

DevSecOps

What Why and How?

Agenda

- **What is DevSecOps?**
- **Why do we need DevSecOps?**
- **How do we do DevSecOps?**
- **Integrate Security in DevOps Pipeline**
- **Tools of Trade**
- **Sample Implementation (On Prem and Cloud Native)**
- **Case Studies**

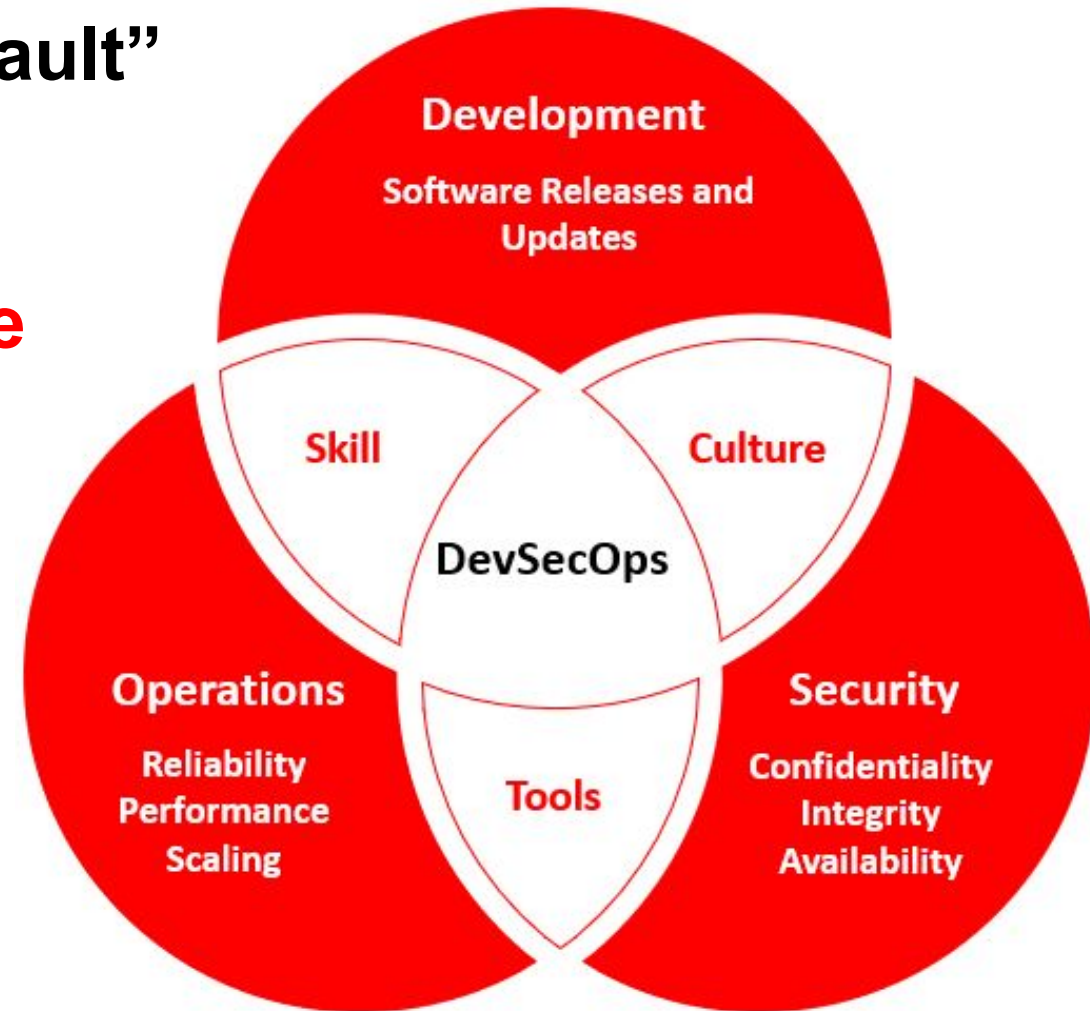
Disclaimer

- **I will be listing a lot of tools, It's not an exhaustive list**
- **I don't endorse or recommend any specific tool / vendor**
- **Every environment is different: Test and validate before implementing any ideas**

What is DevSecOps?

Effort to strive for “Secure by Default”

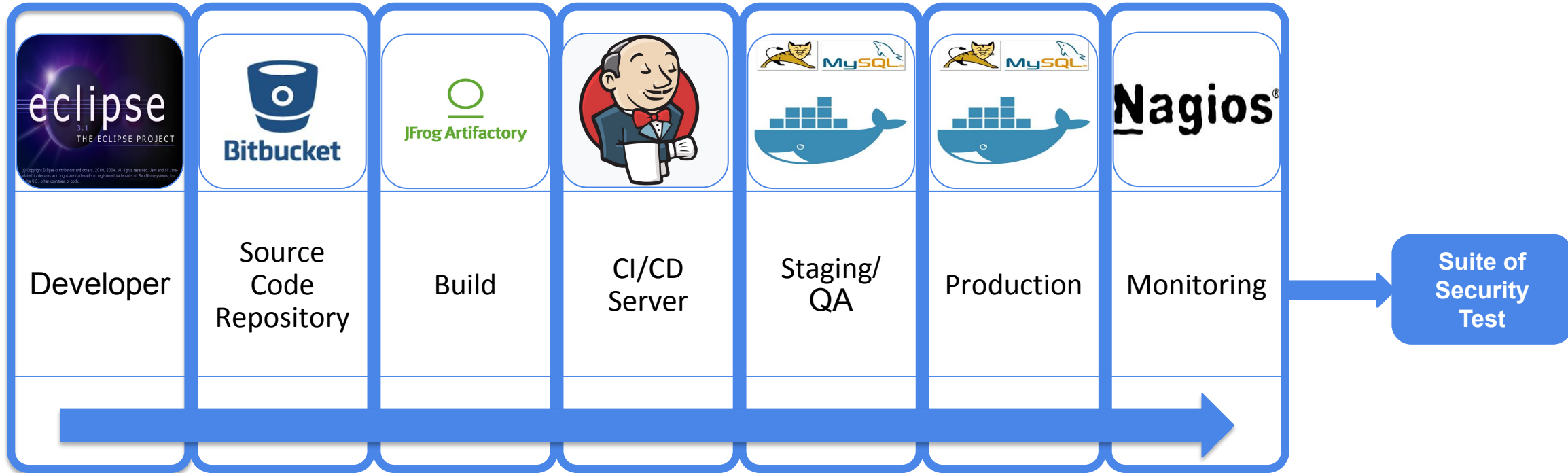
- Integrate Security via **tools**
- Create Security as Code **culture**
- Promote cross **skilling**



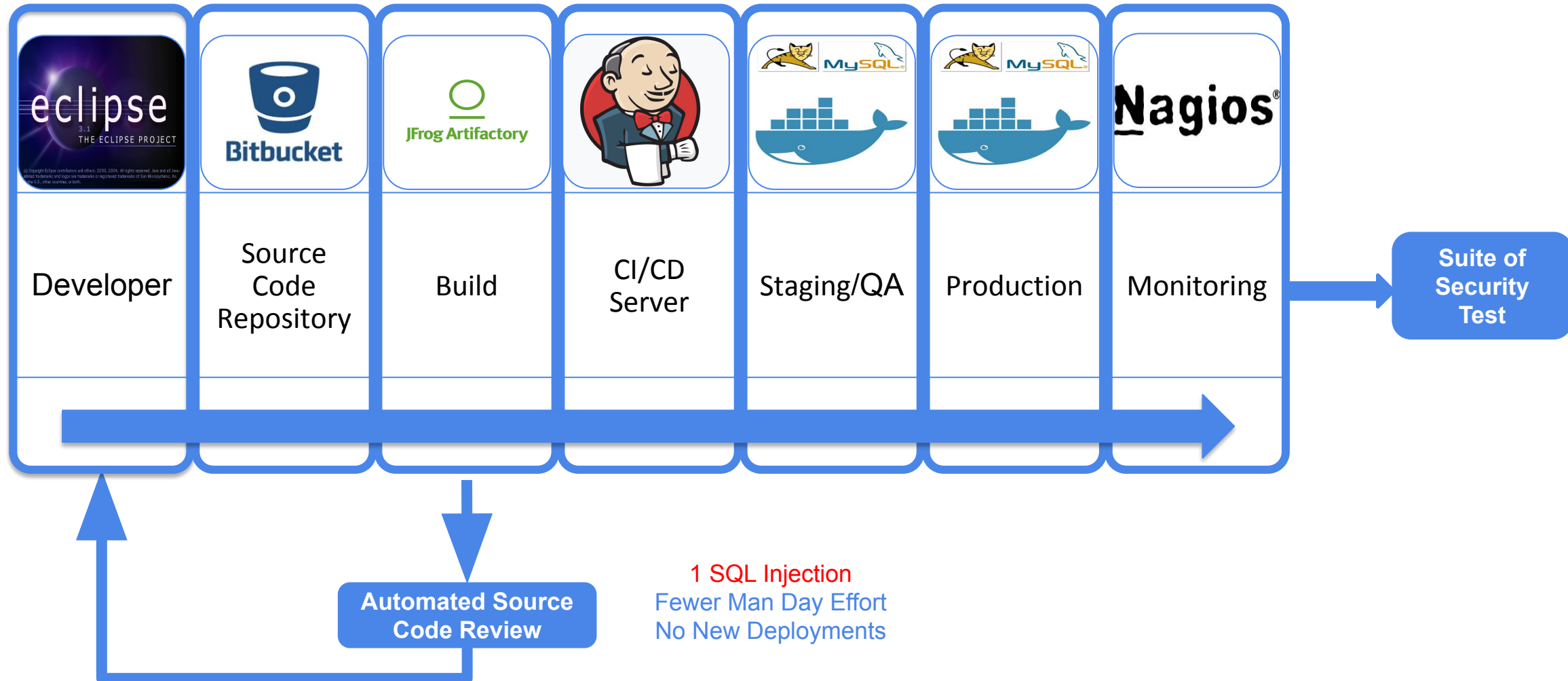
Why do we need DevSecOps?

- **DevOps moves at rapid pace, traditional security just can't keep up**
- **DevSecOps makes it easier to manage rapid pace of development & large scale secure deployments**
- **DevSecOps allows for much smoother scaling of process**
- **Security as part of process is the only way to ensure safety**

Shifting Left saves cost & time



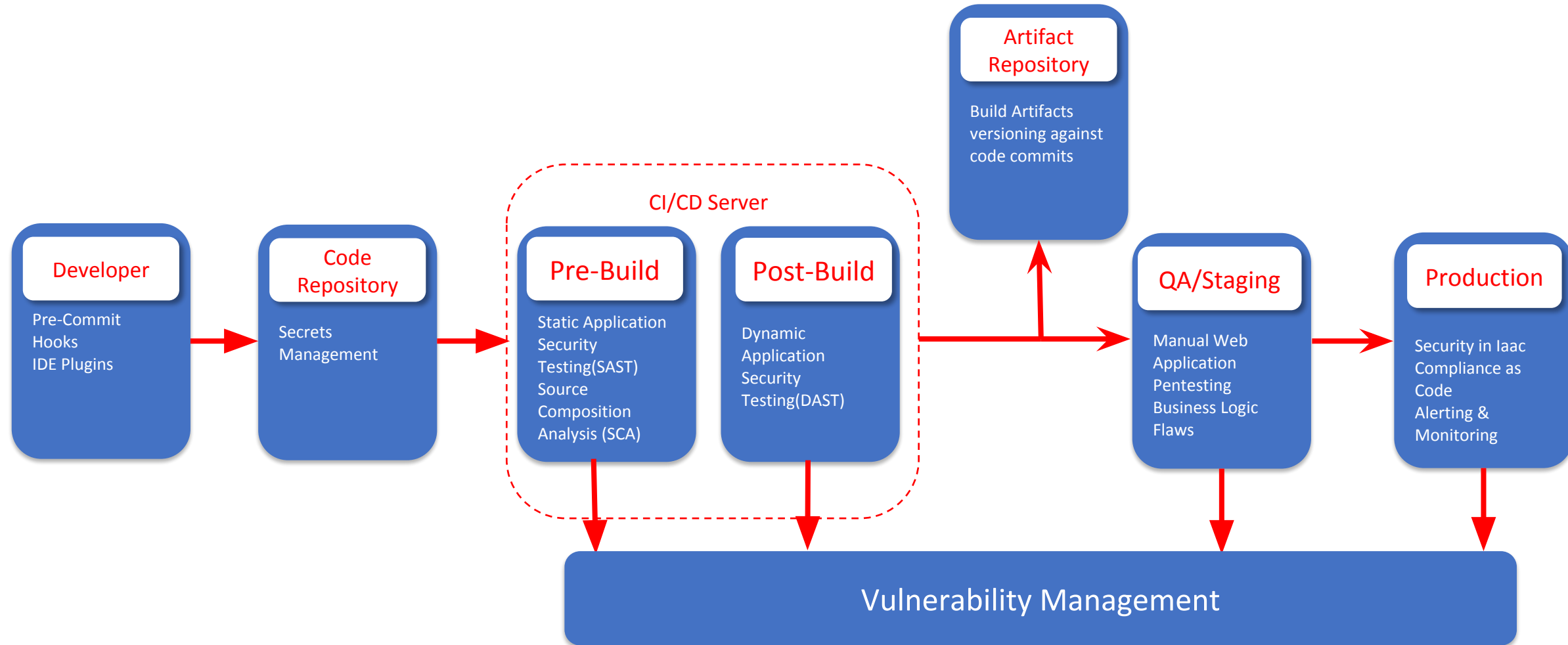
Shifting Left saves cost & time



How do we do DevSecOps?

- **DevSecOps is Automation + Cultural Changes**
- **Integrate security tools into your DevOps Pipeline**
- **Enable cultural changes to embrace DevSecOps**

Injecting Sec in DevOps

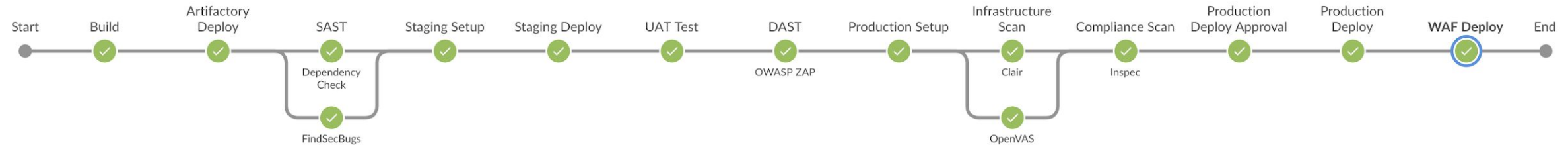


DevOps ---> DevSecOps

DevOps Pipeline



DevSecOps Pipeline



Pre-Commit Hooks

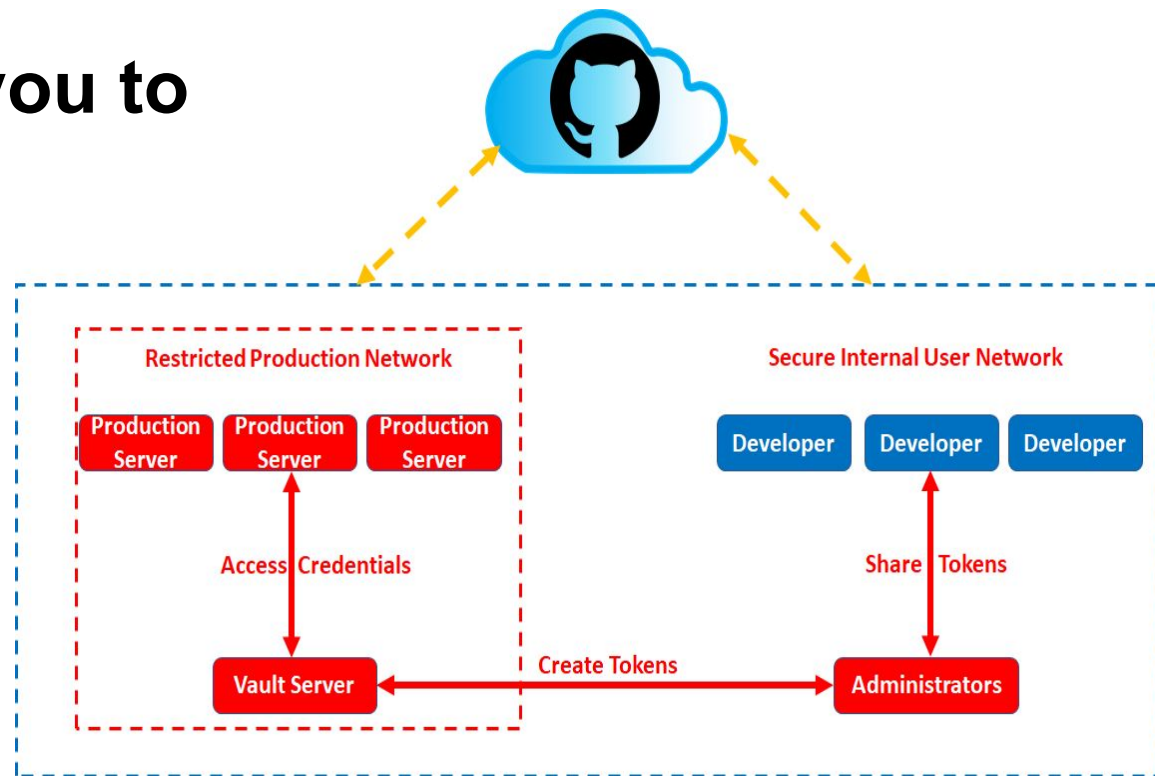
- Sensitive information such as the access keys, access tokens, SSH keys etc. are often erroneously leaked due to accidental git commits
- Pre-commit hooks can be installed on developer's workstations to avoid the same
- Work on pure Regex-based approach for filtering sensitive data
- *If developers want they can circumvent this step hence use it like a defense in depth but don't fully rely on it*

IDE Security Plugin

- IDE Plugin's provide quick actionable pointer to developers
- It is useful to stop silly security blunders
- Work on pure Regex-based approach
- *If developers want they can circumvent this step hence use it like a defense in depth but don't fully rely on it*

Secrets Management

- Often credentials are stored in config files
- Leakage can result in abuse scenario
- Secrets Management allows you to tokenize the information



Software Composition Analysis

- We don't write software's, we build on frameworks
- Biggest portion of software is now third party libraries
- Major languages provide module managements
 - PIP, NPM, Gems, go get, perl cpan, php packager and more
- Software Composition Analysis performs checks to identify vulnerable/outdated 3rd party libraries

Static Analysis Security Testing

- **White-box security testing using automated tools**
- **Useful for weeding out low-hanging fruits like SQL Injection, Cross-Site Scripting, insecure libraries etc**
- **Tool by default configured with generic setting, needs manual oversight for managing false-positives**

Dynamic Analysis Security Testing

- **Black/Grey-box security testing using automated tools**
- **SAST may not get full picture without application deployment**
- **DAST will help in picking out deployment specific issues**
- **Results from DAST and SAST can be compared to weed out false-positives**
- **Tools may need prior set of configuration settings to give good results**

Security in Infrastructure as Code

- **Infrastructure as a code allows you to document and version control the infra**
- **It also allows you to perform audit on the infrastructure**
- **Docker / K8s infra relies on base images**
- **Environment is as secure as the base image**
- **Base images need to be minimal in nature and need to be assessed to identify inherited vulnerabilities**

Compliance as Code

- **Compliance could be industry standard (PCI DSS, HIPAA, SOX) or org specific**
- **Compliance is essentially a set of rules and hence can be converted into written test cases**
- **Having written code format this can again be version controlled**

Vulnerability Management

- All the tools discussed above result in report fatigue
- Every tool has a different style of presentation
- A central dashboard is required to normalize the data
- Vulnerability Management System can then be integrated to bug tracking systems to allow devs to work on items

Alerting and Monitoring

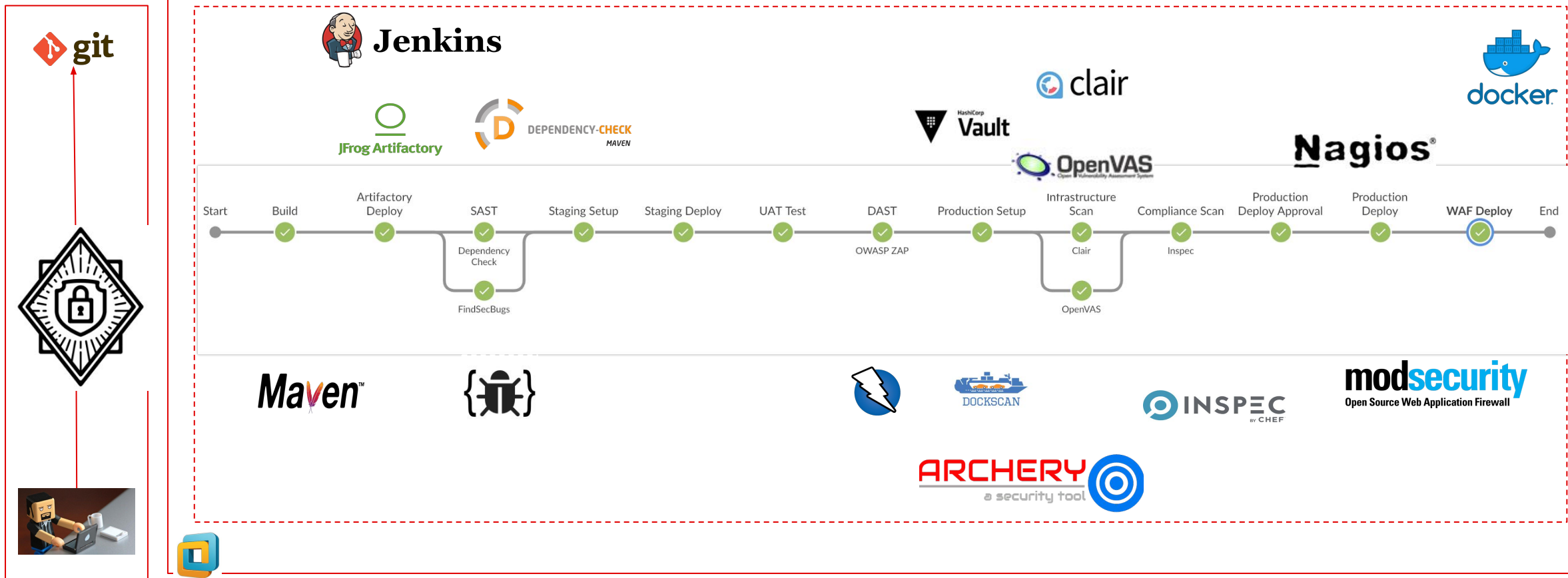
- **Monitoring is needed for two end goals**
 - **Understand if our security controls are effective**
 - **What and where we need to improve**
- **To test Security control effectiveness:**
 - **When did an attack occur**
 - **Was it blocked or not**
 - **What level of access was achieved**
 - **what data was bought in and bought out**

Asset Monitoring

- **With recent advancements assets now should include anything and everything where organization data resides**
- **With rapid development & provisioning the asset inventory can't be a static inventory**
- **We need to monitor the assets constantly both on premise and Cloud**

Sample Implementation - Java

A simplistic flow of DevSecOps Pipeline incorporating various stages



Tools of The Trade

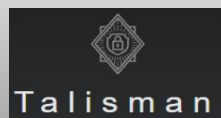
Threat Modelling Tools



ThreatSpec

Microsoft Threat
Modeling Tool

Pre-Commit Hooks



git-secret

truffleHog

Git Hound

Software Composition Analysis



DEPENDENCY-CHECK

Requires.io

Retire.js

Static Analysis Security Testing (SAST)



Bandit



RIPS

sonarqube



IDE Plugins



CAT.net



Secret Management



HashiCorp
Vault

Keywhiz



Confidant

Preference given to open-source tools; we don't endorse any tool

Tools of The Trade

Vulnerability Management



Dynamic Analysis Security Testing (DAST)



Security in Infrastructure as Code



Compliance as Code



WAF



Preference given to open-source tools; we don't endorse any tool

To be or Not to Be in Pipeline

- **API / command line access**
- **Execution start to final output should be 15 minutes max**
- **Tools should be Containerizable / scriptable**
- **Minimal licensing limitations (parallel scans or threads)**
- **Output format parsable / machine readable (no to stdout, yes to json / xml)**
- **Configurable to counter false negatives / false positives**






















Pipeline Optimization

- Pipeline to be tweaked based on Milestone (Initiative/Epic/Story)
- Remember initial onboarding is tedious
- Ensure dataset dependent tool get frequent data refresh
- Sample optimization
 - Only CSS Changes: no need for SCA
 - Only pom.xml or gradle changes: no need of SAST
 - If Infra as code has zero changes skip or fast track infra scan
- Ensure to run full (non optimized) pipeline periodically

Does Programming Language Matter

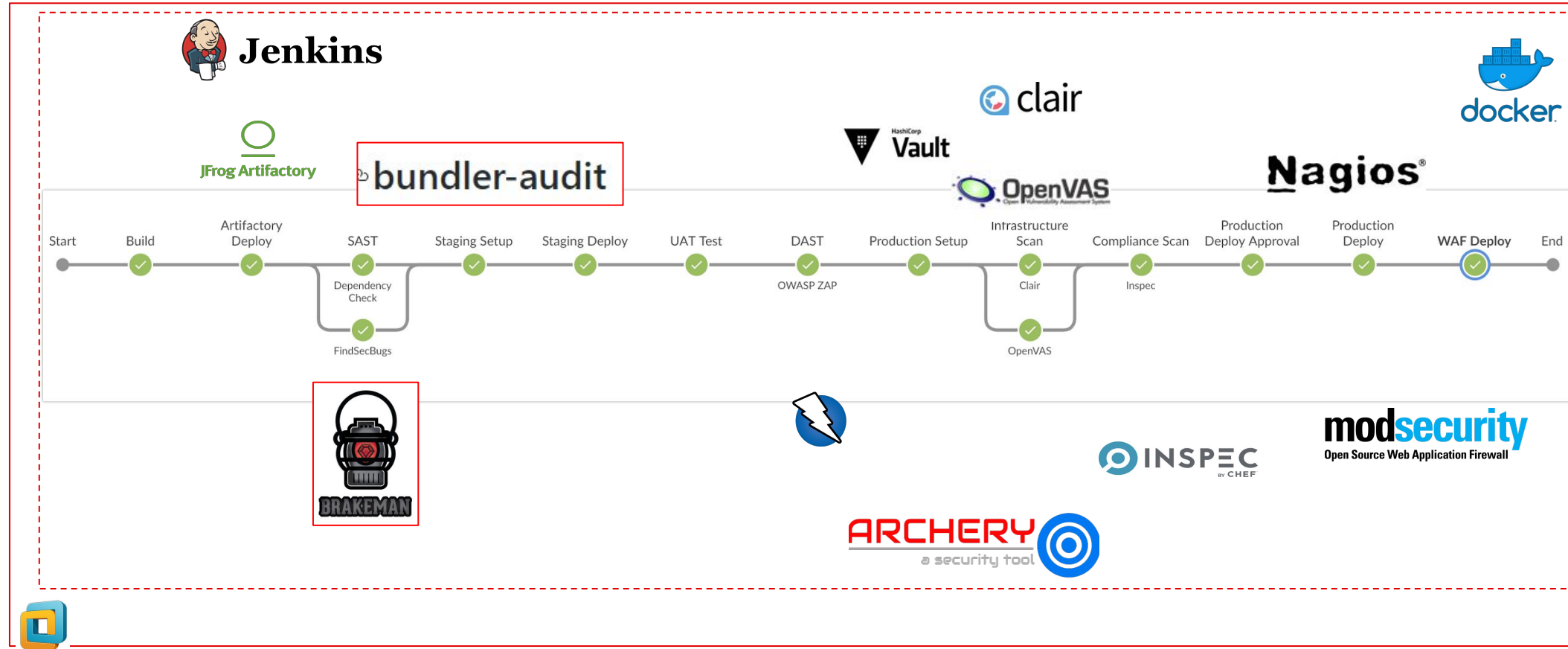
- **Different programming languages need different tools for static analysis and software composition analysis**
- **Some tools support multiple languages like sonarqube**
- **Others are focused on one language**

Language Specific Tools

Languages	Software Composition Analysis	Source Code Static Analysis
JAVA	 DEPENDENCY-CHECK  ClearlyDefined  OSS Review Toolkit	 sonarqube  graudit
PHP	 OSS Review Toolkit  sonatype	sonarqube  graudit
Python	 DEPENDENCY-CHECK  Safety  sonatype	 Bandit graudit sonarqube 
.NET	DotNET Retire SafeNuGet  OSS Review Toolkit	 PUMA SCAN  DotNet Security Guard
Ruby/Rails	 OSS Review Toolkit  sonatype	 Brakeman sonarqube  graudit
Node JS	 ClearlyDefined  OSS Review Toolkit	npm-check NodeJsScan sonarqube 

Preference given to open-source tools; we don't endorse any tool

DevSecOps Lab - Ruby



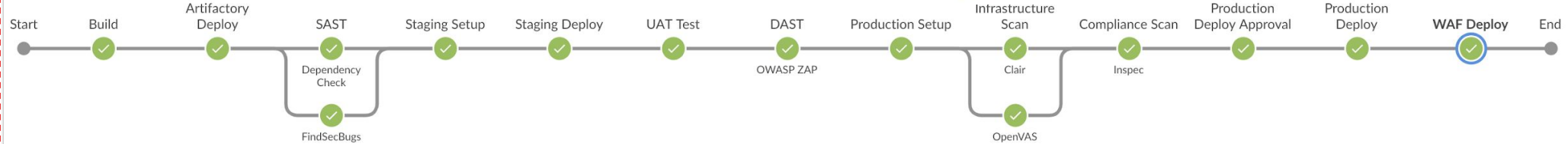
DevSecOps Lab - PHP



SensioLabs Security Checker



Nagios



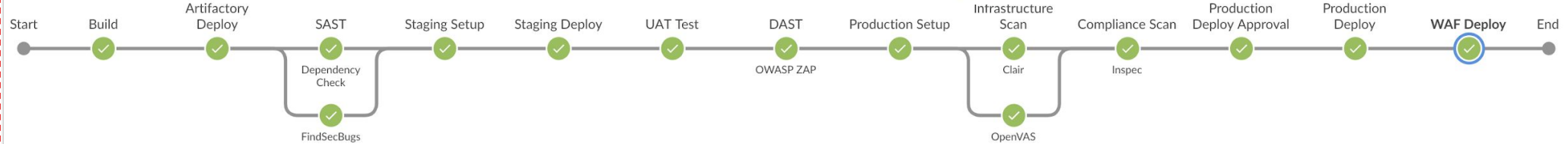
phpcs-security-audit v2



modsecurity
Open Source Web Application Firewall



DevSecOps Lab - Python



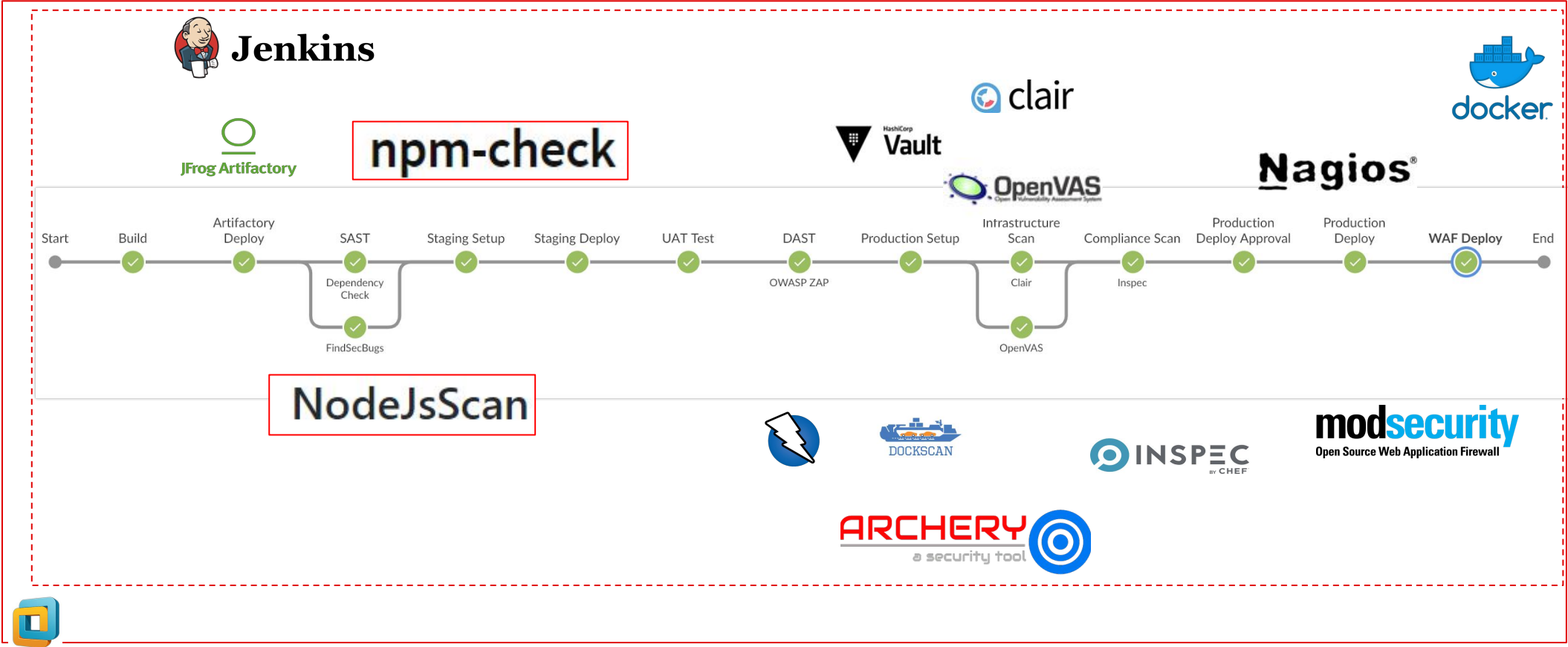
DevSecOps Lab - NodeJs



git







What about Cloud

- **The Threat Landscape changes**

- Identity and Access Management
- Asset Inventory
- Billing



- **Infrastructure as Code allows quick audit / linting**



- **Focus more on:**

- Security groups
- Permissions to resources
- Rogue / shadow admins
- Forgotten resources (compromises / billing)



Cloud Native Approach to Security

- **Different Service Providers Approach Security Differently**
- **All of them provide some of the ingredient in-house**
- **Irrespective of Cloud provider some tools will still need to be sourced**
 - **Static Code Analysis Tool**
 - **Dynamic Code Analysis Tool**
 - **Software Composition Analysis**
 - **Vulnerability Management Tool**

Cloud Native Dev[Sec]Ops

	Conventional Infra	AWS	Azure	GCP
Source Code Management	Bitbucket, Github, Gitlab etc..	AWS CloudCommit	Azure Repos	Cloud Source Repositories
Infrastructure As a Code	Chef, Puppet, Ansible more..	Amazon CloudFormation	Azure DevTest Labs	Cloud Code
CI/CD Server	Jenkins, Bamboo, Gitlab, Travis CI, Circleci more	AWS CodeBuild AWS CodeDeploy AWS CodePipeline	Azure Pipelines, Azure Test Plans	Cloud Build, Tekton
Artifactory Repository	jFrog Artifactory, Sonatype Nexus, more..	Amazon S3	Azure Artifacts	Cloud Firestore
Stg/Prod Servers	VMWare, On-premises servers	EC2 ECS (Elastic Containers) EKS (Elastic Kubernetes)	Virtual Machines, Azure Lab Services, Azure Kubernetes Service (AKS)	Compute Engine, App Engine, Shielded VMs
Monitoring & Alert	Nagios, Graphite, Grafana	AWS CloudWatch	Azure Monitor, Network Watcher	Access Transparency
Firewall	Modsecurity	AWS Firewall Manager, AWS WAF	Azure Firewall	Application Gateway
DLP	MyDLP, OpenDLP	Amazon Macie	Azure Information Protection	Cloud Data Loss Prevention
Threat Detection	Snort, Kismet	Amazon GuardDuty	Azure Advanced Threat Protection	Event Threat Detection (beta)
Vulnerability Scanning	OpenVAS, Nessus	Amazon Inspector	Azure Security Center	Cloud Security Scanner
Secrets Management	Hashicorp Vault, Docker Secrets	AWS Secrets Manager	Azure Key Vault	Secrets management

Cultural Aspect

- Automation alone will not solve the problems
- Encourage security mindset especially if outside sec team
- Cultivate/Identify common goals for greater good
- Build allies (security champions) in company
- Focus on collaboration and inclusive culture
- Avoid Blame Game



Security team should try to eliminate the need of dedicated security team

Security Champion

- **Bridge between Dev, Sec and Ops teams**
- **Single Person per team**
- **Everyone provided with similar cross skilling opportunities**
- **Incentivize other teams to collaborate with Sec team**
 - **Internal Bug bounties**
 - **Sponsor Interactions (Parties / get-togethers)**
 - **Sponsor cross skilling trainings for other teams**

Security Enablers

People

- Build relationships between teams, don't isolate
- Identify, nurture security conscious individuals
- Empower Dev / ops to deliver better and faster and secure, instead of blocking.
- Focus on solutions instead of blaming




Process

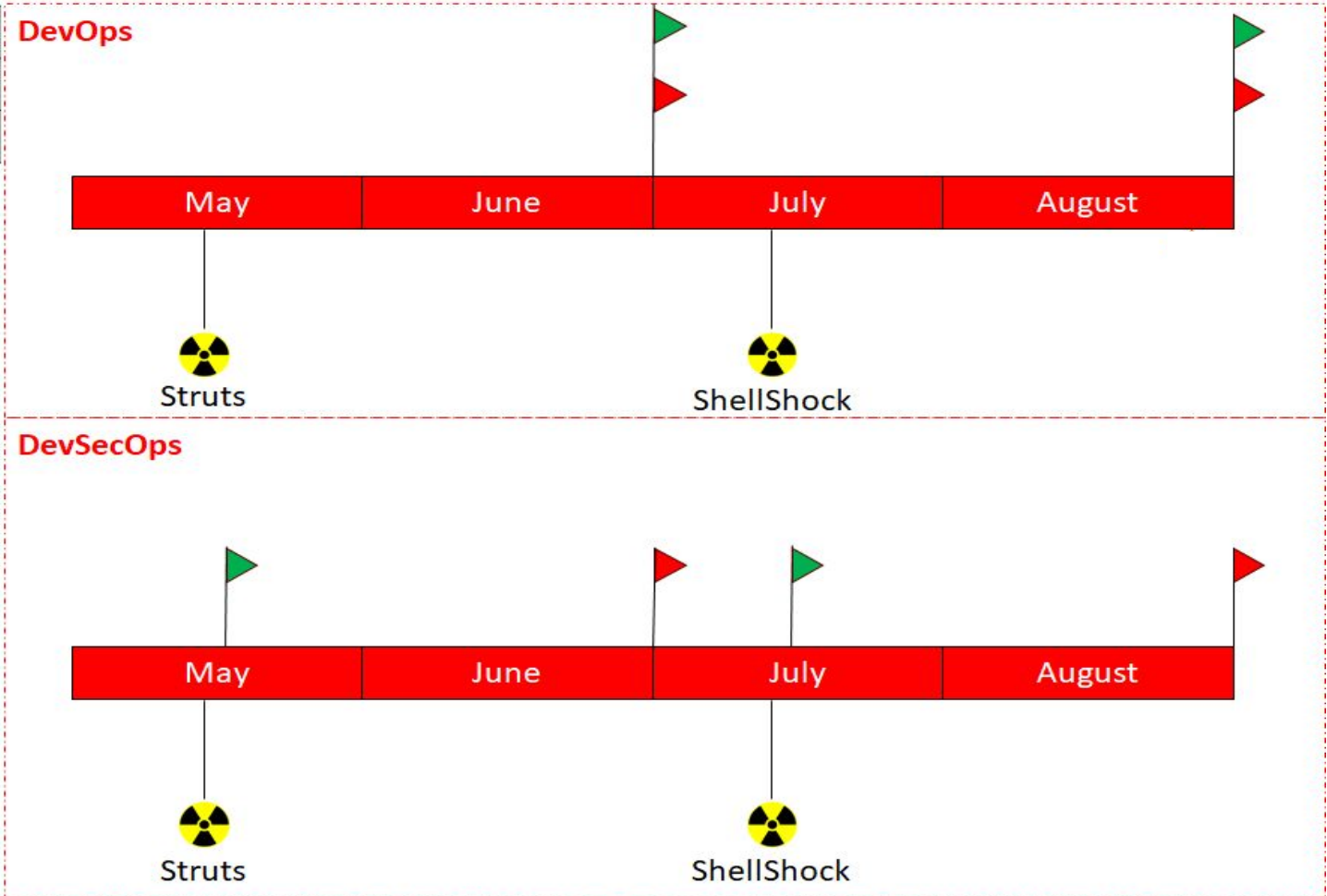
- Involve security from get-go (design or ideation phase)
- Fix by priority, don't attempt to fix it all
- Security Controls must be programmable and automated wherever possible
- DevSecOps Feedback process must be smooth and governed

Technology

- Templatize scripts/tools per language/platform
- Adopt security to devops flow don't expect others to adopt security
- Keep an eye out for simpler and better options and be pragmatic to test and use new tools

Generic Case Study

	Manual Pentest
	Zero Day
	Zero Day Resolved



Key Takeaways

- **Security is everyone responsibility**
- **Embrace security as an integral part of the process, use feedback to refine the process**
- **DevSecOps is not a one size fit all: your mileage will vary**