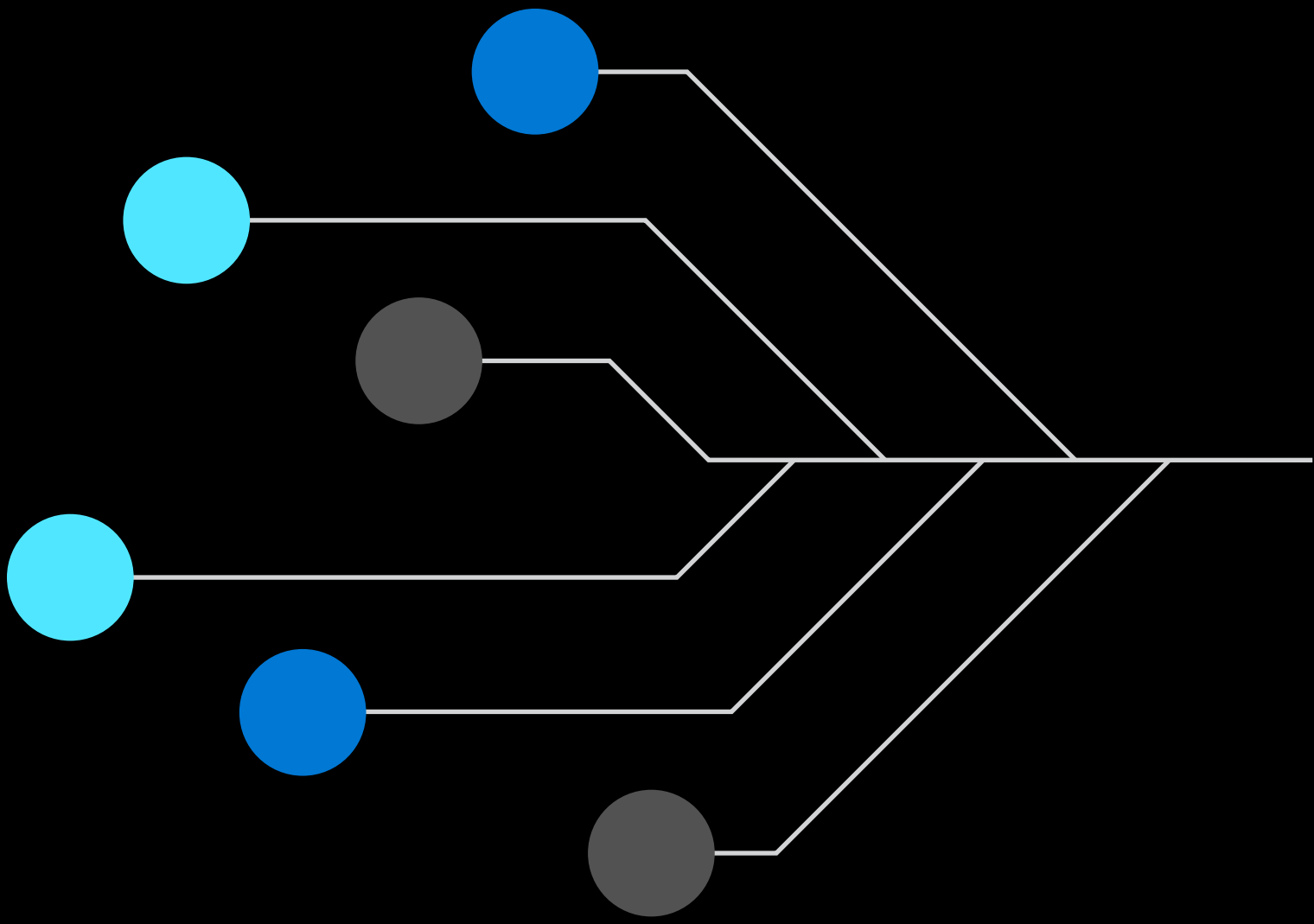
Modern App Development
and Enterprise DevOps Series

Microsoft sogeti Part of Capgemini

6 Tips to Integrate Security into Your DevOps Practices



Summary

DevOps proven practices illustrate how collaboration between developer and operations teams leads to faster software delivery. Now, the issue facing digital leaders is the security and compliancy of their code, workflows, and infrastructure. The logical next step: integrate your security team with the existing DevOps team—breaking down another organizational silo. The most challenging part of DevSecOps adoption is to make security complement existing business processes, culture, and people. How can technical leaders develop cross-function collaboration and unite developer, security, and operations teams around the culture of security as a shared responsibility?

The tips inside include how to:

Develop a security-first company culture to drive DevSecOps adoption

Proactively secure your code, workflows, infrastructure, and software supply chain against vulnerabilities

Provide your teams with shared tooling and best practices to enable end-to-end visibility and traceability

Leverage improved insights and policy automation to realize continuous compliancy

The most challenging part of DevSecOps adoption is to make security complement existing business processes, culture, and people

Table of contents

6 tips to integrate security into your DevOps practices

TIP 1 Build a security-first culture across the business 5

TIP 2 Integrate security in the early stages of the development lifecycle 9

TIP 3 Monitor and observe continuously with purpose 15

TIP 4 Embrace everything-as-code 23

TIP 5 Realize compliancy with policy automation 31

TIP 6 Secure and visualize your software supply chain 35

Closing thoughts 38

How Microsoft and Sogeti can help 40

Introduction

Throughout the years, software development practices evolved to serve the needs and the speed of business. Recently, DevOps methodologies provided software engineers and operations teams with a faster and more efficient way to develop code. However, efficient DevOps practices uncovered a new bottleneck, pushing security to the end of application development and management. This bottleneck is part of the reason organizations typically take 218 days[1] to uncover a security vulnerability, which can be extremely costly. NIST[2] estimated the cost of fixing a security defect in production can be up to 60 times more expensive than during the development cycle. Which is why research by McKinsey[3] indicates embedding security early into the stages of application development and management—or shifting left—is a major investment focus for digital leaders. These leaders recognize that integrating security into their pipelines and leveraging modern platform capabilities is the next logical evolution of the DevOps methodology, DevSecOps.

Now, the issue facing digital leaders is the security and compliancy of their code, workflows, and infrastructure—all of which deal with external pressure from tight delivery deadlines. To make rigid deadlines, organizations often forego security best practices and deploy code with known vulnerabilities. Compliance also remains a key issue due to relating audits' exhaustive and time-consuming nature. Forbes reports[4] "Some CISOs spend 30% or more of their time dealing with compliance issues." So how can your enterprise harden security and address compliance at the same time?

Yet again, the answer is collaboration. It's time to include security within your DevOps teams. Your DevSecOps team's collaborative success relies on shared tooling and visibility into application health at every stage of application development and management (ADM). Through early detection, organizations drive efficient and cost-effective fixes of security vulnerabilities. At the same time, capturing opinionated insight at every stage of ADM enables organizations to achieve continuous compliancy. The most challenging task is to make security complement existing business processes, culture, and people. It's crucial to develop cross-function collaboration and unite development, security, and operations teams around the culture of security as a shared responsibility.

Improving security posture isn't just about moving security to an earlier phase of ADM, it's about adopting a different way of working, one that emphasizes cross-team collaboration, shared empathy, and shared responsibility. Ideally, security is baked into ADM, so teams don't see it as an extra step but as an integral step to software delivery. Embracing DevSecOps requires organizations to shift their culture, evolve existing processes, leverage modern platform capabilities, and strengthen governance. Here are six tips for you, a technical leader, to integrate security with your DevOps practices.

Your DevSecOps team's collaborative success relies on shared tooling and visibility into application health at every stage of application development and management (ADM).

[1]GitHub, Octoverse Security Report, 2020
[2]Security Boulevard, The Importance of Fixing and Finding Vulnerabilities in Development, 2020
[3]Microsoft, Developer Velocity: Lessons from Digital Leaders, 2021
[4]Forbes, Awash In Regulations, Companies Struggle With Compliance, 2019

TIP 1

Build a security-first culture across the business

Developing a security community in your enterprise improves buy-in across the organization and energizes employees

DevSecOps starts with the people. You do not "implement" DevSecOps, you embrace it. And for that to happen, your organization needs to adopt a DevSecOps culture while "living and breathing" it, via executive buy-in. For success, you'll need the entire organization, not just the IT people, product teams, and project managers.

Modern security is dependent on teamwork. A security group alone no longer offers sufficient protection for your enterprise. Novel threats push security earlier in the software lifecycle to occur closer to applications. Now, effectiveness depends on the ability of the security, developer, and operations teams to work together and share knowledge. Often, this starts with the security architect working alongside the DevSecOps teams—adding security principles to the very first stages of development. Developers should also expand their toolkit with operations knowledge. Tooling can help, but awareness and mindset are key. It all starts with teaching everyone to appreciate a true DevSecOps way of working.

Culture is the most crucial part of the adoption process. Thus, it is recommended to start with people, move to processes, and then support it with technology. Heavy investment in technology fails if your people have no interest in adoption. It requires a cultural shift for people to consistently practice a security mindset. DevSecOps is based on a shared security model, wherein teams need to collaborate. In this model, security is not viewed as any one team's responsibility, but as a collective. Note: this does not mean that specialized security and infrastructure personnel are no longer required.

To start your DevSecOps journey, you'll need to cultivate the understanding that security is a shared responsibility throughout the organization through (1) training that empowers your teams and (2) a strong InnerSource community.

Revitalize your training efforts

Training is crucial for everyone in your DevSecOps team to understand not only their role, but also how it intersects with other responsibilities on the team. Through cross-team knowledge sharing, the hope is for everyone to raise their security awareness. Remember, your DevSecOps program will take time to grow. Widespread adoption also requires backing from management, so teams aren't overly pressured as they take time to mature with new processes and tooling.

Start building a security-aware team through early adoption of the Security Champion Model. The champion is nominated or chosen from the team and becomes the voice of security for the team. They'll engage in all the security-related activities from inception to release, but the security champion is not solely responsible for all security issues in a given release. Instead, they coordinate and track security issues while communicating with relevant stakeholders. Use the security champion to act as an on-site security advisor who can anticipate potential security issues and work on risk analysis—outlining security requirements early in the development phase. These insights help build a security foundation for your DevSecOps team. Generally, think about assigning this role to a senior and experienced stakeholder as it requires an optimum level of communication, practical knowledge of security, and willingness to confidently voice their opinion on red flags.

Assess your organization's readiness for a DevSecOps cultural change with the following types of questions:

Do senior stakeholders openly and explicitly support the use of Lean, Agile, and DevOps methods in the program?

Does the organization support direct collaboration among development, testing, and operations teams?

Are security leaders aware of the new role their teams will play in modern application development and management?

Will the organization provide the physical and social environments needed for team success?

Will management understand and advocate for the extra time and effort to ramp up DevSecOps techniques?

Use the security champion to act as an on-site security advisor who can anticipate potential security issues and work on risk analysis—outlining security requirements early in the development phase.

Kickstart a strong InnerSource community

Keeping the security mindset alive means working hard to build community. Just training your people and starting a culture is not enough. Long-lasting success means cultivating a vibrant and energetic community of people.

One way to kickstart your community is through adopting InnerSource best practices, where teams share and adopt ready-to-use reference architectures, code, and common components to streamline and optimize their workstreams. This collaborative way of thinking delivers increased delivery velocity, smoother collaboration between groups, higher-quality development, and better documentation. Have your community set up events around InnerSource patterns and position it as an enabler for a new way of working.

Appoint a group of key individuals to guide the onboarding process for those interested in joining the community. These guides should outline InnerSource best practices and develop useful educational content. At the same time, instruct the guides to build out the InnerSource repository—setting a baseline and standards for others to view and eventually contribute to. This central location enables others in the community to help, either by reviewing or adding remarks.

Focus evangelism efforts on community events, around either topics in the InnerSource Library, or common DevSecOps problems and solutions. DevSecOps is powered by involvement, so keep the door open to everyone in your company.

Focus evangelism efforts on community events, around either topics in the InnerSource Library, or common DevSecOps problems and solutions.

To illustrate what a security-first culture looks like in practice, let's look at the fictional story of the online retail company "Custom City Clothing."

Custom City Clothing's story

Using a security champion to unite teams

Mark opens up his lunch, hoping the cobb salad will take his mind off the company culture initiative he's working on. As CEO, Mark leads Custom City Clothing and is excited about the upcoming DevSecOps transformation. But, at the same time, Mark's aware that the security, application, and operations teams don't interact much in their daily routines and will resist the change. Looking for a path forward, Mark reached out to Jodie, a senior application developer, to brainstorm some solutions.

As Jodie and Mark talk through some ideas, a Security Champion Model immediately catches their attention. They like the idea of a champion acting as an on-site security advisor. This champion should be someone capable of anticipating potential security issues and working on risk analysis. Ideally, at some point, this champion starts outlining security requirements earlier in the development phase.

Jodie and Mark then introduce this idea to the security, development, and Ops teams for nominations. They did, however, make a few requirements for any nomination. The first is that the role must be a senior and experienced stakeholder—ready to communicate across teams effectively. More so, they want a person with practical knowledge of security so they wouldn't feel scared voicing their opinion on red flags. After reviewing the nominations, they select Maddie, a developer with experience helping security initiatives in the past. Before adopting this role, Maddie met with security leaders for an accelerated security training course. This added knowledge gives her the context to be situationally aware when meeting across teams.

Maddie hit the ground running and started engaging in all security activities from inception to release. Now the liaison between teams, Maddie articulates the needed security updates, requirements, and compliance guardrails to the security teams based on operations and developer team requirements. She's also helped spearhead the development of InnerSource practices within the company—hoping to create a smoother collaboration between teams and grow documentation of reference architectures, application and infrastructure code, and common components. Having a single point of contact for security helps Mark gather buy-in from across the teams. And now that the cross-team cooperation and knowledge sharing has started in earnest, Mark feels more ready to introduce the process and technology changes that would complete Custom City Clothing's DevSecOps transformation.

TIP 2

Integrate security in the early stages of the development lifecycle

Thinking about security upfront and embedding security practices early will help you avoid vulnerabilities and common roadblocks

Choosing a security solution at the end of the development process offers limited protection for your critical code, data, and software supply chains. Instead of pushing security to the end of the development process, the idea is to do it as part of day-to-day development. It ensures security checks and best practices occur early and throughout the development process, reducing the likelihood of vulnerabilities.

So, how do we shift security left?

There's much to consider, including modern tooling, adoption of best practices, and buy-in across the organization. It begins with developers including security scans as part of their CI/CD workflows. These continuous security scans position enterprises to secure applications by design and minimize vulnerabilities.

More tactically, your enterprise can power a successful shift left with two specific practices: (1) moving from batch scanning to continuous assessments and compliance checks and (2) ensuring code quality, security posture, and compliance with both static and dynamic analysis.

Move from batch scanning to continuous assessments and compliance checks

It's time to evolve past the batch approach that many security teams currently use to scan for vulnerabilities. Batch scanning not only requires human involvement, it's also prone to errors and cannot work on demand. Moreover, these security checks occur independently from the developer team—limiting the context for security professionals. Delays may occur between software lifecycles and feedback often arrives late.

The fastest path to an optimum security posture is through continuous assessments and compliance checks. This happens at multiple levels from code analysis to unit testing of security functions. Here are some foundational tenets for successful continuous assessments:

Find common vulnerabilities and exposures (CVEs): Often the most common application vulnerabilities scanned for remediation are the OWASP Top 10 vulnerabilities. A tool like the Azure Security Center's regulatory compliance dashboard scans your Azure subscription in search of these vulnerabilities.

Expand your scans for compliance: Enterprises that don't scan for compliance or advanced threats leave themselves open to modern threats. The answer is to scan for more comprehensive vulnerabilities including compliance readiness using best practices like the NIST framework or the CIS Benchmark.

It's time to evolve past the batch approach that many security teams currently use to scan for vulnerabilities.

Detect early, fix early: Set processes to scan container images and infrastructure as code files for CVEs before they launch to ensure that no vulnerabilities go into production and remediation occurs as early as possible.

Automate processes: Build efficiency within your enterprise by switching to automated processes that find and remediate issues faster by scanning whenever a new change occurs, all while requiring no human intervention.

Improve the traceability: Ensure every step of the pipeline generates data that can later be used for auditing or trend analysis.

Manage by the metrics: Generate reports that grade your enterprise on the total CVE in your enterprise's infrastructure, level of code smell, and duplicated code.

Use quality gates to ensure compliance: Before each release, use a security gate to measure the quality of code against prescribed standards. If the code does not meet quality standards, pause the release to fix the vulnerabilities immediately before approving.

The fastest path to an optimum security posture is through continuous assessments and compliance checks.

Although the enterprise benefits greatly from automated processes, those used to batch scanning may feel they are no longer needed in the team. It's crucial to articulate this as a move away from the repetitive tasks taking up their time. Think about ways you can upskill this employee to be more effective in driving high-business value. Transitioning to continuous assessments and compliance checks may also cause friction with developer teams who see their application security roles expand. Dissuade the notion of security as time consuming and improve developer commitment by offering them continuous feedback and remediation recommendations on their code and the libraries they use (deprecation, vulnerabilities, OSS license incompatibilities).

Evaluate code quality and harden security with continuous static and dynamic analysis

Gathering a holistic understanding of the quality of your enterprise's delivery pipeline, code, and applications is called "Qualimetry." Using static and dynamic scans together helps to outline the Qualimetry of your enterprise's continuous integration pipelines. Consistently looking at quality indicators helps reduce the risk of drift and technical debt. Further, you allow the development team even greater autonomy to address code quality. Take this concept even further and implement a delivery criterion that limits functional bugs and bad code quality ratings. These CI pipeline quality checks harden the security of the whole system.

In many situations, enterprises often compile findings from different tools to get a more accurate picture of their Qualimetry. This picture includes not only code-based vulnerabilities but also dependency-related vulnerabilities, infrastructure issues, and image evaluation. Here are some considerations when compiling tools for a comprehensive security toolset:

Scan for vulnerabilities daily: Implement a code-based vulnerability tool like CodeQL within the daily pipeline to give the development team regular updates of known vulnerabilities. CodeQL integrates very easily with GitHub to report errors and vulnerabilities across development languages, such as Java and JavaScript. This reduces code correction time when modifications are made to qualify technical evolutions faster.

Incorporate a dependency checker: You'll also want to utilize a powerful dependency checker within your CI pipeline to scan for both dependencies pulled in the CI and assets in the repository. What's more, you can set up a firewall to block the pipeline based on security level objectives, customized for each application.

Search your container images: To safeguard secure deployment to containers, it's necessary to scan the container for image vulnerabilities.

Remember to scan infrastructure: Static application security testing (SAST) is not only a question of identifying application security threats, but also infrastructure issues. If your enterprise is already equipped with infrastructure-as-code (IaC), then it becomes easily controlled against a known list of issues and patterns. Azure Resource Manager and other IaC tools offer linting capabilities to better scan infrastructure code included in the CI.

Form a comprehensive analysis: Note that these tools should be used together to provide a complete vulnerability report.

Consistently looking at quality indicators helps reduce the risk of drift and technical debt.

As it stands, many organizations rely only on dynamic analysis of code, which gives an application-centric view using "black-box" techniques to evaluate code. These scans cannot offer the holistic view needed to accurately understand the current state of your enterprise's delivery pipelines. It's important to complement dynamic analysis with static analysis of code in delivery pipelines, so teams can be informed of potential problems as early as possible. Static code analysis enables you to measure code quality through indicators such as bugs, vulnerabilities, technical debt, unit test coverage or code duplication. It also helps locate security flaws through the recognition of known CVEs in certain dependencies.

Unfortunately, static analysis is a source of "false positives". So, once you perform SAST against applications and infrastructure, it's important to continue dynamic application security testing (DAST) to correct the false positives. From your CD pipeline, deploy a temporary environment and start dynamically checking security breaches against this temporary environment. You can then integrate a vulnerability assessment solution purpose-built for this task like Azure Defender. When performed on a temporary environment, you lower threat risk against critical data and systems while preventing a potentially dangerous new deployment.

This figure shows GitHub code scanning in context—alerting users to an untrusted URL redirection.

Most importantly, DAST and SAST scans cannot be performed alone or just once. They must run continuously. Regular and consistent scanning allows updates to be addressed as they appear rather than during elongated reviews occurring prior to each production release. These technical evolutions can thus benefit from the functional qualification phases to ensure technical non-regression. This combination of continuous testing, SAST, and DAST enables your enterprise to enhance code quality at the earliest opportunity while locking down vulnerabilities across infrastructure, images, and applications.

Regular and consistent scanning allows updates to be addressed as they appear rather than during elongated reviews occurring prior to each production release.

To illustrate how cultivating holistic vulnerability threat analysis looks in practice, let's look at the fictional story of financial services company "King Banking."

King Banking's story

Expanding security scans for compliance

Angela, the CEO of King Banking, sat down at her desk with a cup of matcha tea to start the day. She's equal parts excited and nervous about the upcoming launch of King Banking's personal finance app. On one hand she's very confident that users will love the design and functionality of the app. But, at the same time, she doesn't feel reassured that the app will launch with enough security and compliance controls. Underpinning this thought is that King Banking is based in Germany and subject to strict GDPR compliance requirements.

With launch only 9 months away, Angela set up a meeting with King Banking's CISO, Jonas, to develop a plan for alleviating their security and compliance concerns. Jonas knew he needed to bring in leaders from both the development and security teams to form a comprehensive DevSecOps team. Jonas started by inviting one of the security team leads, Johan, who conducts application security scans. From the application team, Jonas selected Mary to overview their current delivery pipelines.

The newly formed King Banking's DevSecOps team quickly got to work. Johan suggested that they move from conducting batch scanning to continuous assessments and compliance checks. With automated scanning, Mary knew the team needed a tool to find CVEs quickly. She knew the Secure DevOps Kit for Azure automatically scans for CVEs like the OWASP Top 10 vulnerabilities. Jonas also wanted insight into compliance and insisted they expand scans using best practices like the NIST framework or the CIS Benchmark. While impressed with the suggestions, Johan articulated the need to effectively manage the new data that would come from expanding scans. He outlined that they should use quality gates to ensure compliance standards before each release. If the code does not meet King Banking's quality standards, their team can pause the release to fix it before approval. Pleased with the steps outlined, Angela set out the DevSecOps team to execute against these suggestions and prepare for the personal finance app to launch.

Three months have passed since King's Banking launched their wildly successful personal finance app. King Banking's continuous assessment and compliance check approach protected their product launch from falling flat. And now, with every release, their DevSecOps team puts processes in place to scan for compliance and vulnerabilities before each code release is approved. With their commitment to compliance and data security, the DevSecOps team pleased both Angela and the GDPR regulators. Angela now eagerly awaits the launch of new product features instead of worrying about application security and loss of customer data.

TIP 3

Monitor and observe continuously with purpose

Planning out objectives for continuous, context-based monitoring and observation enables your enterprise to be more proactive against threats

Too often, enterprises leverage a monitoring or observability solution without adapting it to work for their organization. When enterprises fail to fully plan their monitoring initiative, they overload themselves with data. This can be like looking for a digital needle in a haystack. What's more, without gathering the right data in the right ways, data is often not actionable for your enterprise. The first step to enabling continuous monitoring—and growing the previously limited subset of intelligence—is strategic planning.

What does purposeful and successful continuous monitoring look like?

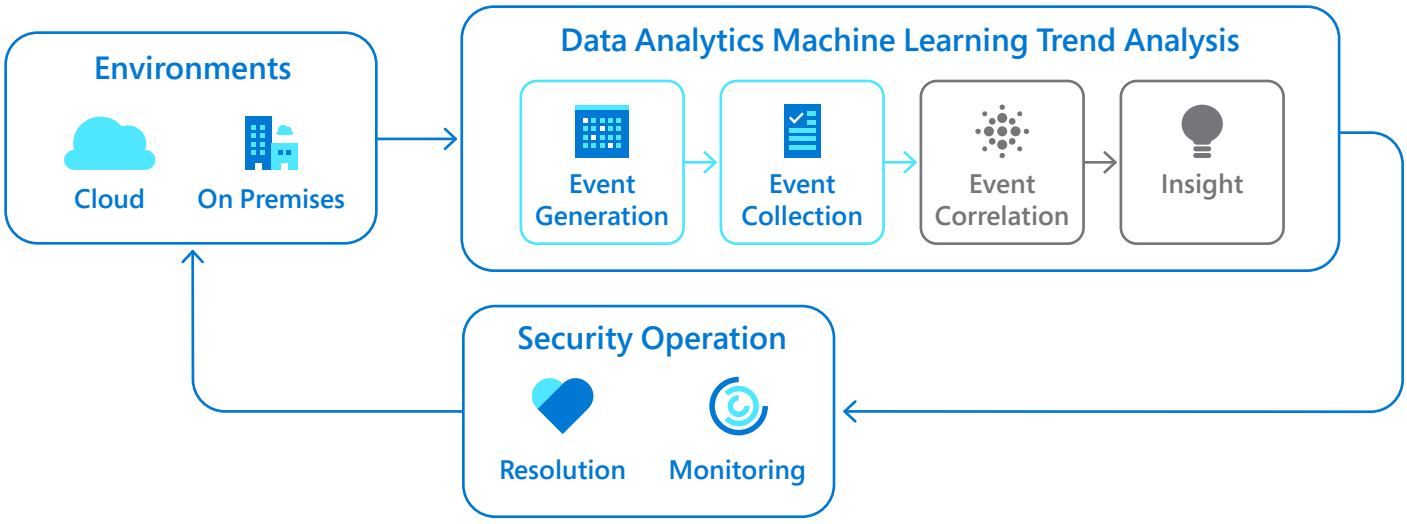
Successful monitoring practices rely on four things: (1) gathering holistic data that provides a complete picture, (2) structuring data for analysis, (3) using actionable alerts and threat intelligence to proactively react to threats faster, and (4) incorporating a robust monitoring toolchain built for modern threats.

Form a complete picture with structured data

Monitoring provides the most value when you observe everything occurring within your enterprise, be it active directory, firewall, syslog, application log, etc. Gathering data from an incomplete selection of sources gives your business blind spots, so don't forget to collect infrastructure data coming from outside the change management process.

Successful organizations don't just capture all their data; they organize it carefully. You'll need to decide what to log based on potential targets. Consider the 'signal-to-noise' ratio when collecting data from various sources. For instance, the syslog from a backup server might not harbor useful information like the syslog of the server hosting the identity solution. Use parameters to determine the baseline for any application, considering elements like user login/logout, network activity, system activity, transactions, etc. In general, log data should consist of who (user identity), when (activity start and end timestamp), what (activity performed), and where (source IP). In some instances, it's an industry or security standard that dictates the logging requirement, e.g., PCI-DSS, ISO27001.

Successful organizations don't just capture all their data; they organize it carefully



In this figure, you'll notice the Data Analytics Machine Learning Trend Analysis provides telemetry into an organization's security operation. So far in this Tip, we've discussed how event generation and event collection are important processes to conduct effective analysis. Later, we'll discuss how to use those in tandem with event correlation to provide meaningful insights to your DevSecOps team.

Leverage machine driven monitoring for analysis and reporting

Current systems generate more data and events than humans can interpret on their own. Too often, raw data is useless. Furthermore, collected events must be correlated together to provide a wider picture. The way to overcome these challenges is machine-driven monitoring. Ideally, it should collect and aggregate your logs and event data from various sources, and identify, analyze, standardize, and help find correlations. By correlating events, you can reduce the number of false positives and surface threats that would have stayed undetected otherwise. For instance, let's say your systems witnessed three events that on their own would not set off alarms. But, after correlating these events together, the incident would clearly reveal itself as a bigger issue.

Machine-driven monitoring starts with setting up the baseline for "normal" application behavior to detect meaningful deviation. Incorporating advanced analytics and machine learning helps indicate any unusual behavior signaling a breach. This next step involves modelling the normal behavior of the system using training data and keeping watch for any deviations. Whereas generically evaluated log data and data from other sources often delivers false positive results, mature machine learning solutions consistently make correct predictions.

While machine-driven monitoring unlocks insights to better protect the enterprise, the next complementary step is useful reporting. Useful reporting means that the reporting is flexible enough to provide insights using different levels of data, often engineered for different audiences. Useful reporting can pinpoint and zoom in on a precise domain while also providing a wider picture of the business.

Another tenet of useful reporting is human understanding. Because the human mind is designed to notice patterns in forms and colors, data visualization helps understanding and correlation analysis. It is far quicker to detect events out of the range of usual activity when displayed as graphs, rather than data tables. It is also a way of improving your team members' engagement, as graphics are often easier to grasp. Use them as a first approach when introducing developers to security matters before moving to more complex issues.

While machine-driven monitoring unlocks insights to better protect the enterprise, the next complementary step is useful reporting.

Combine actionable alerts with threat intelligence to enable proactive security

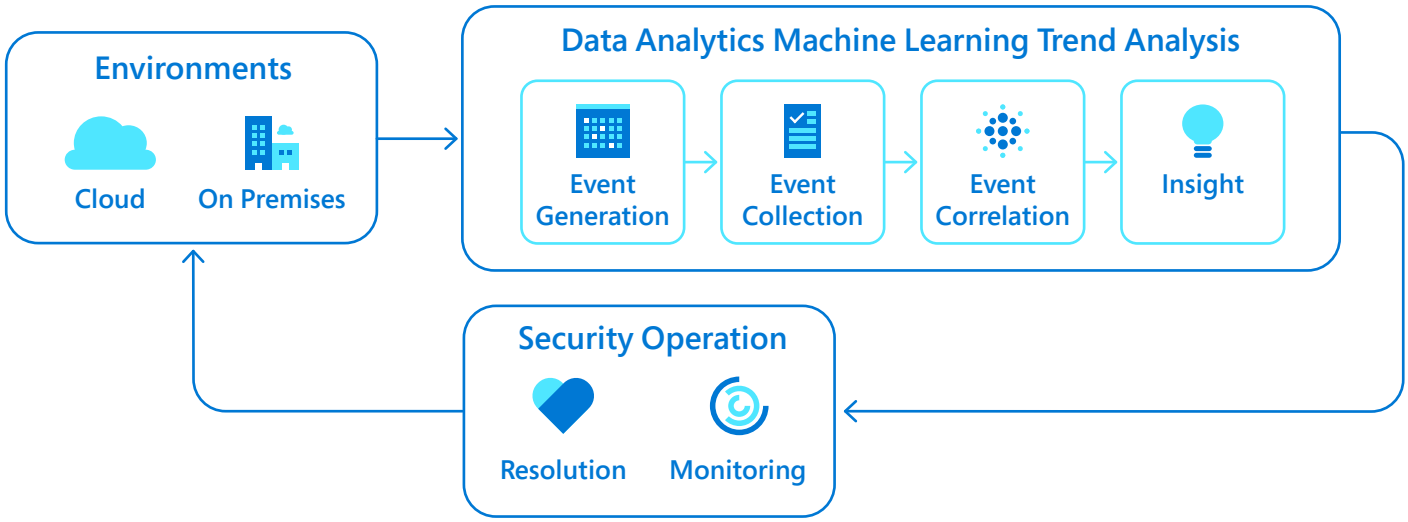
Informed enterprises use threat intelligence to gauge potential threats vs. recorded ones. Threat intelligence gathered from several sources about emerging and existing threats provides a greater understanding of threat capability, IOCs (Indicator of Compromise), and the tactics, techniques, procedures (TTPs), and mitigation controls to use against it.

Sharing threat intelligence data is a win-win situation. Organizations can share experiences with different threats according to their own unique diversity of services, geo-location, technology, etc. This equips all organizations with the knowledge to detect such threat actors. They can locate threat actors based on patterns of exploitation, repeated tools and techniques used, and common locations or vertical targets. It's also important that you integrate vulnerability management with threat intelligence to determine which vulnerabilities represent the biggest risks based on the threat landscape.

However, monitoring alone will not solve the issue if there is no response. Timely response to security incidents is critical. At the same time, responding requires ample data points for analysis, which in turn takes time and is subject to the availability of a security analyst. Help solve the incident by gathering data such as user location, compromised device name, IP, type of device, and last patch date across various systems like Active Directory and Configuration Management Database (CMDB).

Automating responses can be crucial to mitigating incidents in a timely manner and preventing a major incident. You'll need to integrate an orchestration tool with various other systems to analyze collected data for automated response and remediation. A couple of quick remediation tactics are: 1. blocking users via identity tools when detecting malicious behavior, and 2. blocking firewall ports in response to a DDOS attack.

Threat intelligence gathered from several sources about emerging and existing threats provides a greater understanding of threat capability, IOCs (Indicator of Compromise), and the tactics, techniques, procedures (TTPs), and mitigation controls to use against it.



With the steps we've discussed, your enterprise is now set up with a powerful Data Analytics Machine Learning Trend Analysis loop to better inform your security teams.

For proactive monitoring, you must also integrate with change management tools, preventing deployment of unauthorized changes to production. Make sure to run all updates through the change management system to validate it prior to release. If remediation is necessary, the change management system will execute the correct update and raise related high-priority service desk tickets for further investigation.

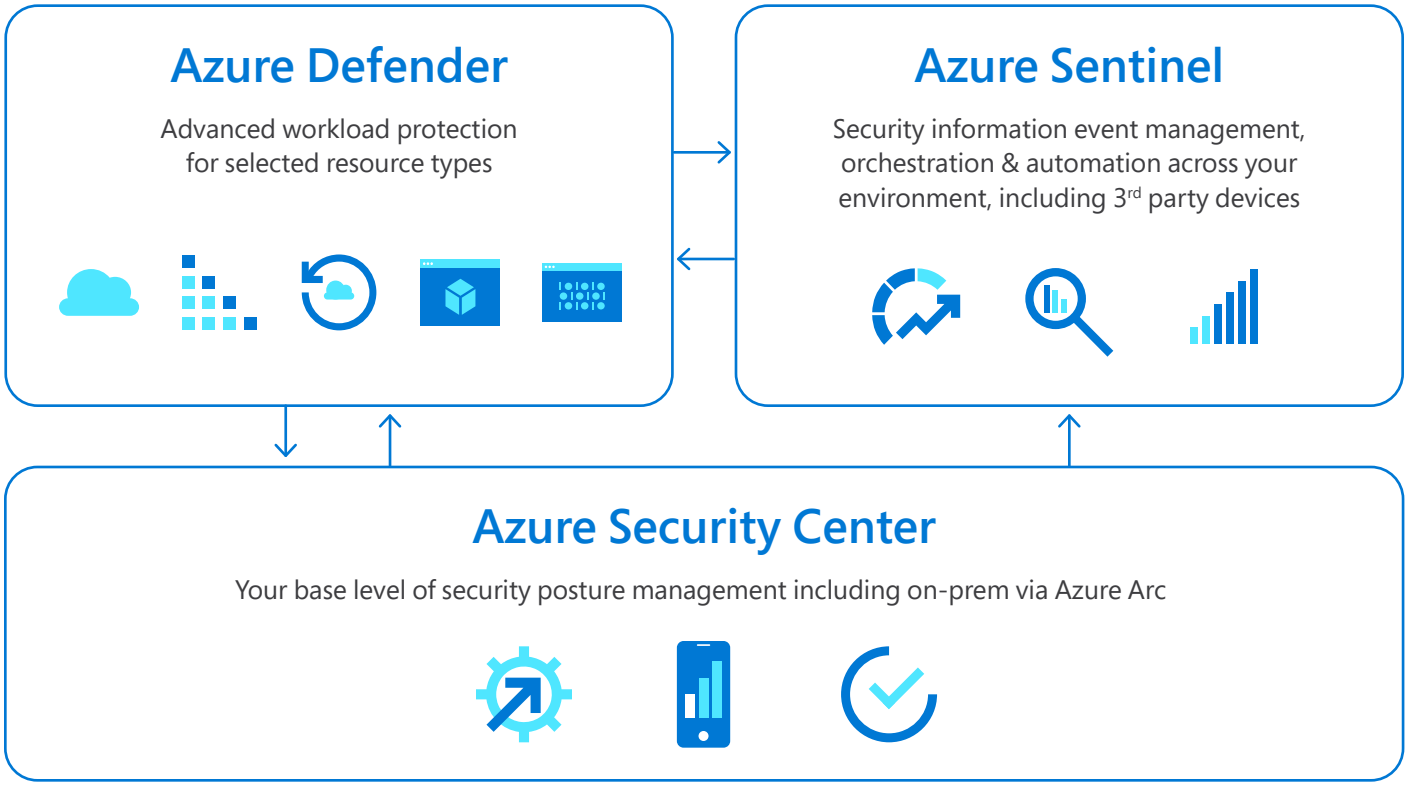
To illustrate which tools help perform consistent observability and monitoring, let's check out some Microsoft products mapped to the software development lifecycle.

Incorporate a robust toolchain built for modern threats

Often the hardest task is finding a security toolchain that works for your enterprise. Piecemeal solutions complicate matters as well. It's crucial that components of the toolchain work in harmony and protect your enterprise from all angles. Azure is designed to equip your organization with a tool chain that protects the entire software development lifecycle (SDLC), seamlessly. The figure below overviews which tool relates to each function of NIST Cybersecurity framework:

Identify	Protect	Detect	Respond	Recover
Azure Security Center	Azure Defender	Azure Sentinel		

Let's look closer to see how each component functions together:



Azure Security Center: Azure Security Center offers security posture management and threat protection for your hybrid cloud workloads by constantly monitoring resources for security misconfigurations. Working together with Azure Security Center, Azure Arc helps to standardize visibility, operations, and compliance across a wide range of resources and locations by extending the Azure control plane. Configuration guidance can be driven by best practices or the relevant standards for an organization (PCI, HIPAA, ISO, GDPR, etc.). The security state for all resources is visualized via the Azure Secure Score, which gives insight into the current security posture of your organization. From there, Azure Security Center makes recommendations for improving the score, so that your enterprise has a path forward.

Azure Defender: Azure Defender is one of the main components of Azure Security Center, providing protection to strategic workloads, resources, and services in your cloud environments. Azure Defender offers security for servers, App Service instances, Storage, SQL and SQL Databases, Key Vault, Resource Manager, DNS, Kubernetes clusters, and container registries. Don't worry, Azure Defender also provides protection to non-Azure servers, such as those hosted in on-premises data centers or other cloud providers. Azure Defender is a centralized tool with wide coverage and depth of security protection capabilities across the verity of cloud native workload.

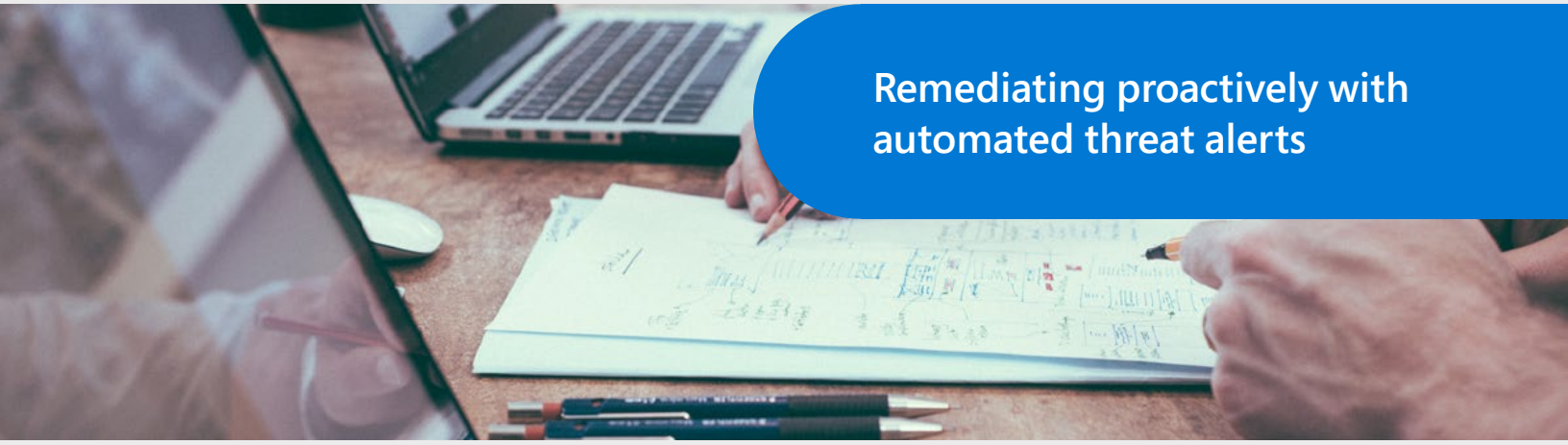
Start protecting workloads by scanning VMs, SQL servers, and container registries for vulnerabilities (analyzed by Qualys' cloud service). Next, you should set up Azure Defender to provide scanned results prioritized by criticality and paired with the latest available patch. Then, leverage Azure Defender's adaptive application control to baseline any known safe applications and VMs. Any deviation from this baseline will in turn trigger a security alert. This is just one way to leverage Azure Defender. When you enable Azure Defender for servers, you can use just-in-time VM access to lock down the inbound traffic to your VMs, reducing exposure to attacks while providing easy access to connect to VMs when needed. Other use cases include file integrity checks, adaptive network hardening, and network maps.

Azure Sentinel: Azure Sentinel delivers Security Information and Event Management (SIEM) and Security Orchestration, Automation, and Response (SOAR) capabilities to Azure for both cloud native resources and on-prem resources. It's also ready to integrate with non-Microsoft solutions using APIs. Azure Sentinel analyzes collected data/logs for threats and event correlations. Any detected threat is then reported as an incident for remediation. And for orchestration, Azure Sentinel comes with a playbook detailing 200+ connectors with different solutions. The playbook helps smooth the integration process with other tools and outlines proposed automation steps for the incident, like raising a ticket in ServiceNow.

Azure Security Center, Azure Defender, and Azure Sentinel work together using workflow automation via Azure Logic Apps that can trigger responses – Apps – on predefined events. For instance, Azure Security Center can trigger workflows to respond to threats using Azure Defender, such as sending an email notification to a security team on high severity alerts so that they can investigate. What's more, it can create a Network Security Group to counter a brute force attack. Likewise, Azure Sentinel can trigger a block on users in Azure Active Directory in the event any identity is compromised. These services not only provide meaningful insight for your resources but can also be configured to automatically respond to any security breach.

To illustrate how consistent threat monitoring looks in practice, let's look at the fictional story of the financial services company, Pension Experts.

Pension Experts' story



Remediating proactively with automated threat alerts

Matilda is the CEO of Pension Experts, a financial services firm in Amsterdam, Netherlands. In the past she'd felt confident about the security posture of her enterprise, but it's been years since the last security upgrade. Pension Experts provides full-service pension administration and collects personally identifiable information (PII). Matilda knows that PII like pension allocation, account information, and payment record demand a thorough approach to security—especially in the Netherlands where they have to comply with requirements from the De Nederlandsche Bank (DNB) and the Netherlands Authority for the Financial Markets (AFM).

Needing to understand more about their current security processes, Matilda called on her CISO, Stephanie, and a security team lead, Gregory, for more insight. Eager to help, Gregory overviewed their current security posture and how they secure different workloads. He noted that while functional, their current system couldn't find and remediate threats in a quick and proactive way. Stephanie agreed and soon after started researching solutions that help with security posture management. The team looked to regroup in a week and review some of Stephanie's top choices.

After Pension Experts' DevSecOps team hotly debated top security offerings presented by Stephanie, eventually they selected a solution comprised of Azure Security Center, Azure Defender, and Azure Sentinel. Using Azure Security Center, the team could consistently monitor all their environments while evaluating the current state of their organization effectively. Truthfully, the team was most excited to use Azure Defender and Azure Sentinel capabilities to quickly locate and remediate threats. Not only could this system conduct vulnerability scanning but it provided them with an opportunity to configure automated threat responses.

Fast forward six months and the DevSecOps team have fully deployed and implemented the solution. Gregory and Stephanie love the greater threat response time and telemetry their solution delivers. Instead of waiting for their scheduled security scans to address threats, Gregory receives an automated reminder whenever a threat appears. Now, he doesn't worry about a threat existing for days before a scan occurs. He even set up the new system to scan vulnerabilities with every code release so he can find threats as they occur. Best of all, CEO Matilda no longer worries about her company's ability to fend off threats before a disastrous breach.

TIP 4

Embrace everything-as-code

Adopting an everything-as-code approach helps your enterprise's deployment reliability, version control, and testing effectiveness

When your teams manually perform tedious tasks like provisioning infrastructure or managing application deployments, it prevents them from developing new, innovative code. An everything-as-code approach streamlines software development, delivery, and management, freeing up your developer teams to focus on development.

Similar to DevOps, an everything-as-code approach enables more efficient operations by standardizing the mechanisms of software development. An everything-as-code approach codifies aspects of development like infrastructure, schema, and pipelines, enabling you to manage governance from policy files rather than manual processes. Think of it as the ideological application of applying an application development approach to other components of IT (including DevOps) to ensure that best practices get defined and followed with minimum effort.

One of the biggest benefits of an everything-as-code approach is the decreased risk of human error. With workflows defined as code, there is less chance for an engineer following a manual checklist to forget a step or click a wrong button by mistake. It's easier to pass audits with everything-as-code configurations automatically logging the system's update history through Git changes.

An effective everything-as-code approach covers a variety of elements: (1) infrastructure-as-code, (2) immutable infrastructure, (3) a secure version control system, (4) configuration-as-code, (5) pipeline-as-code, and (6) policy-as-code.

Start your transformation with infrastructure-as-code

Many enterprises start their transition to everything-as-code by adopting infrastructure-as-code (IaC), which enables them to manage their IT infrastructure using configuration files. One IaC tool is Terraform, a tool developed by Hashicorp that enables you to predictably create, change, and improve secure infrastructure. We'll dive into some practical applications of Terraform later in this tip.

Most enterprises adopt IaC to unburden their teams from manual infrastructure management. Manual and tedious work slows down your enterprise and can't provide the needed efficiency, scalability, and testing required for modern security.

Below are some examples of how a successful infrastructure-as-code implementation alleviates the pain points of manual IT management:

Manual infrastructure management challenges

Scalability and inconsistency issues

Since manual configuration is slow, applications are unable to scale automatically. Complicating this, system administrators usually manage the load by creating servers manually. These delays can also impact general system availability as workers lose access to systems.

Aside from scalability, inconsistency is also a top concern. Many deployments or configurations are not repeatable, leading to heterogenous environments, and discrepancies between development environments and production environments.

Infrastructure-as-code's improvements

Repeatability of deployments

With IaC, you enable scalable deployments and configurations across your development environment, your staging environment, and your production environment. You also minimize drift between environments. For instance, every environment is built with the same version of every component across IT systems: development, staging and production environments. What's more, you can ensure that your middleware, OS, and your security patches are consistent throughout environments.

Manual infrastructure management challenges (cont.)

High cost

Manual infrastructure management is an expensive endeavor. Creating servers, networks, and configuring machines costs money and manpower. This also means your enterprise must employ specialists in every IT domain to create and maintain infrastructure. Further, you'll also need security specialists to scan and fix vulnerabilities, misconfigurations, and hacking attempts on your infrastructure. Running environments 24/7 is not cost effective. Often, teams won't destroy or shut down their environments because it's hard for them to recreate or reconfigure them easily.

Insufficient monitoring capabilities

Without IaC, you must manually ensure the system performs optimally without any bottlenecks. Issues can be hard to pinpoint and stem from numerous causes: bad configurations, wrong server sizes, network issues, or even poor application design.

Infrastructure-as-code's improvements (cont.)

Environment efficiency savings

With IaC, you save through efficient environment management. Your teams will be able to destroy unused environments without worry of loss. With everything stored and versioned, you can quickly recreate your environments by running your CI/CD pipelines. For example, you can create ephemeral environments for testing purposes in your CI/CD pipeline—running tests like integration tests, smoke tests, user acceptance tests, etc.

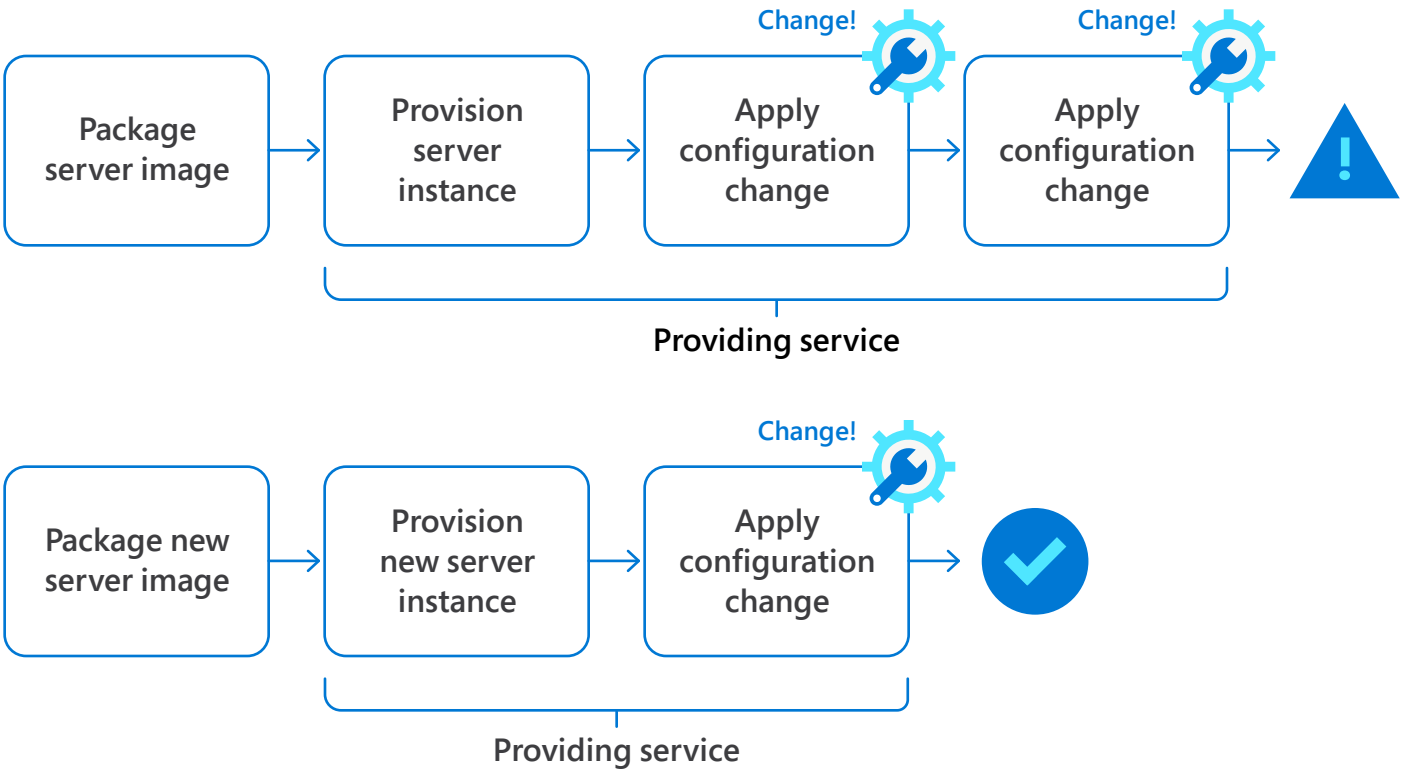
Improved versioning and testing

With IaC, your enterprise can now leverage developer paradigms for infrastructure. For example, you can store your infrastructure in Git, version it, and view the history of all the developer changes. You can then easily track when a bug appears and when the remediation occurs. You can also apply direct tests on your code, called "Test-driven infrastructure," where you can test an environment before deploying. This helps your infrastructure stay compliant with what's required.

IaC is just one effective way to increase repeatability, improve versioning and reduce costs, but IaC alone, however, has limitations—in the next section, we'll explore how adopting Immutable Infrastructure adds increased security controls to your everything-as-code approach.

Adopt immutable infrastructure

Even with infrastructure-as-code in place, long-running servers are at risk of configuration drift, a process in which a data center's production or data hosting infrastructure becomes altered over time from back-up and recovery configurations. Manually managed servers are especially prone to configuration drift—it's not possible to manage a server's configuration completely, meaning there are many opportunities for configuration drift or other unexpected server changes to occur. Unfortunately, every time a drift appears or a server crashes, you must rebuild it from scratch and apply a configuration.



The above figure outlines the difference between failing and effective change management. The top line showcases a system where too many changes are applied—leading to configuration drift—while the second line shows a new deployment system with fewer update occurrences. Ultimately, limiting the opportunities for updates within a system drives down the chance of configuration drift.

Practicing Immutable Infrastructure means a new approach to updating infrastructure. In the past, whenever an update was required, an update would subsequently push to the VM in question. This opens your organization to configuration drift. The difference with Immutable Infrastructure is that now instead of updating existing VMs, you create an entirely new VM that is an updated copy of the older server. This enables you to automatically install and scan images before deploying them to the cloud. In a DevSecOps approach, images are heavily tested for any present CVEs before being deploying at scale—thus, it's important not to use any configuration management tools that create opportunities for untested changes. A best practice is to apply any needed changes to the base image, test it, and then roll it out. Make sure to tear down and replace any servers that are yet to receive updates.

Taking it a step further, you can create an immutable image without any remote access or secure shell (SSH) access. This further secures the servers and protects against hacks—especially if it is exposed on the internet. And if a server becomes compromised, you can detach it from the cluster pool and put it aside for a forensic study. Later, recreate a new server with a valid and known configuration applied to the base immutable image. To accelerate server start up time, tools like Azure Image Builder enable you to automate the creation of base images set up with everything you need, including the application.

Immutable Infrastructure is very useful for passing security audits as your teams can track vulnerability creation and fixes in real time. If a new CVE is detected, you can rebuild a new base image with the proposed security fix, and automatically roll it in your cluster—fixing the vulnerability without any service interruption. In the end, Immutable Infrastructure adopts the same paradigm as building Docker images—deploying them inside a cluster and updating them every time a new image is created.

The ability to version control images is one of the biggest benefits of Immutable Infrastructure. With Immutable Infrastructure, you can create a file, the "recipe" to create the image, and store it in version control. The file is then versioned, wherein you can see every change subsequently made by your teams. Of course, it's important to ensure that the image is managed securely.

Immutable Infrastructure is very useful for passing security audits as your teams can track vulnerability creation and fixes in real time.

Store code in a secure version control system

The best way to store any version control files you create via Immutable Infrastructure is inside a secure version control system. These systems allow your enterprise to define granular access to any repository containing company code. Today, the most used format is Git.

Connecting your enterprise directory to your Git secures access to both application and infrastructure repositories. For example, you can connect GitHub to your Active Directory to help secure access via single sign-on (SSO). Additionally, you can tighten access control further by leveraging multi-factor authentication (MFA) alongside an Active Directory.

It's crucial that you define fine-grained access to your repository. Start by defining different roles (contributor, owner, reader) set up with different access policies to your code. Ensure your master (or main) branch is also secured. Do this by allowing only a select few to contribute to the master or main branch. Lastly, to further protect your production systems, limit contributions to pull requests. These version control and access policies in place, not only make your enterprise more secure, they also help make you better prepared for security audit.

The best way to store any version control files you create via Immutable Infrastructure is inside a secure version control system.

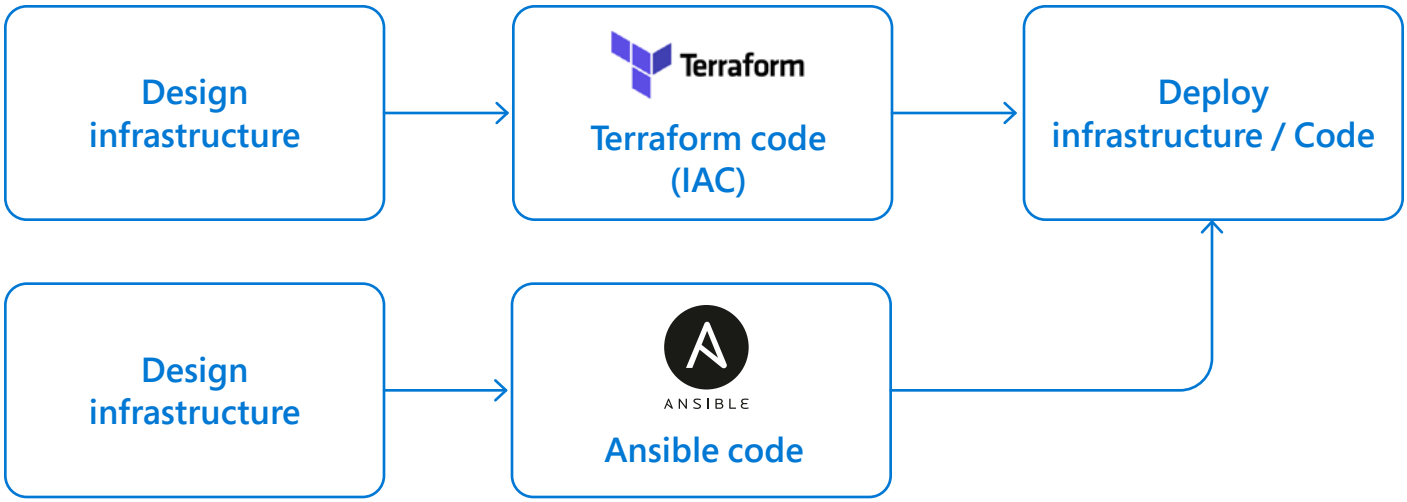
Set up configuration-as-code

At this point, we've only discussed infrastructure-as-code, but an everything-as-code approach goes beyond just your infrastructure. Configuration-as-code (CaC) takes it one step further to manage configuration resources as well. Ultimately, CaC allows server configurations to be replicated across environments, all without human intervention. Ansible is a great example of an open-source automation tool for configuration management and application deployment that can help you realize full CaC adoption.

Ansible's powerful automation simplifies long and complex tasks, allowing your teams to concentrate on developing added value. It works at a level above development languages, executing on YAML files to deploy configurations on specific figure types. This means you don't need to install any other software on the server.

With Ansible, you can quickly deploy applications and split their configurations into individual modules. Ansible executes each module using playbooks that act as an instruction manual to automatically control services, packages, and files.

CaC tools like Ansible work in tandem with other infrastructure-as-code tools, like Terraform. Terraform helps to define and create your system infrastructure. Ansible in turn, configures and deploys applications by executing its playbooks on the provided server instances. Use Terraform's integrations to execute a given Ansible script so you can use the programs together. Once united, these programs form a strong platform for effective configuration-as-code adoption.



This figure shows a complete IaC environment. In this example, the pipeline deploys infrastructure after it's designed first with Terraform. Following this step, the infrastructure is configured with Ansible.

Implement pipeline-as-code

To set up pipeline-as-code, take the application development-styled approach used for infrastructure-as-code or configuration-as-code. In a pipeline-as-code approach, you store all the CI/CD pipelines inside the version control system as files—enabling tighter version control for security reviews.

Successful adoption of pipeline-as-code in your application pipeline hardens security at every stage of deployment. For example, now you can scan your holistic application code using Static (SAST), Dynamic (DAST) and Interactive (IAST) Application Security Testing methods. You can also integrate Runtime Application Self-Protection (RASP) inside servers to further protect applications.

Think of pipeline-as-code as an application deployment blueprint for your team. This blueprint overviews all the stages needed to compile and deploy secured applications. With all these aspects now defined "-as-code," it's possible to enforce and deploy secure applications at scale for all of your teams.

Think of pipeline-as-code as an application deployment blueprint for your team. This blueprint overviews all the stages needed to compile and deploy secured applications.

In the next tip we'll explore another element of the everything-as-code approach, policy-as-code, and its powerful applications when used together with the other "-as-code" elements.

TIP 5

Realize compliancy with policy automation

Both the regulatory landscape and the software it governs are constantly changing, demanding an automated approach to policy compliance

Regulatory regimes are increasingly rigorous. That's a good thing. Nonetheless, ensuring compliance across a burgeoning application landscape isn't easy. The Enterprise DevOps 2020-2021 Report[1] states that almost half of the surveyed executives said they were not sure which data compliance standards they needed to meet. The report further noted that simply verifying the security of an application or environment when it's first deployed is no longer sufficient. Clearly, the logical solution is to ensure continuous compliance at every step of application development and management (ADM).

The best way to achieve this is with policy automation.

First, however, you need to set policies that relate to your enterprise's regulatory and compliance requirements, industry standards, and organizational goals—so that when an audit occurs, your security compliance is never in question. When done correctly, these policies offer a set of controls to your DevSecOps teams—ensuring security vulnerabilities are monitored and remedied at every stage of ADM.

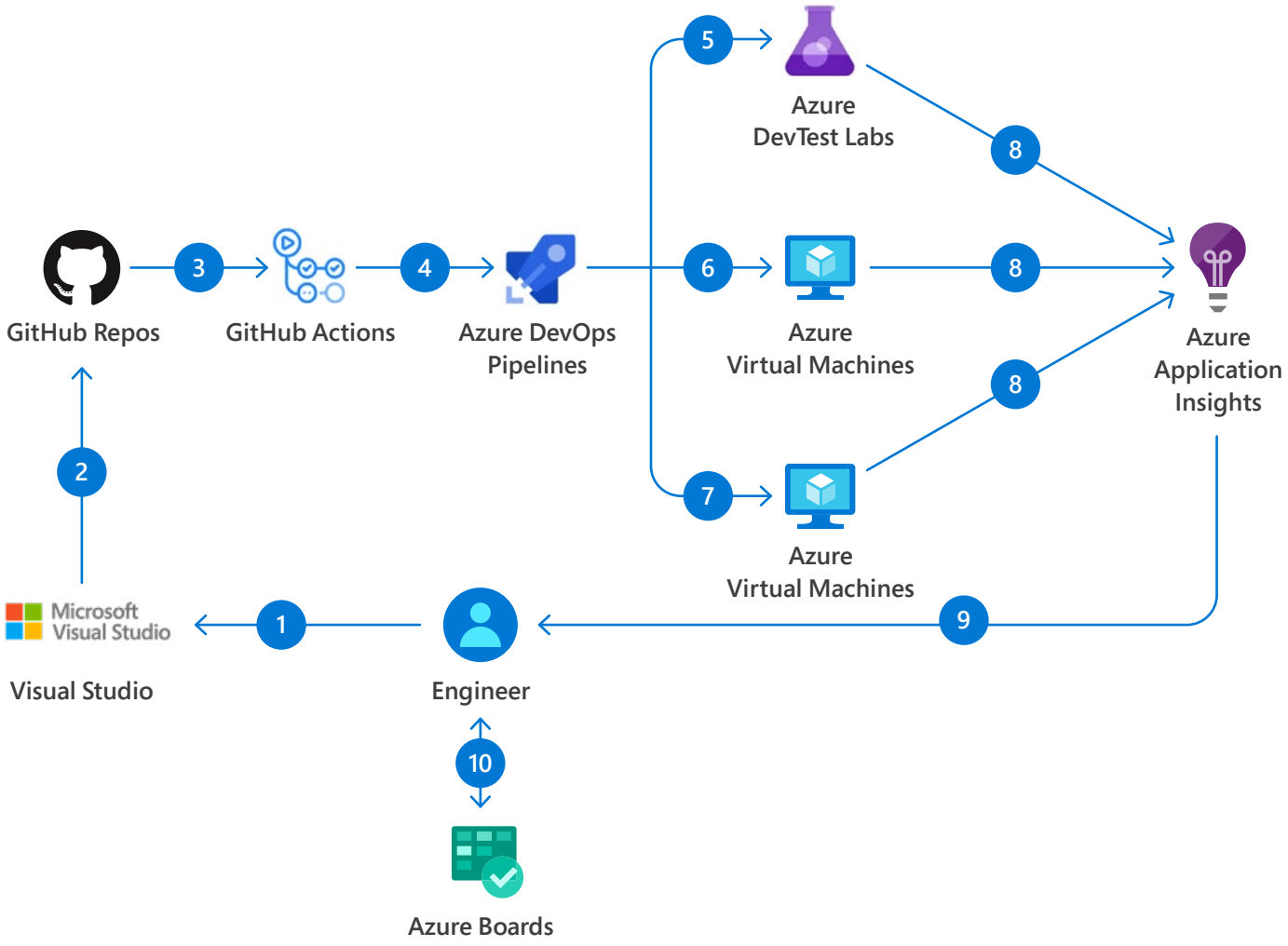
To realize continuous compliancy, your organization should follow six distinct steps toward policy automation. Implementing these six steps, in turn, will also aid in realizing closed loop policy automation.

[1]Microsoft, Enterprise DevOps 2020-2021 Report, 2020

Six steps to successful policy automation

How do you continuously apply compliance standards throughout the DevOps cycle? Let's not forget that, in some cases, a system life can extend for many years. This length of time makes policy non-conformance inevitable, often due to the evolving regulatory landscape. However, you can ensure that all workloads and applications remain compliant by leveraging cloud providers' policy best practices to act as guardrails. These policies are often managed by the security or foundational team and should be automated.

In the example shown here, virtual machines (VMs) are deployed onto Azure and a telemetry-based feedback loop that links back to the development team. In this scenario, it was defined that Azure VMs could not have a public IP. However, this requirement would go unnoticed until deployment occurs in steps 5, 6, and 7. Prior to those steps, the engineering team did not receive updates on these policies. This opens the enterprise to vulnerabilities through outdated policy. For any update, the team would need to review the updated cloud platform policies and ensure all their code conformed.



Policy automation builds on the argument in Tip 2 (Integrate security in the early stages of the development lifecycle) to illustrate why shifting security left is critical for every enterprise. Taking Microsoft Azure as our reference, here are six steps for your enterprise to realize continuous compliance with policy automation:

1. **Determine your policy set**
2. **Adopt a policy-as-code model**
3. **Update policies in code and push to Azure**
4. **Close the loop with compliance scanning**
5. **Shift left using a quality gate**
6. **Use Azure Security Center to monitor and observe**

Step 1: Determine your policy set

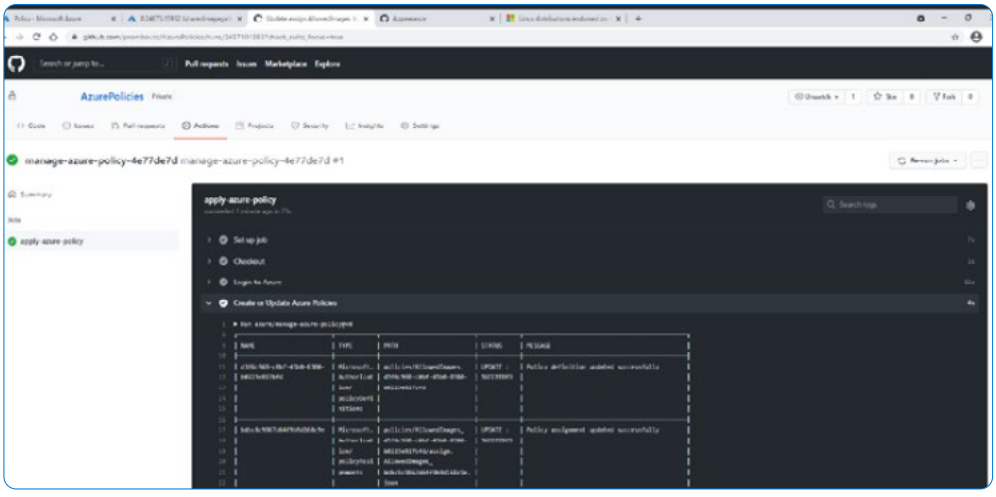
Step one in the process is to determine what policies you want to use. Your industry sector (e.g. banking, healthcare, etc.) will have specific compliance requirements to adhere to. In Azure, you can select a sample template to get started, like CIS, ISO27001, and PCI-DSS. It's easy to set up these baselines in Azure very quickly using Azure Blueprints.

Step 2: Adopt a policy-as-code model

The second step is to make sure your policies are stored in a code repository. Using a repository like GitHub allows you to export these policies and ensure they are version controlled. This also sets the operating state for making future policy updates and pushing changes back to Azure via code.

Step 3: Update policies in code and push to Azure

After the policies are version controlled and updated in GitHub, you can push changes back to Azure. You can configure this action to automatically trigger when policies are updated. This makes tracking changes easy with GitHub's versioning control.



Step 4: Close the loop with compliance scanning

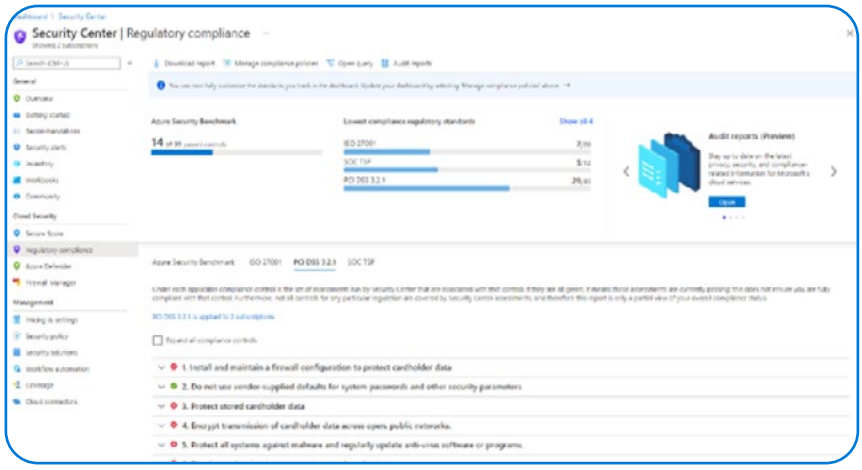
With so many teams across the enterprise, cloud services are constantly spun up, elevating the possibility that the business becomes non-complaint. With your policies as code, you can now use GitHub to run on-demand evaluation scans against your Azure Cloud environments. This closes the loop, outlining your current state while empowering your teams with actionable steps to restore compliancy as necessary.

Step 5: Shift left using a quality gate

Now that your policies are created and maintained via code itself, teams across the enterprise can understand, validate, and test policies against code even earlier in the software lifecycle. Helpful tools like Open Policy Agent act as a quality gate in your Kubernetes (k8s) automated deployment system and test Terraform code in your repository before deploying to any environment. This means your teams can check policies even before a pull request is closed and subsequently remediate at the earliest stage.

Step 6: Use Azure Security Center to monitor and observe

The last step is monitoring and observing your enterprise's compliance state using Azure Security Center. The figure below shows an Azure Security Center compliance check summary—giving real-time compliance telemetry. This is extremely helpful in the event an auditor wants to review the current state of a subscription. This summary also offers remediation tasks for Azure Security Center to resolve any issues.



The end goal of these six steps is enabling closed loop policy automation. When set up, compliancy is continuously monitored and aligned with all changes, at any time.

TIP 6

Secure and visualize your software supply chain

Understanding your software systems dependencies stemming from open source and third-party platforms, frameworks, and components further secures your software supply chain

Proprietary code isn't the only source of modern security vulnerabilities. Often the bigger danger lies in dependencies—any code referenced and bundled to make a software package work. In modern software, 80% or more[1] of most applications' code comes from dependencies. These dependencies themselves rely on dependencies, which results in a complex relationship diagram. The result? A dependency tree of complex and dynamic relations within software that poses a significant security concern for enterprises. Not understanding a system's complex and full dependency tree provides a pathway for malicious actors to attack your systems.

So, how can you protect your software supply chain?

Step one is establishing transparency within each component's update history, including: the releases, the quality checks completed, versions, and documentation. This helps establish something akin to a chain of custody on your code, components, and subsequent dependencies. But this is only a start.

To truly protect your software supply chain your enterprise will need to start (1) acting on insights gathered by dependency tree visualization, and (2) growing transparency with a software bill of material.

[1]GitHub, Octoverse Security Report, 2020

Act on insights gathered by dependency tree visualization

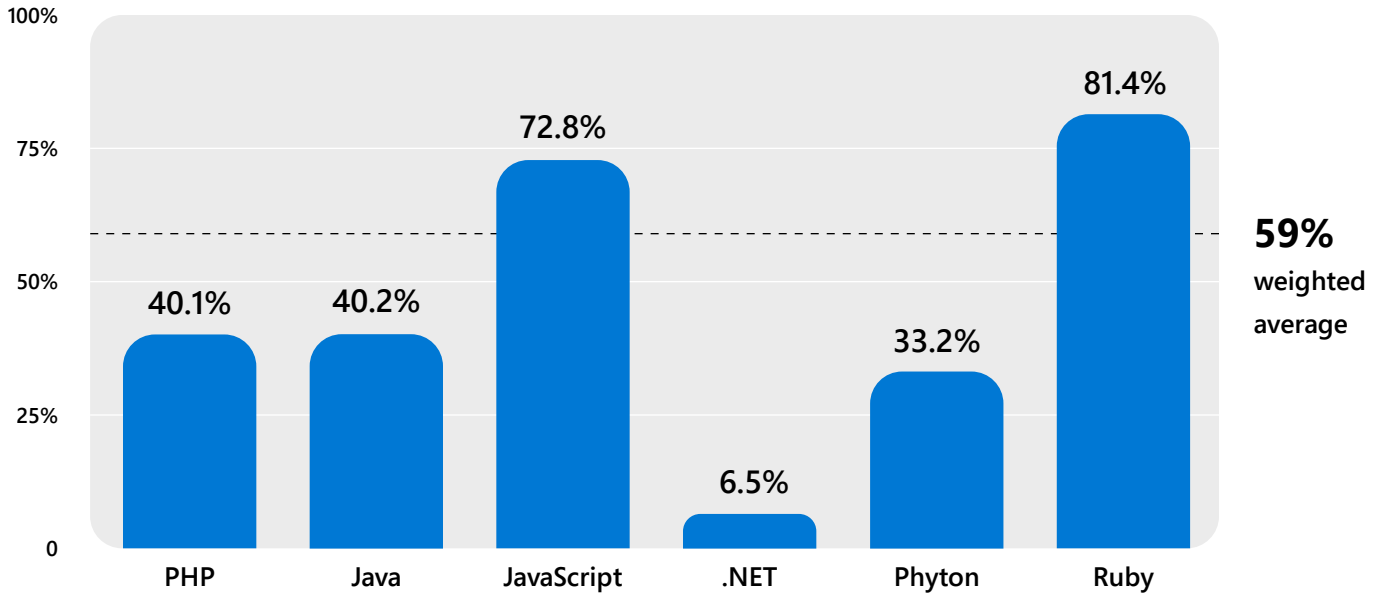
Just visualizing dependencies alone does not offer your enterprise heightened security. While it can offer great insights to your organization, the key step is acting on those insights to keep your systems out of dependency hell. Using these insights to practice proper dependency management is the next logical step.

When practicing dependency management, consider activities like scanning dependencies for vulnerabilities, updating dependencies, and taking control of dependency usage—limiting a potentially long list of interdependent components. These are often called conflicting, circular, and diamond dependencies, and all result in hard to update and insecure systems.

Dependabot from GitHub helps drive proper dependency management by visualizing the dependencies of any components within a system—including all connected dependencies. But the best part is how proactive the tool can be. Dependabot will notify your team when it locates a known vulnerability or when updates become available for a specific dependency. It even aids remediation practices by preparing and suggesting necessary changes for updates in the codebase.

While powerful tools like Dependabot help solve dependency management, maintaining a clear set of policies remains paramount. It's a must for teams to form a clear policy for embracing dependencies, managing updates, and locating possible vulnerabilities. Without consistent policy adoption across teams, an organization's dependency management funnels into chaos and security issues.

Percentage of active repositories that received Dependabot alerts[1]



[1]GitHub, Octoverse Security Report, 2020

Grow transparency with a Software Bill of Material (SBOM)

Transparency brings trust to systems. Teams who are intimately familiar with the software modules they rely on, develop best practices for updates and understand the impact one module can have on their system and whole delivery lifecycle.

An important practice to consider when building transparency is creating a SBOM. Similar to how a manufacturing bill of material details a product's construction, the same counts for a SBOM. A SBOM describes the software module construction and provides insight for security management. Further than just hardening security, a SBOM can also lower license and compliance risks.

SBOM standardization is underway. Multiple initiatives across the security industry are attempting to solidify the model for what a standardized, machine-readable SBOM looks like. For now, keeping an updated list of components, all accompanying version and update strategies, along with known vulnerabilities and maintainers is a good start. What's more, a thorough SBOM is seen as a quality indicator for a software product.

Similar to how a manufacturing bill of material details a product's construction, the same counts for a SBOM. A SBOM describes the software module construction and provides insight for security management.

Closing thoughts

By integrating security into your DevOps practices, you're committing to the creation and management of a DevSecOps team. Just as DevOps in the past blended operations and developer teams and practices, the introduction of security to these teams requires patience. Large-scale changes often impact teams' abilities to meet deadlines, so employees will resist DevSecOps adoption if the preparation isn't methodical. Successful DevSecOps adoption requires a thoughtful, intentional blend of cross-team collaboration, a security-first culture, and cutting-edge technology.

It's tempting to adopt powerful technology without first considering company culture or existing processes. This is a mistake. Building a security-first culture is paramount to DevSecOps resonating with your teams. Consider creating an InnerSource community or adopting a Security Champion Model to spur collaboration across teams and to spread cross-team knowledge.

After preparing your organization for a cultural shift, the next challenge is harnessing cutting-edge technology effectively. With a wealth of options, selecting the perfect solution is tough. Start by selecting easily adoptable tools and technology. Focus on technology that delivers value either through improving observability, strengthening security at early stages in application development, or proactively providing remediation fixes.

Successful DevSecOps adoption requires a thoughtful, intentional blend of cross-team collaboration, a security-first culture, and cutting-edge technology.

Creating and managing a DevSecOps team is always distinct to your organization. Not all recommendations will directly apply. It's alright to explore unique ways to complement existing business processes, culture, and people with security. Use these tips as a guide to equip the newly formed DevSecOps team with all the information and resources needed to navigate the change.

Embracing DevSecOps is a software delivery advantage. Remove bottlenecks clogging your delivery pipeline and provide the necessary controls for compliance and security. By uncovering vulnerabilities earlier, your teams save time remediating issues and realizing compliancy, while also minimizing any associated costs. Return to what's important: propelling innovation with efficient and secure software delivery.

By uncovering vulnerabilities earlier, your teams save time remediating issues and realizing compliancy, while also minimizing any associated costs.

How Microsoft & Sogeti can help

DevSecOps teams succeed with cross-team collaboration, a focus on developer velocity, and cutting-edge tooling. Microsoft offers learning resources, products, and services to position all DevSecOps teams for innovation, regardless of language, framework, or cloud.

Azure expert managed service provider, Sogeti, continuously improves business-focused digital delivery for DevSecOps teams utilizing cloud and DevOps Centers around the globe. Sogeti uses its DevSecOps Adoption Framework and CloudBoost library to drive continuous improvement and InnerSource adoption within DevSecOps teams—leveraging the full platform capabilities of Azure for governance, security, and compliance as a cloud-native foundation.

Share this

Inspire other technical leaders to integrate security into their DevOps practices by sharing this whitepaper on social media or email.

Resources

Microsoft DevSecOps solution: Integrate security into every aspect of the software delivery lifecycle. Learn about how Microsoft offers a complete solution to enable DevSecOps, or secure DevOps, for apps on the cloud (and anywhere) with Azure and GitHub.

GitHub: Secure at every step. GitHub helps enterprises stay ahead of security issues, leverage the security community's expertise, and use open source securely.

Microsoft Azure: Security is integrated into every aspect of Azure. Strengthen your security posture with Azure. Azure offers you unique security advantages derived from global security intelligence, sophisticated customer-facing controls, and a secure hardened infrastructure.

Documentation: Learn how Azure security helps to protect your applications and data, support your compliance efforts, and provide cost-effective security for organizations of all sizes.

Talk to our sales team: Talk to our specialists to see how Microsoft can help your DevSecOps team.

Sogeti: We're working with Microsoft to ensure our clients create value from new DevOps tooling, approaches, and cloud capabilities.

Sogeti DevSecOps Adoption Framework: Release faster with a DevSecOps Enterprise Reference Architecture, product descriptions, and DevSecOps Blueprints.

Sogeti CloudBoost library: Automate the environment provisioning on Azure cloud platform with our repository of reusable templates and scripts.

Sogeti OneNative services: Enable dynamic public cloud development from design, to build, to run.

Authors

Samit Jhaveri is the Director of Product Marketing with Microsoft Azure focused on cloud application development and DevOps with GitHub. He serves as the business leader working across product management, sales leadership & finance with responsibility for defining and executing the e2e go-to-market strategy including pricing & offers and execution plans such as campaigns and field & partner motions for growing the business. Prior to the current role, Samit led an engineering team at Microsoft's Server and Tool Division and was responsible for shipping several B2B solutions for different vertical industries. Samit earned his MBA from the University of Washington and Masters in Management Information Systems from the University of Arizona.

Clemens Reijnen is Sogeti's Global CTO Cloud Services and DevOps leader. He has been awarded the Microsoft Most Professional Award for 10 years in a row and is a SogetiLabs Technical Fellow. He co-authored the book 'Enterprise DevOps Report 2020-2021' with Microsoft and writes frequently on cloud and DevOps on Sogeti.com. As a global DevOps leader, he works closely with Sogeti's large enterprise customers to ensure their cloud adoption and Enterprise DevOps transformation programs create value for the business.

Sogeti co-authors:
André Andersen, Gwendal Jabot, Laurent Grangeau, Matt Braafhart, Olivier Dupré, Peter Rombouts, Rahul Sharma, Sandra Parlant, & Tony Jarriault.