
# SCHOOL OF COMPUTING

## DEPARTMENT OF SOFTWARE ENGINEERING

**COURSES TITLE:- SOFTWARE ENGINNERING TOOL AND PRACTICE**

**COURSE CODE:-SEng3051**

**INDIVDUAL ASSIGMENT**


| NAME | ID |
|---|---|
| DAWIT ENDASHAW ..... | WDU145838 |


**SUBMITED TO : MR ESMAEL M.**

**SUBMITTED DATE: 21/9/2016 E.C**

# Contents

# Introduction

DevSecOps is a compound of Development, Security, and Operations, is a methodology that integrates security practices into the DevOps process. This synergistic approach aims to create a culture of shared responsibility for security, where teams collaborate seamlessly to deliver secure and reliable software at a faster pace. In this document, we delve into the meaning and significance of DevSecOps, exploring how it transforms traditional approaches to software development by emphasizing security from the outset. We will discuss the key benefits of implementing DevSecOps, such as enhanced security posture, faster time- tomarket, and improved collaboration among teams. Furthermore, we will examine the lifecycle of DevSecOps, outlining the stages involved from planning and development to testing, deployment, and monitoring. Understanding the lifecycle is crucial for organizations looking to adopt DevSecOps practices effectively. Lastly, we will explore the tools and technologies that support DevSecOps implementation. These tools play a pivotal role in automating security processes, enabling continuous integration and deployment, and facilitating the seamless integration of security into every phase of the software development lifecycle. Join us on this journey as we unravel the essence, advantages, lifecycle, and tools of DevSecOps, and discover how it can revolutionize your approach to software development

# 1: DevSecOps Challenges

While DevSecOps can benefit your organization in many ways, we've observed several challenges to its adoption, the most common of which are the following:

1. lack of security assurance at the business and project levels

2. organizational barriers related to collaboration, tooling, and culture

3. impact to quality because security is not a priority while systems are getting more complex

4. lack of security skills for developers, business stakeholders, and auditors

5. insufficient security guidance due to lack of resources, standards, and data

The rest of this section examines each of these challenges and provides approaches for overcoming them.

## 1.1: Lack of Security Assurance

How do we know that the security practices we've adopted for our development lifecycle and built into our software are adequate and appropriate for the business purpose we're trying to address? Addressing this challenge can be hard, especially when your industry, business, and/or project lacks security assurance.

- Your Industry Lacks Security Assurance Models

The security requirements for your industry or domain are different from those of other industries or domains. For instance, the health care industry has different security requirements from the financial sector. If your industry lacks assurance models, you might begin by assessing your own organization's security posture and requirements, then engage with similar organizations in your industry or domain to share information. In addition, we recommend the following:

- Don't wait for an industry standard to emerge.
- Join or create informal working groups with industry peers.
- Attend conferences and network with like organizations.
- Share your experiences and lessons learned.
- Work with others to extend the body of knowledge and establish best practices.

### ➢ Your Business Lacks Security Assurance

There is often a disconnect between business needs, mission, and vision when it comes to security. Developers need to understand the business context of security. They must think about the organization's security policies, its business drivers, and how these apply to the software being developed. In so doing, they should address security as early as possible in the lifecycle, preferably during the requirements stage. As you do, keep the following recommendations in mind:

- Focus on fundamentals: What are the threats? What are the business drivers? Balance the two.
- Align with development with business needs (time to market, cost savings, resilience).
- Conduct external audits.
- Understand the business context.
- Identify, link, and rank business and technical risks.
- Identify security requirements in early.
- Define the risk mitigation strategy.
- Educate top management and get them onboard.
- Engage more senior technical people first to work with security teams.
- Make security part of senior technical reviews; organically spread the word.

### ➢ Your Project Lacks Assurance of Security

Once you've identified security assurance needs within your industry or domain (and perhaps your specific business within that domain), you need to map that data to your project. For instance, perhaps your organization is already following guidance, such as General Data Protection Regulation (GDPR) or the Health Insurance Portability and Accountability Act (HIPAA). You need to account for any security activities stipulated in that guidance in your project planning, and you need to do so early in the lifecycle when there's still time to address it. As you do, keep in mind the following recommendations:

- Map reporting data from tools to form a continuous view of value.
- Run security tools on all code to measure code quality and standards.
- Review code changes for security and document approval prior to release.
- Use dedicated testing resources in the case of significant changes.
- Track all changes and approvals for incident purposes.
- Conduct code reviews.
- Expose security team to your metrics and data.

## 1.2: Organizational Barriers

If you're not sharing your DevSecOps journey across the organization, from concept to product, you should expect problems since you won't have a clear understanding of the business needs your software needs to address. You might not even have a clear vision of the customer's needs and the environment in which the customer operates. Communication is key to breaking down organizational barriers, but often different units inside an organization use different communications tools, structures, and goals.

To begin to break down these barriers, briefly document your journey from idea to product. Look at points of interaction among the various components of your organization. Educate executives who likely don't know the details of the DevSecOps process. Build connections and a culture that reinforce the sharing of the same goals and vision. Often, poor stakeholder collaboration, difficulty integrating pipeline security, and an unwillingness to make security a priority stand in the way of successful DevSecOps implementation.

- Poor Stakeholder Collaboration

The product you're creating touches many other stakeholders in your organization, including marketing, IT, and legal teams, but communication among them might be lacking. For example, you may have different tools, may not share the same infrastructures, and may not even share the same vision and goals. To address these issues, you need to come together and document your journey from concept to product. A simple cheat sheet will suffice, one that reminds all stakeholders of the vision, the mission, and the roles they will play in the lifecycle. Our recommendations for improving stakeholder collaboration include the following:

- Document your current state and identify silos (e.g., development, infrastructure, and security).
- Start building collaboration between the Security, Dev, and Ops teams.
- Be prepared: people generally don't want to change their culture and/or workflow.
- Make sure everyone gets on the same page regarding the importance of security (from executives to DevSecOps teams).
- Instill a continuous security mindset.
- Focus on partnership, not unhealthy conflict. Destroy the blame culture.
- Get stakeholders to agree on a shared a vision for the project.
- Balance workload among teams involved.
- Put security people into development teams.

- Integrating Pipeline Security

The pipeline is not only the infrastructure supporting DevSecOps. Instead, it is the heartbeat of your entire DevSecOps ecosystem, including source control, communications, issue tracking systems, documentation systems, CI/CD, and code review. This infrastructure should be connected, i.e., all the tools should communicate to each other.

For instance, your source control should be able to communicate with your build server, your communication systems, and your issue tracking systems. When your infrastructure is connected this way, you can apply threat modeling, static analysis, dynamic analysis, project management, or interactive application security analysis. Think about tool integrations to overcome pipeline security problems and then design your infrastructure to address security.

The pipeline is where transparency happens and where all the stakeholders enforce their expertise via automation. One way to achieve this transparency is through metrics dashboards fed by pipeline data that are easy to read. The tool should be tailored to the product. The bigger you are, the harder this is to do, but the result is worth it. Recommendations for integrating pipeline security include the following:

- Integrate your process with threat modeling (TM), static application security testing (SAST), dynamic application security testing (DAST), and interactive application security testing (IAST).
- Install security requirements traceability.
- Apply metrics: mean time to repair (MTTR), mean time to detect (MTTD), vulnerability escape rate, repeated incident root cause, time to deploy the app from development to production.
- Look at different approaches: abuse cases, architectural risk analysis, application penetration testing.
- Design for security.
- fail securely and fail safe defaults
- least privilege
- defense in depth
- Automate where possible.
- infrastructure as code (IaC), virtualization, containers, and load balancing
- configuration management
- continuous application and performance monitoring

- Making Security a Priority

You need to plan for security if you want to make it a priority. Treat security as a feature that your software must have. In some cases, the choice is out of your hands: a software bill of materials (SBOM), for instance, might mandate building security into the product. But how do you make security a priority? We recommend the following:

- Use evangelists to drive culture change.
- Explain why security is an important, shared responsibility, and its impact.
- Embed security into operations escalation.
- Invite the security team to postmortems.
- Create a plan in small parts; start with a pilot and be mindful of cross-team resource constraints.
- Keep it simple; don't overwhelm the system. If there are too many things to do, the plan is likely to fail.
- Incrementally chase real risk and threats first.
- Test whether your organization is ready for the culture change; no single technology/tool will get you DevSecOps.

## 1.3: Lack of Quality

Security is integral to quality. In our observation, lack of quality is often associated with the security team getting involved too late, a lack of confidence in the release, and system complexity.

- Security Team Involved Too Late

Too often, developers make security a secondary priority, especially when they are under pressure to keep production moving forward. As we stated earlier, security is a key aspect of quality. When the security team engages towards the end of the SDLC process, it's often too late to avoid the disruption and expensive rework that flows from the security flaws it identifies. Depending on the project cadence, "too late" may mean two months, two weeks, or even two days.

Consider a team using Agile development with two-week sprints. Within that sprint, given a scrum every day, the developer would need to know of any problems as early as possible. However, the Sec team only analyzes the code a month later (or maybe two months later or just before deployment to the production environment). Any problems discovered by the Sec team at this point will require tremendous work, and developers will push back. Moreover, the later problems are discovered in the SDLC, the more expensive they are to fix. To avoid these issues, we recommend the following:

- Start getting security and compliance requirements in early.
- Tie compliance objectives into providing assurance back to the business.
- Test compliance against security policies to identify gaps.
- Define a risk mitigation strategy early.

➢ Lack of Confidence in the Release

Correcting problems and patching security vulnerabilities late in the development lifecycle when the pressure is on to get the product out the door opens room for doubt about the quality of your release. This lack of confidence hinders planning and effective use of resources as you need to reserve resources to address flaws and vulnerabilities discovered after release. These flaws and vulnerabilities represent an opportunity cost, a dollar cost, and a reputational cost. But there are ways to improve confidence in your release, including the following:

- Instill risk-based security testing.
- Move the conversation from CABs and phase gates to compliance driven releases.
- Automate reporting for compliance violations and stop the pipeline when the threshold is exceeded, or policy not met.
- Move toward frequent, automated audits.
- Audit yourself to demonstrate compliance with policies or regulations.
- Establish security requirements traceability (a feature DevOps provides) and trace everything: code, artifacts, pipeline, test cases, etc.

➢ System Complexity

Consider a complex system with multiple application programming interfaces (APIs) and microservices. How do you gauge its quality? How do you know that each of the services is following the right security controls? How do you know that each API is following centralized communications? How do you know that they are following the security policies that you enforce in organizations? You need to incorporate these policies in your code, in your architectures, in your microservices. To do so, you need to collect the right data and metrics that enable you to examine all the components throughout your complex system. The more complex your system, the more you need a testing environment that mirrors your production environment. In short, we suggest the following:

- Identify proxy metrics for complexity, such as the number of issues in production and the time to deploy an application.
- Drive security policies into production by integrating security tasks in early stages of the DevSecOps pipeline.

## 1.4: Lack of Security Skills

Developers, architects, scrum masters, and other key players in an organization should have the right vocabularies and skills. By vocabularies, we mean some common knowledge or skillset, or a common understanding, such as a knowledge of how to write secure code. In our experience, this lack of a common vocabulary often manifests in three ways: The business lacks security skills, developers lack security skills, and/or auditors lack security skills.

### 1.4.1 The Business Lacks Security Skills

Business stakeholders need to use the vocabulary of security. Of course, not everyone can be an expert at everything, but the business stakeholders should be able to understand and articulate security requirements and connect those security requirements to organizational risks. An acquisition team, for instance, should be able to know it's acquiring the right security practices when it's acquiring a product. To improve in this area, we recommend the following:

- Shift the conversation to risk and quality.
- Service and protect the business interests to lower risk (identify risk and/or security value).
- Identify architectural risk and uncertainty and map these risks to compliance, resiliency, and feature delivery.

### 1.4.2 Developers Lack Security Skills

We like to think that developers know everything needed to perform their tasks successfully. Developers certainly know how to write code, but they are often not trained for secure coding in specific languages at the university level or in skills development programs, where the focus remains on feature development. It takes time to learn these skills, and to practice using them, but it's worthwhile for the organization to develop these security skills among its staff, to develop. This up skilling can take the form of security training or other programs and resources that incentivize and motivate development staff to acquire and develop these skills.

You can start small with a narrow focus. For instance, ask yourself, "What are the top 10 vulnerabilities we have addressed in our organizations? What are the top 10 CVEs? What are the top 10 CWEs?" Focus on training that addresses those issues and widen your scope over time. Or start with the programming language(s) used by your organization and focus security training on those languages. Other recommendations for building security know-how among your development staff include the following:

- Keep the endgame in mind and build a collaborative security culture.
- Implement compliance automation to drive business thinking into the SDLC.
- Motivate and unblock the path to your goal: Remove task ambiguity, set clear role targets, and don't overload.
- Aim for long-term retention and apply learning in context repeatedly.
- Rotate security experts on the team, where possible.

### 1.4.3 Auditors Lack Security Skills

Auditors review code and products, rendering a thumbs-up or a thumbs-down based on established criteria, but they generally don't have the skills and knowledge to provide an accurate judgment about security. To compensate, foster strong relationships among auditors and developers. Educate auditors on your architecture and systems. As we noted earlier in the section on business stakeholders, make sure your auditors understand and use the same vocabularies. Other recommendations include the following:

- Build working relationships and collaboration across silos.
- Make security a part of informal discussions.
- Provide cross-functional training for both technical and compliance domains.
- Integrate low-disruption workflows.
- Get familiar with some common standards and frameworks (OWASP Top 10, NIST 800-53, and ISO 27001).

## 1.5: Insufficient Security Guidance

Organizations need to take stock of their compliance practices and security policies and implement them in their products or capabilities. For instance, you may have adopted zero trust policies, but how will you implement them? Consider where your organization is in its DevSecOps journey and map out your future: What are you doing for year one? Year two? Beyond? Remember, if you want to create a new standard for your organization, you might think you're unique, but you're not. Instead of starting from scratch, use an existing standard or try to tailor one to your needs. For instance, you might start with the OWASP Top 10. How can you implement those frameworks in guidance under CI practices? Incorporate the policies into the workflow from beginning to end.

In our experience, problems in this area stem from three deficiencies: lack of security resources, lack of security standards, and/or lack of proactive monitoring.

- Lack of Security Resources

You probably don't have enough security people on the team, yet there are policies and guidance to develop. Moreover, there's so much else that must happen. If you're lucky, you might have some unicorns on your team, some overachievers, but you need to get beyond that. Here are some recommendations for organizations that want to embark on a DevSecOps journey but lack security resources:

- Start small by introducing a policy and assess your gaps. Grow from there.
- Map policies to domain-specific procedures (e.g., development and testing) and implement in product (e.g., zero trust).
- Aim for long-term sustainability: When you evaluate an upgraded capability a couple of years after deployment, is the change still there?
- Spread security responsibility across multiple people.

- Lack of Security Standards

Here's the good news: As we've noted, several security standards have already been developed. You don't have to start from scratch. You can start with policies derived from existing standards, tailor them to your needs, and incorporate them into your practices and products. When doing so, keep these recommendations in mind:

- Don't go big bang. Start with something manageable, such as static analysis, and grow from there.
- Start with a well-known framework like OWASP Top 10 and create a few policies derived from that.
- Aim at low hanging fruit (CI or testing, for example) and measure security against your initial policies.
- Grow by looking upstream and downstream for the next-easiest implementation.
- Bake policies into the workflow to avoid regression.

- Lack of Proactive Monitoring

Proactive monitoring [DS1] identifies and addresses security risks before an attack happens. The key to developing more proactive monitoring is to look at cases in which you've been forced to react, then plan ways to get in front of the problem. For instance, you may have discovered certain vulnerabilities, or classes of vulnerabilities, after past releases. Think about how you can address those kinds of vulnerabilities in your pipeline and develop a practice around that and then incorporate it as early as you can in your SDLC.

There are great open-source and commercial monitoring tools available. While each of these tools has an individual dashboard, ideally pe the results from all of them should be integrated into a common dashboard. Doing so will provide an overarching view into your DevSecOps journey for both the organization and your individual products. We also recommend the following:

- Start with Ops log monitoring before attempting expensive tools.
- Create feedback loop from Ops back to development.
- Update security documentation, including trust boundaries, new threats, and component verification.

.

## 2: What is DevSecOps

DevSecOps is a set of practices and principles that aim to integrate security into the software development process, from the initial design phase through to deployment and ongoing operations. The term "DevSecOps" combines "Development" (Dev), "Security" (Sec), and "Operations" (Ops), highlighting the collaborative and integrated approach to software development, security, and operations.

Key aspects of DevSecOps include:Shift-Left Security:

- DevSecOps emphasizes addressing security concerns early in the development process, rather than as an afterthought. This involves integrating security practices into the planning, coding, testing, and deployment stages of the software development lifecycle.
- Automation of Security Processes: Automation is a central tenet of DevSecOps. It involves automating security testing, code analysis, vulnerability scanning, and compliance checks to identify and remediate security issues as early as possible.
- Collaboration and Communication: DevSecOps encourages close collaboration between development, security, and operations teams. This collaboration helps ensure that security considerations are integrated into every aspect of the software development process.
- Continuous Monitoring and Feedback: DevSecOps promotes continuous monitoring of applications and infrastructure for security vulnerabilities and threats. This enables rapid detection and response to security incidents.
- Shared Responsibility for Security: In DevSecOps, all team members share responsibility for security, rather than it being solely the domain of a dedicated security team. This shared responsibility helps to create a culture of security awareness and accountability.

# 3: DevSecOps lifecycle

DevSecOps Lifecycle is the set of phases that includes DevSecOps for taking part in development and Operation group duties for quicker software program delivery. DevOps follows positive techniques that consist of code, building, testing, releasing, deploying, operating, displaying, and planning**.** DevSecOps lifecycle follows a range of phases such as non-stop development, non-stop integration, non-stop testing, non-stop monitoring, and non-stop feedback. Each segment of the DevSecOps lifecycle is related to some equipment and applied sciences to obtain the process. Some of the frequently used tools are open source and are carried out primarily based on commercial enterprise requirements. DevSecOps lifecycle is effortless to manipulate and it helps satisfactory delivery.

1. Continuous Development
2. Continuous Integration
3. Continuous Testing
4. Continuous Deployment/Continuous Delivery
5. Continuous Monitoring
6. Continuous Feedback
7. Continuous Operations

## 3.1. Continuous Development

In Continuous Development code is written in small, continuous bits rather than all at once, Continuous Development is important in DevOps because this improves efficiency every time a piece of code is created, it is tested, built, and deployed into production. Continuous Development raises the standard of the code and streamlines the process of repairing flaws, vulnerabilities, and defects. It facilitates developers' ability to concentrate on creating high-quality code.

## 3.2. Continuous Integration

Continuous Integration can be explained mainly in 4 stages in DevOps. They are as follows:

1. Getting the Source Code from SCM
2. Building the code
3. Code quality review
4. Storing the build artifacts

The stages mentioned above are the flow of Continuous Integration and we can use any of the tools that suit our requirement in each stage and of the most popular tools are GitHub for source code management(SCM) when the developer develops the code on his local machine he pushes it to the remote repository which is GitHub from here who is having the access can Pull, clone and can make required changes to the code. From there by using Maven we can build them into the required package (war, jar, ear) and can test the Junit cases.SonarQube performs code quality reviews where it will measure the quality of source code and generates a report in the form of HTML or PDF format. Nexus for storing the build artifacts will help us

to store the artifacts that are build by using Maven and this whole process is achieved by using a Continuous Integration tool Jenkins.

## 3.3. Continuous Testing

Any firm can deploy continuous testing with the use of the agile and DevOps methodologies. Depending on our needs, we can perform continuous testing using automation testing tools such as Testsigma, Selenium, LambdaTest, etc. With these tools, we can test our code and prevent problems and code smells, as well as test more quickly and intelligently. With the aid of a continuous integration platform like Jenkins, the entire process can be automated, which is another added benefit.

## 3.4. Continuous Deployment/ Continuous Delivery

Continuous Deployment: Continuous Deployment is the process of automatically deploying an application into the production environment when it has completed testing and the build stages. Here, we'll automate everything from obtaining the application's source code to deploying it.

Continuous Delivery: Continuous Delivery is the process of deploying an application into production servers manually when it has completed testing and the build stages. Here, we'll automate the continuous integration processes, however, manual involvement is still required for deploying it to the production environment.

## 3.5. Continuous Monitoring

DevOps lifecycle is incomplete if there was no Continuous Monitoring. Continuous Monitoring can be achieved with the help of Prometheus and Grafana we can continuously monitor and can get notified before anything goes wrong with the help of Prometheus we can gather many performance measures, including CPU and memory utilization, network traffic, application response times, error rates, and others. Grafana makes it possible to visually represent and keep track of data from time series, such as CPU and memory utilization.

## 3.6. Continuous Feedback

Once the application is released into the market the end users will use the application and they will give us feedback about the performance of the application and any glitches affecting the user experience after getting multiple feedback from the end users' the DevOps team will analyze the feedbacks given by end users and they will reach out to the developer team tries to rectify the mistakes they are performed in that piece of code by this we can reduce the errors or bugs that which we are currently developing and can produce much more effective results for the end users also we reduce any unnecessary steps to deploy the application. Continuous Feedback can increase the performance of the application and reduce bugs in the code making it smooth for end users to use the application.

## 3.7. Continuous Operations

We will sustain the higher application uptime by implementing continuous operation, which will assist us to cut down on the maintenance downtime that will negatively impact end users' experiences. More output, lower manufacturing costs, and better quality control are benefits of continuous operations.

- Different Phases of the DevOps Lifecycle

- Plan: Professionals determine the commercial need and gather end-user opinions throughout this level. In this step, they design a project plan to optimize business impact and produce the intended result.
- Code – During this point, the code is being developed. To simplify the design process, the developer team employs lifecycle DevOps tools and extensions like Git that assist them in preventing safety problems and bad coding standards.
- Build – After programmers have completed their tasks, they use tools such as Maven and Gradle to submit the code to the common code source.
- Test – To assure software integrity, the product is first delivered to the test platform to execute various sorts of screening such as user acceptability testing, safety testing, integration checking, speed testing, and so on, utilizing tools such as JUnit, Selenium, etc.
- Release – At this point, the build is prepared to be deployed in the operational environment. The DevOps department prepares updates or sends several versions to production when the build satisfies all checks based on the organizational demands.
- Deploy – At this point, Infrastructure-as-Code assists in creating the operational infrastructure and subsequently publishes the build using various DevOps lifecycle tools.
- Operate – This version is now convenient for users to utilize. With tools including Chef, the management department take care of server configuration and deployment at this point.
- Monitor – The DevOps workflow is observed at this level depending on data gathered from consumer behavior, application efficiency, and other sources. The ability to observe the complete surroundings aids teams in identifying bottlenecks affecting the production and operations teams' performance.

# 4: How dose DevSecOps works

DevSecOps works by integrating security practices into the DevOps process, creating a culture of security throughout the software development lifecycle. Here are some key aspects of how DevSecOps works:

## 4.1. Automation:

DevSecOps emphasizes automation of security processes, such as code scanning, vulnerability testing, and compliance checks. By automating these tasks, security can be integrated seamlessly into the development pipeline without slowing down the process.

## 4.2. Shift-Left Approach:

DevSecOps promotes a "shift-left" approach, where security is addressed early in the development process rather than as an afterthought. By incorporating security practices from the planning and design phase, vulnerabilities can be identified and fixed earlier in the development cycle.

## 4.3. Collaboration:

DevSecOps encourages collaboration between development, operations, and security teams. By breaking down silos and fostering communication between these teams, security concerns can be addressed more effectively throughout the development process.

## 4.4. Continuous Monitoring:

DevSecOps emphasizes continuous monitoring of applications and infrastructure for security threats and vulnerabilities. By monitoring in real-time, security incidents can be detected and addressed promptly.

## 4.5. Compliance and Governance:

DevSecOps ensures that security practices align with regulatory requirements and industry standards. By incorporating compliance checks into the development pipeline, organizations can ensure that security controls are in place to meet regulatory obligations.

## 4.6. Feedback Loops:

DevSecOps relies on feedback loops to continuously improve security practices. By collecting data on security incidents, vulnerabilities, and compliance issues, organizations can learn from past experiences and enhance their security posture over time.

# 5: Exline well known DevSecOps tools.

There are several well-known DevSecOps tools that organizations can use to integrate security into the DevOps process. Here are some popular tools in different categories:

## 5.1. Static Application Security Testing (SAST):

- **SonarQube:** A platform for continuous inspection of code quality and security vulnerabilities.
- **Checkmarx:** Provides static application security testing to identify and fix security vulnerabilities in code.

## 5.2. Dynamic Application Security Testing (DAST):

- **OWASP ZAP (Zed Attack Proxy):** An open-source tool for finding security vulnerabilities in web applications.
- **Netsparker:**An automated web application security scanner that identifies vulnerabilities in web applications.

## 5.3. Container Security:

- **Docker Bench for Security**: A script that checks for common best practices around deploying Docker containers securely.
- **Clair:** An open-source tool for the static analysis of vulnerabilities in containers.

## 5.4. Infrastructure as Code Security:

- **Terraform:** Infrastructure as code tool that can be used with security modules to scan and enforce security policies.
- **Chef InSpec:** A tool for automating compliance and security testing of infrastructure.

## 5.5. Secrets Management:

- **HashiCorp Vault:** Securely store and manage sensitive information, such as passwords and API keys.
- **AWS Secrets Manager:** A service that helps you protect access to your applications, services, and IT resources.

## 5.6. Compliance and Policy Enforcement:

- **Open Policy Agent (OPA):** A policy engine that enables fine-grained, context-aware policy enforcement across the stack.
- **Chef Automate:** Provides compliance automation and reporting capabilities to ensure infrastructure and applications meet security standards

# 6: Benefits of DevSecOps

DevSecOps, which combines development, security, and operations practices, offers several benefits to organizations looking to enhance their security posture while maintaining agility and efficiency in their software development processes. Some of the key benefits of DevSecOps include:

- Benefits of DevSecOps

DevSecOps enables a development team to deliver and deploy code quickly without sacrificing security. This results in several auxiliary benefits.

Save Time

Delivering code quickly is fairly easy. A DevOps team could write the code and release it—often without noticing or even ignoring—potential security issues. However, over time, the vulnerabilities that were not addressed in the development process may come back to haunt the organization, the development team, and those the application is meant to serve. This would likely result in the developers having to waste time going back and addressing security issues.

With development security operations as an inherent part of the process, vulnerabilities are addressed at each design phase. Therefore, the development team can release a more secure iteration of the program faster.

- Reduce Costs

Security issues can cause expensive, time-consuming delays. The person-hours necessary to develop an application greatly increase when developers have to go back and redo much of the coding to address vulnerabilities. Not only does this involve more time invested in a project but also keeps those same professionals from working on other projects that could benefit the organization's bottom line.

If an organization uses a DevSecOps lifecycle, on the other hand, the need to go back and make changes can be significantly reduced, conserving person-hours and freeing up the development team to engage in other work.

In addition, this could lead to a better return on investment (ROI) for your security infrastructure. As the security team fixes problems upfront in the design process, their work precludes many future problems. This not only results in a more secure application but also reduces the number of issues your security infrastructure will have to deal with down the road.

- Proactive Security

Vulnerabilities in code can be detected early if you implement a DevSecOps approach. The DevSecOps model involves analyzing code and performing regular threat assessments. This proactive approach to security enables teams to take control of an application's risk profile

instead of merely reacting to issues as they pop up—particularly those that would have been detected during threat assessments.

- Continuous Feedback

DevSecOps creates a continuous feedback loop that interweaves security solutions during the software development process. Whether your DevOps is done using on-premises servers or you use cloud DevOps, developers get constant feedback from the security specialists on the team. Likewise, the security team obtains continuous feedback from developers, which they can use to design solutions that better fit the application's infrastructure and function.

Continuous feedback also improves the development of automated security functions. The security team can gather information about the application's workflow from the development team and use that feedback to design automation protocols that benefit processes specific to that exact application.

Furthermore, continuous feedback allows the team to program alerts signaling the need for adjustments in the design of the application or tweaks to its security features. Knowledge regarding what each team needs to be aware of and how that affects the process of building the application can be used to decide the various conditions that should trigger different alerts. With well-designed secure DevOps automation, the team can produce secure products in less time.

- Build Collaboration Between Teams

A more collaborative environment is one of the cultural benefits of a DevSecOps approach. Throughout the entire development lifecycle, communication is enhanced because team members must understand how each facet of an application interfaces with the necessary security measures. As the different teams combine minds to solve this puzzle, collaboration is increased, and in the end, you get a more cohesive organization and product.

- Early Detection and Mitigation of Security Vulnerabilities: By integrating security practices into the development process, security vulnerabilities can be identified and addressed early on, reducing the likelihood of security incidents in production.
- Automated Security Testing: DevSecOps tools automate security testing processes, enabling continuous monitoring and assessment of code and infrastructure for security issues, leading to faster detection and remediation of vulnerabilities.
- Improved Collaboration: DevSecOps promotes collaboration between development, security, and operations teams, fostering a shared responsibility for security and enabling cross-functional teams to work together towards common security goals.
- Faster Time to Market: By incorporating security into the development pipeline, organizations can reduce the time it takes to address security concerns, allowing them to release software more quickly without compromising on security.
- Compliance and Regulatory Alignment: DevSecOps practices help organizations align with industry regulations and compliance standards by integrating security

controls and audits into the development process, ensuring that applications meet security requirements.

- Enhanced Resilience and Risk Management: By proactively addressing security risks throughout the software development lifecycle, organizations can improve the resilience of their applications and reduce the likelihood of security breaches or downtime.
- Continuous Improvement: DevSecOps encourages a culture of continuous improvement by providing feedback loops for security practices, enabling organizations to learn from incidents and make iterative improvements to their security posture over time.

DevSecOps helps organizations build more secure, resilient, and compliant applications by integrating security practices into the development process from the outset, ultimately leading to better outcomes for both the organization and its customers.

## 7: Local and international DevSecOps career opportunities, career path.

Local and international DevSecOps career opportunities are abundant, given the increasing demand for professionals who can bridge the gap between development, security, and operations. Here are some insights into DevSecOps career paths and opportunities:

- DevSecOps Engineer/Analyst: This role involves integrating security practices into the software development lifecycle, implementing security tools and automation, and collaborating with development and operations teams to ensure secure and efficient software delivery. DevSecOps engineers often work on implementing security controls, conducting security assessments, and driving cultural change within organizations.
- Security Automation Engineer: Professionals in this role focus on automating security processes, such as vulnerability scanning, compliance checks, and security testing, to enable continuous security monitoring and remediation within DevSecOps pipelines.

- Cloud Security Engineer: With the increasing adoption of cloud technologies, there is a growing need for professionals who can specialize in securing cloud environments and ensuring that DevSecOps practices are applied effectively within cloud infrastructure.

- Security Architect: Security architects design and implement security solutions for software systems, ensuring that DevSecOps principles are integrated into the architecture of applications and infrastructure.

- DevSecOps Manager/Director: Experienced professionals may take on leadership roles, overseeing the implementation of DevSecOps practices across an organization, driving cultural change, and aligning security initiatives with business goals.

Career paths in DevSecOps typically involve a combination of technical skills, security knowledge, and experience with development and operations. Professionals often start with a background in software development, system administration, or cybersecurity, and then acquire specialized skills in areas such as automation, containerization, cloud security, and secure coding practices.

In terms of international opportunities, many organizations worldwide are actively seeking DevSecOps talent to enhance their security posture and accelerate their software delivery. As a result, professionals with DevSecOps expertise have the potential to pursue opportunities in various industries and regions globally.

To advance in the field of DevSecOps, continuous learning and staying updated with the latest tools, technologies, and best practices are essential. Obtaining relevant certifications such as Certified DevSecOps Professional (CDP), Certified Kubernetes Security Specialist (CKS), or Certified Cloud Security Professional (CCSP) can also bolster career prospects in this domain. Additionally, active participation in the DevSecOps community through conferences, meetups, and open-source contributions can help professionals expand their networks and stay connected with industry trends.

## Conclusion

DevSecOps represents a paradigm shift in the world of software development, where security is not an afterthought but an integral part of the entire development process. By weaving security practices into the fabric of DevOps, organizations can reap a myriad of benefits, ranging from improved security posture to accelerated delivery cycles and enhanced collaboration among disparate teams. Throughout this document, we have explored the meaning and essence of DevSecOps, underlining its pivotal role in fostering a culture of shared responsibility and proactive security measures. We have also delved into the array of benefits that come with embracing DevSecOps, including heightened resilience against cyber threats, enhanced customer trust, and increased efficiency in software delivery. Understanding the lifecycle of DevSecOps is paramount for organizations embarking on this transformative journey. From the initial planning stages through development, testing, deployment, and monitoring, each phase of the lifecycle presents unique opportunities to embed security into the software delivery pipeline. Lastly, the tools and technologies that support DevSecOps implementation serve as enablers for seamless integration of security into every step of the development lifecycle. Automation, continuous monitoring, and actionable insights provided by these tools empower organizations to proactively address security vulnerabilities and deliver robust and secure software solutions. As we conclude our exploration of DevSecOps, it is evident that this methodology presents a holistic approach to software development that prioritizes security without compromising speed and agility. Embracing DevSecOps is not just a choice; it is a strategic imperative for organizations looking to stay ahead in today's rapidly evolving digital landscape. By adopting DevSecOps principles, organizations can build a strong foundation for secure, reliable, and resilient software solutions, safeguarding their assets, reputation, and stakeholders in an increasingly interconnected world.

## Reference

1: https://www.mavhem.securitv/blog/the-devsecops-lifecycle-how-to-automate-securitv-in-softwaredevelopment. "Lifecycle of DevSecOps"

2: https://ranianiitian.medium.com/top-devsecops-tools-for-2023-open-source-solutions-for-enterprises- 7c146f80b325. "what are well known tools for