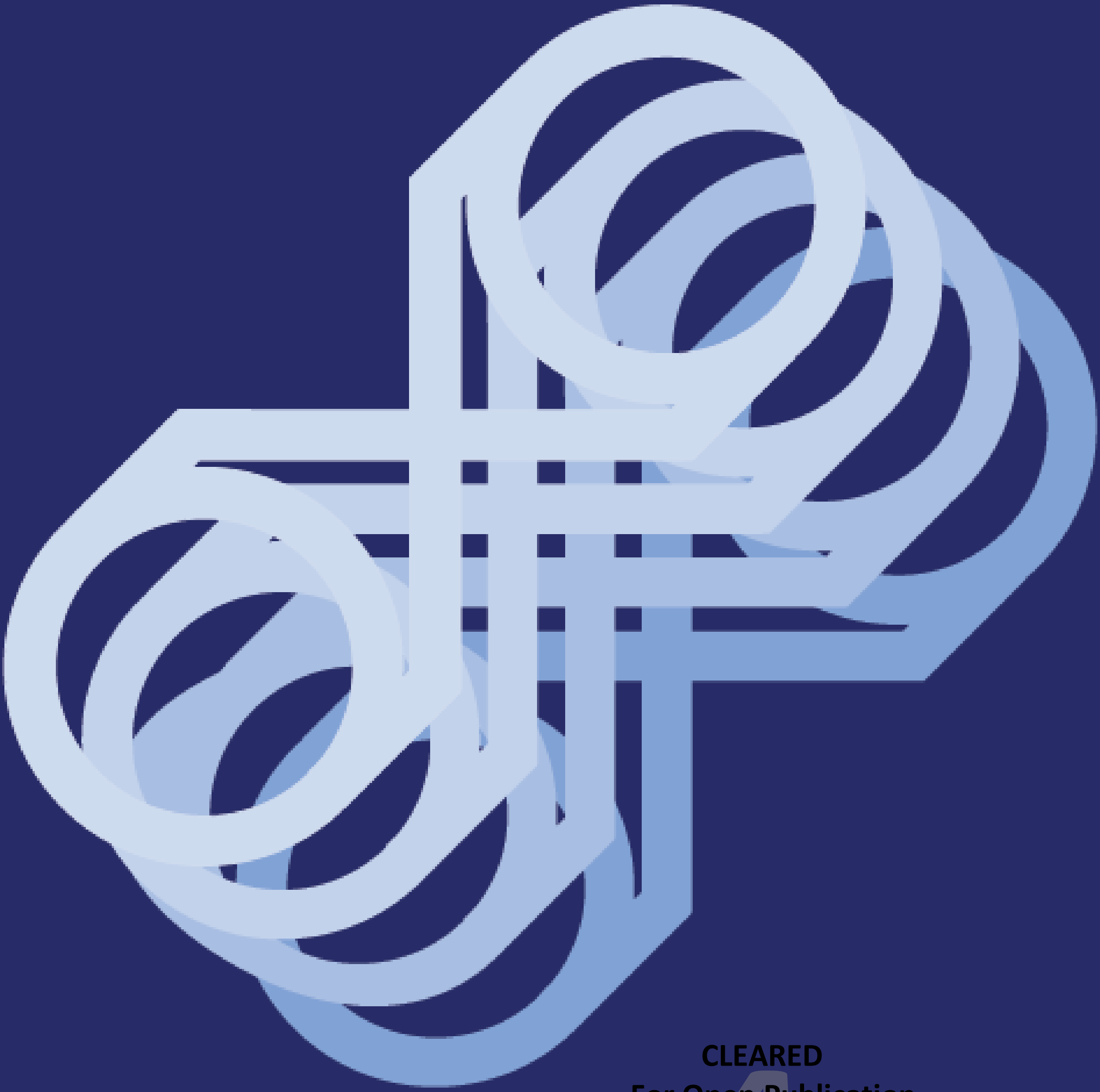
DoD Enterprise DevSecOps

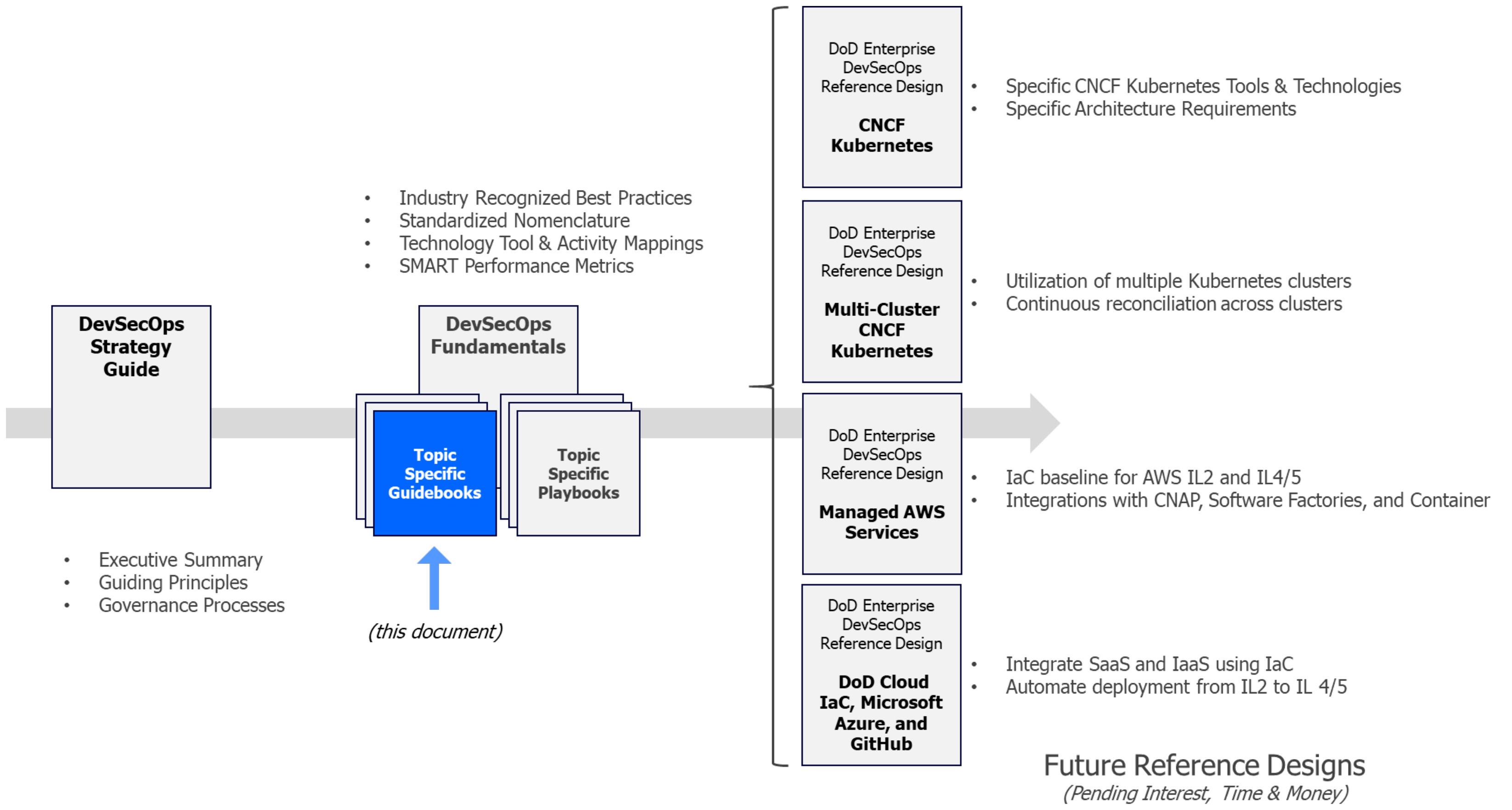
Activities & Tools Guidebook



April 2025
Version 2.5

Trademark Information

Document Set Reference



DevSecOps Strategy Guide
- Executive Summary
- Guiding Principles
- Governance Processes
- Industry Recognized Best Practices
- Standardized Nomenclature
- Technology Tool & Activity Mappings
- SMART Performance Metrics
DevSecOps Fundamentals
Topic Specific Guidebooks
Topic Specific Playbooks
(this document)
DoD Enterprise DevSecOps Reference Design
CNCF Kubernetes
- Specific CNCF Kubernetes Tools & Technologies
- Specific Architecture Requirements
DoD Enterprise DevSecOps Reference Design
Multi-Cluster CNCF Kubernetes
- Utilization of multiple Kubernetes clusters
- Continuous reconciliation across clusters
DoD Enterprise DevSecOps Reference Design
Managed AWS Services
- IaC baseline for AWS IL2 and IL4/5
- Integrations with CNAP, Software Factories, and Container
DoD Enterprise DevSecOps Reference Design
DoD Cloud IaC, Microsoft Azure, and GitHub
- Integrate SaaS and IaaS using IaC
- Automate deployment from IL2 to IL 4/5
Future Reference Designs
(Pending Interest, Time & Money)

Contents

Figures

Tables

1 Introduction

Practicing DevSecOps requires a wide range of activities supported by an array of purpose-built tools. This document and the associated spreadsheet convey the relationship between each DevSecOps phase, the set of activities that occur at each phase, and a taxonomy of supporting tools for a given phase. While the document and spreadsheet are published separately, they should be considered a single package. The document is descriptive while the spreadsheet includes the details of each phase and activity. It is expected that the spreadsheet also be used as a template for:

- submitting new reference designs that include an activity mapping to specific elements of the design
- building a continuous ATO package that demonstrates process and responsibility mapping

The spreadsheet It will also be the basis for future system modeling tools that further expand the details that occur at each activity.

1.1 Audience and Scope

The target audience for these documents include:

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

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

1.2 Baselines and Tailoring

The contents of these documents are generalized and include guidance for all DevSecOps practitioners. Adopters of DevSecOps are encouraged to tailor the content of these documents to their specific requirements and mission needs, combining the activities and tools included in these documents to build their processes.

The Baseline column of each Activities table provides guidance for whether a specific Activity is REQUIRED, PREFERRED, or AS REQUIRED. REQUIRED activities are not negotiable. PREFERRED and AS REQUIRED may be included at the discretion of the Mission Owner and Authorizing Official.

The complexity or frequency of DevSevOps activities may also be scaled upward or downward along a continuum to fit the specific needs of each adopting organization. Larger organizations, or those with a lower risk tolerance, may choose to perform more or more frequent activities, while other organizations may choose to perform activities less frequently.

Lastly, each adopter of DevSecOps may manage their work using different approaches or methodologies. DevSecOps is most frequently adopted using Scrum[1] and 2 week Sprints, but may also be successfully adopted using a different Scrum cadence, Kanban[2], or other Agile approach. Waterfall project management approaches are not often flexible enough for the iterative nature of DevSecOps lifecycles, and often require significant effort to effectively manage the volatility expected from fast-moving DevSecOps programs.

[1] Agile Alliance, "Scrum", https://www.agilealliance.org/glossary/scrum
[2] Agile Alliance, "Kanban", https://www.agilealliance.org/glossary/kanban

2 DevSecOps Lifecycle and Infinity Loop

The DevSecOps software lifecycle is an iterative process addressing the reality that **software is never done**. The traditional approach for DOD software, also known as Waterfall, is replaced with frequent minimally viable deliveries that receive feedback and enable course correction based on production experience. Each delivery is accomplished through a fully automated or semi-automated process with minimal human intervention. This ensures consistency and accelerates integration and delivery.

The DevSecOps software lifecycle is most often represented as an infinity loop, depicted in Figure 1. This representation emphasizes that the software development follows a cyclic process overall yet includes sub-cycles for dev and op phases that iterate independently. Transitions between phases are gated by functional and non-functional test and cybersecurity activities. The infinity loop includes 10 specific phases: plan, develop, build, test, release, deliver, deploy, operate, monitor, and feedback. Some descriptions combine the Release and Deliver phases into a single phase and some combine Plan and Feedback into a single phase. We call out Release and Deliver separately to highlight potential transitions between ATO boundaries in DoD software factories. We call out Plan and Feedback for the same reason, although the feedback phase typically uses the same tools and is managed like the plan phase.

Each phase has activities that are performed within the phase, including specific testing and cyber security activities applicable for the phase.

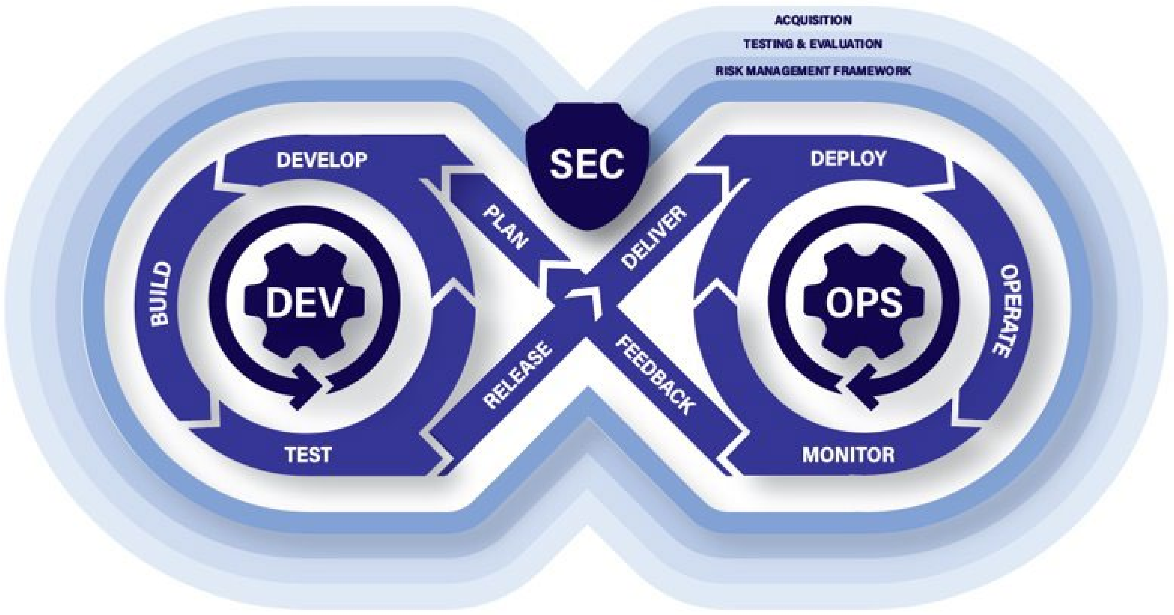


Figure 1: DevSecOps Distinct Lifecycle Phases and Philosophies

The allure of DevSecOps is improved customer outcomes and mission value through modern tools and technologies combined with automated processes that aid in the *delivery of software at the speed of relevance*. Automation and monitoring occur at every phase of the software lifecycle. Improving speed and quality is a primary goal of the DoD's software modernization effort.

Successful DevSecOps programs adopt an agile software **culture and philosophy** that is realized through the unification of software development (Dev), security (Sec) and operations (Ops) personnel into a singular team. Teams new to DevSecOps are encouraged to start small and build up their capabilities, progressively, striving for continuous process improvement at each of the lifecycle phases.

3 DevSecOps Phases and Activities

The activities in the DevSecOps Activities and Tools Guidebook are common across all DevSecOps ecosystems. An aggregated list of tools for the DevSecOps activities is available in the spreadsheet under the Tools tab.

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 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
- **Baseline**: Either a status of REQUIRED, PREFERRED, or AS REQUIRED, where required indicates that the activity must be performed within the software factory as part of the Minimal Viable Product (MVP) release, and preferred indicates an aspirational capability obtained as the ecosystem matures
- **SSDF**: NIST 800-218[3] Secure Software Development Framework (SSFD) alignment
- **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

Specific reference designs may elevate a specific activity from PREFERRED to REQUIRED, as well as add additional activities and/or tools that specifically support the nuances of a given reference design. Reference designs cannot lower an activity listed in these documents from REQUIRED to PREFERRED. However, contract, requirement, or reference design may make an activity "AS REQUIRED" or NOT APPLICABLE".

[3] National Institute for Standards and Technology (NIST), "Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities", February, 2022, https://csrc.nist.gov/publications/detail/sp/800-218/final

NIST Special Publication 800-218: Secure Software Development Framework (SSDF), Version 1.1 has been integrated through the phases enumerated in this guidebook. The phases that follow Release and Delivery are also covered by different NIST standards, such as SP 800-53[4]. This guidebook has applied the parenthetical notation used by the SSDF:

- Prepare the Organization (PO)
- Protect the Software (PS)
- Produce Well-Secured Software (PW)
- Respond to Vulnerability (RV)

Each *Activity* or *Tool* row has been notated with applicable secure software development practices.

As noted in NIST SP 800-218[5]: *Organizations should adopt a risk-based approach to determine what practices are relevant, appropriate, and effective to mitigate the threats to their software development practices*. The additional rows incorporated in the tables are one possible application of SP 800-218. Each software factory should interpret these as a starting point, not a destination. SP 800-218 is an outcome-based framework, and so there is no singular way of interpreting and applying the guidance found in this important framework!

[4] National Institute for Standards and Technology (NIST), "Security and Privacy Controls for Information Systems and Organizations", December 10, 2020, https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final
[5] National Institute for Standards and Technology (NIST), "Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities", February, 2022, https://csrc.nist.gov/publications/detail/sp/800-218/final

3.1 Continuous Activities Cross-References

While there are many activities that are localized to a single phase, there are other activities that occur repeatedly over multiple phases of the DevSecOps lifecycle for Security, Testing, and Configuration Management. These tables aggregate those continuous activities.

3.1.1 Security Activities Cross-References

Security is integrated into the core of the DevSecOps phases, woven into the fabric that touches each phase depicted in Figure 1. This approach to security facilitates automated risk characterization, monitoring, and risk mitigation across the application lifecycle. The Security Activities section, represented here as Table 1, of the Continuous Activities tab summarizes this security posture by representing all the security activities, the linked DevSecOps phase, and the activities and tools references.

The operations ("Ops") segment 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 ten DevSecOps Software Development Life Cycle (SDLC) phases. Integration with these capabilities must be considered at every phase in order to properly practice DevSecOps. This requirement substantially differentiates DevSecOps from legacy software development approaches, where integration was done "after the fact" using a "bolt-on" mentality.

Table 1: Continuous Security Activities

Security Activities	**Phase**	**Cyber Tool Dependencies**
Mission Based Cyber Risk Assessments	All Phases	Mission Cyber Risk Assessment tool
Threat modeling	Plan	Threat modeling tool
Code commit scan	Develop	Source code repository security plugin
Security code development	Develop	IDE
Static code scan before commit	Develop	IDE security plugins
Dependency vulnerability checking	Build	Dependency checking / BOM checking tool
Static application security test and scan (SAST)	Build, Test	SAST tool
Database security test	Test	Security compliance tool
Dynamic application security test and scan (DAST)	Test	DAST tool or IAST tool
Interactive Application Security Tests (IAST)	Test	DAST tool or IAST tool

Manual security testing (such as penetration test)	Test	Various tools and scripts (may include network security test tool)
Service security test	Test	Security compliance tool
Post-deployment security scan	Deploy	Security compliance tool
Compliance Monitoring (resources & services)	Monitor	Compliance tool; Operational dashboard
Compliance Monitoring (COTS)	Monitor	Compliance tool; Operational dashboard
Database monitoring and security auditing	Monitor	Security compliance tool
Runtime Application Security Protection (RASP)	Monitor	Security compliance tool
System Security monitoring	Monitor	Information Security Continuous Monitoring (ISCM)
SBOM Software Composition Analysis	Post-Build	SBOM & Software Factory Risk Continuous Monitoring tool
Software Factory Risk Continuous Monitoring	Post-Build	SBOM & Software Factory Risk Continuous Monitoring tool
API Security Tests	Build	API Security Test tool
Cooperative & Adversarial Tests	Operate	Cooperative & Adversarial Test tool
Persistent Cyber Operations Tests	Operate	Persistent Cyber Operations test tool
Chaos engineering	Operate	Chaos engineering tool

3.1.2 Test Activities and Tools Cross-References

Testing is integrated into the core of all DevSecOps phases. And like Security, testing is woven into the fabric that touches each phase depicted in Figure 1. This approach to testing facilitates automated evaluation of functional and operational performance across the application lifecycle. The Testing Activities section of the Continuous Activities tab, represented here as Table 2, summarizes this testing focus by representing all the testing activities, the linked DevSecOps phase, and the activities and tools references.

Expanding on DODI 5000.89, Test and Evaluation[6], the DOT&E Software Developmental Test and Evaluation in DevSecOps Guidebook[7] emphasizes that “T&E strategy, planning, execution and analysis must adapt to support the rapid iterative framework of DevSecOps regardless of acquisition pathway.”

[6] Office of the Under Secretary of Defense for Research and Engineering & Office of the Director, Operational Test and Evaluation. DODI 5000.89, Test and Evaluation, https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500089p.PDF, November 19, 2020
[7] Office of the Under Secretary of Defense for Research and Engineering, Software Developmental Test and Evaluation in DevSecOps Guidebook, https://www.cto.mil/wp-content/uploads/2025/01/Software_DTE_DEVSECOPS_GB_Jan2025_Signed.pdf, January 2025

Additional guidance is provided in the Joint Interoperability Test Center (JITC) Guidebook for DevSecOps.[8]

Table 2: Continuous Test Activities

Test Activities	**Phase**	**Test Tool Dependencies**
Test Audit	All Phases	Test management tool
Test Deployment	Plan, Develop	Configuration automation tool; IaC
Test Plan	Plan	Test tool suite, Work management system
Unit Test	Develop	Test tool suite; Test coverage tool
Component Test	Develop	Static analysis tool
Dynamic Analysis	Develop	Dynamic analysis tool
Service Component Test	Develop	Service functional test tool
Database Component Test	Develop	Database test tools
Regression Test (includes smoke tests)	Build, Test	Test tool suite
Software Integration Test	Build	Test report about whether the integrated units performed as designed.
System Test	Build	Test result about if the system performs as designed.
Functional Test	Test	Functional test tool
Integration Test	Test	Includes both System & Sub-System
Mission-Oriented Developmental Tests	Test	Test tool suite
Performance Test	Test	Test tool suite; Test data generator
Acceptance Test	Test	Test tool suite; Non-security compliance scan
Compliance Scan	Test/Deliver	Non-security compliance scan; Software license compliance checker; Security compliance tool
Development Tests	Release	Development testing tool
User Story Review and Demonstration	Release	Work management system

[8] Joint Interoperability Test Center, JITC Guidebook for DevSecOps, https://jitc.fhu.disa.mil/organization/references/publications/downloads/JITC_Guidebook_for_DevSecOps_v1.0_August_2021.pdf, 06 August 2021

Operations Team Acceptance	Deliver	n/a
Configuration Integration Testing	Deliver	Configuration integration test tool
Operations Tests	Deliver	Operations testing tool
Post-deployment checkout	Deploy	Test scripts
Operational Test and Evaluation	Deploy	Operational Test and Evaluation tool
User Evaluation / Feedback	Post Deploy	n/a
Sustainment and Chaos Testing	Operate	Sustainment and Chaos Testing tool
Test Configuration Audit	Monitor	Track test and security scan results

3.1.3 Configuration Management Full Lifecycle Activities

Configuration management (CM) 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, libraries, 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 Configuration Management Database (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 CMDB and artifact repository 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 included in the phase-specific activity tables, but also consolidated in the Configuration Management section of the Continuous Activities tab, represented here as Table 3.

Table 3: Continuous Configuration Management Activities

Configuration Management Activities	Phase	Tool Dependencies
Configuration management planning	Plan	Team collaboration system; Issue tracking system
Configuration identification	Plan	CMDB; Source code repository; Artifact repository; Team collaboration system
Design review	Plan	Team collaboration system
Documentation version control	Plan	Team collaboration system
Code commit	Develop	Source code repository
Code review	Develop	Code quality review tool
Store artifacts	Build	Artifact repository
Build configuration control and audit	Build	Team collaboration system; Issue tracking system; CI/CD orchestrator
Test Audit	Test	Test management tool
Test configuration control	Test	Team collaboration system; Issue tracking system; CI/CD orchestrator
Infrastructure provisioning automation	Deploy	Configuration automation tool; IaC
Asset inventory to include SBOMs	Monitor	Inventory Management
System performance monitoring	Monitor	Operation monitoring; Issue tracking system; Alerting and notification; Operations dashboard
System configuration monitoring	Monitor	ISCM; Issue tracking system; Alerting and notification; Operations dashboard

3.2 Plan Phase Activities

Software development planning activities include configuration management planning, change management planning, project management planning, system design, software design, test planning, and security planning. Planning tools support software development activities. 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.

The activities supported by the plan phase are listed in the Plan tab of the Activities & Tools Guidebook spreadsheet, represented here as Table 4. 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 4: Plan Phase Activities

Activities	Baseline	SSDF	Description	Inputs	Outputs	Tool Dependencies	Security / Testing / CM
Change management planning	REQUIRED	PO.1.1, PS.1.1, PS.3.1, PW.6.1	Plan the change control process	Organizational policy; Software development best practices	Change control procedures; Review procedures; Control review board; Change management plan	Team collaboration system; Issue tracking system	
Configuration identification	REQUIRED	PO.2.1, PS.1.1, PW.2.1, PW.4.1, PW.4.2, PW.6.2	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
Configuration management (CM) planning	REQUIRED	PO.3.1, PO.3.3, PO.4.1, PO.4.2, PW.2.1	Plan the configuration control process; Identify configurations 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	Configuration Management
Database design	PREFERRED	PO.1.2, PO.3.1, PO.5.2, PW.1.1, PW.5.1	Data modeling; Database selection; Database deployment topology	System requirement; System design	- Database design document -	Data modeling tool; Teams collaboration system	

Design review	PREFERRED	PO.1.2, PW.1.2, PW.2.1, PW.8.2, RV.2.2	Review and approve plans and documents	Plans and design documents;	Review comments; Action items	Team collaboration system	Configuration Management
DevSecOps process design	REQUIRED	PO.1.1	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	
Documentation version control	REQUIRED	PO.1.1, PO.1.2, PO.1.3, PS.1.1	Track design changes	Plans and design documents;	Version controlled documents	Team collaboration system	Configuration Management
IaC deployment	REQUIRED	PO.3.2, PO.3.3	Deploy infrastructure and set up environment using Infrastructure as Code	Artifacts (Infrastructure as Code) Infrastructure as Code	The environment ready	Configuration automation tool; IaC	
Mission-Based Cyber Risk Assessments	REQUIRED	PW.7.2, RV.1.1, RV.1.2, RV.2.1, RV.3.1, RV.3.2, RV.3.3,	An assessment of risks based upon the stated mission of the system or platform	NIST 800-53 RMF Control Implementations FIPS 199 system categorization Information types	Risk assessment	Risk assessment tool	Security
Project/Release planning	REQUIRED	PS.3.1, PS.3.2	Project task management Release planning	Project charter Project constraints	Project Plan Task plan & schedule Release plan & schedule	Team collaboration system; Project management system	

Project team onboarding planning	REQUIRED	PO.2.1, PO.2.2, PO.2.3	Plan the project team onboarding process, interface, access control policy	Organization policy	Onboarding plan	Team collaboration system	
Risk management	REQUIRED	PO.1.2, PO.3.1, PO.4.1, PW.1.1, PW.1.2, PW.2.1, RV.2.1	Risk assessment	System architecture; Supply chain information; Security risks	Risk management plan	Team collaboration system;	
Software requirement analysis	REQUIRED	PO.1.1, PO.1.2, PO.1.3	Gather the requirements from all stakeholders	Stakeholder inputs or feedback; Operation monitoring feedback; Test feedback	Requirements Documents: - Feature requirements - Performance requirements - Privacy requirements - Security requirments	Requirements tool; Team collaboration system; Issue Tracking system	
System design	REQUIRED	PO.1.1, PO.1.2, PO.1.3	Design the system based on the requirements	Requirements database or documents	System Design Documents: - System architecture - Functional design - Data flow diagrams - Acceptance Criteria - Infrastructure configuration plan - Tool selections - Ecosystem Tools: - Development tool - Test Tool - Deployment platform	Team collaboration system; Issue tracking system; Software system design tools	

Test Audit	REQUIRED	PO.2.1, PS 2.1, PW.1.2, PW.2.1	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	Testing
Test Deployment	REQUIRED	PW.1.3	Deploy test infrastructure	Test environment applications and infrastructure	Test environment instrumentation	Testing tool, Team collaboration system	Testing
Test Plan	REQUIRED	PO.1.1, PO.1.2, PO.1.3, PW.8.1, PW.8.2	Plan testing, including validating Acceptance Criteria, specifying Component Tests, Component integration Tests, and Software Integration Tests	Requirements database or documents, system design	Test plans	Testing tool, Team collaboration system	Testing
Threat modeling	PREFERRED	PW.1.1, PW.2.1, RV.2.1	Identify potential threats, weaknesses and vulnerabilities. Define the mitigation plan	System design	Potential threats and mitigation plan	Threat modeling tool	Security

3.3 Develop Phase Activities

Develop phase activities, listed on the Develop tab and represented here as Table 5, mainly convert requirements into source code and are supported by a variety of tools. 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 Continuous Integration/Continuous Deployment (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.

Table 5: Develop Phase Activities

Activities	Baseline	SSDF	Description	Inputs	Outputs	Tool Dependencies	Security / Testing / CM
Application code development	PREFERRED	PO.1.2, PO.3.1	Application coding	Developer coding and appropriate unit, integration, etc. testing input	Source code & test results	IDE	
Code commit	REQUIRED	PS.1.1, PW.4.4	Commit source code into version control system	Source code	Version controlled source code	Source code repository	Configuration Management
Code Commit Logging	REQUIRED	PO.3.1, PO.3.3, PO.5.1, PO.5.2	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	Logging tool	
Code commit scan	REQUIRED	RV.1.2	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	Security

Code review	PREFERRED	PW.7.1, PW.7.2	Perform code review to all source code. Note that pair programming counts.	Source code	Review comments	Code quality review tool	Configuration Management
Component Test	PREFERRED	RV.8.1, RV.8.2	Closed-box testing, evaluating the behavior of the program without considering the details of the underlying code	Test plan Test cases Test data	Test results	Test tool suite, Test coverage tool	Testing
Database Component Test	REQUIRED	RV.8.1, RV.8.2	Closed-box database testing, evaluating the behavior of the database without considering the details of the underlying code	Test plan Test cases Test data	Test results	Test tool suite, Test coverage tool	Testing
Database development	PREFERRED	PO.3.1	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	
Database functional test (optional)	PREFERRED	PW.8.1, PW.8.2	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	Testing

Documentation	REQUIRED	PO.1.1, PO.1.2, PO.1.3, PW.7.2	Detailed implementation documentation	- User input; - Developed Source Code	- Documentation; - Auto generated Application Programming Interface (API) documentation	IDE or document editor or build tool	
Dynamic analysis	PREFERRED	RV.1.2, RV.2.1, RV.3.1, RV.3.2, RV.3.3	Dynamic code analysis involves running code and examining the outcome.	Source code, fuzz data	Various outcomes depending on the code execution	Dynamic code analysis tool	Testing
Functional test	REQUIRED	PW.8.1, PW.8.2	Functional tests determine if code is acting in accordance with the pre-determined requirements.	- Test data; - Test Scenarios	Text results	Functional testing tool	Testing
Infrastructure code development	PREFERRED	PO.5.1, PW.8.2, PW.9.1	System components and infrastructure orchestration coding Individual component configuration script coding. Includes Database deploy and high availability configuration	Developer coding and appropriate unit, integration, etc. testing input	Source code & test results	IDE	
Mission Based Cyber Risk Assessments	REQUIRED	PW.7.2, RV.1.1, RV.1.2, RV.2.1, RV.3.1, RV.3.2, RV.3.3,	An assessment of risks based upon the stated mission of the system or platform	NIST 800-53 RMF Control Implementations FIPS 199 system categorization Information types CNSSI 1253	Risk assessment	Risk assessment tool	Security

Security code development	REQUIRED	PO.1.2, PW.5.1, PW.6.1, PW.6.2	Security policy enforcement script coding	Developer coding and appropriate unit, integration, etc. testing input	Source code & test results	IDE	Security
Service functional test	PREFERRED	PW.8.1, PW.8.2	Perform unit test and functional test to services	- Test data; - Test Scenarios	Test results	Service testing tool	Testing
Static analysis	REQUIRED	RV.1.2, RV.2.1, RV.3.1, RV.3.2, RV.3.3	Static code analysis examines code to identify issues within the logic and techniques.	Source code	Static code analysis report	Static code analysis tool	Testing
Static code scan before commit	REQUIRED	PW.7.1, PW.7.2	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	Security
Test Audit	REQUIRED	PO.2.1, PS 2.1, PW.1.2, PW.2.1	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	Testing

Test development	REQUIRED	PW.9.1, PW.8.2	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
							Testing

Unit test	REQUIRED	PW.8.1, PW.8.2	Assist unit test script development and unit test execution. It is typically language specific. Whenever possible, the unit test should be automated.	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	Testing

3.4 Build Phase Activities

Activities in the Build phase, listed in the Build tab and represented here as Table 6, perform the tasks of building and packaging applications, services, and microservices into artifacts. For statically linked 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 "late binding" languages, such as 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 dynamically 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. In all cases, code quality tools that scan for well-known issues are highly recommended.

Table 6: Build Phase Activities

Activities	Baseline	SSDF	Description	Inputs	Outputs	Tool Dependencies	Security / Testing / CM
API Security Tests	REQUIRED	PO.3.2, PO.4.1, PW.1.3, PW.4.4, RV1.2	Closed-box test, evaluating the Application Programming Interfaces' compliance with security requirements	Executable system / application Test Plan Test Cases Test data	Test results	Test tool suite	Security

Build	REQUIRED	PO.3.1, PO.3.2, PO.3.3, PO.4.1	Compile and link	Source code; dependencies	Binary artifacts
					Build Report
						Build tool;
						Lint tool;
						Artifact repository	

Build configuration control and audit	REQUIRED	PS 3.2	Track build results, SAST and dependency checking report;	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	Configuration Management
Component Test	REQUIRED	RV.8.1, RV.8.2	Closed-box testing, evaluating the behavior of the program without considering the details of the underlying code	Test plan Test cases Test data	Test results	Test tool suite, Test coverage tool	Testing
Dependency vulnerability checking	REQUIRED	PO.3.1, PO.3.2, PW.4.4, RV.1.1, RV.1.2, RV.1.3	Identify vulnerabilities in the open source dependent components	Dependency list or BOM list	Vulnerability report	Dependency checking / BOM checking tool	Security
Functional test	REQUIRED	PW.8.1, PW.8.2	Functional tests determine if code is acting in accordance with the pre-determined requirements.	Test plan; Requirement documents and/or database; Acceptance criteria	Functional test scripts, the software units under test, test input, and expected output data	Test report documenting the performance of the integrated unit.	Testing
Integration test	REQUIRED	PW.8.1, PW.8.2	Develops the integration test scripts and execute the scripts to test one or more software units as a group with the interaction between the units as the focus.	Test plan; Requirement documents and/or database; Acceptance criteria; APIs for integrated systems	Integration test scripts, the software units under test, test input data, and expected output data	Test results documenting the functioning of the system.	Testing

Mission Based Cyber Risk Assessments	REQUIRED	PW.7.2, RV.1.1, RV.1.2, RV.2.1, RV.3.1, RV.3.2, RV.3.3,	An assessment of risks based upon the stated mission of the system or platform	NIST 800-53 RMF Control Implementations FIPS 199 system categorization Information types CNSSI 1253	Risk assessment	Risk assessment tool	Security
Regression Test (includes smoke tests)	REQUIRED	RV.8.1, RV.8.2	Regression testing is re-running functional and non-functional tests to ensure that previously developed and tested software still performs as expected after a change.	Test plan, Test cases, Test data	Test results	Test management tool	Testing
Release packaging	REQUIRED	PS.2.1, PS.3.1, PS.3.2	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	
Software Integration Test	REQUIRED	RV.8.1, RV.8.2	Tests where individual softwa re modules are combined and tested as a group	Test plan, Test cases, Test data	Test results	Test management tool	Testing
Static application security test and scan	REQUIRED	PO.3.1, PO.3.2, PO.3.3, PO.4.1, PO.4.2	Perform SAST to the software system	Source code; known vulnerabilities and weaknesses	Static code scan report and recommended mitigation.	SAST tool	Security

Store artifacts	REQUIRED	PO.3.1, PO.3.3	Store artifacts to the artifact repository	Binary artifacts; Database artifacts; Scripts; Documentation	Versioned controlled artifacts	Artifact Repository	Configuration Management
System test	PREFERRED	PW.8.1, PW.8.2	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.	Test plan; Requirement documents and/or database; Acceptance criteria	System test scripts, the software system and external dependencies, test input data and expected output data	Test results documenting the performance of the system.	Testing
Test Audit	REQUIRED	PO.2.1, PS 2.1, PW.1.2, PW.2.1	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	Testing

3.5 Test Phase Activities

Like security, testing is something that should be integrated in all the DevSecOps phases. With maturity, testing should become more and more automated. The Test Activities section of the Continuous Activities tab of the Activities and Tools Guidebook spreadsheet summarizes the list of testing activities and identifies the Phase in which that activity is done.

The discipline of testing changes within the automated processes of DevSecOps. Testing focuses on how the system supports the mission. 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 understanding the intent of the mission and how to test that using automation. The testers will need to become coders of that automation and be involved from the beginning of the DevSecOps process.

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.

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.
- Production stage: operational test and evaluation with mission users.

Test audit happens at all stages.

Table 7: Test Phase Activities

Activities	Baseline	SSDF	Description	Inputs	Outputs	Tool Dependencies	Security / Testing / CM
API Security Tests	REQUIRED	PO.3.2, PO.4.1, PW.1.3, PW.4.4, RV1.2	Closed-box test, evaluating the Application Programming Interfaces' compliance with security requirements	Executable system/application Test Plan Test Cases Test data	Test results	Test tool suite	Security

Compliance scan	REQUIRED	RV.1.2	Compliance audit	Artifacts; Software instances; System components	Compliance reports	Non-security compliance scan; Software license compliance checker; Security compliance tool	Testing
Database security test	PREFERRED	PW.8.1, PW.8.2, PW.9.2	Perform security scan; Security test	Test data; Test scenarios	Test results	Vulnerability findings; Recommended mitigation actions	Security
Dynamic application security test (DAST) and scan	PREFERRED	RV.1.2, RV.2.1, RV.3.1, RV.3.2, RV.3.3	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	Security
Interactive Application Security Tests (IAST)	PREFERRED	PO.4.1, PO.4.2, PS.2.1, PW.1.1, PW.5.1, PW.8.1, PW.8.2	IAST (interactive application security testing) analyzes code for security vulnerabilities while the app is run by an automated test, human tester, or any activity “interacting” with the application functionality.	Software instances, Test scenarios, Test data	Test results	IAST tool	Security
Manual security test	REQUIRED	PO.4.1, PO.4.2, PS.2.1, PW.1.1, PW.5.1, PW.8.1, PW.8.2	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.	Running application, underlying OS, and hosting environment	Vulnerability report and recommended mitigation	Various tools and scripts (may include network security test tool)	Security

			CI/CD orchestrator does not automate the test, but the test results can be a control point in the pipeline.				
Mission Based Cyber Risk Assessments	REQUIRED	PW.7.2, RV.1.1, RV.1.2, RV.2.1, RV.3.1, RV.3.2, RV.3.3,	An assessment of risks based upon the stated mission of the system or platform	NIST 800-53 RMF Control Implementations FIPS 199 system categorization Information types CNSSI 1253	Risk assessment	Risk assessment tool	Security
Performance test	PREFERRED	PO.3.1, PO.3.2, PO.3.3	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	Testing
Regression test	REQUIRED	PW.4.4, PW.7.2	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	Testing
SBOM Software Composition Analysis	REQUIRED	PS.3.2	Collect and analyze provenance data for all components of each release	Software Bill of Materials Vulnerabilities	Analysis report	SBOM Analysis tool	Security

Service security test	REQUIRED	PO.3.1, PO.3.2	Perform security scan; Security test	Test data; Test scenarios	Test results	Vulnerability findings; Recommended mitigation actions	Security
Software Factory Risk Continuous Monitoring	REQUIRED	PS.3.2	Monitor Software Factory controls	Software Factory controls	Alerts	Monitoring tool suite	Security
Software Integration Test	REQUIRED	RV.8.1, RV.8.2	Testing where software modules are integrated logically and tested as a group	Test data; Test scenarios	Test results	Test management tool	Testing
Static application security test (SAST) and scan	REQUIRED	RV.1.2, RV.2.1, RV.3.1, RV.3.2, RV.3.3	Perform Static Application Security Tests (SAST) on the software system.	Source code, known vulnerabilities and weaknesses	Static code scan report and recommended mitigation	SAST tool	Security
Suitability test	PREFERRED	n/a	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	Testing
System Test	REQUIRED	RV.8.1, RV.8.2	Test where the entire system is tested as a whole	Test data; Test scenarios	Test results	Test management tool	Testing
Test Audit	REQUIRED	PO.2.1, PS.2.1, PW.1.2, PW.2.1	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	Testing

3.6 Release Phase Activities

In the release phase, the activities for which are included in the Release tab and represented here as Table 8, the software artifacts are digitally signed to verify that they have passed build, all tests, and security scans. Then, they are 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 a 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.

Table 8: Release Phase Activities

Activities	Baseline	SSDF	Description	Inputs	Outputs	Tool Dependency	Security / Testing / CM
Artifacts replication	PREFERRED	PS.2.1, PS.3.1, PS.3.2, PW.4.1, PW.4.2	Replicate newly released artifacts to all regional artifact repositories	Artifacts	Artifacts in all regional artifact repositories	Artifacts repositories (release, regional)	

Developmental Cyber Tests	PREFERRED	PO.4.1, PO.4.2, PS.2.1, PW.2.1, PW.8.1, PW.8.2, RV.3.2, RV.3.4	Testing the system in development and operational environments	Known CVEs, privacy requirements, security requirements, and potential threats	Recommendati ons		Testing
Mission Based Cyber Risk Assessments	REQUIRED	PW.7.2, RV.1.1, RV.1.2, RV.2.1, RV.3.1, RV.3.2, RV.3.3	An assessment of risks based upon the stated mission of the system or platform	NIST 800-53 RMF Control Implementation s FIPS 199 system categorization Information types CNSSI 1253	Risk assessment	Risk assessment tool	Security
Operational Readiness Test	REQUIRED	n/a	Suitability and effectiveness test of the entire system	The tested system Supporting system Test data	Test report	Test tool suite, Non-security compliance scan	Testing
Release go / no-go decision	REQUIRED	PW.2.1, RV.3.4	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	
SBOM Software Composition Analysis	REQUIRED	PS.3.2	Collect and analyze provenance data for all components of each release	Software Bill of Materials Vulnerabilities	Analysis report	SBOM Analysis tool	Security

Software Factory Risk Continuous Monitoring	REQUIRED	PS.3.2	Monitor Software Factory controls	Software Factory controls	Alerts	Monitoring tool suite	Security
Test Audit	REQUIRED	PO.2.1, PS.2.1, PW.1.2, PW.2.1	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	Testing
User Story Review and Demonstration	PREFERRED	n/a	Review of the user story, including description and acceptance criteria, and a demonstration of the completed work.	User Story, artifact repository, and test environment.	Push go/no-go decision	Artifact repository, test environment	Testing

3.7 Deliver Phase Activities

In the deliver phase, the activities for which are in the Deliver tab and represented here as Table 9,the software artifacts are received and verified, ensuring that they have passed build, all tests, and security scans. 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 a 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 release stage artifact repository to promote 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. To increase deployment velocity, a remote operational region may have its own local artifact repository that replicates the authoritative artifact repository completely or partially. During release, a new artifact is pushed into the artifact repository and then replicated to other regional artifact repositories.

Once a product is released, Developmental Testing is completed.

Table 9: Deliver Phase Activities

Activities	Baseline	SSDF	Description	Inputs	Outputs	Tool Dependency	Security / Testing / CM
Configuration Integration Testing	REQUIRED	PO.4.1, PO.4.2, PS.2.1, PW.2.1, PW8.1, PW8.2	Testing the fully integrated system to ensure that it meets requirements	Accepted Release Package	Configuration Results		Testing
Deliver released artifacts	REQUIRED	PS.2.1, PS.3.1, PS.3.2, PW.4.1, PW.4.2	Push released artifacts to the artifact repository	Release package	New release in the artifact repository	Artifacts repository	
Delivery Results Review	REQUIRED	PO.3.3, RV.3.2, RV.3.4	Review of the release package and all associated artifacts, configuration results, and recommendations	Configuration results and Recommendations	Production Push go/no-go decision		
Mission Based Cyber Risk Assessments	REQUIRED	PW.7.2, RV.1.1, RV.1.2, RV.2.1, RV.3.1, RV.3.2, RV.3.3	An assessment of risks based upon the stated mission of the system or platform	NIST 800-53 RMF Control Implementations FIPS 199 system categorization Information types CNSSI 1253	Risk assessment	Risk assessment tool	Security

Operational Cyber Tests	PREFERRED	PO.4.1, PO.4.2, PS.2.1, PW.2.1, PW8.1, PW8.2, RV.3.2, RV.3.4	Testing the system in operational environments	Known CVEs, privacy requirements, security requirements, and potential threats	Recommendations		Testing
Operations Team Acceptance	REQUIRED	PO.4.1, PO.4.2, PS.2.1, PW.2.1, PW8.1, PW8.2	Testing on the delivered artifacts to ensure that they meet operational requirements	Release package; Test results; SBOM Software Composition Analysis;	Accepted release package		Testing
SBOM Software Composition Analysis	REQUIRED	PS.3.2	Collect and analyze provenance data for all components of each release	Software Bill of Materials Vulnerabilities	Analysis report	SBOM Analysis tool	Security
Software Factory Risk Continuous Monitoring	REQUIRED	PS.3.2	Monitor Software Factory controls	Software Factory controls	Alerts	Monitoring tool suite	Security
Test Audit	REQUIRED	PO.2.1, PS 2.1, PW.1.2, PW.2.1	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	Testing

3.8 Deploy Phase Activities

The dominant deployment options have historically been virtual machines and software containers. However, serverless Cloud Service Offerings (CSOs) have matured and are rapidly increasing in popularity. The tools used in the Deploy phase are environment and deployment stage dependent. Deploy phase activities and their related tools are listed in the Deploy tab of the Activities and Tools Guidebook spreadsheet and represented here in Table 10.

Once a product has reached the Deploy phase, Operational Testing begins.

Table 10: Deploy Phase Activities

Activities	Baseline	SSDF	Description	Inputs	Outputs	Tool Dependency	Security / Testing / CM

Artifact deployment	PREFERRED	PO.3.1, PO.3.2	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	

Artifact download	PREFERRED	PW.4.1, PW.4.2	Download newly release artifacts from the artifact repository	Artifact download request	Requested artifacts	Artifact repository	
Compliance Tests	REQUIRED	PO.3.1, PO.3.2, PO.3.3	Testing to determine whether a deliverable complies with the requirements of a specification, technical standard, contract, or regulation	Artifacts in the repository; Test scenarios; Data	Test results	Test management tool	Testing
Create linked clone of VM master image	PREFERRED	PW.4.1, PW.4.2	Instantiate VM by creating a link clone of parent VM with master image	VM parent New VM instance parameters	New VM instance	Virtualization Manager	

Database artifact deployment	PREFERRED	PO.3.1, PO.3.2	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	

Database Schema installation	PREFERRED	PO.3.1, PO.3.2, PO.3.3	Database software schema deploy/update.	Artifacts in the repository;
				Data
					Running database system	Artifact repository;
						Database automation tool;
						Data masking or encryption tool if needed	

Formal Qualification Tests	AS REQUIRED	PO.3.1, PO.3.2, PO.3.3	Tests to assess appropriateness or qualifications of deliverables with contractual requirements.	Artifacts in the repository; Test scenarios; Data	Test results	Test management tool	Testing
Human Systems Integration Tests	AS REQUIRED	PO.3.1, PO.3.2, PO.3.3		Artifacts in the repository; Test scenarios; Data	Test results	Test management tool	Testing
Infrastructure provisioning automation	PREFERRED	PO.3.1, PO.3.2, PO.5.1, PO.5.2	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	Configuration Management
Interoperability Tests	REQUIRED	PO.3.1, PO.3.2, PO.3.3	Testing of the communication and cooperation between multiple software and system capabilities.	Artifacts in the repository; Test scenarios; Data	Test results	Test management tool	Testing
Interoperability Certifications	AS REQUIRED	PO.3.1, PO.3.2, PO.3.3	Interoperability Certification, especially relevant for flight and weapon systems				Testing
Mission Based Cyber Risk Assessments	REQUIRED	PW.7.2, RV.1.1, RV.1.2, RV.2.1, RV.3.1, RV.3.2, RV.3.3,	An assessment of risks based upon the stated mission of the system or platform	NIST 800-53 RMF Control Implementations FIPS 199 system categorization Information types CNSSI 1253	Risk assessment	Risk assessment tool	Security

Mission-Oriented Developmental Tests	REQUIRED	PO.3.1, PO.3.2, PO.3.3	Tests to evaluate and validate the performance of new or existing systems, equipment, or technology in real-world mission scenarios. These tests are conducted to determine the ability of the system to meet its intended objectives and to identify any potential issues or areas for improvement.	Artifacts in the repository; Test scenarios; Data	Test results	Test management tool	Testing
Operational Assessments	AS REQUIRED	PO.3.1, PO.3.2, PO.3.3	Evaluation of a minimal capability fielding or early fielding capability in the unique theater of employment and/or for tailored use to support preliminary evaluation of operational effectiveness and suitability, while taking into equal consideration survivability and lethality effects		Assessment results		Testing
Operational Test and Evaluation	AS REQUIRED	PO.3.1, PO.3.2, PO.3.3	Independent test and evaluation of system effectiveness, suitability, and survivability to include cyber resilience by the actual mission user base.	All application artifacts, test plans and reports.	Reports of test results	Test management tool to capture operational data to include instrumentation (as needed).	Testing

Performance Tests	REQUIRED	PO.3.1, PO.3.2, PO.3.3	Testing that determine the stability, speed, scalability, and responsiveness of an application or system under a given workload.	Artifacts in the repository; Test scenarios; Data	Test results	Test management tool	Testing
Post-deployment checkout	PREFERRED	PW.4.1, PW.4.2	Run automated test to make sure the important functions of system are working, including Early Operational Test; Interoperability Test; Performance Test; and Compliance Test	Smoke test scenarios and test scripts	Test results	Test scripts	Testing
Post-deployment security scan	REQUIRED	PO.4.1, PO.4.2, PW.4.1, PW.4.2	System and infrastructure security scan	Access to system components and infrastructure components	Security vulnerability findings	Security compliance tool	Security
SBOM Software Composition Analysis	REQUIRED	PS.3.2	Collect and analyze provenance data for all components of each release	Software Bill of Materials Vulnerabilities	Analysis report	SBOM Analysis tool	Security
Software Factory Risk Continuous Monitoring	REQUIRED	PS.3.2	Monitor Software Factory controls	Software Factory controls	Alerts	Monitoring tool suite	Security
Test Audit	REQUIRED	PO.2.1, PS 2.1, PW.1.2, PW.2.1	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	Testing
User Evaluation / Feedback	REQUIRED	PO.4.2	Collect and analyze user evaluations and/or feedback	Surveys; Bug reports; Trouble tickets			Testing

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 must 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.

3.8.2 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.

3.8.3 Serverless Deployment

Many cloud providers offer serverless capabilities that utilize managed infrastructure to orchestrate application functions. Serverless computing leverages compute scaling, process interconnection, and messaging infrastructure within commercial clouds to simplify and accelerate application delivery, while potentially reducing cost to operate. Cloud-managed CI/CD pipelines may be configured to deliver functions or containers to the operational environments. DevSecOps with DoD Cloud IaC, Microsoft Azure, and GitHub Reference Design [9]provides an implementation of this capability.

3.9 Operate Phase Activities

Operate phase activities include 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 activities and their related tools are listed in the Operate tab of the Activities & Tools Guidebook tables and represented here as Table 11. It is understood that specific reference designs will augment this list with their required and preferred tools for load balancing and scaling.

Table 11: Operate Phase Activities

Activities	Baseline	SSDF	Description	Inputs	Outputs	Tool Dependency	Security / Testing / CM
Business Operations	REQUIRED	PO.3.1, PO.3.2, PO.3.3	Continuously manage resource usage and billing	Cost utilization reports, metrics	Optimized resource allocation	Operations dashboard	
Capacity Management	PREFERRED	PO.3.1, PO.3.2, PO.3.3	Continuously manage CSP service	Operational statistics, logs, metrics	Optimized resource allocation	Operations dashboard	

[9] DOD Office of the CIO and DISA Hosting and Compute Center (HaCC) with Microsoft Federal, “DevSecOps with DoD Cloud IaC, Microsoft Azure, and GitHub”, https://dodcio.defense.gov/Portals/0/Documents/Library/DoDRefDesignCloudGithub.pdf, 2022

			configuration parameters				
Chaos engineering	PREFERRED	PO.3.1, PO.3.2	Chaos Engineering is practice that explores how systems operate when presented with previously unexpected actions or scenarios. It may include injecting faults into systems (such as high CPU consumption, network latency, or dependency loss), changing data (fuzzing), removing capabilities, or otherwise altering the operational environment, observing how the systems respond, and then using that knowledge to initiate improvements. Chaos Engineering is also defined as "experimenting on a distributed system in order to build confidence in the system's capability to withstand turbulent	Test scenarios; Test data	Test results	Chaos engineering tool	Security

			conditions in production."				
Cyber OT&E	REQUIRED	PO.3.1, PO.3.2	Cyber OT&E evaluates the operational effectiveness, suitability, survivability, and lethality of DoD systems in operationally relevant and representative contested cyberspace				Security
Logging	REQUIRED	PO.3.3	Log system events	All user, network, application, and data activities	Logs	Logging	

Mission Based Cyber Risk Assessments	REQUIRED	PW.7.2, RV.1.1, RV.1.2, RV.2.1, RV.3.1, RV.3.2, RV.3.3,	An assessment of risks based upon the stated mission of the system or platform	NIST 800-53 RMF Control Implementations FIPS 199 system categorization Information types CNSSI 1253	Risk assessment	Risk assessment tool	Security
Performance Tests	PREFERRED	PO.3.1, PO.3.2	Tests that assess performance of the asset in the operational environment	Requirements documents and/or database; Service Level Agreements	Reports of observed performance	Operational monitoring tools	Testing
Persistent Cyber Operations Tests	AS REQUIRED	PO.3.1, PO.3.2	Tests that are incorporated into the operational environment and may be used to assess cybersecurity controls at any time, often continuously.	Test scenarios; Test data	Test results; Alerts	Cybersecurity test tools	Security
Roll Forward/Roll Back	PREFERRED	PO.3.1, PO.3.2	Continuously validate procedures to roll forward/roll back	Backups, after-images	Point-in-time recovered file	Backup management; Database automation tool	
SBOM Software Composition Analysis	REQUIRED	PS.3.2	Collect and analyze provenance data for all components of each release	Software Bill of Materials Vulnerabilities	Analysis report	SBOM Analysis tool	Security

Software Factory Risk Continuous Monitoring	REQUIRED	PS.3.2	Monitor Software Factory controls	Software Factory controls	Alerts	Monitoring tool suite	Security
Sustainanbility and Chaos Testing	PREFERRED	PO.3.1, PO.3.2	Create the capability to continuously, but randomly, cause failures in the production system to test resiliency	Chaos scripts	Report and evaluation	Test management tool	Testing
Test Audit	REQUIRED	PO 2.1, PS 2.1, PW 1.2, PW 2.1	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	Testing
User Evaluation / Feedback	REQUIRED	PO 4.2	Collect and analyze user evaluations and/or feedback	Surveys; Bug reports; Trouble tickets			Testing

3.10 Monitor Phase 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. Monitor phase activities are listed on the Monitor tab in the Activities & Tools Guidebook tables and represented here in Table 12.

NIST SP 800-137[10] defines "information security continuous monitoring (ISCM) as maintaining ongoing awareness of information security, vulnerabilities, and threats to support organizational risk management decisions."[11] 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:

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

- DoD Information Security Continuous Monitoring Strategy (January 2023)
- Policy enforcement, including ensuring hardening of CSP managed services as measured against NIST SP 800-53[12].
- Policy enforcement, including Service Categorization and Control Sections for National Security Systems (CNSSI 1253)[13]
- 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.
- 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.

Table 12: Monitor Phase Activities

Activities	Baseline	SSDF	Description	Inputs	Outputs	Tool Dependencies	Security / Testing / CM
Asset Inventory to include SBOMs	REQUIRED	PS.3.2, PW 4.1	Inventory system IT assets	IT assets	Asset inventory	Inventory Management	Configuration Management
Compliance Monitoring (COTS)	REQUIRED	PO 3.1, PO 3.2, PO 3.3	Monitor the state of compliance of deployed COTS against STIGs	Compliance status	Compliance reports	Compliance as Code	Security
Compliance Monitoring (resources & services)	REQUIRED	PO.3.1, PO.3.2, PO.3.3	Monitor the state of compliance of deployed cloud resources and services against NIST SP 800-53 controls	Compliance status	Compliance reports	Compliance Monitor	Security

[12] National Institute for Standards and Technology (NIST), “Security and Privacy Controls for Information Systems and Organizations”, December 10, 2020, https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final
[13] Committee on National Security Systems (CNSS), “Security Categorization and Control Selection for National Security Systems”, 29 July 2022, https://www.cnss.gov/CNSS/issuances/Instructions.cfm

Database monitoring and security auditing	PREFERRED	PO 3.1, PO 3.2, PO 3.3	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
Feedback, including Operational Test and Evaluation (if/as needed)	PREFERRED	PO 3.3	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	
Log Analysis & auditing	REQUIRED	PO.3.1, PO.3.2, PO.3.3	Filter or aggregate logs; Analyze and correlate logs	Logs	Alerts and remediation reports	Log aggregator	Security
Log auditing	REQUIRED	PO 3.1, PO.3.2	Ensure possession of the logs and that aggregation is performed correctly	Logs	Report	Log aggregator Log analysis & auditing	
Mission Based Cyber Risk Assessments	REQUIRED	PW.7.2, RV.1.1, RV.1.2, RV.2.1, RV.3.1, RV.3.2, RV.3.3,	An assessment of risks based upon the stated mission of the system or platform	NIST 800-53 RMF Control Implementations FIPS 199 system categorization Information types CNSSI 1253	Risk assessment	Risk assessment tool	Security

Runtime Application Self Protection (RASP)	PREFERRED	PW 8.2,	Runtime application self-protection (RASP) is a security technology that uses runtime instrumentation to detect and block computer attacks by taking advantage of information from inside the running software. The technology differs from perimeter-based protections such as firewalls, that can only detect and block attacks by using network information without contextual awareness.	Running system Test scenarios Test data	Alerts and remediation reports	RASP tool	Security

System configuration monitoring	PREFERRED	PO.3.1, PO.3.2, PO.3.3	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
							Configuration Management

System performance monitoring	PREFERRED	PO.3.1, PO.3.2, PO.3.3	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
							Configuration Management

System Security monitoring	REQUIRED	PO.3.1, PO.3.2, PO.3.3, PO.5.1, PO.5.2, RV.1.1	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	Security
Test Audit	REQUIRED	PO 2.1, PS 2.1, PW 1.2, PW 2.1	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	Testing
Test configuration audit	PREFERRED	PO 3.3	Track test and security scan results	Test results;	Test configuration audit	Track test and security scan results	Testing
User Evaluation / Feedback	REQUIRED	PO.4.2	Collect and analyze user evaluations and/or feedback	Surveys; Bug reports; Trouble tickets			Testing

3.11 Feedback Phase Activities

The Feedback phase of the DevSecOps lifecycle transmits updates to the product backlog with new features, improvements, bugs, vulnerabilities, and metrics captured from the Operations team, especially in the Monitor phase. Activities for the Feedback phase are listed in Table 13.

Table 13: Feedback Phase Activities

Activities	Baseline	SSDF	Description	Inputs	Outputs	Tool Dependencies	Security / Testing / CM

Revise Product Backlog	REQUIRED	PO.1.1, PO.1.2; PO.1.3, PO.3.1, PO.4.1, RV.2.2, RV.3.1, RV.3.2, RV.3.3, RV.3.4	Update the product backlog with new features, improvements, bugs, vulnerabiliy remediations, and performance improvements based upon collected metrics	Requirements; Improvements; Bugs; Vulnerabilities; Collected metrics	Updates to the Product Backlog	Requirements Management Tool	
User Evaluation / Feedback	REQUIRED	PO.4.2	Collect and analyze user evaluations and/or feedback	Surveys; Bug reports; Trouble tickets			Testing

4 DevSecOps Tools

The Tools tab of the Activities and Tools Guidebook spreadsheet identifies 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 are listed in Table 14.

The Tools tab includes the following 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

Table 14: DevSecOps Tools

Tool	Features	Benefits	Inputs	Outputs
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

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
Asset inventory management	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
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

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

Chaos engineering tool	Orchestrate scripted changes to the environments, applications, or data related to a test scenario or experiment; collect and analyze results	Build confidence in the system's capability to withstand turbulent conditions in production.	Scripted test scenarios; Test data	Test results; Analysis

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
Compliance as Code	Monitor the state of compliance of deployed COTS against STIGs	Automated detection and, ideally, remediation of non-compliance within the infrastructure	Compliance policies, Operational environment	Compliance findings, Remediation recommendations or logs
Compliance Monitor	Monitor the state of compliance of deployed cloud resources and services against NIST SP 800-53 controls	Provide visibility of compliance status	Compliance status	Compliance reports
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

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

Configuration Management Tool	Configuration control	Maintains integrity of the system	Source code, Everything as Code	Configuration control
Cyber Threat Intelligence Subscription(s)	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

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
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
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
Database encryption tool	Encrypt data at rest and in transit	Provide data privacy and security; Prevent data loss	Original data	Encrypted data
Database monitoring tool	Baseline database performance and database traffic; Detect anomalies	Improve database operations continuity	Running database	Logs; Warnings and alerts
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
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;	Reduce the security risks	Test data; Test scenarios	Vulnerability findings; Recommended mitigation actions

	Data access control test; User access control test; Denial of service test			
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
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
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.

InfoSec Continuous Monitoring (ISCM) Tool	Monitor network security Monitor personnel activity Monitor configuration changes Perform periodical security scan to all system components Monitor the IT assets and detect deviations 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.	Detect unauthorized personnel, connections, devices, and software Identify cybersecurity vulnerability Detect security and compliance violation Verify the effectiveness of protective measures	IT asset Network Personnel activities Known vulnerabilities	Vulnerbilities Incompliance Findings Assessments and recommendations
Integrated development environment (IDE)	Source code editor Intelligent code completion Compiler or interpreter Debugger Build automation	Visual representation Increase efficiency Faster coding with less effort Improved bug fixing speed Reproducible builds via scripts	Developer coding input	Source code

	(integration with a build tool)			
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
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.

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

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

Log aggregator	Filter log files for events of interest (e.g., security), and transform into canonical format	Improve investigations by correlating logs from multiple sources	Event Logs, Database Logs, Audit Logs, Database Security Audit Logs	Aggregated, filtered, formatted event log
Log analysis	Analyze and audit to detect malicious threats / activity; Automated alerting and workflows for response Forensics for damage assessment. These are typically SIEM and SOAR tools.	Reduce effort required to identify threats or inappropriate activity	Logs	Alert messages, emails, etc. Remediation report and log
Log auditing	Audit to ensure possession of the logs and that aggregation is performed correctly		Logs	Audit Logs
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	Improve investigations by filtering logs	Event logs, database logs, audit logs, security audit logs	Aggregated, filtered, formatted event log record
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

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

Non-security compliance scan	Such as Section 508 accessibility compliance	Ensures compliance with non-security programs	Artifacts	Compliance report

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
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
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
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.)

Requirements database	Collect and manage requirements; Trace requirements to their source; Trace features or design components to the requirements	Tracing requirements make sure all requirements are handled and minimize unwanted extra features that don't trace to requirements.	Project goal and constraints	Requirements documents
Software Bill of Materials (SBOM) Software Composition Analysis Tool	Software composition analysis (SCA) analyzes custom-built software applications to detect embedded software and detect if it is up-to-date, contains vulnerabilities or security flaws, or has licensing requirements	SCA allows developers and operators to understand what is contained within the software that is entering or operating within their environment. The automatic nature of SCA products is another strength, developers don't have to manually verify the contents when integrating Open-Source Software (OSS) components. The automation also applies to indirect references to other OSS components within code and artifacts	SBOM	Analysis report and recommended mitigation.
Software license compliance checker	Inventory software license; Audit the compliance	Software license compliance and software asset management	Purchased license info; Software instances	Compliance report
Software system design tool	Assist system 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
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

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
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.

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

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.
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

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
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
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 scenarios, Test scripts, Test data	Test results, test report
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

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