

**AWS/GCP/AZURE**

**DevSecOps Hands-on**

Version: 7.0

**Date:** 6/17/2024

|  |
| --- |
| **Document Control** |

**Document Information**

| Document Title | GCP/AZURE DevSecOps Hands-on |
| --- | --- |
| Creation Date | 07-06-2024 |
| Author | Sudhansu Sekhar Sahoo |
| Version | v 7.0 |
| Latest Update | 07-06-2024 |

|  |
| --- |
| **Table of Content** |

[1 DevSecOps Introduction 3](#_Toc168854888)

[1.1 Basic Security Terms 3](#_Toc168854889)

[1.2 Tools for implementing DevSecOps 4](#_Toc168854890)

[2 DevSecOps in GCP 4](#_Toc168854891)

[2.1 Sonar Cloud and its benefits 5](#_Toc168854892)

[2.2 SNYK 8](#_Toc168854893)

[2.3 OWASP ZAP (Zed Attack Proxy) 8](#_Toc168854894)

[2.4 Case Study: Create Java Project DevSecOps Pipeline in GCP (End to End) 9](#_Toc168854895)

[3 DevSecOps with Azure DevOps 10](#_Toc168854896)

[3.1 DevSecOps in Azure DevOps 10](#_Toc168854897)

[3.2 SonarCloud Integration 13](#_Toc168854898)

[3.3 Snyk Integration 13](#_Toc168854899)

[3.4 MODULAR ADO DEVSECOPS PIPELINE 15](#_Toc168854900)

[4 Azure DevOps 16](#_Toc168854901)

[4.1 Azure DevOps Platform 17](#_Toc168854902)

[4.2 Automated Testing 17](#_Toc168854903)

[4.3 Securing CI/CD Pipeline 19](#_Toc168854904)

[4.4 LAB 1 19](#_Toc168854905)

[5 Container Security & CSPM (Cloud security posture management) 19](#_Toc168854906)

[5.1 CSPM (CLOUD SECURITY POSTURE MANAGEMENT) 19](#_Toc168854907)

[5.2 Qualys 20](#_Toc168854908)

[5.3 Red Team vs. Blue Team in Cybersecurity 20](#_Toc168854909)

[5.4 Trivy from AQUA to Scan Container Image 21](#_Toc168854910)

[5.5 Docker Scan to perform Container Scan Using Snyk in the Backend 22](#_Toc168854911)

[5.6 Snyk to perform Container Scan 22](#_Toc168854912)

[5.7 False Positive Analysis 22](#_Toc168854913)

[5.8 Perform FPA for Container Security Scans 22](#_Toc168854914)

[5.9 Q&A 25](#_Toc168854915)

# DevSecOps Introduction



## Basic Security Terms

* **SAST** – Static Application Security Testing

White-box testing that analyzes source code for identifying security issues.

* **SCA** – Software Composition Analysis

Scans your code base to provide visibility into open-source software components, including license compliance and security vulnerabilities.

* **DAST** – Dynamic Application Security Testing

Dynamic application security testing (DAST) is a type of black-box security testing in which dynamic tests are performed on the application.

* **IAST**- Interactive Application security Testing

It combines elements of both SAST and DAST and tries to overcome its limitations. It scans specific workflows of code.

* **IAC –** Infrastructure as Code

Infrastructure as Code is the process of creating infrastructure using Code definition files and testing these files for identifying issues in such files is called **IAC Security Testing**.

* **API Security –** Is also called Security for Microservices (Microservices is a subset of API) e.g. Ecommerce Application will have User Registration API and this API will have microservices to Create/Update/Delete users.

## Tools for implementing DevSecOps

**Security Tools used during App Development:**

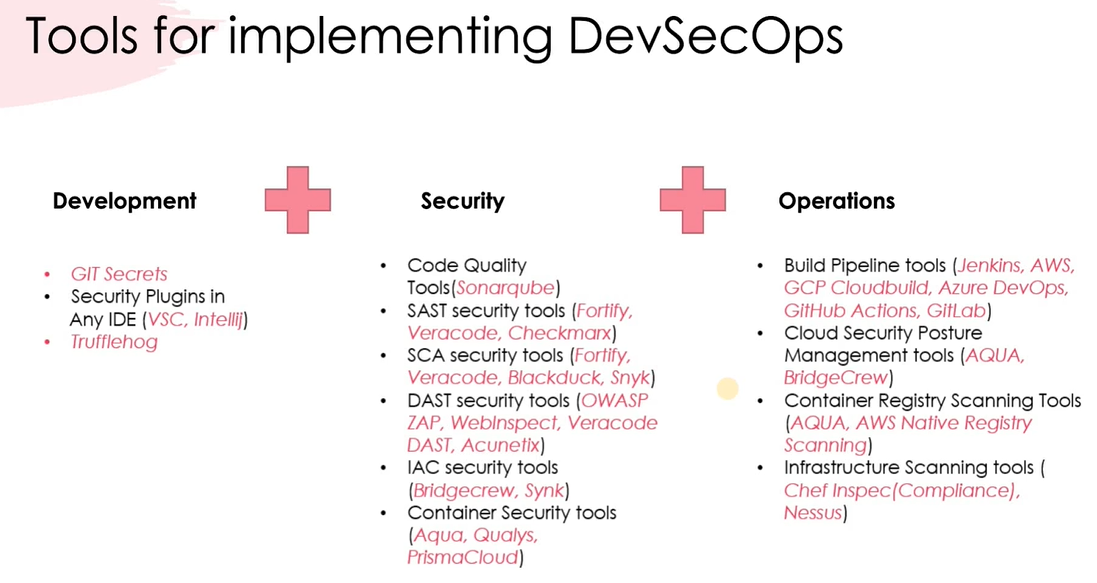
* GIT Secrets
* Security Plugins in any IDE (VS Code, IntelliJ, Eclipse)
* Trufflehog

**Security Tools used within the Build Pipelines:**

* Code Quality Tools (Sonarqube)
* SAST security tools (Fortify, Veracode, Checkmarx)
* SCA security tools (Fortify, Veracode, Blackduck Snyk)
* DAST security tools (OWASP, ZAP, WebInspect, Veracode, DAST, Acunetix)
* IAC security tools (Bridgecrew, Snyk)
* Container Security Tools (Aqua, Qualys, PrismaCloud)

**Security Tools used within Operations:**

* Build Pipeline tools (Jenkins, AWS, GCP Cloudbuild, Azure DevOps, GitHub Actions, GitLab)
* Cloud Security Posture Management tools (AQUA, BridgeCrew)
* Container Registry Scanning Tools (Chef inspect (Compliance), Nessus)
* Cloud Security tools (AWS Security Hub, Azure Defender)



# GCP DevSecOps

GCP provides the following services/tools for implementing DevSecOps:



## Sonar Cloud and its benefits

SonarCloud is a Software As a Service (SaaS) platform that helps to keep our source code free from:



* 1. Code Quality Issues
  2. Code Security Issues
* We can define quality gates within SonarCloud as per project quality standards.

sonar.host.url= https://sonarcloud.io/

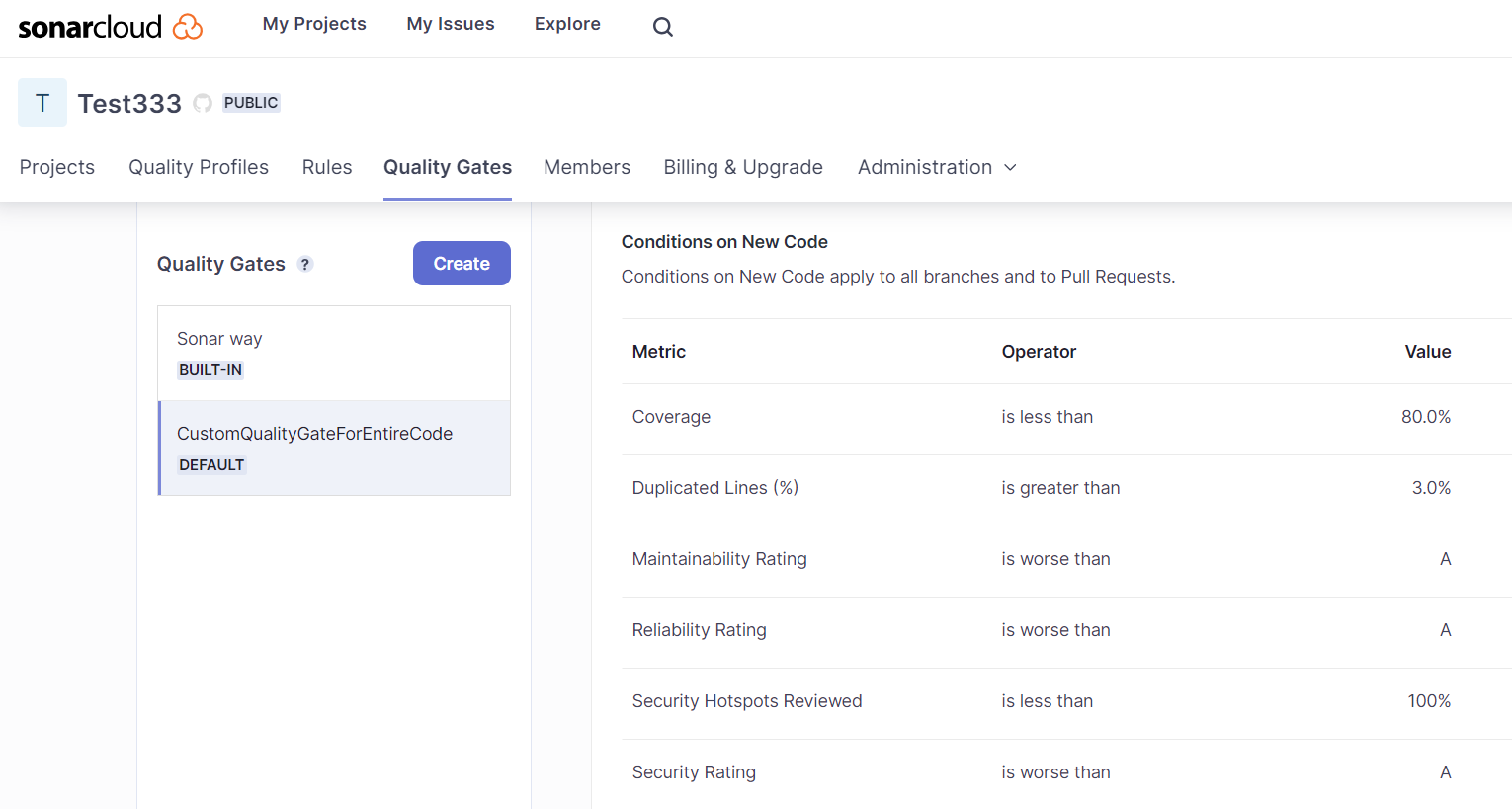
sonar.organization= test333sourcecodescan

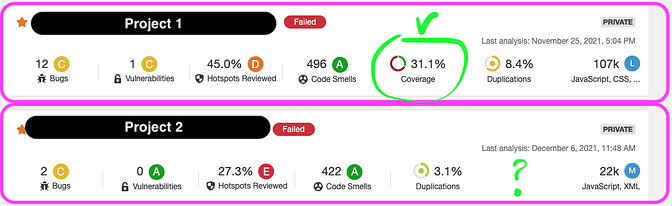
sonar.projectkey= test333sourcecodescan

sonar.login= af9814c0ddba2ea2a23673da337cd6157cc03804

Requirements to populate Code Coverage on SonarCloud Dashboard:

1. Junit Plugin in Pom.xml
2. Report Goal for Jacoco
3. Unit Tests written for the Java Classes present in source code
4. Maven verify goal is used





1. Bugs
2. Vulnerabilities
3. Hotspots Reviewed
4. Code Smells
5. Coverage
6. Duplications

**SonarCloud Findings:**

1. Issues Breakdown
2. Bugs
3. Vulnerabilities
4. Code Smells
5. Hotspots Breakdown

**CWE Top 25**

**CWE/SANS Top 25 Most Dangerous Software Weaknesses**

**OWASP Top 10**

A1: Broken Access Control

A2: Cryptographic Failures

A3: Injection

A4: Insecure Design

A5: Security Misconfiguration

A6: Vulnerable and Outdated Components

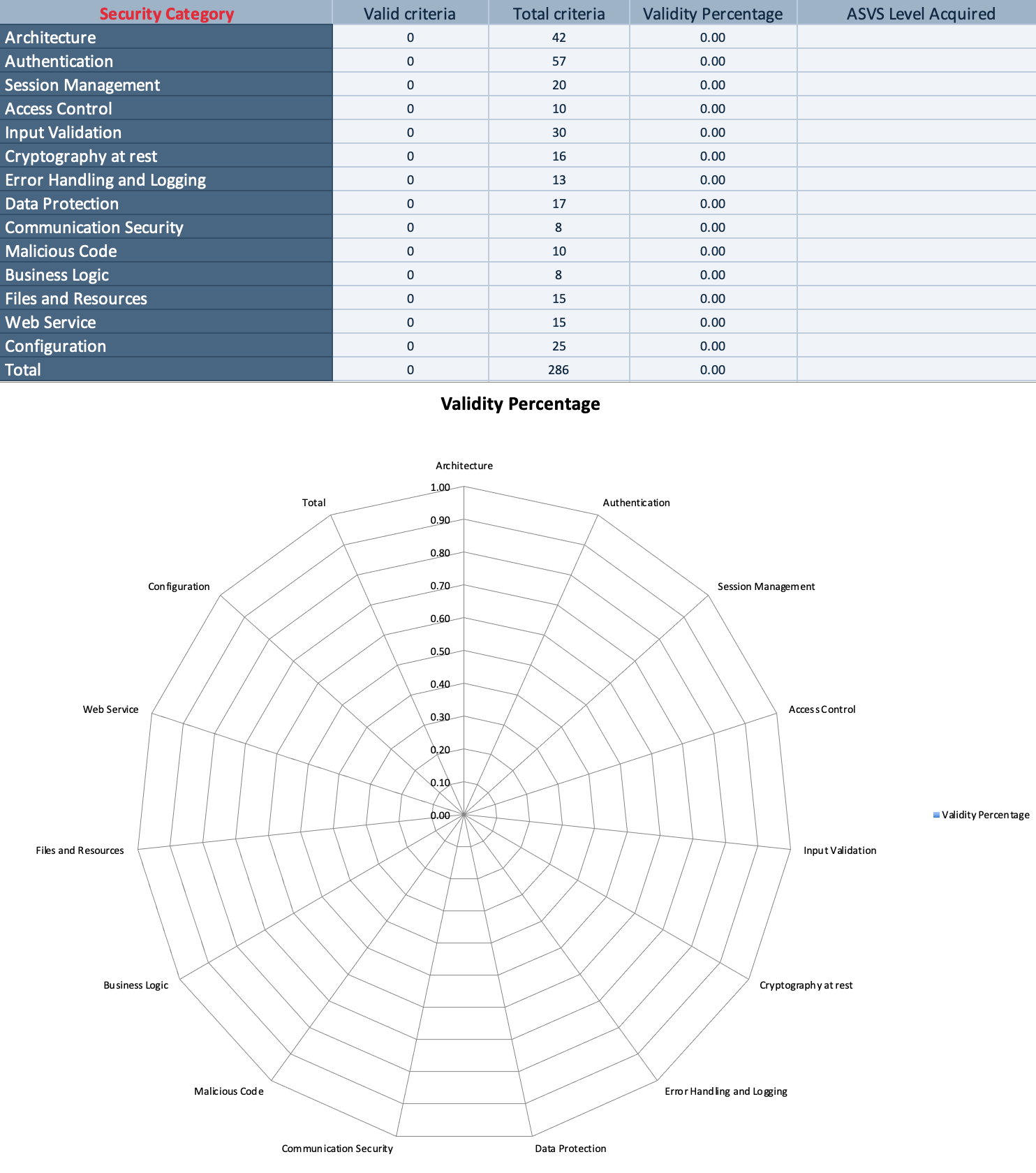
A7: Identification and Authentication Failures

A8: Software and Data Integrity Failures

A9: Security Logging and Monitoring Failures

A10: Server-Side Request Forgery (SSRF)

**OWASP ASVS (Application Security Verification Standard)**



**CPU- Central Processing Unit**

**GPU- Graphical Processing Unit**

**TPU- Tensor Processing Unit**

**DPU- Data Processing Unit**

**Performance Battle:**

**CPU**: Great for general-purpose tasks, but might struggle with heavy graphics or AI work.

**GPU**: King of graphical tasks but may not be as efficient for general computing.

**TPU**: AI’s best friend, lightning-fast for machine learning tasks.

**DPU**: Emerging as the powerhouse for data center optimization.

**Use Case Showdown:**

**CPU**: Your go-to for everyday tasks, office work, and browsing.

**GPU**: Essential for gaming, graphic design, and video editing.

**TPU**: Vital for AI research, data analysis, and deep learning applications.

**DPU**: Revolutionizing data centers, enhancing security, and optimizing network performance.

## SNYK

Snyk is an organization that develops security tools (SaaS) to secure:

* Source Code
* Open Source/Third Party libraries
* Containers
* Infra As Code

We will use Snyk specifically for software composition analysis scan.

So, software composition analysis scan is used to identify security issues within third party libraries that are present in our source code. E.g. Log4j is a 3rd party library that is used in many Spring boot projects, so Snyk will identify all security issues present in Log4j.

Snyk Token: da7e5c98-e59e-43f7-b5db-654ac99bfecf

## OWASP ZAP (Zed Attack Proxy)



**OWASP: OPEN WEB APPLICATION SECURITY PROJECT**

OWASP ZAP is an open-source web application security scanner. It is intended to be used by both those new to application security as well as professional penetration testers. It is one of the most active OWASP projects and has been given Flagship status. It can scan both:

* + Web Applications
  + API Specifications

Creating a Storage Bucket in GCP to store the ZAP report.

ZAP can scan through the web application and detect issues related to:

* SQL injection
* Broken Authentication
* Sensitive data exposure
* Broken Access control
* Security misconfiguration
* Cross Site Scripting (XSS)
* Insecure Deserialization
* Components with known vulnerabilities
* Missing security headers

## Case Study: Create Java Project DevSecOps Pipeline in GCP (End to End)



List of Security Branches as a Career:

* 1. DevSecOps
  2. Cloud Penetration Testing
  3. Container Security
  4. Security Architecture and Design
  5. Web Penetration Testing
  6. Mobile Penetration Testing
  7. Cloud Security & Compliance

# Azure DevSecOps

## DevSecOps in Azure DevOps



**Question-1:**

Write a YAML file that defines an Azure DevOps CI/CD pipeline with two steps: "build" and "test". The pipeline should be triggered whenever there is a code change in "master" branch. The code should be run on "ubuntu-latest" machine.

1) The "build" step should be called "Building Azure DevOps App", which runs the script "mvn clean package".

2) The "test" step should be called "Testing Azure DevOps App", which runs the script "mvn clean test".

trigger:

  branches:

    include:

      - master

pool:

  vmImage: 'ubuntu-latest'

jobs:

- job: build

  displayName: 'Building and Testing Azure DevOps App'

  steps:

    - task: Maven@3

      displayName: 'Building Azure DevOps App'

      inputs:

        goals: 'clean package'

    - task: Maven@3

      displayName: 'Testing Azure DevOps App'

      inputs:

        goals: 'clean test'

----------------------------------------------------------------

trigger:

- master

pool:

  vmImage: ubuntu-latest

steps:

- script: |

    mvn clean package

  displayName: "Building Azure DevOps App"

- script: |

    mvn clean test

  displayName: "Testing Azure DevOps App"

**Question-2:**

Write a YAML file that defines an Azure DevOps DevSecOps pipeline with three steps: "build", "test" and "sast". The pipeline should be triggered whenever there is a code change in "master" branch. The code should be run on "ubuntu-latest" machine.

1) The "build" step should be called "Building Azure DevOps App", which runs the script "mvn clean package".

2) The "test" step should be called "Testing Azure DevOps App", which runs the script "mvn clean test".

3) The "sast" step should be called "SAST Scan Azure DevOps App", which runs the SonarCloud script. Use the below values for integrating SonarCloud in DevSecOps pipeline:

Sonar URL = <https://sonarcloud.io/>

Sonar Org = azuredevopsdevsecopsoorg

Sonar ProjectKey = azuredevopsadodevsecopsprojectkey

Sonar Login = 14ad47384add02744d3f9e34d

trigger:

  branches:

    include:

      - master

pool:

  vmImage: 'ubuntu-latest'

jobs:

- job: build

  displayName: 'Building, Testing, and SAST Scan Azure DevOps App'

  steps:

    - task: Maven@3

      displayName: 'Building Azure DevOps App'

      inputs:

        goals: 'clean package'

    - task: Maven@3

      displayName: 'Testing Azure DevOps App'

      inputs:

        goals: 'clean test'

    - task: SonarCloudPrepare@1

      displayName: 'Prepare analysis on SonarCloud'

      inputs:

        SonarCloud: 'SonarCloud'

        organization: 'azuredevopsdevsecopsoorg'

        scannerMode: 'CLI'

        configMode: 'manual'

        cliProjectKey: 'azuredevopsadodevsecopsprojectkey'

        cliProjectName: 'AzureDevOpsApp'

        cliSources: '.'

    - task: Maven@3

      displayName: 'SAST Scan Azure DevOps App'

      inputs:

        goals: 'clean verify sonar:sonar'

        options: '-Dsonar.projectKey=azuredevopsadodevsecopsprojectkey -Dsonar.organization=azuredevopsdevsecopsoorg -Dsonar.host.url=https://sonarcloud.io/ -Dsonar.login=14ad47384add02744d3f9e34d'

    - task: SonarCloudPublish@1

      displayName: 'Publish quality gate result'

      inputs:

        pollingTimeoutSec: '300'

-------------------------------------------------------------------------------------

trigger:

- master

pool:

  vmImage: ubuntu-latest

steps:

- script: |

    mvn clean package

  displayName: "Building Azure DevOps App"

- script: |

    mvn clean test

  displayName: "Testing Azure DevOps App"

- script: |

    mvn verify package sonar:sonar -Dsonar.host.url=https://sonarcloud.io/ -Dsonar.organization=azuredevopsdevsecopsoorg -Dsonar.projectKey=azuredevopsadodevsecopsprojectkey -Dsonar.login=14ad47384add02744d3f9e34d

  displayName: "SAST Scan Azure DevOps App"

## SonarCloud Integration

## Snyk Integration

Steps to integrate Snyk using azure-pipeline.yaml file:

1. Add Snyk Plugin to Pom.xml
2. Add Snyk Token as a Pipeline Variable
3. Add code changes to azure-pipeline.yaml file

**mvn snyk:test -fn**

flag is used to avoid any build failure in case Snyk identifies security issues within third party.

**Question-1:**

Write a YAML file that defines an Azure DevOps DevSecOps pipeline with three steps: "build", "test" and "sca". The pipeline should be triggered whenever there is a code change in "master" branch. The code should be run on "ubuntu-latest" machine. Also, Snyk should fail the build if security vulnerabilities are identified.

1) The "build" step should be called "Building Azure DevOps App", which runs the script "mvn clean package".

2) The "test" step should be called "Testing Azure DevOps App", which runs the script "mvn clean test".

3) The "sca" step should be called "SCA Scan Azure DevOps App". For running SCA scan, SNYK\_TOKEN is "5464-4564654-7775411-8858"

trigger:

  branches:

    include:

      - master

pool:

  vmImage: 'ubuntu-latest'

jobs:

- job: build

  displayName: 'Building, Testing, and SCA Scan Azure DevOps App'

  steps:

    - task: Maven@3

      displayName: 'Building Azure DevOps App'

      inputs:

        goals: 'clean package'

    - task: Maven@3

      displayName: 'Testing Azure DevOps App'

      inputs:

        goals: 'clean test'

    - script: |

        echo "Running SCA scan with Snyk"

        snyk test --severity-threshold=high

      displayName: 'SCA Scan Azure DevOps App'

      env:

        SNYK\_TOKEN: '5464-4564654-7775411-8858'

      continueOnError: false

-------------------------------------------------------------------------------

trigger:

- master

pool:

  vmImage: ubuntu-latest

steps:

- script: |

    mvn clean package

  displayName: "Building Azure DevOps App"

- script: |

    mvn clean test

  displayName: "Testing Azure DevOps App"

- script: |

    SNYK\_TOKEN='5464-4564654-7775411-8858'

    export SNYK\_TOKEN

    mvn snyk:test

  displayName: "SCA Scan Azure DevOps App"

ZAP Security report will be saved in Azure under **AzureStagingDirectory**

Steps to create End-To-End DevSecOps Pipeline for a Java Project using Azure DevOps:

1. Update azure-pipeline.yaml with scripts for SAST, SCA, and DAST steps.
2. Add Pipeline Variables with security tokens for SonarCloud and Snyk.
3. Update pom.xml file with SonarCloud and Snyk Plugins.

## MODULAR ADO DEVSECOPS PIPELINE

The modularization concept is related to parent child Yaml files concept that helps to make the overall code maintainable and reusable.

There will be a parent Yaml file that will have references to multiple child Yaml files, and these child Yaml files will have different configurations.

So, you can have a single parent Yaml file calling multiple child Yaml files, and each child Yaml file can have a different job, task or stage.

**Advantages:**

1. **Reusability**: Child YAML files can be reused across multiple pipelines.
2. **Readability**: Pipeline configurations are organized into smaller, focused files, improving readability and maintainability.
3. **Parameterization**: Parameters allow customization of child YAML files based on the specific requirements of each pipeline.

# AWS DevSecOps

## Tools for implementing DevSecOps in AWS

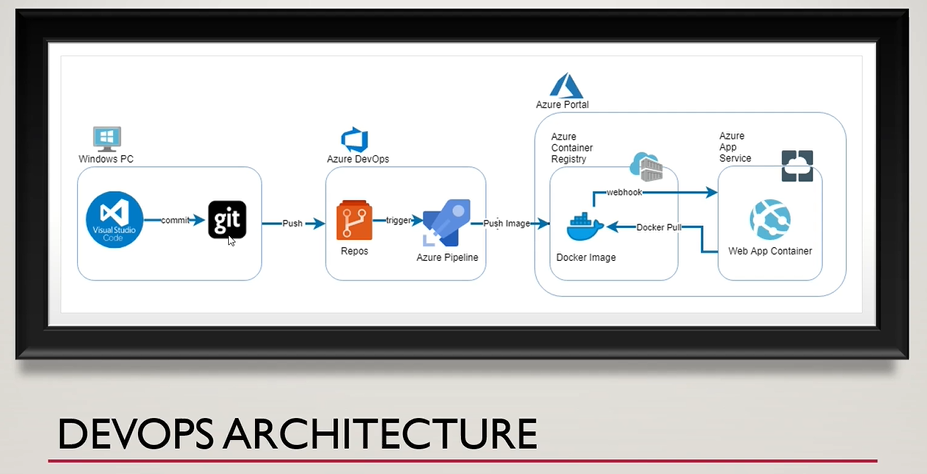


## Bridgecrew by Palo Alto/Prisma Cloud

Automated Iac Security Scanning & Code Fixes

# Azure DevOps





## Azure DevOps Platform







## Automated Testing

* A **unit test** is usually used to test a unit of source code, or a piece of source code, to test functionality of that if acted as expected.
* **User interface testing** is a process used to test if the application is functioning correctly according to your user or business requirements.
* In **code quality testing**, a well known phrase is called code smells. Code smells is to check the quality of code for any possible defects. Code quality need to be conformed to a development best practices, or development standards. Just one example of a best practice is maybe adding comment to your code, the way you declare your variables, or global variables, the way you declare global variables, the use of IF statements.
* It is imperative to evaluate the **security risks** in the software system in order to reduce the probability of a threat. Also, test code for the probability of a threat from being hacked by hackers. Two well known mechanisms are used for security vulnerability testing and it is called vulnerability assessments and penetration testing.
* **Performance testing** is used to test the overall performance of your application and how the user will experience using your system.
* The idea of **integration testing** is to start testing only two modules that connects to each other. Test if they are integrated together and test to make sure that the data is being communicated exactly as expected. For example, you can test connectivity from your user interface to a web service, or APIs, or you can test the integration between your web APIs and your database, or either your user interface and your database.



## Securing CI/CD Pipeline

* CI/CD Pipeline is DevOps Engineer’s production environment
* User Groups to Allow or Deny
* Pre-deployment conditions to request manual approval
* Post-deployment conditions
* Pre-deployment gates
* Post-deployment gates

## LAB 1

### Create MVC Application

# Container Security & CSPM (Cloud security posture management)

**QUALYS, AQUA, TRIVY, SNYK**

* Azure Security Engineer
* AWS Developer Associate
* GCP ACE
* CEHv10
* SC-900
* DIAT

## CSPM (CLOUD SECURITY POSTURE MANAGEMENT)

**What is a CSPM?**

**Cloud Security Posture Management is the process of identifying misconfigurations and compliance issues within cloud.**

Compliance Framework:

1. HIPAA (Health Insurance Portability and Accountability Act) in USA in 2003
2. PCI DSS (Payment Card Industry Data Security Standard) – International in 2004
3. CIS (Center for Internet Security)
4. GDPR (General Data Protection Regulation) in European Union in 2018
5. ISO 27001 (International Organization for Standardization)

**Aqua, which implements cloud security posture management, and automatically identify such kind of misconfiguration and compliance issues within any cloud environment.**

## Qualys

**VMDR – Vulnerability Detection Management and Response**

Create account in Qualys using organization email ID.

TO connect any Cloud (AWS/Azure/GCP) use Total security.

Through IAM Role create permission and add the ARN to Qualys to connect to AWS account.

**Container**: A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

Commercial Container Security Tool: Aqua, Prisma Cloud. These tools can be integrated in CI/CD platforms.

Free or Open source Container Security tools are Snyk Community Edition, Trivy.

## Red Team vs. Blue Team in Cybersecurity

Red teams and blue teams work as attackers and defenders to improve an organization's security.

A **red team** plays the role of the attacker by trying to find vulnerabilities and break through cybersecurity defenses. A **blue team** defends against attacks and responds to incidents when they occur.

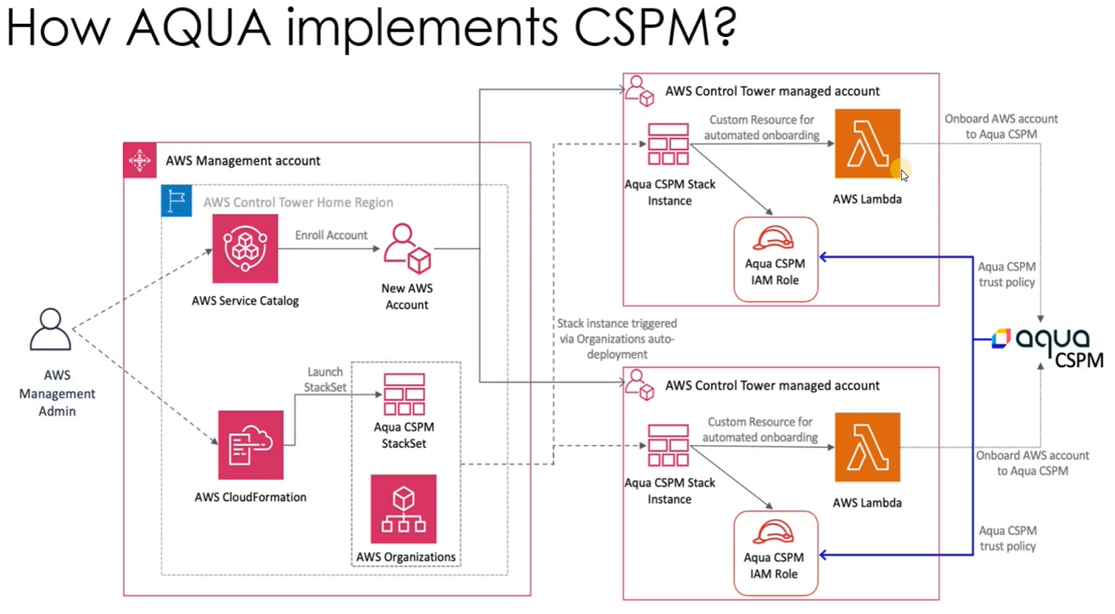
**Benefits of a red team vs. blue team approach**

One-way organizations can assess their security capabilities is to stage a red team/blue team exercise. These two teams of professionals face off to put a security infrastructure to the test in a simulation meant to mimic a real attack. Taking a red team versus blue team approach to cybersecurity can have several benefits, allowing security teams to:

* Find vulnerabilities
* Strengthen network security
* Build experience in detecting and containing attacks
* Develop response plans and procedures
* Create healthy competition and cooperation
* Raise security awareness among other staff

**AQUA CVSS Score (Common Vulnerability Scoring System)**

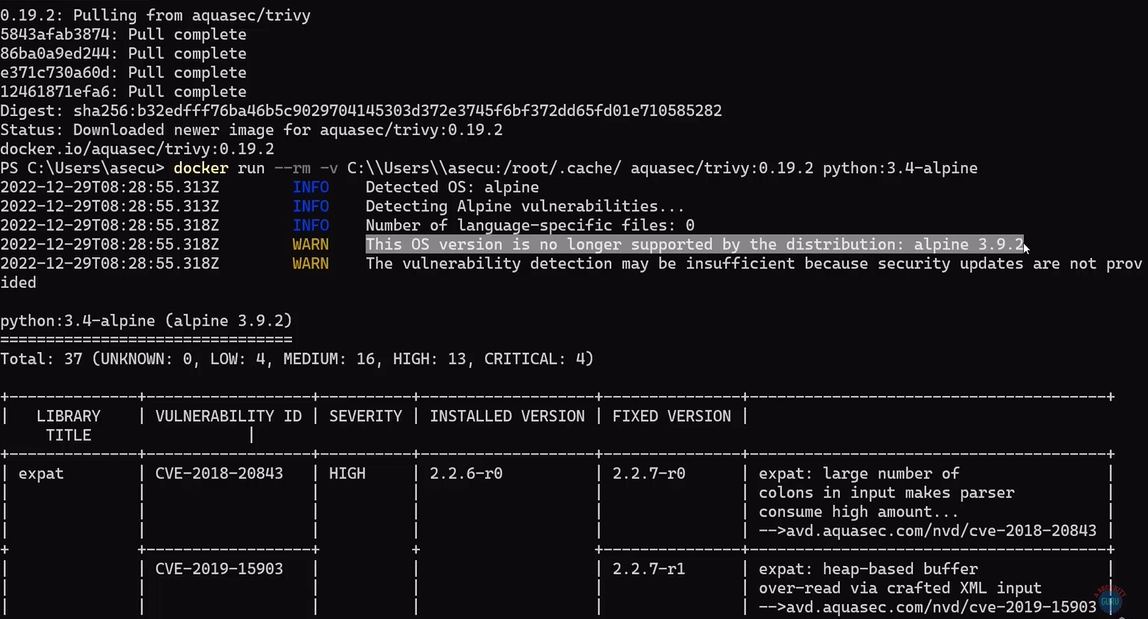
* Scores range from 0 to 10, with 10 being the most severe.



## Trivy from AQUA to Scan Container Image

* docker pull aquasec/trivy:0.19.2
* docker run –rm -v C:\\Users\\asecu:/root/.cache/ aquasec/trivy:0.19.2 python:3.4-alpine

**CVE : Common Vulnerabilities and Exposures**



## Docker Scan to perform Container Scan Using Snyk in the Backend

Docker scan is using snyk to identify any kind of security vulnerabilities in Docker files and container images.

Hence Snyk is providing the capabilities to *Docker Scan*.

>docker scan –accept-license - -version

Version: v0.19.0

Git Commit: 9c79702

Provider: Snyk (1.827.0 (standalone))

>docker scan python:3.4-alpine

## Snyk to perform Container Scan

>docker pull python:3.4-alpine

>snyk-win.exe container test python:3.4-alpine

## False Positive Analysis

* False positive analysis is the process of identifying and analyzing false positives in a system or process. A false positive, also known as a “false alarm”, is an error that occurs when a system or process wrongly identifies something as a threat or problem when it is not.
* In the context of cybersecurity, false positives refer to instances where a security system or process wrongly identifies a code as being vulnerable. For example, If SonarQube says that CSRF token is not implemented in the system but the token is implemented with some other name would lead to False Positive event generated by SonarQube security tool.
* False positives can be a problem because they can waste resources, cause unnecessary disruptions, and lead to a lack of trust in the system or process. They can also lead to “alert fatigue“, where users become desensitized to alerts and may ignore them even when a real threat is present.
* This involves improving the accuracy of the system by marking the incorrect security vulnerabilities in the system as False and then generating the report only with True Positives.

## Perform FPA for Container Security Scans

Write a declarative Jenkins file for a linux server to automate the container security scan process using Trivy and Snyk. The pipeline should build a Docker container image from the source code, perform security scans using Trivy and Snyk, and report the scan results.

**System Details:**

* Jenkins is installed on linux server with Trivy and Snyk CLI installed.
* Docker is installed on the linux server along with Jenkins server or Jenkins agent.
* Docker Pipeline(dockerpipeline) plugin is installed in Jenkins Snyk and Docker credentials are already declared in Jenkins Credential Manager as SNYK\_TOKEN and DOCKER\_LOGIN

**Questions for this assignment**

Your **Jenkinsfile** should include the following stages:

1. **Checkout**: Clone the source code repository (repo url: https://github.com/testrepo) from main branch containing the Dockerfile and application code.
2. **Build**: Build the Docker image(asecurityguru/testeb) using the Dockerfile. Pass the DOCKER\_LOGIN credential using withCredentials.
3. **Trivy Container Security Scan**: Perform a security scan on the built Docker image using Trivy and store results in trivy-report.json file
4. **Snyk Container Security Scan**: Perform a security scan on the Docker image using Snyk CLI and store results in snyk-report.json file. Pass the SNYK\_TOKEN credential using withCredentials.
5. **Publish Results**: Publish both Trivy and Snyk security scan results as build artifacts. Note: Snyk and Docker credentials are already declared in Jenkins Credential Manager as SNYK\_TOKEN and DOCKER\_LOGIN

pipeline {

    agent any

    environment {

        DOCKER\_IMAGE = 'asecurityguru/testeb'

    }

    stages {

        stage('Checkout') {

            steps {

                git url: 'https://github.com/testrepo', branch: 'main'

            }

        }

        stage('Build') {

            steps {

                script {

                    withCredentials([usernamePassword(credentialsId: 'DOCKER\_LOGIN', passwordVariable: 'DOCKER\_PASSWORD', usernameVariable: 'DOCKER\_USERNAME')]) {

                        sh """

                        echo "$DOCKER\_PASSWORD" | docker login -u "$DOCKER\_USERNAME" --password-stdin

                        docker build -t ${DOCKER\_IMAGE} .

                        """

                    }

                }

            }

        }

        stage('Trivy Container Security Scan') {

            steps {

                script {

                    sh """

                    trivy image --format json --output trivy-report.json ${DOCKER\_IMAGE}

                    """

                }

            }

        }

        stage('Snyk Container Security Scan') {

            steps {

                script {

                    withCredentials([string(credentialsId: 'SNYK\_TOKEN', variable: 'SNYK\_TOKEN')]) {

                        sh """

                        snyk container test ${DOCKER\_IMAGE} --json > snyk-report.json

                        """

                    }

                }

            }

        }

    }

    post {

        always {

            archiveArtifacts artifacts: 'trivy-report.json, snyk-report.json', allowEmptyArchive: true

        }

    }

}

## Q&A

**Sample Container Security Interview Questions and Answers**

1. **What is container security, and why is it important in the context of modern software development?**

   Answer: Container security focuses on securing containerized applications to prevent unauthorized access, data breaches, and other security threats. As software development increasingly relies on containerization, securing containers becomes crucial to ensure the integrity and confidentiality of applications and data.

2. **Explain how you would secure container images during the development process.**

   Answer: To secure container images during development, I would implement the following best practices:

   - Regularly update base images and dependencies to avoid known vulnerabilities.

   - Utilize security scanning tools like Trivy or Snyk to identify and fix vulnerabilities in the container image.

   - Apply least privilege principles and only include necessary libraries and packages.

   - Use strong authentication and encryption for private container registries.

3. **How do you handle secrets and sensitive data in containerized applications?**

   Answer: To manage secrets securely, I would use container orchestration platforms like Kubernetes to store secrets in a secure manner using Kubernetes secrets or external vaults. This ensures that sensitive data is not exposed in the container image or environment variables.

4. **What are the key security considerations when deploying containers in a production environment?**

   Answer: In a production environment, key security considerations include:

   - Implementing RBAC (Role-Based Access Control) to control access to containers and resources.

   - Enabling network segmentation and firewall rules to limit container communication.

   - Implementing container runtime security to protect against runtime threats.

   - Monitoring container logs and using intrusion detection systems.

5. **How do you ensure the security of containerized applications across multi-cloud environments?**

   Answer: To ensure security across multi-cloud environments, I would use infrastructure as code (IaC) tools like Terraform or CloudFormation to create consistent and secure container environments. Additionally, I'd leverage cloud-native security services and use identity and access management (IAM) to control access.

6. **What is Kubernetes Network Policies, and how do they enhance container security?**

   Answer: Kubernetes Network Policies are rules that control the communication between pods in a cluster. They enhance container security by limiting access between pods, thereby reducing the attack surface and preventing unauthorized communication.

7. **How do you manage and monitor container security at scale?**

   Answer: To manage and monitor container security at scale, I would use container security tools like Aqua CSPM to monitor for suspicious activities and enforce security policies. Additionally, I'd integrate security checks into CI/CD pipelines to ensure security at each stage of the development process.

8. **Explain how you would handle a security incident involving a compromised container image.**

   Answer: In the event of a compromised container image, I would immediately remove the image from production, initiate an incident response plan, and conduct a thorough investigation to understand the extent of the compromise. I'd then remediate the vulnerabilities and implement security improvements to prevent similar incidents in the future.

9. **How do you ensure compliance with industry standards and regulatory requirements for container security?**

   Answer: To ensure compliance, I would regularly audit container images and configurations against relevant security standards and regulatory requirements such as CIS benchmarks and GDPR. I would also implement security controls and monitor compliance continuously.

10. **What role does DevSecOps play in container security, and how do you collaborate with development and operations teams?**

    Answer: DevSecOps integrates security into the entire software development lifecycle. I collaborate with development and operations teams to ensure security is built into CI/CD pipelines, perform security code reviews, and provide security training. This fosters a culture of security awareness and proactive risk mitigation.

11. **How do you handle software supply chain security and prevent supply chain attacks in containerized environments?**

    Answer: To handle software supply chain security, I would validate the integrity of the container images by verifying their provenance, using signed images, and scanning for known vulnerabilities in third-party dependencies. Additionally, I'd implement container image signing and enforce container image promotion policies.

12. **Explain the concept of "Immutable Infrastructure" and its impact on container security.**

    Answer: Immutable infrastructure refers to the practice of treating infrastructure as unchangeable and replacing instances instead of updating them. Containers support this concept as they are immutable by nature. This approach reduces the risk of configuration drift and minimizes attack vectors by ensuring containers are consistently and securely deployed.

13. **What are the security challenges in running containers with root privileges, and how do you mitigate them?**

    Answer: Running containers with root privileges can lead to increased attack surface and potential privilege escalation attacks. To mitigate this, I would follow the principle of least privilege, ensure containers run with non-root users whenever possible, and leverage container security features like SELinux to restrict container capabilities.

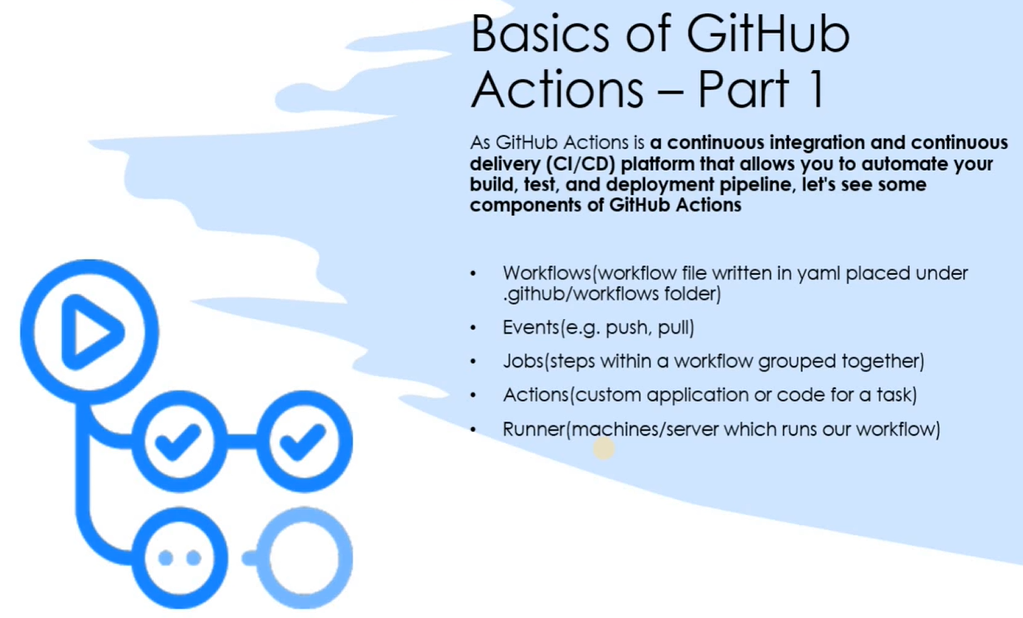
14. **How do you ensure the security of container images obtained from public registries like DockerHub?**

    Answer: To ensure the security of public container images, I would use security scanning tools like Trivy or Anchore to check for vulnerabilities in the images. Additionally, I'd consider using signed images and only use images from trusted sources.

15. **How do you stay updated on the latest container security trends and best practices?**

    Answer: I regularly participate in industry conferences, attend webinars, and read security blogs and whitepapers. I am an active member of container security communities and engage in knowledge-sharing to stay informed about the latest trends and best practices.

# GitHub Actions



## GIT Commands:

**Git Clone is used to create a copy of Repo on local system**​

* git clone​

**Git Fetch is used to only fetch the metadata/new branch changes but does not bring any remote code changes to local changes**​

* git fetch​

**Git Pull is used to fetch code changes from remote branch and merge it with local code**​

* git pull ​

**Git Stash is used to save local changes in a backup directory**​

* git stash ​
* git stash apply​

**Git checkout is used to change your current local branch to other branch** ​

* git checkout <branch name>​

**Example:**git checkout feature/testrepo ​

**Git merge is used to merge local code changes with remote branch changes all together**​

* git merge​

**Git rebase is used to merge local code changes with remote branch changes one by one**​

* git rebase -i <branch name>​

**Example:**git rebase -i origin/master and then git add filename and then git push –f (:x to save any file during rebase)​

**Git add is used to add change to a commit**​

* git add <filename1....filename2>​

**Example:**git add test1.java test2.java​

git add \*​

**Git commit –m "message" is used to pass a commit message**​

**Example:**git commit –m  "This is the first commit in this branch"​

**Git squash is used to merge multiple commits into single commit**​

* git squash​

**Git push is used to push local changes to remote branch**​

* git push​

# Miscellaneous

