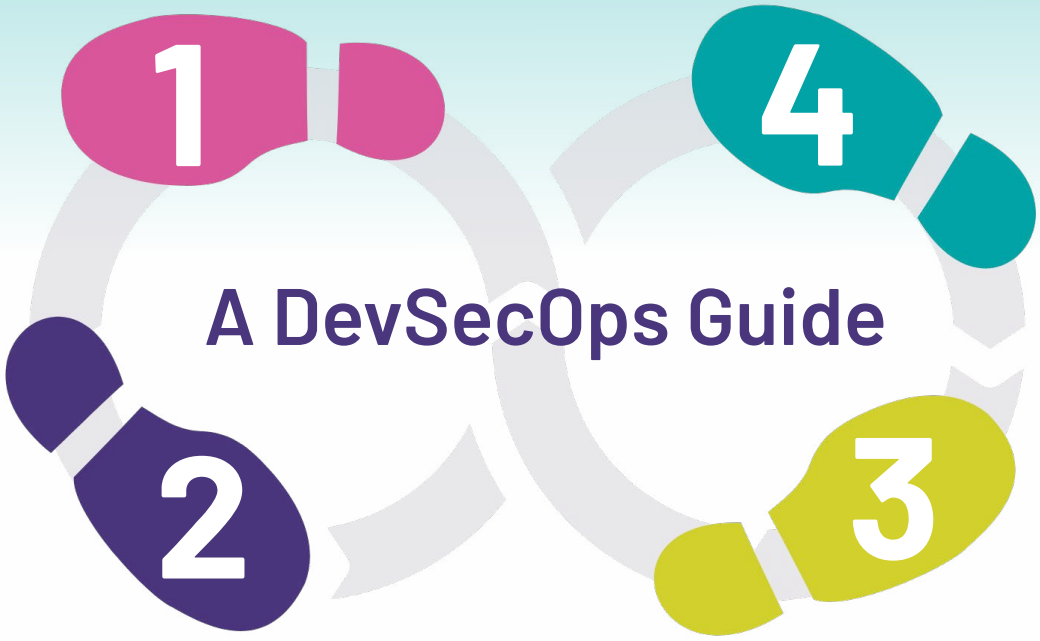
# 4 Step Guide on Improving DevSecOps:

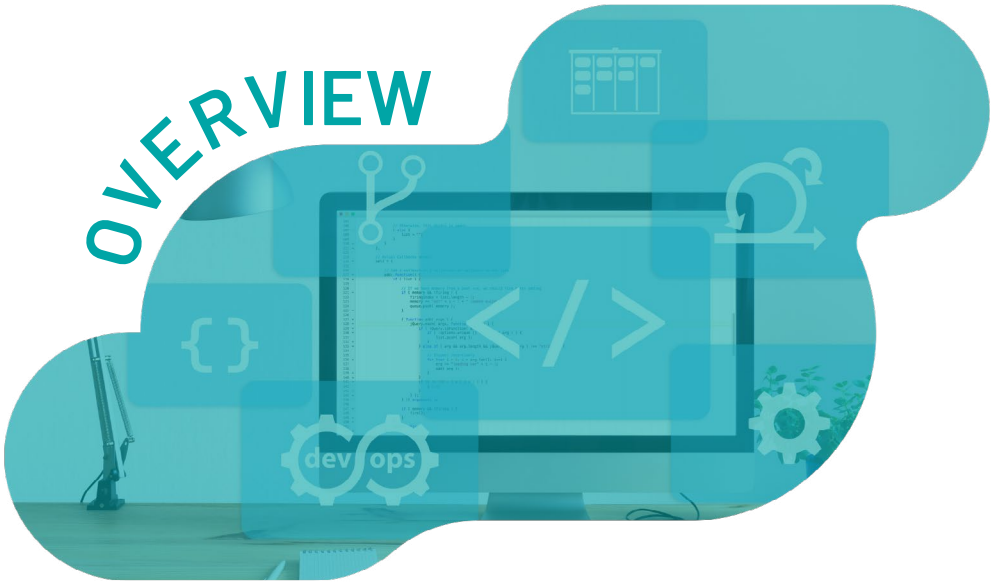



## A DevSecOps Guide

kiuwan
An Idera, Inc. Company

INDEX

# OVERVIEW




A reputation for secure code can increase your product sales, while vulnerabilities can create a public relations nightmare. However, integrating security into your DevOps team can be a challenge. All too often, security testing is performed after coding is complete — just before it goes out the door. By that time, the defects are throughout the code, forcing you to unravel the entire product to fix things.

DevSecOps (developer, security, and operations) bakes in the security from the beginning. New trends demand you do everything in your power to deliver secure code efficiently.

Some ***examples*** include:

- The increased adoption of cloud computing and infrastructure as code
- More attacks via third-party software to prevent
- Expanding serverless architecture
- The implementation of microservers over monolithic app development
- The move to a GitOps practice framework and evolution of Kubernetes

Development operations become more complex by the day, and with complexity comes code vulnerability. You need to add security as early as possible in the development process and keep it front and center throughout.

How is that possible? **DevSecOps** is the answer.

# Why DevSecOps is Essential

As the security landscape matures, different issues erupt within the DevOps community. According to Techwire, in 2019, 90% of software development projects will claim to be following DevSecOps practices by this year. However, the latest survey by GitLab shows that testing — primarily security testing just before release — was the primary reason for delays. This indicates that most development companies have yet to fully integrate security practices into their DevOps teams.

Make no mistake: There are challenges to implementing DevSecOps and continuing to practice it, but there are four steps Kiuwan recommends you take to improve your DevSecOps performance.

Here is Kiuwan's DevSecOps guide so you can successfully integrate security and catch problems before they arise — and solve them as quickly as possible when they do.

# *Challenges* to DevSecOps

The global DevSecOps market could reach over $11 billion by 2027.

Primary drivers of market growth include:

- Growing requirements for a quicker, safer software supply chain
- Higher internet penetration and the increase in cybercrimes
- Rising cases of data breaches and hacks
- The logarithmic increase in companies moving to the cloud, IoT deployment, and 5G

Meanwhile, the average cost of a single data breach could reach over $4 million worldwide. That's the worldwide average — the average for the U.S. alone is expected to be over $9 million. With the proliferation of ransomware attacks, cybercrime may have reached $6 trillion annually in the U.S. as of 2021.

As these challenges increase, so do the challenges for DevSecOps teams, which fall into three areas: **culture**, **skills**, and **compliance**.

# Culture

Many individuals and organizations share an extreme reluctance to change. Security is also often an afterthought, considered once the product is near release. However, you cannot insert quality into a finished product.

Finger-pointing is another issue. Security is often blamed for slowing innovation, development, and release, while developers are blamed for any security issues later in the development pipeline. Friction between silos can drag product release schedules to a standstill.

Misaligned incentives are just as bad. If you don't align the development, operations, and security goals, you have mismatched priorities, confusion about roles, and incentives that don't work.

## Culture Challenges in DevOps

The most common challenges for incentive-based work are:

- **Productivity vs. stability**: Which is more important — for the team to be productive or for the product to be stable?
- **Quality vs. speed:** Do you want it quality, or do you want it fast?
- **Operational efficiency vs. risk reduction:** Do you rush it out the door or stop to ensure everything is secure first?

Faster release schedules cause teams to take shortcuts and utilize shadow IT to get around processes. As information security teams adapt existing testing practices to the pace of development and operations, they worry about loss of control, increased risk, and low compliance.

According to a SANS survey, 75% of organizations deliver at least one software change per month — a 14% increase from five years ago. GitLab reports that 60% of developers release code twice as fast as before, up 25% from pre-pandemic levels. The result is that poorly tested code is released into production.

# Skills

The skill and labor shortage has been headlining for months. The Cybersecurity Ventures Report predicts 3.5 million cybersecurity job openings in 2025.

A shortage of application skills feeds into the overall skill issue. With the emphasis on DevSecOps, some companies labor under the myth that they need "super-developers" to bake security into the software. They do not — they just need competent

Also, many developers on DevOps teams feel they don't have the necessary security programming skills. However, organizations can embrace ongoing training and education for their existing DevOps teams and mandate collaboration with security. Then, everyone can program and test security throughout the development lifecycle.

In addition to skills and labor shortages, over half of survey respondents report insufficient monetary resources for secure development. Unfortunately, the feeling that you can buy DevSecOps is patently false. You can only purchase the tools for processes like Release Management and CI/CD tools. You cannot buy security itself.

## Compliance

As with security, compliance has often been a development afterthought. Almost every industry must show it complies with the correct regulations and standards. In standard DevOps, someone typically performed an audit in an integration/pre-production environment. Any issues were sent back for resolution, delaying the release.

Bake in compliance along with secure code. Incorporating compliance and security from the beginning of development all but assures the end product complies with standards — many of which concern privacy and security. There should be little need to rework for non-compliance issues when it's time for pre-production testing.

Organizations can overcome challenges and move toward more secure development and operations without incurring higher costs by following four steps to improve DevSecOps.

# Steps for *Improving* DevSecOps

An efficient DevSecOps process requires identifying the security criteria that prevent a version release. You want to catch issues as early as possible in the pipeline. You also need to validate each build regularly to avoid unwelcome surprises at release time.


Automation is the answer to efficient code validation and continuous integration in the pipeline. Using the Shift-Left philosophy, you catch vulnerabilities earlier in the development lifecycle where they take fewer resources to fix than if they were found at the end.

Here are four steps to help your team **shift left** and adopt a secure programming posture.

## **Step 1:** Use the same code analysis tools for the entire DevSecOps team


All too often, product development, security, and operations exist in separate silos. Each is considered an autonomous department with responsibility for a specific task. Each team selects its tools and procedures, leading to different toolsets and processes for each department. The silos communicate via reports rather than interactive discussions.

If you incorporate third-party code, your problems multiply.

Silos are not agile or unified. Communication loss leads to delays, miscommunication, and pressure to release vulnerable products. Instead, your teams should have the same goals and use the same tools to meet those goals.

### Unifying and Streamlining Code Analysis Tools

Embed your security tools into your DevSecOps workflows and governance processes. This will keep security awareness at the forefront of team members' minds. Instead of relying on a security test team to check for vulnerabilities near the end of the development lifecycle, everyone is responsible for building and checking security from the very beginning.

The development team can automate any necessary vulnerability checks such as DAST (dynamic application security testing), SAST (static application security testing), and SCA (software composition analysis).

Continue the automation beyond the pipeline and into change management, eliminating barriers to governing DevOps. Create an audit trail with visibility into:

- **Security checks**
- **Users**
- **Code**
- **Code quality**
- **Testing**
- **Configuration changes**

Artificial intelligence and machine learning tools allow DevSecOps to quickly identify and remediate issues before moving into production code. The Kiuwan **governance tool** gives your team everything it requires throughout the development lifecycle.

Using a standard set of security tools for the entire DevSecOps team increases efficiency and quality. Kiuwan provides a tool to help everyone receive the same information about vulnerabilities and defects without losing critical information. Teams can analyze the code and correct or remediate vulnerabilities as they go, confirming each defect has been addressed expeditiously.

### What does this achieve?

- Clear, centralized defect and vulnerability information: Everyone has access to the same results and can run the same tests.
- Increased agility and reduced resolution times
- Security teams integrated into DevSecOps with dedication to more comprehensive controls and innovations
- Automatic transfer of best practices to the entire development team

## Step 2: Clearly define and automate monitoring of acceptance criteria for launching a release


Integrate and automate security testing into the coding phase as part of the development lifecycle. To ensure everyone is on the same page, clearly identify acceptance criteria determined through collaboration among all stakeholders.

Acceptance criteria include mandatory compliance and optional fulfillment criteria.

- **Mandatory compliance criteria:** Must be followed throughout a development project to be eligible for production release. Failing to comply with any mandatory compliance criteria is an acceptance testing failure.
- **Optional fulfillment criteria:** Include desirable criteria for a specific release. Failing any single criteria does not cause acceptance testing failure. Your team can determine whether to expend resources on reworking the code to pass optional fulfillment criteria, even if multiple criteria are involved.

Understand that DevSecOps does not replace Agile — it complements it. Use Agile to deliver code in small, frequent releases. Adapt DevSecOps processes and tools to facilitate agile adoption and use. More companies are adopting Infrastructure as Code (IaC) to streamline management, monitoring, and provision.

According to Gartner, 60% of organizations will use infrastructure automation tools in the DevOps toolchain, improving application deployment efficiency by 25%. As development continues to build complexity, you need all the help you can get to keep up.

### What does this achieve?

- Clear, up-to-date acceptance criteria and understanding of security acceptance criteria
- Automatic criteria evaluation within the lifecycle without expending additional resources
- An objective view of risks present for each change as measured against requirements

## Step 3: Automatically record the delivery status of every Release Candidate


Communicating the delivery status of every Release Candidate is crucial, especially if you outsource any part of your project. Automatically record the delivery status at each juncture in the process for every release candidate. Everyone knows the condition of each build as it's received from the development team or service provider. You can quickly determine if the build is ready to move forward in the pipeline.

You don't need to halt the delivery of a new build automatically. Instead, a human intercedes to determine whether you should stop delivery. This individual is informed by technical information and a management conclusion based on pre-established acceptance criteria for business risk.

The human supervisor determines whether to continue the release process or stop it based on the business risks of non-compliance and the cost of satisfying optional fulfillment criteria.

Each release candidate must have a registered status. Automation streamlines the registration process. Registering the status for each candidate ensures smooth transitions between development and test without losing a release candidate or inadvertently allowing vulnerabilities to enter the production environment.

### What does this achieve?

- Assurance that each release candidate has met objective acceptance criteria
- Decisions based on objective information about progression in the lifecycle
- Identification of: The vulnerabilities that could have been avoided earlier and whether all team members adhere to security coding requirements

# **Step 4:** Classify and prioritize defect and vulnerability repair according to risk and repair effort

Not every defect and vulnerability should be resolved or remediated — at least not immediately. Before diving into resolution efforts, classify and prioritize each identified defect or vulnerability to determine the associated risk and the amount of effort required to resolve.

**Consider:**

- How critical are the vulnerabilities?
- What would the consequences of non-remediation be?
- How many resources would be required to resolve the issue?

Then, take the **following into account:**

- *What are* the business risks or consequences of this vulnerability?
- *What does each risk* include (denial of service, identity theft, etc.)? Map how any technical risk translates into business risk.
- *How much effort* is required for repair? Estimate the repair effort to evaluate the cost and benefit of remediation.

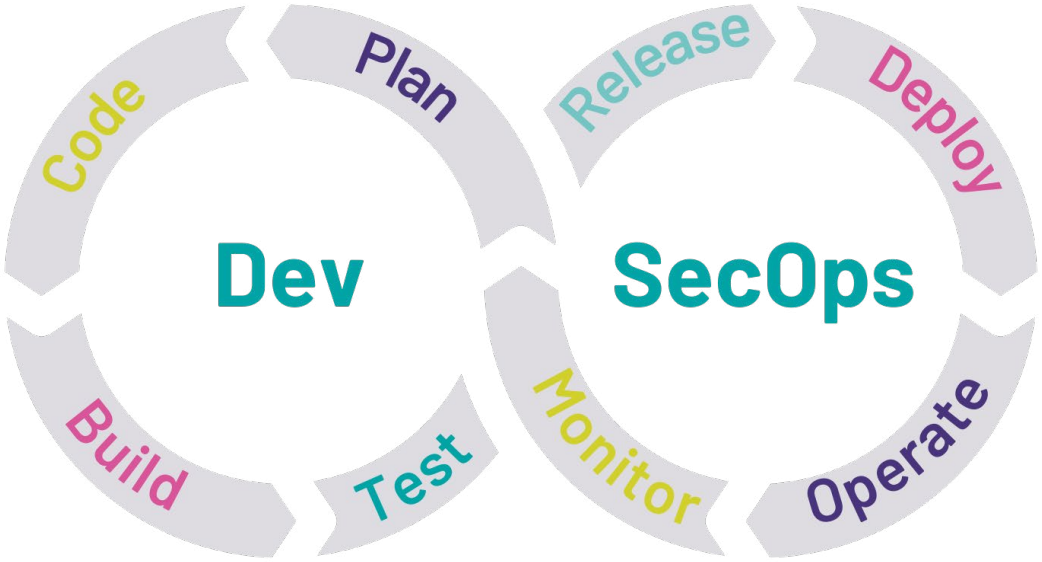
Once you classify your defects, you clearly understand how much risk each poses to your business interests. Prioritize remediation accordingly, expending resources for the higher-risk defects first.

Establish an action plan for addressing the various vulnerabilities and use continuous improvement processes to create efficient, cost-effective resolutions.

## What does this achieve?

- Clarification of the risks and consequences each defect poses for your business
- Increased visibility of the presence of defects and vulnerabilities through the classification, estimation, and management process
- An action plan for addressing the most critical defects first and progressively repairing vulnerabilities from major to minor
- Improvement within the continuous changes to your product to maintain an efficient lifecycle and evolution

# *KPIs to Monitor* Your DevSecOps Results




How do you know if something is working if you don't measure it? When you pursue the four steps above, you need to determine metrics that show your progress. Track and trend key performance indicators (KPIs) over time so you can spot problems early and redirect your resources as needed.

The following KPIs provide essential guidance for project management and monitoring.

## Number of releases deployed with OK status

Establish a clear set of security criteria and automatically measure how many release candidates meet them over time. This process provides a ratio of pass/fail to your pipeline, which can be used for budgeting, root cause analysis, and corrective action.

When analyzing this KPI:

- Define a fixed period of measurement. What time period provides the most actionable data? Measure monthly, quarterly, semiannually, or annually according to your needs and the risk levels identified by the data.
- Measure the percentage of releases in compliance for the entire period and compare it against previous periods.

Measuring specific durations eliminates any confounding variables such as the differences in days per month or time of year. Calculating compliance percentages allows for more accurate comparisons when the number of releases per period varies.

You can adapt this KPI to manage development per unit, including the number of versions you receive from each provider, supplier, and business unit. If your product is used in different technologies, you can track the version per technology type if you desire.

### What does this achieve?

- Determination of the number of releases in each KPI and whether the number is sufficient for measurement purposes
- Finding the KPI period that results in a value of 100 or better

## Number of release candidates delivered for each release deployed

How many release candidates does your team develop to produce a release suitable for deployment? Is there a pattern of defects or vulnerabilities that cause a candidate to fail?

Again, measuring over time allows you to spot trends and compare periods. Use the KPI to measure the efficiency of your process and provide a starting place for continuous improvement.

### What does this achieve?

- Calculation of the number of versions delivered for every one deployed
- Reduction in KPI value over time

The more versions you produce for every deployment, the less efficient your DevSecOps team is.

### Ask yourself:

- Why are so many versions failing the established criteria?
- Why do you need to send the product back into development?

The goal for this KPI is to move the value towards zero. Drive improvements to the process to reduce the effort required for multiple versions per production release, saving time and money.

## Average time between release candidates

Evaluate your time to market by calculating the amount of time between release candidates. How long is it taking your team to produce each release candidate? Along with measuring the number of candidates produced per one deployed, you can determine your team's overall productivity.

Until your product hits the market, your business loses money on it. The longer the production cycle and the more versions you produce before passing a deployable release, the more time and effort you expend before you begin to achieve a return on your investment.

Measure this KPI using a fixed period and do not analyze it without considering the number of release candidates required per release deployment. You do not want to minimize the time between release candidates by increasing the number of candidates. Determine where production is lagging or creating defects before attempting to speed things up.

### What does this achieve?

- KPI value minimization and reduction in the number of release candidates per deployed release
- KPI value reduction strategies if you have the same number of release candidates over several measurements

# Why Is DevSecOps *Essential*?

Insufficient security testing and processes have increased the frequency of high-profile data breaches. Cloud adoption has broadened the potential attack surface for enterprise IT and requires you to be more proactive than ever before.

Your team must address **four key areas**:

1. **Configuration management**
2. **Configuration validation**
3. **Vulnerability code scanning**
4. **Penetration testing/threat modeling**

Not only does DevSecOps identify security threats early in the development pipeline, but it also makes everyone responsible for security. It encourages shared standard processes and execution to strengthen security across the board.

# *Implement DevSecOps* to Maximize Security and Minimize Problems

*DevSecOps reduces the chance* that a defect or vulnerability makes it into production code without creating a bottleneck in the development process. It enables you to continue releasing code quickly and manage production.

Standards and regulatory compliance are simplified when *DevSecOps is used across your organization*. Just a few of the many compliance standards industry must meet to retain the trust of the public and clients include:

- General Data Protection Regulations (**GDPR**)
- Health Insurance Portability and Accountability Act (**HIPAA**)
- Federal Information Security Modernization Act (**FISMA**)

With DevSecOps, you have a natural and easy way to promote cross-enterprise collaboration and address any security issue during the development and operations lifecycle. It improves your ability to scale and gives you more flexibility than Agile alone or depending on security testing at the back end of the release cycle.

Kiuwan has the tools to help you integrate security into your development and operations teams. Contact us today to learn more about turning your DevOps into a secure, collaborative process and building efficiency into your release cycle — without introducing vulnerabilities or slowing delivery.

***YOU KNOW** CODE, **WE KNOW** SECURITY!*

## GET IN TOUCH:


**Headquarters**

2950 N Loop Freeway W, Ste 700

Houston, TX 77092, USA

United States **+1 732 895 9870**

Asia-Pacific, Europe, Middle East and Africa **+44 1628 684407**

**contact@kiuwan.com**

Partnerships: **partners@kiuwan.com**