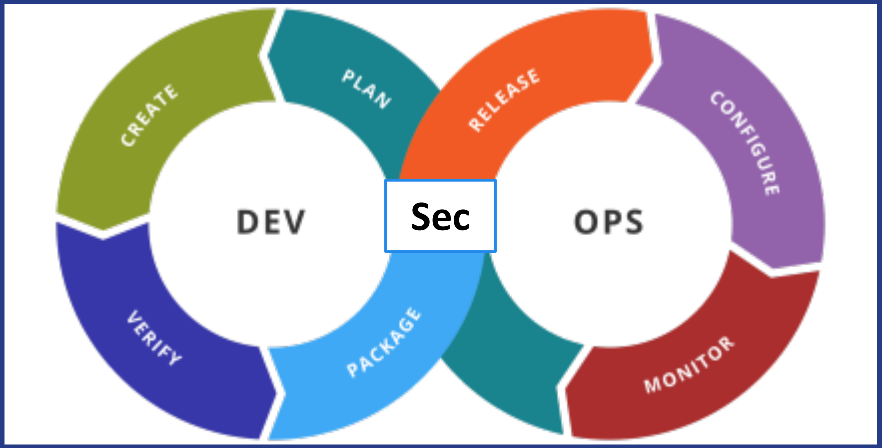
DevSecOps: Injecting Security into DevOps



Graphic: commons.wikimedia.org
With DoD modifications

AFCEA/GMU Critical Issues in C4I Symposium
21-22 May 2019

Gilliam E. Duvall, PhD, Engr
Data Security Strategies, LLC

Data Security Strategies, LLC
www.datasecuritystrategies.com

You need more than a security plan, you need a Strategy…

Agenda

- What is DevOps?
- How Does DevSecOps Work?
    - Role of Culture, Process & Technology
    - Use of metrics & KPIs
- DoD Software Development
    - Current State
    - Desired State
    - DevSecOps example
- DIB Recommendations: for Future DevOps & DevSecOps

What is DevOps?

"DevOps is the process of continuously improving software products through rapid release cycles, global automation of integration and delivery pipelines and close collaboration between teams."

- The goal of DevOps is to **shorten the time** and **reduce the cost** of transforming an idea into a product that customers use
- DevOps makes heavy use of automated processes to speed up development and deployment
- An organization able to build software four times faster than its competitor has a significant competitive advantage
- History has shown that customers value innovative products that may be incomplete at first, but improve quickly and steadily
- Organizations adopt DevOps to reduce the cost and latency of development cycles, and **answer their customer's demands**.

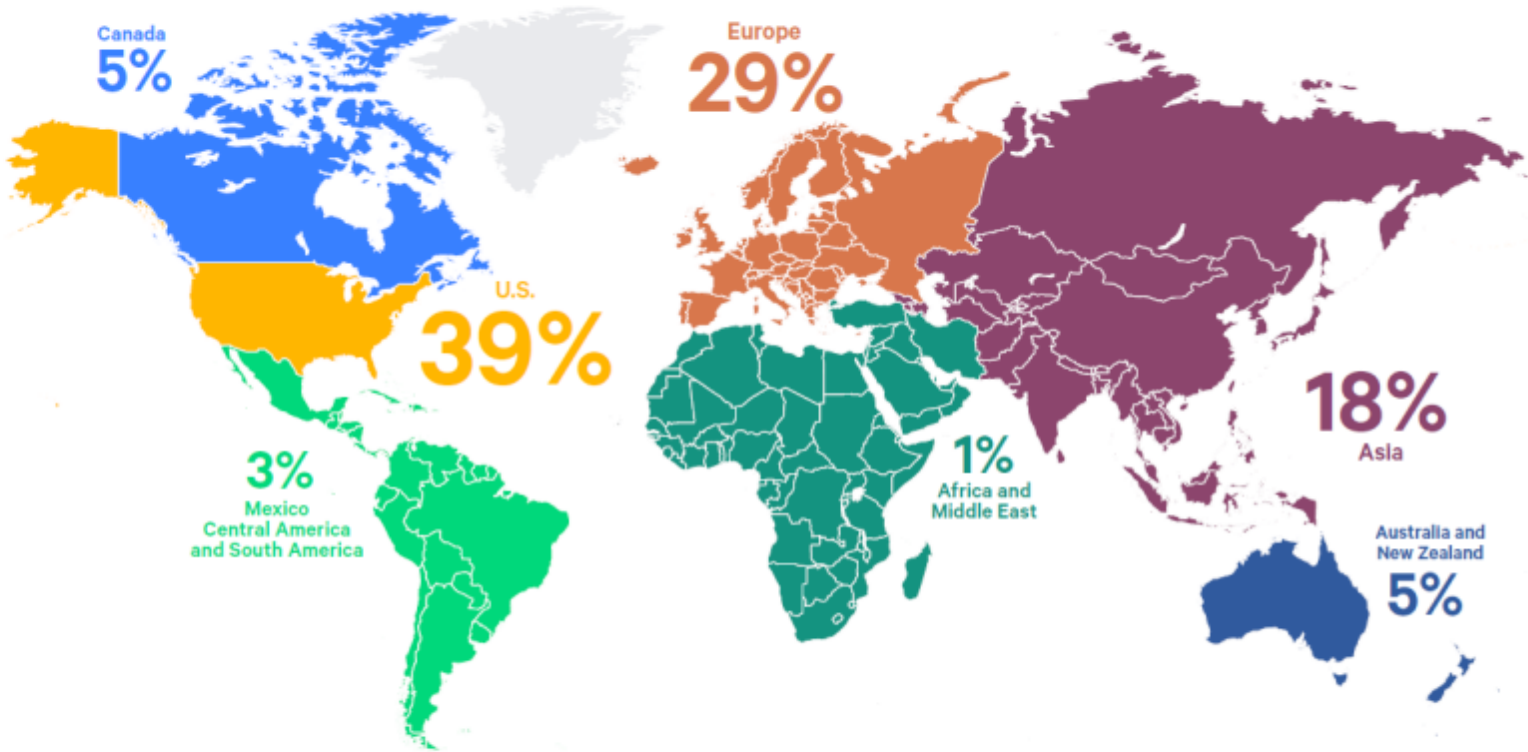
https://freecontent.manning.com/where-security-meets-devops-test-driven-security/

01000111 01000101 01000100

Global DevOps Survey Participation

Puppet | State of DevOps Report 2018

Responses by global region

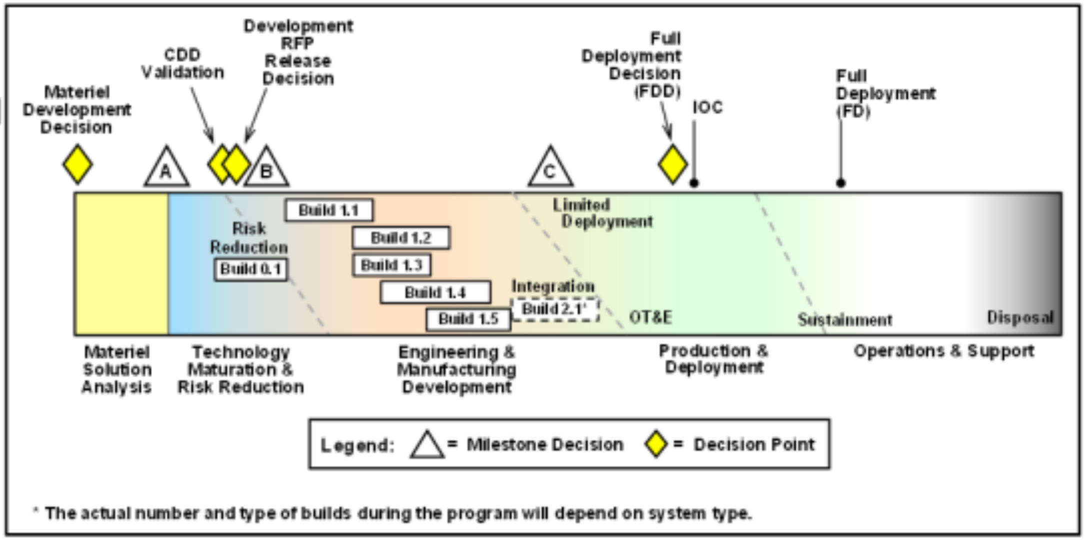


DevOps Concepts

- Combining Development & Operations processes
  - Best solution combining **speed & agility** to managing rapid change of coding (**code change velocity**) in business applications
- Coding evolutions happen in **sprints,** with **cadence**; fast application delivery to customers with business improvements in an **unconstrained software change** process
- The ability to **compete more effectively** than using legacy IT methods
  - An **evolutionary successor** in the world of software development life cycle (SDLC) models (e.g., Waterfall and Agile)
  - **High performing DevOps** - **automated** code development /testing/delivery
  - Infrastructure-as-code **(IaC) – refresh cloud environment using machine readable code,** vice physical hardware configuration
- Continuous integration/continuous delivery **(CI/CD)**

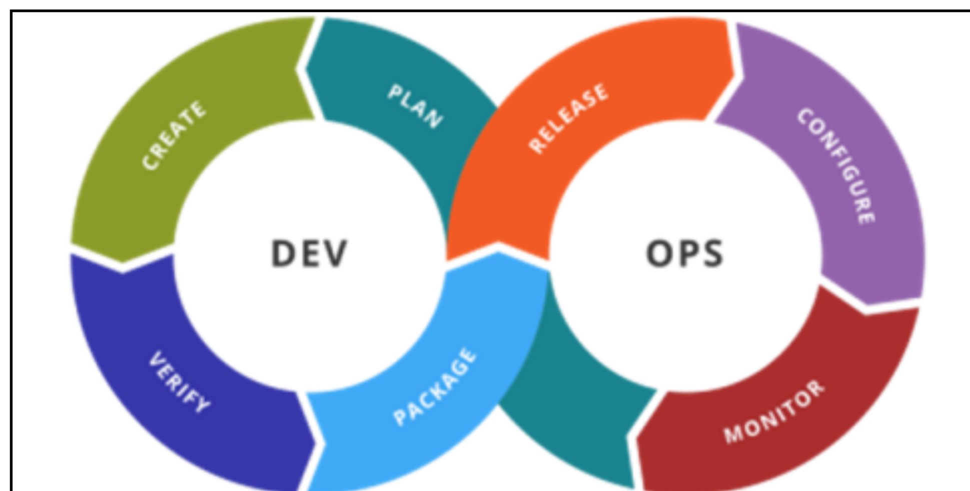
01000111 01000101 01000100

Traditional Build SDLC *vs* a DevOps Loop



DoD Instruction 5000.02, with Change 3, 2017

'As Is' state: Slow & rigid Build Schedules



'To Be' state: Fast & agile CI/CD

Graphic: commons.wikimedia.org

DSS
01000111 01000101 01000100

Stages of DevOps Evolution

	Defining practices* and associated practices	Practices that contribute to success
Stage 0	• Monitoring and alerting are configurable by the team operating the service. • Deployment patterns for building applications or services are reused. • Testing patterns for building applications or services are reused. • Teams contribute improvements to tooling provided by other teams. • Configurations are managed by a configuration management tool.	
Stage 1	• **Application development teams use version control.** • **Teams deploy on a standard set of operating systems.**	• Build on a standard set of technology. • Put application configurations in version control. • Test infrastructure changes before deploying to production. • Source code is available to other teams.
Stage 2	• **Build on a standard set of technology.** • **Teams deploy on a single standard operating system.**	• Deployment patterns for building applications and services are reused. • Rearchitect applications based on business needs. • Put system configurations in version control.
Stage 3	• **Individuals can do work without manual approval from outside the team.** • **Deployment patterns for building applications and services are reused.** • Infrastructure changes are tested before deploying to production.	• Individuals can make changes without significant wait times. • Service changes can be made during business hours. • Post-incident reviews occur and results are shared. • Teams build on a standard set of technologies. • Teams use continuous integration. • Infrastructure teams use version control.
Stage 4	• **System configurations are automated.** • **Provisioning is automated.** • Application configurations are in version control. • Infrastructure teams use version control.	• Security policy configurations are automated. • Resources made available via self-service.
Stage 5	• **Incident responses are automated.** • **Resources available via self-service.** • Rearchitect applications based on business needs. • Security teams are involved in technology design and deployment.	• Security policy configurations are automated. • Application developers deploy testing environments on their own. • Success metrics for projects are visible. • Provisioning is automated.

*** The practices that define each stage are highlighted in bold font.**

Puppet | 2018 State of DevOps Report

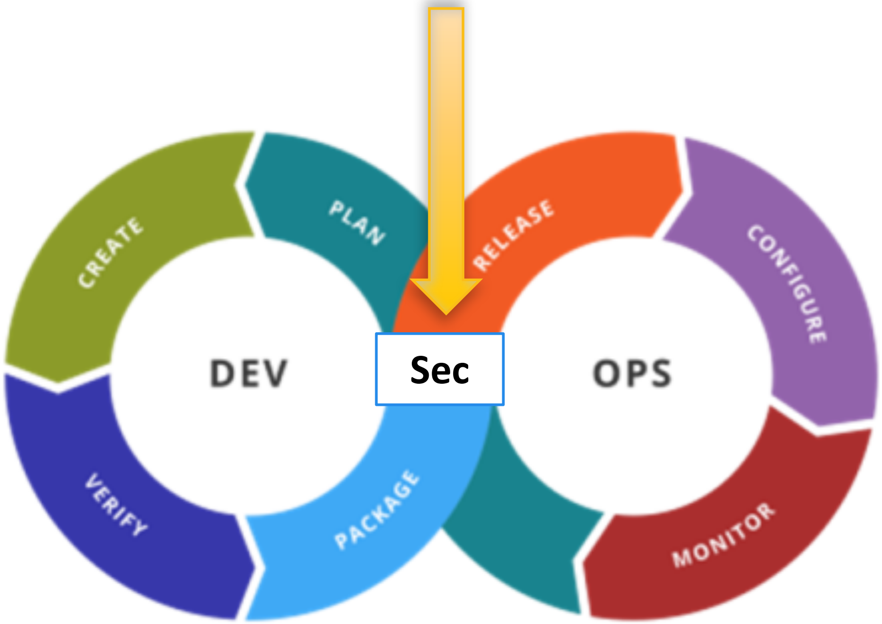
Agenda

- What is DevOps?
- **How Does DevSecOps Work?**
  - **Role of Culture, Process & Technology**
  - **Use of metrics & KPIs**
- DoD Software Development
  - Current State
  - Desired State
  - DevSecOps example
- DIB Recommendations: for Future DevOps & DevSecOps

What is DevSecOps?

- Integrates security with CI/CD into daily mission/business application development
  - 'Secure & safe' practices are injected into each of the seven 'Fast & agile' phases
- DevSecOps concepts integrate well with enterprise objectives to incorporate:
  - Cost savings
  - Automation
  - Cloud adoption
- Studies* have indicated DevSecOps high performers spend 50% less time remediating security issues

'Secure & safe' Practices



DevOps
CI/CD
'Fast & agile' phases

Graphic: commons.wikimedia.org w/DoD modification

*Puppet | 2016 State of DevOps Report

DevSecOps Principles

- Successful implementation involves
  - Culture (people)
  - Processes (communication, feedback)
  - Technology (to deliver security at developer's speed)
- "Moves security to the left" by empowering developer teams to 'do' security
- Integrates security & QA teams into the development process sooner
- DevSecOps principles to follow:
  1. Automate security in all phases (esp. testing, monitoring, audit & response)
  2. Allow developers to fail quickly (Test Driven Security)
  3. No false alarms (threshold mgmt.)
  4. Build security champions (within the developer community)
  5. Process transparency (communicates "normal")

Graphic: commons.wikimedia.org w/DoD modification

DevSecOps Cultural Change

Collaboration (teaming) between Developers, Operators & the cybersecurity SMEs

In a DevSecOps world, security professionals will have:

NEW RESPONSIBILITIES	NEW SKILL REQUIREMENTS
Enable developers to find and fix security-related code defects	Ability to provide remediation coaching and guidance on security-related code defects
Govern the use of open source components	Basic understanding of application development and why and how third-party components are used
Implement developer training on secure coding	Understanding of the basics of software development
Manage and report on application security policy, KPIs and metrics	The ability to measure meaningful metrics at each point in the SDLC process
Understand the requirements for security testing solutions in a DevSecOps environment — including the need for immediacy and accuracy of results to avoid impacting the delivery cycle — and enable dev to use these solutions	Basic understanding of developer role and tools, and the operation of a modern software delivery pipeline/factory
Create developer security champions	Be empathetic and consultative

VERACODE GUIDE - THE SECURITY PROFESSIONAL'S ROLE in a DevSecOps World

DevSecOps Process Elements

Processes to improve: Communication, Collaboration, Reporting, Measurements, Concept Integration

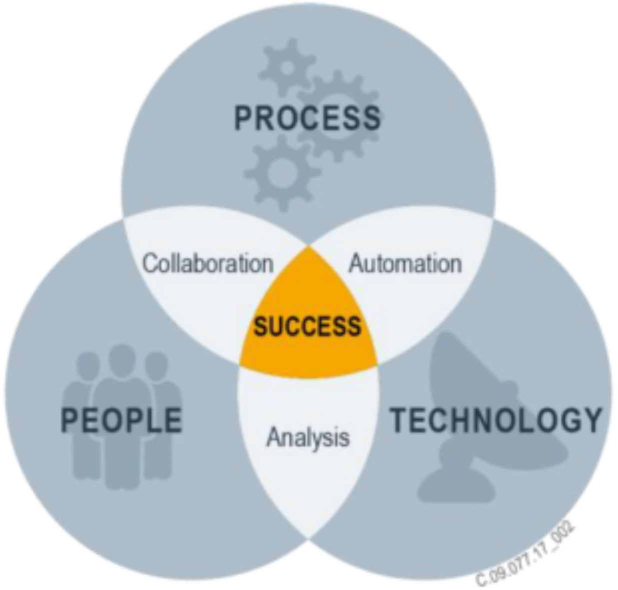
Collaboration: People + Process: Every participant in the process must understand the entire process and their contribution.

Automation: Process + Technology. Technology must support the process and eliminate repetitive or tedious tasks

Analysis: Technology + People. Technology must improve workflow and the analysis of bottlenecks in order to improve results with cross-functional skills.

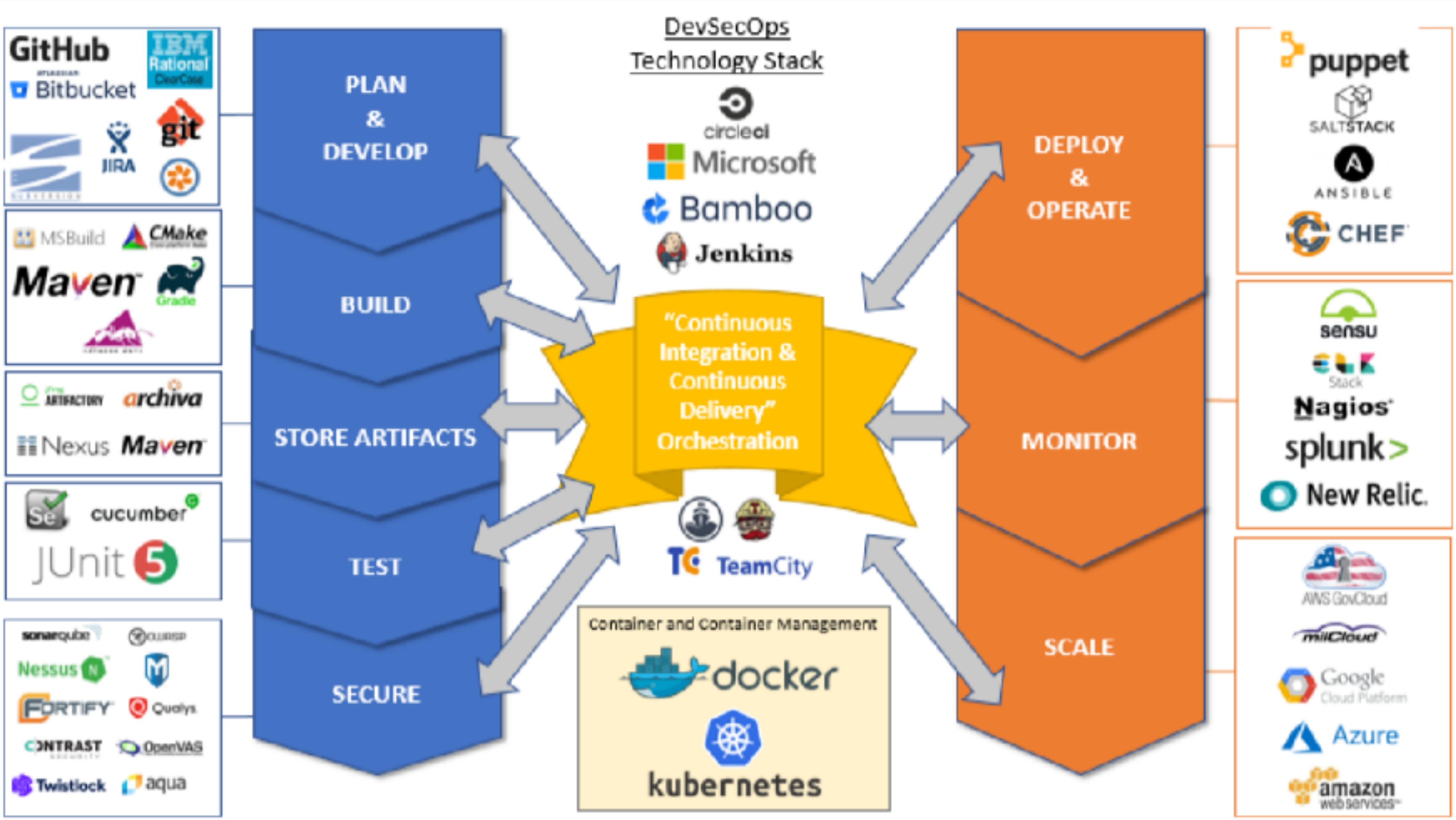
Success = People + Process + Technology: With a successful implementation, people are able to collaborate effectively and drive results efficiently, thus overcoming the "silo effect."

FIGURE 1: PEOPLE, PROCESS, AND TECHNOLOGY



JIDO, SecDevOps CONOPS, Ver 1.0 , 2017

DevSecOps Technology Stack (example)



DIB SWAP Study Final Release, May 2019

DevSecOps Continuous Monitoring

Continuous monitoring is comprised of metrics and Key Performance Indicators (KPIs)

Elements of a Logging Pipeline

Collect	Stream	Analyze	Store	Access
Log events are recorded from various components of the infrastructure.	Log records are captured and routed to the corresponding Layer.	Log records are inspected in order to detect anomalies and raise alerts.	Log records are stored in short & long-term storage facilities.	Log administrative console to access and review logs/alerts.

Ref: ISC2

- The ideal Logging Pipeline is <u>automated</u> and allows analyze of types of traffic, application-level security metrics & security incidents in real-time
- **KPIs reflect the performance of a DevSecOps program**

01000111 01000101 01000100

DevSecOps KPI Monitoring & Testing

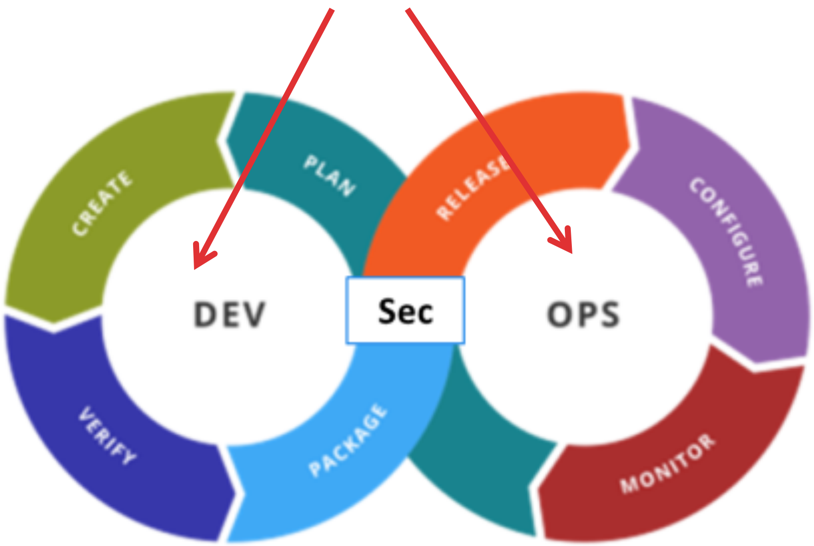
Based on mission/business needs and compliance requirements

Some typical metrics & KPIs

- Availability
- Change Failure
- Change Lead Time
- Change Volume
- Customer Issue Resolution Time
- Customer Issue
- Defect Burn Rate
- Defect Density
- Deployment Frequency
- Logging Availability
- Mean Time Between Failures (MTBF)
- Mean Time to Failure (MTTF)
- Mean Time to Recovery (MTTR)
- Number of Functional/Acceptance Tests
- Number of Passed/Failed Security Tests
- Number of Unit/Integration Tests
- Security Benchmark Deviation
- Security Controls
- Test Coverage
- Time to Patch
- Time to Value
- Vulnerability Patching Frequency
- Vulnerability Patching Lead Time

Ref: ISC[2]

IAST & RASP



Graphic: commons.wikimedia.org w/DoD modification

Interactive App Sec Testing (IAST) / Run-time App Sec Protection (RASP)*

- Continuous security services using embedded agents
- Real-time integrated testing, monitoring & protection

* www.softwaresecured.com

01000111 01000101 01000100

Derived DevSecOps Performance Metrics

Defect Density: the number of bugs identified divided by the codebase of an application. Used to set goals & measure progress within teams and within specific applications or services

Defect Burn Rate: amount of time to fix vulnerabilities in an application. Focus less on the quantity of defects and instead turn to how quickly those defects are addressed by the team.

Top Vulnerability Types and Top Recurring Bugs: security teams track top vulnerability types will be in a much better position to help developers make long-term improvements in the way they code.

Number of Adversaries per Application: security teams that want to improve their developer's risk IQ should be asking them how many adversaries they think an application actually has.

Adversary Return Rate: this metric gets developers invested in thinking about how applications are being attacked and how often an adversary is using the same tactics, techniques and procedures.

Time to Value: Time between a feature request (user story creation) and realization of business value from that feature.

01000111 01000101 01000100

Ref: https://businessinsights.bitdefender.com/seven-winning-devsecops-metrics-security-should-track

DevSecOps Performance Benchmarks

2017 DevOps Research & Assessment (DORA) Report	High performance	Medium performance	Low performance
Deployment frequency How often does your organization deploy code?	On demand (multiple deploys per day)	Between once per week and once per month	Between once per week and once per month
Lead time for changes What is your lead time for changes (i.e., how long does it take 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) How long does it generally take to restore service when a service incident occurs (e.g., unplanned outage, service impairment)?	Less than one hour	Less than one day	Between one week and one day
Change failure rate What percentage of changes results either in degraded service or subsequently requires remediation (e.g., leads to service impairment, service outage, requires a hotfix, rollback, fix forward, patch)?	0-15%	0-15%	31-45%

* Low performers were lower on average (at a statistically significant level), but had the same median as the medium performers (2017 DevOps Report)

01000111 01000101 01000100

Agenda

- What is DevOps?
- How Does DevSecOps Work?
  - Role of Culture, Process & Technology
  - Use of metrics & KPIs
- **DoD Software Development**
  - **Current State**
  - **Desired State**
  - **DevSecOps example**
- DIB Recommendations: for Future DevOps & DevSecOps

Defense Innovation Board – Views on Software Development

Current state – the problem:

- Software is ubiquitous and U.S. national security relies on software
  - The ability to acquire and deploy software is central to national defense and integrating with allies.
- The threats the U.S. faces change rapidly,
  - DoD's ability to adapt and respond is now determined by its ability to develop and deploy software to the field
- **The current approach to software development is a leading source of risk to DoD**
  - It takes too long, is too expensive & exposes warfighters to unacceptable risk
- Software is <u>not</u> being used to enable a more effective force, strengthen our ability to work with allies, and improve the business processes of the Department
- Nothing is changing: most of this has been said before - 1987 DSB report on military software

"Software is Never Done: Refactoring the Acquisition Code for Competitive Advantage" -- Defense Innovation Board, 3 May 2019

01000111 01000101 01000100

DoD DevOps Desired State

- **Speed and cycle time are the most important metrics for managing software**
  - DoD needs to deploy and update software that works for its users at the speed of (mission) need
  - Execute inside the OODA loop of our adversaries to maintain advantage
- **Software is made by people and for people, so digital talent matters**
  - DoD's current personnel processes and culture will not allow its military and civilian software capabilities to grow nearly enough to meet its needs.
  - New mechanisms are required.
- **Software is different than hardware (and not all software is the same)**
  - Hardware can be developed, procured, and maintained
  - Software is an enduring and evolving capability that must be supported and continuously improved throughout its lifecycle

From "Software is Never Done: Refactoring the Acquisition Code for Competitive Advantage" -- Defense Innovation Board, 3 May 2019

01000111 01000101 01000100

Defense Innovation Board
Ten Commandments of Software

1. Make computing, storage, and bandwidth abundant to DoD developers and users.
2. All software procurement programs should start small, be iterative, and build on success – or be terminated quickly.
3. The acquisition process for software must support the full, iterative life cycle of software.
4. Adopt a DevSecOps culture for software systems.
5. Automate testing of software to enable critical updates to be deployed in days to weeks, not months or years.
6. Every purpose-built DoD software system should include source code as a deliverable.
7. Every DoD system that includes software should have a local team of DoD software experts who are capable of modifying or extending the software through source code or API access.
8. Only run operating systems that are receiving (and utilizing) regular security updates for newly discovered security vulnerabilities.
9. Security should be a first-order consideration in design and deployment of software, and data should always be encrypted unless it is part of an active computation.
10. All data generated by DoD systems - in development and deployment - should be stored, mined, and made available for machine learning.

DoD must develop/deploy software as fast or faster than adversarial tactics -- building on commercially available tools and technologies for the four software types.

DIB Software Acquisition and Practices (SWAP) study, May 2019

DoD DevSecOps Metrics

- Traditional metrics within DoD is that software complexity/productivity is often estimated based on number of source lines of code (SLOC)

- While easily measured, it is not necessarily predictive of cost, schedule, or performance
  - Obsolete metrics are irrelevant at best and, at worst, could be misleading

- The process for software DevSecOps to manage travel is different from what is required to manage the software on an F-35 – suggesting a taxonomy with four types of software requiring four different approaches:
  - **Type A:** commercial ("off-the-shelf") software with no DoD-specific customization required
  - **Type B:** commercial software with DoD-specific customization needed
  - **Type C:** custom software running on commodity hardware (in data centers or in the field)
  - **Type D:** custom software running on custom hardware (e.g., embedded software)

Defense Innovation Board Metrics for Software Development, 3 May 2019

DoD DevSecOps Metrics – cont.

Alternatively, measures useful for DoD to track DevSecOps performance and drive improvement in cost/schedule, performance & security include the following:

Deployment Rate Metrics

Response Rate Metrics

Code Quality Metrics

Functionality metrics

#	Metric	Target value (by software type)[5]				Typical DoD values for SW
		COTS apps	Custom -ized SW	COTS HW/OS	Real-time HW/SW	
1	Time from program launch to deployment of simplest useful functionality	<1 mo	<3 mo	<6 mo	<1 yr	3-5 yrs
2	Time to field high priority fcn (spec → ops) or fix newly found security hole (find → ops)	N/A <1 wk	<1 mo <1 wk	<3 mo <1 wk	<3 mo <1 wk	1-5 yrs 1-18 m
3	Time from code committed to code in use	<1 wk	<1 hr	<1 da	<1 mo	1-18 m
4	Time req'd for full regression test (automat'd) and cybersecurity audit/penetration testing	N/A <1 mo	<1 da <1 mo	<1 da <1 mo	<1 wk <3 mo	2 yrs 2 yrs
5	Time required to restore service after outage	<1 hr	<6 hr	<1 day	N/A	?
6	Automated test coverage of specs/code	N/A	>90%	>90%	100%	?
7	Number of bugs caught in testing vs field use	N/A	>75%	>75%	>90%	?
8	Change failure rate (rollback deployed code)	<1%	<5%	<10%	<1%	?
9	% code avail to DoD for inspection/rebuild	N/A	100%	100%	100%	?
10	Number/percentage of functions implemented	80%	90%	70%	95%	100%
11	Usage and user satisfaction	TBD	TBD	TBD	TBD	?

Defense Innovation Board Metrics for Software Development, 3 May 2019

DoD DevSecOps Metrics – cont.

Program Management, Assessment, and Estimation Metrics

<table>
<tr><td>12</td><td>Complexity metrics</td><td colspan="4" rowspan="2">#/type of specs # programmers
structure of code #/skill level of teams
#/type of platforms #/type deployments</td><td rowspan="2">Partial/ manual tracking</td></tr>
<tr><td>13</td><td>Development plan/environment metrics</td></tr>
<tr><td>14</td><td>“Nunn-McCurdy” threshold (for any metric)</td><td>1.1X</td><td>1.25X</td><td>1.5X</td><td>1.5X each effort</td><td>1.25X Total $</td></tr>
</table>

12. Structure of specifications, code, and development and execution platforms
13. Structure and type of development & operational environment
14. Tracking software program progress
    - 25% unit cost growth and 50% total program cost growth thresholds often will not make sense for continuously developed software programs

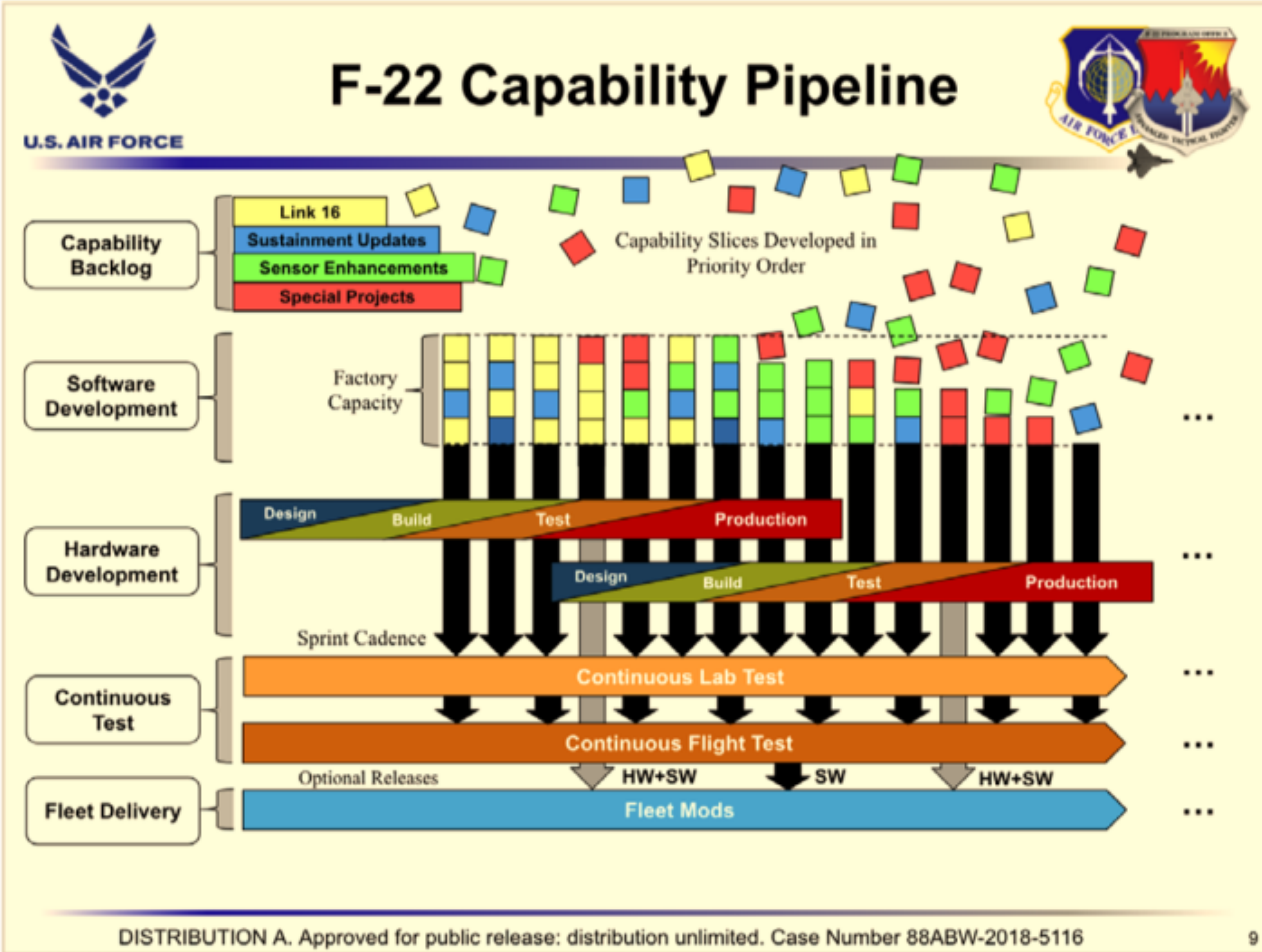
01000111 01000101 01000100

Defense Innovation Board Metrics for Software Development, 3 May 2019

DevOps on a Hardware Platform

Lessons Learned

- Culture change has been the biggest hurdle
- The program must recognize and accept that things will go wrong.
- Security controls limit flexibility and communication

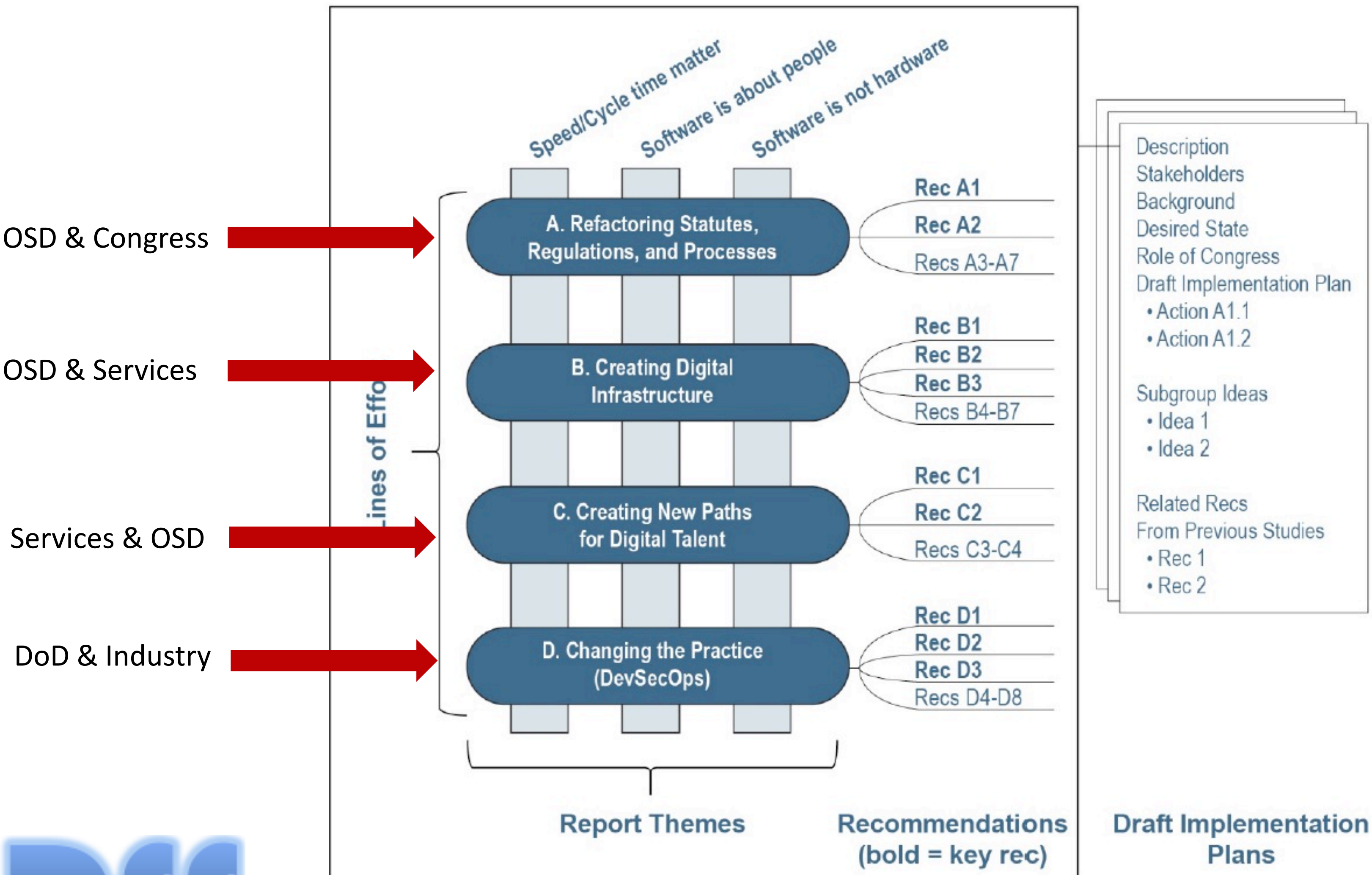


Slide image received for briefing from F22A Raptor Program Office.

Agenda

- What is DevOps?
- How Does DevSecOps Work?
  - Role of Culture, Process & Technology
  - Use of metrics & KPIs
- **DoD Software Development**
  - **Current State**
  - **Desired State**
  - **DevSecOps example**
- **DIB Recommendations: Future DoD DevOps & DevSecOps**

Future DoD DevOps & DevSecOps Concept



DIB Software Acquisition and Practices (SWAP) study, May 2019

Future DoD DevOps & DevSecOps Concepts – cont.

Line of Effort A (Congress and OSD): Refactor statutes, regulations, and processes for software	
A1	Establish one or more new acquisition pathways for software that prioritize continuous integration and delivery of working software in a secure manner, with continuous oversight from automated analytics
A2	Create a new appropriation category for software capability delivery that allows (relevant types of) software to be funded as a single budget item, with no separation between RDT&E, production, and sustainment
Line of Effort B (OSD and Services): Create and maintain cross-program/cross-Service digital infrastructure	
B1	Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field, and incentivize its use by contractors
B2	Create, implement, support, and use fully automatable approaches to testing and evaluation (T&E), including security, that allow high-confidence distribution of software to the field on an iterative basis
B3	Create a mechanism for Authorization to Operate (ATO) reciprocity within and between programs, Services, and other DoD agencies to enable sharing of software platforms, components, and infrastructure and rapid integration of capabilities across (hardware) platforms, (weapon) systems, and Services

DIB Software Acquisition and Practices (SWAP) study, May 2019

Future DoD DevOps & DevSecOps Concepts – cont.

Line of Effort C (Services and OSD): Create new paths for digital talent (especially *internal* talent)	
C1	Create software development units in each Service consisting of military and civilian personnel who develop and deploy software to the field using DevSecOps practices
C2	Expand the use of (specialized) training programs for CIOs, SAEs, PEOs, and PMs that provide (hands-on) insight into modern software development (e.g., Agile, DevOps, DevSecOps) and the authorities available to enable rapid acquisition of software
Line of Effort D (DoD and industry): Change the practice of how software is procured and developed	
D1	Require access to source code, software frameworks, and development toolchains—with appropriate IP rights—for DoD-specific code, enabling full security testing and rebuilding of binaries from source
D2	Make security a first-order consideration for all software-intensive systems, recognizing that security-at-the-perimeter is not enough
D3	Shift from the use of rigid lists of requirements for software programs to a list of desired features and required interfaces/characteristics to avoid requirements creep, overly ambitious requirements, and program delays

DIB Software Acquisition and Practices (SWAP) study, May 2019

01000111 01000101 01000100

In Conclusion

We discussed:

- What is DevOps?
- How Does DevSecOps Work?
  - Role of Culture, Process & Technology
  - Use of metrics & KPIs
- DoD Software Development
  - Current State
  - Desired State
  - DevSecOps example
- DIB Recommendations: for Future DevOps & DevSecOps

Questions?

Dr. Gil Duvall

President & CEO

Data Security Strategies, LLC

e-mail: gil@datasecuritystrategies.com

website: www.datasecuritystrategies.com