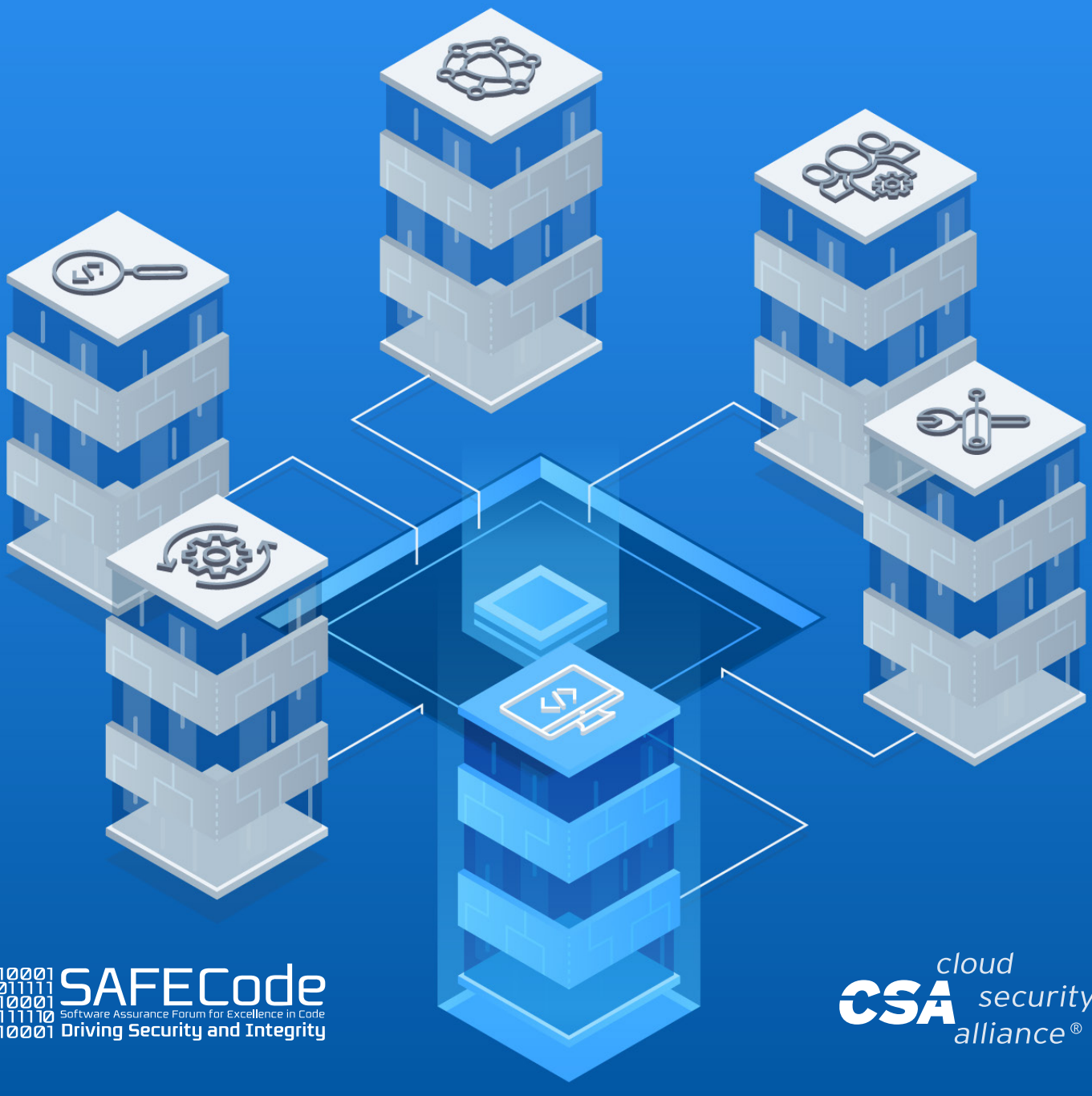
The Six Pillars of DevSecOps:

Bridging Compliance and Development



10001
011111
10001
111110
10001 SAFECode
Software Assurance Forum for Excellence in Code
Driving Security and Integrity

CSA cloud security alliance®

The permanent and official location for DevSecOps Working Group is https://cloudsecurityalliance.org/research/working-groups/devsecops/.

© 2022 Cloud Security Alliance – All Rights Reserved. You may download, store, display on your computer, view, print, and link to the Cloud Security Alliance at https://cloudsecurityalliance.org subject to the following: (a) the draft may be used solely for your personal, informational, non-commercial use; (b) the draft may not be modified or altered in any way; (c) the draft may not be redistributed; and (d) the trademark, copyright or other notices may not be removed. You may quote portions of the draft as permitted by the Fair Use provisions of the United States Copyright Act, provided that you attribute the portions to the Cloud Security Alliance.

Acknowledgments

Lead Authors:

Roupe Sahans

Contributors:

Ashleigh Buckingham
Chris Hughes
Michael Roza

Reviewers:

Ashish Vashishtha
Eric Gauthier
Altaz Valani

CSA Analysts:

Josh Buker
Hillary Baron

Special Thanks:

Bowen Close

The DevSecOps (DSO) Working Group is a Cloud Security Alliance (CSA) Working Group whose research is dedicated to defining best practices and providing guidance and playbooks to help teams implement security into their DevOps process.

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Table of Contents

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Foreword

The Cloud Security Alliance and SAFECode are both deeply committed to improving software security outcomes. The paper "Six Pillars of DevSecOps,"[1] published in August 2019, provides a high-level set of methods and solutions successfully implemented by its authors to build software at speed with minimal security-related bugs. Those six pillars are:

Pillar 1: "Collective Responsibility" *(Published 02/20/2020)*[2]
Pillar 2: "Training and Process Integration"
Pillar 3: "Pragmatic Implementation"
Pillar 4: "Bridging Compliance and Development" *(Published 2/3/2022)*
Pillar 5: "Automation" *(Published 07/06/2020)*[3]
Pillar 6: "Measure, Monitor, Report, and Action"

The successful solutions that underpin each of these pillars are the subjects of a much more detailed set of joint publications by the Cloud Security Alliance and SAFECode. This paper is the third of the six follow-up publications.

1 Cloud Security Alliance. (2019, August 7). *Six Pillars of DevSecOps*. https://cloudsecurityalliance.org/artifacts/six-pillars-of-devsecops/
2 Cloud Security Alliance. (2021a, February 21). *The Six Pillars of DevSecOps: Collective Responsibility*. https://cloudsecurityalliance.org/artifacts/devsecops-collective-responsibility/
3 Cloud Security Alliance. (2020b, July 6). *The Six Pillars of DevSecOps: Automation*. https://cloudsecurityalliance.org/artifacts/devsecops-automation/

© Copyright 2022, Cloud Security Alliance. All rights reserved.

1. Introduction

Given the rapid evolution of software development paradigms and practices, it has become a challenge to align monolithic security compliance activities with software development. Compliance teams have a vested interest in proving that a process and controls are in place. However, most DevOps aligned engineers believe that the proof is in the code, not in the process or in its documentation.

The DevSecOps practice is designed to bridge compliance and development, which requires a collaborative (collective responsibility[4]) effort between security teams and software developers. The goal of compliance in a DevSecOps model is to improve the overall security of an application or environment while reducing the effort taken to validate compliance with security objectives.

The methods explored in this paper allow DevSecOps teams to translate security and compliance requirements into the development cycle such that they are:

- Actionable for software developers.
- Objectively measurable.
- Pragmatic to the reduction of risks.

This paper also explores the requirements, methods, and recommendations for systematic collaboration to security and development teams and is sectioned into three parts, illustrated in Figure 1. Compliance and Development functions should consider the following components:

- **Assessment:** An approach to compartmentalize and assess with minimal operating impact.
- **Mindset:** The shift in thought and practice for how compliance can be designed and implemented into applications.
- **Tooling:** The different practices in security tooling that can provide assurance to compliance requirements.

The reference to stakeholders in this paper and in Figure 1 recognizes two roles, "Compliance" and "Development":

- **"Compliance"** is identified as the management of regulatory or industry compliance standards that are cascaded to teams like information security — as well as legal, risk and audit — to form requirements and policies that shape an organization's business operations. Whereas,
- **"Development"** is engineers and members of a product team (including developers, platform engineers, architects, and business analysts) that have an influence on the design, configuration, and build of applications.

4 Cloud Security Alliance. (2021a, February 21). *The Six Pillars of DevSecOps: Collective Responsibility*. https://cloudsecurityalliance.org/artifacts/devsecops-collective-responsibility/

© Copyright 2022, Cloud Security Alliance. All rights reserved.

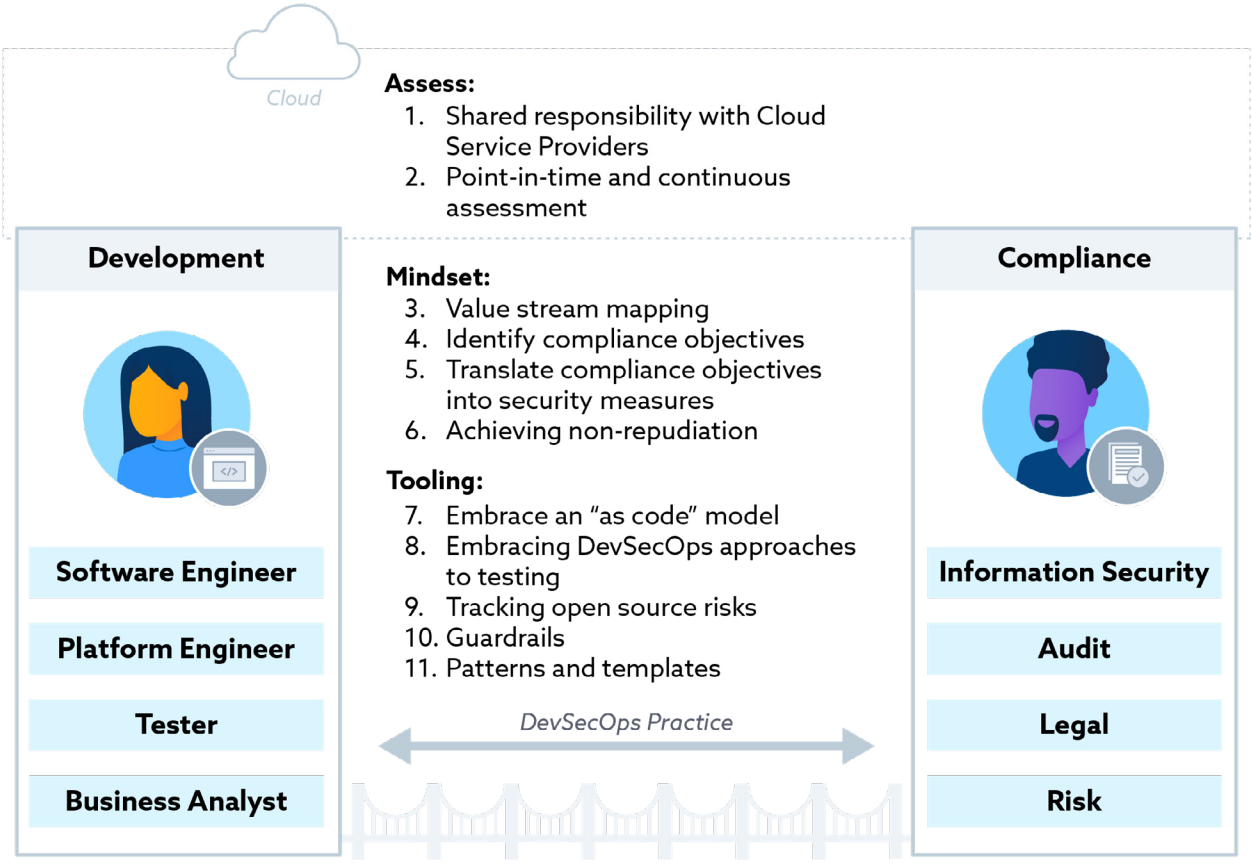


Figure 1. Framework for Bridging Compliance and Development

1.1 Goal

This document provides guidance to ensure that the gap between compliance and development is addressed by recognizing compliance objectives, translating them to appropriate security measures, and identifying inflection points within the software development lifecycle where these controls can be embedded, automated, measured, and tested in a transparent and easily understood way.

This paper does not identify the source of compliance objectives and is not a comparison of standards or guidelines across DevSecOps.

1.2 Audience

The target audience of this document includes those involved in the management and operational functions of risk, information security, and information technology. This consists of the CISO, CIO, and especially the individuals involved in the following functional areas: automation, DevOps, quality assurance, information security, governance, risk management, internal audit, and compliance.

© Copyright 2022, Cloud Security Alliance. All rights reserved.

2. Assessment

Assessments are typically the first port of call for gauging maturity and effectiveness in DevSecOps process and control. The key considerations for organizations to assess their applications and use of DevSecOps are:

- **Shared responsibility with cloud service providers:** Determining where risk is transferred and where the cloud customer should apply controls or seek assurances.
- **Point-in-time and continuous assessment:** The methods by which assessments are appropriate and how continuous review can be automated to improve the accuracy of findings.

2.1 Shared Responsibility with Cloud Service Providers

The design and operation of cloud environments for DevOps and Site Reliability Engineering are critical in most, if not all, uses of application deployment. Cloud security is dependent on the hosting infrastructure. As a result, cloud security controls and the management of shared responsibility are imperative in DevSecOps practice.

When an organization maps compliance goals to security requirements, it is critical to understand the cloud customer's responsibility given their choice of solutions and technologies. Security tools and solutions must align with technologies (e.g., containerized workloads, virtual machines, configuration state of cloud-native platform service).

The Cloud Service Provider (CSP) and Cloud Service Customer (CSC) should agree and document shared responsibilities in a Service Level Agreement (SLA), which often results in the CSC accepting the CSP's services' terms, benefits, and pitfalls. CSPs typically provide this information publicly. The generic model in Figure 2 demonstrates the range of divided responsibilities between the CSC and CSP from on-premise to SaaS.

© Copyright 2022, Cloud Security Alliance. All rights reserved.

	On-Premises/ Private Cloud	Public Cloud IaaS	Public Cloud PaaS	Public Cloud SaaS
Data	●	●	●	●
Applications	●	●	●	●
Runtime	●	●	●	●
Middle Ware	●	●	●	●
Operating System	●	●	●	●
Virtual Network	●	●	●	●
Hypervisor	●	●	●	●
Servers	●	●	●	●
Storage	●	●	●	●
Physical Network	●	●	●	●

● Cloud Customer is responsible ● Cloud Service Provider (CSP) is responsible

Figure 2. Shared Responsibility in Hybrid Clouds[5]

DevSecOps activities should reflect the shared responsibility model in Figure 2. For example, a CSP offering an IaaS service may secure the lower layers (hypervisor and below) of the stack, leaving the CSC with the upper layers (e.g., network and above) to secure.

Once organizations are able to identify their consumed services and differing cloud deployment models, they can start considering "next steps" for security activities.

2.1.1 Next Steps

In most cases, the cloud customer is responsible for application security controls and the management of cloud environments. Development teams should disseminate their requirements for an application that consumes cloud services — CSP reliant controls should be identified, and CSC requirements can be assessed and packaged as remediation activities / sprints.

Where responsibilities belong to the CSP, the CSC should obtain third-party assurances from its CSPs, by either sourcing SOC 2[6] reports or CSA STAR assurances. Cloud customers should review assurances provided by CSPs to determine the appropriateness of controls and processes and implement compensating controls where required.

---

5 Cloud Security Alliance. (2020c, July 13). *Hybrid Cloud and Its Associated Risks*. https://cloudsecurityalliance.org/artifacts/hybrid-clouds-and-its-associated-risks/

6 American Institute of CPAs. (n.d.-b). *SOC for Service Organizations*. AICPA. Retrieved December 22, 2021, from https://www.aicpa.org/interestareas/frc/assuranceadvisoryservices/socforserviceorganizations.html

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Security and compliance functions should work with development teams to accurately identify responsibilities against cloud deployment models. Security controls and processes should be reviewed frequently by security and compliance functions against existing organizational policies and security frameworks, checklists, and patterns. Where responsibility belongs to the CSP, the compliance team can review SOC 2 reports or the CSA STAR to determine appropriate controls by the respective CSP.

2.2 Point-in-Time and Continuous Assessments

Assessments are universal approaches across all organizations for understanding current state maturity and effectiveness for controls and processes. However, with the speed of change and deployments in Agile/DevOps methodologies, questions are raised about the frequency and relevance of assessments.

Point-in-time assessments[7] can be slow with lead times of weeks, if not months. The development and deployment environment is likely to change from the start of an assessment to the finalization of a report. At the same time, point-in-time assessments can provide a holistic view of the security posture of an application and its hosting infrastructure and can be coupled with additional activities like penetration tests and privacy data reviews.

The introduction of security developer tooling for applications and infrastructure can help compartmentalize components for review individually. Findings can be identified and remediated during development and deployment which removes the reliance on point-in-time reviews. The following uses of tooling are examples of how compartmentalizing and scanning an application can provide opportunities for continuous assessment:

- **Software composition analysis:** An opportunity to identify, report, remediate, and standardize the use of high-risk open-source tools. This can be applied and addressed at the build stage and made compliant prior to a point-in-time assessment.
- **Static analysis:** An opportunity to scan source code for weaknesses and vulnerabilities. This will assist developers to write secure code during the build stage and remediate issues from scans from larger code bases.
- **Infrastructure as code (IaC) analysis:** An opportunity to scan coded cloud builds against cloud compliance standards. This helps engineers address security on the cloud at the build stage rather than relying on a point-in-time assessment.
- **Cloud posture management:** A common method of continuously scanning cloud builds and environments holistically against industry standards and guidelines. Cloud posture management can replace or even provide an opportunity to perform point-in-time assessments.

Striking the right balance between scanning at build and performing holistic assessments is reliant on the perceived return of investment — i.e., how can teams perform fast complete scans without adversely impacting deployment speeds.

7 SecurityScorecard. (2018, February 15). *Limitations of Point-in-Time IT Security Risk Assessments.* https://securityscorecard.com/blog/limitations-of-it-security-risk-assessment

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Code scanning (e.g., IaC analysis) is effective for reviewing an entity (application / platform / infrastructure) during the build stage and can in turn reduce the likelihood of vulnerabilities and security weaknesses being written into code. This leads to organizations shifting security left and reducing the work for post-deployment security remediation. However, it needs to be understood that scanning at build does not always yield a complete and holistic set of findings when compared to post deployment scanning (e.g., Cloud posture management).

An example of code scanning completeness is Bridgecrew's (an IaC vendor) research on its open-source IaC analysis tool, "Checkov." Bridgecrew disclosed that only approximately half of industry cloud security benchmark controls — recommended by Centre for Internet Security (CIS) — are addressed at IaC analysis (see Figure 3). This is not a reflection on the quality "Checkov," but rather a reflection on the scale of security issues and completeness of assessment during the build stage across major CSPs.

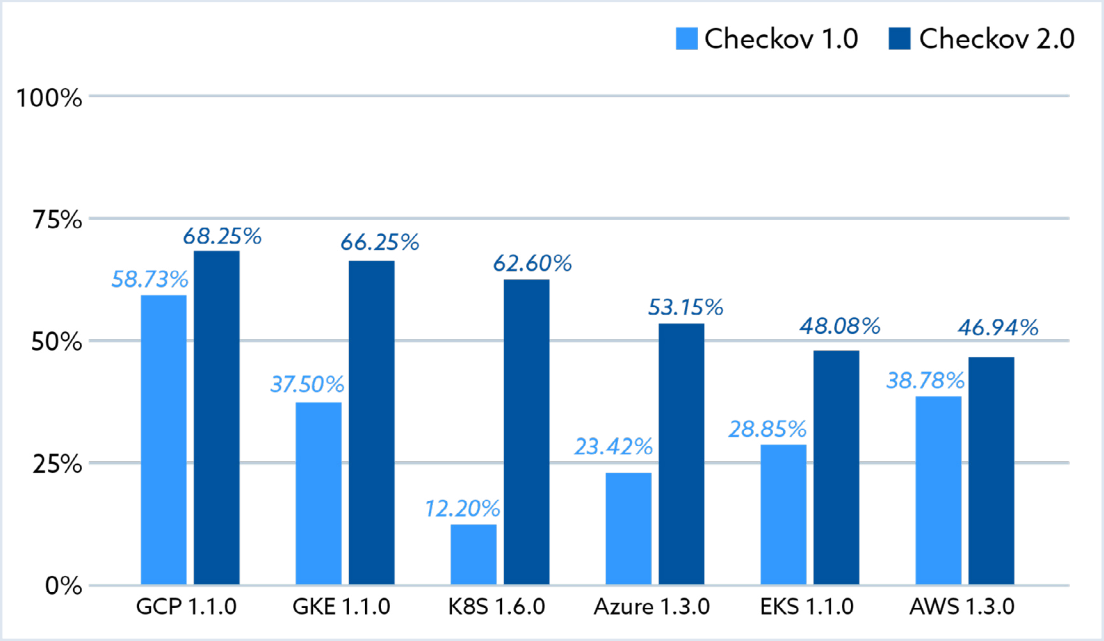


Figure 3. Checkov CIS Benchmark Coverage[8]

The compartmentalization of code scanning does not replace the requirement for point-in-time assessments; however, it does reduce the reliance on them. Organizations can now address issues at build, rather than being adversely impacted by stark security findings from assessments prior to deploying to production.

As organizations apply methods to perform continuous DevSecOps assessments, they can leverage point-in-time assessments as complete, consolidated, and fully integrated reviews as assurance activities with the knowledge that a significant number of security issues have been addressed during build.

8 Johnson, M. (2021, April 8). *Checkov 2.0: Deeper, broader, and faster IaC scanning.* Bridgecrew. https://bridgecrew.io/blog/checkov-2-0-release/

© Copyright 2022, Cloud Security Alliance. All rights reserved.

3 Mindset

Organizations can be quick to address risk directly through tooling, but they can easily overlook the importance of the appropriate mindset in DevSecOps transformation. Mindset is the approach security and product stakeholders (e.g., software developers) take as a set of activities that can often bring two isolated teams closer together. The key considerations that can help bridge conflicting values of security and software development are:

- **Value stream mapping:** Identifying teams, lead times, and process times to understand how work flows from idea to customer outcome. This provides an opportunity to identify security involvement through manual and automated activities.
- **Translating compliance objectives into security measures:** How objectives can be packaged into security measures for developers to consume.
- **Monitoring at the pipeline:** Methods of monitoring and maintaining control of developer activities while not hindering productivity.

3.1 Compliance Using Value Stream Mapping

While compliance and development activities can be aligned in their security aims, the way in which compliance activities are traditionally conducted — lengthy point-in-time reviews — is heavily reliant on documentation. The traditional approach is not well matched to the fast-moving continuous release environment associated with development, agile working, and the continuous delivery nature of DevSecOps.

Therefore, understanding how work moves through the application development lifecycle is important before thinking about applying security controls. Value stream mapping (VSM) is a useful technique to understand the context of the application. As illustrated in Figure 4, a VSM can include the following components:

- **Stakeholders:** The accountable and responsible teams and individuals.
- **Activity:** The generic activity performed during the phase.
- **Lead time:** The time from the point a process accepts a piece of work to the point it hands that work off to the next downstream process.
- **Process time:** The time it takes to complete a single item of work if the person performing it has all the necessary information and resources to complete it and can work uninterrupted.
- **Percent complete and accurate (%C/A):** The proportion of times that a process receives something from an upstream process that it can use without requiring rework.

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Figure 4. VSM Example[9]

Through understanding the activities in a VSM for building and deploying an application, organizations can start identifying and planning where security activities and tools can fit. DevSecOps is founded on the concept of embedding security practices throughout the build stage and deployment pipeline (as shown in Figure 5 below) consistently, as a collective shared responsibility.[10]

9 Google. (n.d.). *DevOps process: Visibility of work in the value stream*. Google Cloud. Retrieved December 22, 2021, from https://cloud.google.com/architecture/devops/devops-process-work-visibility-in-value-stream

10 Cloud Security Alliance. (2021a, February 21). *The Six Pillars of DevSecOps: Collective Responsibility*. https://cloudsecurityalliance.org/artifacts/devsecops-collective-responsibility/

© Copyright 2022, Cloud Security Alliance. All rights reserved.

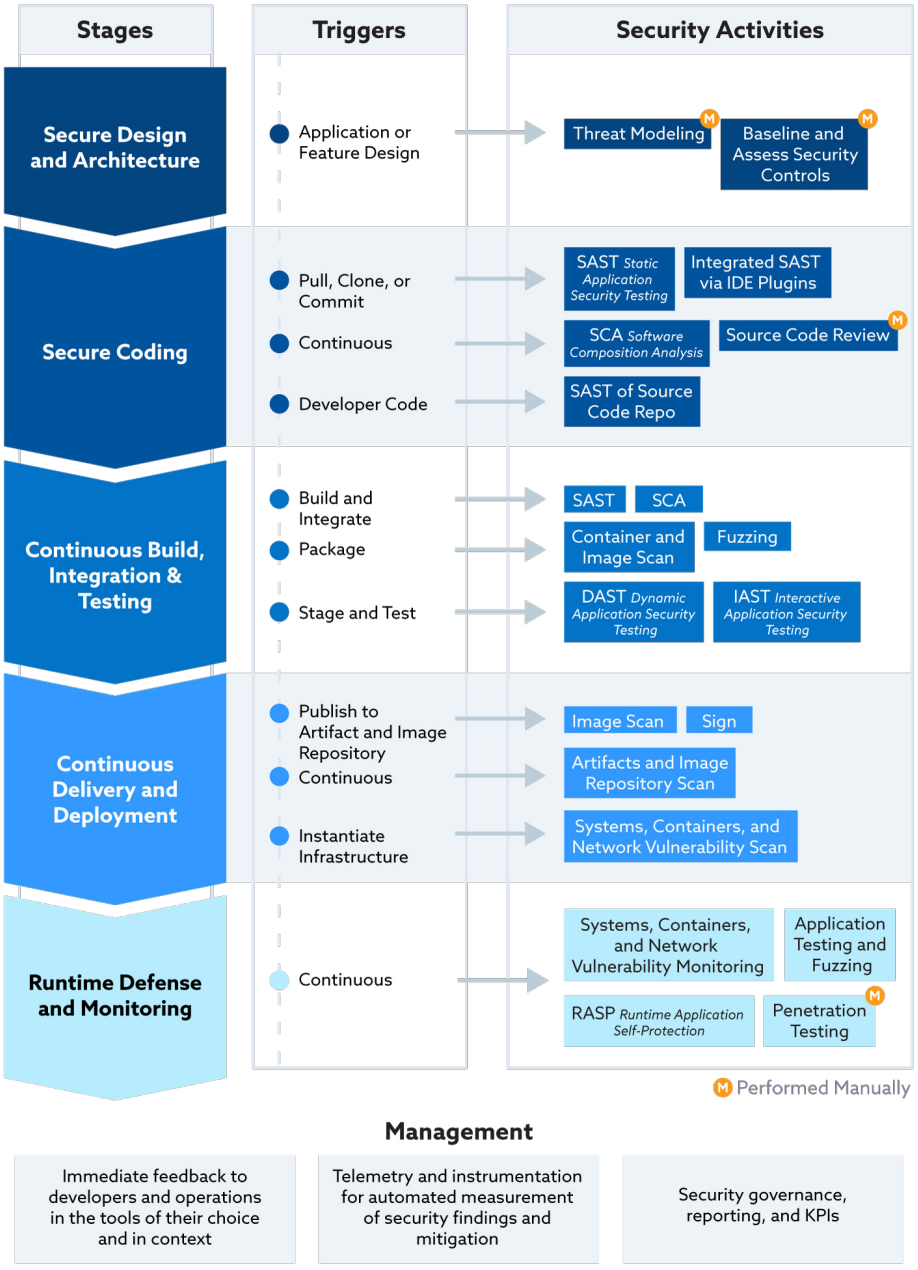


Figure 5. The CSA DevSecOps Delivery Pipeline[11]

Typically security and compliance activities are manual and organized into gates, making the process heavy and potentially impacting delivery timelines. To address this challenge, compliance must be "continuous," such that security is continuously reviewed and integrated within code. The ability to demonstrate compliance to agreed security requirements as part of the delivery pipeline integrates security into the development process as a whole.

11 Cloud Security Alliance. (2020b, July 6). *The Six Pillars of DevSecOps: Automation*. https://cloudsecurityalliance.org/artifacts/devsecops-automation/

© Copyright 2022, Cloud Security Alliance. All rights reserved.

The process of VSM can help achieve this by measuring effort and the operating impact of each security measure. By identifying the end-to-end processes in a VSM, organizations can map areas that require improvement, further control, and automation — including compliance/security requirements.

Once value stream mapping is optimized, organizations can begin to automate compliance using practices like static analysis, policy-as-code, and dependency checking. This paper explores different methods for embedding autonomous security measures into delivery pipelines, which can help bridge compliance and development.

The DevSecOps delivery pipeline itself is the foundation of applying security to products. Organizations should seek to establish practices within the "security activities" section of Figure 5, supporting architectural artifacts (i.e., hardening guides and patterns) continuously to provide a reference point for assessment.

3.2 Translate Compliance Objectives to Security Measures

A goal of DevSecOps is to "shift left" security, i.e., to identify the critical security controls as an integral part of the application development process, rather than the traditional approach of conducting a retrospective audit activity at the point of delivery.

Compliance objectives, policy, processes, and controls typically originate from compliance frameworks, governance, risk, and information security teams, which isn't news to most organizations with mature security functions that align compliance to industry guidelines (e.g., ISO27001[12] or the NIST CSF[13]).

Transitioning from traditional compliance to a more dynamic approach requires breaking down the silos between security compliance and software development teams (including those involved in DevOps and Site Reliability Engineering) to instill a cultural and technical understanding of DevSecOps methodology as a new way of working.[14]

Security and compliance stakeholders (e.g., information security), architecture, and software development teams should work closely together to define clear compliance objectives within the context of the proposed build. There should be precise mapping between the security activities performed and controls achieved against compliance objectives. These should be integrated into the delivery pipeline with automation and transparent reporting in mind.

---

12 International Organization for Standardization. (n.d.). *ISO/IEC 27001 — Information security management*. ISO. Retrieved December 22, 2021, from https://www.iso.org/isoiec-27001-information-security.html

13 National Institute of Standards and Technology. (2021, October 26). *Cybersecurity Framework*. NIST. https://www.nist.gov/cyberframework

14 Cloud Security Alliance. (2021a, February 21). *The Six Pillars of DevSecOps: Collective Responsibility*. https://cloudsecurityalliance.org/artifacts/devsecops-collective-responsibility/

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Transitioning can be achieved by involving security and compliance stakeholders in ongoing project sprint meetings to:

- Plan sessions to identify the necessary tools and security requirements/controls throughout the application development process;
- Provide iterative updates of proposed changes to security/control requirements to help introduce compliance requirements into designing and building applications; and
- Determine whether application features require additional security measures.

Establishing security compliance sprints and user stories for software developers/engineers is an effective method of baking security into software development. Existing security compliance requirements, organizational policies, industry guidelines, and standards can be referenced, which will require collaboration between project stakeholders to materialize. Approaches like INVEST[15] can help support the creation of security user stories and tasks where each should be:

- **Independent:** Should be self-contained;
- **Negotiable:** Should leave space for discussion;
- **Valuable:** Must deliver value to the stakeholders;
- **Estimable:** Should be able to estimate the size;
- **Small:** Should be able to plan, task, and prioritize it; and
- **Testable:** The development should be able to be tested.

Security compliance user stories, sprints, and tasks can better demonstrate to business stakeholders — via internal audit and governance — that code has been designed and configured with organizational-approved security controls.

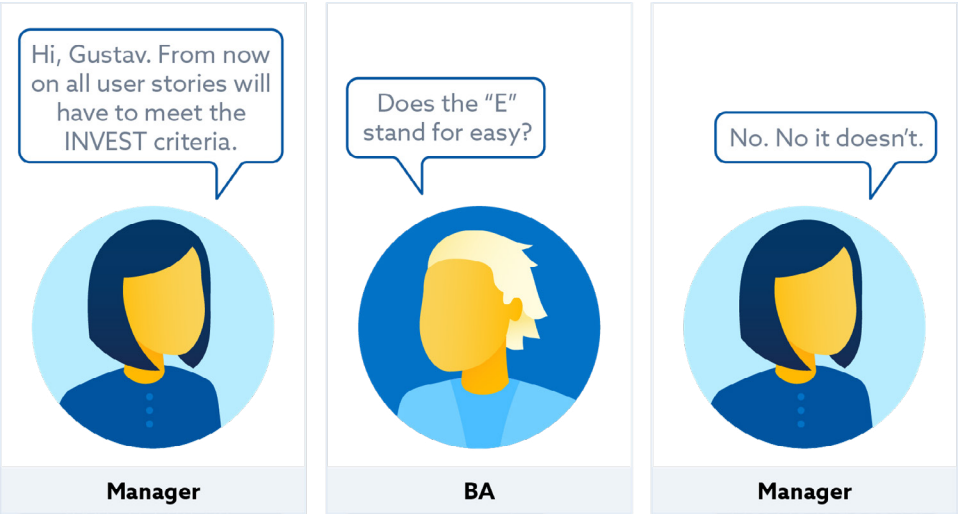


Figure 6. INVEST User Stories[16]

15 Agile Alliance. (2021, July 26). *What does INVEST Stand For?* https://www.agilealliance.org/glossary/invest/

16 S. (2019, June 7). *How INVEST helps team write effective user stories. ...* Medium. https://sabaimam.medium.com/investing-in-user-stories-c7cfb1fc5e85

© Copyright 2022, Cloud Security Alliance. All rights reserved.

The bridge built between compliance and developers should not be one-way. Security policies should be mapped to atomic, technical stories that developers and operations can complete. Software developers, DevOps engineers, and architects should provide feedback on the completion status, so that security compliance teams that write information security policies have visibility to the security posture of the current SDLC stage / sprint. The creation of security patterns and hardening guides will support security compliance activities.

3.2.1 Monitoring at the Pipeline

Groups of software developers can produce copious amounts of code and application changes in a short time and rapidly commit these to production environments. These activities create substantial business advantages to organizations deploying software features frequently and are typically categorized as follows:

- **Continuous integration (CI):** The frequent commitment and merging of new code and code changes to coding repositories that are validated by creating a build and running automated tests against the build.
- **Continuous deployment (CD):** The process of automatically deploying code into an environment which includes automated scanning, testing, and building activities.

Although the autonomy and level of shared responsibility across development teams doing DevOps introduces benefits in quality and speed, the lack of control on roles introduces a security risk. The fast-paced nature of continuous activities increases the risk of human error (e.g., poor coding practice) and nefarious insider attacks (e.g., creating exploitable features), which is why CI/CD pipelines should include a clear and immutable audit trail of all actions.

Audit trails are typically provided natively within the CSP environment. It is critical to evidence the trail of actions when deploying code and ensuring that non-repudiation is embedded within these measures.

Organizations should aim to achieve monitoring of actions on the application and CI/CD pipeline. These can include the following steps/actions:

- At build and integration, gated pull requests should be introduced to prevent unauthorized merges to master branches and commitments to production. Gate controls should not be modified and/or overwritten by users. Deployments should reference only approved master branches.
- Log retention, rotation, archiving, and deletion policies should be agreed upon and implemented. Where possible, logs should be archived in different environments for retention for legal purposes.
- To ease reviewing and managing audit trails, logs should be tagged based on events and centrally aggregated to reduce human error and increase the speed of reviewing logs. Tags are created with relation to modifications, deployments, and the associated environment of the event.
- To uphold log integrity, user accounts accessing log data should be read only and tamper proof, with access granted to personnel on a case-by-case basis. The tampering of logs can prevent nefarious activity from being identified. This can also be achieved using a Write Once Read Many (WORM) compliant storage mechanism, which prevents data from being deleted or overwritten.

© Copyright 2022, Cloud Security Alliance. All rights reserved.

4. Tooling

Tooling is often where value is realized in applying security controls and measures. Tooling can help introduce security checks, scans, and management of data and be automated with triggers at the deployment pipeline — these tools can be proprietary or open source. The key considerations during and prior to tooling procurement are:

- Embracing an "as-Code" model;
- Embracing DevSecOps approaches to testing;
- Tracking open-source risks;
- Guardrails; and
- Patterns and templates.

4.1 Embrace an "as-Code" Model

Code quality doesn't just start and stop with the software developer; tools are available that improve quality and apply policy checks to code in the integrated development environment (IDE), which can be applied to source code and infrastructure as code (IaC).

IaC eliminates the traditional provisioning infrastructure through consoles and manual tasks and instead provides infrastructure through code. IaC is achievable through Cloud Service Provider (CSP) native (e.g., AWS Cloud Formation) tooling or vendor-neutral third-party IaC capabilities (e.g., Chef, Ansible, Terraform). The use of IaC embraces automation, version control, and governance.

Furthermore, when coupled with Compliance-as-Code (CaC) / Policy-as-Code (PaC), organizations can bake their compliance requirements directly into their IaC templates and manifests. These compliance requirements could be from applicable regulatory frameworks, organizational security policies, or a combination of both.

As cloud deployments become automated and flexible, the practice of "Compliance as Code" on IaC for the cloud has become the de facto method for deploying cloud resources, using tools like Terraform, Puppet and Chef.

IaC, as illustrated in Figure 7, is the process of using programmable infrastructure to build cloud and on-premise environments autonomously and at scale. IaC code can be treated like source code, such that it can be version controlled, tested, and managed in repositories.

© Copyright 2022, Cloud Security Alliance. All rights reserved.

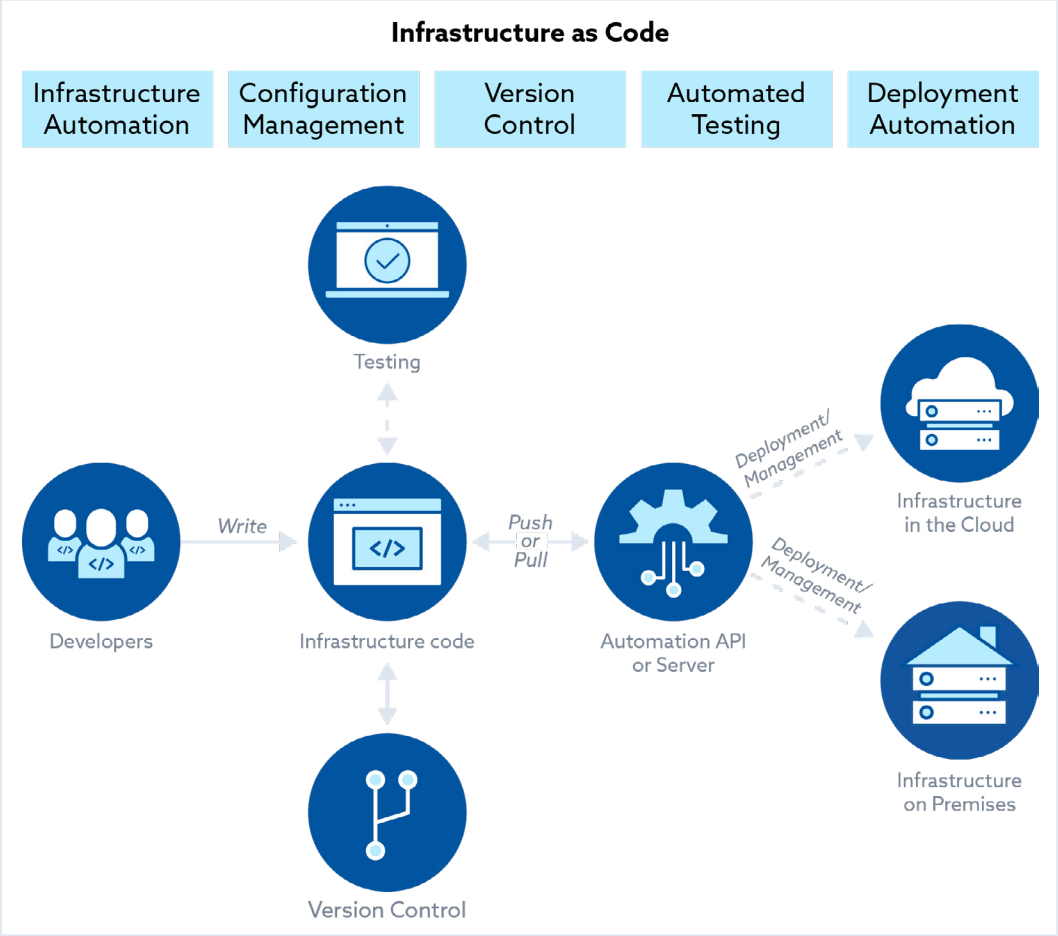


Figure 7. Infrastructure as Code Operational Workflow[17]

Compliance as code can be achieved using tools and plugins for IaC, where cloud resources are configured in scripts and deployed across multiple environments. IaC security plugins can identify the infrastructure (e.g., Chef, Puppet, Terraform, Ansible) and deploy corresponding security analysis to promote correct coding syntax and secure code (e.g., ansible-lint, cfn_nag, cookstyle, Foodcritic, puppet-lint, Checkov and Terrascan).

The Open Policy Agent (OPA) for IaC is used to create and modify cloud environments and has defined a set of policies for how OPA expects infrastructure as code to behave. The policy set in OPA provides opportunities for Software developers, DevOps/DevSecOps engineers, and architects to demonstrate compliance in code on the cloud against documented information security policies (i.e., opportunities to perform automated checks in code as opposed to manual reviews).

17 Velimirovic, A. (2020, September 9). *What Is Infrastructure as Code? Benefits, Best Practices, & Tools*. PhoenixNAP Blog. https://phoenixnap.com/blog/infrastructure-as-code-best-practices-tools

© Copyright 2022, Cloud Security Alliance. All rights reserved.

By building compliance into the DevOps pipeline, product teams can increase flexibility while reducing duplication of effort to provide continuous detection and remediation of compliance violations.[18]

CaC / PaC are also achievable at the source code level via an Integrated Development Environment (IDE). IDEs can achieve this through "linting" — using a static code analysis tool to flag programming errors — using tools like SonarLint, DevSkim, OWASP FindSecurityBugs, and Puma Scan. Although in the IDE, an organization's information security policies does not completely apply to the linting tool, linting does perform a quality review and identifies potential bugs and duplications as developers write code.

IDE security plugins are designed to identify security bugs and fixes during the development of source code, allowing security to shift left to software developers. Figure 8 denotes a gated approach to IDE security controls and how feature branches (duplicate code files) can be established, reviewed, and merged to a master branch for deployment. At this point security bugs in code can be identified earlier in development instead of closer to release windows when they may impact release timelines.

Figure 8. IDE plugins[19]

Embracing PaC / CaC can help introduce compliance from the onset of the development lifecycle, and implementing guardrails ensures that infrastructure and source code are provisioned in a compliant and secure manner. This approach also brings compliance activities closer to the disciplines being implemented by their developer and operations peers, further facilitating the concept of DevSecOps.

4.2 Embracing DevSecOps Approaches to Testing

The continuous nature of DevSecOps — continuous testing — requires testing to be performed in a far more segmented and frequent way than traditional applications. Testing no longer becomes a tick-box exercise to release an application into production.

Testing applies to the capabilities of an application and is often broken down into requirements. The frequency of testing is as important as the frequency of changes and deployments to an application.

18 Cloud Security Alliance. (2020b, July 6). *The Six Pillars of DevSecOps: Automation*. https://cloudsecurityalliance.org/artifacts/devsecops-automation/

19 SonarQube. (n.d.). *SonarLint Integration*. Retrieved December 22, 2021, from https://www.sonarqube.org/sonarlint/

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Code committed to a repository is typically run in a Continuous Integration (CI) tool and subject to automated tests. Having sufficient automated testing decreases the risks associated with deploying new code changes and reduces the likelihood of regressing the code or introducing new defects. The DORA[20] research group's four key metrics highlight that deployment frequency and lead time for changes are correlated with the efficacy of automated test suites.

> - **Deployment Frequency:** How often an organization successfully releases to production
> - **Lead Time for Changes:** The amount of time it takes a commit to get into production
> - **Change Failure Rate:** The percentage of deployments causing a failure in production
> - **Time to Restore Service:** How long it takes an organization to recover from a failure in production

Test automation increases the frequency and reduces the lead time to change, which helps reduce the risk of regression defects and lowers risk by enabling a quicker response to threats linked to attack maps and discovered vulnerabilities.

The approaches and use cases in Table 1 below are typically considered in software development.

Test	Segmentation	Frequency
Security Smoke and Unit Testing	Testing small individual units of source code and integrated compartments of an application as they are developed allowing defects to be found earlier and remediated faster and at less cost. Typically performed as an activity by developers as code is produced before deployment.	Continuous activities for developers to be reviewing and testing code. Requirements, sprints, and tasks should allocate time to perform Security Smoke and Unit Testing.
Static Application Security Testing (SAST)	A method typically supported by a tool and integrated into a build pipeline designed to analyze an application's source code before it's compiled to find security vulnerabilities or weaknesses that may be exploited.	During development via IDE and continuous automated activity at every build or deployment.

20 Portman, D. G. (2020, September 22). *Use Four Keys metrics like change failure rate to measure your DevOps performance*. Google Cloud Blog. https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Test	Segmentation	Frequency
Dynamic Application Security Testing (DAST)	Automation A method typically supported by a tool and integrated into a build pipeline designed to analyze an application at runtime after its source code is compiled to identify security vulnerabilities or weaknesses that may be exploited.	Continuous automated activity at every deployment — typically time-consuming scans that can be performed during off-peak hours. Can be "shifted right" as a non-developer activity and performed as nightly scans after most/all application compartments are integrated.
Interactive Application Security Testing (IAST)	Combines elements of DAST with additional code visibility. It is implemented as an agent within the test runtime environment to test operation of attack and identify vulnerabilities.	Continuous automated activity at every build or deployment as well as during real-time execution of application. Can be "shifted right" as a non-developer activity and performed as nightly scans after most/all application compartments are integrated.
Fault Injection	A method of chaos engineering by subjecting an application to real-world failures and dependency disruptions it could face in production. Fault injection is the introduction of failure into application design to validate its robustness and error handling.	Planned activity (automated and manual) that typically requires cooperation from all application stakeholders. Requirements, sprints, and tasks should be dedicated to Fault Injection.
Penetration Testing	Testing an application either as internal or external, white-box or black-box approach to exploit application weaknesses in configuration and design. Typically testing is performed periodically by a specialist penetration tester at release.	Planned activity on a releasable version of the application. Requirements, sprints, and tasks should be dedicated to penetration tests.

Table 1. Testing Types Segmentation and Testing Frequency

Table 1 demonstrates the ideal testing parameters of an application with activities that are both manual and automated. These should typically be introduced at build and deployment phases and can be mapped and aligned to CSA's DevSecOps delivery pipeline in Figure 5.

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Figure 5. The CSA DevSecOps Delivery Pipeline[21]

21 Cloud Security Alliance. (2020b, July 6). *The Six Pillars of DevSecOps: Automation*. https://cloudsecurityalliance.org/artifacts/devsecops-automation/

© Copyright 2022, Cloud Security Alliance. All rights reserved.

4.3 Tracking Open-Source Risk

Organizations are increasingly adopting open-source code and libraries as part of their environments and software development activities. While the open-source community represents tremendous potential for value and innovation, it also comes with its associated risks and concerns.

Organizations should vet all of the open-source software and components connected to their codebase and environments to ensure they are not introducing unwarranted risks and vulnerabilities. The identification of open-source tools and dependencies tends to be overlooked; applications reliant on unidentified open-source tools will have an unknown measure of supply chain risk as represented in Figure 9.

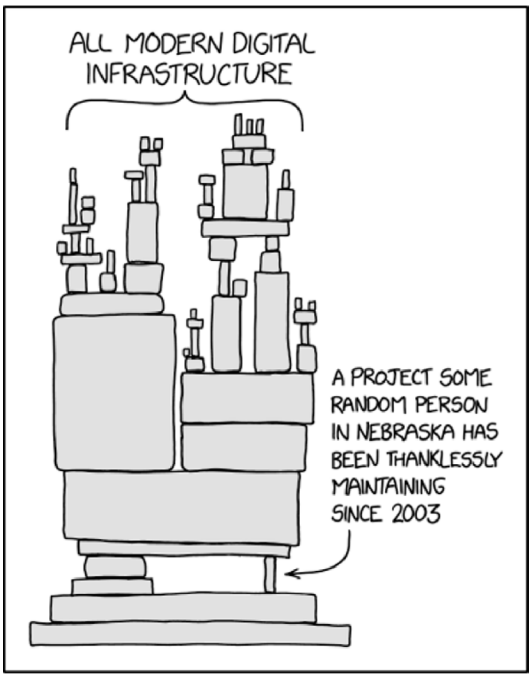


Figure 9. Illustration of Dependency Risk[22]

External components present a risk that should be vetted appropriately, which is achievable through the following pragmatic approach of process and tooling:

- **Process:** The organization should ensure that a genuine need or business case for the open-source library exists. Unnecessary libraries can result in an increased attack surface and potential for exposure. Open-source tooling updates, releases, and contribution frequency are indicators that can measure open-source risk.
- **Tooling:** The organization agent should utilize Software Composition Analysis (SCA) tools for scanning code, projects, and codebases, looking for third-party libraries on the technical front. SCAs can help inventory open-source components used in an application and identify vulnerabilities.

22 Munroe, R. (n.d.). *Dependency*. Xkcd. Retrieved December 22, 2021, from https://xkcd.com/2347/

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Using this approach does not eliminate risk, but it does help limit the attack surfaces given how extensively open-source software and components are used. Figure 10 illustrates at stage 3 where an SCA can fit into AWS infrastructure. Figure 10 also illustrates where an SCA can be placed during the code build / secure coding stage. SCAs are an effective approach of shifting left as an SCA is a security control at the build stage prior to deployment.

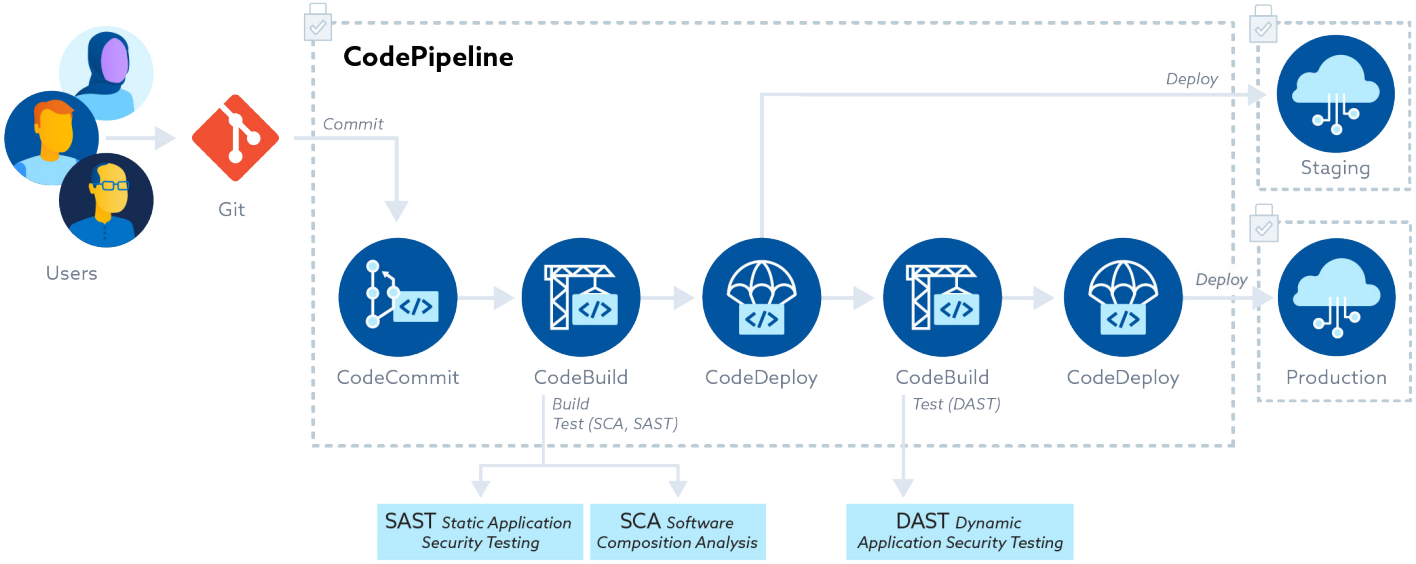


Figure 10. Introducing SCA for Code Builds on the Cloud[23]

4.4 Guardrails

Guardrails are integrated tools and, in the best case, automated in the software development process to ensure alignment with the organization's goals and objectives, including compliance. Guardrails establish a baseline that can constantly monitor deployments. These baselines can be represented as a high-level set of rules with detective and preventive policies applied.

Guardrails may be implemented as a means of compliance reporting (e.g., number of machines running OS in the currently-approved image list) or as an enforced / auto-remediating set of controls (e.g., devices running any OS other than those approved are automatically shut down).

A focus on in-line guardrails and feedback within the DevOps pipeline — achieved through tightly integrated tools and processes — can shift the approach to compliance from point-in-time to continuous. Consideration must be given to ensure that the tools and techniques are implemented to align with the agreed compliance objectives. Evidence must be produced easily to achieve external validation of these controls through internal and external governance.

23 Manepalli, S. (2021, January 21). *Building end-to-end AWS DevSecOps CI/CD pipeline with open source SCA, SAST and DAST tools.* Amazon Web Services. https://aws.amazon.com/blogs/devops/building-end-to-end-aws-devsecops-ci-cd-pipeline-with-open-source-sca-sast-and-dast-tools/

© Copyright 2022, Cloud Security Alliance. All rights reserved.

"The Six Pillars of DevSecOps: Automation" discusses "the implementation of a framework for security automation and programmatic execution and monitoring of security controls to identify, protect, detect, respond, and recover from cyber threats"[24] as it pertains to securing code, applications, and environments. The output of these automated processes should be used to provide the evidence required to meet compliance objectives.

In DevSecOps the deployment pipeline can consume or even build environments that align to compliance objectives. This may be a combination of monitoring and enforcing controls and will likely vary depending on the environment, data classification, and business risk (i.e., development vs. production).

With a target state defined in code, templates, and guardrails, security objectives should be integrated directly into the deployment pipeline where possible. The goal should be an automated audit trail for ease of review, as well as relevant security metrics for continuous review by both the DevOps team and compliance team to drive security decisions. An example of how guardrails can be applied to cloud environments can be seen in Figure 11 below. Cloud constraints are applied (detective and preventative) prior to building and provisioning accounts and environments.

Figure 11. Cisco's Example of AWS Guardrails[25]

24 Cloud Security Alliance. (2020b, July 6). *The Six Pillars of DevSecOps: Automation*. https://cloudsecurityalliance.org/artifacts/devsecops-automation/

25 Ramamoorthy, S. (2018, February 14). *DevSecOps: Security at the Speed of Business*. Cisco Blogs. https://blogs.cisco.com/security/devsecops-security-at-the-speed-of-business

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Guardrails can be tailored to an organization's compliance requirements and risk appetite. The success criteria for guardrails rely on three fundamental principles identified below and supported by an example set of control categories in Table 2. For a lower-level set of cloud security controls, see the Cloud Control Matrix v4.[26] The organization should:

- **Understand the varying degrees of assurance** from informational/feedback, vulnerability reporting, or tracking compliance deviation on infrastructure deployments (e.g., using the approved image for a VM) to enforce/blocking (guardrails blocking unapproved VM image to run). This helps strike a balance between security and developer flexibility based on context (i.e., development or production environments) and associated risk appetite.
- **Define a desired target security state** / Key Performance Indicators (KPI), for example, patching, Service Level Agreements (SLA), vulnerability assessment, and remediation time. The desired target state will be dependent on the organization's security and risk appetite, context of its cloud and application landscape, and regulatory requirements. The "Pillar 6: 'Measure, Monitor, Report, and Action' paper" will go into more detail on defining a desired target security state.
- **Establish Segregation of Duties** (SoD) for production environments where at least two individuals are required to make changes to a critical feature of an application to prevent fraud and error. For environments where proof of concepts are performed (i.e., development and test), SoD controls can be removed to allow developers to freely create features with fewer security controls that could inhibit productivity. This developer freedom comes with the assumption that the associated environments do not have access to production data. The implementation of SoDs can begin in pre-production and quality assurance environments to ensure that security controls and application features work as intended. Application owners will also need to ensure application availability can be restored using an SoD approach in an incident.

Control Category	Guardrail Control
Data Protection	• Information categorization and protection are automated. • Data at rest is encrypted. • Data in transit is encrypted. • Data is categorized to identify Sensitive (SPII) and Personally Identifiable Information (PII) information. • IT systems are categorized by the type of information processed and business criticality. • Cloud storage options are configured to "private" by default. Configuring storage to "public" is an exception. • Keys and secrets should be stored and managed in key vaults.
Governance	• Policies are in place to manage the security of cloud environments. • Proper configuration management and inventory is in place. • Proper metrics are defined to track compliance and Key Performance Indicators (KPI) and Key Risk Indicators (KRI). • Control testing review is in place.

26 Cloud Security Alliance. (2021b, June 7). *Cloud Controls Matrix and CAIQ v4*. https://cloudsecurityalliance.org/artifacts/cloud-controls-matrix-v4/

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Control Category	Guardrail Control
Access Control	• Access is granted based on the Principle of Least Privilege. • Role-Based Control Access (RBAC) is implemented. • Applications have assigned owners responsible for access control. • Privileged Access (i.e., access with escalated privileges such as root) should be kept in a secure database and access granted only on exception basis with approvals. • Access to Critical Systems is managed centrally (i.e., Security operations or DevOps team).
Environment Security	• Internal networks use VPN or VPC to separate and protect internal environments. • A Zero Trust[27] security model is used. • Web Application Firewalls and anti-DDOS solutions are in place. • Servers, application dependencies, and containers are frequently patched and up-to-date with the latest versions. • Usage of third-party software components should be verified. • Ongoing vulnerability scans must be carried out. • Environment separation is in place (Production, Staging, Development, Sandbox).
Application Changes	• Changes go through a centrally managed CI/CD pipeline. • Images and libraries are centrally managed and approved. • New images are cryptographically signed. • Changes are security scanned before deployment. • Major releases are security tested (see Chapter Increasing level of assurance in code). • Changes are code reviewed, and security smoke or unit tested.
Resiliency	• Environments have resilience built-in, based on the value and criticality of the service (i.e., failover, backup). • Load balancing is in place to ensure the management of traffic volume. • Regular backup is executed to facilitate recoverability within allotted Service Level Agreements. • High Availability environments and Backups are both in place. • Business Continuity exercises and Fault Injection (see Chapter Increasing level of assurance in code) include the production environment. • Recoverability from backups is tested.

27 Cloud Security Alliance. (2020a, May 27). *Software-Defined Perimeter (SDP) and Zero Trust.* https://cloudsecurityalliance.org/artifacts/software-defined-perimeter-and-zero-trust/

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Control Category	Guardrail Control
Monitoring	• Environmental monitoring and event-based alerting are in place. • Alerting rules are configurable. • Logging of critical events and associated actions is in place. • Event threat detection is in place for anomalous and emerging threats. • Endpoints are remotely investigated for forensic purposes. • Logs should be retained based on the retention policies and should be tamper proof.

Table 2. Guardrail Categories and Controls

Cloud service providers, like AWS, can help by providing suggested guardrail controls within their environments.[28] However, it's recommended to perform the exercise in identifying the controls relevant to the organization, as can be seen in Table 2, prior to implementation.

4.5 Patterns and Templates

The use of patterns (standardized methods to deploy and handle resources) and templates (approved resource builds for consumption) can assist in secure build activities; hence, developers are empowered to create or consume secure resources.

Patterns can be the creation of documentation supported by guardrails. Documentation can support examples for creating resources on the cloud like storage, virtual private cloud, compute services and container management. Patterns can be used where developers are provided autonomy to create and manage their own resources. Patterns are an effective measure of mapping security policies and compliance requirements to the design and configuration of cloud resources. They can be produced as a shared responsibility item between developers and security teams, or even belong to a security champion.

The use of anti-patterns (bad practices and methods to deploy and handle resources) is also a noteworthy approach. However, anti-patterns have a significant drawback in being time consuming to identify, create, maintain, and consume, as well as a challenge to verify whether a cloud resource or extract of code "has not" included bad practice.

28 Amazon Web Services. (n.d.). *Guardrail reference - AWS Control Tower*. Retrieved December 22, 2021, from https://docs.aws.amazon.com/controltower/latest/userguide/guardrails-reference.html

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Templates are packaged and approved resources like images, containers and IaC code. Templates can be produced during the deployment stage, however in a continuous development model its consumption is designed to be during build. Developer created application components will leverage templates for orchestration to hosted infrastructure (cloud) with the knowledge that the templates are scanned and securely configured. The key considerations to take into account with setting up templates (e.g., docker images, containers, and IaC scripts) are the following:

- **Images:**
  - Identification of image requirements and types for each possible component consuming images. This will help produce separate templates per use case and aid in hardening activities as some components may need docker write privileges.
  - A scan of resources for vulnerabilities included in the Common Vulnerabilities and Exposures (CVEs) and/or the adoption of specific versions of software such as OS packages and libraries that remediate CVEs.
  - Hardening of images to meet security requirements for their use case, and/or to meet industry hardening guidelines (e.g., Red Hat, Palo Alto).
  - For each scanned and hardened image, the resource should be packaged and tagged as a template in an accessible repository for other developers and teams to consume. The template should be used as a baseline for secure packages and should be continuously or periodically scanned for vulnerabilities and acceptable hardening controls.
- **Containers:**
  - Identification of container use cases for each type of deployment — some instances may require additional features (e.g., volume / block storage) by design.
  - A scan of resources for weaknesses against poor design and configuration using industry hardening guidelines (e.g., CIS, NSA Kubernetes hardening) and open-source tools (e.g., kubescape).
  - For each scanned and hardened container, the resource should be packaged and tagged as a template in an accessible repository for other developers and teams to consume. The template should be used as a baseline for secure packages and should be continuously / periodically scanned for acceptable hardening controls.
- **IaC:**
  - Identification of IaC use cases and deployment of resources across all CSPs.
  - A scan of resources for weaknesses and vulnerabilities against bad configuration using industry hardening guidelines (e.g., CIS) and open-source tools (e.g., Checkov, Terrascan).
  - For each scanned and hardened IaC script, the script should be packaged and tagged as a template in an accessible repository for other developers and teams to consume. The template should be used as a baseline for secure packages and continuously / periodically scanned for acceptable hardening controls.

Licensed tooling for templates on images, containers, and IaC can allow for the creation of bespoke policies which might be advantageous to organizations with specific requirements. For organizations with a plethora of applications and product teams, templates can assist in ensuring that a consistent level of security control are applied.

© Copyright 2022, Cloud Security Alliance. All rights reserved.

5. Summary

Before DevSecOps, risk-related requirements were difficult to translate into security activities; these were often introduced at security gates with development and security teams working in separate silos. These silos created managerial and communication issues, resulting in security requirements poorly translated in the design, build and testing phases; security was often an afterthought in the entire process. The increasing speed and frequency of deployments in application development today mandated a solution that was efficient and more automated without compromising security and quality.

DevSecOps emerged as "the modern approach" to secure applications and embed compliance throughout the secure software development lifecycle (SSDL), allowing security to "shift-left." This approach involves applying during the five SSDL stages (design and architecture, coding, continuous build, integration and testing, continuous delivery and deployment, and runtime defense and monitoring[29]) the following principles and practices help to bridge the gap between compliance and development:

- Leverage methods for performing assessments continuously while amalgamating with point-in-time reviews.
- Introduce value stream mapping to identify existing practices that can be automated to improve velocity.
- Identify and translate compliance objectives to security user stories for developers to consume.
- Embrace compliance as code/policy as code to codify security requirements in cloud and infrastructure builds.
- Test frequently across the various methods of security testing (Smoke Testing, SAST, DAST, Fault Injection, Penetration Testing).
- Track and control open source through identifying, vetting and measuring risk and using SCA tooling.
- Define and create security guardrails to monitor deployments and find deviations from desired baselines autonomously.
- Leverage the use of patterns and templates to scale security consistently.
- Understand methods for monitoring potentially malicious events at the CI/CD pipeline.
- Improve the quality of source code and IaC in IDE's using linters, scanners and plugins.

Addressing the compliance and development gap requires stakeholder commitment to the practices recommended within this paper. Organizations that follow DevSecOps correctly will realize that tooling and automation, with the support of effective process, culture and governance, will improve the quality of risk management and speed of applying security controls at scale.

29 Cloud Security Alliance. (2020b, July 6). *The Six Pillars of DevSecOps: Automation*. https://cloudsecurityalliance.org/artifacts/devsecops-automation/

© Copyright 2022, Cloud Security Alliance. All rights reserved.

References

Agile Alliance. (2021, July 26). *What does INVEST Stand For?* https://www.agilealliance.org/glossary/invest/

Amazon Web Services. (n.d.). *Guardrail reference - AWS Control Tower.* Retrieved December 22, 2021, from https://docs.aws.amazon.com/controltower/latest/userguide/guardrails-reference.html

American Institute of CPAs. (n.d.-a). *Segregation of Duties. AICPA.* Retrieved December 22, 2021, from https://www.aicpa.org/interestareas/informationtechnology/resources/value-strategy-through-segregation-of-duties.html

American Institute of CPAs. (n.d.-b). *SOC for Service Organizations.* AICPA. Retrieved December 22, 2021, from https://www.aicpa.org/interestareas/frc/assuranceadvisoryservices/socforserviceorganizations.html

Cloud Security Alliance. (n.d.). *Security, Trust, Assurance and Risk (STAR).* Retrieved December 22, 2021, from https://cloudsecurityalliance.org/star/

Cloud Security Alliance. (2019, August 7). *Six Pillars of DevSecOps.* https://cloudsecurityalliance.org/artifacts/six-pillars-of-devsecops/

Cloud Security Alliance. (2020a, May 27). *Software-Defined Perimeter (SDP) and Zero Trust.* https://cloudsecurityalliance.org/artifacts/software-defined-perimeter-and-zero-trust/

Cloud Security Alliance. (2020b, July 6). *The Six Pillars of DevSecOps: Automation.* https://cloudsecurityalliance.org/artifacts/devsecops-automation/

Cloud Security Alliance. (2020c, July 13). *Hybrid Cloud and Its Associated Risks.* https://cloudsecurityalliance.org/artifacts/hybrid-clouds-and-its-associated-risks/

Cloud Security Alliance. (2021a, February 21). *The Six Pillars of DevSecOps: Collective Responsibility.* https://cloudsecurityalliance.org/artifacts/devsecops-collective-responsibility/

Cloud Security Alliance. (2021b, June 7). *Cloud Controls Matrix and CAIQ v4.* https://cloudsecurityalliance.org/artifacts/cloud-controls-matrix-v4/

CloudPassage. (2020, August 26). *Shared Responsibility Model Explained.* Cloud Security Alliance. https://cloudsecurityalliance.org/blog/2020/08/26/shared-responsibility-model-explained/

Google. (n.d.). *DevOps process: Visibility of work in the value stream.* Google Cloud. Retrieved December 22, 2021, from https://cloud.google.com/architecture/devops/devops-process-work-visibility-in-value-stream

© Copyright 2022, Cloud Security Alliance. All rights reserved.

International Organization for Standardization. (n.d.). *ISO/IEC 27001 — Information security management*. ISO. Retrieved December 22, 2021, from https://www.iso.org/isoiec-27001-information-security.html

Johnson, M. (2021, April 8). *Checkov 2.0: Deeper, broader, and faster IaC scanning*. Bridgecrew. https://bridgecrew.io/blog/checkov-2-0-release/

Manepalli, S. (2021, January 21). *Building end-to-end AWS DevSecOps CI/CD pipeline with open source SCA, SAST and DAST tools*. Amazon Web Services. https://aws.amazon.com/blogs/devops/building-end-to-end-aws-devsecops-ci-cd-pipeline-with-open-source-sca-sast-and-dast-tools/

Munroe, R. (n.d.). *Dependency*. Xkcd. Retrieved December 22, 2021, from https://xkcd.com/2347/

National Institute of Standards and Technology. (2021, October 26). *Cybersecurity Framework*. NIST. https://www.nist.gov/cyberframework

Portman, D. G. (2020, September 22). *Use Four Keys metrics like change failure rate to measure your DevOps performance*. Google Cloud Blog. https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance

Ramamoorthy, S. (2018, February 14). *DevSecOps: Security at the Speed of Business*. Cisco Blogs. https://blogs.cisco.com/security/devsecops-security-at-the-speed-of-business

S. (2019, June 7). *How INVEST helps team write effective user stories. . ..* Medium. https://sabaimam.medium.com/investing-in-user-stories-c7cfb1fc5e85

SecurityScorecard. (2018, February 15). *Limitations of Point-in-Time IT Security Risk Assessments*. https://securityscorecard.com/blog/limitations-of-it-security-risk-assessment

SonarQube. (n.d.). *SonarLint Integration*. Retrieved December 22, 2021, from https://www.sonarqube.org/sonarlint/

Velimirovic, A. (2020, September 9). *What Is Infrastructure as Code? Benefits, Best Practices, & Tools*. PhoenixNAP Blog. https://phoenixnap.com/blog/infrastructure-as-code-best-practices-tools

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Glossary

CSA STAR The STAR registry documents the security and privacy controls provided by popular cloud computing offerings. This publicly accessible registry allows cloud customers to assess their security providers to make the best procurement decision.[30]

Point-In-Time Assessments A time-bound period for assessing the maturity and compliance of a technology or set of processes. They typically last between 4 weeks or 4 months and review the current state at a particular point in time.

Compartmentalization The separation of application, platform, and infrastructure services and resources.

Segregation of Duties A building block of sustainable risk management and internal controls for a business. The principle of SOD is based on shared responsibilities of a critical process that disperses the essential functions of that process to more than one person or department.[31]

Shared Responsibility The customer security team maintains some responsibilities for security as the organization moves applications, data, containers, and workloads to the cloud. At the same time, the provider takes some responsibility, but not all. Defining the line between customer responsibilities and providers is imperative for reducing the risk of introducing vulnerabilities into public, hybrid, and multi-cloud environments.[32]

SOC 2 Report on a service organization's controls over its system relevant to security, availability, processing integrity, confidentiality, or privacy.[33]

30 Cloud Security Alliance. (n.d.). *Security, Trust, Assurance and Risk (STAR)*. Retrieved December 22, 2021, from https://cloudsecurityalliance.org/star/

31 American Institute of CPAs. (n.d.-a). *Segregation of Duties.* AICPA. Retrieved December 22, 2021, from https://www.aicpa.org/interestareas/informationtechnology/resources/value-strategy-through-segregation-of-duties.html

32 CloudPassage. (2020, August 26). *Shared Responsibility Model Explained.* Cloud Security Alliance. https://cloudsecurityalliance.org/blog/2020/08/26/shared-responsibility-model-explained/

33 American Institute of CPAs. (n.d.-b). *SOC for Service Organizations.* AICPA. Retrieved December 22, 2021, from https://www.aicpa.org/interestareas/frc/assuranceadvisoryservices/socforserviceorganizations.html

© Copyright 2022, Cloud Security Alliance. All rights reserved.

Acronyms

AWS – Amazon Web Services

CAC – Compliance as Code

CSA CCM – Cloud Security Alliance Security, Cloud Control Matrix

CI/CD – Continuous Integration, Continuous Delivery

CSA STAR – Cloud Security Alliance Security, Trust & Assurance Registry

CSC – Cloud Service Customer

CSP – Cloud Service Provider

DAST – Dynamic Application Security Testing

DORA – DevOps Research and Assessment

GDPR – General Data Protection Regulations

IaaS – Infrastructure as a Service

IAC – Infrastructure as Code

IAM – Identity and Access Management

IDE – Integrated Development Environment

INVEST – Independent, Negotiable, Valuable, Estimable, Small, Testable

ISO – International Standards Organization

KPI – Key Performance Indicator

NIST – National Institute of Standards and Technology

OPA – Open Policy Agent

OS – Operating System

OWASP – Open Web Application Security Project

PaC – Policy as Code

© Copyright 2022, Cloud Security Alliance. All rights reserved.

PaaS – Platform as a Service

PII – Personally Identifiable Information

SaaS – Software as a Service

SAST – Static Application Security Testing

SCA – Software Composition Analysis

SLA – Service Level Agreement

SOC – System and Organization Controls

SOC 2 – System and Organization Controls Two

SOD – Segregation of Duties

SPII – Sensitive Personally Identifiable Information

SSDL – Secure Software Development Lifecycle

STAR – Security Trust Assurance and Risk

VPC – Virtual Private Cloud

VPN – Virtual Private Network

VM – Virtual Machine

© Copyright 2022, Cloud Security Alliance. All rights reserved.