DevSecOps – Security and Test Automation

Vibha Dhawan

Rock Sabetto

March 2019

The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions, or viewpoints expressed by the author.

Approved for Public Release; Distribution Unlimited. 19-0769

Purpose

- **Clearly describe how Security and Testing can be integrated into a DevSecOps environment without compromising speed, security, or quality**

- **Provide a baseline of the terminology, methodologies, processes, environments, and automation technologies used in DevSecOps programs**

Bottom Line Upfront

- **DevSecOps Value Proposition**
  - Programs can realize significant value by implementing DevSecOps. But, testing and security should not be sacrificed
- **Shift Left**
  - Programs must truly shift Security and Test to the left to realize time and cost savings
- **Agile and DevSecOps go together**
  - DevSecOps must be fed by Agile software development. Security user stories must be part of each sprint
- **Automation is key**
  - Security and test automation can reduce delivery time, improve quality and security, and eliminate human error

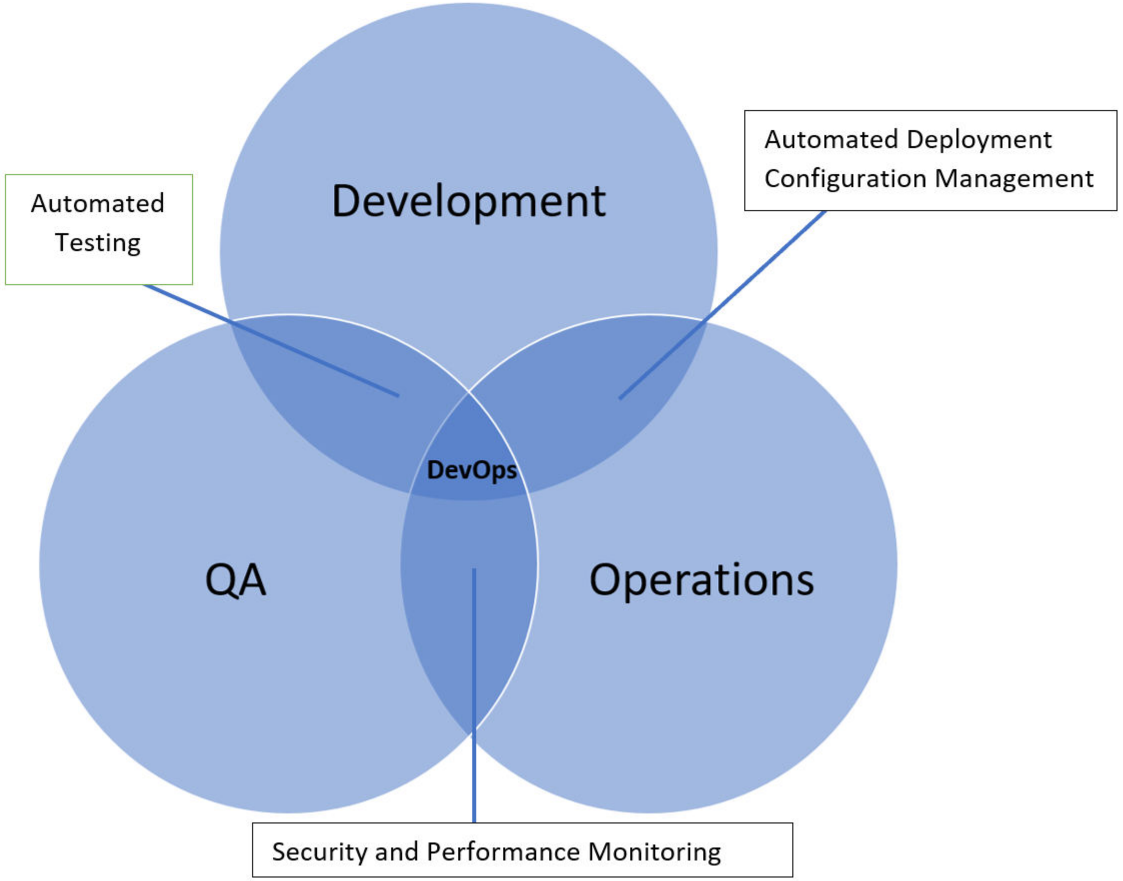
Outline

- DevSecOps Background
- Security and Testing Considerations for DevSecOps
- DevSecOps Processes and Technical Considerations
- Platform Deployment Options in DevSecOps
- Conclusion

DevSecOps Background

DevSecOps

- **DevOps** – the union of people, process and tools to achieve building, testing and releasing of software more frequently and reliably
- DevOps can also be referred to as **DevSecOps** to emphasize the importance of security
- DevSecOps <u>is not</u> Agile software development. Agile feeds new code / functionality into DevSecOps



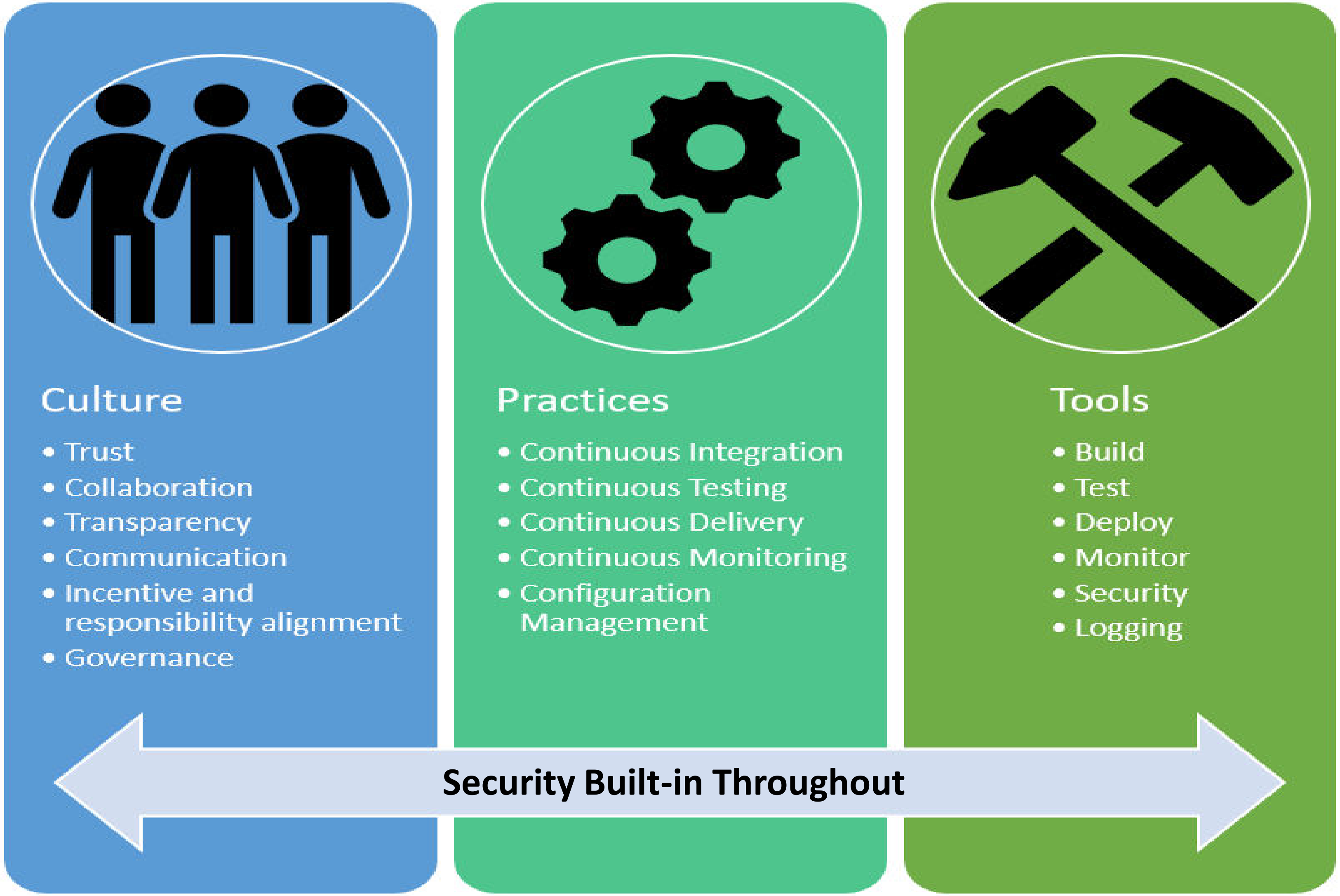
What is Agile + DevSecOps

- No silos exist between Development, Test, and Operations
  - More teamwork and information sharing
  - Better integration throughout the lifecycle
- Iterative development and deployment
  - Design, develop, test, and deploy incremental changes
  - Deploy changes to business users faster
- Automate as much as possible
  - Reduce delivery time
  - Improve quality and security
  - Eliminate human error
- Streamlined, repeatable, routinized processes
  - Faster delivery cycles – satisfied customers
- Culture, Practices and Tools all part of the DevSecOps equations
  - Empowered, trained teams leverage technologies to make it happen

DevSecOps Value Proposition

Traditional Development Challenges	DevSecOps Benefits
➢ Repetitive, Manual Processes	➢ Automated configuration and software deployment
➢ Deployment requires Days to Weeks	➢ Deployment takes Minutes
➢ Not Repeatable – Error prone	➢ Continuous and repeatable process
➢ Human intervention causes inconsistencies	➢ Consistent
➢ Frequent downtime	➢ Minimum downtime
➢ Easier – less technical skill required	➢ Harder – more technical skill needed
➢ Teams work in silos	➢ Continuous collaboration
➢ Early security testing not performed on the code	➢ Early, automated security testing during coding

DevSecOps is a union of Culture, Practices and Tools providing continuous delivery to the end user



Culture

- Trust
- Collaboration
- Transparency
- Communication
- Incentive and responsibility alignment
- Governance

Practices

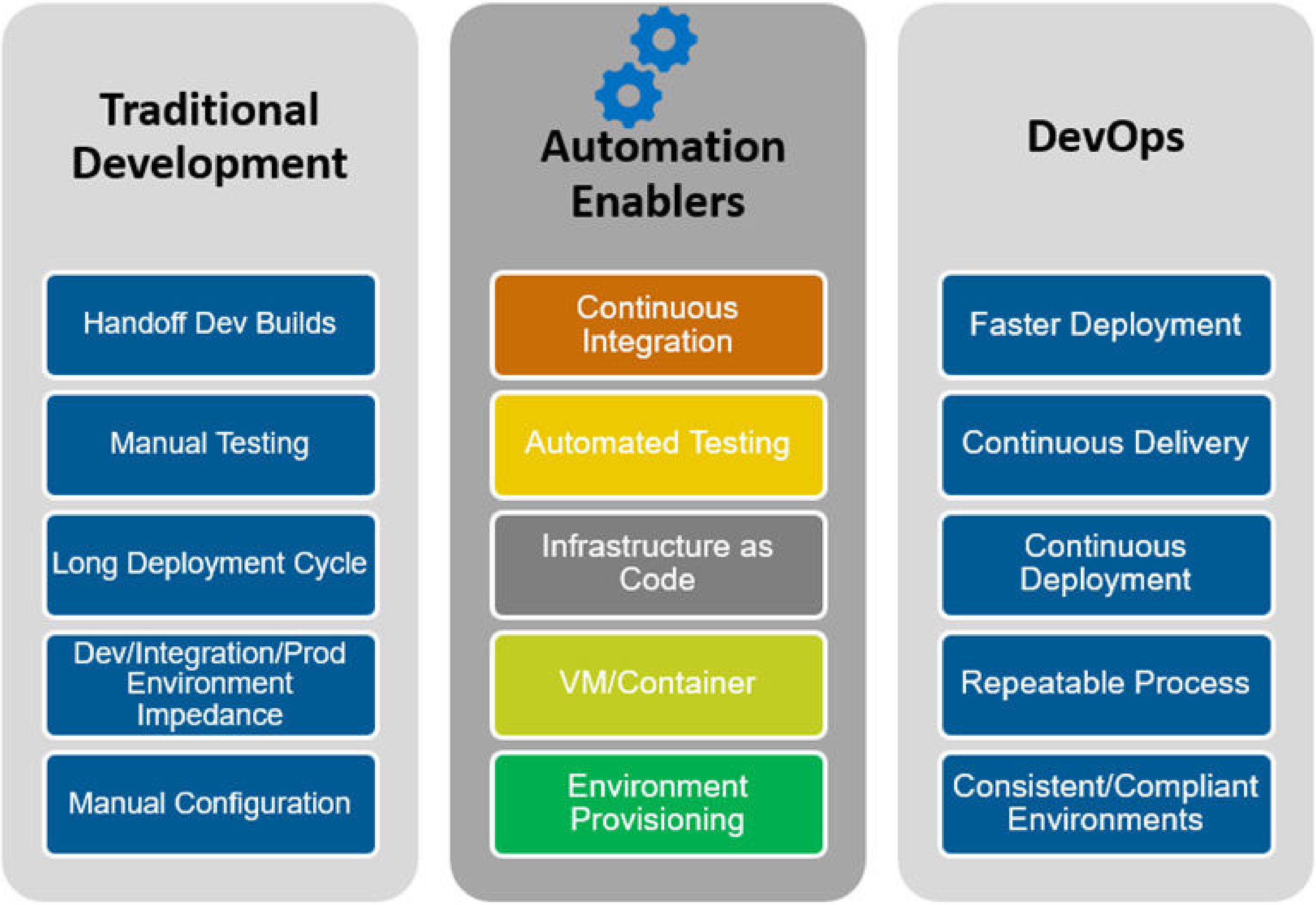
- Continuous Integration
- Continuous Testing
- Continuous Delivery
- Continuous Monitoring
- Configuration Management

Tools

- Build
- Test
- Deploy
- Monitor
- Security
- Logging

Security Built-in Throughout

DevSecOps applies automation to deliver functionality: Speed, without sacrificing security and test rigor



Traditional Development	Automation Enablers	DevOps
Handoff Dev Builds	Continuous Integration	Faster Deployment
Manual Testing	Automated Testing	Continuous Delivery
Long Deployment Cycle	Infrastructure as Code	Continuous Deployment
Dev/Integration/Prod Environment Impedance	VM/Container	Repeatable Process
Manual Configuration	Environment Provisioning	Consistent/Compliant Environments

DevSecOps Opportunities and Risks

Opportunities

- Speed and Repeatability
    - Automation of testing
    - Automation of security policy enforcement
    - Continuous improvement
- Agility
    - Can be integrated seamlessly with Agile development
    - Removes post-Agile sprint/release chokepoints

Risks

- Security and Test
    - Organizations must fundamentally relook their value proposition.
- Physical Ownership
    - Infrastructure is no longer the model (exceptions include private / community clouds)
- Shared Responsibilities:
    - Security and test responsibilities are now shared by programs, CSOs, and third parties (e.g. 3PAO, CASB)
- System architecture
    - Must address security and test equities

Security and Test Considerations for DevSecOps

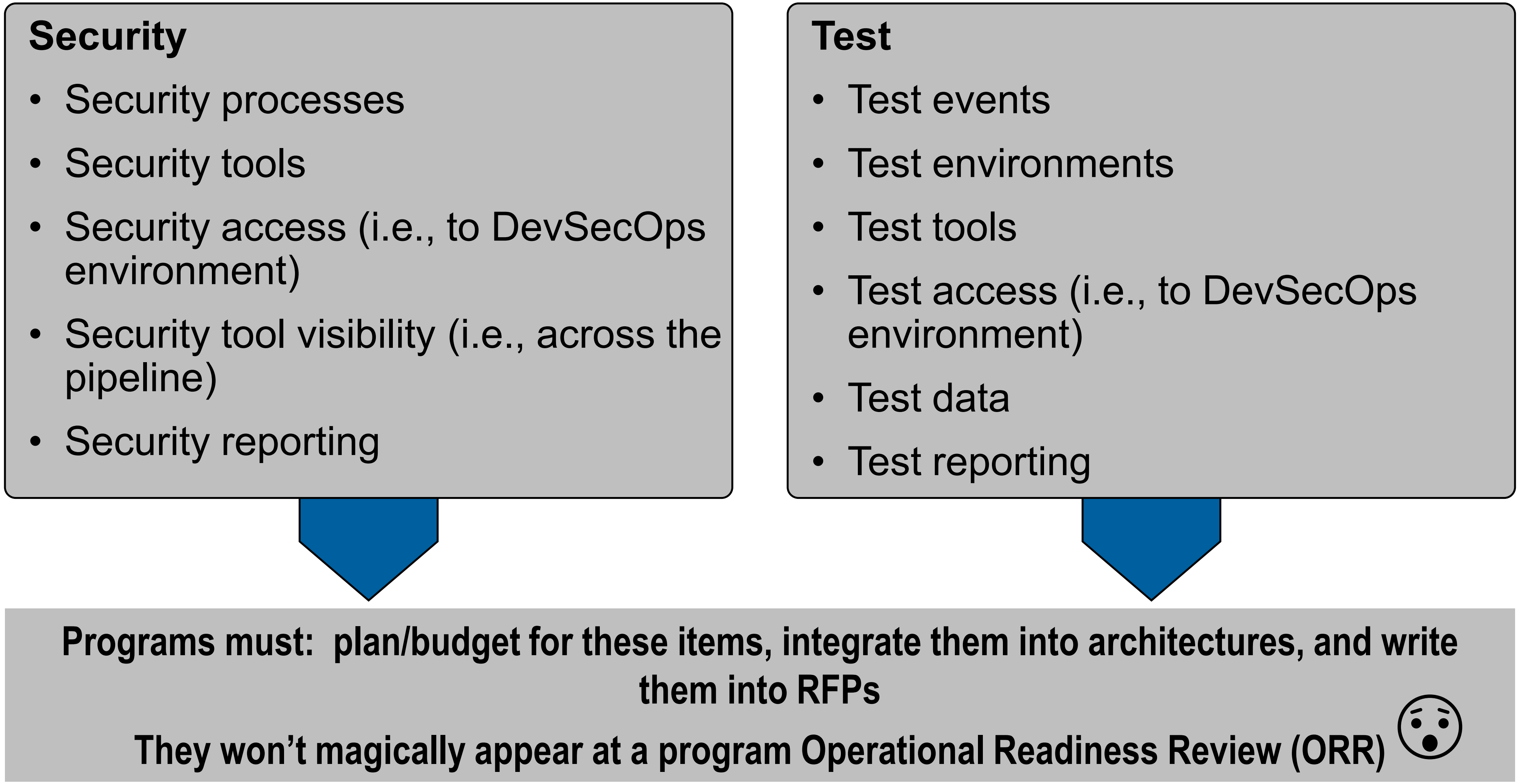
Security Considerations

- **Security should be built into the entire DevSecOps process**
  - Agile process that feeds DevSecOps also must be secure
  - Security user stories must be in the backlog

- **Embed security throughout software lifecycle to identify vulnerabilities earlier, perform faster fixes therefore reduce cost**

- **Continuous monitoring to ensure devices, tools, accounts, etc. are continuously discovered and validated**

Shift Left: Security and Test Considerations



Security

- Security processes
- Security tools
- Security access (i.e., to DevSecOps environment)
- Security tool visibility (i.e., across the pipeline)
- Security reporting

Test

- Test events
- Test environments
- Test tools
- Test access (i.e., to DevSecOps environment)
- Test data
- Test reporting

Programs must: plan/budget for these items, integrate them into architectures, and write them into RFPs

They won't magically appear at a program Operational Readiness Review (ORR) 😯

Test Oversight Influence Areas

Type	Artifacts to Influence	Proactive Measures
Input	Acquisition Strategy, SOW Technical Requirements, Program TEMP	• Acquisition. Develop and communicate the Test Strategy (including security test activities), including: major test events, automation strategy and requirements (e.g., needed tools / standards), required access to Dev/Test environments, plan for test data • Required Test Artifacts. Ensuring the contract(s) mandates test plans, test cases, test reports, traceability matrices, shared with the govt. Formats, ability to comment are important • Testability. Requirement for testability of contractor-derived requirements, testable code including security
Pre-Develop	Architecture, Use Cases, Scenarios, System/Functional Requirements	• Interfaces. Understand and define interfaces, both internal and external systems • Test Environment. To model / influence the test environment to closely mirror production (and development) • Test Data. Identify test data sources and ability to access (or emulate); security use cases
During-Develop	Design Specs, Demos, Test Events, Test Cases	• New Interfaces and Data Sources. What is the developer changing? Understand how the developer is deriving requirements, interfaces, and functionality • Traceability. Do the developer's changes align to the system-level requirements and architecture? • Observe. Automated testing, live test events / demos • Risk Assessment. Is the evolving design going to work? What new risks have been introduced?
Output	Test Reports, Working Software, Data Model(s)	• Review of Test Outputs. Increased visibility to stakeholders of metrics around tests (automated test suite vs manual test time, code coverage, etc.) • Recommendations. How can we reduce risk without killing the benefits of "agility"? TEST AUTOMATION!

Test Event Levels, Challenges, DevSecOps Considerations

Test Level	Conducted By	Overseen By	Focus Area	Challenges (bolded words are important)	DevSecOps
Unit	Contractor	FFRDC / SETA	Code	**Automation** Access and Tools Test **Output** Access Test **Traceability**	• Automate unit tests • Any failed unit tests fail the DevSecOps Pipeline • Development, security, and test work together
Integration	Contractor	Oversight body: DT&E	Interface / API	**Interfaces** / Interface Design Test **Environment** Test **Data** **External** Systems	• User Stories are the "requirements" to be tested • Each User Story should have corresponding automated tests and acceptance criteria, including Security User Stories
System	Mission Owner	Oversight body: DT&E / OT&E	End-to-End Functionality	Test Environment Test Data External Systems	• Automated user functional tests via tools (e.g., UFT, Selenium, OWASP Zap)
Acceptance	Operator	Oversight body: OT&E	End-to-End Operations	**Not slowing everything down!** Timely Validation, Feedback Loop **Feasibility** (what can actually be changed)	• Automated acceptance tests
Release	Contractor	Oversight body: OT&E	Deployment	Successful delivery of working software	• Minimize manual system installation • Treat Infrastructure as Code and use deployment automation

Continuous Delivery Testing – Software Release Approaches

Continuous Delivery Test Techniques	Description
• **Blue Green Deployment**	This requires 2 identical infrastructures to host the application. • Green environment runs the current version of the application. • Blue environment hosts the new version of the software to be tested. • User load is then incrementally shifted from the previously accepted version to the new version. • If there are any issues encountered in the new version, rollback can be done easily to the older accepted version. • This technique increases availability and reduces risk of the application.
• **Canary Releases**	This testing is often automated and includes a limited set of early adopter users. These users assist in identifying issues before the application is released to a wider range of users.
• **A/B Testing**	This method compares two versions of a single webpage or app to determine which one performs better over the other. A/B testing is an experiment in which 2 variants of a page are shown to users randomly and then determine which version performs better.

Additional Test Types – Leveraged as Needed

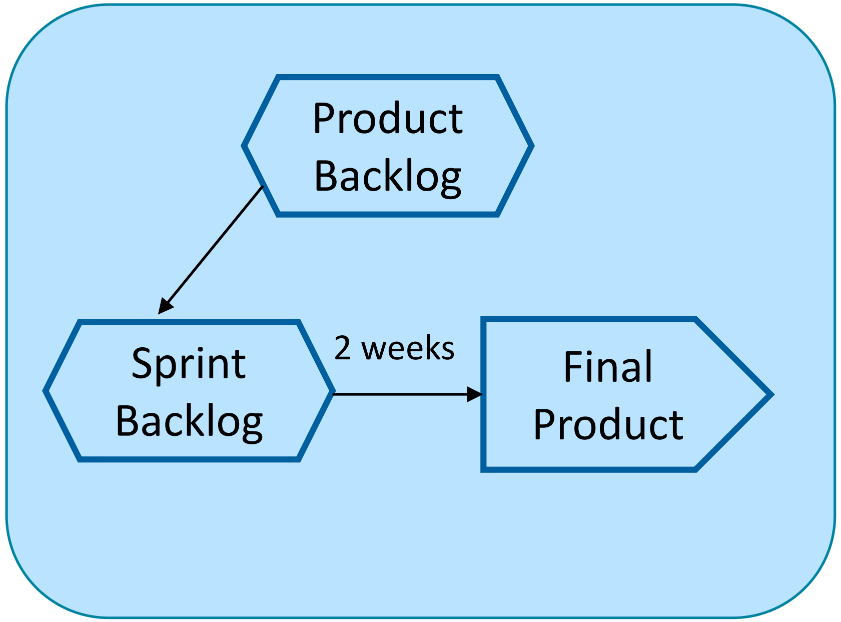
- **Smoke Testing**
- **Functional Testing**
- **Security Testing**
- **Performance Testing**
  - Load Testing
  - Stress Testing
  - Spike Testing
- **Regression Testing**
- **Compliance Testing**

DevSecOps Processes and Technical Considerations

Agile + DevSecOps Pipeline

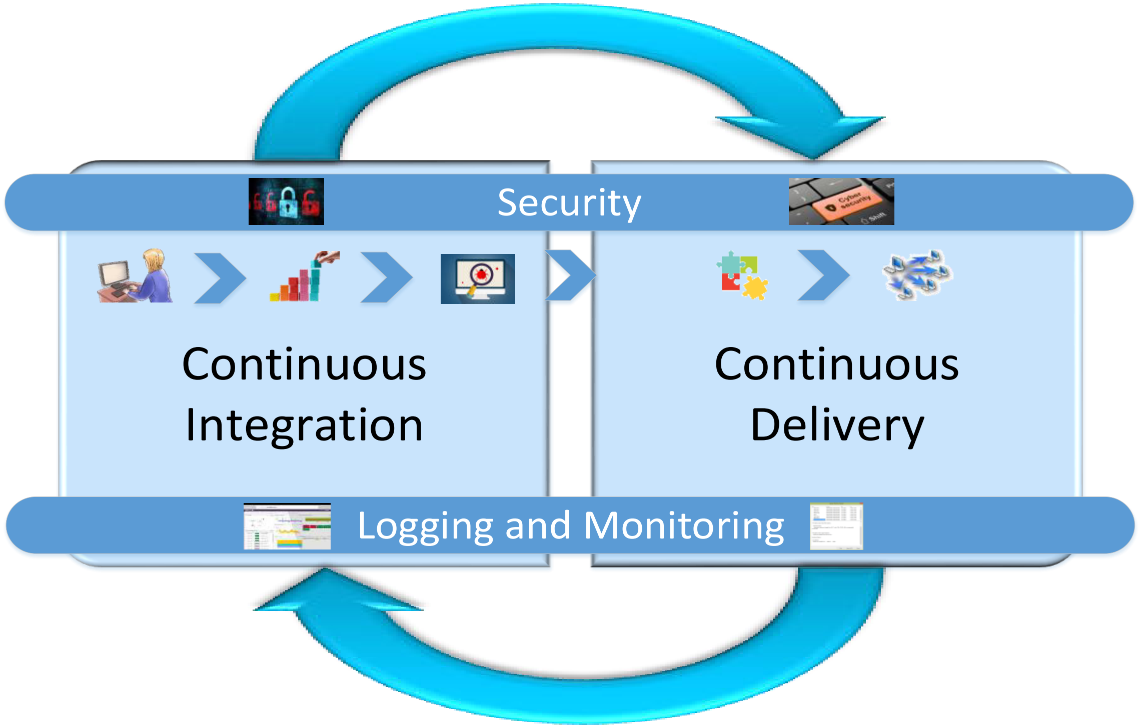
Agile Development

Developing functionality based upon user needs



DevSecOps Pipeline

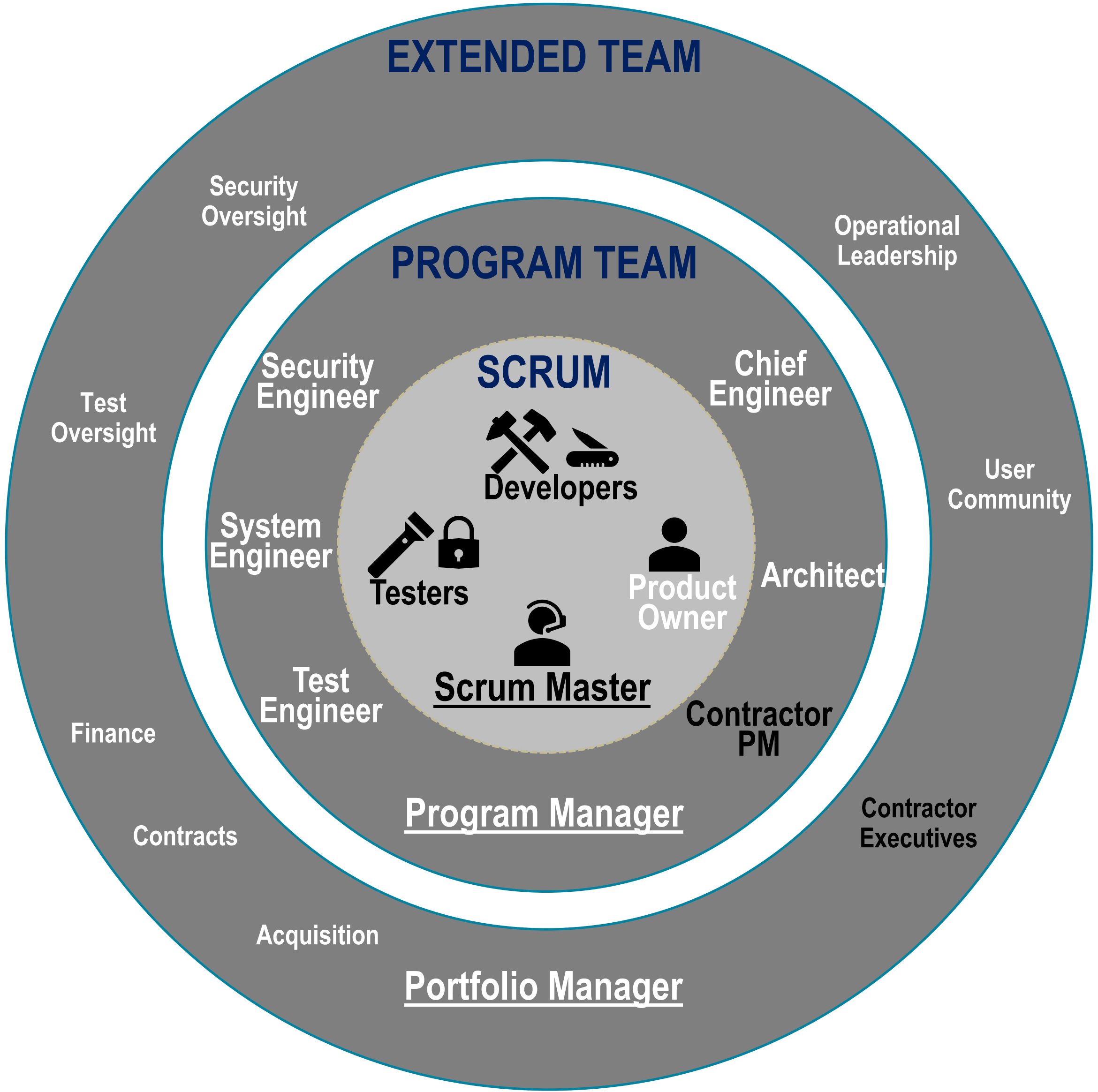
Automating build, secure / test, deploy, and monitor



Security: is part of the entire process, from Backlog through Production Operations

Agile SCRUM – Team Composition

- **Design Decisions**. Many design choices are made by the Agile team. Programs need to ensure that these decisions are:
  - Consistent with the program architecture
  - Compliant with the security approach
  - Testable
- **Variance**. Organizational composition and roles will vary from program to program
- **Multiple Teams (e.g., Scaled Agile Framework (SAFE))**. Most large programs will have multiple Agile development teams contributing to a common architecture.



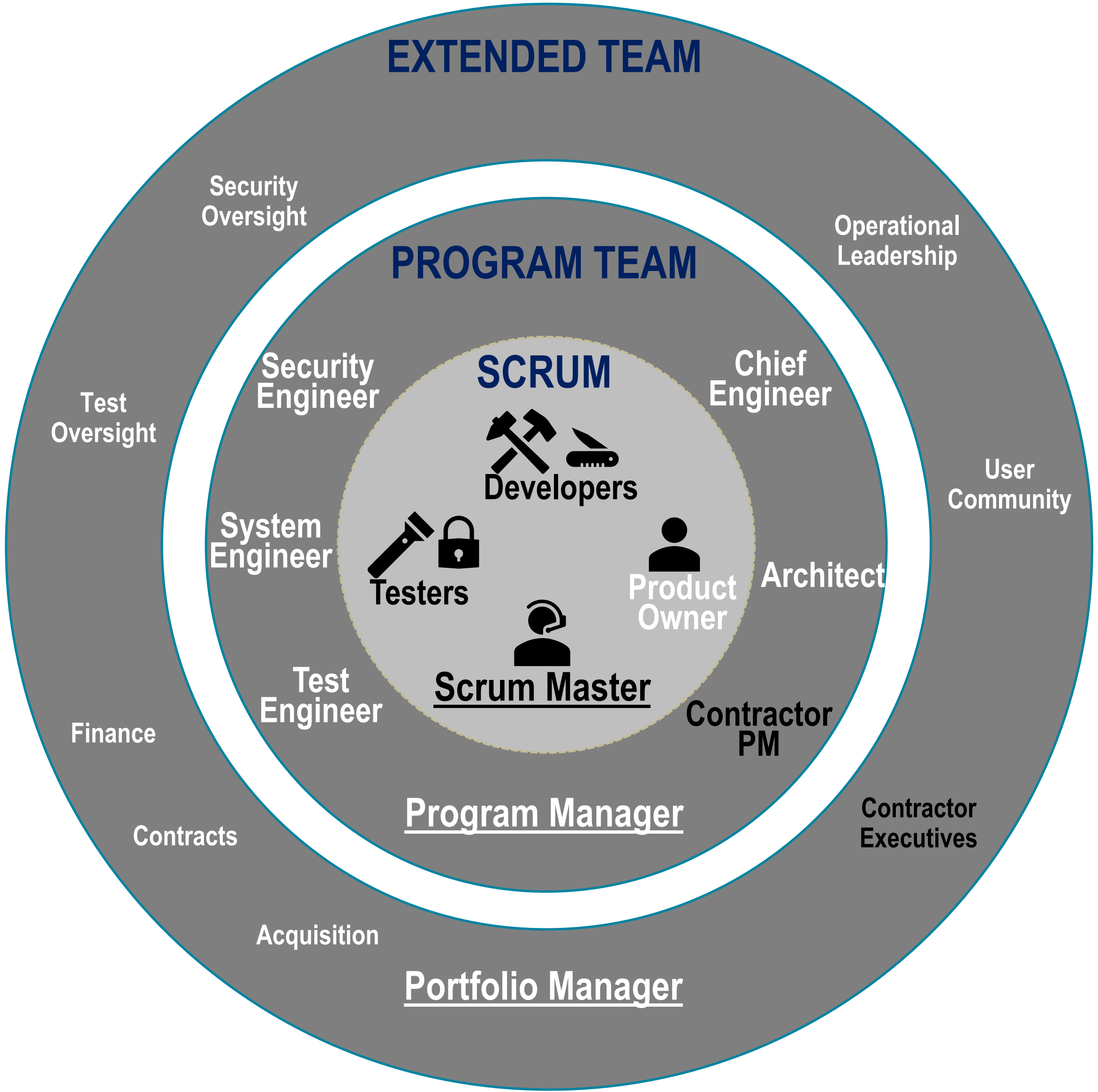
Others Stakeholders:

- Operators
- External System Owners and Operators
- External Government or Commercial Data Providers / Consumers
- IT Administrators
- Cloud Service Providers
- Security: CASBs, MSSPs
- SETA Contractors
- FFRDCs

Legend:
Team Name / Level – **Blue Font**
Government – **White Font**
Contractor – **Black Font**
Leader – **Underline Font**

Agile SCRUM – Test and Security

- **Contractor Testing**. Resources are embedded in the SCRUM. Test coverage includes application functionality and security
- **Government Testing**. Should take place at the end of each sprint, and can be done via a test event or other verification method (e.g., demo, report, etc.). Test coverage should include application functionality and security
- **Testing Environments**. Government can test in either (or both) Test and Pre-Prod environments
- **Tailoring Roles**. Government vs. Contractor roles and responsibilities should be adapted for specific program needs



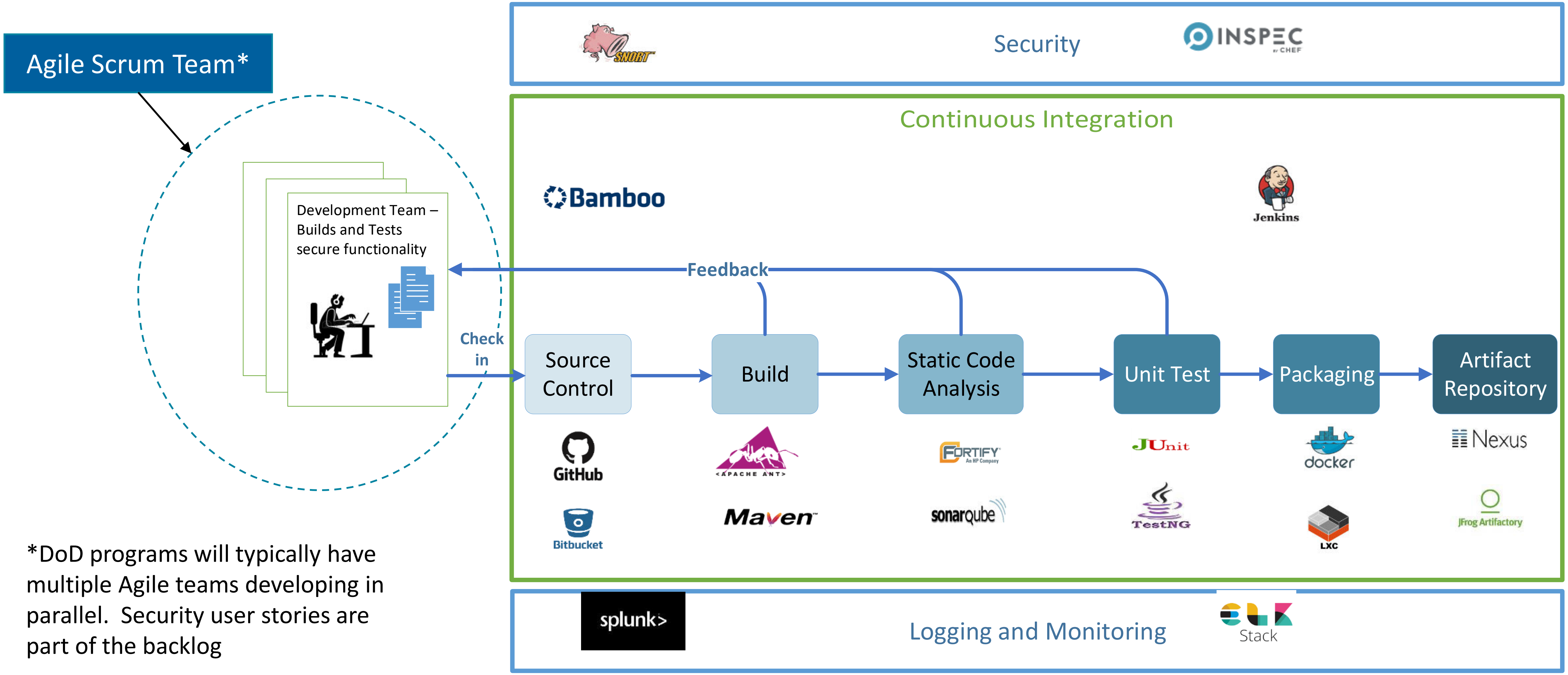
Others Stakeholders:

- Operators
- External System Owners and Operators
- External Government or Commercial Data Providers / Consumers
- IT Administrators
- Cloud Service Providers
- Security: CASBs, MSSPs
- SETA Contractors
- FFRDCs

Legend:
Team Name / Level – **Blue Font**
Government – **White Font**
Contractor – **Black Font**
Leader – **Underline Font**

MITRE

Continuous Integration (CI)



Agile Scrum Team*

Development Team – Builds and Tests secure functionality

Security

Continuous Integration

Feedback

Check in

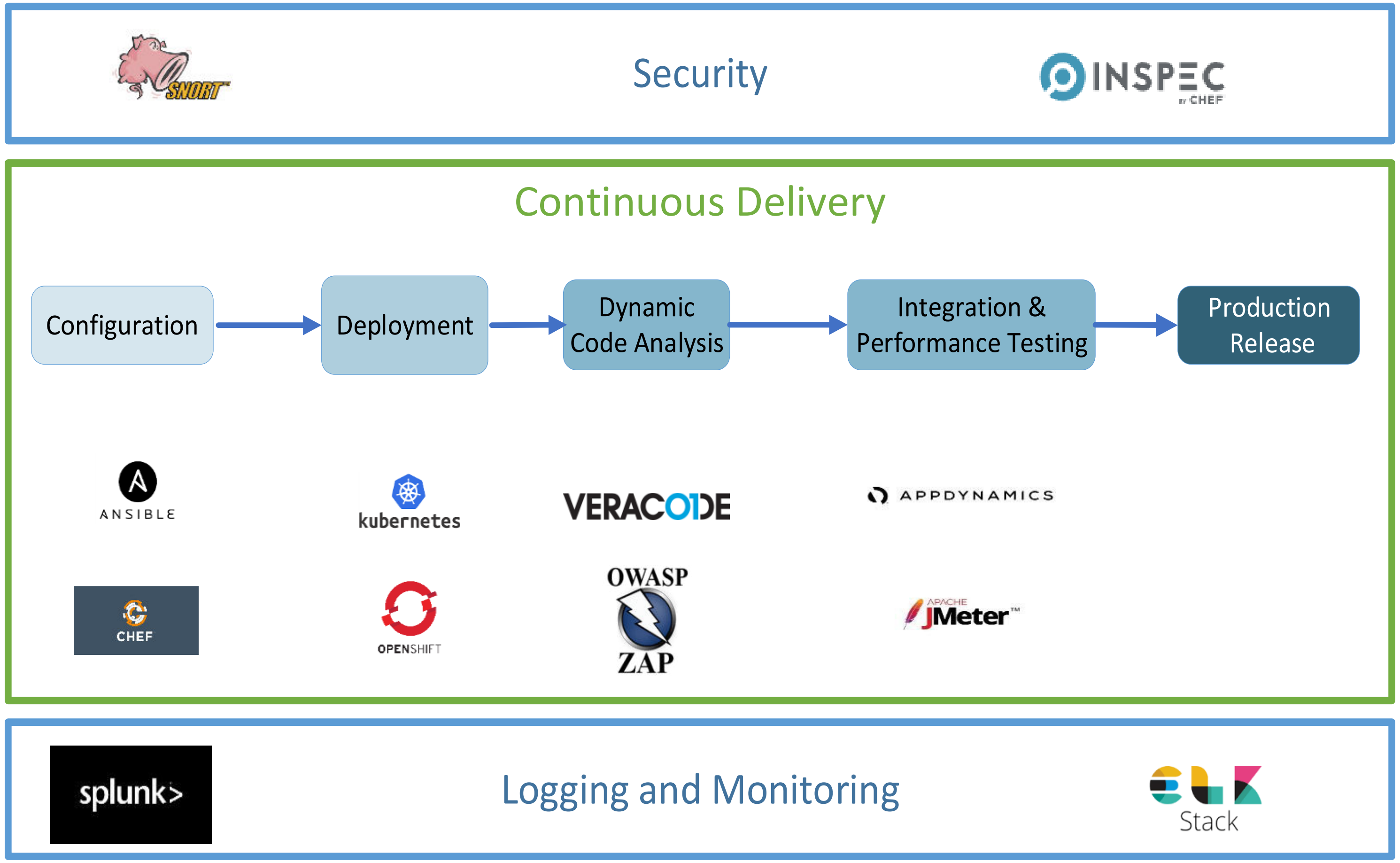
Source Control → Build → Static Code Analysis → Unit Test → Packaging → Artifact Repository

Logging and Monitoring

*DoD programs will typically have multiple Agile teams developing in parallel. Security user stories are part of the backlog

Feedback loop ensures continuous error correction and vulnerability remediation at each stage in the DevSecOps pipeline

Continuous Delivery (CD)



Security

Continuous Delivery

Configuration → Deployment → Dynamic Code Analysis → Integration & Performance Testing → Production Release

Logging and Monitoring

Continuous Delivery promotes the working software from lower environments to higher environments after security and tests are satisfied

DevSecOps Tools - Examples

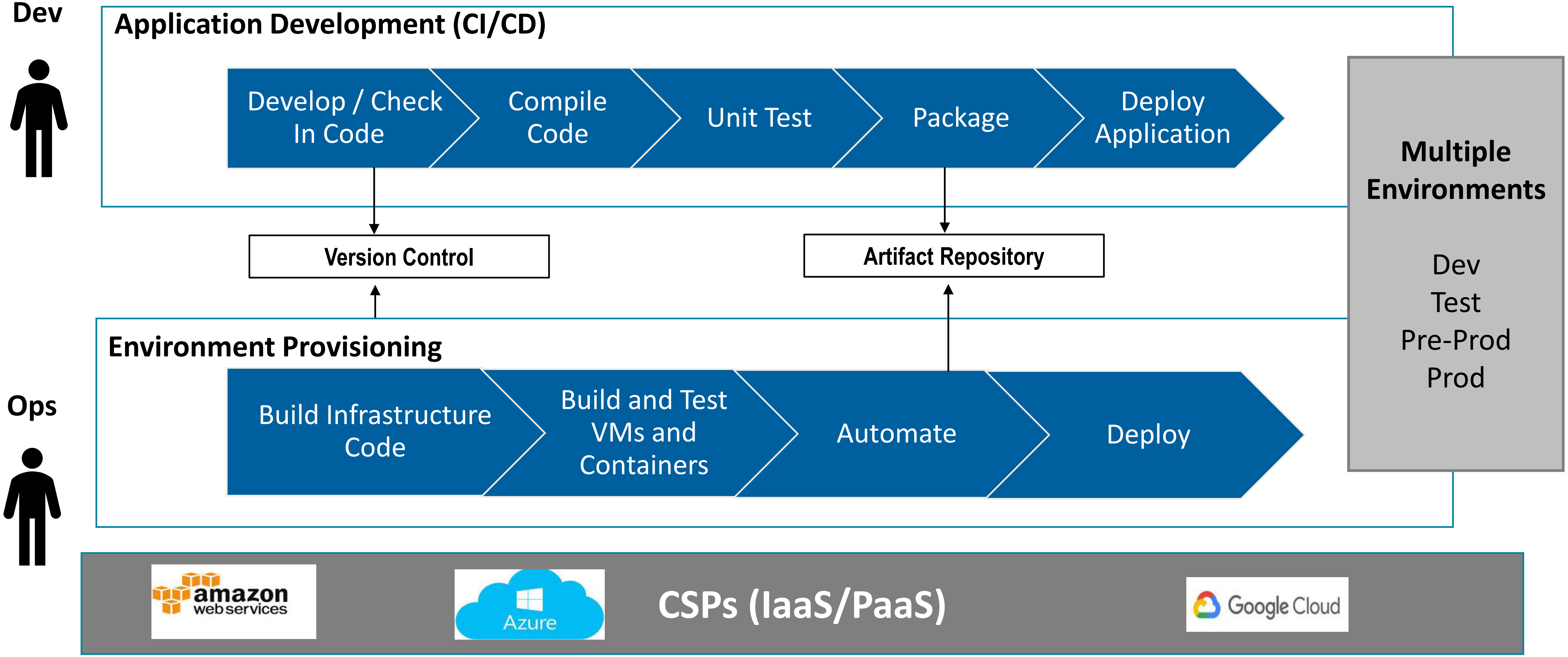
- **Security**
  - Snort, Splunk, Fortify SCA, Vault, OWASP Zap, SonarQube
- **Source Control**
  - GitHub, GitLab, Bitbucket, Artifactory
- **Continuous Integration Tools**
  - Jenkins, Bamboo
- **Testing Tools**
  - JUnit, Selenium, JMeter, TestNG, SoapUI
- **Config/Provisioning Tools**
  - Ansible, Chef, Puppet
- **Logging and Monitoring Tools**
  - ELK (Elasticsearch, Logstash & Kibana) Stack, Splunk
- **Release Orchestration**
  - Kubernetes, OpenShift
- **Containers**
  - Docker, Docker Swarm

Security: Tools used throughout the process, regardless of the specific tools being used

Example: Snort signatures are applied to all flows that are visible

Example: Splunk collects and aggregates all logs that are available throughout the process

CI-CD on Cloud


Dev

Application Development (CI/CD)

Develop / Check In Code

Compile Code

Unit Test

Package

Deploy Application

Version Control

Artifact Repository

Ops

Environment Provisioning

Build Infrastructure Code

Build and Test VMs and Containers

Automate

Deploy

Multiple Environments

Dev

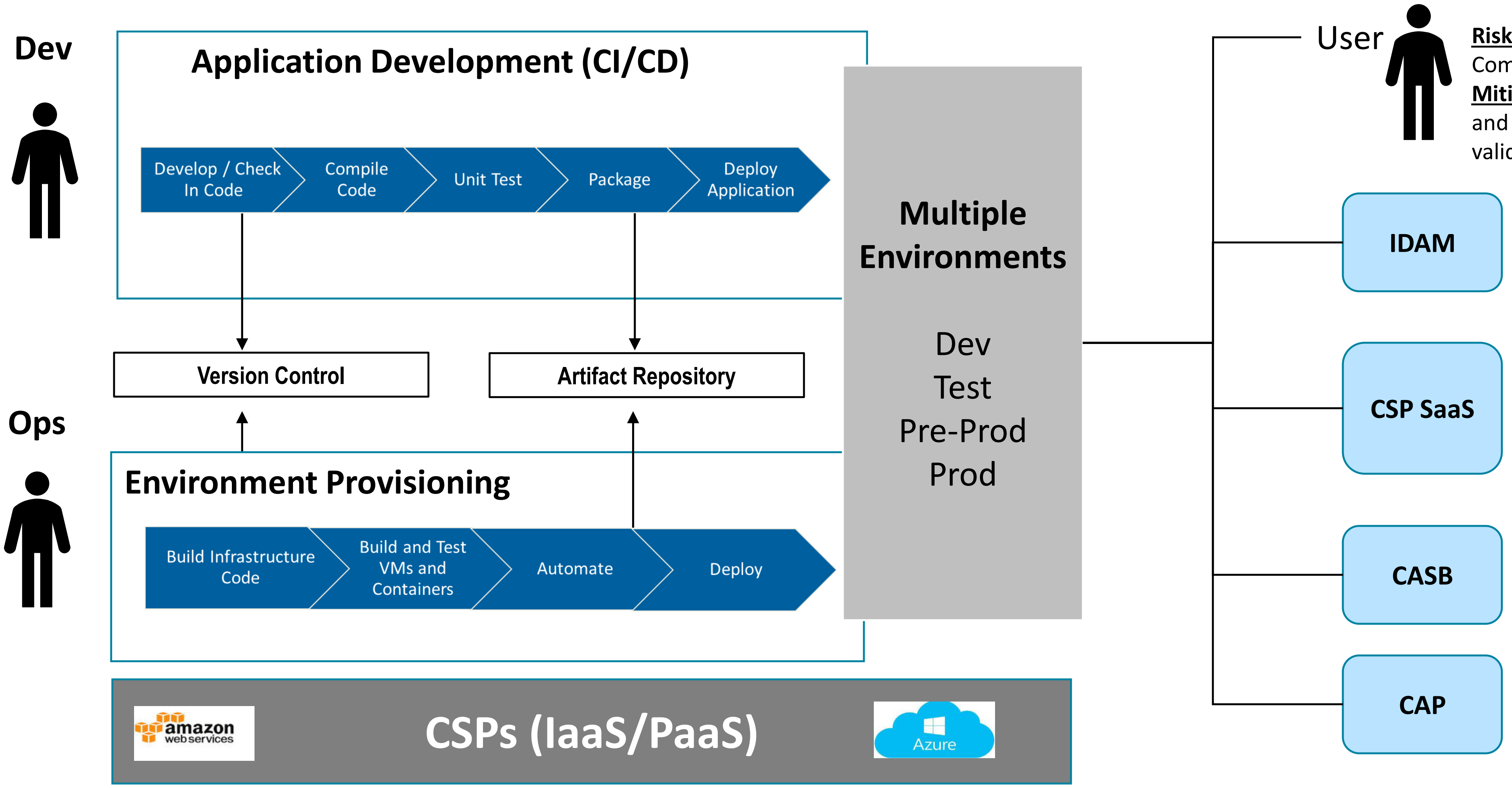
Test

Pre-Prod

Prod

CSPs (IaaS/PaaS)

CI-CD System in Operations



Dev

Application Development (CI/CD)

Develop / Check In Code → Compile Code → Unit Test → Package → Deploy Application

Version Control

Artifact Repository

Ops

Environment Provisioning

Build Infrastructure Code → Build and Test VMs and Containers → Automate → Deploy

CSPs (IaaS/PaaS)

amazon web services

Azure

Multiple Environments

Dev
Test
Pre-Prod
Prod

User

Risk: User Identity Compromised
Mitigation: Ensure all accounts and devices are continuously validated

IDAM

Risk: Privilege Accounts compromised
Mitigation: Deploy identity governance and PAM tools

CSP SaaS

Risk: Misconfiguration of SaaS
Mitigation: Perform scans to identify and fix potential misconfigurations and identify shadow IT

CASB

Risk: API is not secure
Mitigation: API security testing and continuous monitoring

CAP

Risk: CAP becomes bottleneck
Mitigation: Direct connect to CSP (policy dependent), or use VPN with split tunnel + whitelisted IPs

Security in DevSecOps

- **Embed security throughout software lifecycle to identify vulnerabilities earlier, perform faster fixes therefore reduce cost.**
- **Different aspects of DevSecOps security in the software lifecycle including tools**
  - Static Code Analysis – Scans for vulnerabilities in the code after coding but before unit testing during development (e.g. SonarQube)
  - Configuration Management and Compliance – Know how your application is configured and whether it follows your policies (e.g., Ansible, Chef, Puppet)
  - Dynamic Code Analysis – Scan your code for vulnerabilities in how it performs. Execute unit tests to find errors (e.g., SonarLint, VeraCode)
  - Vulnerability Scanning – Automatically identify known issues in your application for penetration testing (e.g., Nessus)
  - Infrastructure as Code – Ensures the application is deployed securely and without errors in a repeatable manner (e.g., Ansible)
  - Continuous Monitoring – Information on how the application is running, collected and monitored to identify issues and feed future improvements. This is done in production environment. (e.g. Splunk, AppDynamics)
  - Container Security – monitor and protect containers (e.g., BlackDuck)

DevSecOps Security Tools – Examples

Security Tool	Description	Focus Area	Test Oversight	Open Source
Snort	It is a Network intrusion detection and prevention system. Scrutinizes each packet on the network for anomalies and monitors traffic real time.	IDS	OT&E	Yes
Fortify SCA	Static code analyzer helps to identify security vulnerabilities efficiently in source code during development.	Code Security	DT&E	No
Gauntlt	Gauntlt provides hooks to a variety of security tools and puts them within reach of security, dev and ops teams to collaborate to build rugged software.	Security Test Automation	DT&E	Yes
HashiCorp Vault	Improves how software teams store important keys, tokens, passwords, and other secrets in their projects. Vault is an environment- and infrastructure-agnostic open toolset for secrets management.	Credential Protection	DT&E	Yes
Sonar Qube	Continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities.	Code Security	DT&E	Yes
OWASP Zap	Used to identify security vulnerabilities in an application while it is being developed. Useful in penetration testing.	Vulnerability Scanning	DT&E and OT&E	Yes

DevSecOps Testing Tools – Examples

Testing Tool	Description	DT&E Applicability	URL	Focus Area
JUnit	Open source, automated unit test framework for Java programming language	Applicable for DT and OT	http://junit.org	Unit Testing
Selenium	Suite of tools to automate web application testing across many platforms. Supported by many popular browsers such as Firefox, Chrome. Robot framework built on top of Selenium enables continuous testing.	Applicable for DT and OT	http://docs.seleniumhq.org	Unit, System, Integration Testing
SoapUI	Open-source web service testing application framework for SOAP and REST APIs	Applicable for DT and OT	https://www.soapui.org	Unit, Functional and Integration Testing
Rational Functional Tester	It is capable of Functional, API, Performance Testing and Regression testing.	Applicable for DT and OT	https://www.ibm.com/us-en/marketplace/rational-functional-tester	Functional Testing
JMeter	Load testing tool for analyzing and measuring performance of services, with a focus on web applications	Applicable for DT and OT	http://jmeter.apache.org/	Performance (Load) Testing
TestNG	Testing framework to cover all categories of tests: unit, functional, end-to-end, integration etc.	Applicable for DT and OT	http://testng.org/doc/index.html	Unit and Integration Testing
Unified Functional Test (UFT)	Automates functional and regression testing for applications and environments.	OT only	https://www.microfocus.com/en-us/products/unified-functional-automated-testing/overview	System Testing

Cloud Native (AWS, Azure) DevSecOps Testing and Security Tools

- **DevSecOps Pipeline**
  - AWS CodePipeline
  - Azure DevOps
- **Infrastructure Provisioning**
  - AWS Cloud Formation
  - Azure Automation, Azure Resource Manager
- **Security**
  - AWS Inspector, AWS GuardDuty, AWS CloudWatch
  - Azure Security Center, Azure AD, Azure Application Insights

Platform Deployment Options: Containerization verses Virtualization

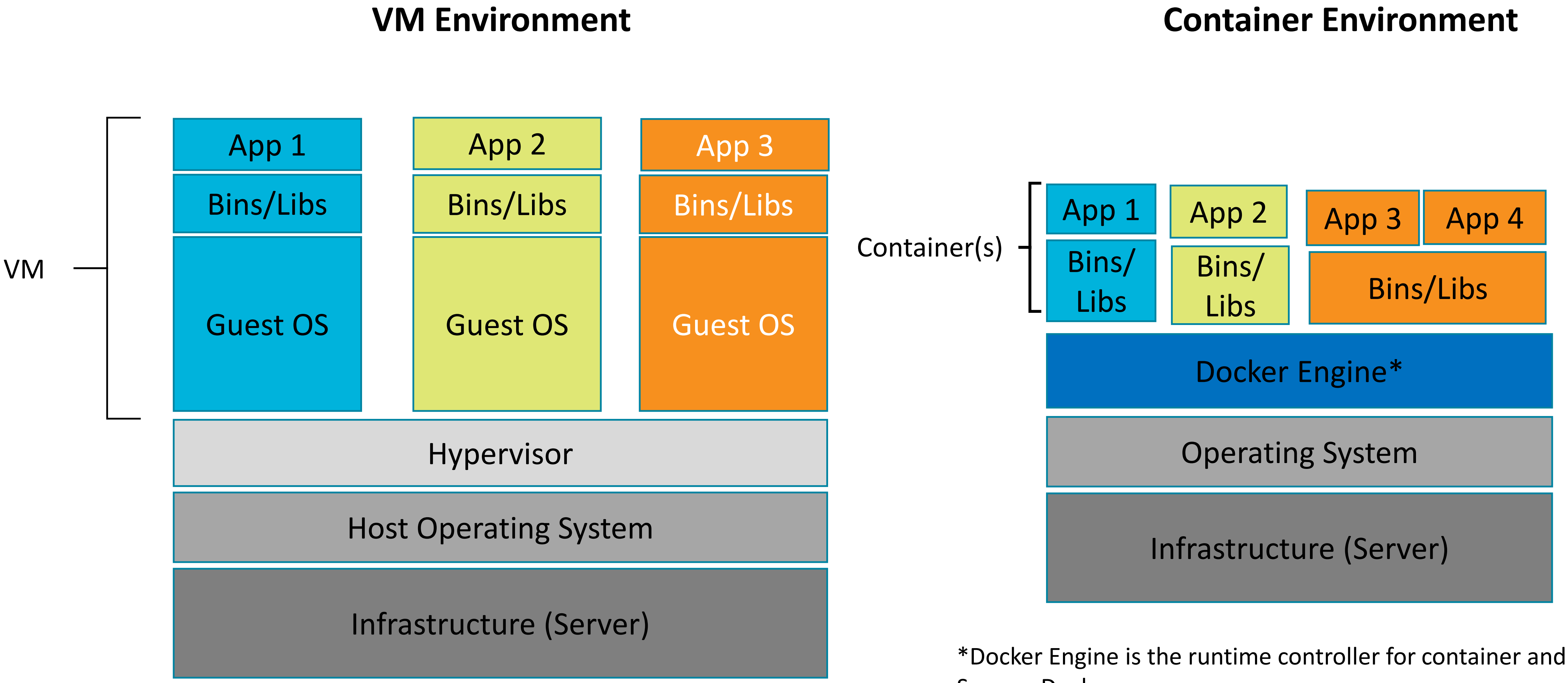
Containerization vs. Virtualization

- **In DevSecOps, software applications can be deployed in Containers or Virtual Machines (VMs)**

- **VMs**
  - Self-contained computing unit with host operating system (OS)
  - Each application runs dedicated software binaries/ libraries (bins/libs) and a guest OS
  - Managed by a hypervisor

- **Containers**
  - All applications share the OS and software bins/libs
  - Containers are managed by a controller. Example: Docker Daemon (which sits in a sibling container)

Containers vs. VMs

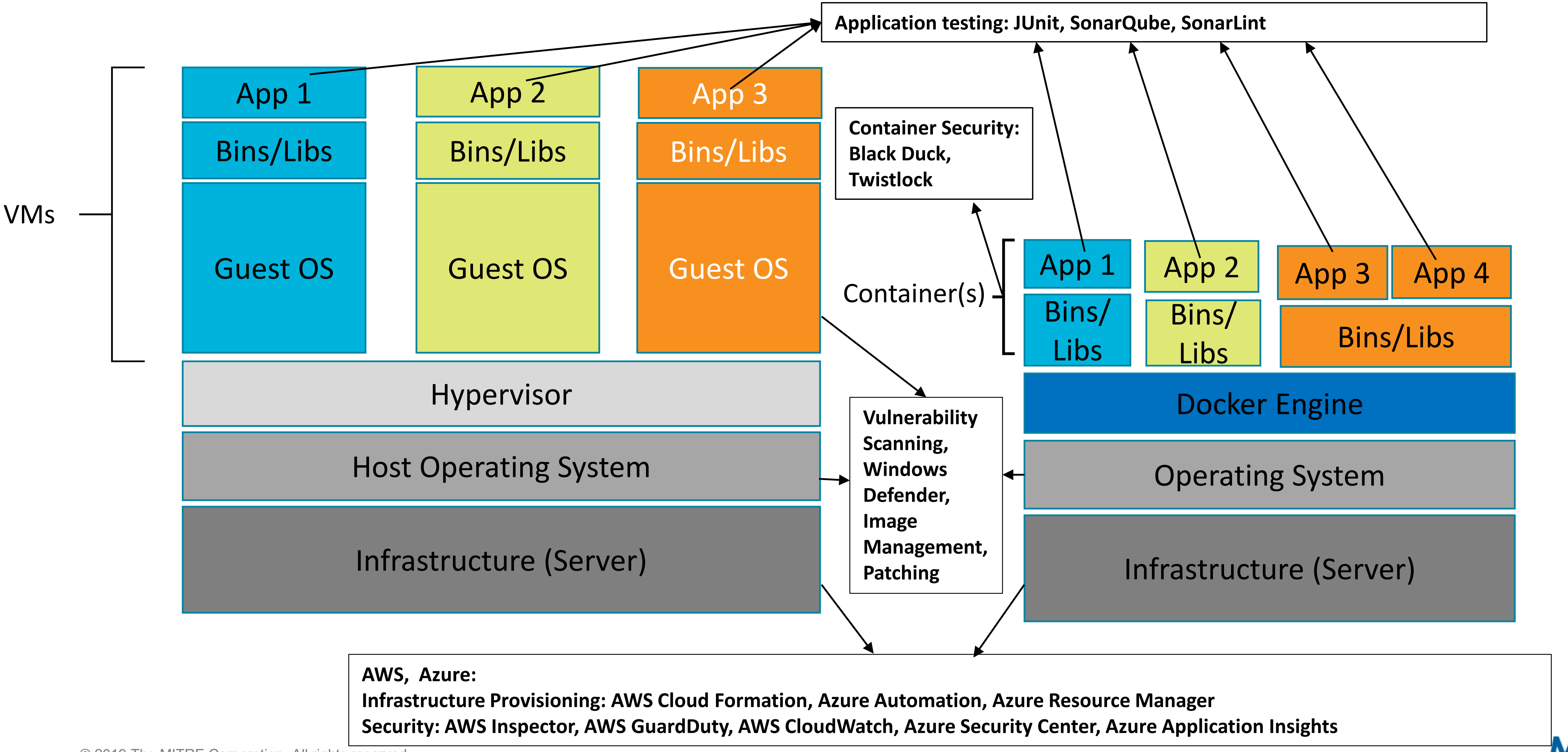


*Docker Engine is the runtime controller for container and images
Source: Docker.com

Containers vs. VMs Comparison Criteria

Criteria	Virtual Machines	Containers
Popular Examples	VMWare vSphere, Microsoft Hyper-V	Docker, Google Kubernetes, Red Hat OpenShift
Hosting Environment	On or Off Premise cloud environments	On or off premises cloud environments
Runtime Environment	Full OS with dedicated resources; one or more microservices per VM	Shared OS, resources per container; single microservice per container
Portability	Microservice portability is tied to the portability of selected VMs	Microservices are decoupled from the OS, allowing greater portability
Security	VM security tools and procedures are more mature	Larger number of services and interfaces to monitor and protect
Scalability	VMs can be automatically scaled based on demand	Containers can be automatically scaled based on demand
Performance	Dedicated resources in a VM mean more overhead	Better performance than VMs due to smaller footprint than VMs
Admin Burden	Less time/effort to spin up and configure vs. physical machines. However, more time to spin up than containers	Simpler packaging and deployment vs. VMs
Interoperability	VMs with separate OSs may complicate cross-service communications, plug-and-play interoperability	Single-OS microservice deployments are more interoperable
Agility	Requires some degree of planning and coordination	Single-function containers can support faster development lifecycle
Market Trend	Still popular but losing ground to container deployment	Increasingly popular option for app migrations and microservice deployment

Containers vs. VMs – Security Examples



Application testing: JUnit, SonarQube, SonarLint

Container Security: Black Duck, Twistlock

Vulnerability Scanning, Windows Defender, Image Management, Patching

AWS, Azure:
Infrastructure Provisioning: AWS Cloud Formation, Azure Automation, Azure Resource Manager
Security: AWS Inspector, AWS GuardDuty, AWS CloudWatch, Azure Security Center, Azure Application Insights

Conclusions

- **DevSecOps Value Proposition**
  - Programs can realize significant value by implementing DevSecOps. But, test and security should not be sacrificed.
- **Shift Left**
  - Programs must truly shift Security and Test to the left to realize time and cost savings
- **Agile and DevSecOps go together**
  - DevSecOps must be fed by Agile software development. Security user stories must be part of each sprint.
- **Automation is key**
  - Security and test automation can reduce delivery time, improve quality and security, and eliminate human error

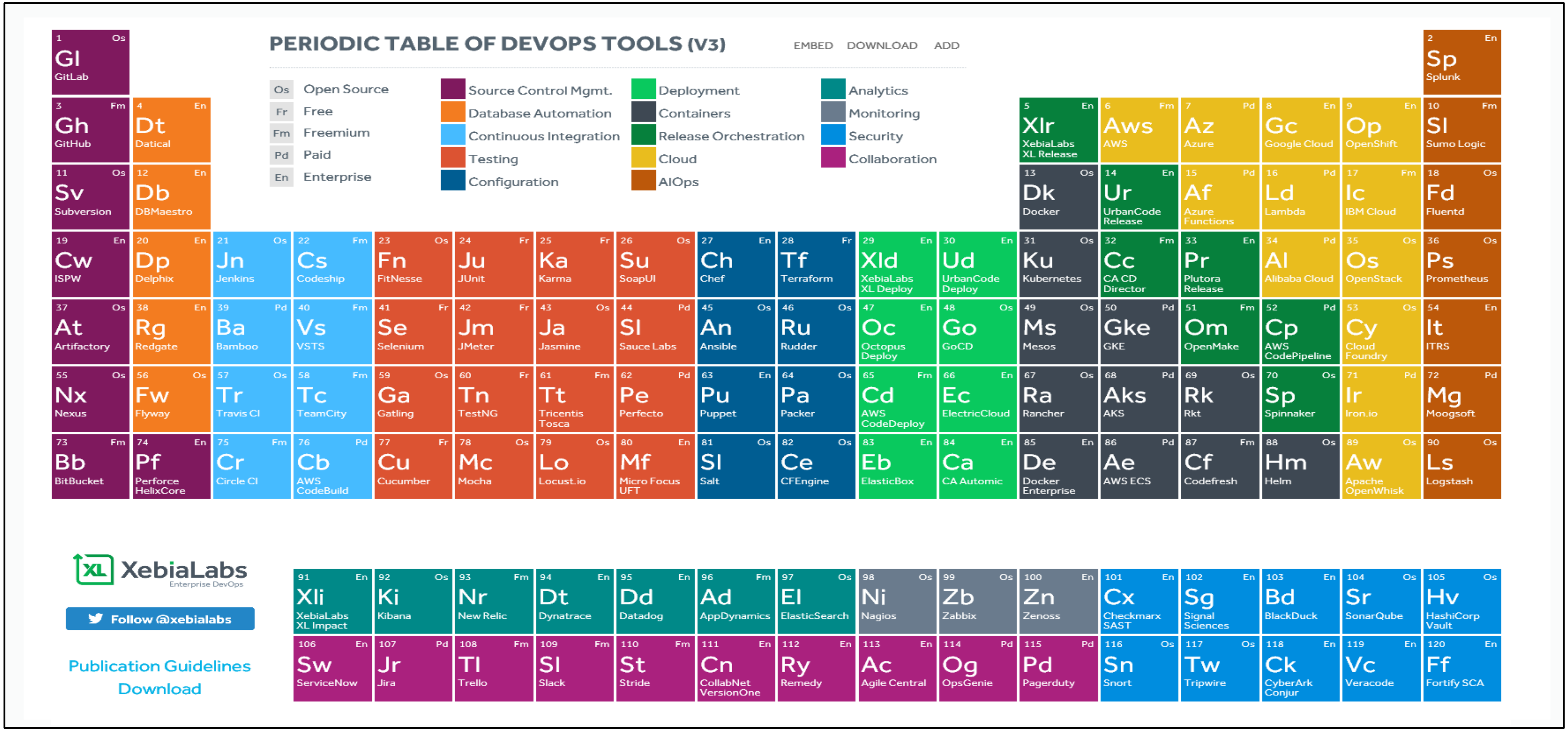
Appendix

Acronyms

Acronym	Description
3PAO	Third Party Assessment Organization
API	Application Programming Interface
AWS	Amazon Web Services
CAP	Cloud Access Point
CASB	Cloud Access Security Broker
CSO	Cloud Service Offering
CSP	Cloud Service Provider
DT	Development Test
DT&E	Developmental, Test and Evaluation
FFRDC	Federally Funded Research and Development Center
IaaS	Infrastructure as a Service

Acronym	Description
MSSP	Managed Security Service Provider
ORR	Operational Readiness Review
OT	Operational Test
OT&E	Operational Test and Evaluation
PaaS	Platform as a Service
PAM	Privileged Access Management
RFP	Request for Proposal
SAFE	Scaled Agile Framework
SETA	Systems Engineering and Technical Assistance
VPN	Virtual Private Network

Periodic Table of DevSecOps Tools



PERIODIC TABLE OF DEVOPS TOOLS (V3)

EMBED DOWNLOAD ADD

Os	Open Source	Source Control Mgmt.	Deployment	Analytics
Fr	Free	Database Automation	Containers	Monitoring
Fm	Freemium	Continuous Integration	Release Orchestration	Security
Pd	Paid	Testing	Cloud	Collaboration
En	Enterprise	Configuration	AIOps	

1 Os Gl GitLab																	2 En Sp Splunk
3 Fm Gh GitHub	4 En Dt Datical											5 En Xlr XebiaLabs XL Release	6 Fm Aws AWS	7 Pd Az Azure	8 En Gc Google Cloud	9 En Op OpenShift	10 Fm Sl Sumo Logic
11 Os Sv Subversion	12 En Db DBMaestro											13 Os Dk Docker	14 En Ur UrbanCode Release	15 Pd Af Azure Functions	16 Pd Ld Lambda	17 Fm Ic IBM Cloud	18 Os Fd Fluentd
19 En Cw ISPW	20 En Dp Delphix	21 Os Jn Jenkins	22 Fm Cs Codeship	23 Os Fn FitNesse	24 Fr Ju JUnit	25 Fr Ka Karma	26 Os Su SoapUI	27 En Ch Chef	28 Fr Tf Terraform	29 En Xld XebiaLabs XL Deploy	30 En Ud UrbanCode Deploy	31 Os Ku Kubernetes	32 Fm Cc CA CD Director	33 En Pr Plutora Release	34 Pd Al Alibaba Cloud	35 Os Os OpenStack	36 Os Ps Prometheus
37 Os At Artifactory	38 En Rg Redgate	39 Pd Ba Bamboo	40 Fm Vs VSTS	41 Fr Se Selenium	42 Fr Jm JMeter	43 Os Ja Jasmine	44 Pd Sl Sauce Labs	45 Os An Ansible	46 Os Ru Rudder	47 En Oc Octopus Deploy	48 Os Go GoCD	49 Os Ms Mesos	50 Pd Gke GKE	51 Fm Om OpenMake	52 Pd Cp AWS CodePipeline	53 Os Cy Cloud Foundry	54 En It ITRS
55 Os Nx Nexus	56 Os Fw Flyway	57 Os Tr Travis CI	58 Fm Tc TeamCity	59 Os Ga Gatling	60 Fr Tn TestNG	61 Fm Tt Tricentis Tosca	62 Pd Pe Perfecto	63 En Pu Puppet	64 Os Pa Packer	65 Fm Cd AWS CodeDeploy	66 En Ec ElectricCloud	67 Os Ra Rancher	68 Pd Aks AKS	69 Os Rk Rkt	70 Os Sp Spinnaker	71 Pd Ir Iron.io	72 Pd Mg Moogsoft
73 Fm Bb BitBucket	74 En Pf Perforce HelixCore	75 Fm Cr Circle CI	76 Pd Cb AWS CodeBuild	77 Fr Cu Cucumber	78 Os Mc Mocha	79 Os Lo Locust.io	80 En Mf Micro Focus UFT	81 Os Sl Salt	82 Os Ce CFEngine	83 En Eb ElasticBox	84 En Ca CA Automic	85 En De Docker Enterprise	86 Pd Ae AWS ECS	87 Fm Cf Codefresh	88 Os Hm Helm	89 Os Aw Apache OpenWhisk	90 Os Ls Logstash
			91 En Xli XebiaLabs XL Impact	92 Os Ki Kibana	93 Fm Nr New Relic	94 En Dt Dynatrace	95 En Dd Datadog	96 Fm Ad AppDynamics	97 Os El ElasticSearch	98 Os Ni Nagios	99 Os Zb Zabbix	100 En Zn Zenoss	101 En Cx Checkmarx SAST	102 En Sg Signal Sciences	103 En Bd BlackDuck	104 Os Sr SonarQube	105 Os Hv HashiCorp Vault
			106 En Sw ServiceNow	107 Pd Jr Jira	108 Fm Tl Trello	109 Fm Sl Slack	110 Fm St Stride	111 En Cn CollabNet VersionOne	112 En Ry Remedy	113 En Ac Agile Central	114 Pd Og OpsGenie	115 Pd Pd Pagerduty	116 Os Sn Snort	117 Os Tw Tripwire	118 En Ck CyberArk Conjur	119 En Vc Veracode	120 En Ff Fortify SCA

XebiaLabs
Enterprise DevOps

Follow @xebialabs

Publication Guidelines
Download

References

- https://www.mitre.org/sites/default/files/publications/MITRE-Defense-Agile-Acquisition-Guide.pdf
- https://xebialabs.com/
- https://www.docker.com/resources/what-container

MITRE

MITRE is a not-for-profit organization whose sole focus is to operate federally funded research and development centers, or FFRDCs. Independent and objective, we take on some of our nation's—and the world's—most critical challenges and provide innovative, practical solutions.

Learn and share more about MITRE, FFRDCs,
and our unique value at www.mitre.org