for continuous integration, LogStash for log management, SonarQube for code security, and ZooKeeper for monitoring.

The best DevOps platforms allow you to declare customizable toolchains that bundle and deploy your preferred tools together into a single toolchain.

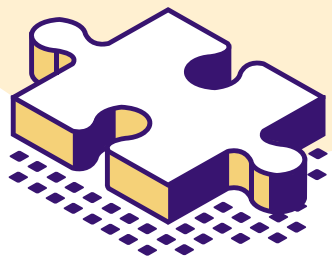
“A DevOps toolchain is a collection of tools, often from a variety of vendors, that operate as an integrated unit to design, build, test, manage, measure, and operate software and systems.”

- Atlassian

ATLASSIAN



Operating with declared toolchains - instead of individual tools - has multiple benefits:



- Improved **standardization**.
- **Governance** over your tooling - Keep granular control over the DevOps stack by team, product, dev environment, and infrastructure.
- Increased **quality control** (specify the minimal security checks for each CI/CD pipeline),
- **Save time** when deploying tools.

Pro tip: check whether your DevOps platform provides role-based access control for toolchain automation, to lower overhead and risks while increasing control and quality.

Automate end-to-end with the CI/CD pipeline builder

CI/CD pipeline builders allow you to declare pipelines that chain DevOps tools together into a single continuous integration/continuous delivery process.

There are many advantages to CI/CD pipeline builders:

1. Reuse	Once a pipeline is declared and deployed, you can reuse it at each code change. Saving you engineering time and effort.
2. Embedded quality and security standards	Because the quality and security standards are defined as a necessary step in the pipeline, you can deploy the CI/CD pipeline with more confidence that everything is according to governance and productization standards.
3. Lower human error rate	Instead of building and triggering each step manually, the pipeline can be triggered automatically in an all-or-nothing manner. Lowering the chances of introducing human errors either by failing to execute a CI/CD pipeline or by missing a step.
4. Scale	CI/CD pipelines are the first step to making your DevOps processes scalable. Because they are reusable, they can be copied and used over many different workflows, projects, and environments.

Improve engineering productivity with the right CI/CD pipeline builder features

Most CI/CD pipeline builders offer users:

1. Intuitive pipeline declaration

The best DevOps platforms ease pipeline creation via **no-code** features such as a **drag-and-drop** UI. This saves time when building new pipelines and simplifies the understanding of existing complex workflows. Some DevOps platforms offer script-based pipelines. These are good, but more time-consuming to create and manage.

2. Multiple trigger options

Declared pipelines can be triggered based on events, time schedulers, or manually.

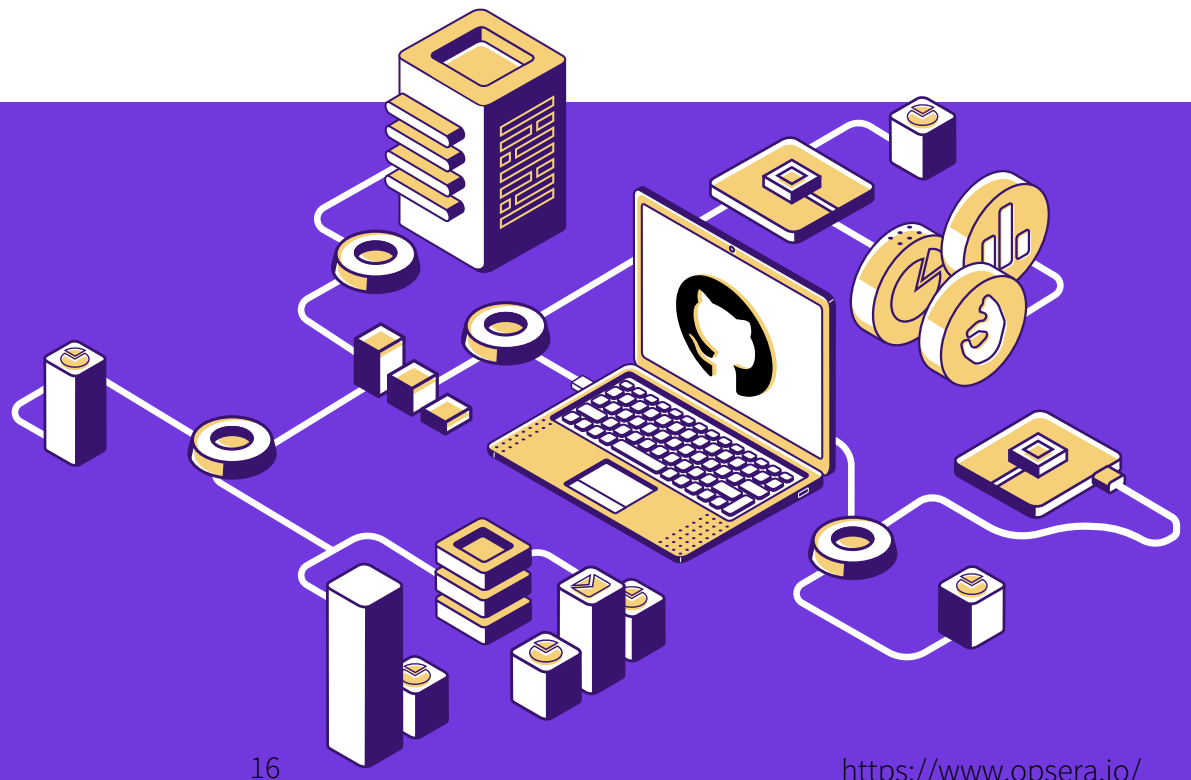
3. Logging and alerts

Each declaration, modification, and execution of the pipeline should be logged to comply with governance and lineage demands. This helps your DevOps team debug errors once they occur. Additionally, users should have the option to set alerts for successful/failed pipeline steps.

4. Pass/fail logic for each step

Define the threshold above which a pipeline step passes or fails.

CI/CD pipeline builders speed up the feedback loops that help your developers improve productivity even faster.



Advanced pipeline builder features - What to look for?

DevOps platform can offer advanced pipeline builder features that help you set up your CI/CD pipelines even faster:

1. **Templetizable and shared pipelines.** Turn your existing pipelines into templates that can be shared between individual contributors, teams, and product departments. Templates don't just save time when building a new pipeline, they can also be used to enforce organization-wide security and quality standards.

2. **Role-based access control (RBAC).** RBAC allows you to set up granular access control over your pipelines and shared templates, specifying which part of the pipelines (triggers, tool configuration, steps, etc.) users can alter. Say hello to increased security and shared quality standards!

3. **Pipelines search functionality.** Companies build many pipelines and not all of them are used in regular DevOps processes. DevOps even have horror stories about the forgotten job that crashed their system. Keep an eye on what's running by searching pipelines with custom tags, filters by product/author, and sorting by creation/run date. Find obsolete pipelines and archive them.

4. **Refined execution control.** Want to trigger a pipeline right after another has finished? Or maybe speed up the CI/CD process by running two pipelines in parallel? Execution control over pipeline dependencies allows you to set up rules in place that automate the complex logic behind running pipelines manually.

5. **Infrastructure-as-code (IaC) pipelines.** Pipelines can also be used to specify and spin up infrastructure. IaC pipelines are a great feature for keeping your environments consistent, reproducible, and monitored.

Whether basic or advanced, CI/CD features will shave hours off your DevOps time.

Industry case study: City of Hope uses Opsera's CI/CD pipeline builder to build DevOps pipelines 80% faster

City of Hope is a nationally-recognized cancer center that has been providing outstanding care to patients for more than 100 years. City of Hope is not a software company, however, a huge amount of data is created from the medical treatments and reports to be analyzed. The organization needed custom software applications to provide ongoing and timely analysis of data in their precision medicine data and systems division.

The challenge
Their previous software development process was time-consuming and manual. As part of the transformation process, City of Hope's goal was to create a new software development process from scratch. They were looking to standardize their SDLC process with security and quality gates.

The solution
With Opsera, City of Hope was able to select the tools from the toolchain automation catalog and integrate them with existing tools using tools like source code management, planning, and collaboration tools. Also, using no-code pipelines, the City of Hope DevOps team built the pipelines with quality and security gates (SAST, Vault, Unit testing) to automate the end-to-end software delivery management.

The results
With the new Opsera's CI/CD pipeline builder, City of Hope was able to build a holistic and integrated DevOps pipeline in 30 minutes instead of the previous 2.5 hours - an 80% increase in productivity. While introducing new tools, deployment abilities, and additional security gates.

Read the full case study.

Improve observability with the right monitoring and logging features

DevOps platforms offer integrations with 3rd Party monitoring (AppDynamics, Dynatrace, Nagios Monitoring) and logging (ZooKeeper, Kibana, Elastic Stack, Logstash) DevOps tools.

Additionally, the best-performing DevOps platforms offer their own logging and monitoring features that span the entire platform ecosystem:

Audit logs and alerts

DevOps platforms log every platform event. From user creation, access revoking, and pipeline modifications. These logs are useful for error debugging, compliance, traceability, and building cross-platform metrics. The best players even offer alerts for important audit log changes (updating a tool configuration or transferring ownership of an asset).

Inspectable and exportable logs

Some DevOps platforms allow you to inspect the raw (non-aggregated) logs yourself, export the logs, and even analyze them. Sure, pre-built analyses are swell. But digging into the logs yourself empowers you to trace issues down to their root causes.

Monitoring

DevOps platforms most commonly expose monitoring by building upon the solutions presented so far. Push audit logs to a 3rd Party tool, set up alerts, or even monitor your resources via metrics and KPIs within unified insights.

Get the entire picture with unified insights

DevOps platforms promise to unify insights across your entire DevOps toolchain. No need to spend time on 10+ different tools trying to get one or two tool-specific KPIs. The platform takes care of metric computation and dashboarding for you.

Which metrics should the DevOps platform report on?

The specific metrics reported differ between DevOps platforms. Look for the following groups of metrics.

Increase developer productivity with developer productivity metrics

Developer productivity metrics collectively provide insights into developer productivity, collaboration, and efficiency within the development process. When broken down by user, developer productivity metrics help you assess the individual contributor's output, identify performance gaps, and set up roadmaps for talent improvement.

"With this automated solution [Opsera DevOps platform], the productivity of our engineers improved by 25%. Now, I can see the development and quality metrics of individual contributors."

- Kishore Gandham, Founder & CEO KeyWest Networks

KEYWEST NETWORKS

Product quality metrics - make sure your software is up to standards

Product quality metrics give you a quick overview of software product quality issues and standard abiding. They can be used to spot bugs before they are shipped to production, determine the speed of error resolution, and set benchmarks for improving issue response time.

DORA performance metrics - 4 metrics that best predict success

DevOps Research and Assessment (DORA) conducted research over the past eight years surveying more than 33,000 professionals around the world (the largest and longest-running research of its kind) to determine what metrics software organizations should track.

DORA determined 4 metrics that measure software delivery and operational (SDO) performance:

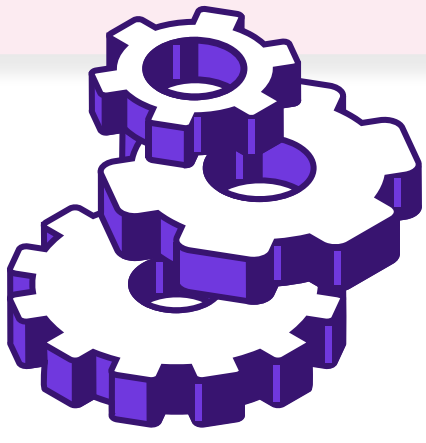
1. Deployment frequency - Frequency of your organization deploying code to production or releasing it to end users. The metric ranges from on-demand (multiple deploys per day) to once every 6 months.

2. Lead time for changes - The time it takes to go from code committed to code successfully running in production. The metric ranges between one day to 6 months.

3. Change failure rate - The percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch). The metric ranges between 0-60%.

4. Time to restore service - The time it takes to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment). The metric ranges from less than a day to 1 month.

Among all the tested metrics, these four DORA metrics showed the highest predictive accuracy, correlating highly with fast and quality software delivery and exceptional operational performance.



Pipeline metrics - keep your end-to-end pipelines performant

Pipeline metrics provide valuable insights into the performance, efficiency, and reliability of the DevOps pipeline. By monitoring and analyzing these metrics, teams can identify areas for improvement, optimize their processes, and deliver software more effectively and reliably.

Industry case study: Measure pipeline metrics and improve CI/CD pipeline performance

Reflektion is an AI-driven customer engagement platform, located in San Mateo California, that offers shopper insights and product intelligence solutions for retailers and brands. Leading retail brands such as TOMS, Ann Taylor, Sur La Table, Godiva, and Destination XL rely on Reflektion's platform.

The challenge
Reflektion had two different CI tools (Jenkins and Codeship) and was unable to establish the relationship between the two for the last couple of years. This caused them to overspend time by reviewing multiple tools and consoles to troubleshoot the issues for their builds, deployment, and other operational issues with their DevOps ecosystem.

The solution
Opsera DevOps platform helped them with the integration of both CI tools (Jenkins and Codeship) without any custom code from Reflektion's DevOps team and provided end-to-end visibility across their build and deployment process. This enabled them to identify the issues and correlate them across multiple DevOps tools in just a matter of minutes.

The result
The pipeline mean time to resolve build issues (MTTR) was reduced by 20%. While providing end-to-end visibility and logs in a single place (Opsera's platform).

Read the full case study.

Security metrics - ensure security by keeping an eye on the right KPIsright KPIs

Security metrics play a crucial role in evaluating and improving the security posture of software applications. By monitoring and analyzing these metrics, teams can proactively identify and address security vulnerabilities, enhance code quality, and ensure the robustness and reliability of their software.

Don’t settle for dashboards that you can’t customize

Without the ability to select metrics and build custom dashboards, you’ll always spend more time than necessary in extracting insights from your data.

When you customize dashboards and reporting - by individual contributor, team, product, pipeline, or another breakdown - you can easily spot the trends, identify productivity bottlenecks, and act upon the data.

Security and quality

DevOps platforms go beyond the sum of the quality and security tests they integrate. The security and quality benefits can be seen on multiple levels:

Standardize tooling

Standardize the required security and quality tests by registering and sharing the quality and security tools to your toolchain automation.

Democratize tooling

Give your team the choice to play with their preferred tools. Standardization helps you keep quality control. But also give freedom to teams to build up their preferred DevOps stacks in the toolchain automaton that works best for them. Freedom, creativity, and innovation go hand in hand.

Codify test logic in security-as-code and quality-as-code tests

Make tests necessary by building them as a requisite step of your CI/CD pipelines. Without a test, the pipeline can’t pass.

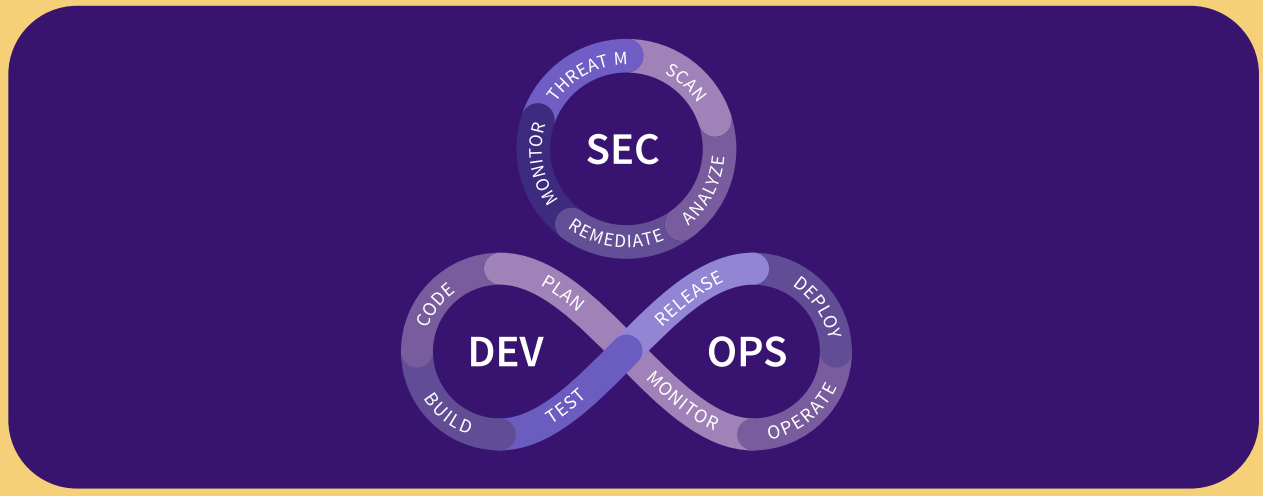
Lower barrier to test

By creating reusable, shareable, and even templatizable pipelines your organization lowers the psychological barriers to testing by diminishing the effort required to embed tests into the CI/CD pipeline.

Resolve security and quality issues faster

Use audit logs and security and quality metrics to quickly spot errors and resolve them faster via log inspections. Set up alerts to respond quickly to any issues.

Industry best practice: Shift-left security



In DevOps, security scanning and evaluations are traditionally performed at the end of the software development lifecycle. As a result, resolving security vulnerabilities has become complicated, expensive, and time-consuming. And DevSecOps emerged as the right solution to this challenge. To do so, DevSecOps follows the Shift-Left approach.

The Shift-Left approach in DevSecOps encourages the teams to integrate security into the SDLC lifecycle as early as possible. It moves security from the right (end) to the left (beginning) of the DevOps SDLC lifecycle. So, in DevSecOps, the security is baked into the development process from the beginning, allowing the teams to identify security threats early and ensure those threats are addressed immediately.

Additional advanced security features

Some DevOps platforms take security even more seriously and offer advanced features and tools to help secure your software development initiatives:

Owned solutions to manage secrets and protect sensitive data

From encryption to access barriers, DevOps platforms develop their proprietary security features to protect your configurations, passwords, tokens, and other sensitive data.

Role-based access control (RBAC) at every level

RBAC is a critical component of preventive security. By centralizing all access controls within the platform and offering features to manage roles, companies lower the attack surface of user credentials.

Bring Your Own Vault

Vaults encrypt secrets to help prevent unauthorized users from gaining access. They act as active storage containers for secrets (passwords, API keys, SSH keys, or RSA tokens) as well as an account management system for dealing with multiple privileged accounts. Users can configure their own vault providers to keep secrets secure even from the DevOps platform.

Approval Gate

Approval gates configurations allow users to define the approval for a particular pipeline step execution. The configuration determines if the step can be advanced for execution based on the approver's response to the requests. The approver can choose to either approve or reject the request. This feature is especially useful for sensitive pipelines.

"Opsera's approach of no code addition of security gates and thresholds into every stage of the pipelines by out-of-the-box integration to a choice of security tools greatly improves the proactive security posture for software delivery. Unified real-time security insights are very useful for security managers and add to the value proposition."

- Abhay Salpekar, Director of Engineering, Snowflake

Scalability

The core features of a DevOps platform empower companies with greater scaling ability:

1. Reusable CI/CD pipelines can be used for multiple use cases, environments, and products. Cutting development time whenever you reuse a pre-built pipeline and lowering the chances of new errors.

2. Infrastructure-as-code pipelines can build new infrastructure with one-time configuration and provisioning efforts. Scale your infrastructure to new projects, teams, and environments with a single click.

3. Pipeline parallelization can augment scaling results. No need to wait for one pipeline to finish before you run another. Simply execute in parallel to get more work done at the same time.

4. Pipeline templetization and **toolchain declaration** can standardize quality and security practices across the organization. Making sure you don't sacrifice the quality or security of your products as you scale your operations and processes.

Pricing

Prices range from no upfront costs for Free and Open Source Software (FOSS) that offer bare-bone functionalities to four figures per month for best-in-class enterprise DevOps platforms.

And these solutions are incomparable - FOSS requires lower initial investments (licensing fees, set up fees, subscriptions, etc.), but is more costly down the line (customization, maintenance, error resolution, etc.), and vice versa for enterprise vendors.

To determine whether the pricing makes sense for your business, do two things:

1. Evaluate a solution (look for vendors that offer a free trial).
2. Calculate if the Total Cost of Ownership (TCO) for your existing solution outweighs the vendor price tag.

What to include in the TCO of your existing solution?

Calculate your current integration costs

This includes all the hired external consultants and in-house engineering hours spent integrating, maintaining, updating, and debugging your integration solution.

Consider CI/CD pipeline implementation costs

Evaluate the costs associated with setting up and implementing the DevOps pipelines. This can include expenses related to installation, configuration, customization, training, and consulting services.

Assess staffing costs

Consider the personnel costs associated with managing and operating your existing DevOps solution. This includes salaries, benefits, training, and any additional staff required for deployment, maintenance, and support.

Determine your opportunity costs

These are the hardest to calculate. Perform a back-of-the-envelope calculation anyway to estimate the potential cost savings and additional revenue of time-savings gained from DevOps platforms and increased speed to market.

Though the TCO calculations are hard to make and are rough estimates at best, they can be a great tool for assessing whether a DevOps platform showcases a positive ROI, or whether the price tag is not even in the same ballpark as your existing solution.

Common pitfalls to avoid

Many DevOps platforms look good on paper but live short of the hype once they are implemented. To avoid committing time and resources to the wrong solution for your use case, consider the common pitfalls to avoid when selecting your DevOps platform vendor.

Does the platform cover all your integration needs?

Check if the DevOps platform covers all the tools your DevOps team is currently using. It would be a shame if you went through the entire tool provisioning conversations only to realize half your tools can’t be integrated.

When comparing your existing DevOps stack to what the vendor offers, think ahead. What other tools did your team want to test, but couldn’t due to time constraints and integration complexity? Chances are your vendor

Pro tips for tool integration coverage:

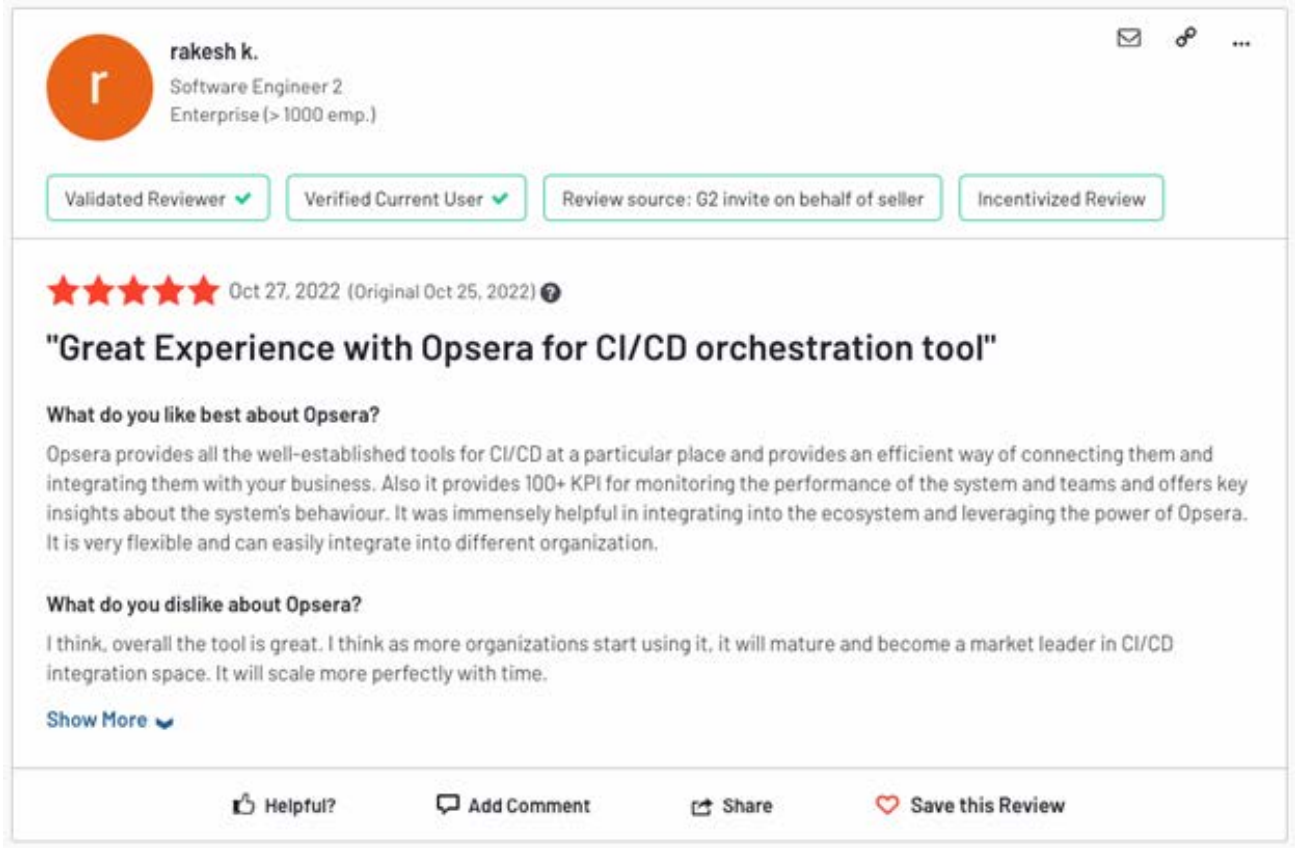
Prioritize DevOps platforms with a higher number of integrations. The more tooling integrations a platform offers, the higher the chance it will cover your existing and future needs.

Ask the DevOps platform what’s their usual process for including a new tool that is currently not covered. Some platforms offer generic integrations (via webhooks or APIs) that you can easily code yourself, an open-source community that develops integrations, or offer a service to build a new integration for you (integration-on-demand).

Follow customer satisfaction and industry praise, not promotional materials

Many vendors know how to sell themselves. Don’t trust marketing and sales brochures, check how satisfied the customers who regularly use the DevOps platform are.

The best places to check for customer feedback are reputable industry vendor assessment platforms such as G2.



Customer feedback is a great place to start, but the above-mentioned platforms also offer in-depth industry analyses comparing tools and recognize the best performers with awards. Look for award badges that signify best-in-class performance.

Are the offered metrics and KPIs informative or regurgitated?

Almost every DevOps platform offers intelligence and insights. The distinguishing factor between leaders and laggers in this space is their ability to compute metrics across multiple tools covering the entire DevOps lifecycle and multiple toolchains.

Check which metrics are offered by your DevOps platform. Chances are they just export GitLab/GitHub metrics, which don't add value, since those can be viewed within the tool themselves.

Pro tip

A good rule-of-thumb for evaluating metric informativeness is their number. Platforms that pre-compute more KPIs are more likely to offer the insights you need.

Should you look for DORA and no other metrics?

Companies sometimes track performance with metrics that seem sound but don't correlate with software delivery performance.

A common example is "lines of code written". On the surface, the metric seems to measure software development productivity, but without a link to product quality and value-added, the metrics can quickly become a vanity tag instead of a key performance indicator.

So why pick DORA metrics instead of "lines of code written" or another metric?

The main argument is data-driven metric validation. The 4 DORA metrics distinguish high-performing teams from low-performing ones, consistently, across companies of all sizes and DevOps complexities. For example, high performers have 417x more deployments than low performers, on average.

Don’t mistake local optima for overall performance improvement. Track the metrics which follow value delivery and correlate with performance.

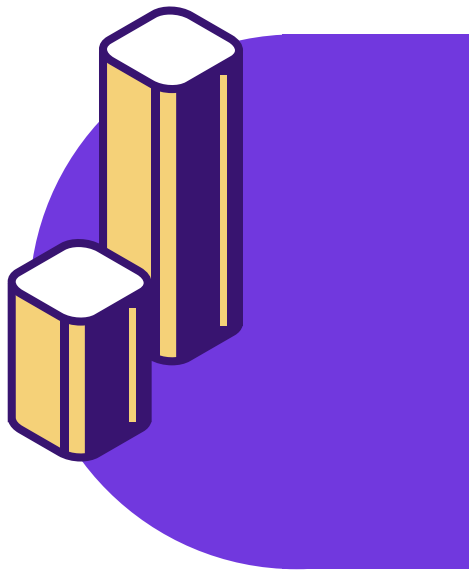
But you might ask yourself: Why would I need any other metric except for DORA metrics? DORA metrics have high predictive validity but offer low contextual information. Once the DORA metrics help you identify bottlenecks, you can turn to other metrics to understand how to resolve those bottlenecks.

For example, a low deployment frequency can be due to a prolonged review process, issues with code quality tests, infrastructure-as-code failures, or a myriad of other causes.

So don’t fall for the DORA-only pitfall. Yes, make sure your DevOps platform tracks DORA. But also make sure you can look at follow-up metrics to dig into the nitty gritty details of why a DORA metric is doing well/poorly.

Don’t just track metrics, track KPIs

Metrics are quantitative measures that help organizations track their progress and performance.



However, not all metrics are Key Performance Indicators (KPIs). KPIs are specific metrics that have been identified as critical to an organization’s success and are used to measure progress toward a specific **business goal** or objective.

To turn a metric into a KPI, you need to follow these steps:

1. Define your business objectives

Identify the goals or objectives that are most important to your organization.

2. Identify the corresponding metrics

Determine which metrics are relevant to measuring progress towards those objectives. The most critical metrics that directly impact your business objectives are your KPIs.

3. Set targets

Set specific targets for each KPI that align with your business objectives. Usually, this is a benchmark or threshold. A metric needs to be above (e.g. deployment frequency) or below (e.g. number of bugs) a given target.

4. Monitor and analyze

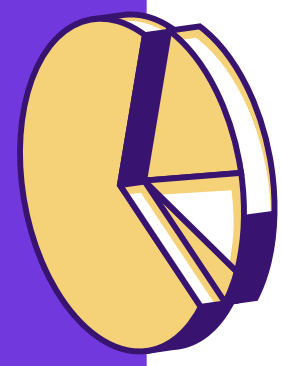
Track the performance of your KPIs over time, and analyze the data to identify trends and opportunities for improvement.

5. Take action

Use the insights gained from analyzing your KPIs to make informed decisions and take actions that will drive progress toward your business objectives.

Remember, a metric is just a number. It becomes a KPI when it is linked to a specific business objective, has a target, and is regularly monitored and analyzed to drive action.

Pro tips for buyers



1. To better track KPIs, pick platforms that allow you to **break down** and analyze metrics by:

- **Time**

Has the metric improved month-over-month?

- **Product**

Which product needs your attention most to improve its value delivery?

- **Team**

Which team is performing better and which needs your assistance?

- **Individual contributors**

Metrics are a great conversation starter for empowering individuals, identifying talent gaps, and bringing objectivity to your PRs and 1-on-1s.

2. Keep an eye on your most important KPIs by setting up **alerts** and **notifications**. Push notifications to Slack, email, Microsoft Teams, or others can either be used to celebrate victories (10 deployments in a day!) or to alert you to a problem before it escalates (a critical number of pipelines failed to build). Shortlist platforms that offer alerts and notifications for crucial metrics.

Ease of use leads to success

No one wants to work with a piece of technology that is a frustration superstar to set up and use.

Check how intuitive the platform is by running a demo with the provider.

Additionally, check for features that lower the barrier to entry for using the DevOps platform:

- **Extensive documentation with how-to guides**

This will help empower your workforce with self-service knowledge for using the platform independently from the platform's account managers.

- **No-code, drag-and-drop features**

No-code features speed up pipeline development. But they also lower the barrier to entry for new talent joining your team, reusability across different contexts, and inclusion of non-technical experts on your time onto your DevOps practices.

- **On-demand webinars for advanced use cases and industry best practices**

DevOps vendors that care about their users constantly update new content that teaches and empowers their users how to get the most out of their product.

Set your team up for success by giving them solutions that delight them, not dishearten them.

Checklist - What to look for when buying a DevOps platform?

[Drum roll] The moment we've all been waiting for is here: say hello to the buyer checklist!

We've summarized all the crucial features of a DevOps platform in the following checklist. So you don't forget or miss any important steps.

Fill in the checklist as you evaluate your tool of choice.

We've already pre-filled Opsera - one of the best DevOps platforms on the market right now - as an example to follow along.

	Feature	Opsera	Vendor 2	Vendor 3
	Integration			
	Number of integrated tools	80+		
	Covers all existing DevOps tools	✓		
	Covers future DevOps tools	✓		
	Generic integration capabilities (webhooks, APIs, integration-on-demand)	✓		
	Easy to use: One-time integration effort	✓		
	Platform takes care of integration maintenance	✓		
	Automated configuration management	✓		
	Secrets management	✓		

Feature		Opsera	Vendor 2	Vendor 3
Toolchain automation				
	Declare and deploy custom toolchains	✓		
	Role-based access control over toolchains	✓		
CI/CD pipeline builder				
	Intuitive pipeline builder (no-code, drag-and-drop)	✓		
	Event-based triggers	✓		
	Time-scheduled triggers	✓		
	Manual triggers	✓		
	Multi-level logging - info (create, update, delete), warning, errors	✓		
	Pipeline alerts for success/fail	✓		
	Customizable thresholds and definitions of pipeline pass/fail	✓		
	Reusable pipelines	✓		
	Shareable pipelines	✓		
	Templatizeable pipelines	✓		
	Role-back access control for pipelines	✓		
	Pipelines search functionality	✓		

Feature		Opsera	Vendor 2	Vendor 3
CI/CD pipeline builder				
	Refined execution control	✓		
	Infrastructure-as-code-pipelines	✓		
Observability: Monitoring and logging				
	Creates audit log	✓		
	Exposes audit log for analysis and export	✓		
	Customizable audit log alertslog	✓		
	Monitoring via 3rd Party integrations	✓		
	Monitoring via audit log alertslog	✓		
	Monitoring via metrics and insights	✓		
Unified insights				
	Number of out-of-the-box metrics	150+		
	Cross-platform metric reporting (not just individual tool reporting)	✓		
	Developer productivity metrics	✓		
	Product quality metrics	✓		
	DORA performance metrics	✓		

Feature		Opsera	Vendor 2	Vendor 3
Unified insights				
	Pipeline metrics	✓		
	Security metrics	✓		
	Customizable dashboards	✓		
	Insights notifications for KPIs	✓		
	DORA performance metrics	✓		
Security and quality				
	Integrates with tools providing quality tests (unit, integration, functional, performance, load, stress, regression, and acceptance tests)	✓		
	Integrates with tools providing security tests (SAST, DAST, SCA, container security testing, IaC security testing, vulnerability scanning, and authentication and authorization testing)	✓		
	Encrypts and secures secrets out-of-the-box	✓		
	Role-based access control at every platform level	✓		
	Bring Your Own Vault	✓		
	Approval Gateplatform level	✓		
	Security alerts	✓		

Feature		Opsera	Vendor 2	Vendor 3
Pricing				
	Platform ROI outweighs the cost	✓		
	Free trial	✓		
Ease of use				
	No-code, drag-and-drop features	✓		
	Extensive documentation with how-to guides	✓		
	On-demand webinars for advanced use cases and industry best practices	✓		
	Scalable	✓		
Customer satisfaction				
	Average customer satisfaction on G2	4.6/5		
	Number of recent awards for best-in-class performance	5		