opentext™

GUIDE

Secure by design: Deliver software faster, more safely, and more securely with DevSecOps

Proactively fortify your software against threats

Contents

Introduction

What if your next software release made headlines—for all the wrong reasons? In the high-stakes world of software delivery, the race to keep up with business demands can feel relentless. But when speed overshadows security, the consequences can be catastrophic. DevSecOps changes the game by embedding security into every stage of your Software Delivery Lifecycle (SDLC), empowering your team to deliver fast, flawless, and secure software. Let your developers focus on innovations that thrill customers and outpace the competition—without compromising security.

What is DevSecOps

IT organizers are faced with increasing pressure to deliver secure software at the speed of business. Unfortunately, rushed software delivery increases the risks of security breaches. It doesn't help that Development, Security, and Operations all operate in silos. Disconnected processes and tools hinder collaboration, limit visibility, and increase the risk of errors while manual efforts are error prone, disconnected, and delay software delivery.

Security is a major concern for software delivery. It shouldn't be an afterthought in the SDLC. Gone are the days when security testing can be done at the end of the development cycle. We are seeing security shift left in DevOps, security testing is being conducted sooner in software and application development.

DevSecOps is a philosophy and practice that recognizes security is an integral part of the DevOps process. It aims to make security a shared responsibility throughout the entire lifecycle of software development, from planning and coding to deployment and maintenance.

Security is a team sport, integrating it into every stage lets you identify and address vulnerabilities early, ensuring rapid, secure deployments. With DevSecOps, organizations can deliver secure, high-quality software quickly and more efficiently, while reducing costs, managing risks, and fostering a culture of collaboration and continuous improvement.

Key aspects of DevSecOps:

- **Security integration:** Traditionally, security checks were conducted at the end of the software development lifecycle, often leading to delays if issues were found. DevSecOps shifts security "left," meaning security is integrated into every stage of the development process.
- **Automation:** DevSecOps relies heavily on automation to ensure security measures are consistently applied across all phases. This includes automated security testing, continuous monitoring, and automated compliance checks, enabling teams to detect and fix vulnerabilities faster.
- **Collaboration:** DevSecOps encourages close collaboration between development, security, and operations teams. By working together, these teams can ensure that security is not an afterthought but a fundamental part of the software development process.
- **Continuous monitoring and improvement:** Security in a DevSecOps environment is continuous. With ongoing monitoring of systems for vulnerabilities and threats, and continuously improving security practices as new risks emerge, you can be proactive and not reactive to security risks.
- **Cultural shift:** DevSecOps represents a cultural shift within organizations, promoting a mindset where everyone is responsible for security, not just the security team. This requires training and awareness for all team members to understand security best practices.

DevSecOps is the new standard, companies who adopt early will see a competitive edge against those who adopt late or miss the boat entirely.

> **"It is not the strongest of the species that survive, nor the most intelligent, but the one most responsive to change."**
>
> **Charles Darwin**

Build a DevSecOps culture

Building a DevSecOps culture requires commitment, collaboration, and continuous improvement. By embedding security into every stage of the development lifecycle, automating security processes, and fostering a security-first mindset, organizations can ensure that their software is delivered quickly and securely. This cultural shift ultimately leads to more resilient systems and a stronger overall security posture, allowing you to release your software faster.

To foster a DevSecOps culture, focus on the following areas:

Embed security from the start

- **Shift security left:** Integrate security early in the development process, during planning and design phases. This approach prevents security from becoming a bottleneck later in the cycle.
- **Threat modeling:** Conduct threat modeling sessions early to identify potential vulnerabilities and design secure architectures.

Promote collaboration across teams

- **Break down silos:** Encourage collaboration between development, security, and operations teams. Establish regular communication, joint planning sessions, and shared goals.
- **Cross-functional teams:** Create teams that include members from development, security, and operations to share the responsibility. This integration helps ensure that security considerations are part of every decision.

Postmortem reviews and feedback loops

- **Continuous integration/Continuous deployment (CI/CD):** Incorporate security checks into the CI/CD pipeline. Automate tasks like static code analysis, vulnerability scanning, and compliance checks to ensure security is continuous and consistent.
- **Infrastructure as code (IaC):** Automate the provisioning and management of infrastructure with security as a core consideration. Use IaC tools to enforce security policies and configurations.

By emphasizing security and building a DevSecOps culture, you will unlock benefits that can transform software delivery and drive your business forward. Key benefits include:

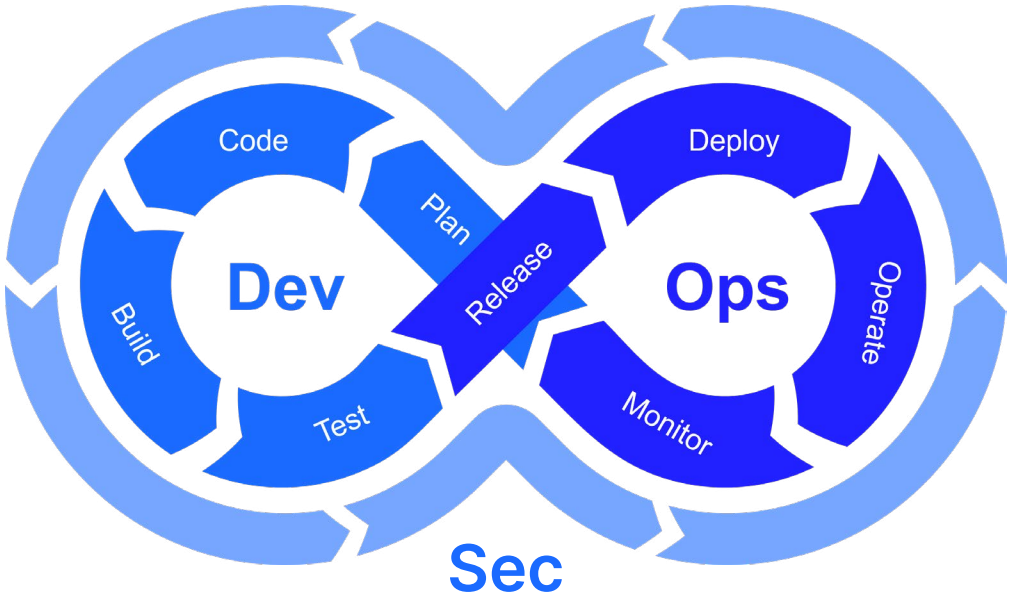
- **Faster time to market:** By integrating security early and continuously, issues are caught and resolved sooner, reducing delays and enabling faster deployment of secure applications without compromising quality.
- **Improved security posture:** Continuous monitoring and automated security checks improve the overall security of the software, reducing the risk of breaches and vulnerabilities.
- **Cost efficiency:** Addressing security issues earlier in the development process is often less costly than fixing them after deployment.
- **Compliance:** DevSecOps helps organizations meet regulatory requirements more effectively by embedding compliance checks into the development process.
- **Increased collaboration:** Streamlined CI/CD pipelines with integrated security testing provide greater visibility and insights so teams can quickly address problems, automate decisions, and better prevent vulnerabilities.

Integrate security into the development pipeline

This is where the rubber meets the road. Integrating security into your CI/CD pipeline is crucial to successful DevSecOps. Bringing security alongside the key stages of the pipeline - code, build, test, deploy, and monitor – ensures your software or applications will be secure by design.

- **Code:** Secure coding practices, code reviews, and static code analysis.
- **Build:** Automating security checks during the build process.
- **Test:** Integrating dynamic application security testing (DAST) and software composition analysis (SCA).
- **Deploy:** Secure configurations, container security, and IaC.
- **Monitor:** Continuous security monitoring, logging, and incident response.

Integration can look different for every company and their unique pipeline, but the core concepts apply to all:



Understand DevSecOps principles

Before diving into integration, it's important to grasp the core principles of DevSecOps:

- **Security as code:** Embed security practices into code and automation.
- **Shift left:** Address security issues early in the development lifecycle.
- **Automation:** Use automated tools to identify and resolve security vulnerabilities.
- **Continuous monitoring:** Keep security continuously integrated and monitored throughout the lifecycle.

Assess your current CI/CD pipeline

Evaluate your CI/CD pipeline to understand its architecture and identify points where security can be integrated. Key components to assess include:

- **Source code repository:** Where your code is stored and versioned.
- **Build system:** How your code is compiled or packaged.
- **Testing frameworks:** How your code is tested for functionality and quality.
- **Deployment:** How your code is deployed to different environments.

Integrate security into your development process

- **Security training:** Provide security training for your development team to ensure they understand secure coding practices and are aware of common vulnerabilities.
- **Static application security testing (SAST):** Implement SAST tools in the CI pipeline to analyze source code for vulnerabilities before it's built. Integrate SAST tools into your source code repository or build process.
- **SCA:** Use SCA tools to identify vulnerabilities in third-party libraries and dependencies. These tools should be integrated into the build process to scan for issues in your dependencies.

Enhance your CI/CD pipeline with security automation

- **Integrate security tools:**
  - **Static analysis tools:** For code analysis during the build process.
  - **Dynamic analysis tools:** For analyzing running applications to identify vulnerabilities.
  - **Dependency scanners:** For identifying vulnerabilities in third-party components.
- **Automate security testing:** Configure your CI/CD pipeline to automatically run security tests during every build. Ensure these tests include:
  - **Unit tests:** Test individual components for vulnerabilities.
  - **Integration tests:** Test interactions between components for security issues.
  - **End-to-end tests:** Test the entire application for security flaws.
- **Use IaC security:** If you're using IaC tools like Terraform or Ansible, integrate IaC security scanning tools to check for misconfigurations and vulnerabilities in your infrastructure code.

Implement continuous security monitoring

- **Security information and event management (SIEM):** Implement SIEM tools to monitor and analyze security events and logs in real time.
- **Vulnerability management:** Continuously scan your deployed environments for vulnerabilities and apply patches or updates as necessary.
- **Incident response:** Develop and integrate an incident response plan into your pipeline to address security issues promptly.

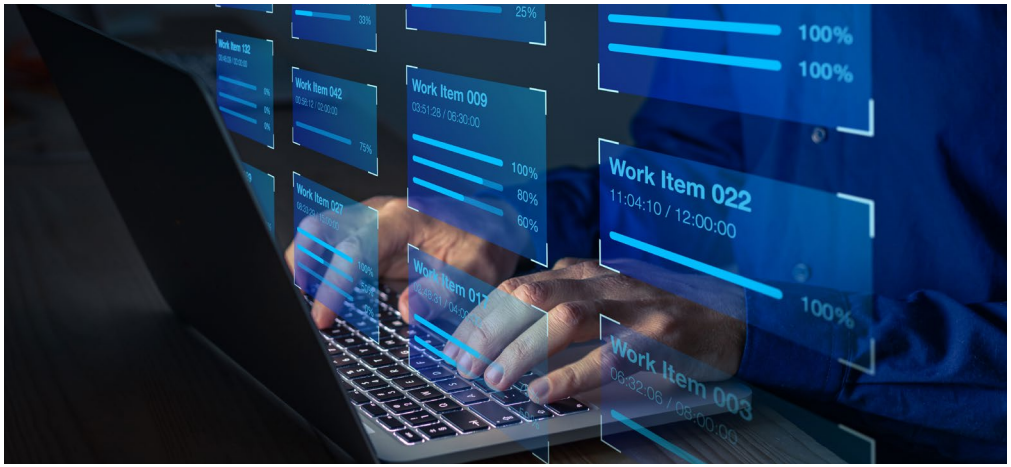
Review and improve

- **Regular audits:** Periodically review your DevSecOps practices and CI/CD pipeline to ensure they are effective and up to date.
- **Feedback loops:** Create feedback loops between development, security, and operations teams to address issues and improve security practices continually.
- **Metrics and reporting:** Track metrics related to security incidents, vulnerabilities found, and response times. Use these metrics to drive improvements in your pipeline and practices.

Foster a security culture

- **Collaboration:** Encourage collaboration between developers, security professionals, and operations teams to build a culture where security is a shared responsibility.
- **Continuous learning:** Promote continuous learning about new security threats and best practices among your team members.
- **Documentation:** Maintain thorough documentation of security practices, tools, and configurations for reference and training purposes.

Integrating security into your DevOps pipeline is crucial to DevSecOps, but so is using the right tools and technologies to get the job done. Selecting the right tools and consolidating your existing solutions will help streamline your end-to-end DevSecOps environment and set your teams up for success.



Tools and technologies for DevSecOps

We know what you're thinking, "Great, another tool to add to our DevOps toolchain." But that doesn't have to be the case. DevSecOps should streamline and consolidate your existing toolchain. Leveraging a unified platform for real-time communication, compliance and governance, shared visibility, and streamlined workflows empowers your teams to build secure software faster.

Categories of tools and technologies commonly used in DevSecOps:

Development and Deployment Support

- Version control systems (VCS)
- Continuous integration/Continuous deployment (CI/CD) tools

Application Security Testing

- Static application security testing (SAST)
- Dynamic application security testing (DAST)
- Software composition analysis (SCA)

Container security

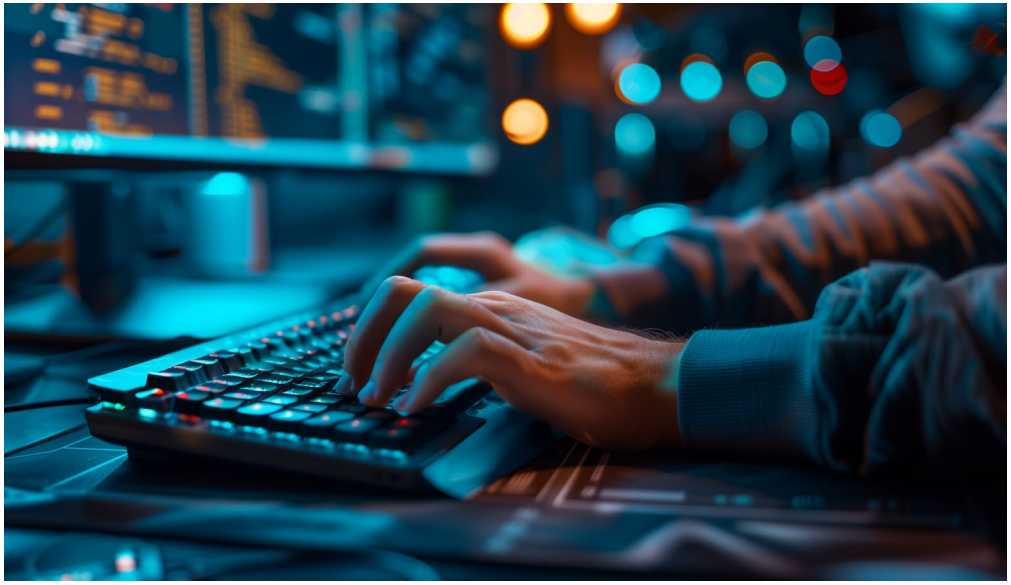
- Infrastructure as code (IaC) security
- Runtime security and monitoring
- Identity and access management (IAM)
- Secrets management

Governance and Compliance

- Compliance and policy management
- Incident response and forensics

Threat Mitigation and Response

- Security information and event management (SIEM)
- Threat modeling tools



Selecting the right DevSecOps platform or tool is crucial for effectively integrating security into your development and operations processes. Key criteria to consider when choosing your holistic DevSecOps solution include:

Integration with existing toolchain

- **Compatibility:** Ensure the tool integrates seamlessly with your existing CI/CD pipeline, version control systems, and other DevOps tools.
- **APIs and plugins:** Look for tools that offer robust APIs and a wide range of plugins to facilitate easy integration with your current workflows.

Automation capabilities

- **Automated testing:** The tool should support automated security testing (e.g., static code analysis, dynamic analysis) that can be triggered as part of the CI/CD pipeline.
- **Policy enforcement:** It should enable the automation of security policies and compliance checks, ensuring that security standards are consistently applied.

Scalability and performance

- **Handling scale:** The tool should be able to handle the scale of your projects, whether you're deploying to a small number of servers or thousands of containers.
- **Performance impact:** Evaluate the tool's performance impact on your development process to ensure it doesn't slow down your CI/CD pipeline.

Comprehensive security coverage

- **Wide security scope:** The tool should cover a broad range of security concerns, including code vulnerabilities, misconfigurations, container security, and IaC security.
- **Threat detection:** It should offer advanced threat detection and real-time monitoring capabilities to identify potential risks as they emerge.

Customizability

- **Flexible configuration:** The tool should allow for customization to fit your organization's specific security policies and development workflows.
- **Rule customization:** It should provide the ability to define and customize security rules and thresholds based on your unique requirements.

Real-time reporting and alerts

- **Insightful dashboards:** Look for tools that offer real-time reporting, dashboards, and alerts to provide visibility into security issues across the development lifecycle.
- **Actionable insights:** The tool should not only identify issues but also offer actionable recommendations for remediation.

Compliance support

- **Regulatory requirements:** The tool should help you maintain compliance with relevant industry standards and regulations (e.g., GDPR, HIPAA, PCI-DSS).
- **Audit trails:** Ensure the tool provides comprehensive audit logs and reports for compliance purposes.

Future-proofing

- **Regular updates:** The tool should receive regular updates to address new security threats and vulnerabilities.
- **Roadmap and innovation:** Consider the vendor's roadmap and commitment to innovation to ensure the tool will evolve alongside your needs.

OpenText solutions

Leading enterprises trust OpenText's AI-powered DevSecOps solution to release secure, high-quality software faster and more frequently, allowing them to respond to market changes quickly, stay ahead of the competition, and eliminate the stress and anxiety of software releases.

OpenText™ Core Software Delivery Platform is the industry's most comprehensive AI-powered platform for securing and accelerating software delivery. It seamlessly integrates security into every stage of your development lifecycle, from code creation to production, enabling continuous delivery without sacrificing quality or security. Seamless integration with OpenText™ Fortify™ security solution provides tools for static code analysis, detecting and fixing vulnerabilities before deployment.

End-to-end DevSecOps is more attainable than you think. With OpenText you can streamline your SDLC, integrate security at every stage, and maximize your ROI to achieve successful business outcomes.

Best practices and future trends

AI and automation

Humans are error prone. AI and automation remove human error from the equation and accelerate processes to help teams deliver software at the speed of business.

AI and machine learning have significantly enhanced security automation by providing faster threat detection, improving response times, and enabling proactive defense strategies. These technologies help reduce manual workloads for security teams, improve the accuracy of threat detection, and make organizations more resilient to evolving cyber threats.

Harness AI to:

- Transform data into actionable insights to propel smarter decision-making.
- Predict and prevent security threats.
- Automate compliance checks to ensure regulations are always met.
- Optimize workflows with data-driven precision.

With AI in the driver's seat, DevSecOps revolutionizes enterprise security, delivering faster, safer, and smarter software solutions. AI adds immense value and delayed adoption can mean the difference between beating your competitors and becoming the next Blockbuster Video.

Continuous improvement

Continuous improvement is another key DevSecOps principle—analyzing data and adjusting to improve the efficiency, quality, and security of your software. To analyze data effectively, we must know what measurements and metrics to track. Measuring the success of DevSecOps involves evaluating how effectively security is integrated into the development process and how well it aligns with broader business goals.

DORA (DevOps Research and Assessment) metrics provide a data-driven approach to measure and improve software delivery performance. They help organizations understand their DevOps maturity and focus on continuous improvement in terms of speed (deployment frequency and lead time) and stability (mean time to restore and change failure rate).

There are four DORA metrics to consider, along with security metrics that need to be monitored to gauge the success and effectiveness of DevSecOps:

- **Deployment frequency (DF):** How often an organization successfully deploys code to production.

High DF indicates that the team can release new features, updates, and fixes quickly and continuously. It reflects the team's agility in delivering value to customers. The goal is to increase DF by streamlining processes and reducing bottlenecks in the CI/CD pipeline.

  - **High performers:** Deploy multiple times per day.
  - **Low performers:** Deploy once a month or less.

- **Lead time for changes (LT):** The amount of time it takes for a commit to go from code to production.

Specifically, this tracks the time from when a change is made (commit) to when it is deployed to production. A shorter lead time for changes reflects how quickly a team can deliver new features or resolve issues. It indicates a smooth, efficient development and deployment process. The goal here is to minimize lead time by optimizing code review, testing, and deployment processes.

  - **High performers:** Have lead times that are less than a day (code to deploy).
  - **Low performers:** May have lead times of several weeks or months.

- **Mean time to restore (MTTR):** The average time it takes to recover from a failure in production (e.g., an outage, service disruption, or critical bug).

A lower MTTR indicates that teams can quickly diagnose and fix issues, reducing downtime and minimizing the impact on users. This reflects the team's ability to respond to incidents effectively and maintain system reliability. The goal is to reduce MTTR by improving monitoring, incident response, and troubleshooting processes.

  - **High performers:** Restore service in under an hour.
  - **Low performers:** May take days or even weeks to restore service after an incident.

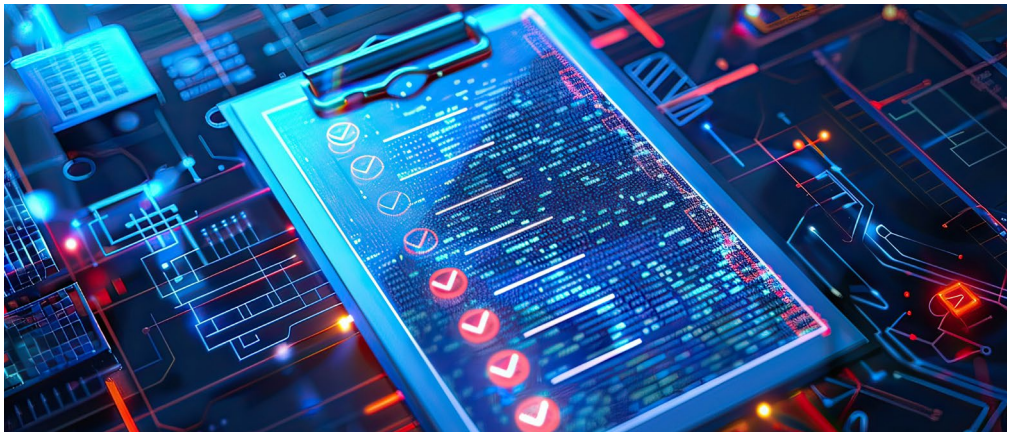
- **Change failure rate (CFR):** The percentage of changes or deployments that lead to a failure in production, such as service outages, degraded performance, or rollback.

A lower CFR suggests that the team is delivering high-quality code with fewer issues. It also indicates that testing and validation processes are effective in catching potential problems before deployment. Teams should work to minimize the CFR by improving code quality, testing, and deployment practices.

  - **High performers:** Have a CFR between 0-15 percent.
  - **Low performers:** May have failure rates above 40 percent.

- **Vulnerability detection and remediation metrics**
  - **Mean time to detection (MTTD):** Measures the average time it takes to identify security vulnerabilities after they are introduced. A lower MTTD indicates better proactive security measures.
  - **Mean time to remediation (MTTR):** Tracks the average time taken to fix identified vulnerabilities. A shorter MTTR reflects more efficient and responsive security processes.
  - **Reduction in vulnerabilities:** Monitor the number of vulnerabilities detected over time. A decreasing trend suggests that the DevSecOps practices are improving code quality and security.
- **Security testing coverage metrics**
  - **Test coverage:** Measure the percentage of code or applications that undergo security testing (e.g., SAST, DAST). Higher coverage indicates more comprehensive security practices.
  - **False positive rate:** Evaluate the accuracy of automated security tools by tracking the number of false positives. Lowering false positives improves developer productivity and confidence in security tools.
- **Integration and automation metrics**
  - **Automated security tests:** Assess the percentage of security tests that are automated within the CI/CD pipeline. Increased automation reduces human error and speeds up the detection of vulnerabilities.
  - **Rollback frequency:** Measure how often deployments are rolled back due to security issues. Fewer rollbacks indicate that security is effectively integrated into the development process.
- **Compliance and audit readiness metrics**
  - **Compliance audit results:** Review the results of compliance audits to determine how well the organization is adhering to regulatory requirements. Fewer non-compliance issues indicate stronger DevSecOps practices.
  - **Time to achieve compliance:** Track the time required to implement compliance-related changes. A reduction in this time suggests that compliance is being seamlessly integrated into the DevSecOps process.

Success in DevSecOps is multifaceted—encompassing technical, operational, and cultural aspects. Regularly measuring these metrics and adjusting strategies based on the results will help ensure that DevSecOps practices are effectively improving security, speeding up development, and delivering business value. Continuous evolution ensures you are staying ahead of security threats in a constantly evolving DevOps world.



Elevate your DevSecOps strategy

To stay competitive, rapid innovation is essential. But don't let speed compromise security. DevSecOps empowers you to deliver smarter software faster and more safely, ensuring you meet customer expectations without unnecessary risks.

By implementing DevSecOps, you can:

- Accelerate delivery: Release software faster and more frequently.
- Enhance security: Reduce vulnerabilities and protect your business.
- Optimize performance: Improve efficiency and collaboration across teams.
- Drive continuous improvement: Adapt to evolving threats and maintain a strong security posture.

Take the next step

Ready to unlock the full potential of DevSecOps? Learn more about how OpenText Core Software Delivery Platform can help you achieve your software delivery goals. Explore the platform and start your journey today.

opentext™