

NIST SP 800-218 SSDF & NCCoE Secure Software Development, Security, and Operations (DevSecOps) Practices

Michael Ogata
michael.Ogata@nist.gov
May 20, 2026

SSDF Publication Basics

Audience

For both *software producers* (e.g., COTS vendors, government software developers, custom software developers) and *software consumers* (federal government agencies and other organizations)

Flexible

Can be used by *organizations* in any *sector* or community, regardless of *size* or cybersecurity *sophistication* to manage risks associated with software

Can be *integrated* into any existing software development workflow and automated toolchain

Customizable

Broadly applicable—not specific to *technologies* e.g., IT, ICS, IoT, or cyber-physical systems (CPS), *platforms*, *programming languages*, *SDLC models*, *development environments*, *operating environments*, *tools*, etc.

Approach Similar to Cybersecurity Framework



Common language for secure
software development
practices



Document and define future
target practices

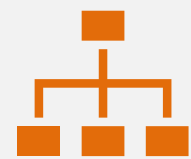


Leverages existing secure
software development
practices



Do no harm

SSDF Practice Groups



Prepare the Organization (PO): Ensure the organization's [people, processes, and technology](#) are prepared to perform secure software development at the organization level and, in some cases, also for each individual project.



Protect the Software (PS): [Protect](#) all components of the software from [tampering and unauthorized access](#).



Produce Well-Secured Software (PW): Produce well-secured software that has [minimal security vulnerabilities](#) in its releases.



Respond to Vulnerabilities (RV): Identify vulnerabilities in software releases and respond appropriately to [address those vulnerabilities](#) and prevent similar vulnerabilities from occurring in the future.

Elements of an SSDF Practice

Practices	Tasks	Implementation Examples	References
Respond to Vulnerabilities (RV)			
Identify and Confirm Vulnerabilities on an Ongoing Basis (RV.1): Help ensure that vulnerabilities are identified more quickly so that they can be remediated more quickly, reducing the window of opportunity for attackers.	RV.1.1: Gather information from purchasers, consumers, and public sources on potential vulnerabilities in the software and third-party components that the software uses, and investigate all credible reports.	- Establish a vulnerability disclosure program, and make it easy for security researchers to learn about your program and report possible vulnerabilities. - Monitor vulnerability databases, security mailing lists, and other sources of vulnerability reports through manual or automated means. - Use threat intelligence sources to better understand how vulnerabilities in general are being exploited. - Regularly check the provenance and software composition data for each software release in use to identify potential new vulnerabilities in its components.	BSAFSS: VM.1-3, VM.3 BSIMM: CMVM1.2, CMVM3.4 CNCFSSCP: Securing Materials—Verification IEC62443: DM-1, DM-2, DM-3 ISO29147: 6.2.1, 6.2.2, 6.2.4, 6.3, 6.5 ISO30111: 7.1.3 OWASPSAMM: IM1-A, IM2-B, EH1-B OWASPSCVS: 4 PCISSLC: 3.4, 4.1, 9.1 SCAGILE: Operational Security Task 5 SCFPSSD: Vulnerability Response and Disclosure SCTPC: MONITOR1 SP800181: K0009, K0038, K0040, K0070, K0161, K0362; S0078

Task: An individual action (or actions) needed to accomplish a practice

Implementation Example: An example of a type of tool, process, or other method that could be used to implement this task

Reference: An established secure development practice document and its mappings to this task

(PO) Prepare the Organization



(PO.1) Define security requirements for software development



(PO.2) Implement roles and responsibilities



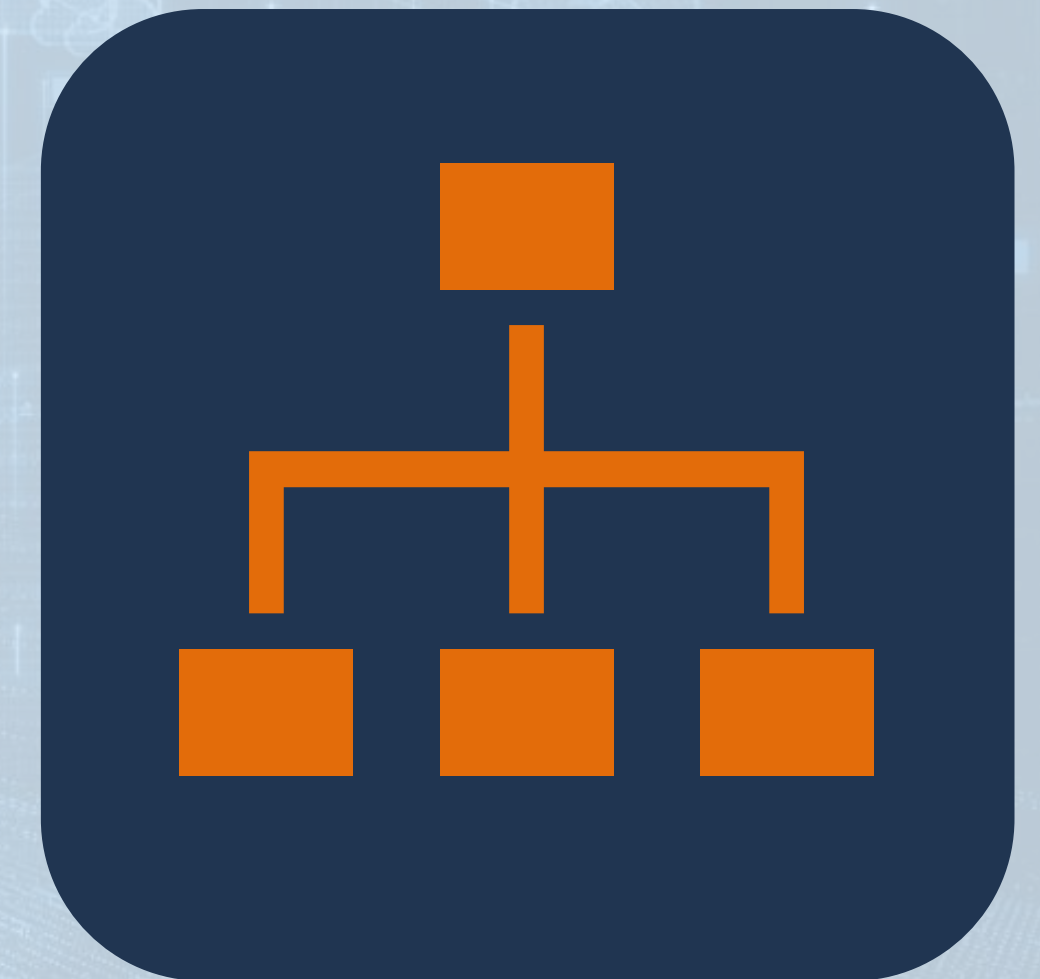
(PO.3) Implement supporting toolchains



(PO.4) Define and use criteria for software security checks



(PO.5) Implement and maintain secure environments for software development



(PS) Protect the Software



(PS.1) Protect all forms of code from unauthorized access and tampering



(PS.2) Provide a mechanism or verifying software release integrity



(PS.3) Archive and Protect each software release



(PW) Produce Well-Secured Software



(PW.1) Design software to meet security requirements and mitigate risks



(PW.2) Review the software design to verify compliance with security requirements



(PW.3) --



(PW.4) Reuse existing, well-secured software when feasible instead of duplicating functionality



(PW.5) Create source code by adhering to secure coding practices



(PW.6) Configure the compilation, interpreter, and build processes to improve executable security



(PW.7) Review and/or analyze human-readable code to identify vulnerabilities and verify compliance with security requirements



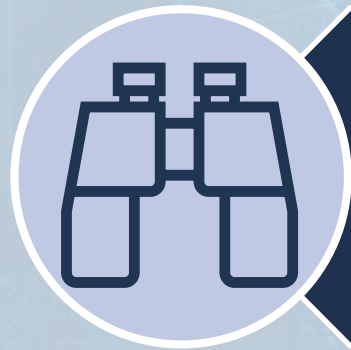
(PW.8) Test executable code to identify vulnerabilities and verify compliance with security requirements



(PW.9) Configure software to have secure settings by default



(RV) Respond to Vulnerabilities



(RV.1) Identify and confirm vulnerabilities on an ongoing basis



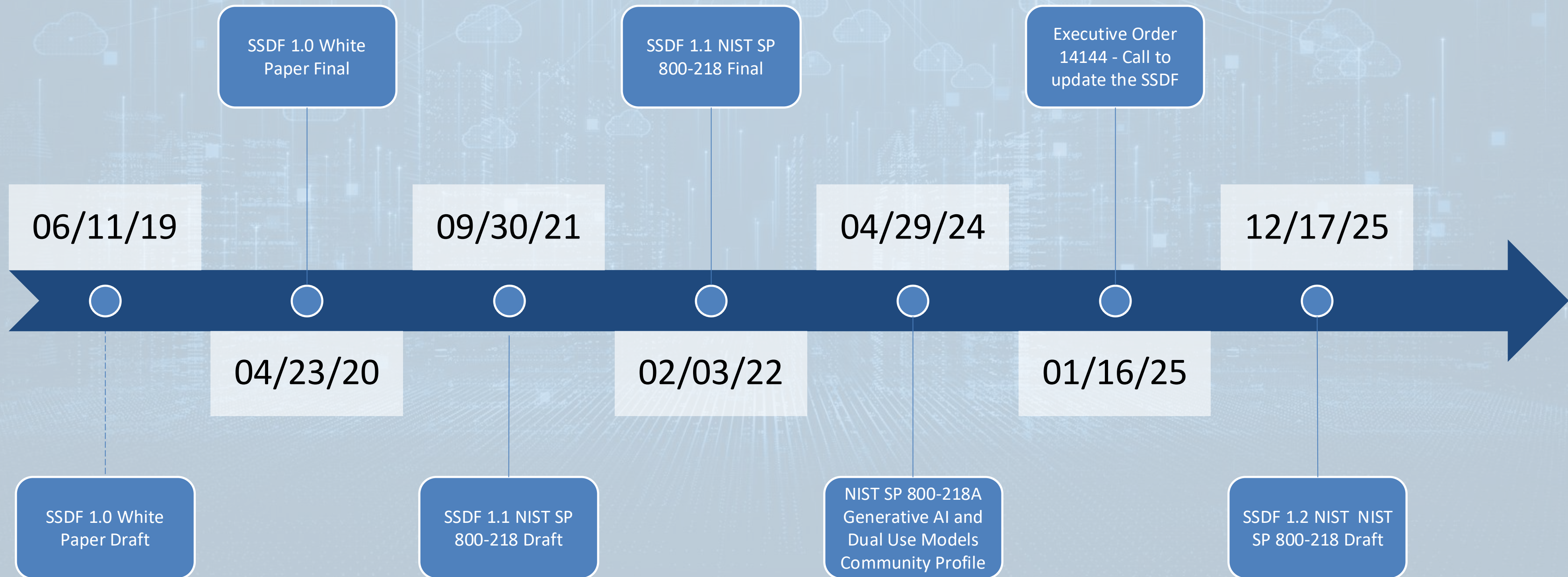
(RV.2) Assess, Prioritize, and Remediate vulnerabilities



(RV.3) Analyze vulnerabilities to identify their root causes



SSDF Publication Timeline



SSDF v1.2 Draft Overview

- Executive Order 14144:
...secure and reliable development and delivery of software as well as the security of the software itself...
- Primarily a response to recent large-scale cybersecurity incidents
- Added two new practice groups:
 - **PO.6** Continuous Improvement
 - **PW 10** Robust and Reliable Updates
- Expanded implementation examples
- Updated reference mappings to SP 800-53, SP 800-161, and SP 800-181

SSDF v1.2 New Practices and Tasks

PO.6 (Continuous Process Improvement): Requires organizations to identify and execute improvements to cybersecurity procedures across all SSDF practices.

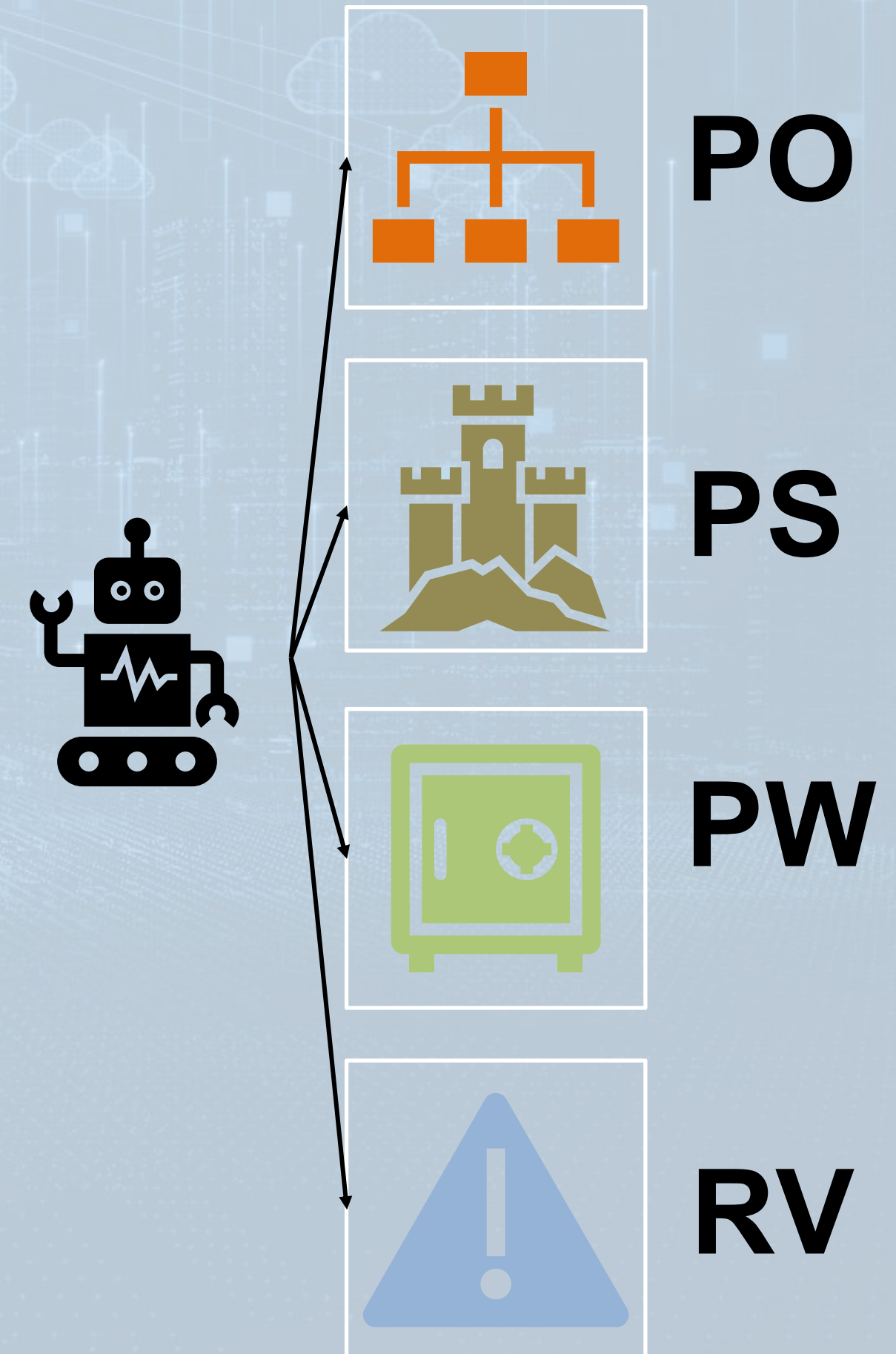
- **PO.6.1:** Update and improve software development environments in response to new threats and as new tools are included in the development process.
- **PO.6.2:** Identify and implement new processes, tools, and techniques that can help avoid software errors
- **PO.6.3:** Improve vulnerability response processes and periodically review prior decisions

PW.10 (Robust and Reliable Updates): Focuses on strategies for testing and delivering software updates responsibly to minimize customer disruption.

- **PW.10.1:** Thoroughly test all software updates following all guidance in **PW**.
- **PW.10.2:** Use tiered update and release strategies, such as canaries, staged rollouts, and test deployments.
- **PW.10.3:** Include robust rollback mechanisms for updates.
- **PW.10.4:** Maintain robust and reliable update engines and associated infrastructure for the delivery of updates.

AI Update – Using AI in the software development process

- Primary Objective: Incorporate AI considerations into the Secure Software Development Framework (SSDF) while maintaining manageable scope.
- Targeted Focus Areas:
 - AI as Code Producers:
 - Concentrates on AI-generated code executed as the final product.
 - Excludes non-code artifacts (e.g., prose, art, audio) to address software-specific cybersecurity risks.
 - AI as Process Participants:
 - Focuses on AI that automates or enhances design, delivery, and maintenance.
 - Includes autonomous agents for build orchestration, testing, and deployment.
 - Covers AI-driven tools for code analysis and testing.





Thank you!

Michael Ogata
michael.Ogata@nist.gov
May 20, 2026

Secure Software Development, Security, and Operations (DevSecOps) Practices

OBJECTIVE:

Develop and document an applied risk-based approach and recommendations for secure software development, security, and Operations (DevSecOps) practices consistent with:

- Secure Software Development Framework (SSDF)
- Other NIST, government, and industry guidance

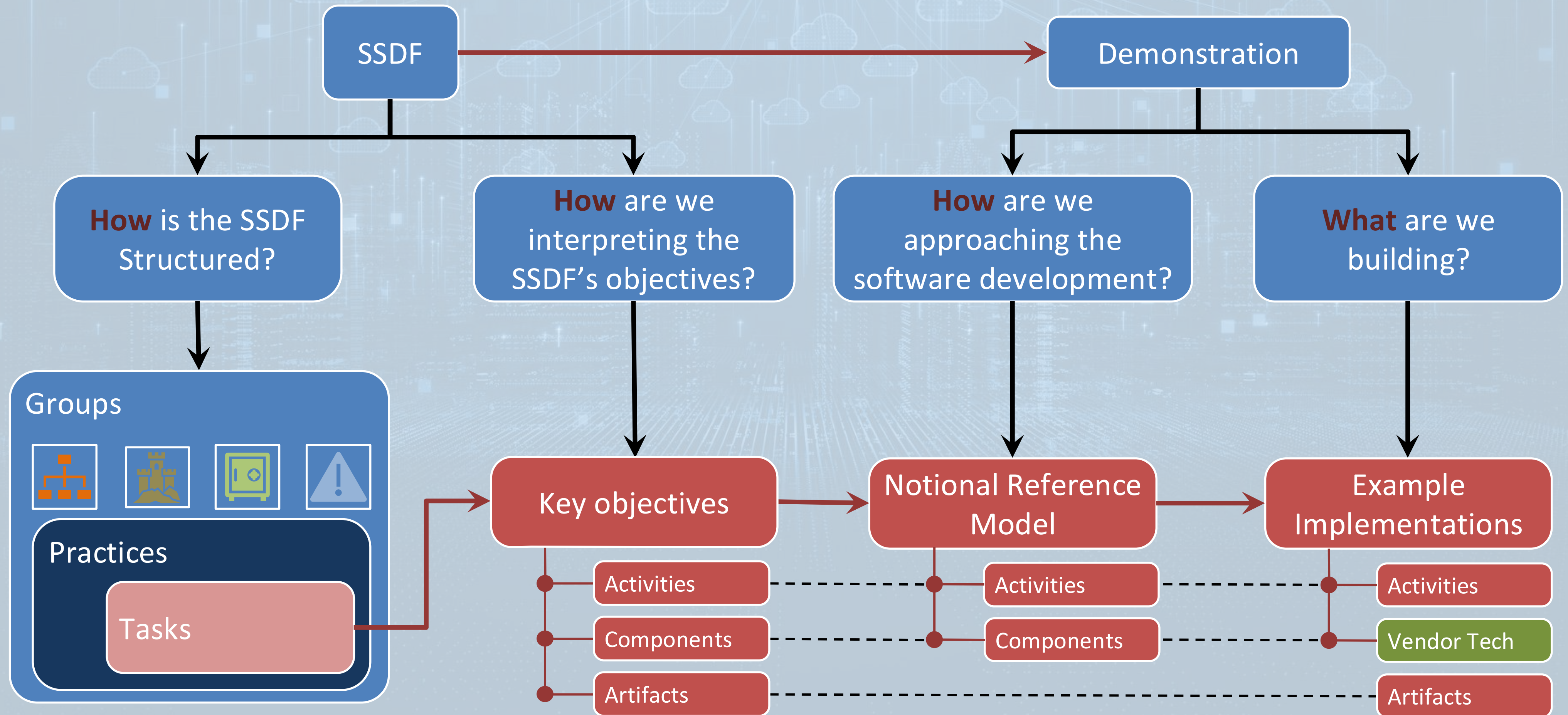
SCENARIOS:

Demonstrate DevSecOps practices in multiple proof-of-concept use case scenarios that involve software development environments integrated with different industry technologies, including security practices associated with the use of generative AI capabilities and Zero Trust Architectures.



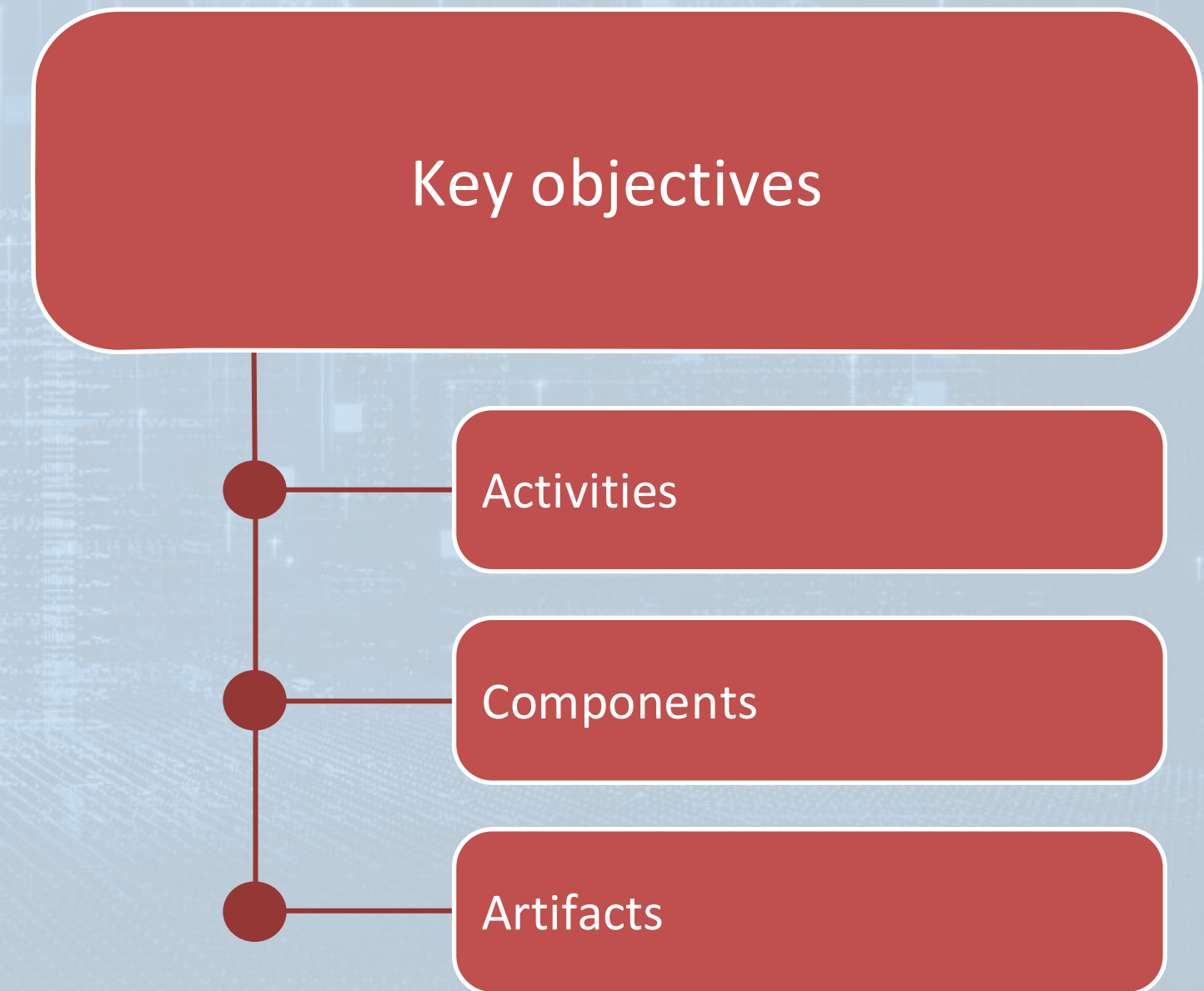
***This project was initiated in March 2025 to support the directive outlined in **Section 2(c)(i) of Executive Order 14306**

Demonstration Strategy

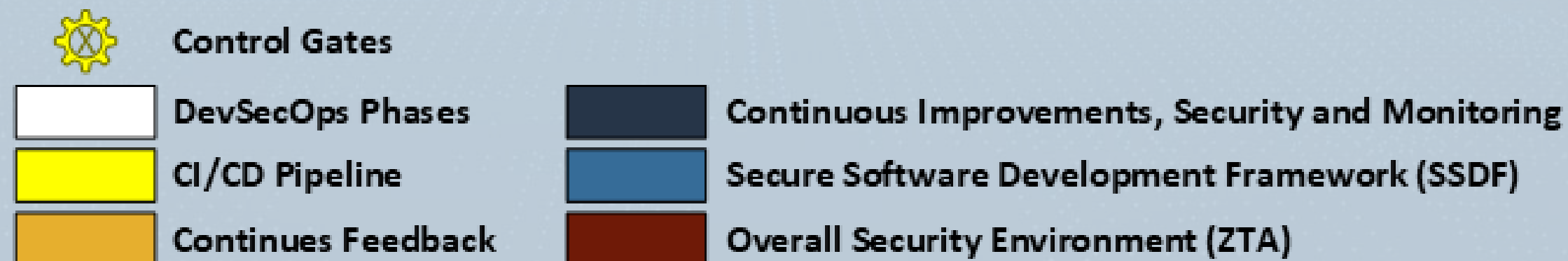
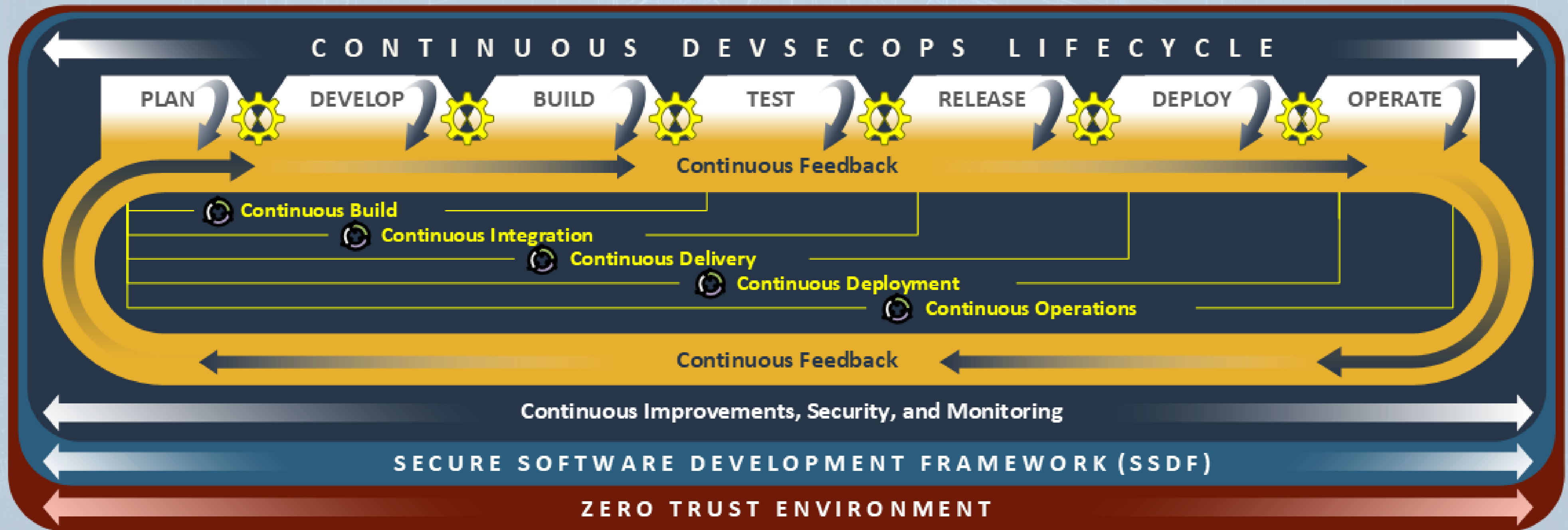


Demonstration Key Objectives

- SSDF is process/domain agnostic
- Key Objectives narrow and focus scope Tasks for our demonstration
- Not exhaustive
- Each task has one or more Key Objective
 - Activities that enable that objective
 - Components that enable the completion of activities
 - Artifacts/evidence should be generated by the components



NCCoE DevSecOps Project Reference Model



Model Activities

Plan

- Team Collaboration
- Requirements Management
- Product Design
- Bug, Defect, and Issue Tracking
- Risk Management
- Threat Modeling
- Configuration Management
- Credential and Secrets Management
- CI/CD Pipeline
- Zero Trust Security

Develop

- Software Development
- Code Analysis
- Develop Infrastructure as Code
- Source Code Management
- Securing Sensitive Information
- Firmware Development
- Artifact Signing and Verification
- Cryptographic Key Management
- Develop CI/CD Pipeline
- Zero Trust Security

Build

- Automate the Build Process
- Implement isolated/hermetic environments
- Analyze Infrastructure as Code (IaC)
- Integration of software Libraries
- Perform Unit Testing
- Automate security checks, code analysis, and build processes
- Securing Sensitive Information
- Detect exposed credentials
- Securing build artifacts
- Firmware Artifact Signing and Verification
- Artifact Scanning
- Assessing Provenance of Build Artifacts
- Generate and Digitally Sign Software Bill of Material (SBOM)
- Analyze Container Images
- Zero Trust Security

Test

- Test
- Automate Test Execution
- Analyze Infrastructure as Code (IaC)
- Analyze Source Code for Vulnerabilities
- Perform Unit Testing
- Perform Regression Testing
- Perform Integration Testing
- Perform Acceptance Testing
- Perform Smoke Testing
- Dynamic Application Security Testing (DAST)
- Interactive Application Security Testing (IAST)
- Perform Fuzz Testing
- Perform API Testing
- CI/CD Execution, Testing and Security Policy Compliance
- Firmware Artifact Integrity Verification
- Artifact Scanning
- Assessing Data Provenance
- Verify Digitally Signed Software Bill of Material (SBOM)
- Zero Trust Security

Release

- Automate of Release and Delivery Processes
- Securing Release Artifacts
- Manage Configurations
- Manage Software Components
- Coordinate Software Releases
- Document Release Process
- Perform Smoke Testing
- Verify Release Criteria
- Analyze Running Application for Vulnerabilities
- Firmware Release Readiness
- Artifact Signing and Verification
- Analyze Container Security
- Validate Software and Data Origins
- Analyze Infrastructure as Code (IaC)
- Zero Trust Security

Deploy

- Automate the Deployment Processes
- Securing Deployment Artifacts
- Manage System and Application Configurations
- Manage Software Releases
- Analyze IaC
- Automate Software Deployment
- Manage Software Supply Chain
- Secure Firmware Deployment and Configuration
- Zero Trust Security

Operate

- Manage Application Environments
- Manage SBOMs for Deployed Applications
- Validate Software Artifact Integrity
- Firmware Services
- Zero Trust Security

Notional Reference Model

Activities

Components

Model Components

Notional Reference Model



	Plan	Develop	Build	Test	Deploy	Release	Operate	Continuous Improvements, Security and Monitoring
API Test Tool				✓				
Acceptance Test Tool				✓		✓		
Artifact Repository		✓	✓	✓	✓	✓		
Artifact Signing and Verification Tool		✓	✓	✓	✓	✓		
Attestation Signing and Verification Tool			✓	✓	✓	✓		
Build Tools			✓					
CI/CD Execution, Test and Security Policy Verification Tool				✓				
CI/CD Pipeline		✓	✓	✓	✓	✓	✓	✓
Certificate Management System	✓							✓
Configuration Management System	✓							✓
Container Image Scanner			✓	✓		✓		
Credential Management System	✓							✓
Cyber Intelligence, Threat, and Security Metadata Feeds	✓							
DAST System				✓		✓		
Deployment Management System					✓			
Developer Tools		✓						
Firmware Services	✓							✓
Fuzz Test Tool				✓				
Hardware Security Module	✓							✓
IAST System				✓		✓		
IaC Scanner		✓	✓	✓	✓	✓		
IaC Scripts		✓	✓	✓	✓	✓	✓	
Integration Test Tool				✓				
Lint Tool		✓	✓					
Operations Monitoring System								✓
Package Management System						✓		
Product Management System	✓							
Project Management System	✓							
Provenance Generation and Verification Tool			✓	✓	✓	✓		
Regression Test Tool				✓				
Release Management System						✓		
Requirements Management System	✓							
Risk Management System	✓							✓
Runtime Signature Verification Tool							✓	
SAST System		✓	✓	✓				
SCA System		✓	✓	✓		✓		
SCM System		✓						✓
Secret Scanner		✓	✓					
Secrets Management System	✓							✓
Security Monitoring System								✓
Smoke Test Tool				✓		✓		
Software Libraries		✓	✓					
Team Collaboration Tools								✓
Threat Modeling System	✓							✓
Ticketing System	✓							✓
Unit Test Framework		✓	✓	✓				

Output and Status

Secure Software Development, Security, and Operations (DevSecOps) Practices

<https://pages.nist.gov/nccoe-devsecops/index.html>

- Live document
- Executive summary
- Normative Reference Model
- Example implementation documentation



NIST / Secure Software Development, Security, and Operations (DevSecOps) Practices [View page source](#)

Secure Software Development, Security, and Operations (DevSecOps) Practices

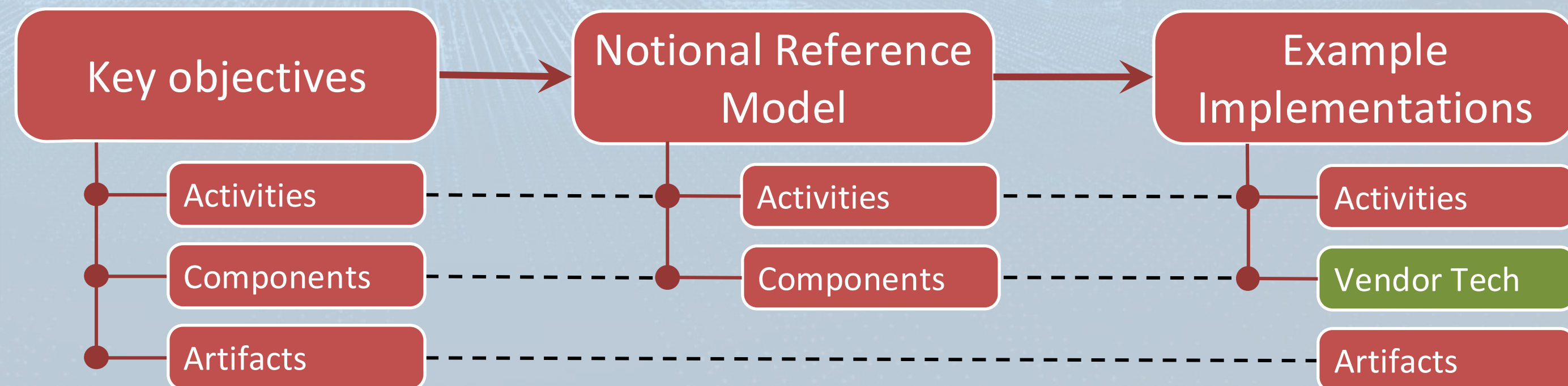
March 2026

This publication is available free of charge from <https://www.nccoe.nist.gov/projects/secure-software-development-security-and-operations-devsecops-practices>

Authors:

Alper Kerman Michael Ogata National Institute of Standards and Technology	Daniel Carroll Daniel Jackson Dell Technologies	Tal De La Rosa Segu Riluvan Mark Svancarek Microsoft
Parisa Grayeli Joshua Klosterman Phillip Millwee Deanna Stanley Allen Tan The MITRE Corporation	Dean Coclin Dave Roche DigiCert	Keng Lim Sameer Shukla NextLabs
William Barker Dakota Consulting	Matt Brown Tom Gleason Endor Labs	Chima Onukwuru Neil Roxburgh Palo Alto Networks
Sudan Ayanam Stefano Righi AMI	Paul Pickhardt MaryGrace Wajda GitLab	Jose Palazon Michael Smith Sagittal AI
Chrissa Constantine Tim Mackey Black Duck	Isaac Hepworth Brandon Lum Google	Guy Chernobrov Daniel Nebenzahl Scribe Security
Rahul Dubey James Imanian CyberArk	Philippe Mulet Harmeet Singh IBM	

National Institute of Standards and Technology



Example Implementations

- **Example Implementation 1 – Completed**

- Microsoft/GitHub (SaaS and IaaS)
- Black Duck
- CyberArk (for AKV only)
- DigiCert
- Endor Labs
- Sagittal AI

- **Example Implementation 2 – In progress**

- CyberArk
- DigiCert
- GitLab (SaaS)
- Microsoft – deployment environment, SIEM/SOAR
- NextLabs
- Sagittal AI
- Scribe Security

- **Example Implementation 3:**

- Google (SaaS and IaaS)
- Other collaborators who have out of box integration with Google

- **Example Implementation 4:**

- IBM
- Other collaborators who have out of box integration with IBM

- **Example Implementation 5:**

- AMI
- Other collaborators who have out of box integration with AMI

- **Example Implementation 6:**

- Palo Alto Network (PAN)
- PAN's self hosted GitLab or Microsoft Azure environment
- Other collaborators who have out of box integration with PAN

AI Technologies Included in Implementations

- Requirements generation and organization
- Task decomposition and organization
- Ticketing and issue generation
- Code generation and modification
- Cloud resource and service configuration
- Pull and merge request support
- Test generation
- Code scanning
- Security analysis
- Vulnerability detection and remediation.

Project Collaborators

- **AMI**
- **Black Duck**
- **CyberArk**
- **Dell Technologies**
- **DigiCert**
- **Endor Labs**
- **GitLab**

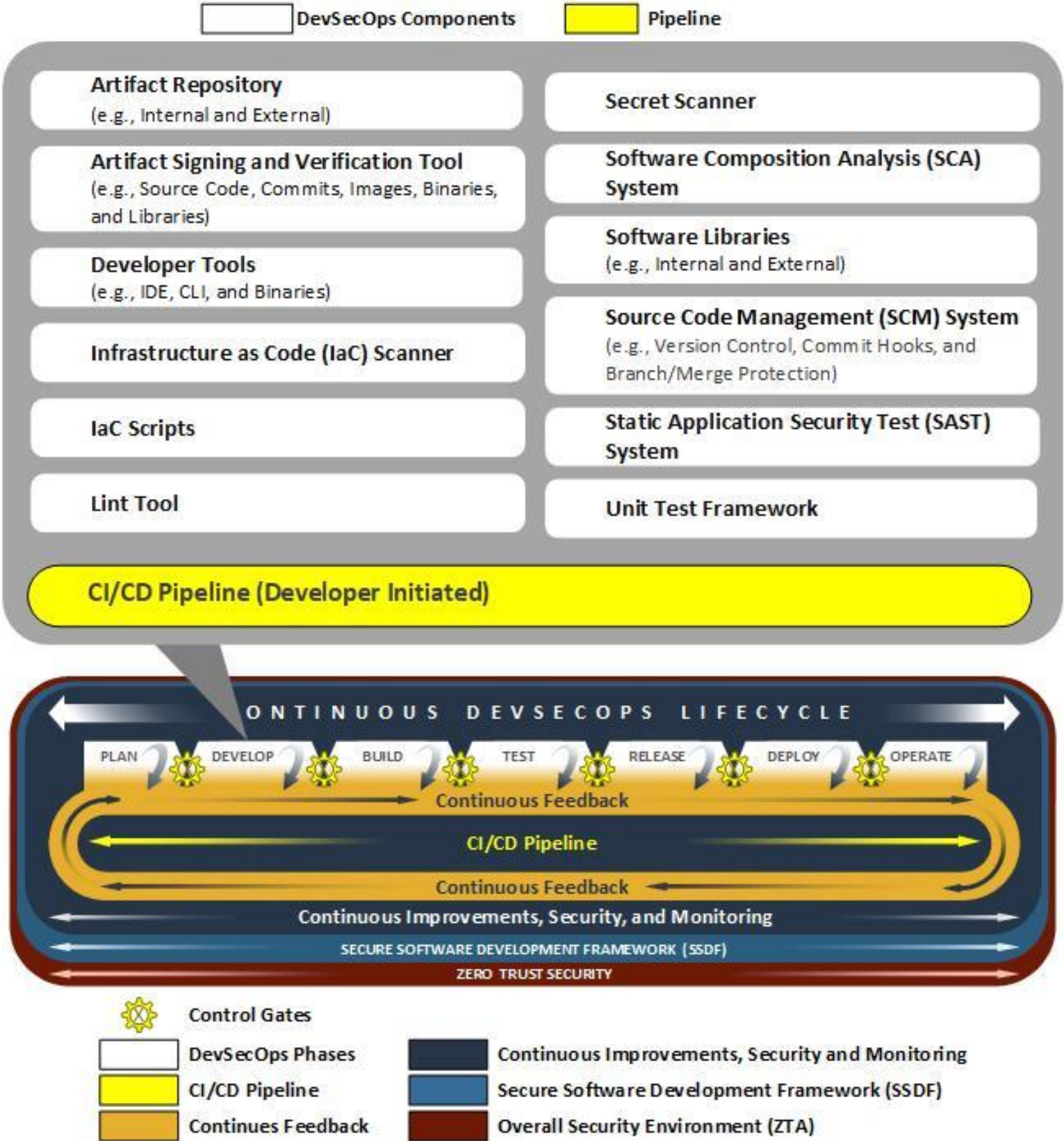
- **Google**
- **IBM**
- **Microsoft**
- **NextLabs**
- **Palo Alto Networks**
- **Sagittal AI**
- **Scribe Security**

Project's DevSecOps Reference Model – **PLAN PHASE**

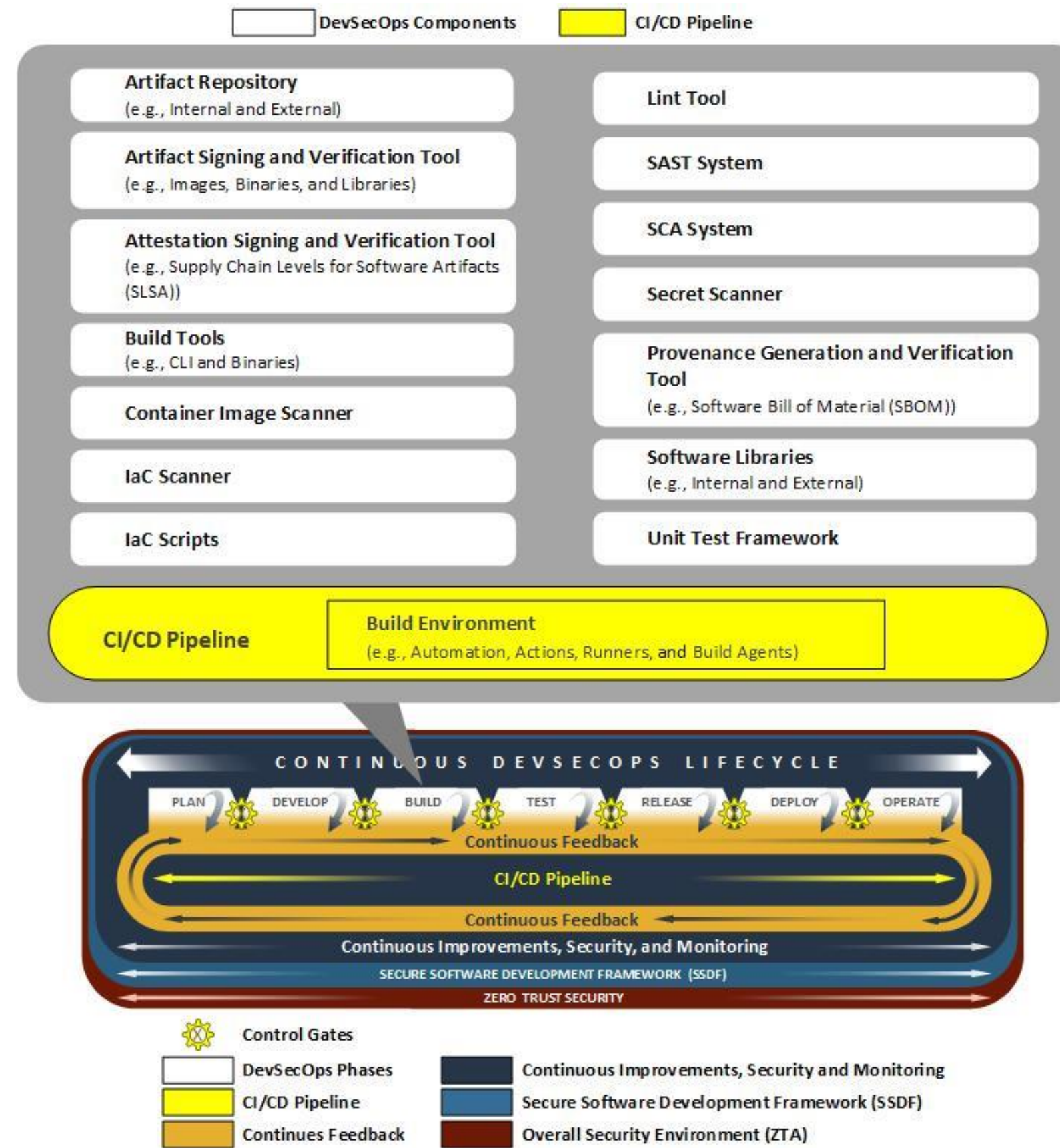


Fig. 3.2 Plan Phase

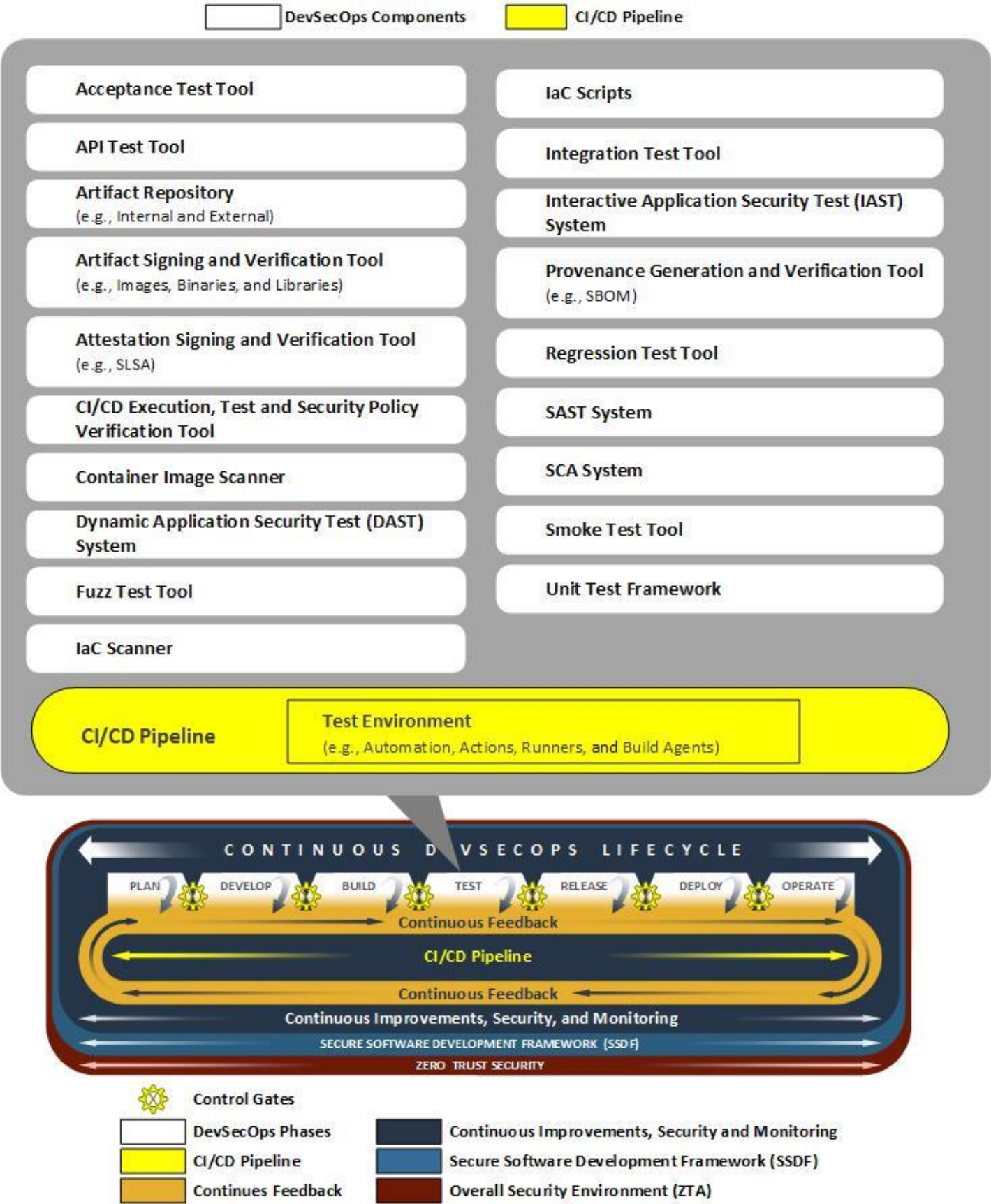
Project's DevSecOps Reference Model – DEVELOP PHASE



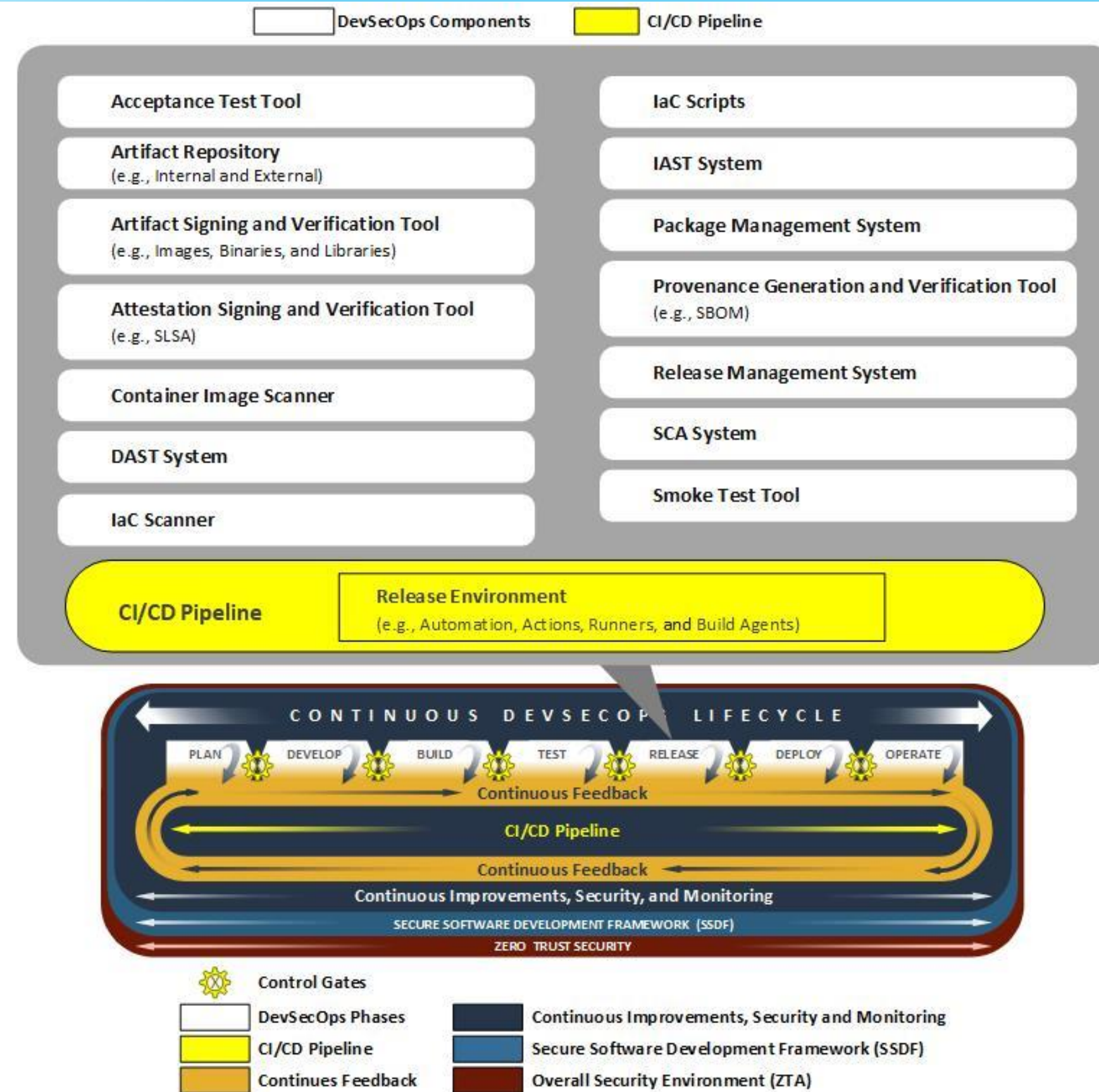
Project's DevSecOps Reference Model – BUILD PHASE



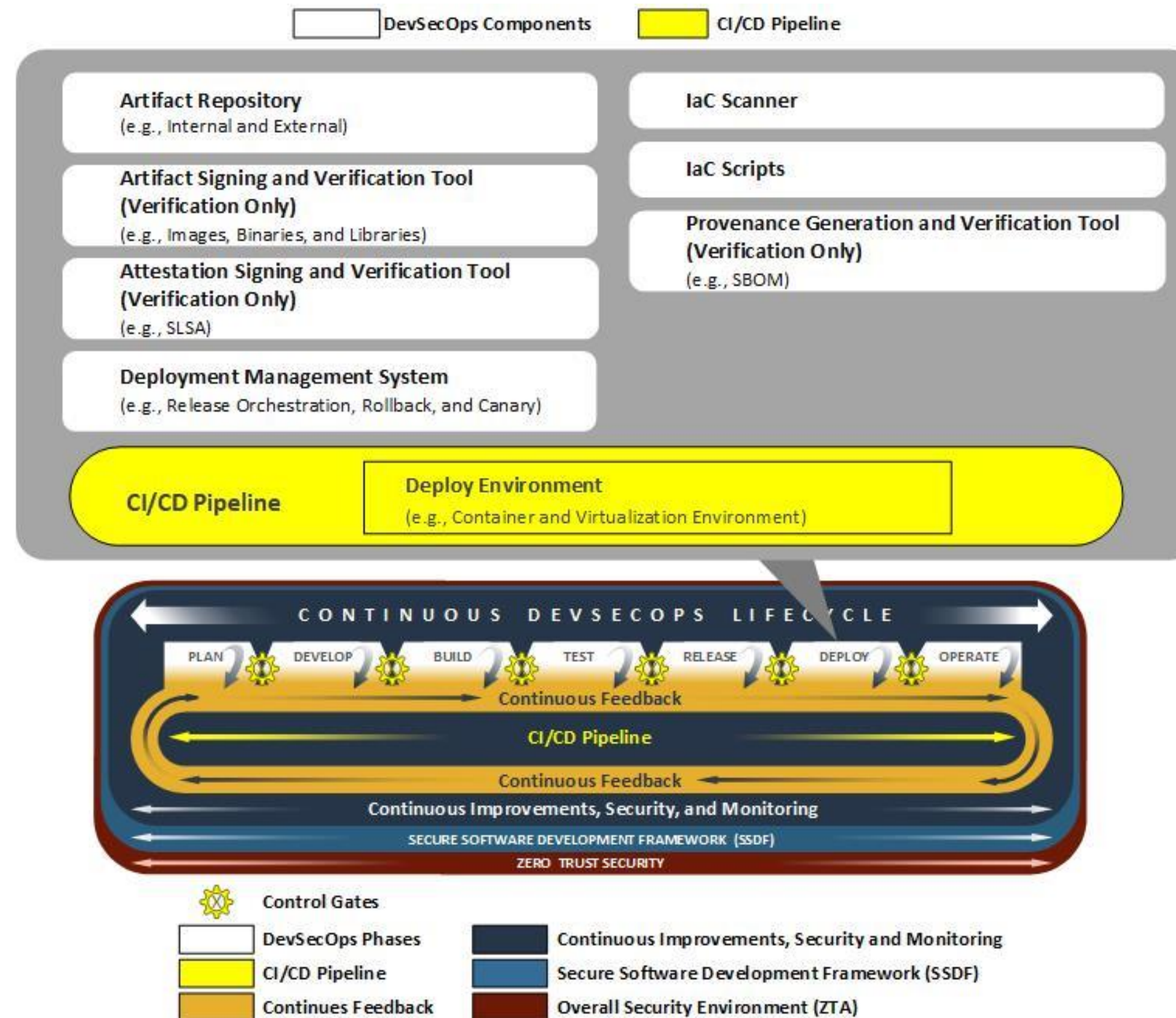
Project's DevSecOps Reference Model – TEST PHASE



Project's DevSecOps Reference Model – **RELEASE PHASE**



Project's DevSecOps Reference Model – **DEPLOY PHASE**



Project's DevSecOps Reference Model – **DEPLOY PHASE**

