DevSecOps Fundamentals Guidebook:

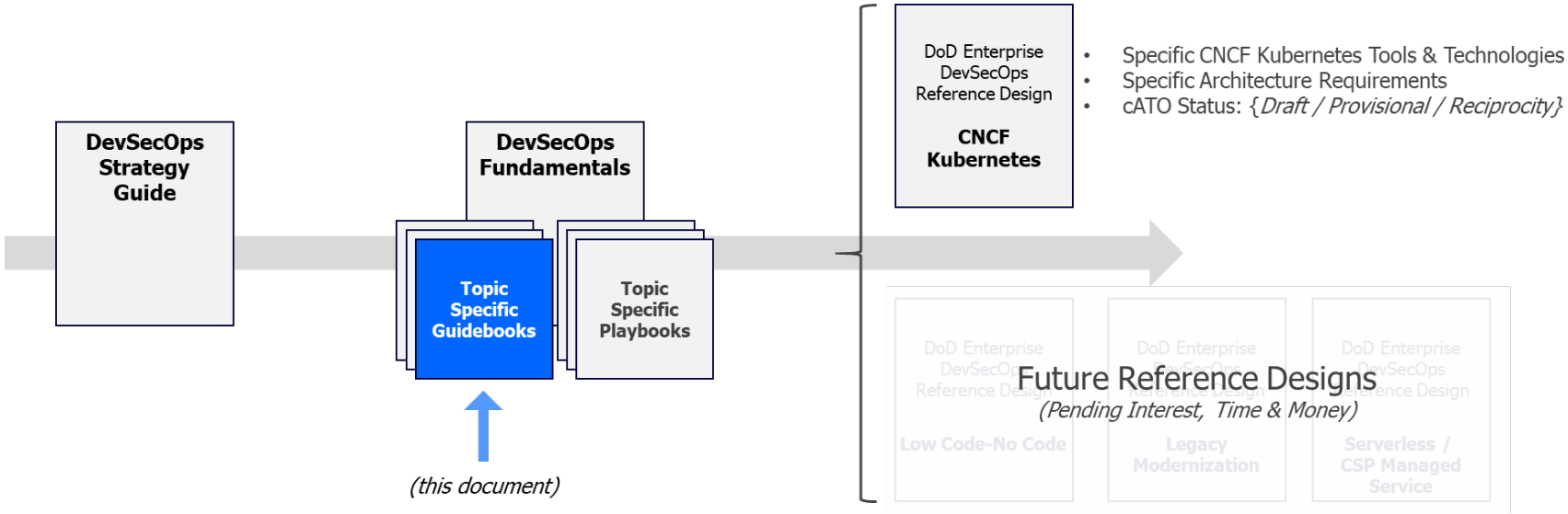
DevSecOps Tools & Activities

March 2021

Version 2.0

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Document Set Reference



Trademark Information

Names, products, and services referenced within this document may be the trade names, trademarks, or service marks of their respective owners. References to commercial vendors and their products or services are provided strictly as a convenience to our readers, and do not constitute or imply endorsement by the Department of any non-Federal entity, event, product, service, or enterprise.

Contents

Document Set Reference 2
Trademark Information 3
Introduction 6
Audience and Scope 6
DevSecOps Tools and Activities 7
Security Tools & Activities Cross Reference 8
Plan Tools and Activities 10
Develop Tools and Activities 15
Build Tools and Activities 19
Test Tools and Activities 22
Release & Deliver Tools and Activities 28
Deploy Tools and Activities 31
Virtual Machine Deployment 31
Container Deployment 31
Operate Tools and Activities 34
Monitor Tools and Activities 36
Configuration Management Tools and Activities Cross-Reference 42

Figures

Figure 1 DevSecOps Phases and Continuous Feedback Loops 6

Tables

Table 1: Security Activities Summary and Cross-Reference 8
Table 2 Specific Security Tools Common to All DevSecOps Reference Designs 9
Table 3: Plan Phase Tools 11
Table 4: Plan Phase Activities 13
Table 5: Develop Phase Tools 16
Table 6: Develop Phase Activities 17
Table 7: Build Phase Tools 20
Table 8: Build Phase Activities 21
Table 9: Test Phase Tools 23
Table 10: Test Phase Activities 25
Table 11: Release and Deliver Phase Tools 29
Table 12: Release and Deliver Phase Activities 30
Table 13: Deploy Phase Tools 32

Table 14: Deploy Phase Activities 32
Table 15: Operate Phase Tools 35
Table 16: Operate Phase Activities 35
Table 17: Monitor Phase Tools 37
Table 18: Monitor Phase Activities 41
Table 19: Configuration Management Activities Summary and Cross-Reference 43

Introduction

The goal of DevSecOps is to improve customer outcomes and mission value through the automation, monitoring, and application of security at every phase of the software lifecycle. *Figure 1 DevSecOps Phases and Continuous Feedback Loops* conveys the software lifecycle phases and continuous feedback loops.

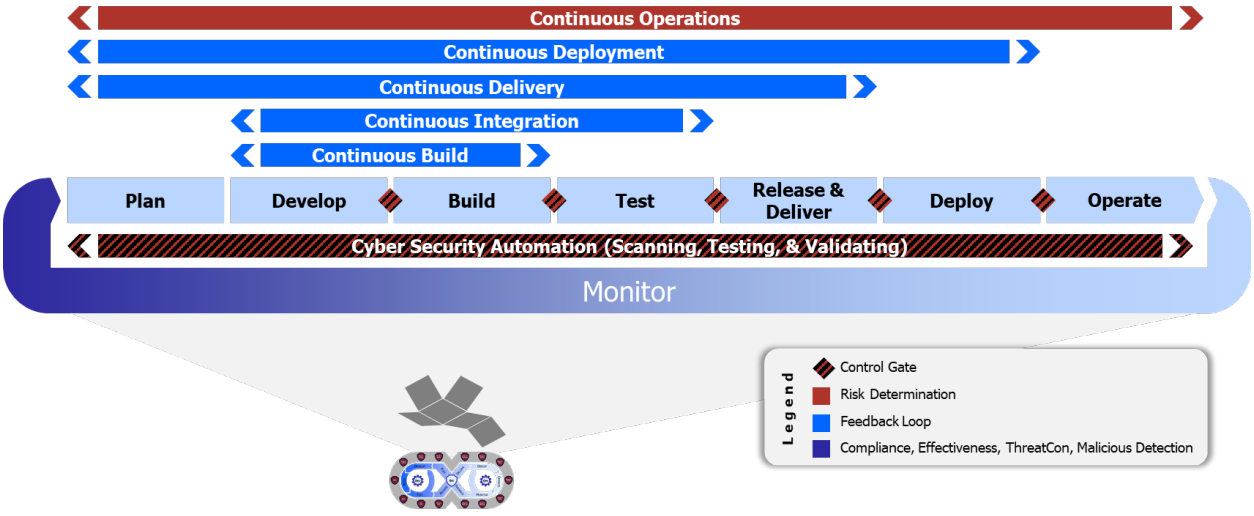


Figure 1 DevSecOps Phases and Continuous Feedback Loops

Practicing DevSecOps requires an array of purpose-built tools and a wide range of activities that rely on those tools. This document conveys the relationship between each DevSecOps phase, a taxonomy of supporting tools for a given phase, and the set of activities that occur at each phase cross-referenced to the tool(s) that support the specific activity.

Audience and Scope

The target audience for this document include:

- DoD Enterprise DevSecOps platform capability providers
- DoD DevSecOps teams
- DoD programs

The Tools and Activities that follow are foundational, but incomplete when considered in isolation. Each DoD Enterprise DevSecOps Reference Architecture additively defines the complete set of Tools and Activities required to achieve a specific DevSecOps implementation.

DevSecOps Tools and Activities

The tools and activities that follow are common across all DevSecOps ecosystems.

Tools and Activities

All Activities and Tools are listed in table format throughout this document.

Tools tables identify specific categories of tooling required to support the proper operation of a software factory within a DevSecOps ecosystem. The tools captured are categorical, not specific commercial products and/or versions. Each program should identify and select tools that properly support their software development needs. When possible, DoD enterprise-wide tooling that has already either been approved or has obtained provisional authorization is preferred.

Tools tables include the below columns:

- Tool: A specific tool category
- Features: Common characteristics used to describe the tool category
- Benefits: Simple value-proposition of the tool category
- Inputs: Types of data collected by the tool category
- Outputs: Types of artifacts that result from using the tool category
- Baseline: Either a status of REQUIRED or PREFERRED, where required indicates that the tool must be available within the software factory as part of the Minimal Viable Product (MVP) release, and preferred indicates an aspirational capability obtained as the ecosystem matures

Specific reference designs may elevate a specific tool from PREFERRED to REQUIRED, as well as add additional tools and/or activities that specifically support the nuances of a given reference design. Reference designs cannot lower a tool listed in this document from required to preferred.

Activity tables list a wide range of activities for DevSecOps practices. The activities captured here do not diminish the fact that each program should define their own unique processes, choose proper and meaningful activities, and select specific software factory tools suitable for their software development needs. The continuous process improvement that results from the DevSecOps continuous feedback loops and performance metrics aggregation should drive the increase of automation across each of these activities.

Activities tables include the below columns:

- Activities: Actions that occur within the specific DevSecOps phase
- Description: Simple explanation of the activity being performed
- Inputs: Types of data that feed the activity
- Outputs: Types of data that result from the activity
- Tool Dependencies: List of tool categories required to support the activity

Security Tools & Activities Cross Reference

Security is integrated into the core of the DevSecOps phases, weaved into the fabric that touches each phase depicted in *Figure 1 DevSecOps Phases and Continuous Feedback Loops*. This integrated and wrapped approach to security facilitates automated risk characterization, monitoring, and risk mitigation across the totality of the application lifecycle. *Table 1: Security Activities Summary and Cross-Reference* summarizes this security posture by representing all of the security activities, the linked DevSecOps phase, and the activities and tools references.

The "Ops" part of DevSecOps means that security information and event management (SIEM) and security orchestration, automation, and response (SOAR) capabilities are baked-in throughout each of the eight DevSecOps SDLC phases. Integration into these tools must be considered at every phase in order to properly practice DevSecOps. This requirement substantially differentiates DevSecOps from legacy ways of development software where integration was done after the fact using a "bolt-on" mentality.

Table 1: Security Activities Summary and Cross-Reference

Activities	Phase	Activities Table Reference	Tool Dependencies	Tool Table Reference
Threat modeling	Plan	Table 4	Threat modeling tool	Table 3
Security code development	Develop	Table 6	IDE	Table 5
Static code scan before commit	Develop	Table 6	IDE security plugins	Table 5
Code commit scan	Develop	Table 6	Source code repository security plugin	Table 5
Static application security test and scan	Build	Table 8	SAST tool	Table 7
Dependency vulnerability checking	Build	Table 8	Dependency checking / BOM checking tool	Table 7
Dynamic application security test and scan	Test	Table 10	DAST tool or IAST tool	Table 9
Manual security testing (such as penetration test)	Test	Table 10	Varies tools and scripts (may include network security test tool)	Table 9
Post-deployment security scan	Deploy	Table 14	Security compliance tool	Table 13
Operational dashboard	Operate	Table 16	Backup	Table 15
System Security monitoring	Monitor	Table 18	Information Security Continuous Monitoring (ISCM)	Table 17

Table 2 Specific Security Tools Common to All DevSecOps Reference Designs

Tool	Features	Benefits	Baseline
Runtime Defense	Creates runtime behavior models, including whitelist and least privilege	Dynamic, adaptive cybersecurity	REQUIRED
Vulnerability Management	Provides cyber vulnerability management capabilities for the software factory and the artifacts produced	Ensures everything is appropriately patched to avoid known vulnerabilities.	REQUIRED
CVE Service/Host Based Security	Provides CVEs. Used by the vulnerability management agent.	Ensures the system is adequately aware of ever-evolving cyber threats across all software artifacts.	REQUIRED
Artifact Repository	Storage and retrieval of software artifacts. These may be dependency libraries, COTS components, FOSS components, etc.	Iron Bank is the DoD enterprise artifact repository for hardened software artifacts, including containers.	REQUIRED
Zero Trust Architecture	Accepting the position that perimeter only and/or “bolt-on” cybersecurity tooling is no longer enough. Zero Trust principles, including mTLS tunnels, must be baked in to each of the eight phases of the DevSecOps SDLC.	Reduces the attack surface and improves baked-in security, further reducing the risk of exposure and compromise.	REQUIRED
Behavior Detection	Ability to establish the common types of behaviors that exist both within the software factory and across each environment.	Alerting to the effect of “I saw something.”	REQUIRED
Behavior Prevention	Ability to proactive or rapidly deny or stop an anomaly from occurring either in the software factory or across any of its environments.	Alerting and notification to the effect of “I inhibited something.”	PREFERRED

Plan Tools and Activities

Planning tools support software development planning, which includes configuration management planning, change management planning, project management planning, system design, software design, test planning, and security planning. Some tools will be used throughout the software lifecycle, such as a team collaboration tool, an issue tracking system, and a project management system. Some tools are shared at the enterprise level across programs. Policy and enforcement strategy should be established for access controls on various tools.

Table 3: Plan Phase Tools lists the typical tools that assist the planning process. The activities supported by the plan phase are listed in *Table 4: Plan Phase Activities*. Some activities are suitable at enterprise or program level, such as DevSecOps ecosystem design, project team onboarding planning, and change management planning. Others fit at the project level and are considered continuous in the DevSecOps lifecycle.

Table 3: Plan Phase Tools

Tool	Features	Benefits	Inputs	Outputs	Baseline
Team collaboration system	Audio/video conferencing; chat/messaging; brainstorming discussion board; group calendars; file sharing; Wiki website	Simplify communication and boost team efficiency	Team meetings; Design notes; Documentation	Organized teamwork; Version controlled documents	REQUIRED
Issue tracking system	Bugs and defect management; Feature and change management; Prioritization management; Assignment management; Escalation management; Knowledge base management	Easy to detect defect trends Improve software product quality Reduce cost and improve Return on Investment (ROI)	Bug report Feature/change request Root cause analysis Solutions	Issues feature/change tickets. Issue resolution tracking history	REQUIRED
Asset inventory management	Collect information about all IT assets; Maintain a “real-time” inventory of all applications, software licenses, libraries, operating systems, and versioning information	Increase situation awareness	IT assets (applications, software licenses, libraries, operating systems, and versioning information)	Asset inventory	PREFERRED
Configuration management database (CMDB)	Auto-discovery; Dependency mapping; Integration with other tools; Configuration auditing	Centralized database used by many systems (such as asset management, configuration management, incident management, etc.) during development and operations phases.	IT hardware and software components information	Configuration items	PREFERRED
Project management system	Task management Scheduling and time management Resource management Budget management Risk management	Assist project progress tracking Optimize resource allocation	Tasks, scheduling, resource allocation, etc.	Project plan	REQUIRED

Tool	Features	Benefits	Inputs	Outputs	Baseline
Software system design tool	Assist requirement gathering, system architecture design, components design, and interface design	Independent of programming languages Helps visualize the software system design	User requirements Design ideas	System design documents, Function design document, Test plan, System deployment environment configuration plan	PREFERRED
Threat modeling tool	Document system security design; Analyze the design for potential security issues; Review and analysis against common attack patterns; Suggest and manage mitigation	Allows software architects to identify and mitigate potential security issues early.	System design	Potential threats and mitigation plan	PREFERRED
Data modeling tool	Model the interrelationship and flows between different data elements	Ensure the required data objects by the system are accurately represented	System requirement; Business logic	Data model	PREFERRED (if using a database)

Table 4: Plan Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependencies
DevSecOps ecosystem design	Design the DevSecOps process workflows that are specific to this project	- Change management process; - System design; - Release plan & schedule.	DevSecOps process flow chart; DevSecOps ecosystem tool selection; Deployment platform selection	Team collaboration system
Project team onboarding planning	Plan the project team onboarding process, interface, access control policy	Organization policy	Onboarding plan	Team collaboration system
Change management planning	Plan the change control process	- Organizational policy; - Software development best practice.	Change control procedures; Review procedures; Control review board; change management plan	Team collaboration system; Issue tracking system
Configuration identification	Discover or manual input configuration items into CMDB; Establish system baselines	-IT infrastructure asset; - Software system components (include DevSecOps tools); -code baselines -document baselines.	Configuration items	CMDB; Source code repository; Artifact repository; Team collaboration system
Configuration management (CM) planning	Plan the configuration control process; Identify configuration items	- Software development, security and operations best practice; - IT infrastructure asset; - Software system components.	CM processes and plan; CM tool selection; Responsible configuration items; Tagging strategy	Team collaboration system; Issue tracking system
Software requirement analysis	Gather the requirements from all stakeholders	- Stakeholder inputs or feedback; - Operation monitoring feedback; - Test feedback.	Requirements Documents -Feature requirements -Performance requirements -Privacy requirements -Security requirements	Team collaboration system; Issue tracking system
System design	Design the system based the requirements	Requirements documents	System Design Documents: -System architecture -Functional design -Data flow diagrams -Test plan	Team collaboration system; Issue tracking system Software system design tools

Activities	Description	Inputs	Outputs	Tool Dependencies
			-Infrastructure configuration plan -Tool selections - Ecosystem Tools: -Development tool -Test tool -Deployment platform	
Project planning	Project task management Release planning		-Project Plan -Task plan & schedule; -Release plan & schedule.	Team collaboration system; Project management system
Risk management	Risk assessment	- System architecture; - Supply chain information; - Security risks.	Risk management plan	Team collaboration system;
Threat modeling	Identify potential threats, weaknesses and vulnerabilities. Define the mitigation plan	System design	Potential threats and mitigation plan	Threat modeling tool
Database design	Data modeling; database selection; Database deployment topology	System requirement; System design	Database design document	Data modeling tool; Team collaboration system
Design review	Review and approve plans and documents	Plans and design documents;	Review comments; Action items	Team collaboration system
Documentation version control	Track design changes	Plans and design documents;	Version controlled documents	Team collaboration system

Develop Tools and Activities

Develop phase tools support the development activities that convert requirements into source code. The source code includes application code, test scripts, Infrastructure as Code, Security as Code, DevSecOps workflow scripts, etc. The development team may rely on a single modern integrated development environment (IDE) for multiple programming language support. The IDE code assistance feature aids developers with code completion, semantic coloring, and library management to improve coding speed and quality. The integrated compiler, interpreter, lint tools, and static code analysis plugins can catch code mistakes and suggest fixes before developers check code into the source code repository. Source code peer review or pair programming are other ways to ensure code quality control. All the code generated during development must be committed to the source code repository and thus version controlled. Committed code that breaks the build should be checked in on a branch and not merged into the trunk until it is fixed.

Although not considered an explicit tool or activity, it is important that DevSecOps teams establish a firm strategy to design and create composable software artifacts that contain new or updated capabilities released through a CI/CD pipeline. Only through application decomposition into a discrete set of manageable services is it possible to properly avoid high-risk monolithic development practices.

The components that facilitate code development, along with their inputs and outputs, are listed in *Table 5: Develop Phase Tools*, and the activities supported by these tools are listed in *Table 6: Develop Phase Activities*.

Table 5: Develop Phase Tools

Tool	Features	Benefits	Inputs	Outputs	Baseline
Integrated development environment (IDE)	Source code editor Intelligent code completion Compiler or interpreter Debugger Build automation (integration with a build tool)	Visual representation Increase efficiency Faster coding with less effort Improved bug fixing speed Reproducible builds via scripts	Developer coding input	Source code	REQUIRED
Integrated development environment (IDE) security plugins	Scan and analyze the code as the developer writes it, notify developer of potential code weakness and may suggest remediation	Address source code weaknesses and aid developers to improve secure coding skills	Source code; known weaknesses	source code weakness findings	PREFERRED
Source code repository	Source code version control Branching and merging Collaborative code review	Compare files, identify differences, and merge the changes if needed before committing. Keep track of application builds	Source code Infrastructure as code	Version controlled source code	REQUIRED
Source code repository security plugin	Check the changes for suspicious content such as Secure Shell (SSH) keys, authorization tokens, passwords and other sensitive information before pushing the changes to the main repository. If it finds suspicious content, it notifies the developer and blocks the commit.	Helps prevent passwords and other sensitive data from being committed into a version control repository	Locally committed source code	Security findings and warnings	PREFERRED
Code quality review tool	View code changes, identify defects, reject or approve the changes, and make comments on specific lines. Sets review rules and automatic notifications to ensure that reviews are completed on time.	Automates the review process which in turn minimizes the task of reviewing the code.	Source code	Review results (reject or accept), code comments	PREFERRED

Table 6: Develop Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependencies
Application code development	Application coding	Developer coding input	Source code	IDE
Infrastructure code development	-System components and infrastructure orchestration coding -Individual component configuration script coding	Developer coding input	Source code	IDE
Security code development	Security policy enforcement script coding	Developer coding input	Source code	IDE
Test development	Develop detailed test procedures, test data, test scripts, test scenario configuration on the specific test tool	Test plan	Test procedure document; Test data file; Test scripts	IDE; Specific test tool
Database development	Implement the data model using data definition language or data structure supported by the database; Implement triggers, views or applicable scripts; Implement test scripts, test data generation scripts.	Data model	Database artifacts (including data definition, triggers, view definitions, test data, test data generation scripts, test scripts, etc.)	IDE or tools come with the database software
Code commit	Commit source code into version control system	Source code	Version controlled source code	Source code repository
Code commit scan	Check the changes for sensitive information before pushing the changes to the main repository. If it finds suspicious content, it notifies the developer and blocks the commit.	Locally committed source code	Security findings and warnings	Source code repository security plugin
Code review	Perform code review to all source code. Note that pair programming counts.	Source code	Review comments	Code quality review tool
Documentation	Detailed implementation documentation	User input; Developed Source Code	Documentation; Auto generated Application Programming Interface (API) documentation	IDE or document editor or build tool

Activities	Description	Inputs	Outputs	Tool Dependencies
Static code scan before commit	Scan and analyze the code as the developer writes it. Notify developers of potential code weakness and suggest remediation.	Source code; known weaknesses	source code weakness findings	IDE security plugins
VM hardening	Harden the deliverable for production deployment	Running VM	-Vulnerability report and recommended mitigation	Security compliance tool
Code Commit Logging	Logging of successful code commits, or analysis of rejected commits, which will have benefits to security and insider threat protections	-Review Comments -Source Code Weakness Findings -Version-Controlled Source Code -Security Findings and Warnings	Code Commit Log	

Build Tools and Activities

Build tools perform the tasks of building and packaging applications, services, and microservices into artifacts. For languages like C++, building starts with compiling and linking. The former is the act of turning source code into object code and the latter is the act of combining object code with libraries to create an executable file. For Java Virtual Machine (JVM) based languages, building starts with compiling to class files, then building a compressed file such as a jar, war, or ear file, which includes some metadata, and may include other files such as icon images. For interpreted languages, such as Python or JavaScript, there is no need to compile, but lint tools help to check for some potential errors such as syntax errors. Building should also include generating documentation, such as Javadoc, copying files like libraries or icons to appropriate locations, and creating a distributable file such as a tar or zip file. The build script should also include targets for running automated unit tests.

Modern build tools can also be integrated into both an IDE and a source code repository to enable building both during development and after committing. For those applications that use containers, the build stage also includes a containerization tool.

Build-related tools along with their inputs and outputs are listed in *Table 7: Build Phase Tools*, and the activities supported by the build-related tools are listed in *Table 8: Build Phase* Activities.

Table 7: Build Phase Tools

Tool	Features	Benefits	Inputs	Outputs	Baseline
Build tool	Dependency Management Compile Link (if appropriate) Built-in lint stylistic checking Integration with IDE	Reduces human mistakes Saves time	Source code under version control Artifacts	Binary artifacts stored in the Artifact repository	REQUIRED
Lint tool	Analyzes source code to flag programming errors, bugs, stylistic errors, and suspicious constructs. Applicable to both compiled or interpreted languages	Improve code readability; Pre-code review; Finding (syntax) errors before execution for interpreted languages	Source code or scripts	Analyze results	PREFERRED
Artifact Repository	Binary artifact version control	Separate binary control from source control to avoid external access to source control system. Improved build stability by reducing reliance on external repositories. Better quality software by avoiding outdated artifacts with known issues.	Artifacts	Version controlled artifacts	REQUIRED
Static Application Security Test (SAST) tool	SAST analyzes application static codes, such as source code, byte code, binary code, while they are in a non-running state to detect the conditions that indicate code weaknesses.	Catch code weaknesses at an early stage. Continuous assessment during development.	Source code; known vulnerabilities and weaknesses	Static code scan report and recommended mitigation.	REQUIRED
Dependency checking /Bill of Materials checking tool	Identify vulnerabilities in the dependent components based on publicly disclosed open source vulnerabilities	Secure the overall application; Manage the supply chain risk	BOM, including: -Dependency list - Licensing	Vulnerability report	PREFERRED

Table 8: Build Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependencies
Build	Compile and link	Source code; dependencies	-Binary artifacts -Build Report	Build tool; Lint tool; Artifact repository
Static application security test and scan	Perform SAST to the software system	Source code; known vulnerabilities and weaknesses	Static code scan report and recommended mitigation.	SAST tool
Dependency vulnerability checking	Identify vulnerabilities in the open source dependent components	Dependency list or BOM list	Vulnerability report	Dependency checking / BOM checking tool
Release packaging	Package binary artifacts, VM images, infrastructure configuration scripts, proper test scripts, documentation, checksum, digital signatures, and release notes as a package.	Binary artifacts; Scripts; Documentation; Release notes	Released package with checksum and digital signature	Release packaging tool
Store artifacts	Store artifacts to the artifact repository	Binary artifacts; Database artifacts; Scripts; Documentation;	Versioned controlled artifacts	Artifact Repository
Build configuration control and audit	Track build results, SAST and dependency checking report; Generate action items; Make go/no-go decision to the next phase	Build results; SAST report; Dependency checking report	Version controlled build report; Action items; Go/no-go decision	Team collaboration system; Issue tracking system; CI/CD orchestrator

Test Tools and Activities

The discipline of testing changes within the automated processes of DevSecOps. Test moves away from the traditional "test the system as implemented" and becomes "test the code that implements the system." One implication of this evolution is that re-skilling of the test team is needed; the old skill set of "sit at a screen and use the app as you were trained for 3 days to use it" is no longer applicable. Rather, testing is about automation, and testers will need to become coders of that automation.

Test tools support continuous testing across the software development lifecycle. Test activities may include, but are not limited to, unit test, functional test, integration test, system test, regression test, acceptance test, performance test, and variety of security tests. All tests start with test planning and test development, which includes detailed test procedures, test scenarios, test scripts, and test data. Automated testing can be executed by running a set of test scripts or running a set of test scenarios on the specific test tool without human intervention. If full automation is not possible, the highest percentage of automation is desired. It is highly recommended to leverage emulation and simulation to test proper integration between components such as microservices and various sensors/systems so integration testing can be automated as much as possible. Automation will help achieve high test coverage and make continuous ATO practicable, as well as significantly increase the quality of delivered software.

The components involved with the test phase are listed in *Table 9: Test Phase Tools*. The activities supported by the test phase are listed in *Table 10: Test Phase Activities*. These activities happen at different test stages:

- Development stage: unit test, SAST discussed in the build phase
- System test stage: DAST or IAST, integration test, system test
- Pre-production stage: manual security test, performance test, regression test, acceptance test, container policy enforcement, and compliance scan

Test audit, test deployment, and configuration audit happen at all stages.

Table 9: Test Phase Tools

Tool	Features	Benefits	Inputs	Outputs	Baseline
Test development tool	Assists test scenario, test script, and test data development. The specific tool varies, depending on the test activity (such as unit test, penetration test) and the application type (e.g., web application, or Hadoop data analytics)	Increase the automation and rate of testing	Test plan	test scenarios, test scripts, test data	REQUIRED
Test data generator	Generates test data for the system (such as network traffic generator, web request generator)	Increase test fidelity	Test scenario, test data	Input data for the system under test	PREFERRED
Test tool suite	A set of test tools to perform unit test, interface test, system test, integration test, performance test and acceptance test of the software system. Generate test report Specific tool varies depending on the type of tests, software application, and programming language	Increase test automation, speed	Test scenario, test scripts, test data	Test results, test report	REQUIRED
Test coverage tool	Measures how much code is exercised while the automated tests are running	Shows the fidelity of the test results	Application code, automated tests	The percentage of code that is exercised by the tests.	REQUIRED
Test Management Tool	Manages requirements, streamlines test case design from requirements, plans test activities, manages test environment, tracks test status and results.	Increases QA team collaboration and streamlines test processes.	Requirements, test cases, test results	Test progress, test results statistics	PREFERRED
Non-security compliance scan	Such as Section 508 accessibility compliance	Ensures compliance	Artifacts	Compliance report	PREFERRED
Software license compliance checker	Inventory software license; Audit the compliance.	Software license compliance and software asset management	Purchased license info; Software instances	Compliance report	PREFERRED

Tool	Features	Benefits	Inputs	Outputs	Baseline
Dynamic Application Security Test (DAST) tool	DAST tools analyze a running application dynamically and can identify runtime vulnerabilities and environment related issues.	Catch the dynamic code weakness in runtime and under certain environment setting. Identify and fix issues during continuous integration.	Running software application; fuzz inputs	dynamic code scan report and recommended mitigation.	PREFERRED
Interactive Application Security Test (IAST) tool	Analyze code for security vulnerabilities while the application is run by an auto-test, human tester, or any activity "interacting" with the application functionality	Provide accurate results for fast triage; pinpoint the source of vulnerabilities	Running application, and operating systems; Fuzz inputs	Analysis report and recommended mitigation.	PREFERRED
Network security test tool	Simulate real-world legitimate traffic, distributed denial of service (DDOS), exploits, malware, and fuzzing.	Validate system security; increase attack readiness; reduce the risk of system degradation.	Test configuration	Test traffic	PREFERRED
Database test tool suite	Tools that facilitate database test; It includes test data generator, database functional test tool, database load test tool;	Automate or semi-automate the database tests	Test data; Test scenario	Test results	PREFERRED if using a database
Database security scan and test tool	Find the database common security vulnerabilities, such as weak password, known configuration risks, missing patches; Structured Query Language (SQL) injection test tool; Data access control test; User access control test; Denial of service test	Reduce the security risks	Test data; Test scenarios	Vulnerability findings; Recommended mitigation actions	PREFERRED if using a database

Table 10: Test Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependencies
Unit test	Assist unit test script development and unit test execution. It is typically language specific.	Unit test script, individual software unit under test (a function, method or an interface), test input data, and expected output data	Test report to determine whether the individual software unit performs as designed.	Test tool suite, Test coverage tool
Dynamic application security test and scan	Perform DAST or IAST testing to the software system	Running application and underlying OS; fuzz inputs	Vulnerability, static code weakness and/or dynamic code weakness report and recommended mitigation	DAST tool or IAST tool
Integration test	Develops the integration test scripts and execute the scripts to test several software units as a group with the interaction between the units as the focus.	Integration test scripts, the software units under test, test input data, and expected output data	Test report about whether the integrated units performed as designed.	Test tool suite
System test	System test uses a set of tools to test the complete software system and its interaction with users or other external systems. Includes interoperability test, which demonstrates the system's capability to exchange mission critical information and services with other systems.	System test scripts, the software system and external dependencies, test input data and expected output data	Test result about if the system performs as designed.	Test tool suite
Manual security test	Such as penetration test, which uses a set of tools and procedures to evaluate the security of the system by injecting authorized simulated cyber-attacks to the system. CI/CD orchestrator does not automate the test, but the test results can be a control point in the pipeline.	Running application, underlying OS, and hosting environment	Vulnerability report and recommended mitigation	Varies tools and scripts (may include network security test tool)

Activities	Description	Inputs	Outputs	Tool Dependencies
Performance test	Ensure applications will perform well under the expected workload. The test focus is on application response time, reliability, resource usage and scalability.	Test case, test data, and the software system	Performance metrics	Test tool suite, Test data generator
Regression test	A type of software testing to confirm that a recent program or code change has not adversely affected existing features.	Functional and non-functional regression test cases; the software system	Test report	Test tool suite
Acceptance test	Conduct operational readiness test of the system. It generally includes: Accessibility and usability test failover and recovery test performance, stress and volume test security and penetration test interoperability test compatibility test supportability and maintainability	The tested system Supporting system Test data	Test report	Test tool suite, Non-security compliance scan
Compliance scan	Compliance audit	Artifacts; Software instances; System components	Compliance reports	Non-security compliance scan; Software license compliance checker; Security compliance tool
Test audit	Test audit keeps who performs what test at what time and test results in records	Test activity and test results	Test audit log	Test management tool
Test deployment	Deploy application and set up testing environment using Infrastructure as Code	Artifacts (application artifacts, test code) Infrastructure as Code	The environment ready to run tests	Configuration automation tool; IaC
Database functional test	Perform unit test and functional test to database to verify the data definition, triggers, constrains are implemented as expected	Test data; Test scenarios	Test results	Database test tools
Database non-functional test	Conduct performance test, load test, and stress test; Conduct failover test	Test data; Test scenarios	Test results	Database test tools

Activities	Description	Inputs	Outputs	Tool Dependencies
Database security test	Perform security scan; Security test	Test data; Test scenarios	Test results	Vulnerability findings; Recommended mitigation actions
Test configuration audit	Track test and security scan results;	Test results; Security scan and compliance scan report	Version controlled test results; Action items	Team collaboration system; Issue tracking system; CI/CD orchestrator
Test configuration control	Generate action items; Make go/no-go decision to the next phase. (There may be several iterations for several tests across stages)	Version controlled test results	Go/no-go decision	Team collaboration system; Issue tracking system; CI/CD orchestrator

Release & Deliver Tools and Activities

In the release and deliver phase, the software artifacts are digitally signed to verify that they have passed build, all tests, and security scans. They are then delivered to the artifact repository. The content of the artifacts depends on the application. It may include, but is not limited to, container images, VM images, binary executables (such as jar, war, and ear files), test results, security scan results, and Infrastructure as Code deployment scripts. Artifacts will be tagged with the release tag if GO release decision is made based on the configuration audit results. The artifacts with the release tag are delivered to production.

The mission program could have more than one artifact repository, though more than likely there is a centralized repo where separate artifact types are appropriately tagged. One artifact repository (or set of tags) is used in the build stage to store build results. The test deployment activity can fetch the artifacts from the build stage artifact repository to deploy the application into various environments (development, test, or pre-production). Another artifact repository (or set of tags) may be used to stage the final production deliverables. The production deployment will get all the artifacts from the production artifact repository to deploy the application.

Some mission program application systems have geographically distributed operational regions across the country or even overseas. In order to increase deployment velocity, a remote operational region may have its own local artifact repository that replicates the artifact repository completely or partially. During release, a new artifact is pushed into the artifact repository and then replicated to other regional artifact repositories.

The tools that support the release and deliver phase are listed in *Table 11: Release and Deliver Phase Tools*, and the common activities supported by the release and deliver-related tools are listed in *Table 12: Release and Deliver Phase Activities.*

Table 11: Release and Deliver Phase Tools

Tool	Features	Benefits	Inputs	Outputs	Baseline
Release packaging tool	Package binary artifacts, VM images, infrastructure configuration scripts, proper test scripts, documentation, release notes as a package; generate checksum and digital signature for the package. The package may be prepared for a specific installer or it is a self-extracting installer itself.	Release package (such as a bundle of artifacts, self-extracting software installer, software tar file, etc.)	Binary artifacts, VM images, infrastructure configuration scripts, proper test scripts, documentation, release notes	Release package with checksum and digital signature (a bundle of artifacts, such as a self-extracting software installer, or a tar file, etc.)	REQUIRED if using VMs

Table 12: Release and Deliver Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependency
Release go / no-go decision	This is part of configuration audit; Decision on whether to release artifacts to the artifact repository for the production environment.	Design documentation; Version controlled artifacts; Version controlled test reports; Security test and scan reports	go / no-go decision; Artifacts are tagged with release tag if go decision is made	CI/CD Orchestrator
Deliver released artifacts	Push released artifacts to the artifact repository	Release package	New release in the artifact repository	Artifacts repository
Artifacts replication	Replicate newly release artifacts to all regional artifact repositories	Artifacts	Artifacts in all regional artifact repositories	Artifacts repositories (release, regional)
Ops Team Acceptance	Testing on the delivered artifacts to ensure that they meet operational requirements	Release package	Accepted release package	
Configuration Integration Testing		Accepted Release Package	Configuration Results	
Development Test and Operational Test		Known CVEs, privacy requirements, security requirements, and potential threats	Recommendations	
Parallel government testing		Feature requirements and performance requirements	Recommendations	
Delivery Results Review		Configuration results and Recommendations	Production Push Go/No-Go Decision	

Deploy Tools and Activities

The tools used in the Deploy phase are environment and deployment stage dependent. The two dominant deployment options include virtual machines and software containers.

Virtual Machine Deployment

Legacy applications can be deployed as virtual machines using a standards-based format such as Open Virtualization Format (OVF), which can be imported by the market-leading hypervisors. The virtualization manager manages the virtual compute, storage, and network resources. In some hosting environments, such as a general-purpose cloud, the virtualization manager also provides some security capabilities, such as micro-segmentation, which creates security zones to isolate VMs from one another and secure them individually. Several capabilities of the virtualization manager are keys to the success of mission application runtime operation and security, such as health checking, virtual resource monitoring, and scaling. The application production environment infrastructure has to leverage these capabilities in its architecture and configuration.

The use of “clones” from a master image library enables VMs to be created quickly. A clone is made from a snapshot of the master image. The use of clones also enables the concept of immutable infrastructure by pushing updated, clean images to the VM each time it is started. Only the master image needs to be patched or updated with the latest developed code; each running image is restarted to pick up these changes.

Container Deployment

A container manager provides capabilities that check for new versions of containers, deploys the containers to the production environment, and performs post-deployment checkout. The container manager consists of an OCI-compliant container runtime and a CNCF Certified Kubernetes, which is an orchestration tool for managing microservices or containerized applications across a cluster of nodes. The nodes could be bare metal servers or VMs. The container manager may be owned by a mission program or provided by the cloud hosting environment. It simplifies container management tasks, such as instantiation, configuration, scaling, monitoring, and rolling updates. The CNCF Certified Kubernetes interacts with the underlying virtualization manager in the cloud environment to ensure each node’s health and performance, and scale it as needed. This scaling includes container scaling within the CNCF Certified Kubernetes cluster, but when running in a cloud, it also includes the ability to auto-scale a number of nodes in a cluster by adding or deleting VMs.

Deploy phase tools and their related activities are listed in *Table 13: Deploy Phase Tools* and *Table 14: Deploy Phase Activities*, respectively.

Table 13: Deploy Phase Tools

Tool	Features	Benefits	Inputs	Outputs	Baseline
Virtualization Manager	VM instance management VM resource monitoring (provided on hosting environment)	Centralized VM instantiation, scaling, and monitoring	VM instance specification and monitoring policy	Running VM	REQUIRED if using VMs
Data masking tool	Shield personally identifiable information or other confidential data	Provide data privacy; Reduce the risk of data loss during data breach	Original data	Masked data	PREFERRED if database contains sensitive data
Database encryption tool	Encrypt data at rest and in transit	Provide data privacy and security; Prevent data loss	Original data	Encrypted data	REQUIRED if database contains highly sensitive data
Database automation tool	Automate database tasks, such as deployments, upgrades, discovering and troubleshooting anomalies, recovering from failures, topology changes, running backups, verifying data integrity, and scaling.	Simplify database operations and reduce human errors	Database artifacts; Data; Running status and events	Status report; Warnings; alerts	PREFERRED if using a database
Configuration automation tools	Execute the configuration scripts to provision the infrastructure, security policy, environment, and the application system components.	Configuration automation Consistent provisioning	Infrastructure configuration scripts Infrastructure configuration data	Provisioned deployment infrastructure	REQUIRED

Table 14: Deploy Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependency
Artifact download	Download newly release artifacts from the artifact repository	Artifact download request	Requested artifacts	Artifact repository
Infrastructure provisioning automation	Infrastructure systems auto provisioning (such as software defined networking, firewalls, DNS, auditing and logging system, user/group permissions, etc.)	Infrastructure configuration scripts / recipes / manifests / playbooks	Provisioned and configured infrastructure	Configuration automation tools; IaC
Create linked clone of VM master image	Instantiate VM by creating a link clone of parent VM with master image	VM parent New VM instance parameters	New VM instance	Virtualization Manager
Post-deployment security scan	System and infrastructure security scan	Access to system components and infrastructure components	Security vulnerability findings	Security compliance tool
Post-deployment checkout	Run automated test to make sure the important functions of system are working	Smoke test scenarios and test scripts	Test results	Test scripts
Database installation	Database software installation; Cluster or high availability setup	Artifacts in the repository; data	Running database system	Artifact repository; Database automation tool; Data masking or encryption tool if needed
Database artifact deployment	Database artifacts deployment and data loading	Artifacts in the repository; data	Running database system	Artifact repository; Database automation tool; Data masking or encryption tool if needed

Operate Tools and Activities

Operate phase tools are used for system scaling, load balancing, and backup.

Load balancing monitors resource consumption and demand, and then distributes the workloads across the system resources. Scaling helps dynamic resource allocation based on demand. Consider the popularity of virtual machines and software containers in a CNCF Certified Kubernetes cluster as deployment options, both support load balancing and scaling capabilities. Kubernetes handles the load balancing and scaling at the software container level, while the virtualization manager works at the VM level.

Application deployment must have proper load balancing and scaling policies configured. During runtime, the management layer will continuously monitor the resources. If the configured threshold is reached or exceeded (for example if memory or Central Processing Unit (CPU) usage exceeds a pre-set threshold), then the system triggers the load balancing or scaling action(s) automatically. Auto-scaling must be able to scale both up and down.

Operate phase tools and their related activities are listed in *Table 15: Operate Phase Tools* and *Table 16: Operate Phase Activities*, respectively. It is understood that specific reference designs will augment this list with their required and preferred tools for load balancing and scaling.

Table 15: Operate Phase Tools

Tool	Features	Benefits	Inputs	Outputs	Baseline
Backup management	Data backup System components (VM or container) snapshot	Improve failure recovery	Access to the backup source	Backup data System VM or container snapshot	REQUIRED
Operations dashboard	Provide operators a visual view of operations status, alerts, and actions.	Improve operations management	All operational monitoring status, alerts, and recommended actions	Dashboard display	PREFERRED

Table 16: Operate Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependency
Backup	Data backup; System backup	Access to backup system	Backup data or image	Backup management; Database automation tool
Scale	Scale manages VMs/containers as a group. The number of VMs in the group can be dynamically changed based on the demand and policy.	Real-time demand and VM performance measures Scale policy (demand or Key Performance Indicator (KPI)threshold; minimum, desired, and maximum number of VMs/containers)	Optimized resource allocation	VM management capability on the hosting environment;
Load balancing	Load balancing equalizes the resource utilization	Load balance policy Real time traffic load and VM/container performance measures	Balanced resource utilization	VM management capability on the hosting environment;
Feedback	The Second Way: Feedback	Technical feedback as to “is the system built right” and operational feedback as to “was the right system built”	Updated requirements / backlog	Various planning tools

Monitor Tools and Activities

In the monitor phase, tools are utilized to collect and assess key information about the use of the application to discover trends and identify problem areas. Monitoring spans the underlying hardware resources, network transport, applications / microservices, containers, interfaces, normal and anomalous endpoint behavior, and security event log analysis.

NIST SP 800-137 defines "information security continuous monitoring (ISCM) as maintaining ongoing awareness of information security, vulnerabilities, and threats to support organizational risk management decisions."[1] It continuously inventories all system components, monitors the performance and security of all components, and logs application and system events. Other policy enforcement and miscellaneous considerations include:

- Policy enforcement, including ensuring hardening of CSP managed services as measured against NIST SP 800-53.
- Policy enforcement, including ensuring compliance of COTS against STIGs.
- Zero Trust concepts, including bi-directional authentication, Software Defined Perimeter (SDP), micro-segmentation with authenticated and authorized data flows, separation of duties, and dynamic authorization to provide secure access from untrusted environments.
- A logging agent on each resource to push logs to a centralized logging service. Log analysis should be performed using a Security Information and Event Manager (SIEM) / Security Orchestration Automation and Response (SOAR) capability.

Monitor phase tools and their related activities are listed in *Table 17: Monitor Phase Tools* and *Table 18: Monitor Phase Activities*, respectively.

[1] NIST, *NIST SP 800-137, Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations,* 2011.

Table 17: Monitor Phase Tools

Tool	Features	Benefits	Inputs	Outputs	Baseline
Compliance Monitor	Monitor the state of compliance of deployed cloud resources and services against NIST SP 800-53 controls				REQUIRED
Compliance as Code	Monitor the state of compliance of deployed COTS against STIGs				PREFERRED
Logging	Logging events for all user, network, application, and data activities	Assist troubleshooting the issues. Assist detection of advanced persistent threats and forensics.	All user, network, application, and data activities	Event logs	REQUIRED
Log aggregator	Filter log files for events of interest (e.g., security), and transform into canonical format		Event Logs, Database Logs, Audit Logs, Database Security Audit logs	Aggregated, filtered, formatted event log	REQUIRED
Log promotion	Filter log files for events of interest (e.g., security), and transform into canonical format before pushing the logs to DoD Common Security Services		Event logs, database logs, audit logs, security audit logs	Aggregated, filtered, formatted event log record	REQUIRED
Log analysis	Analyze and audit to detect malicious threats / activity;		Logs	Alert messages, emails, etc. Remediation report and log	REQUIRED

Tool	Features	Benefits	Inputs	Outputs	Baseline
	Automated alerting and workflows for response Forensics for damage assessment. These are typically SIEM and SOAR tools.				
Log auditing	Audit to ensure possession of the logs and that aggregation is performed correctly		Logs	Audit Logs	REQUIRED
Operations monitoring	Report various performance metrics such as resource utilization rates, number of concurrent user sessions, and Input/Output (IO) rates; Provide dashboards to display performance; Alert performance issues Establish a baseline for comparison	Improve operations continuity Identify the area to improve Better end-user experience	Performance KPI and Service Level Agreement (SLA)	Performance statistics Performance alerts	REQUIRED
InfoSec Continuous Monitoring (ISCM)	Monitor network security Monitor personnel activity Monitor configuration changes Perform periodical security scan to all system components Monitor the IT assets and detect deviations	Detect unauthorized personnel, connections, devices, and software Identify cybersecurity vulnerability Detect security and	IT asset Network Personnel activities Known vulnerabilities	Vulnerabilities Incompliance Findings, assessments and recommendations	REQUIRED

Tool	Features	Benefits	Inputs	Outputs	Baseline
	from security, fault tolerance, performance best practices. Monitor and analyze log files Audit IT asset's configuration compliance Detect and block malicious code Continuous security vulnerability assessments and scans Provide browse, filter, search, visualize, analysis capabilities Generate findings, assessments and recommendations. Provide recommendations and/or tools for remediating any non-compliant IT asset and/or IT workload.	compliance violation Verify the effectiveness of protective measures			
Cyber Threat Continuous Monitoring	Varying set of tools, from actor activity based detection, tech stack, etc.	Helps with risked-based decisions in a proactive manner in lieu of reactivity when new vulnerabilities are announced	Cyber threat condition feeds	Recommend changes in CSRP	PREFERRED

Tool	Features	Benefits	Inputs	Outputs	Baseline
Alerting and notification	Notify security teams and/or administrators about detected events. Support automatic remediation of high-priority time-critical events.	Improve visibility of system events Reduce system downtime Improve customer service	Aggregated filtered logs from the Log Aggregator, vulnerability and non-compliance findings from Information Security Continuous Monitoring, recommendations from Information Security Continuous Monitoring, performance statistics from Operations Monitoring, and performance alerts from Operations Monitoring	Alert messages, emails, etc. Remediation report Issue ticket	REQUIRED
Database monitoring tool	Baseline database performance and database traffic; Detect anomalies	Improve database operations continuity	Running database	Logs; Warnings and alerts	PREFERRED if using a database
Database security audit tool	Perform user access and data access audit; Detect anomalies from events correlation; Detect SQL injection; Generate alert	Enhance database security	Running database	Audit logs; Warnings and alerts	REQUIRED if using a database

Table 18: Monitor Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependencies
Compliance Monitoring (resources & services)	Monitor the state of compliance of deployed cloud resources and services against NIST SP 800-53 controls			Compliance Monitor
Compliance Monitoring (COTS)	Monitor the state of compliance of deployed COTS against STIGs			Compliance as Code
Logging	Log system events	All user, network, application, and data activities	Logs	Logging
Log analysis	Filter or aggregate logs; Analyze and correlate logs	Logs	Alerts and remediation report	Log aggregator Log analysis & auditing
Log auditing	Ensure possession of the logs and that aggregation is performed correctly	Logs	Report	Log aggregator Log analysis & auditing
System performance monitoring	Monitor system hardware, software, database, and network performance; Baselining system performance; Detect anomalies	Running system	Performance KPI measures; Recommended actions; Warnings or alerts	Operation monitoring Issue tracking system; Alerting and notification; Operations dashboard
System Security monitoring	Monitor security of all system components Security vulnerability assessment System security compliance scan	Running system	Vulnerabilities; Incompliance Findings; assessments and recommendations; Warnings and alerts.	ISCM; Issue tracking system; Alerting and notification; Operations dashboard
Asset Inventory	Inventory system IT assets	IT assets	Asset inventory	Inventory Management;
System configuration monitoring	System configuration (infrastructure components and software) compliance checking, analysis, and reporting	Running system configuration; Configuration baseline	Compliance report; Recommended actions; Warnings and alerts	ISCM; Issue tracking system; Alerting and notification; Operations dashboard
Database monitoring and security auditing	Database performance and activities monitoring and auditing	Database traffic, event, and activities	Logs; Warnings and alerts	Database monitoring tool; Database security audit tool; Issue tracking system; Alerting and notification; Operations dashboard

Configuration Management Tools and Activities Cross-Reference

Configuration management plays a key role in DevSecOps practices. Without configuration management discipline, DevSecOps practices will not reach their full potential. CM ensures the configuration of a software system's infrastructure, software components, and functionalities are known initially and well-controlled and understood throughout the entirety of the DevSecOps lifecycle.

CM consists of three sets of activities:

- Configuration Identification: Identify the configuration items. This can be done manually or with assistance from a discovery tool. The configuration items include infrastructure components, COTS or open source software components used in the system, documented software design, features, software code or scripts, artifacts, etc.
- Configuration Control: Control the changes of the configuration items. Each configuration item has its own attributes, such as model number, version, configuration setup, license, etc. The CMDB, source code repository, and artifact repository are tools to track and control the changes. The source code repository is used primarily during development. The other two are used in both development and operations.
- Configuration Verification and Audit: Verify and audit that the configuration items meet the documented requirements and design. Configuration verification and audit are control gates along a pipeline to control the go/no-go decision to the next phase.

These configuration management activities are represented in *Table 19: Configuration Management Activities Summary and Cross-Reference.*

Table 19: Configuration Management Activities Summary and Cross-Reference

Activities	Phase	Activities Table Reference	Tool Dependencies	Tool Table Reference
Configuration management planning	Plan	Table 4	Team collaboration system; Issue tracking system	Table 3
Configuration identification	Plan	Table 4	CMDB	Table 3
Design review	Plan	Table 4	Team collaboration system	Table 3
Documentation version control	Plan	Table 4	Team collaboration system	Table 3
Code review	Develop	Table 6	Code quality review tool	Table 5
Code Commit	Develop	Table 6	Source code repository	Table 5
Store artifacts	Build	Table 8	Artifact repository	Table 7
Build phase configuration control and audit	Build	Table 8	Team collaboration system; Issue tracking system	Table 3
Test phase configuration control and audit	Test	Table 10	Team collaboration system; Issue tracking system	Table 3
Infrastructure provisioning automation	Deploy	Table 14	Configuration automation tool	Table 13
Post-deployment security scan	Deploy	Table 14	Security compliance tool	Table 9
Post-deployment checkout	Deploy	Table 14	Test scripts	
Asset inventory	Monitor	Table 18	Asset inventory tool	Table 17
System performance monitoring	Monitor	Table 18	Operation monitoring Issue tracking system; Alerting and notification; Operations dashboard	Table 3 Table 15 Table 17
System configuration monitoring	Monitor	Table 18	ISCM; Issue tracking system; Alerting and notification; Operations dashboard	Table 3 Table 15 Table 17