# DevSecOps Fundamentals Playbook

March 2021

Version 2.0

## Trademark Information

## Contents

## Play 1: Adopt a DevSecOps Culture

DevSecOps is a software engineering culture that guides a team to break down silos and unify software development, deployment, security and operations. Critical to the success of DevSecOps adoption is buy-in from all stakeholders, including: leadership, acquisition, contracting, middle-management, engineering, security, operations, development, and testing teams. Stakeholders across the organization must change their way of thinking from “I” to “we”, while breaking team silos, and understanding that the failure to successfully deliver, maintain, and continuously engineer software and its underlying infrastructure is the failure of the entire organization, not one specific team or individual.

Before beginning a DevSecOps journey, it is imperative to understand that a successful implementation of DevSecOps cannot be measured by a completely automated pipeline or the interaction between development and operations teams alone; all stakeholders in the organization must be committed to changing the way they view their job responsibilities and, most importantly, interact with each other.

## Key Cultural Practices

- Stakeholder transparency and visibility.
- Complete transparency across team members in real-time.
- All project resources easily accessible to the entire team; not everyone needs commit privileges.
- Adopt and embrace ChatOps as the communication backbone for the DevSecOps team.
- All technical staff should be concerned with, and have a say in, baked-in security.

## Checklist

- ☐ Learn what is involved in the DevSecOps culture.
- ☐ Embrace automation for anything done repeatedly.
- ☐ Read *How to Build a Strong DevSecOps Culture* by K. Casey, available online at: https://enterprisersproject.com/article/2018/6/how-build-strong-devsecops-culture-5-tips
- ☐ Read *The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win* by G. Kim, K. Behr, and G. Spafford, IT Revolution Press, Jan. 10, 2013
- ☐ Fail fast, learn fast, fail small, and do not fail twice for the same reason!

## Play 2: Adopt Infrastructure as Code

Infrastructure as Code (IaC) is infrastructure definition and configuration that is defined with text files that are checked-in to a source code repository and kept under configuration management. It includes the management of networks, storage, virtual machines, load balancers, and even connection topologies. IaC evolved to solve a real-world problem referred to as environment drift in the release pipeline. Succinctly, the development environment fails to align with the production environment configuration. The goal is to automate all infrastructure provisioning and configuration in a repeatable, consistent way that also lends itself to peer reviews of the changes prior to any configuration changes actually being made.

IaC can take many forms. One is a template for instantiating a cloud service in a secure way. Another is through configuration files or scripts. It is important to consider vendor lock-in versus product lock-in when selecting technology or IaC formats. Blueprints and Cloud Formation only apply to Microsoft Azure and Amazon Web Services (AWS) respectively, creating a degree of vendor lock-in; Cloud-agnostic solutions, such as those provided by popular tools like Ansible and Terraform, avoid vendor lock-in but create product lock-in. In all cases, the IaC is specified via one or more text files.

GitOps is a paradigm where systems are described and observed declaratively, using code to specify the desired state. The benefits of GitOps build upon IaC, emphasizing the role of git and a git driven workflow. IaC is one of the three core practices of GitOps, along with merge requests and the reliance upon a CI/CD pipeline.

### Key Advantages

- IT infrastructure supports and enables change, rather than being an obstacle or a constraint.
- Mitigates drift between environments by leveraging automation and push-button deployment.
- Enforces change management through GitOps with multiple approvers, as needed.
- Environmental changes are routine and fully automated, pivoting staff to focus on other tasks.
- Quicker recovery from failures, rather than assuming failure can be completely prevented.
- Empowers a continuous improvement ecosystem rather than “big bang” one and done activities.

### Checklist

- ☐ Learn how to describe the value proposition of IaC.
- ☐ Understand the benefits of applying GitOps to infrastructure configurations.
- ☐ Understand how IaC tooling selection is a trade-off between vendor lock-in or product lock-in.
- ☐ Explore popular IaC tooling options, including:
  - Terraform
  - Ansible
  - Chef
  - CSP managed service tooling

## Play 3: Adopt Containerized Microservices

A modular open system approach (MOSA) is an acquisition and design strategy consisting of a technical architecture that adopts open standards and supports a modular, loosely coupled and highly cohesive system structure.[1] U.S. Code Title 10 Section 2446a, and DoD Instruction 5000.02 require MOSA. A modern software architecture predicated upon microservices and software containers meet MOSA requirements.

A container is a lightweight, standalone, executable package of software that includes everything needed to run a business service except the OS; code, runtime, system tools, system libraries and settings. Containers run in isolated processes from one another, so several containers can run in the same host OS without conflicting with one another. All containers must be Open Container Initiative compliant.[2] The DoD DevSecOps Strategy requires a CNCF Certified Kubernetes cluster for container orchestration; there over 90 Certified Kubernetes implementations and counting.[3]

A microservice architecture is an approach to application development where discrete, modular business services are bundled inside of a software container. These business services are then loosely coupled and rapidly composed using lightweight protocols. The primary functional benefit of this approach when executed properly is that each service can advance independently from the other services. Numerous non-functional benefits also exist, including more agility in scaling to demand, multiple upgrade options that don't impact the user population, more precise cyber hardening at a per-service level, and inherent support for failure and recovery.

## Key Characteristics of a Containerized Microservice

- Componentization via services.
- Organized around business capabilities.
- Product over project.
- Smart endpoints, dumb pipes.
- Decentralized governance and data management.
- Infrastructure automation support via IaC.
- Design for failure.
- Evolutionary design support.

## Checklist

- ☐ Research and understand the benefits of a microservices architecture.
- ☐ Only adopt CNCF Certified Kubernetes to ensure software conformance of required APIs.
- ☐ Leverage Iron Bank for hardened containers and other software artifacts.
- ☐ Always inject the Sidecar Container Security Stack (SCSS) to maximize runtime security.
- ☐ Always adopt a service mesh to further secure east-west network traffic.

[1] Defense Acquisition University, "MOSA Defense Acquisition Guidebook, Ch 3-2.4.1." [Online]. Available: https://www.dau.edu/guidebooks/Shared%20Documents%20HTML/Chapter%203%20Systems%20Engineering.aspx#toc20
[2] The Linux Foundation Projects, "Open Container Initiative," [Online] Available at: https://opencontainers.org.
[3] Cloud Native Computing Foundation, "Software Conformance," [Online] Available at: https://www.cncf.io/certification/software-conformance/

## Play 4: Adopt a Capability Model, not a Maturity Model

Google's DORA research program advocates that rather than use a maturity model, research shows that a capability model is a better way to both encourage and measure performance improvement.[4,5] Multiple studies have shown that four key metrics support software development and delivery performance.[6] The two categories are *tempo metrics* and *stability metrics*. Under tempo, measure the *deployment frequency* and the *lead time from commit to production deployment*. Under stability, measure the *mean time to recover from downtime or mean time to restore (MTTR)* and the *change failure rate (or percentage)*.

| **Metric** | **High Performers** | **Medium Performers** | **Low Performers** |
|---|---|---|---|
| **Deployment frequency** – How often the organization deploys code. | One demand (multiple deploys per day) | Between once per week and once per month | Between once per week and once per month |
| **Change lead time** – Time it takes to go from code commit to code successfully running in production. | Less than one hour | Between one week and one month | Between one week and one month |
| **Mean time to recover (MTTR)** – Time it takes to restore service when a service incident occurs (e.g., unplanned outage, service impairment). | Less than one hour | Less than one day | Between one day and one week |
| **Change failure rate** – Percentage of changes that results in either degraded service or requires remediation (e.g., leads to service impairment, service outage, requires a hotfix, rollback, patch, etc.) | 0-15% | 0-15% | 31-45% |

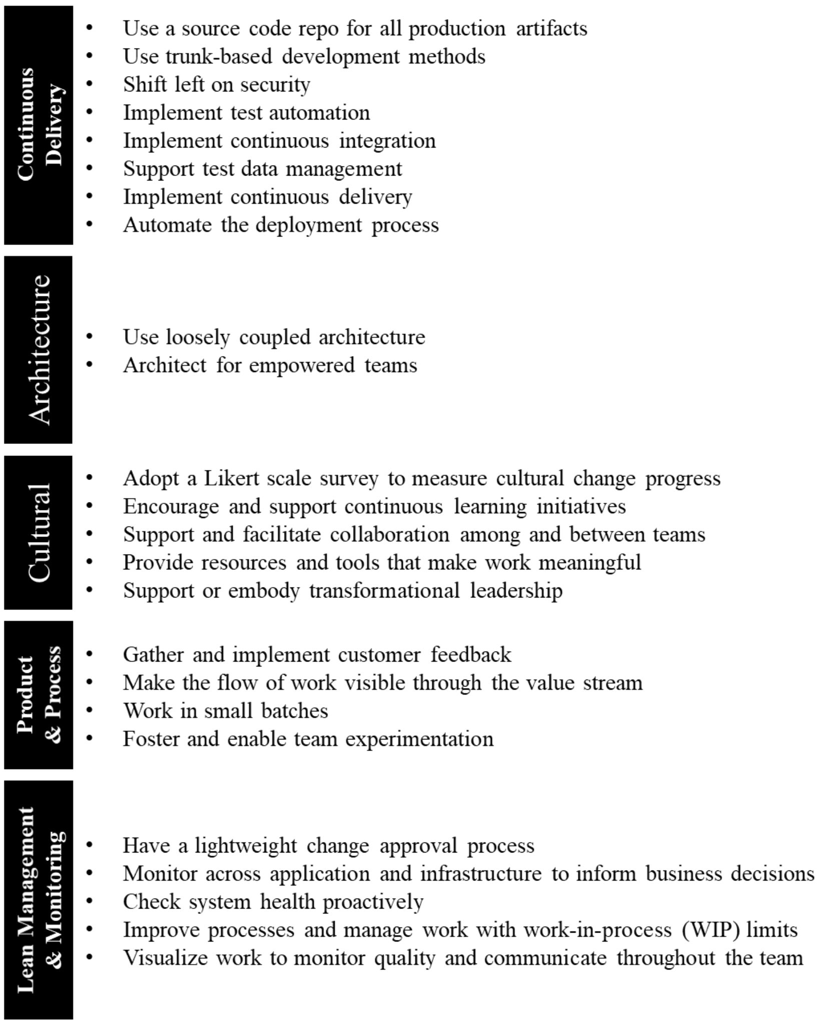
## Checklist

- ☐ Become fluent with the four key metrics: deployment frequency, lead time, MTTR, and change failure rate.
- ☐ Evaluate your project and organization on each metric to measure DevSecOps capability progress.
- ☐ Continuously strive to improve each metric through process and automation improvements.
- ☐ Read *The DevOps Handbook* and learn *The Three Ways*7

[4] Google Cloud, "Explore DORA's research program," [Online]. Available at: https://www.devops-research.com/research.html.
[5] N. Forsgren, J. Humble, G. Kim, and, "Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations." 2018.
[6] DevOps Research and Assessment (DORA), "Accelerate: State of DevOps 2019." 2019, [Online]. Available at: https://services.google.com/fh/files/misc/state-of-devops-2019.pdf
[7] G. Kim, J. Humble, P. Debois, and J. Willis, "The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations." IT Revolution Press, Oct. 06, 2016.

## Play 5: Drive Continuous Improvement through Key Capabilities

There are 24 key capabilities that drive improvements across both the DevSecOps team and its organization.[8] The capabilities are organized into five broad categories: Continuous Delivery, Architecture, Product and Process, Lean Management & Monitoring, and Cultural. **Cultural change is often the hardest thing to address.** The 24 key capabilities include:



**Continuous Delivery**

- Use a source code repo for all production artifacts
- Use trunk-based development methods
- Shift left on security
- Implement test automation
- Implement continuous integration
- Support test data management
- Implement continuous delivery
- Automate the deployment process

**Architecture**

- Use loosely coupled architecture
- Architect for empowered teams

**Cultural**

- Adopt a Likert scale survey to measure cultural change progress
- Encourage and support continuous learning initiatives
- Support and facilitate collaboration among and between teams
- Provide resources and tools that make work meaningful
- Support or embody transformational leadership

**Product & Process**

- Gather and implement customer feedback
- Make the flow of work visible through the value stream
- Work in small batches
- Foster and enable team experimentation

**Lean Management & Monitoring**

- Have a lightweight change approval process
- Monitor across application and infrastructure to inform business decisions
- Check system health proactively
- Improve processes and manage work with work-in-process (WIP) limits
- Visualize work to monitor quality and communicate throughout the team

## Checklist

- ◻ Read *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations.*
- ◻ Pay special attention to driving the cultural changes necessary for successful transformation.

[8] N. Forsgren, J. Humble, G. Kim, and, "Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations." 2018.

## Play 6: Establish a Software Factory

All custom software development should be driven through the software factory construct using DevSecOps. There are several ways to instantiate a DoD DevSecOps Software Factory / Platform. At this time, the option with the least friction is to use the DoD-approved DevSecOps Managed Service Provider (MSP), Platform One. Platform One is operated as an authorized-to-use Platform with integrated continuous authorization to operate (cATO) practices. It leverages several enterprise-class services, including *Iron Bank* as a recognized DoD hardened artifact repository and *Repo One* for source code management.

Another option is to establish a software factory using a Cloud Service Provider (CSP) with a DoD ATO or Provisional Authorization (PA). Leverage the CSP's managed services, ideally through IaC practices, to establish a DevSecOps Software Factory. *Figure 1* illustrates the key phases of building a software factory:

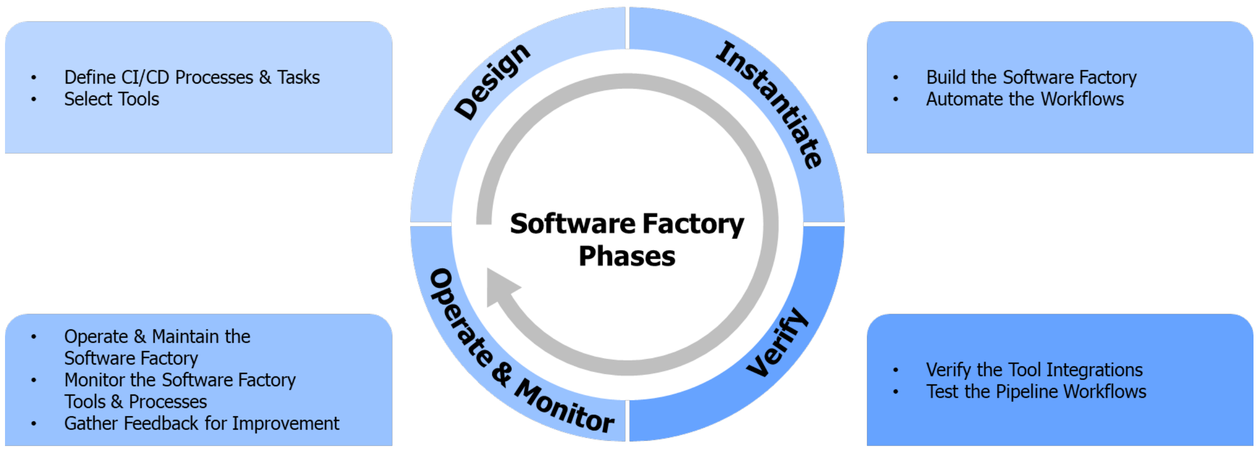

Define CI/CD Processes & Tasks
Select Tools
Design
Instantiate
Build the Software Factory
Automate the Workflows
Software Factory Phases
Operate & Maintain the Software Factory
Monitor the Software Factory Tools & Processes
Gather Feedback for Improvement
Operate & Monitor
Verify
Verify the Tool Integrations
Test the Pipeline Workflows

*Figure 1 Software Factory Lifecycle Phases*

## Checklist

- ◻ Recognize that a software factory must align to the **DoD Enterprise DevSecOps Strategy**, comply with all *required* **DevSecOps Tools and Activities Guidebook**, and clearly identify its interconnects between the various layers, as defined within the **DevSecOps Fundamentals** document.
- ◻ Software factories are inherently designed to be multi-tenet, and they are expensive to build and operate; establish clear reasons why a new factory is required over adopting an existing factory.

## Play 7: Define a Meaningful DevSecOps Pipeline

Each software factory executes multiple DevSecOps Pipelines, where a pipeline is analogous to a manufacturing assembly line. Each pipeline is dedicated to a specific process uniquely tailored for the artifact being produced. There are no *one size fits all* solutions for cybersecurity testing. Therefore, every DevSecOps pipeline is a collection of process workflows and scripts running on a set of DevSecOps tools operating in unison with their associated software factory. The design of each pipeline must clearly identify the process flows and automation activities across the various DevSecOps stages, depicted below in *Figure 2.*

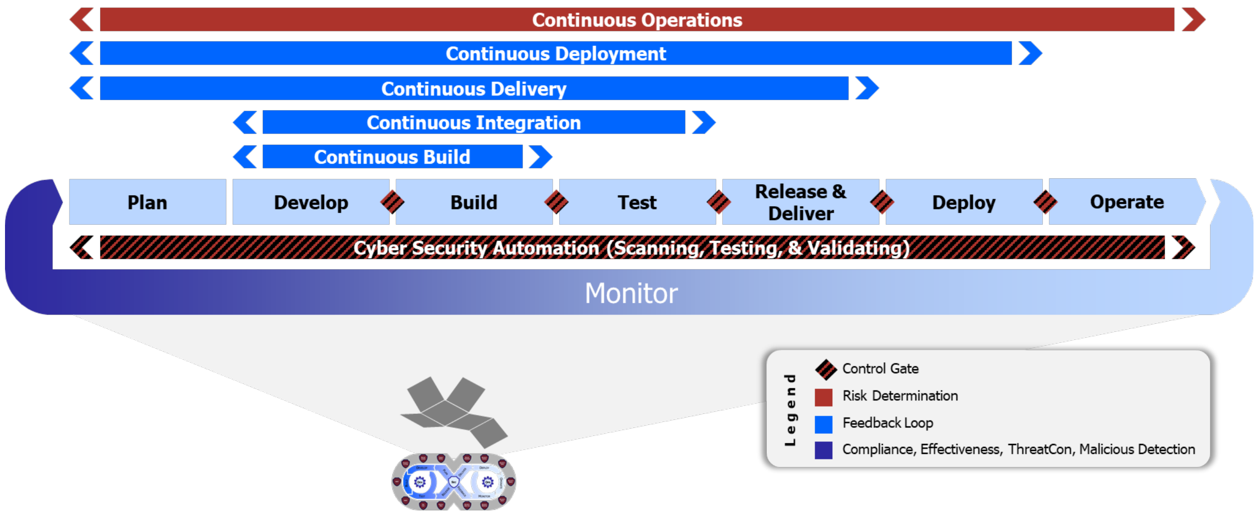


*Figure 2 Unpacked DevSecOps infinity loop showing continuous feedback loops*

## Checklist

- ☐ Read **DoD Enterprise DevSecOps Fundamentals** document.
- ☐ Read **DevSecOps Tools and Activities Guidebook.**
- ☐ Define a software lifecycle within the pipeline that uses management processes that meets the unique needs of the mission environment, system complexity, system architecture, software design choices, risk tolerance level, and system maturity level.
- ☐ Do not try to implement the pipeline using a “big bang” approach – start small, iterate, automate repetitive processes.
- ☐ Recognize the value of the continuous feedback loops across the software lifecycle phases.
- ☐ Work closely with the AO to understand precisely what each control gate must validate before an artifact can be promoted to the next lifecycle phase.
- ☐ Measure capabilities across each of the lifecycle phases.

## Play 8: Adapt an Agile Acquisition Policy for Software

The Office of Acquisition Enablers (AE) is a new organization within the Office of the Under Secretary of Defense for Acquisition & Sustainment (A&S). The office is the lead for enabling innovative acquisition approaches that deliver warfighting capability at the speed of relevance. *DoD Instruction 5000.02, Operation of the Adaptive Acquisition Framework,* restructures defense acquisition guidance to improve process effectiveness and implement the Adaptive Acquisition Framework.[9] As part of this framework, DoD Instruction 5000.87, *Operation of the Software Acquisition Pathway*, became effective October 2, 2020.[10] The 5000.87 instruction:

- Establishes the Software Acquisition Pathway as the preferred path for acquisition and development of software-intensive systems.
- Simplifies the acquisition model to enable continuous integration and delivery of software capability on timelines relevant to the warfighter/end user.
- Establishes business decision artifacts to manage risk and enable successful software acquisition and development.

Defense Acquisition University (DAU) provides training in the form of an interactive web application that educates the audience specifically on the Software Acquisition Pathway, where agile software acquisition processes are discussed in the context of acquisition personnel. For more information: https://aaf.dau.edu/aaf/software/

## Checklist

- ◻ Review DoDI 8000.87 to understand the formal definition of what constitutes a “software-intensive” system.
- ◻ Review the DIB SWAP study’s key findings.[11]
- ◻ Review the acquisition guidance in the TechFAR hub, https://techfarhub.cio.gov/.
- ◻ Recognize that the software can be acquired via DoDI 8000.87, while other program elements can be acquired through different pathways.
- ◻ Leverage Enterprise Level Services as a first choice, if available, before creating unique services.
- ◻ Ensure your acquisition plan recognizes that technology enhancements *never* end.
- ◻ Do not lock technical requirements into legal contracts; enable new technologies.

[9] Office of the Under Secretary of Defense for Acquisition and Sustainment, “DoD Instruction 5000.02, Operation of the Adaptive Acquisition Framework.” Jan. 23, 2020, [Online]. Available: https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500002p.pdf?ver=2020-01-23-144114-093.

[10] Office of the Under Secretary of Defense for Acquisition and Sustainment, “DoD Instruction 5000.87, Operation of the Software Acquisition Pathway.” Oct. 20, 2020, [Online]. Available: https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500087p.PDF?ver=virAfQj4v_LgN1JxpB_dpA%3D%3D.

[11] Defense Innovation Board (DIB), “Software Acquisition and Practices (SWAP) Study.” May 03, 2019, [Online]. Available: https://innovation.defense.gov/software.

## Play 9: Tirelessly Pursue Cyber Resilience

Cyber Resilience is “the ability to anticipate, withstand, recover from, and adapt to adverse conditions, stresses, attacks, or compromises on the systems that include cyber resources.”[12] A primary goal of DevSecOps adoption is to “bake-in” cyber resiliency into applications as part of the software factory’s DevSecOps pipeline process.

Cybersecurity touches each of the eight phases of the DevSecOps lifecycle, and the various control gates serve as Go/No-Go decision points. As discussed in Play 7: Define a Meaningful DevSecOps Pipeline, the precise set of testing and processes will vary from pipeline to pipeline. However, all pipelines must use these control gates to ensure that cybersecurity is both “baked in” and transparently identified. Culturally, the AO and their staff must pivot from relying on post-process paperwork evaluations to near real-time continuous monitoring of both the software factory’s pipelines and the production environment performance metrics.

Moving to DevSecOps includes moving towards a Continuous Authorization to Operate (cATO) for an application developed using DevSecOps processes, including a software factory with a CI/CD pipeline. cATO is equivalent to an ongoing authorization as defined in NIST 800-137, and it is fundamentally related to the ongoing understanding and acceptance of security and privacy risk.[13] Every cATO is centered around a transparently defined and well-understood continuous monitoring program.

A separate guidebook on cATO is forthcoming; it centers around assessment and authorization of the *platform*, assessment and authorization of the *process* (including continuous monitoring), and finally, assessment and authorization of the *team*.

## Checklist

- ☐ Do not use Fast Track Authority to Operate for software produced by a DevSecOps software factory CI/CD pipeline.
- ☐ Pursue cyber resilience at each phase of the DevSecOps lifecycle.
- ☐ Understand Recommendation B6, “Shift from certification of executables for low- and medium-risk deployments to certification of code/architectures and certification of the development, integration, and deployment toolchain.”
- ☐ Establish a continuous monitoring program.
- ☐ Partner with your AO and help them move to near real-time metrics dashboard.

[12] R. Ross, V. Pillitteri, R. Graubart, D. Bodeau, and R. McQuiad, “NIST Special Publication 800-160 Volume 2, Developing Cyber Resilient Systems: A Systems Security Engineering Approach.” 2019–Nov., [Online]. Available: https://doi.org/10.6028/NIST.SP.800-160v2.

[13] National Institute of Standards and Technology, “Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy (SP 800-37 Rev. 2).” Dec. 2018, [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-37/rev-2/final.

## Play 10: Shift Operational Test and Evaluation (OT&E) Left into the Pipeline

The Defense Innovation Board succinctly summed the goal of this play like this: “Speed and cycle time are the most important metrics for managing software. DoD must be able to deploy software faster without sacrificing its abilities to test and validate software.”[11]

OT&E activities are intended to gather data that helps leadership make informed decisions. The value of shifting these activities into the software factory’s pipeline is that risk is reduced by finding problems early and fixing them fast *while* the change that created the problem is still in the forefront of the developer’s mind. Integration continues to be difficult to achieve between disparate systems, and the push for access to raw data to feed AI/ML algorithms is increasing, not decreasing. The ability to ensure these integrations work earlier in the process, not as a bolt-on after-the-fact integration, drives the delivery of relevant software at the speed of operations.

Tests must be planned, and they are formally identified within the DoDI 5000.87.[10] Testers should receive formal training in both Agile and DevSecOps to ensure they are fully integrated team members. Further, the DevSecOps culture emphasizes that *everyone is* responsible for testing and quality regardless of team position or job title. The test plan must plan and identify the metrics that best reflect functional and non-functional requirements and how the metrics will be collected in an automated fashion, respectively. Lastly, and most importantly, the end user or their representative must be closely involved in all aspects of testing and acceptance of an artifact as it transitions through the CI/CD pipeline.

### Common Testing Categories

- Unit and Functional Testing.
- Integration Testing.
- Performance Testing.
- Interoperability Testing.
- Deployment Testing (normally conducted in a staging environment).
- Operational Testing (normally conducted in a production environment).
- Static Application Security Testing (SAST).
- Dynamic Application Security Testing (DAST).
- Interactive Application Security testing (IAST).
- Runtime Application Self-Protection (RASP).

### Checklist

- ☐ Start OT&E planning at the inception of the program to influence strategy, requirements, RFPs, etc.
- ☐ Establish the plan to automate the collection of test data metrics in the first sprint.
- ☐ Incessantly work to compress test reporting timelines as much as possible to speed corrections.
- ☐ Include operational users in both Deployment and Operational Testing.
- ☐ Incorporate all forms of Application Security Testing in the pipeline to ensure cyber resilience.
- ☐ Consider functional, non-functional, and cyber testing at each of the eight phases of the DevSecOps lifecycle.