

Pipeline



What is a Pipeline?

- A DevSecOps pipeline is a CI/CD pipeline that incorporates security practices and tooling at every stage of the software development lifecycle. It ensures security is integrated throughout the process, rather than being treated as a separate phase. This approach aims to deliver secure, high-quality software by automating security checks and embedding security as a core development practice.
- Continuous Integration (CI):
 - Integrates security checks into the code integration process, ensuring vulnerabilities are identified early.
- Continuous Delivery (CD):
 - Automates the delivery of secure software to different environments, including testing and production.
- Security Testing:
 - Includes automated security scans and testing at various stages, such as Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and penetration testing.
- Security Tools:
 - Utilizes security tools and processes to identify and address vulnerabilities, including code scanners, threat intelligence tools, and policy enforcement mechanisms.
- Security Governance:
 - Establishes security policies, procedures, and compliance requirements to ensure applications meet security standards

What is a Pipeline?

- Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating *continuous delivery pipelines* into Jenkins.
- A *continuous delivery (CD) pipeline* is an automated expression of your process for getting software from version control right through to your users and customers. Every change to your software (committed in source control) goes through a complex process on its way to being released. This process involves building the software in a reliable and repeatable manner, as well as progressing the built software (called a "build") through multiple stages of testing and deployment.
- <https://www.jenkins.io/doc/book/pipeline/>

What is a Pipeline

CI/CD stands for continuous integration/continuous deployment/delivery. While CI automatically releases every change to production, CD ensures changes are production-ready but may require a manual step for deployment. It is a software development practice that involves frequently integrating code changes into a central repository and then automatically building and deploying those changes to a test or production environment. CI is the practice of regularly merging code changes into a shared repository, where automated tests are run to catch any issues that may have been introduced. CD is the practice of automatically deploying code changes that pass all tests to a production environment. A CI/CD pipeline automates the process of testing, building, and deploying software changes.

- Sehgal, Vandana Verma. *Implementing DevSecOps Practices: Understand application security testing and secure coding by integrating SAST and DAST*. Packt Publishing.

Pipelines

In the world of software, a CI/CD pipeline can include these steps: Fetching the latest code: Grabbing the latest “photos” (code changes). Building: Turning code into a runnable application. This could involve setting the page layout. A developer writes and commits code to a version control system (such as Git). Deployment: Making the new changes live and showing off that scrapbook page. Testing: Checking if the new changes “fit” well and don’t “spoil” anything, as well as making sure that the photo isn’t upside down! If tests pass, the CD process starts. Depending on the setup, the code might either be automatically deployed to a production environment (continuous deployment) or might be moved to a staging environment, where it awaits its final approval (continuous delivery).

- Sehgal, Vandana Verma. Implementing DevSecOps Practices: Understand application security testing and secure coding by integrating SAST and DAST. Packt Publishing.

Pipelines

The Role of Secure Pipelines Secure pipelines form the backbone of DevSecOps, enabling automated and consistent security checks at every stage of software delivery. A secure pipeline includes:
Automated code scanning for vulnerabilities during commits.
Integration of compliance checks before deployment. Continuous monitoring of applications in production for emerging threats. This automation

- Learning, Maxwell. Advanced DevSecOps: A Guide to Advanced DevSecOps Practices .

Pipelines

Another key benefit of automating security in CI/CD pipelines is the reduction of manual intervention. Traditional security practices often rely heavily on manual reviews, audits, and checks, which are not only time-intensive but also prone to human error. By automating repetitive tasks such as code scanning, dependency analysis, and compliance checks, teams can ensure consistent and accurate security enforcement across the pipeline. Automation also frees up security professionals to focus on more strategic and complex challenges, such as threat modeling and incident response, thereby optimizing resource utilization.



- Learning, Maxwell. Advanced DevSecOps: A Guide to Advanced DevSecOps Practices . Kindle Edition.

DevSecOps Platform

A DevSecOps Platform is defined as a multi-tenet environment that brings together a significant portion of a software supply chain, operating under cATO or a provisional ATO (Authority to Operation, conditional Authority to Operation). The components of a DevSecOps platform can be instantiated in many ways, and each will include a mixture of options. Each reference design's unique platform configurations must be clearly defined across each of these three distinct layers: **Infrastructure, Platform/Software Factory, Applications.**

- -DoD Enterprise DevSecOps Fundamentals

DevSecOps Platform

The *Infrastructure Layer* supplies the hosting environment for the Platform/Software Factory layer, explicitly providing compute, storage, network resources, and additional CSP managed services to enable function, cybersecurity, and non-functional capabilities. Typically, this is either an approved or DoD provisionally authorized environment provided by a Cloud Service Provider (CSP), but is not limited to a CSP.

- -DoD Enterprise DevSecOps Fundamentals



DevSecOps Platform


The *Platform/Software Factory Layer* includes the distinct development environments of the software factory, its CI/CD pipelines, a clearly implemented log aggregation and analysis strategy, and continuous monitoring operations. In between each of the architectural constructs within this layer is a Reference Design Interconnect. This layer should support multi-tenancy, enforce separation of duties for privileged users, and be considered part of the cyber survivability supply chain of the final software artifacts produced.

- -DoD Enterprise DevSecOps Fundamentals
-

DevSecOps Platform

The *Application Layer* includes application frameworks, data stores such as relational or NoSQL databases and object stores, and other middleware unique to the application and outside the realm of the CI/CD pipeline.

- -DoD Enterprise DevSecOps Fundamentals



Continuous Integration/Continuous Deployment (CI/CD)

Setting up a Continuous Integration/Continuous Deployment (CI/CD) pipeline on AWS with IaC and DevSecOps involves several steps and AWS services. Here is a high-level overview of the process:

- **Source Code Repository (AWS CodeCommit):** First, we need a repository for our source code. AWS CodeCommit is a fully managed source control service that allows teams to host secure and highly scalable Git repositories.
- **Build (AWS CodeBuild):** After the source code is in the repository, the next step is to compile the code, run tests, and create deployable artifacts. AWS CodeBuild is a fully managed build service that compiles our source code, runs tests, and produces software packages ready to deploy.
- **Security analysis:** As part of DevSecOps, we should integrate security checks at every stage of our pipeline. For example, we could use tools like AWS CodeStar, AWS CodeCommit, or third-party tools to scan our code for vulnerabilities during the build stage.
- **Deploy (AWS CodeDeploy):** AWS CodeDeploy automates code deployments to any instance, including Amazon EC2 instances and servers running on-premises. AWS CodeDeploy makes it easier for us to rapidly release new features, helps avoid downtime during application deployment, and handles the complexity of updating our applications.
- **Pipeline orchestration (AWS CodePipeline):** AWS CodePipeline is a fully managed continuous delivery service that helps us automate our release pipelines for fast and reliable application and infrastructure updates.

GitLab Pipeline

Configuration:

- Pipelines are defined in a YAML file named `.gitlab-ci.yml` located in the root of your project's Git repository. This file outlines the entire CI/CD workflow.

Components:

- **Jobs:** These are the individual tasks within a pipeline, such as compiling code, running tests, building Docker images, or deploying to a server. Each job includes a script that specifies the commands to be executed.
- **Stages:** Stages define the order of execution for jobs. Jobs within the same stage can run in parallel, while stages typically run sequentially (e.g., build -> test -> deploy).
- **Runners:** These are the agents or servers that execute the jobs defined in your pipeline. GitLab provides shared runners, or you can configure your own specific runners.
- <https://docs.gitlab.com/ci/pipelines/>



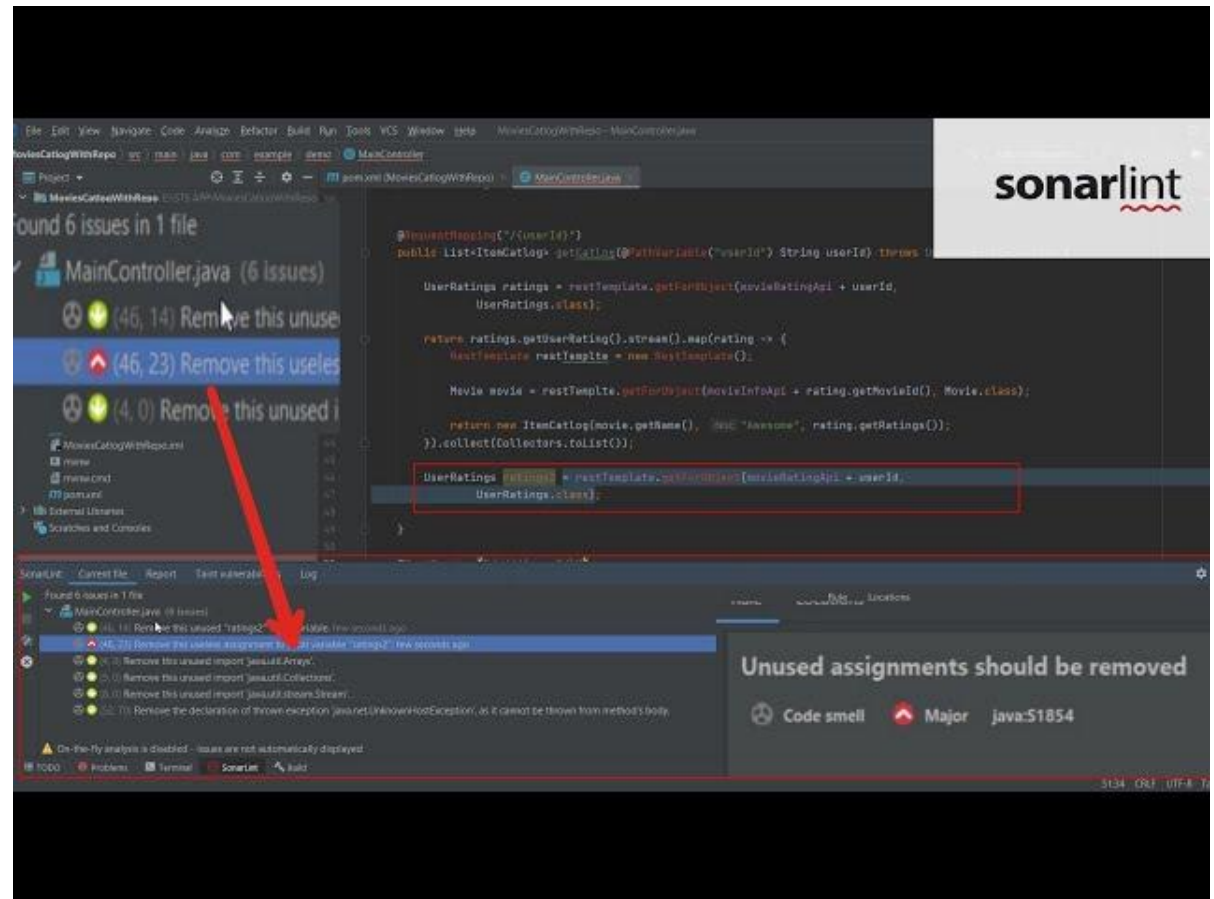
GitLab Pipeline

Pipelines are composed of:

- Global YAML keywords that control the overall behavior of the project's pipelines.
- Jobs that execute commands to accomplish a task. For example, a job could compile, test, or deploy code. Jobs run independently from each other, and are executed by runners.
- Stages, which define how to group jobs together. Stages run in sequence, while the jobs in a stage run in parallel. For example, an early stage could have jobs that lint and compile code, while later stages could have jobs that test and deploy code. If all jobs in a stage succeed, the pipeline moves on to the next stage. If any job in a stage fails, the next stage is not (usually) executed and the pipeline ends early.

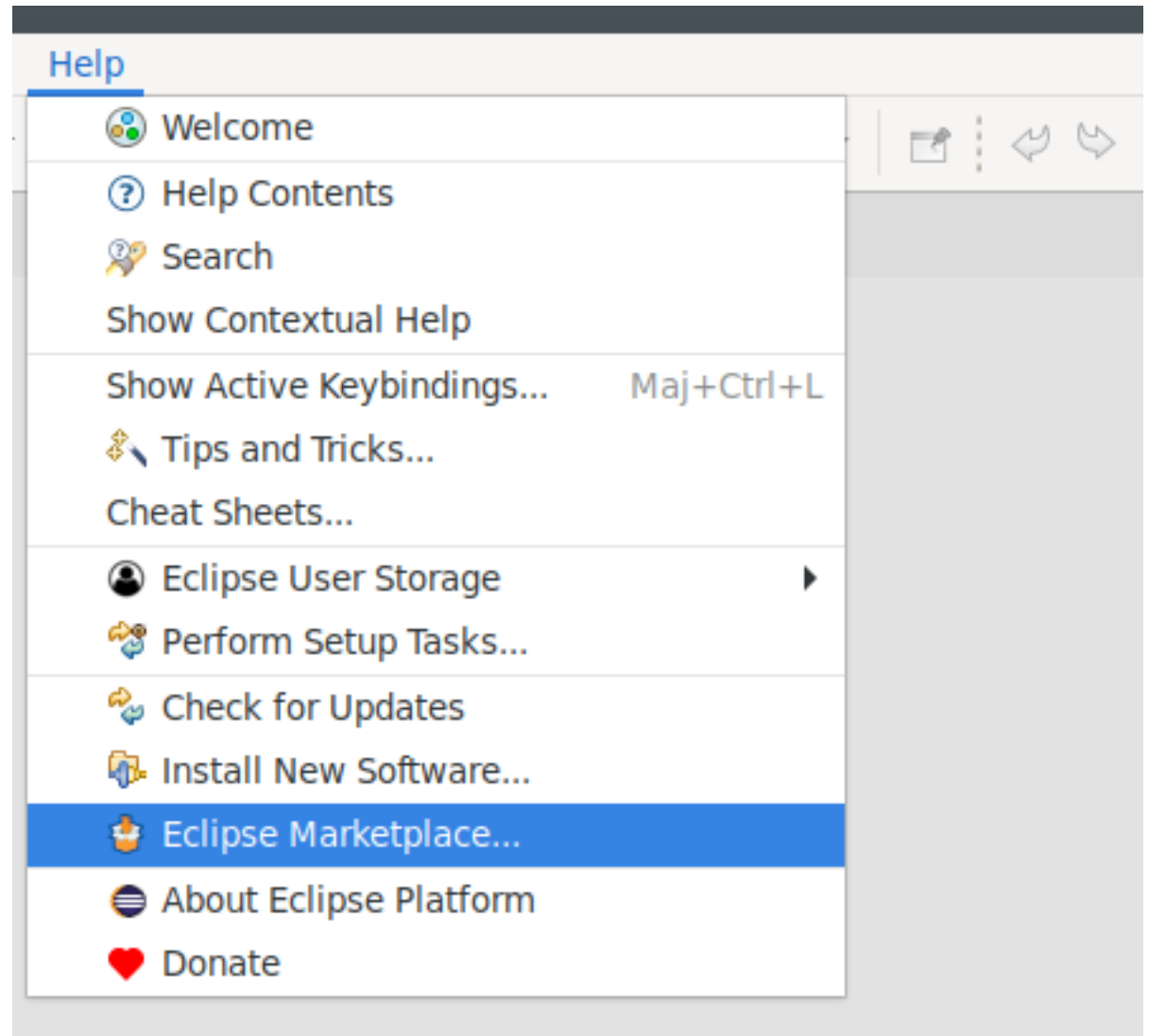
Security Plugin SonarLint

- SonarLint is an **open-source IDE extension** that detects and remediates security issues during code development
- It provides **instant feedback and recommendations** to mitigate the security issues, which helps the developer in producing robust software products
- It highlights all the **impacted locations** in the application code to help developers in finding and fixing the security issues with ease
- SonarLint detects common mistakes, tricky bugs, and known vulnerabilities
- It can be integrated with Eclipse, IntelliJ, Visual Studio, and VS Code
- Some of the major languages it can analyse are Java, JavaScript, TypeScript, Php, C#, Python, Ruby, HTML, VB.net



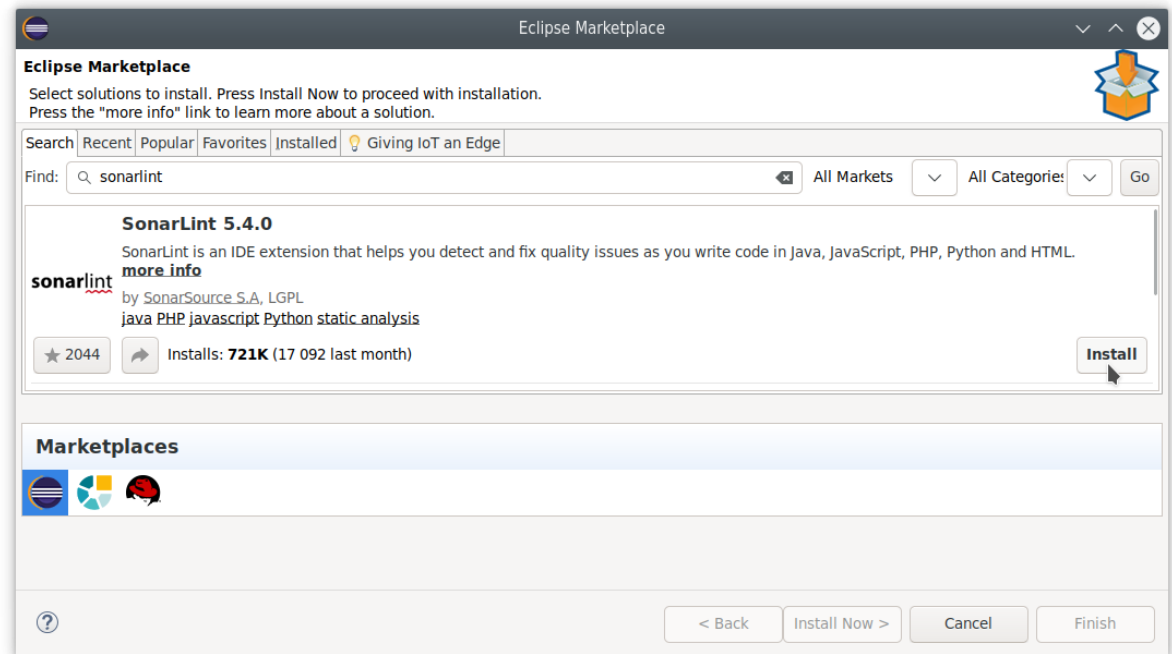
Integrate SonarLint in Eclipse IDE

- SonarQube for Eclipse is a plugin that can be installed in most Eclipse-based IDEs (including Spring Tool Suite, PyDev, and others). You can find the SonarLint for Eclipse IDE extension directly on the [Eclipse Marketplace](#)



Integrate SonarLint in Eclipse IDE

- Code Quality is an integral part of any software pipeline nowadays. It's about preventing bugs from impacting end users, preventing security vulnerabilities from making it to the open world, and also easing the maintainability of your code. Static Code Analysis plays an essential role here.
- Static code analysis typically happens as part of a Continuous Integration (CI) pipeline. All standard CI engines (e.g. Jenkins, Travis CI, Azure DevOps etc.) allow for many different build/test/analysis tools to be part of the pipeline. But that means the code must be committed into the repository and submitted to the CI server before it can be analysed.
- At SonarSource, we've been writing code analyzers for more than a decade. And along the journey of offering CI-friendly tools (SonarQube and SonarCloud, enabling Continuous Code Quality across more than 25 languages), we rapidly wondered: what if we could provide code quality feedback to developers earlier in the process? We envisioned a spell-checker type tool that would instantaneously report quality issues when you write code! That's how SonarLint was born.



Integrate SonarLint in VS Code IDE

SonarQube for IDE is a **Free** and **Open Source** IDE extension that identifies and helps you fix Code Quality and Code Security issues as you code. Analogous to a spell checker, SonarQube for IDE squiggles flaws and provides real-time feedback and clear remediation guidance so you can deliver clean code from the get-go.

- SonarQube for IDE: Visual Studio is more than your average linting tool.
- Scans code written in C#, VB.NET, C, C++, Javascript, TypeScript.
- Open source JavaScript, TypeScript, C# & VB.NET code analyzers.
- C and C++ support for Application, Dynamic Library and Static Library types of MSBuild (.vcxproj) projects
- Deep code analysis algorithms using pattern matching and dataflow analysis
- Hundreds of language-specific static code analysis rules, and growing
- In-context help and remediation guidance with detailed examples
- Highlights issues in your code, tells you why they are harmful, and how they should be fixed
- SonarQube for IDE provides Visual Studio developers a comprehensive in-IDE solution for improving the quality and security of the code they deliver.

• <https://marketplace.visualstudio.com/items?itemName=SonarSource.SonarLintforVisualStudio2022>

JFrog

- JFrog IDE integrations allows the developer to **detect security** and **compliance issues** in the source code. Using the plugins or extension, you can scan the project dependencies and view security issues during SDLC
- Currently, JFrog can be integrated with four IDE environments, and it works in all four environments with same functionalities:
 - **Visual Studio Code:** In this IDE, JFrog scans Python, Maven, npm, and Go project dependencies
 - **IntelliJ:** In this IDE, JFrog scans Go, Gradle, Maven, npm and Pypi project dependencies; in addition, it can support PyCharm, GoLand, WebStorm, and Android Studio
 - **Eclipse:** In this IDE, JFrog scans Gradle, npm, and Maven project dependencies
 - **Visual Studio:** In this IDE, JFrog scans the NuGet project dependencies

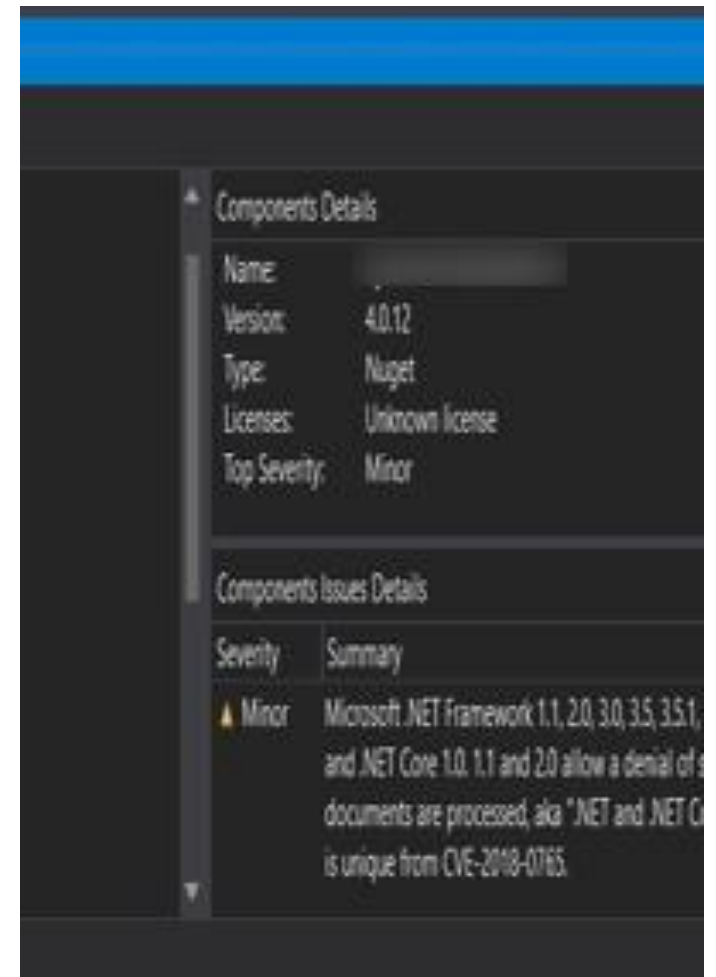
JFrog Eclipse IDE Plugin

- JFrog Eclipse IDE plugin will add JFrog Xray scans of Gradle, npm, and Maven project dependencies in your Eclipse IDE environment
- With the integration of JFrog Eclipse IDE plugin, the developer can view the vulnerabilities information related to the component and its dependencies directly in the IDE environment
- This will help the developer deciding whether to utilize the component before incorporating it into the software product
- **Steps for Installation and Configuration of JFrog Eclipse IDE Plugin**
 - Install JFrog plugin
 - If JFrog is behind the HTTP proxy, then set up the proxy settings
 - For connecting to JFrog Xray, set up the plugin
 - Next, perform scan and view the results
 - Then, **filter** the Xray scanned results



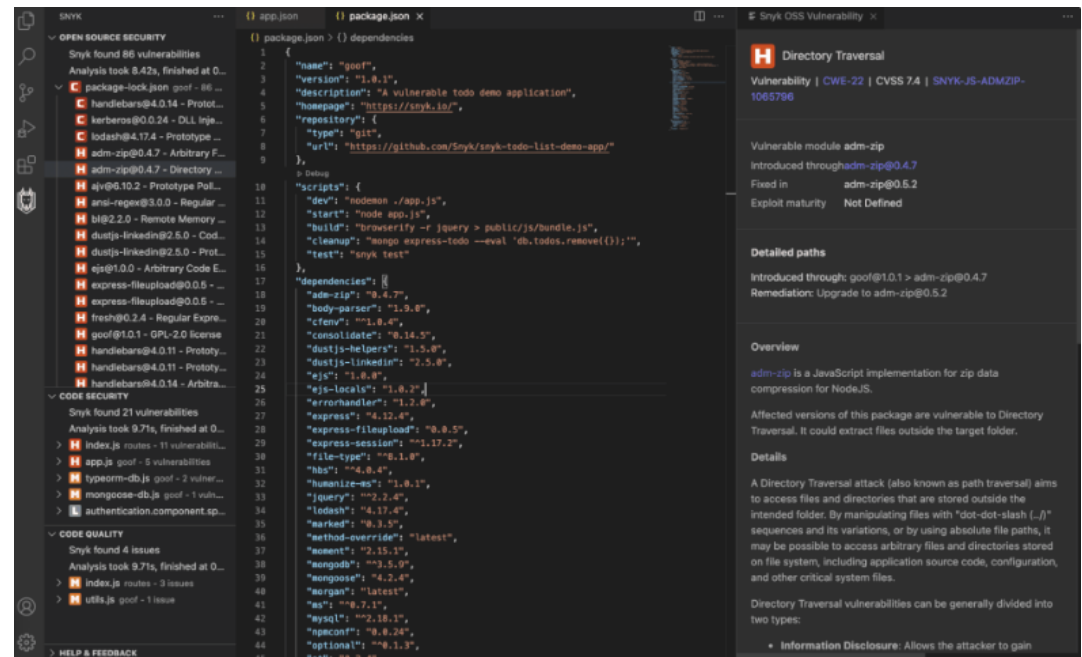
JFrog Visual Studio Extension

- Developer can integrate Visual Studio IDE with JFrog tools using JFRog Visual Studio extension. This extension will help the developer in gaining critical insights regarding the security issues detected by JFrog Xray in the NuGet packages
- Enter **nuget** command in the terminal
- In case, it is not recognized as command, go to **PATH** environment variable and please add **nuget.exe**
- Then, open **VS**
- Navigate to **Tools** and then go to **Extension and Updates**; then search **Jfrog** and download it
- After successfully completing the installation, restart Visual Studio



Snyk Security Plugin for Visual Studio Code IDE

- The Snyk Visual Studio Code plugin scans and provides analysis of your code, including open-source dependencies and infrastructure as code configurations

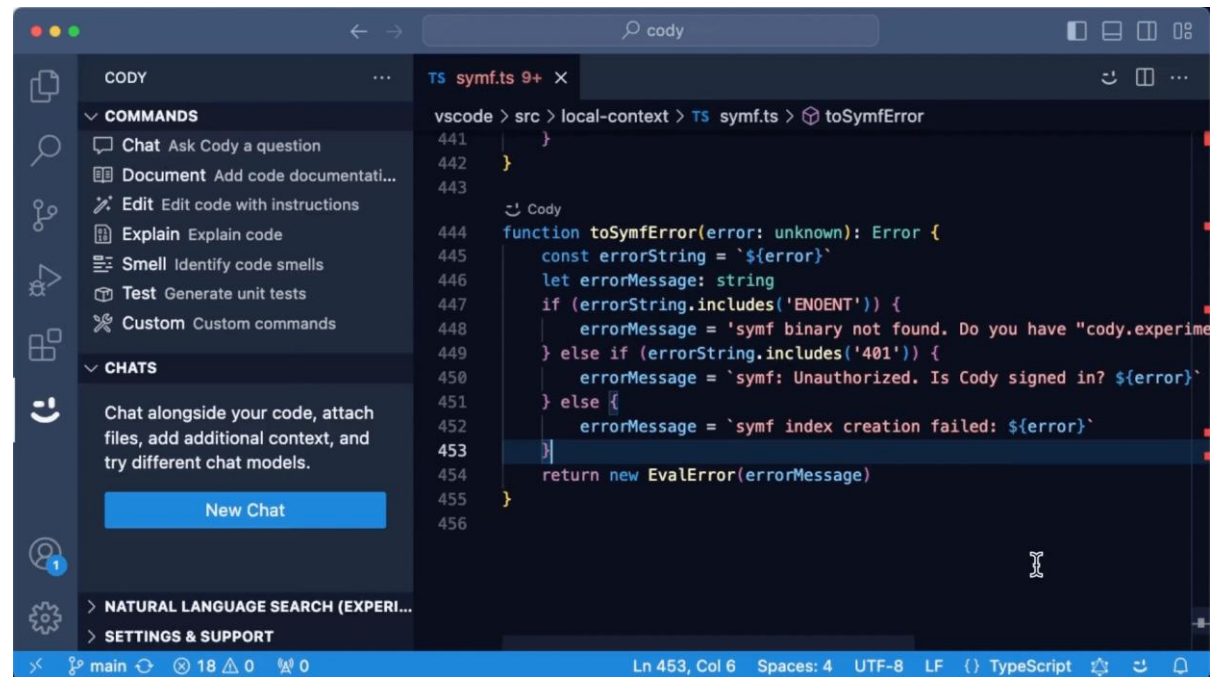


Microsoft Code Analysis for Visual Studio IDE

- Microsoft Code Analysis Extension will provide live analysis of the code while you are developing the code. Using it, you can quickly remediate the identified security issues in the code
- The extension offers diagnostics for security, performance, API design, etc
- The diagnostics will appear in the error list, editor, and scroll bar

Cody AI Coding Assistant for Visual Studio IDE

- Integrate **Cody AI** with **Visual Studio IDE** to streamline the CI/CD pipeline by providing real-time code suggestions and automated refactoring
- **Cody AI Coding Assistant** for Visual Studio IDE enhances the CI/CD workflow with contextual code recommendations, error detection, and continuous integration support

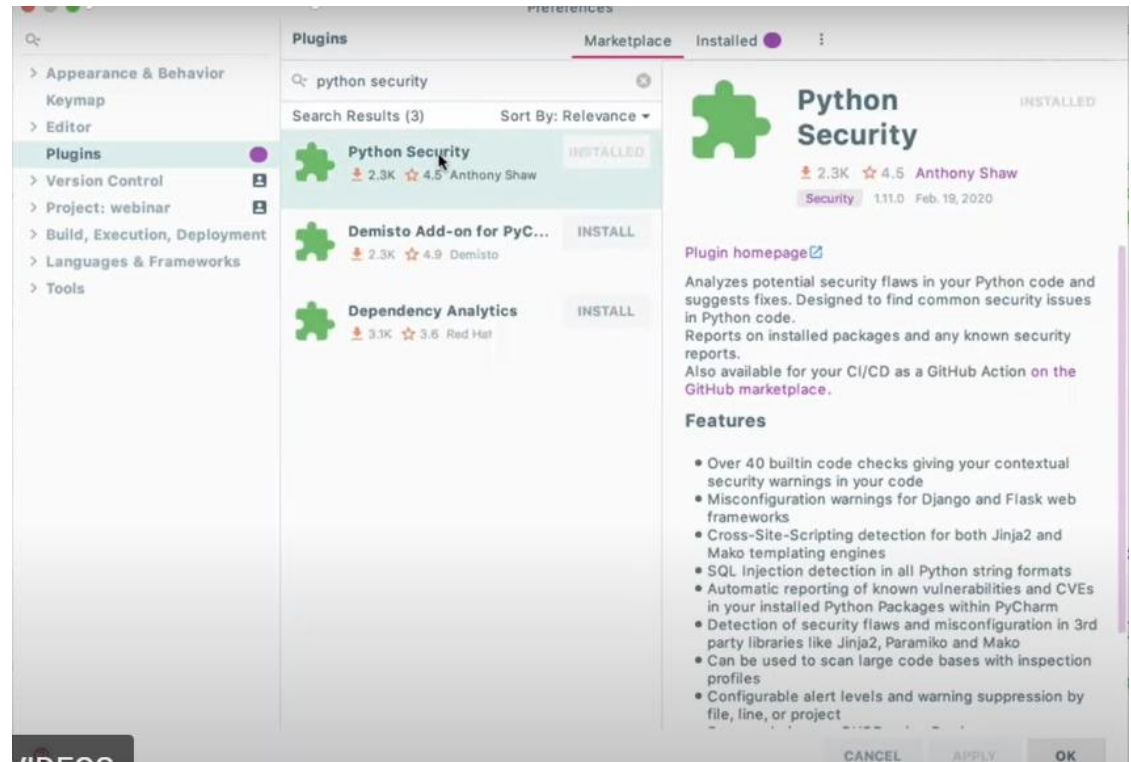


```
vscode > src > local-context > TS symf.ts > toSymfError
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456

function toSymfError(error: unknown): Error {
  const errorString = `${error}`
  let errorMessage: string
  if (errorString.includes('ENOENT')) {
    errorMessage = 'symf binary not found. Do you have "cody.experim
  } else if (errorString.includes('401')) {
    errorMessage = `symf: Unauthorized. Is Cody signed in? ${error}`
  } else {
    errorMessage = `symf index creation failed: ${error}`
  }
  return new EvalError(errorMessage)
}
```

Use PyCharm Security Plugin to Secure Python Application Code in PyCharm IDE

- PyCharm security plugin will identify the vulnerabilities in python application code and provide recommendations on how to remediate them
- To install **PyCharm security plugin** in **PyCharm IDE**, go to **preferences** and then navigate and click on **Plugins**; then type **Python security** in the search box
- Select **Python security**, click on **install**, and then click on **OK** button; restart **PyCharm IDE**



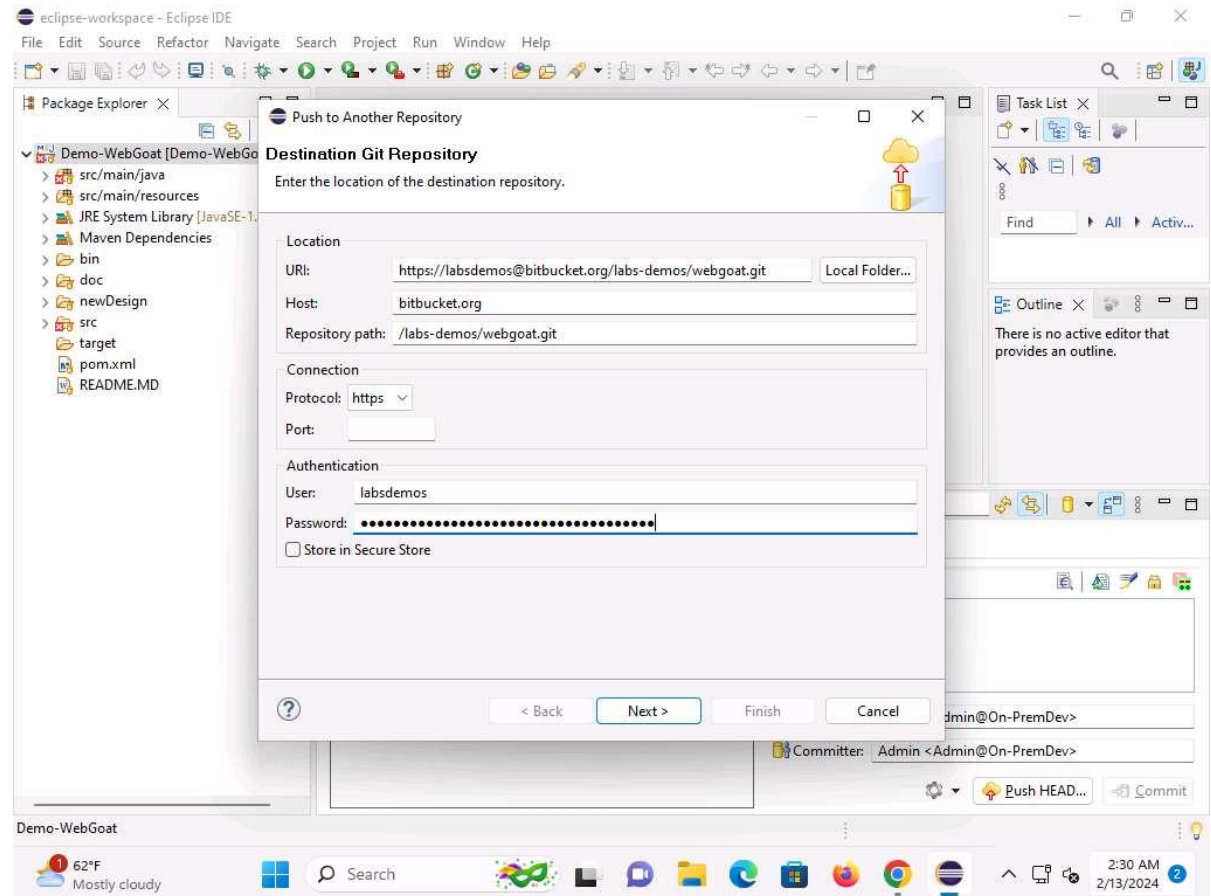
Setting up Code Scanning in GitHub

- Code scanning allows the DevSecOps teams to evaluate the source code within GitHub repository to identify security issues and coding errors
- Using code scanning, the team can **find, triage, and prioritize** the remediation process for security issues in their code; apart from this, code scanning stops developers from adding new issues to the code
- In GitHub repository, code scanning can be set up either using CodeQL or third-party tools
 - **CodeQL:** It is developed by GitHub and it automates security checks
 - **Third-party Code Scanning Tool:** These tools provide **Static Analysis Results Interchange Format (SARIF)** output for code scanning



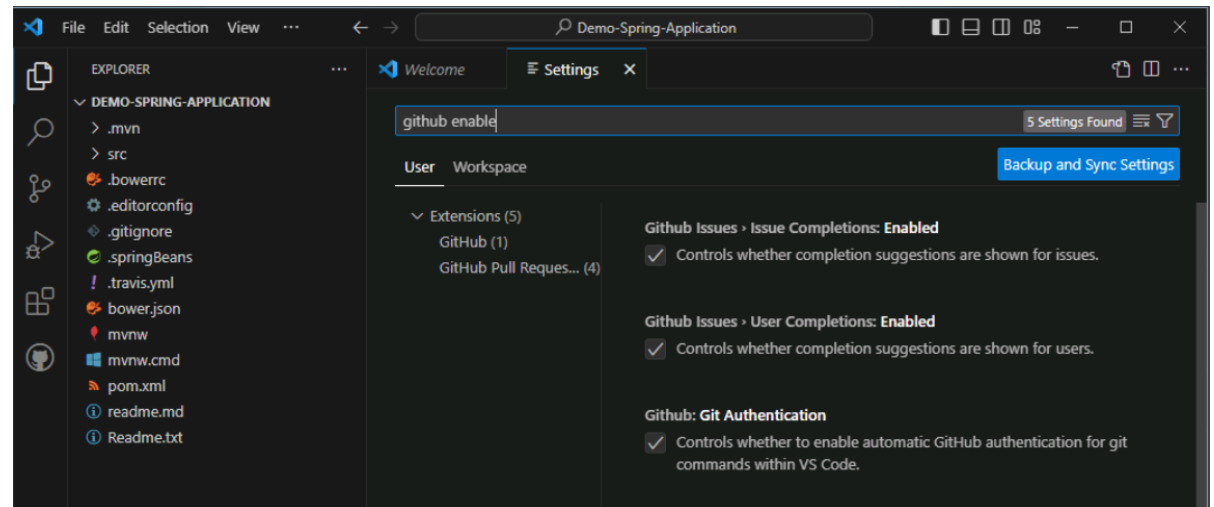
Integrating Bitbucket Repository with Eclipse IDE

- Bitbucket is a **distributed version control system** that offers free unlimited **private repositories**, and it can be easily integrated with third-party tools to enhance the **code quality**
- It provides the teams single place for projects' planning and collaboration on coding, testing, and deploying the software application
- Bitbucket's integration with Eclipse IDE makes code sharing easier
- To integrate Bitbucket repository with Eclipse IDE, **create an account** in Bitbucket and **link the Eclipse IDE** with Bitbucket remote repository; then **commit the application code** from Eclipse IDE to local repository, and **push the code** to Bitbucket remote repository



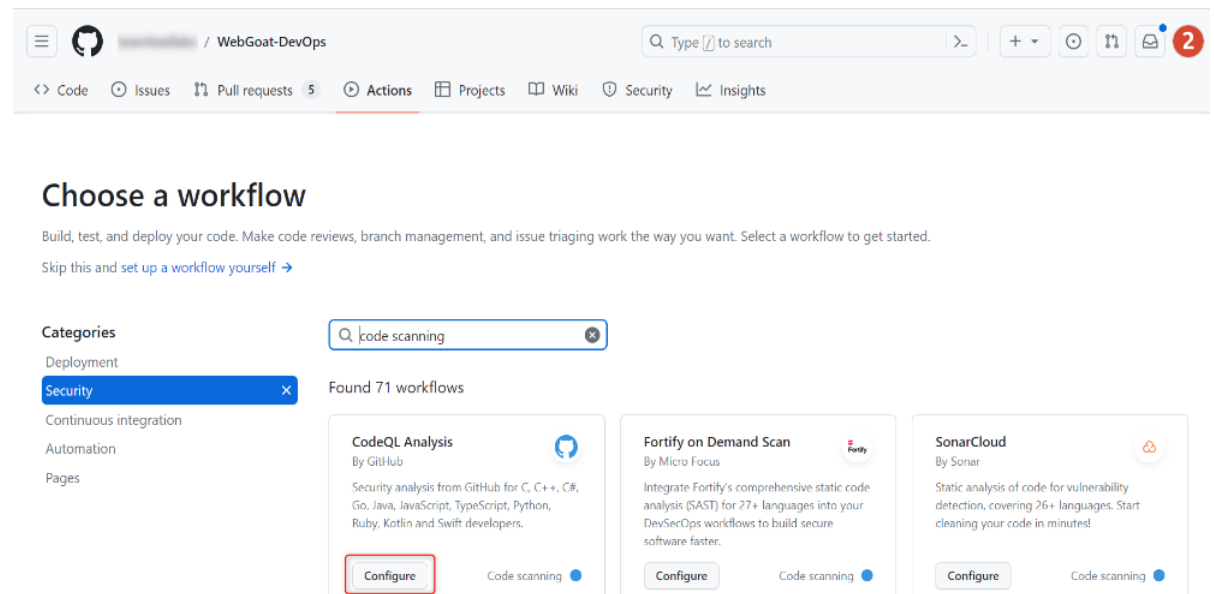
Integrating GitHub Repository with Visual Studio Code

- Git is an open-source version control system that allows several developers to simultaneously make changes to web pages
- GitHub is a web-based interface that utilizes Git and allows developers to work on one project simultaneously, minimizes the risk of conflicting or duplicative work, and helping in reducing production time
- Using Git from within VS Code can improve the efficiency and reliability of the workflow
- To integrate GitHub repository with Visual Studio code, install **git** on your system; then, create an account on GitHub repository, install **VS code** and ensure that **Git Enabled** is checked under **Settings**



Scan GitHub Repository using CodeQL

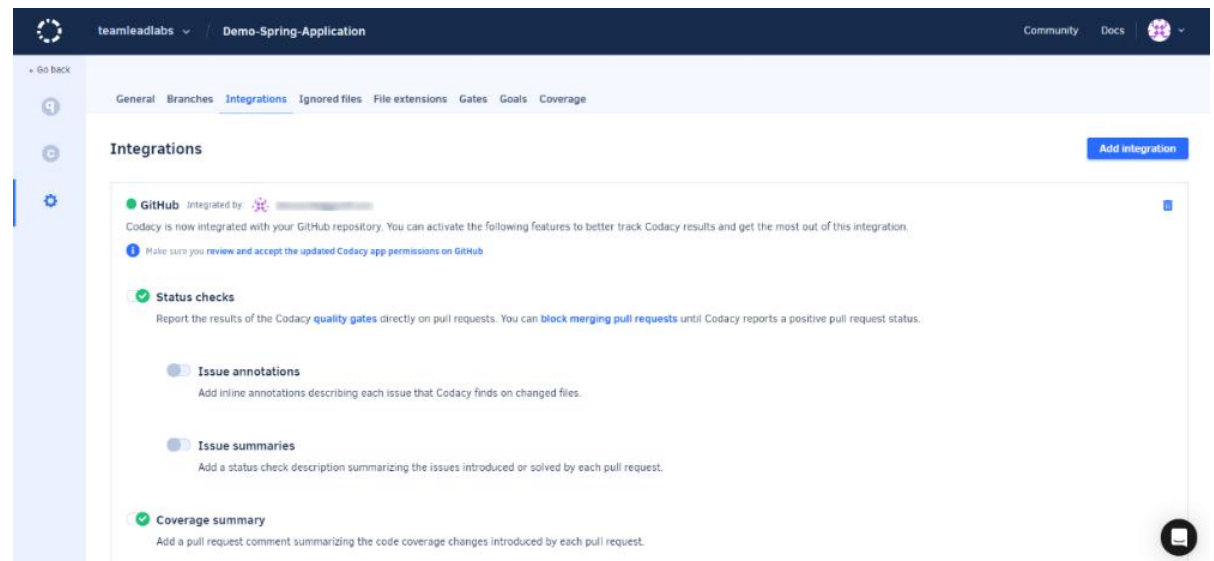
- The CodeQL code scanning feature of GitHub repository allows the DevSecOps team to detect security vulnerabilities and coding errors in application code
- The CodeQL analysis workflow can be configured to scan code on a schedule or when certain events in a repository occur
- Access CodeQL from the **Security** tab in the GitHub repository and set up the CodeQL workflow



The screenshot shows the GitHub repository page for 'WebGoat-DevOps'. The navigation bar includes 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', and 'Insights'. The 'Security' tab is active. Below the navigation bar, the 'Choose a workflow' section is displayed. It includes a search bar with 'code scanning' entered, showing 'Found 71 workflows'. Three workflow cards are visible: 'CodeQL Analysis' (By GitHub), 'Fortify on Demand Scan' (By Micro Focus), and 'SonarCloud' (By Sonar). The 'CodeQL Analysis' card has a red box around its 'Configure' button. The 'Code scanning' status is shown as active (blue dot) for all three workflows.

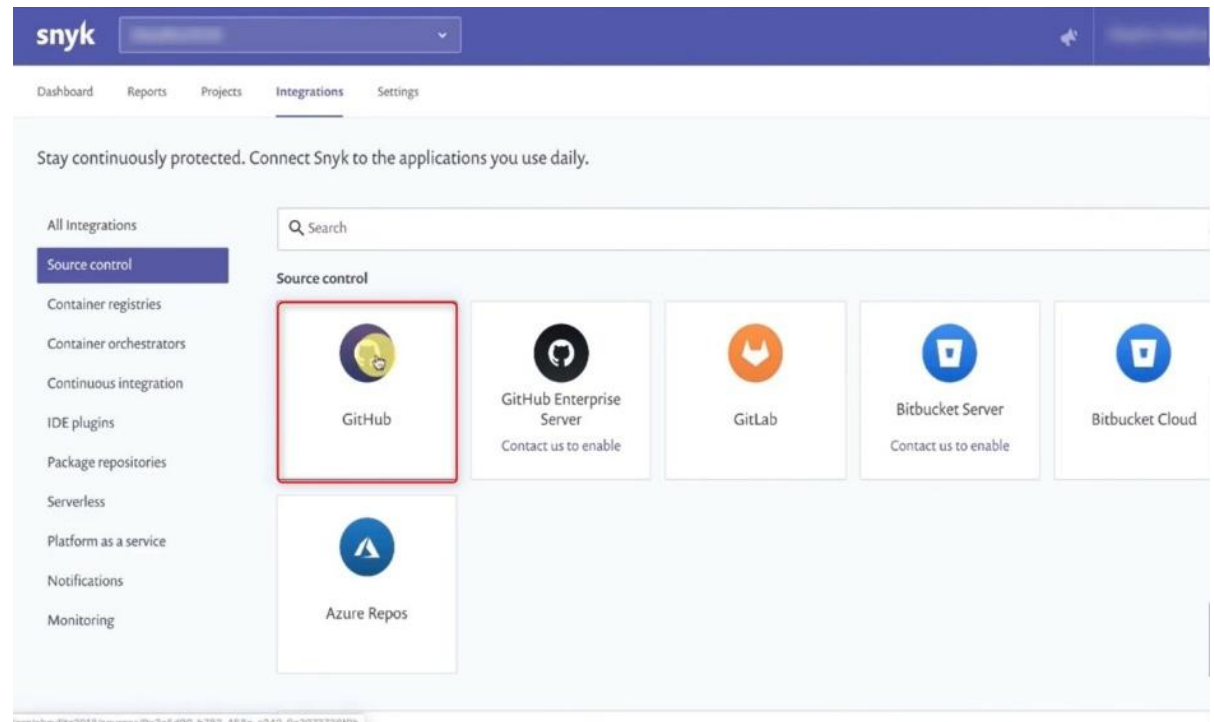
Integrate and Scan GitHub Repository using Codacy

- Codacy is an automatic code analysis/quality tool that assists DevSecOps teams in delivering robust software quickly
- To establish the GitHub integration, log in to **Codacy**, select your organization, and navigate to **Settings**; click the **Integrations tab**, choose **GitHub**, and click on **Add integration**
- Go to **Repositories** and click on the **Manage repositories** button to add a new GitHub repository for scanning
- Scanning the project stored in the GitHub repository using Codacy allows the DevSecOps team to identify the vulnerabilities in the code, duplicated code, code coverage, etc



Integrate and Scan GitHub Repository using Snyk

- Snyk checks and monitors the applications' code for security flaws
- The DevSecOps teams can integrate GitHub with Snyk to perform continuous security scanning throughout all integrated repositories, detect vulnerabilities in the open-source components, and deliver automated solutions
- Login to Snyk Account and navigate to the **Integrations** page and select **GitHub**, and import the repositories in Snyk to start the scan



Fortify Static Code Analyzer Integration with Jenkins

- Integrating Micro Focus Fortify Static Code Analyzer in Jenkins allows the developer to
 - Analyze the source code for security issues
 - Send the results to Micro Focus Fortify Software Security Center
 - Display analysis results in a summary
 - Based on analysis results, **set build failure criteria**

Jenkins

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Project
- Configure
- Rename
- Fortify Assessment (FreeStyle Java-RC1)

List of Fortify SSC issues

Summary

Build	Total	Critical	High	Medium	Low
#49 (#48)	174 (1170)	166 (203)	8 (98)	0 (11)	0 (858)

Issues breakdown by Priority Order

Critical (1 to 50 out of 166) High (0) Medium (0) Low (0) All (174)

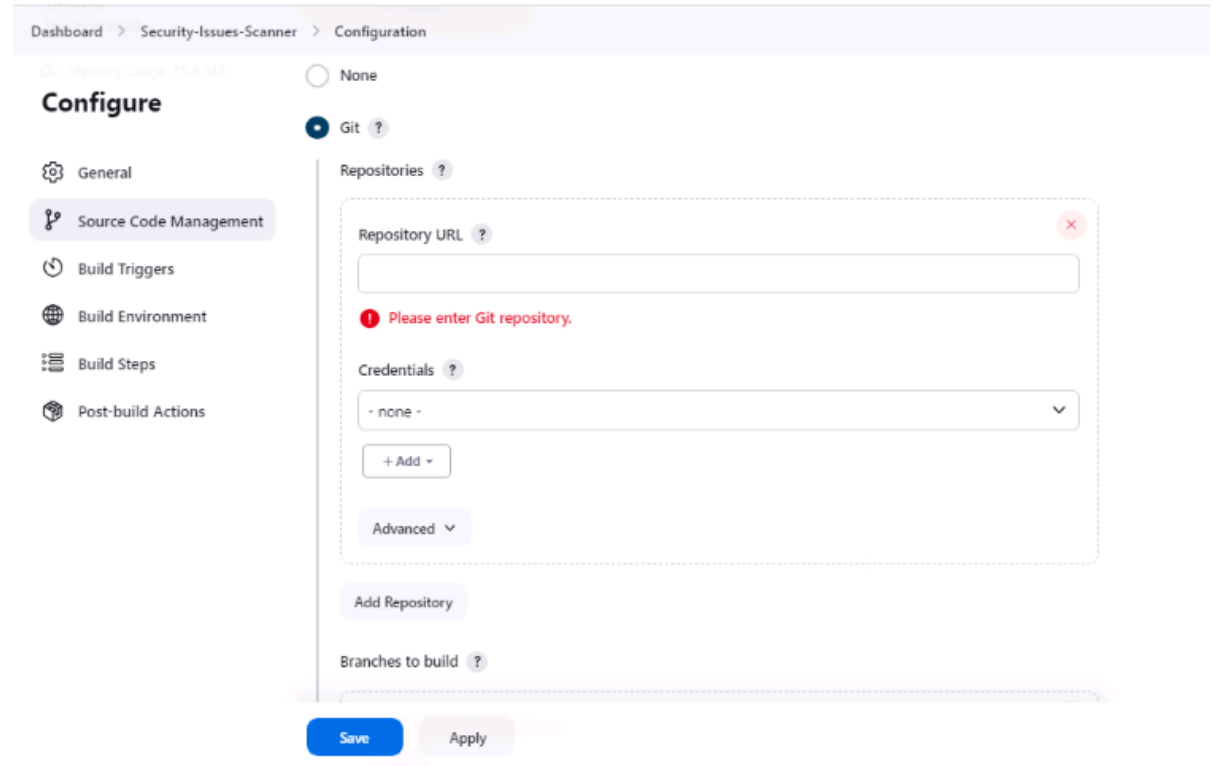
Primary Location	Category
Exec.java_292	Command Injection
Exec.java_103	Command Injection
WSDLScanning.java_143	Command Injection
WSDLScanning.java_221	Cross-Site Scripting Persistent
JavaScriptValidation.java_152	Cross-Site Scripting Reflected
ReflectedXSS.java_150	Cross-Site Scripting Reflected
JavaScriptValidation.java_154	Cross-Site Scripting Reflected
ReportCardScreen.java_295	Cross-Site Scripting Reflected
Encoding.java_369	Cross-Site Scripting Reflected
ReflectedXSS.java_206	Cross-Site Scripting Reflected
WeakSessionID.java_262	Cross-Site Scripting Reflected

Build History

Build	Time
#49	Feb 21, 2019 11:28 PM
#48	Feb 21, 2019 11:15 PM
#47	Feb 21, 2019 7:44 PM
#46	Feb 21, 2019 5:25 AM
#45	Feb 21, 2019 3:38 AM

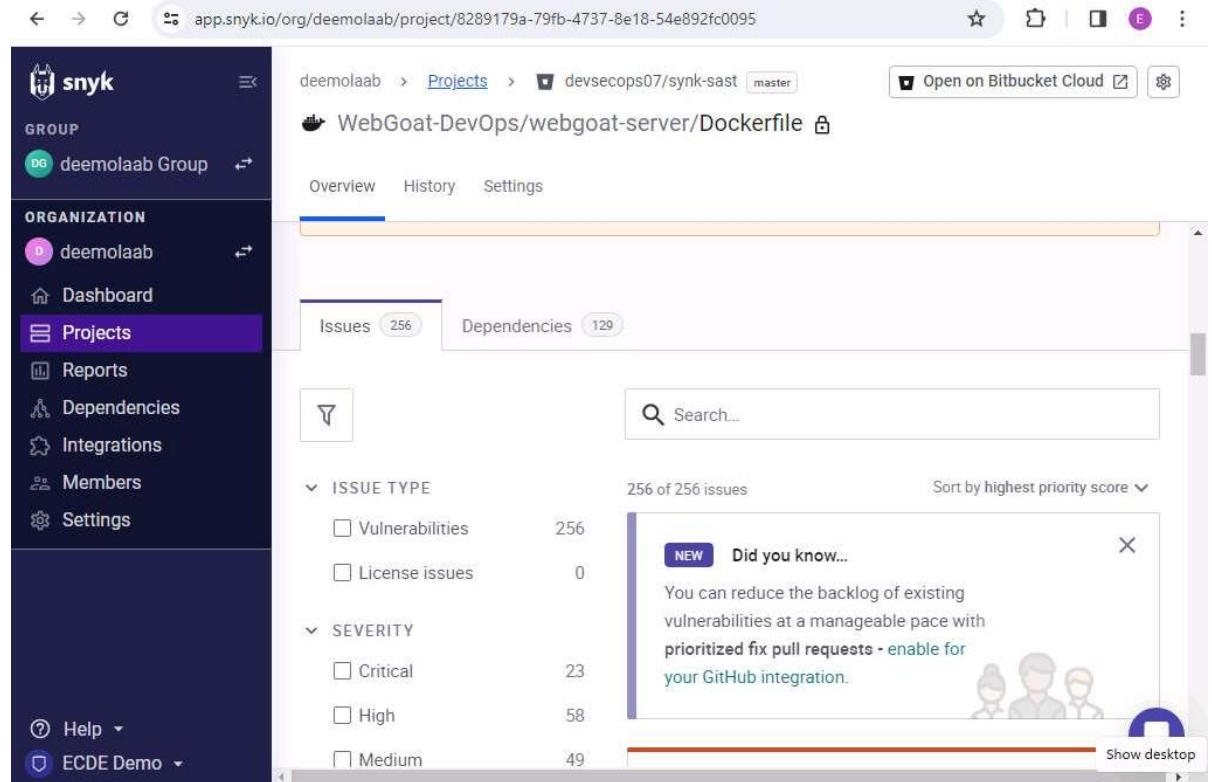
Bandit SAST Tool Integration with Jenkins

- Bandit is a SAST tool that scans for the **common security issues** in the source code, such as security misconfigurations, sensitive data exposure in python code, etc
- Install Bandit and create a Jenkins job. In the **Source Code Management** section, provide your repository URL, credentials, and the branch you want to analyze
- Integrating Bandit with Jenkins helps the DevSecOps teams to detect common security vulnerabilities in Python code
- As Bandit is installed on Jenkins server, select **Execute shell** for the **Build step** and **configure the Bandit script**



Snyk SAST Tool Integration with Bitbucket

- **Snyk** is a security scanner that performs static application security testing of the source code repository
- **Snyk** can be integrated with **Bitbucket** Cloud to perform continuous scanning of the source code repositories for vulnerabilities



The screenshot displays the Snyk web interface for a project named 'WebGoat-DevOps/webgoat-server/Dockerfile'. The interface includes a navigation sidebar on the left with options like 'Dashboard', 'Projects', 'Reports', 'Dependencies', 'Integrations', 'Members', and 'Settings'. The main content area shows a search bar, a filter icon, and a table of issues. The table is filtered to show 256 issues and 129 dependencies. The issues are categorized by type and severity.

ISSUE TYPE	Count
<input type="checkbox"/> Vulnerabilities	256
<input type="checkbox"/> License issues	0

SEVERITY	Count
<input type="checkbox"/> Critical	23
<input type="checkbox"/> High	58
<input type="checkbox"/> Medium	49

A notification banner at the bottom right of the interface reads: "NEW Did you know... You can reduce the backlog of existing vulnerabilities at a manageable pace with prioritized fix pull requests - enable for your GitHub integration." A "Show desktop" button is visible in the bottom right corner.

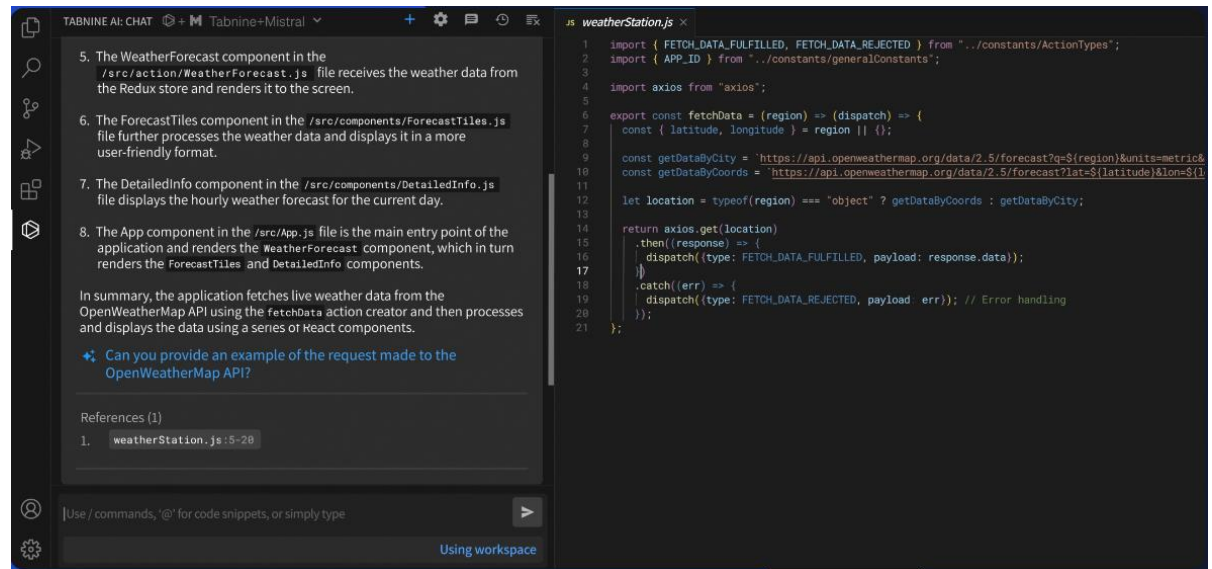
Kiuwan Plugin for Jenkins

- Integrate Kiuwan with Jenkins to run static analysis as a post-build activity or as a pipeline step for detecting security vulnerabilities in the source code

The screenshot shows the Jenkins Dashboard interface. At the top, there is a breadcrumb 'Dashboard >'. Below this is a navigation menu with the following items: '+ New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. Below the navigation menu are three sections: 'Build Queue' (with a dropdown arrow and the text 'No builds in the queue.'), 'Build Executor Status' (with a dropdown arrow), and 'Welcome to Jenkins!' (with a sub-header 'Start building your software project'). The 'Welcome to Jenkins!' section contains a 'Create a job' button with a plus sign and a 'Set up a distributed build' section with a 'Set up an agent' button and a speech bubble icon. There is also an 'Add description' link in the top right corner of the main content area.

AI-based Secure Code Review Tool: Tabnine

- Tabnine is an artificial intelligence tool designed to assist developers in writing and debugging code
- It offers support for wide range of programming languages and integrates with popular development environments VS Code, Eclipse, Visual Studio 2022, and JetBrains IDEs
- It analyzes a developer's coding patterns to generate tailored code suggestions and also offers various code completion options



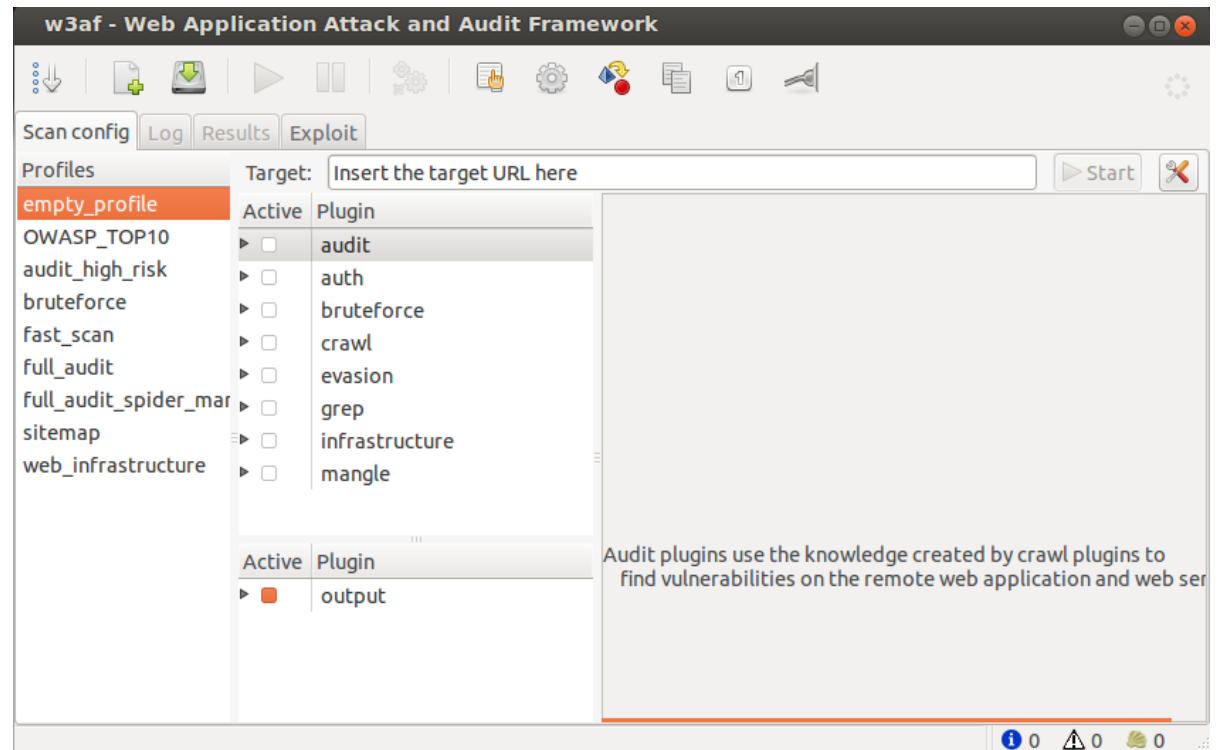
The screenshot displays the Tabnine AI Chat interface. On the left, a list of steps describes the application's data flow: 5. WeatherForecast component receives data from Redux; 6. ForecastTiles component processes and displays data; 7. DetailedInfo component displays hourly forecast; 8. App component is the main entry point. Below this is a summary and a question: "Can you provide an example of the request made to the OpenWeatherMap API?". A reference to "weatherStation.js:5-28" is shown. On the right, the code for "weatherStation.js" is displayed, showing imports for actionTypes and axios, and functions for fetching data by city and coordinates, with error handling for rejected data.

```
1 import { FETCH_DATA_FULFILLED, FETCH_DATA_REJECTED } from "../constants/ActionTypes";
2 import { APP_ID } from "../constants/generalConstants";
3
4 import axios from "axios";
5
6 export const fetchData = (region) => (dispatch) => {
7   const { latitude, longitude } = region || {};
8
9   const getDataByCity = `https://api.openweathermap.org/data/2.5/forecast?q=${region}&units=metric&appid=${APP_ID}`;
10  const getDataByCoords = `https://api.openweathermap.org/data/2.5/forecast?lat=${latitude}&lon=${longitude}&units=metric&appid=${APP_ID}`;
11
12  let location = typeof(region) === "object" ? getDataByCoords : getDataByCity;
13
14  return axios.get(location)
15    .then((response) => {
16      dispatch({ type: FETCH_DATA_FULFILLED, payload: response.data });
17    })
18    .catch((err) => {
19      dispatch({ type: FETCH_DATA_REJECTED, payload: err }); // Error handling
20    });
21  };
```


DAST

Tools: w3af

- w3af is uses a common web application attack and audit framework
- It creates a framework that secures web applications by finding and mitigating web application vulnerabilities



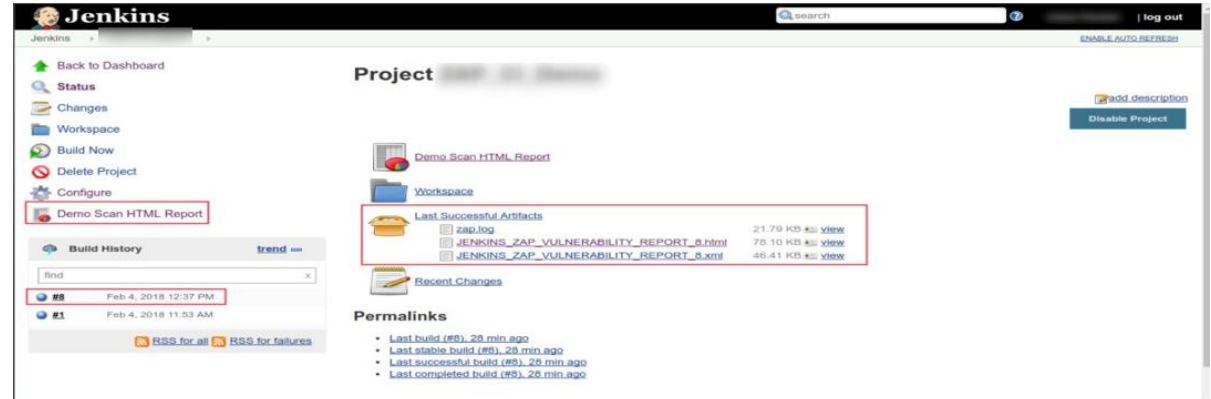
DAST Tool: Codename SCNR

- Codename SCNR offers a user-friendly web interface for easy execution, management, and scheduling of security scans and their results
- It includes features for efficient web application security assessments, such as parallel scanning for faster audits, recurring scans to continuously track vulnerabilities, and intuitive filtering for focused analysis

The screenshot shows the Codename SCNR web interface. The top navigation bar includes 'Sites', 'Profiles', 'Devices', and 'Settings'. The main header displays the URL 'http://testhtml5.vulnweb.com' and the scan status 'Scanning'. Below this, there are tabs for 'Summary', 'Issues', 'Coverage', 'Health', 'Configuration', and 'Events'. The 'Summary' tab is active, showing 'Excellent health' for '11 pages' and '8 issues, with a maximum severity of Medium'. A summary of issues is provided: 'There are 2 medium, 3 low, and 3 informational severity issues.' The scan duration is '00:00:33'. A list of scanned pages is shown, including '/ #/about', '/ #/ajax/latest?offset=0', '/ #/ajax/popular?offset=0', '/ #/archive', '/ #/calendar', '/ #/contact', '/ #/latest', '/ #/popular', and '/ #/popular/page/1'. The 'Issues' section on the right lists 8 issues: 2 medium severity (Common directory at /samples/ and Unencrypted password form in /login), 3 low severity (Missing 'Access-Control-Allow-Origin' header, Missing 'X-Frame-Options' header, and Password field with auto-complete), and 3 informational severity (Allowed HTTP methods, interesting response, and interesting response).

OWASP ZAP DAST Plugin Integration with Jenkins

Integrating OWASP ZAP with Jenkins allows the DevSecOps teams to automatically detect **security loopholes** in web applications while building and testing them



The screenshot displays the Jenkins web interface for a project. The left sidebar contains navigation options: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, and Demo Scan HTML Report. The main content area shows the project name, a workspace, and a list of artifacts under 'Last Successful Artifacts'. The artifacts are:

Artifact Name	Size	Action
zap.log	21.79 KB	view
JENKINS_ZAP_VULNERABILITY_REPORT_S.html	78.10 KB	view
JENKINS_ZAP_VULNERABILITY_REPORT_S.xml	46.41 KB	view

Below the artifacts, there are 'Recent Changes' and 'Permalinks' sections. The 'Permalinks' section includes links for the last build, last stable build, last successful build, and last completed build, all from 28 minutes ago.

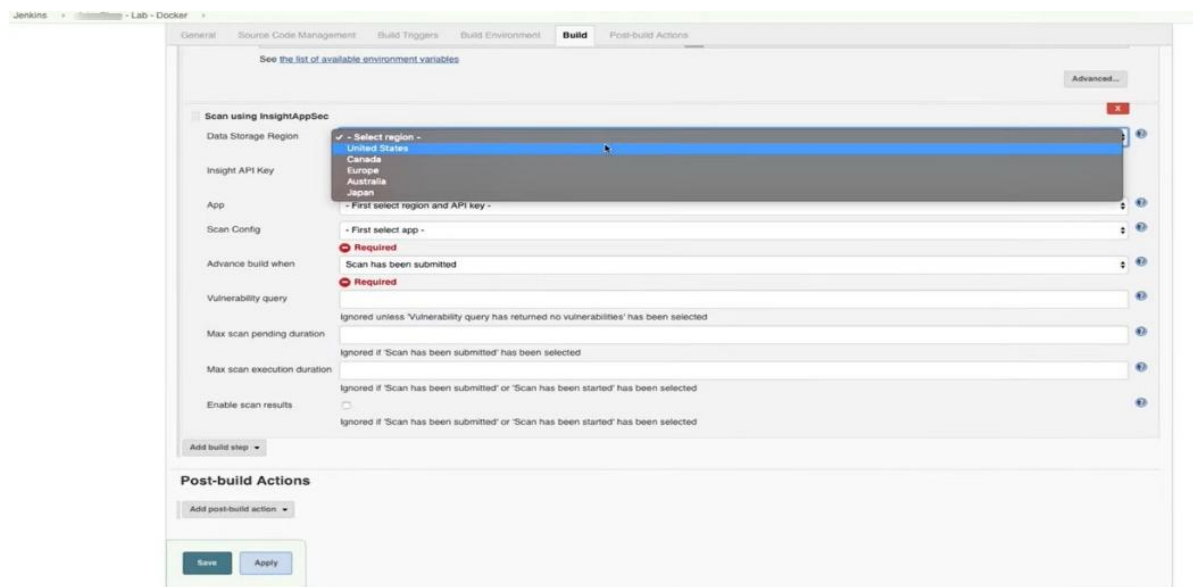
Scan Web Applications Using StackHawk in the Jenkins Pipeline

- Create a new Jenkins Pipeline job from the Jenkins console
- Configure the Pipeline portion of the Job Configuration settings to link to a **Jenkinsfile** in the source code repository
- Construct a **Jenkinsfile** in the root of the source code repository
- The StackHawk API key is extracted from the credentials database and saved as an environment variable during the **Run HawkScan Test** stage
- Launch the **Jenkins job**. Once the build is finished, a summary of results on the job console can be viewed
- Access the StackHawk account to see the scan details

```
pipeline {
  agent any
  stages {
    stage ("Checkout code") {
      steps {
        checkout scm
      }
    }
    stage ("Pull HawkScan Image") {
      steps {
        sh 'docker pull stackhawk/hawkscan'
      }
    }
    stage ("Run HawkScan Test") {
      environment {
        HAWK_API_KEY = credentials('HAWK_API_KEY')
      }
      steps {
        sh '''
          docker run -v ${WORKSPACE}:/hawk:rw -t \
            -e API_KEY=${HAWK_API_KEY} \
            -e NO_COLOR=true \
            stackhawk/hawkscan
          ...
        '''
      }
    }
  }
}
```

Integrate InsightAppSec DAST Tool with Jenkins

- Integrating InsightAppSec plugin with Jenkins helps the DevSecOps teams in automatically executing a scan against a web app
- Based on the scan results, Jenkins will make a decision regarding the status of the build
- The decision is based on various rules, such as exceeding a maximum number of vulnerabilities that are identified or reaching the vulnerability severity threshold
- You generate a **new API Key** in the Insight platform. The plugin can be installed from the plugin manager in Jenkins, or it can be installed manually by creating an **.hpi** file and uploading it
- To configure the plugin, select **Add Build Step** from the drop-down menu and choose **Scan with InsightAppSec**
- Provide the Insight API key and other necessary configuration information in the plugin configuration window



Integrate InsightAppSec DAST Tool with JIRA

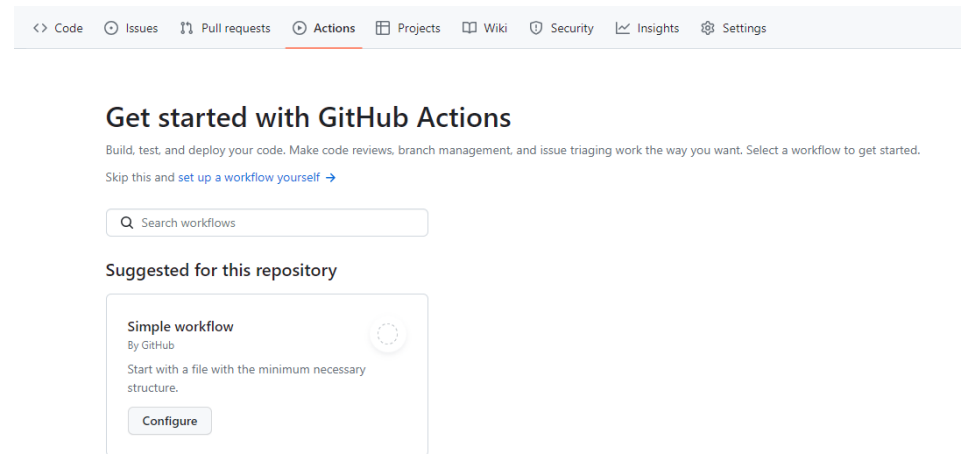
- The integration of InsightAppSec with JIRA reduces operational complexity between security and development teams to accelerate remediation of issues
- You generate an **API token** using an Atlassian account
- To set up a JIRA ticketing connection, go to the **Administration** tab and select **Add a new JIRA server**
- Provide the URL, username, and API Token in the New JIRA Server Connection form, and then click **Save**
- Click the **New Configuration** button on the Project Configurations for InsightAppSec screen, and select the projects from the Configuration window. Add configuration information
- Click **Export to JIRA** after selecting the vulnerabilities from the Scan Details screen

The screenshot displays the 'Scan Status' dashboard. At the top, it shows '16 Vulnerabilities Discovered' (with 11617 Attacks Performed) and '250 Crawled Links'. On the right, it indicates '22174 Network Requests' (with 1 Failed Request). Below this, there's a filter bar and a summary for '16 Vulnerabilities' with counts for Unreviewed (14), Ignored (0), False Positive (0), Verified (2), and Remediated (0). A table lists the vulnerabilities, including columns for URL, Parameter, Module Name, Severity, First Discovered, Variances, and Status. The table shows four entries, all with a severity of 'High' and status of 'Unreviewed'.

URL	Parameter	Module Name	Severity	First Discovered	Variances	Status
http://[redacted]/datastore/search_get_by_name.php	name	Blind SQL	High	07/06/17 9:38 AM	3	Unreviewed
http://[redacted].com/login.php	login	Blind SQL	High	07/06/17 9:38 AM	1	Unreviewed
http://[redacted]/datastore/search_by_name.php	name	Blind SQL	High	07/06/17 9:38 AM	3	Unreviewed
http://[redacted]/hutterdb/search_get_by_id4.php	id	Blind SQL	High	07/11/17 11:00 AM	3	Unreviewed

Web Application Security Scanning using GitHub Actions and OWASP ZAP

- Using **GitHub Actions**, it is convenient and simple to automate security testing and conduct ZAP scans without the use of any other infrastructure
- Choose new workflow from the repository on GitHub
- Enter **ZAP** in the Marketplace search box and choose **OWASP ZAP Full Scan** or **OWASP ZAP Baseline Scan**
- Hit the **Copy** button to obtain the YAML code snippet and paste it into the editor
- Once the YAML code is pasted, GitHub Actions is set up and ready
- Click **Commit changes..** to enable the GitHub Action



OWASP Code Analysis Tools

[https://owasp.org/www-community/Source Code Analysis Tools](https://owasp.org/www-community/Source_Code_Analysis_Tools)

- ABOM Scanner - Free ABOM is an online SCA (software composition analysis) tool that scans your application for open-source vulnerabilities using only a manifest file.
- Automated Security Helper - FREE ASH is a one stop shop for security scanners, and does not require any installation. It will identify the different frameworks, and download the relevant, up to date tools. ASH is running on isolated Docker containers, keeping the user environment clean, with a single aggregated report. The following frameworks are supported: Git, Python, Javascript, Cloudformation, Terraform and Jupyter.
- Fluid Attack's Scanner - free SAST, DAST and SCA vulnerability detection tool
- Google CodeSearchDiggity - Free Uses Google Code Search to identify vulnerabilities in open source code projects hosted by Google Code, MS CodePlex, SourceForge, Github, and more. The tool comes with over 130 default searches that identify SQL injection, cross-site scripting (XSS), insecure remote and local file includes, hard-coded passwords, and much more.

Pipeline Monitoring

Monitoring and managing

Monitoring and managing the DevSecOps pipeline in production Pipeline monitoring: Keep an eye on the CI/CD pipeline itself to ensure that all security checks are being executed, and no step is bypassed

Monitoring

Runtime monitoring: Continuously monitor the application in production for abnormal behavior or unauthorized access

Ensure

Feedback loops: Ensure a mechanism is in place for developers and operations teams to receive immediate feedback on the build and security status

Understand

Sehgal, Vandana Verma. Implementing DevSecOps Practices: Understand application security testing and secure coding by integrating SAST and DAST . Packt Publishing.

GitHub CodeQL

CodeQL is a semantic code analysis engine developed by GitHub that focuses on finding security vulnerabilities and other issues within a codebase. It operates by treating code as data, creating a queryable database from the source code of a project. This database contains a detailed, hierarchical representation of the code, including its abstract syntax tree, data flow graph, and control flow graph.



Shift Left in the Pipeline

Play 10: Shift Operational Test and Evaluation (OT&E) Left into the Pipeline

- The Defense Innovation Board succinctly summed the goal of this play like this: “Speed and cycle time are the
- most important metrics for managing software. DoD must be able to deploy software faster without sacrificing
- its abilities to test and validate software.”

- DoD DevSecOps Fundamentals Playbook March 2021



DevSecOps Metrics

DevSecOps provides demonstrable quality and security improvements over the traditional software lifecycle. Proficiency can be measured against both *tempo* and *stability* metrics, both of which are thoroughly defined through the DevOps Research and Assessment (DORA) Quick Check. As an introduction to DORA, the following key measurements are strong indicators of a team's proficiency at DevSecOps:

- Deployment Frequency – cycle Time, planning to production.
- Lead Time – the measurement between commit time to production deployment.
- Mean Time to Resolution (MTTR) – how long to get your code back up and running, if there is an incident.
- Change Failure Rate (CFR) – % changes going into production that require rework.
- DoD Enterprise DevSecOps Fundamentals

+

•

0