OpsMx

Best Practices for Implementing DevSecOps

Table of Contents

Introduction to DevSecOps

DevSecOps is about integrating security and compliance testing into the DevOps pipeline without compromising the speed and agility of continuous delivery. From a team perspective, the collaboration between IT security and the product team must be enhanced to make the software lifecycle agile and responsive.

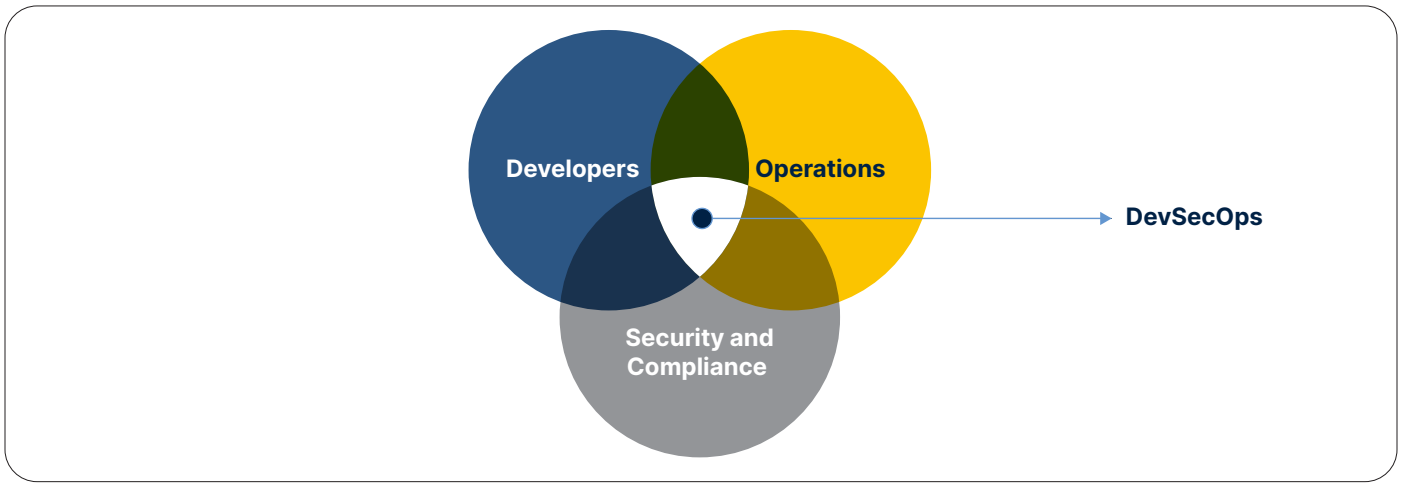


Figure 1: Representation of various teams that form a DevSecOps group

With the rapid speed of software release, security and compliance checks at the end of the process can make the DevOps process rigid and slow. Be it highly regulated banking applications or e-commerce sites, maintaining a security posture has become critical to businesses.

With the rise in ransomware, use of open-source software, and increasing need for faster software deliveries, it is critical to include security as an integral part of DevOps processes. Below are examples of security and compliance checks that need to be performed:

- Vulnerable new libraries
- Expired licenses
- Exposed passwords
- Misconfigured Kubernetes components
- Adherence to organizational standards

Manually analyzing security issues in an application is time consuming and not scalable. And multiplying requests for developers to look into security and compliance issues delay the delivery process.

Top Drivers for DevSecOps

Wide Adoption of Microservices - Modern enterprises use a host of browser and mobile-based applications which use microservices. Microservices expose communication APIs, which means they increase the attack surface.

Too Many Underlying Cloud Technologies - There are many services that use microservices such as databases, message brokers, and service meshes, which introduce even more applications and more surface area for a security attack.

The Rise of Containers - Most modern applications are based on Kubernetes containers, which further raises security concerns. If one Kubernetes pod is breached due to a security issue, all other pods can also be easily attacked (also known as complex attack vectors).

Using microservices, cloud platforms, and Kubernetes makes applications prone to attack and establishes security as a fundamental problem in IT organizations.

The security team needs to learn all these technologies to identify the issues. The learning curve becomes a significant bottleneck for software security testing. Hence, security has to be automated and integrated into your software delivery pipelines.

Three Pillars to Achieve DevSecOps

Security cannot be treated as siloed operations. It must be a shared operation between the product team and the security team. Integration of security best practices into all stages of deployment is essential to mitigate software risks while frequently deploying software into production.

There are three essential DevSecOps pillars to ensure your software and software delivery process are secured without hampering release velocity:

Figure 2: The Three Pillars of DevSecOps

Integrate Security into the Delivery Pipeline

Every application has to go through a list of security tests: static application security test (SAST), dynamic application security test (DAST), fuzzy testing, dependency scanning, container scanning, license compliance, and secret detection. The security team uses various tools to understand the security posture of the project or the group of tasks:

- SAST- Sonarqube or Bandit or HCL AppScan, etc.
- DAST- HCL Appscan, OWASP ZAP
- Container Image scanning (Kubernetes or VM Scanning) - Aquasec

The security team developers and project managers should have access to the security scan reports containing the issues and vulnerabilities. These tools have to be part of the delivery pipeline so managers can view the information and make decisions to halt the deployment pipeline in case of critical errors. Developers can then immediately iterate the merge request, without any context switch, until all the vulnerabilities are resolved.

Examples:

Potential vulnerabilities like SQL injection vector are found during SAST, by scanning the application source code (like JSP, XML, CSS, etc.). If this occurs, the delivery pipeline must fail automatically.

During DAST, a security tool scans dependencies and finds the application is using older versions of libraries. In this case, the delivery pipeline must fail automatically.

Leverage OpsMx Intelligent Software Delivery (ISD) to Integrate Security into the CD Pipeline

OpsMx Intelligent Software Delivery (ISD) is an AI/ML-based Continuous Delivery platform that can enable continuous security by integrating security checks in the delivery automation. OpsMx ISD contains an extensible and scalable orchestration layer, OpsMx Enterprise for Spinnaker (OES), that is based on open-source Spinnaker, and a delivery intelligence layer, OpsMx Autopilot, that helps IT teams deploy software safely and securely.

OpsMx ISD can integrate with any SAST/DAST tools (see Fig. 3A) to ensure security checks are not bypassed during the delivery process and provide the results in the approval dashboard. With consolidated information such as Jenkin builds, JIRA tickets, and security scan results, project managers can take a call to approve or abort a pipeline (see Fig. 3B). They can then notify developers to work on the issue or allow the deployment by dismissing false positives.

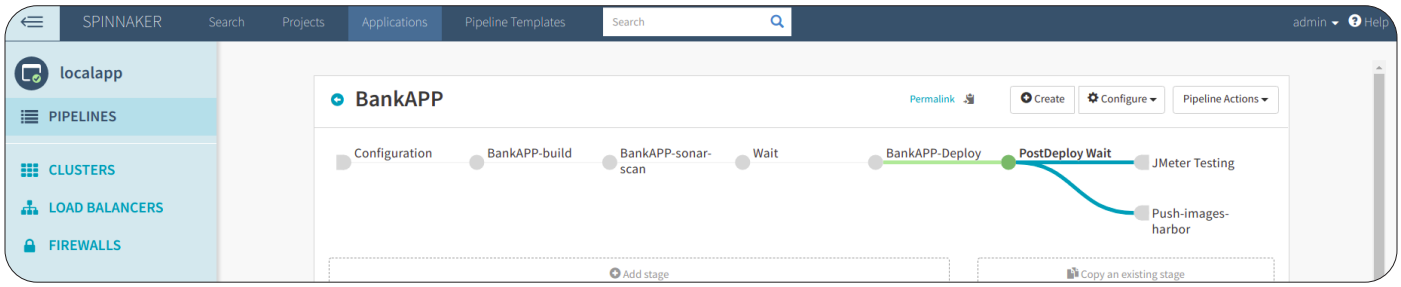


Figure 3A: Making security a part of a Spinnaker pipeline

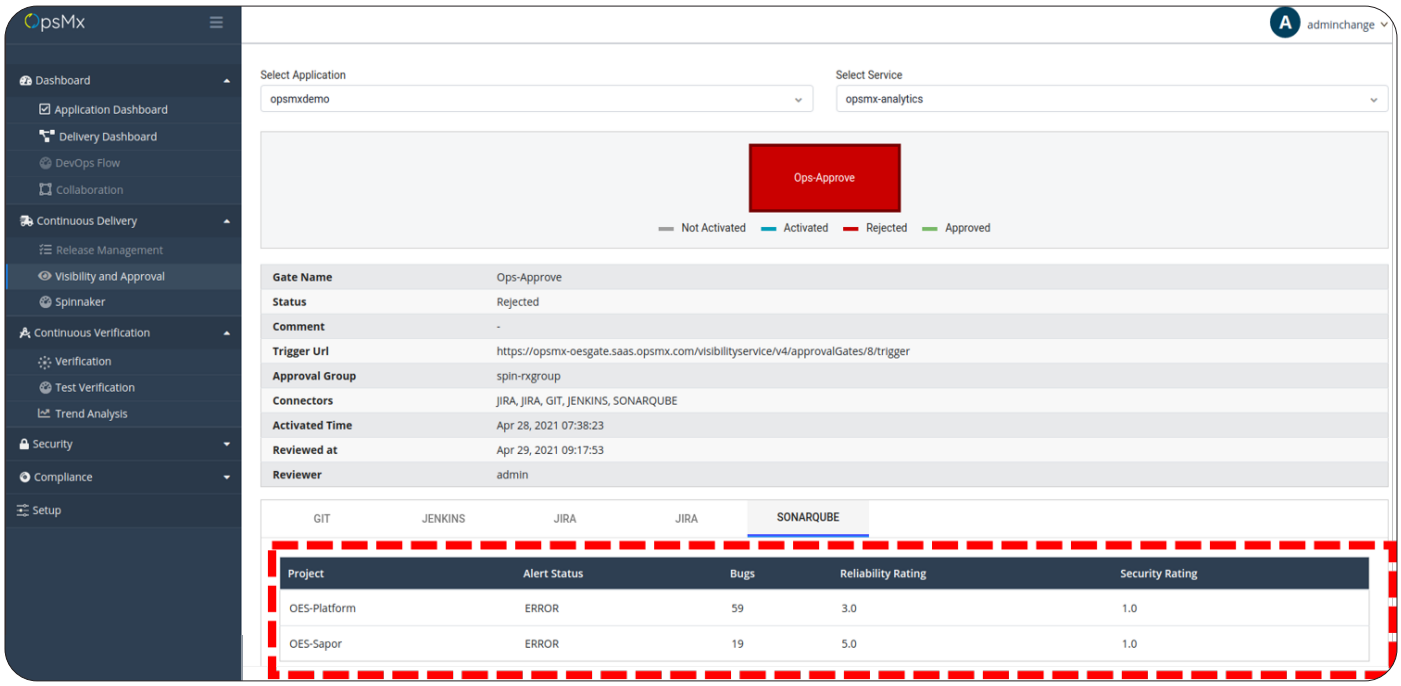


Figure 3B: Making security reports a part of the manual approval process in OpsMx Enterprise Spinnaker

OES can also be orchestrated to take automated decisions to (dis)approve a delivery pipeline from execution if it detects errors from security tools.

Identify the Risk of Releases Early

The risk of a software release is mitigated when it is detected early in the delivery cycle, starting from the build process. However, risk assessment of releases requires a manual review of a large number of logs and metrics produced in the build, test, deploy, and production stages. This can be a manually intensive process and extremely time-consuming.

Examples of identifying risks include:

- Identifying an architectural regression by analyzing changes in communication patterns in the logs such as a change in communication between services by use of HTTP protocol rather than HTTPS.
- Identifying patterns of AWS access keys in logs to highlight exposed passwords.

Deep analyses or quality regressions willl not be possible when changes are deployed to production every few minutes.

Leverage OpsMx ISD to Automatically Identify Risk

OpsMx Autopilot, the intelligence layer in OpsMx ISD, does risk assessment of releases in all stages of software delivery. Automatically scanning through thousands of build and test case logs, OpsMx ISD can point to the root cause of a failure in an instant.

Detailed log views in OpsMx ISD provide instant access to important log events, eliminating the need to examine massive log files. The system summarizes duplicate events and cluster-related events to enhance visibility and reduce clutter. The error events can be filtered through predefined or user-defined categories to reduce analysis time.

Similarly, the system can perform quality and performance checks during the deploy and production stages to determine the risk of release by applying Machine Learning techniques to the metrics data from the monitoring tools.

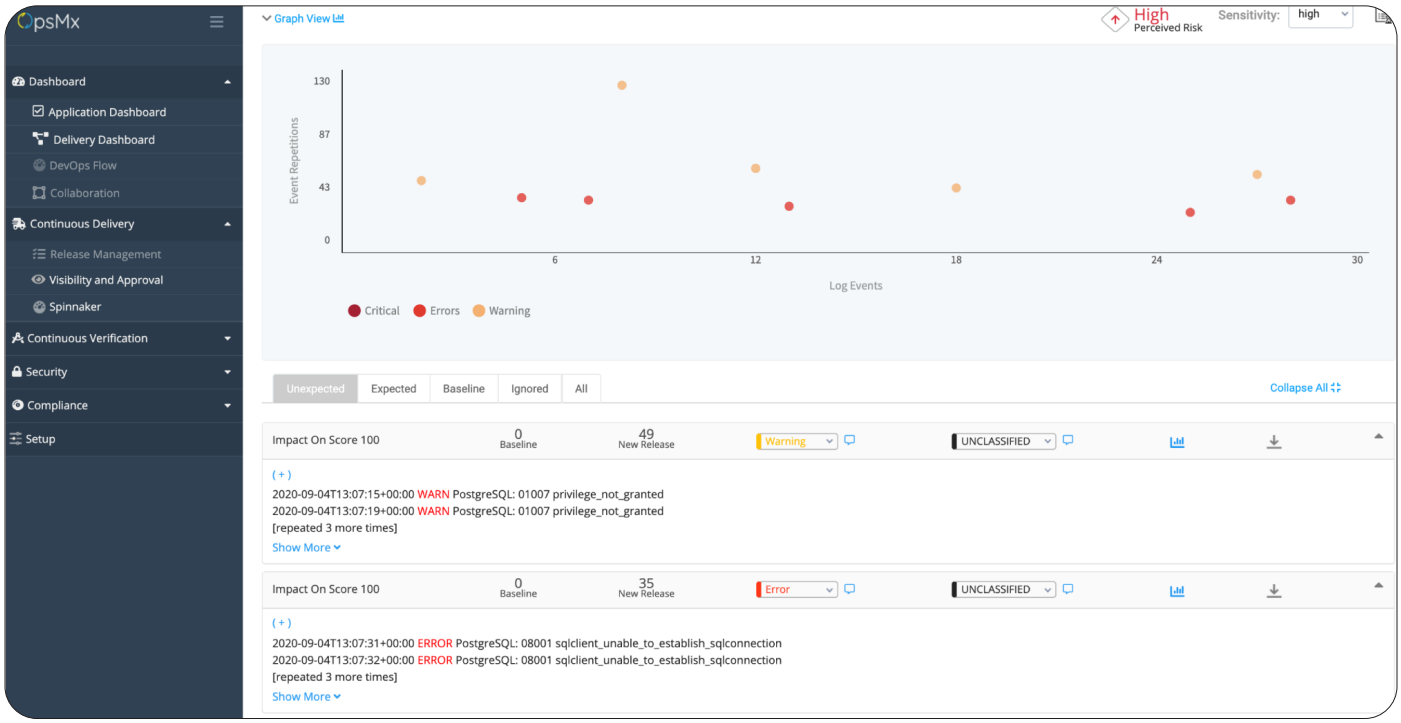


Figure 4A: Log analysis to find security vulnerabilities

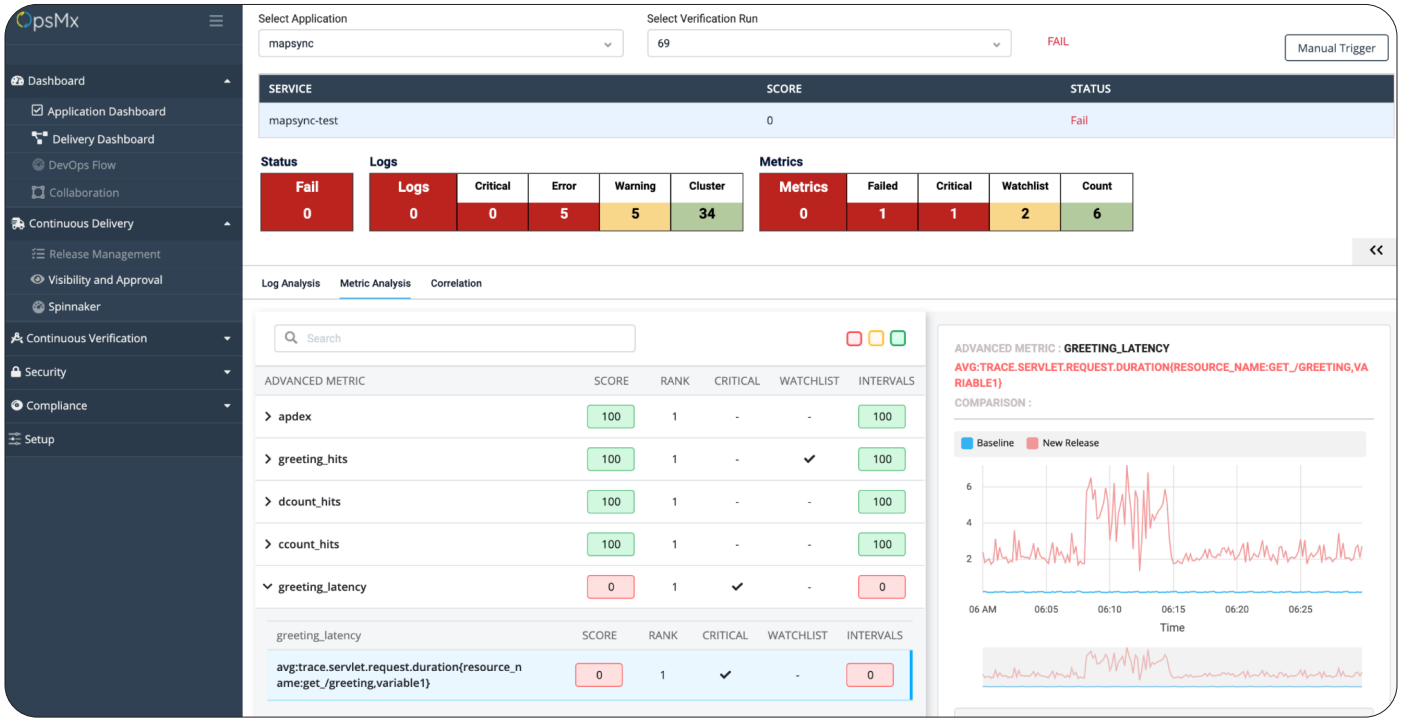


Figure 4B: Metric analysis to find performance regressions

With OpsMx ISD, you can avoid high-ceremony releases into production and even stop the CD pipeline from progressing in the early stages of software delivery, saving time and cost.

3 Enforce Policies in the Delivery Pipeline

Application security does not only mean securing the application code or monitoring its risks. It includes issues related to compliance and policies. There are new policies formed to comply with growing regulatory requirements for limiting breaches of application security. For example, a **policy to have no port other than 443 exposed or an application service can be deployed only from the approved registry, or an application cannot be deployed in certain regions**. Apart from numerosity, the policy definition changes from one region to another, such as GDPR rules.

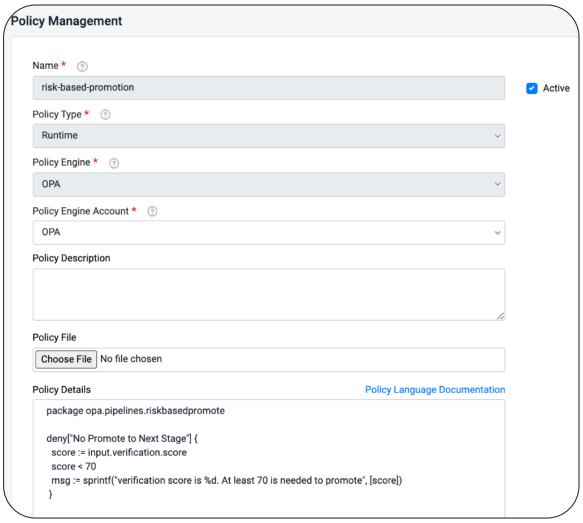
Security and compliance managers should define policies to improve compliance, security enforcement, and developer efficiency. They need a tool to protect the application and software delivery process at every stage. An example of a policy would be that an application cannot be considered for deployment if testing is not completed.

Leverage OpsMx ISD for Security and Compliance in DevOps

OpsMx ISD offers Continuous Governance to define security policies in your software delivery pipeline. As a result, you can ensure your DevOps process complies with organizational governance and rules while shipping your code, upgrades, and application to production.

ISD allows security and compliance managers to create policies and enforce them in each stage of the software lifecycle. Compliance managers can:

1. Create policies to check various software release parameters and deployment conditions before and during the execution of the delivery pipeline
2. Edit, delete, and modify policies
3. Audit and investigate policy violations, failed compliance checks, application names, etc.



Enable DevSecOps Using OpsMx ISD

Security might look like a deviation from the usual DevOps cycle. However, it is essential to make it a part of the DevOps pipeline to avoid service disruptions, security breaches, cyberattacks, and compliance non-adherence. In addition, continuous delivery is incomplete without security and risk mitigation strategies.

OpsMx offers an intelligent and highly secured platform for DevOps teams to incorporate security in the early stages of software delivery.

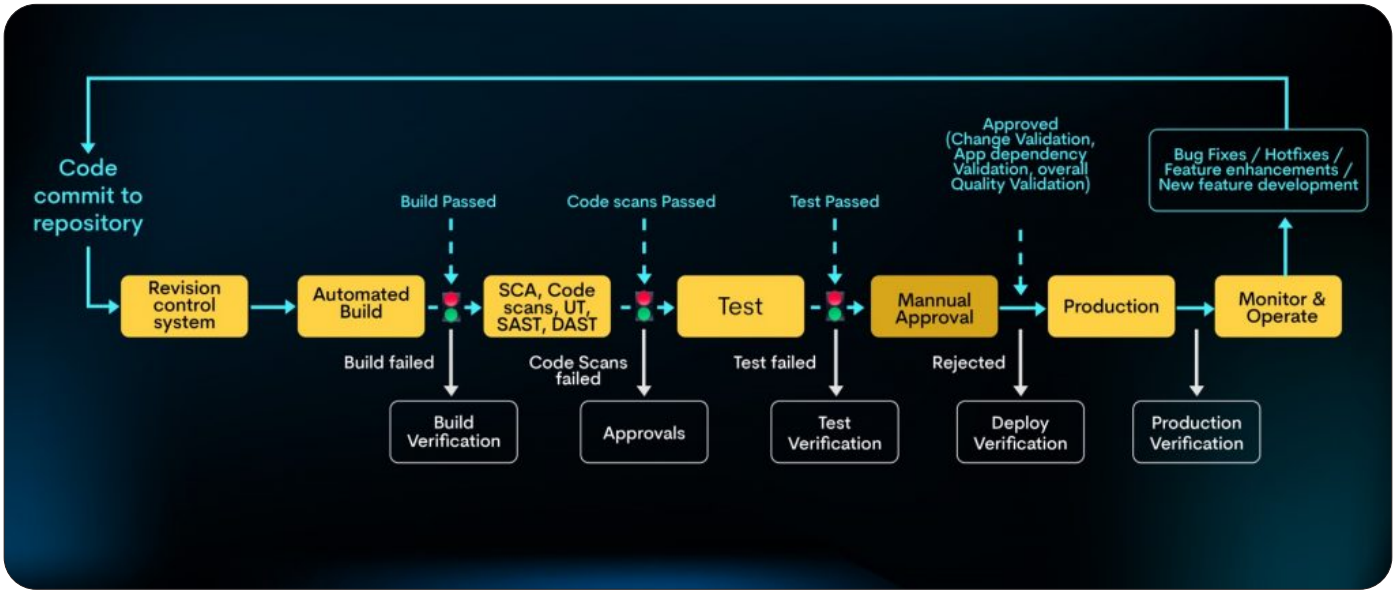


Figure 6A: Snapshot of security and compliance checks, and verification gates implemented at various stages in the software delivery process

Out-of-the-box security features in our CD platform:

1. Authentication and Authorization by following security protocols like LDAP, SAML, and RBAC
2. Secure communication between external services with Spinnaker and data encryption using mTLS (mutual Transport Level Security) and X509 (public-key) certificates-based authentication
3. Enterprise-level secrets management by storing sensitive information and secrets (e.g., Jenkins passwords, Kubernetes kubeconfig files, tokens, etc.) in Vault, Git, S3

For more information, please contact us.

OpsMx

About OpsMx

Founded with the vision of “delivering software without human intervention,” OpsMx enables customers to transform and automate their software delivery process. OpsMx’s intelligent software delivery platform is an AI/ML-powered software delivery and verification platform that enables enterprises to accelerate their software delivery, reduce risk, decrease cost, and minimize manual effort. Follow us on Twitter @Ops_Mx and learn more at www.opsmx.com