### **Code Signing tools**

* all releases => <https://ebourg.github.io/jsign/> => opensource/microsoft Auth
* docker images - DCT(docker content trust)
* tarball - openssl
* rpm - Gpg key signer

### **Why DevSecOps?**

DevOps is well-understood in the IT world by now, but it's not flawless. Imagine you have implemented all of the DevOps engineering practices in modern application delivery for a project. You've reached the end of the development pipeline—but a penetration testing team (internal or external) has detected a security flaw and come up with a report. Now you have to re-initiate all of your processes and ask developers to fix the flaw.

**What is DevSecOps?**

It integrates active and automated security audits and penetration testing into agile application development.

To utilize [DevSecOps](https://resources.whitesourcesoftware.com/blog-whitesource/devsecops), you need to:

* Introduce the concept of security right from the start of the SDLC to minimize vulnerabilities in software code.
* Ensure everyone (including developers and IT operations teams) shares responsibility for following security practices in their tasks.
* Integrate security controls, tools, and processes at the start of the DevOps workflow. These will enable automated security checks at each stage of software delivery.

**Understanding the DevSecOps pipeline**

* **Plan**: Execute security analysis and create a test plan to determine scenarios for where, how, and when testing will be done.
* **Code**: Deploy linting tools and Git controls to secure passwords and API keys.
* **Build**: While building code for execution, incorporate static application security testing (SAST) tools to track down flaws in code before deploying to production. These tools are specific to programming languages.
* **Test**: Use dynamic application security testing (DAST) tools to test your application while in runtime. These tools can detect errors associated with user authentication, authorization, SQL injection, and API-related endpoints.
* **Release**: Just before releasing the application, employ security analysis tools to perform thorough penetration testing and vulnerability scanning.
* **Deploy**: After completing the above tests in runtime, send a secure build to production for final deployment.

DevOps => DevSecOps Tools

1. Developer => Pre-commit hooks, IDE plugins, Linters
2. SCM => Token based secret management service
3. CI/CD server => SAST, Software Composition Analysis(SCA)
4. Nexus Repo (binary repo)
5. Staging/QA => DAST, Vulnerability Management
6. Production => Code Compliance, Pentesting, Vulnerability assessment
7. Monitoring => Alerting and Monitoring deployed instance for OWASP

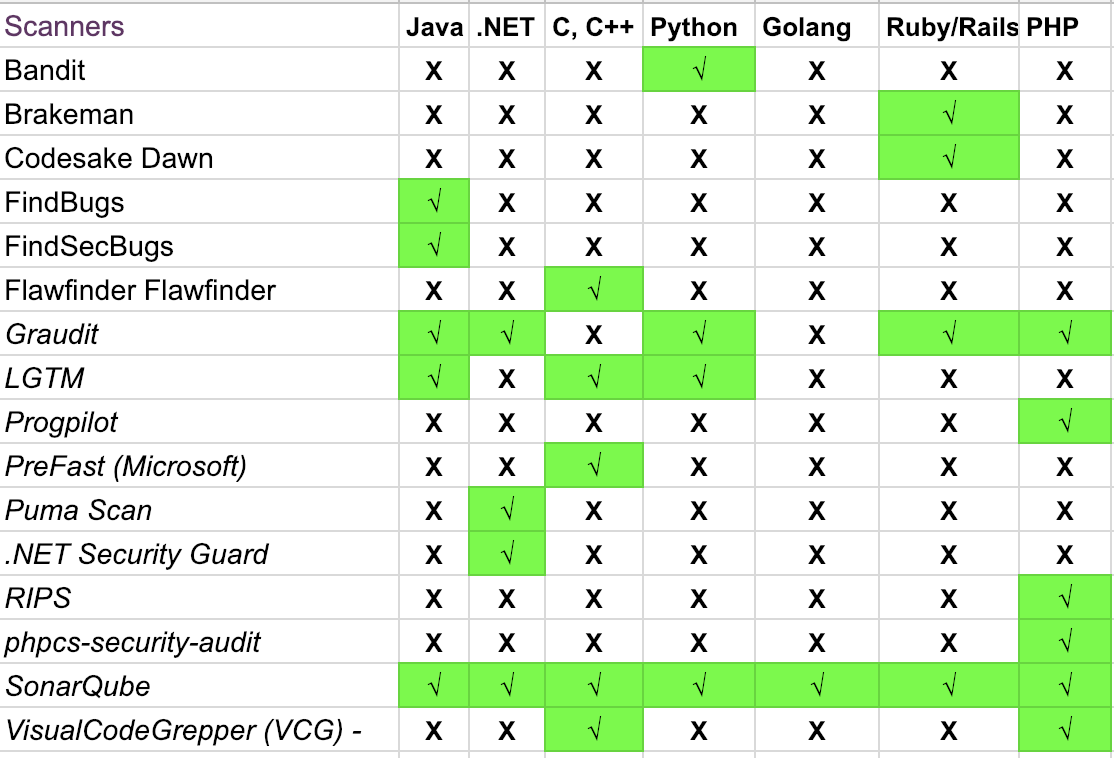
DevSecOps

1. Pre Commit Hooks - Sensitive information such as the AWS keys, access tokens, SSH keys etc
   1. Talisman – <https://github.com/thoughtworks/talisman>
   2. Crass – <https://github.com/floyd-fuh/crass>
   3. Git Hooks – <https://githooks.com/> => use this as by default available with gitlab
   4. Git Secrets – <https://git-secret.io/>
   5. Pre Commit – <https://pre-commit.com/>

https://towardsdatascience.com/pre-commit-hooks-you-must-know-ff247f5feb7e

* 1. Detect Secrets – <https://github.com/Yelp/detect-secrets>
  2. Git Hound – <https://github.com/ezekg/git-hound>
  3. Truffle Hog – <https://github.com/dxa4481/truffleHog>

1. Secret Management - Storing credentials in the files or configuration can lead to exposure of credentials to an unintended audience. This can be segregated by leveraging secret management services like “Hashicorp vault”. This allows segregation of credentials on a seperate level and every environment can pull credentials from a specific environment and use it programmatically.
   1. Hashicorp Vault – <https://www.vaultproject.io/>
      1. centralize secrets management
      2. Dynamic secrets
      3. Provides encryption as a service
      4. Popular features are free
   2. Torus – <https://www.torus.sh/>
   3. Keywhiz – <https://square.github.io/keywhiz/>
   4. EnvKey – <https://www.envkey.com/>
   5. Confidant – <https://github.com/lyft/confidant>
   6. AWS Secrets Manager – <https://aws.amazon.com/secrets-manager/>
      1. Available from AWS
      2. paid
2. Software Composition Analysis - it is necessary to perform an analysis of all the dependencies being utilised in the application and check them for vulnerabilities arising from missing security patches. For Java and .NET applications, we can make use of a tool called “Dependency-Check” which can be run before creating the builds to identify if any vulnerable software is being used in the application.
   1. OWASP Dependency Check – <https://www.owasp.org/index.php/OWASP_Dependency_Check>
      1. free
   2. SonaType (Free for Open Source) – <https://ossindex.sonatype.org/>
   3. Snyk (Free for Open Source) – <https://snyk.io/>
      1. free/paid
   4. Bunder Audit – <https://github.com/rubysec/bundler-audit>
   5. Rubysec – <https://rubysec.com/>
   6. Retire JS – <https://github.com/RetireJS/retire.js>
3. SAST(Static Analysis Security Testing) - For Java based applications we can make use of a tool called “FindSecBugs” which performs an in-depth analysis of the code(without giving out too many false-positives) and gives a comprehensive report for all the vulnerabilities that have been identified in the code.



1. DAST(Dynamic Analysis Security Testing) - Web application scanners are an important part in doing a vulnerability assessment of a web application
   1. OWASP ZAP – <https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project>
   2. Arachni Scanner – <http://www.arachni-scanner.com/>
   3. Nikto – <https://cirt.net/Nikto2>
   4. Gitlab DAST - https://docs.gitlab.com/ee/user/application\_security/dast/
2. Security in Infrastructure as a code - Docker image scanning

**Hosts**: run scheduled checks to determine if the host configuration is invalid or outdated. Some tools give you a ‘security score’. You can easily see how your security improves over time if it is integrated with monitoring systems.

**Containers**: monitor every call/command that your container makes. Anything suspicious or non-compliant is blocked.

**External Vulnerabilities**: a check that is run periodically against your infrastructure (updated continuously with the latest threats) to identify vulnerabilities.

* 1. Clair – <https://github.com/coreos/clair>
     1. From coreos
     2. Installed from helm charts
  2. Anchore Engine – <https://github.com/anchore/anchore-engine>
     1. Anchore container image scanner jenkins plugin available
  3. Dagda – <https://github.com/eliasgranderubio/dagda>
  4. Open-Scap – <https://www.open-scap.org/getting-started/>
  5. Docksan – <https://github.com/kost/dockscan>

### Vulnerability Assessment(VA) - It is a general practice to perform vulnerability assessment on the production systems to identify various services running in the environment and the associated vulnerabilities. While pointing a VA tool on the servers that have been created using Docker, it would execute the scan only on the service that is being exposed on that host. However, if we attach the tool to the docker network and then execute the scan, then it would give us a good picture of services which are actually running.

* 1. OpenVAS – <http://openvas.org/>
     1. **Open-source and Free of Cost**
     2. **Custom Configurable**
     3. **Good CVE Coverage**
     4. **Detailed Documentation and Tutorials**
     5. **Cons:**
        1. **Covers Less Vulnerabilities Compared to Nessus**
        2. **OS Support Limitation**
  2. DockScan – <https://github.com/kost/dockscan>
  3. Qualys - https://www.qualys.com/apps/web-app-scanning/
     1. Patch Management
     2. Prioritization
     3. Risk Management
     4. Vulnerability Assessment
     5. Web Scanning
  4. Nessus - <https://www.tenable.com/products/nessus>
     1. Pros:
        1. Very low price which every organization can afford
        2. Vulnerability scanning of network including IPv4 network ,IPv6 network
        3. You can scan all common companies' network devices, all common virtualization companies' platforms, and all the operating systems, etc.

1. Compliance as a code - With “Infrastructure as Code” in DevOps, the production environment is never retained, it is always torn down and re-created again and hence it is a strong requirement to test the updated/newly created environment after it has been set up. “Inspec” is one such tool which can help us in performing these tests as we only need to supply a ruby file containing the tests to be conducted in a very simple and lucid manner which is easy for every audit professional to write and code.
   1. Inspec – <https://www.inspec.io/>
      1. **Platform Agnostic**
      2. **Free to run anywhere**
      3. **Easily extensible for new languages**
   2. Serverspec – <https://serverspec.org/>
   3. DevSec Hardening Framework – <https://dev-sec.io/>
   4. Kitchen CI – <https://kitchen.ci/>
2. Vulnerability Management - The tools that we shall use to create a DevSecOps pipeline will generate plenty of vulnerabilities and each tool will have its own separate format. It becomes difficult to manage this data let alone track and remediate the vulnerabilities. Hence, vulnerability management solutions are at the core of a DevSecOps process where all tools are required to spool their data into those solutions so that it can be centrally managed, triaged, tracked and remediated.
   1. ArcherySec – <https://github.com/archerysec/archerysec>
   2. DefectDojo – <https://www.defectdojo.org/>
   3. JackHammer – <https://github.com/olacabs/jackhammer>
   4. Qualys - Already we are using qualys so lets go ahead with that and its best in industry
3. Alerting and Monitoring - Production applications are always faced with new threats from unknown and unforeseen vectors. This can be mitigated by having an active intrusion monitoring and prevention solution. One such opensource solution is the “ModSecurity WAF(Web Application Firewall) which detects OWASP Top 10 vulnerabilities like SQL injection,Cross-site scripting etc. being attempted against the application
   1. ModSecurity WAF – <https://modsecurity.org/>

## **What Are the Benefits of DevSecOps?**

* Cost reduction is achieved by detecting and fixing security issues during the development phases.
* Speed of delivery is increased as security bottlenecks are minimised or eliminated.
* Speed of recovery is enhanced in the case of a security incident by utilising templates
* Enhanced monitoring and auditing leads to improved threat hunting, which reduces the likelihood of a breach, avoiding bad publicity and reputational damage
* Immutable infrastructure allows companies to tear down infrastructure while managing an attack vector identified by scanning. If a node is compromised, it won’t remain compromised for long, as it will be torn down and rebuilt with new credentials.
* Immutable infrastructure improves overall security by reducing vulnerabilities, and increasing code coverage and automation.
* Security auditing, monitoring, and notification systems are managed and deployed so that they can be continuously enhanced, to keep in step with the frantic innovation intrinsic to cybercrime.
* Ensures the ‘secure by design’ principle by using automated security review of code, automated application security testing, educating, and empowering developers to use secure design patterns.
* Creates targeted customer value through secure iterative innovation at speed and scale.
* Security is federated and becomes the responsibility of everyone, not just a specialised team, or even individual

=================Devsecops discussion============

Should we have sonarqube integration on master branch or developer branch? => master

check more on snyk

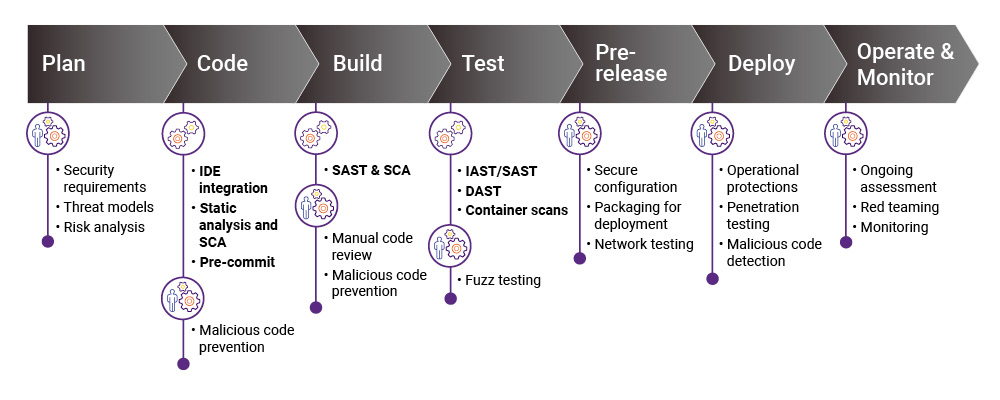
paladino =>

cantino =>

Whitesource =>

Qualys =>

include cost as well



Snyk Features:

* Snyk Open Source - Find and automatically fix open source vulnerabilities.
* Snyk Code - Find and fix vulnerabilities in your application code in real time
* Snyk Container - Find and fix container vulnerabilities and Kubernetes Applications
* Snyk Infrastructure as a Code - Find and fix insecure configurations In Terraform and Kubernetes Code

Pricing:

Enterprise Plan:

Automatic and manual fixes for vulnerable dependencies

* Automatic and manual updates for vulnerable container base images
* Public container registry integration (Docker Hub, ECR, ACR, GCR)
* Cloud source code integration (GitHub, GitLab, Bitbucket, and Azure Repos)
* CI/CD pipeline integration
* Priority scoring
* IDE plugins
* Jira integration
* License policy management
* Kubernetes monitoring and prioritization
* Rich API
* Reports
* Customized SSO setup
* On-prem container registries
* Private registries (Artifactory, Nexus)
* Service Accounts
* Security policy management (beta)
* Self-managed source code (GitHub Enterprise, Bitbucket Server, GitLab Enterprise, and Azure Devops Server)

Self-hosted option

Qualys:

<https://www.qualys.com/solutions/devops/#developers>

* Comprehensive bug, misconfiguration detection
  + Catches coding and configuration errors throughout development, early and often, before launching apps in production
* Remediation prioritization
  + Pinpoints the most critical vulnerabilities present in code being written, so you can eliminate the biggest risks right away
* Compliance assurance
  + Verifies that as applications are developed, the code is compliant with your internal policies and external regulations
* Intrusion vigilance
  + Identifies indicators of compromise so your combined development, operations, QA and security team responds and secures systems immediately
* Streamlined security and compliance audits
  + Automates checks of security controls and configurations, and expedites demonstration of compliance
* Easy integration into your DevOps toolchain
* Access all services through robust REST APIs
* Easily track and communicate results
* Comprehensive security and compliance platform
* Public cloud platforms integration
* Container Security

WhiteSource:

<https://www.whitesourcesoftware.com/whitesource-pricing/>

* Detailed remediation guidance
* License compliance
* Native integration with Azure pipelines
* Package scanning
* Dependency detection including transitive dependencies
* Real-time vulnerability alerts
* Auto fix pull requests
* Outdated dependency update score
* Native integration with the most common browsers, IDEs, and hosted repositories
* Premium support\*
* Professional services\*
* Automatic workflows and policies
* Containers support
* Integrates with any package manager, build tool, or CI server
* Native integration with Github.com
* Prioritization of effective vulnerabilities
* Supports all languages, frameworks & development environments

Auto fix pull requests

Auto fx pull requests