

Engenharia de Software Seguro

Laboratório 5 — Containerização, CI/CD e DevSecOps

Grupo 6

Elementos: Hugo Nunes, Miguel Teixeira

Repositório GitHub: [miguelteixeira77/ess-task-api-lab5](https://github.com/miguelteixeira77/ess-task-api-lab5)

Tag(s): lab5

Data: 22-02-2026

Objetivos principais:

- (i) Criar imagem Docker multi-stage.
- (ii) Validar execução via container.
- (iii) Publicar imagem em registry.
- (iv) Automatizar build/test/publish em github Actions.
- (v) Integrar controlos de segurança (secret scanning, image scanning, assinatura e SBOM).

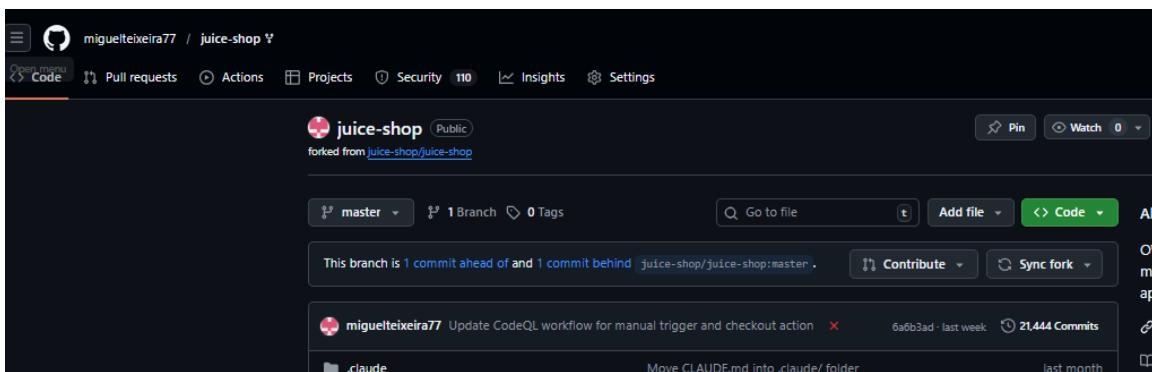
Índice

1. GitHub Actions – Aplicação Juice Shop.....	3
1.a Fork do Repositório	3
1.b Ativação do Code Scanning.....	3
1.c Alteração do Trigger do Workflow	4
1.d Inicialização da Base de Dados CodeQL	4
1.e Análise da Vulnerabilidade CWE-89 (SQL Injection).....	4
2. GitHub Actions – API de Tarefas.....	5
2.a Criação do Repositório	5
2.b Adição do Código Fonte	5
2.c Criação do Workflow Hello World	5
2.d Localização do Workflow	6
2.e Pipeline de Build com Maven e CodeQL	6
2.f Vulnerabilidade Zip Slip (CWE-22)	7
3. Containerização da Aplicação	8
3.1 Criação do Dockerfile (Multi-stage).....	8
3.2 Build da Imagem Docker	8
3.3 Execução da Imagem.....	8
3.4 Publicação no DockerHub.....	9
3.5 Commit do Dockerfile	10
4. Pipeline CI/CD Seguro.....	11
4.1 Criação do Workflow Base (cicd-base.yml).....	11
4.2 Deteção de Segredos com Trivy.....	11
4.3 Separação do Pipeline em Dois Jobs	12
4.4 Scan de Vulnerabilidades da Imagem Docker.....	13
4.5 Assinatura da Imagem com Cosign	13
4.6 Geração de SBOM.....	14
5. Conclusão	16

1. GitHub Actions – Aplicação Juice Shop

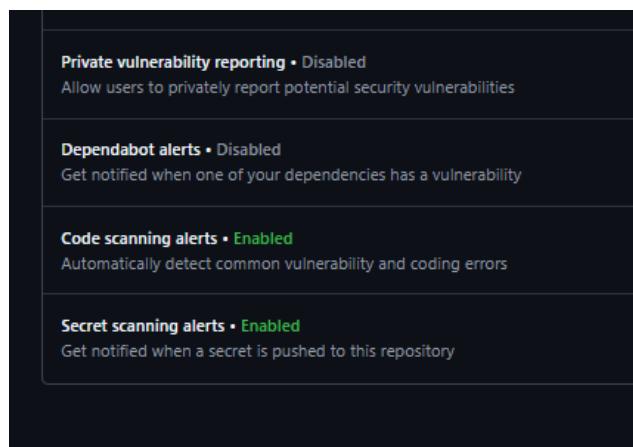
1.a Fork do Repositório

Foi realizado o fork do repositório oficial OWASP Juice Shop para a conta GitHub do grupo, mantendo o repositório público e adicionando todos os elementos do grupo como colaboradores.



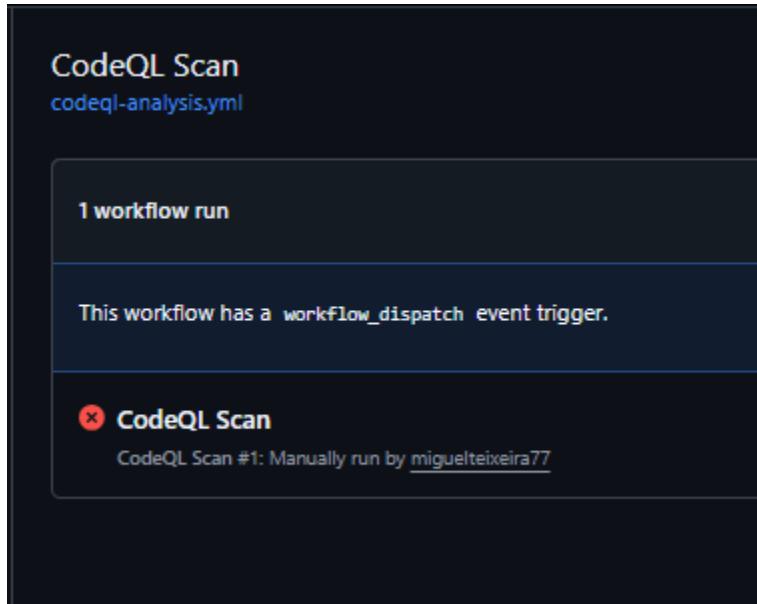
1.b Ativação do Code Scanning

Foi ativado o Code Scanning no menu Security do repositório, utilizando a configuração padrão do CodeQL disponibilizada pelo GitHub.



1.c Alteração do Trigger do Workflow

O workflow do CodeQL foi alterado para ser executado manualmente utilizando o trigger 'workflow_dispatch', permitindo a execução sob pedido no separador Actions.



1.d Inicialização da Base de Dados CodeQL

Nos logs da execução foi identificada a etapa 'Initialize CodeQL', responsável pela criação da base de dados de análise estática. Esta etapa executa o comando 'codeql database init', que prepara a representação intermédia do código fonte para posterior análise.

1.e Análise da Vulnerabilidade CWE-89 (SQL Injection)

O CodeQL identificou a vulnerabilidade 'Database query built from user-controlled sources' no ficheiro routes/search.ts. Esta vulnerabilidade ocorre porque dados provenientes de uma source controlada pelo utilizador (parâmetros HTTP) fluem diretamente para um sink sensível (execução de query SQL) sem validação ou utilização de prepared statements.

Source: Parâmetros HTTP (req.query, req.body, req.params).

Sink: Execução de queries SQL através de funções como db.query().

Esta situação corresponde à CWE-89 – Improper Neutralization of Special Elements used in an SQL Command.

The screenshot shows a GitHub security alert for a database query built from user-controlled sources. The alert is labeled '#49' and has a severity of 'High'. It was first detected 38 minutes ago. The alert points to a file named 'routes/search.ts' at line 23. The code snippet is as follows:

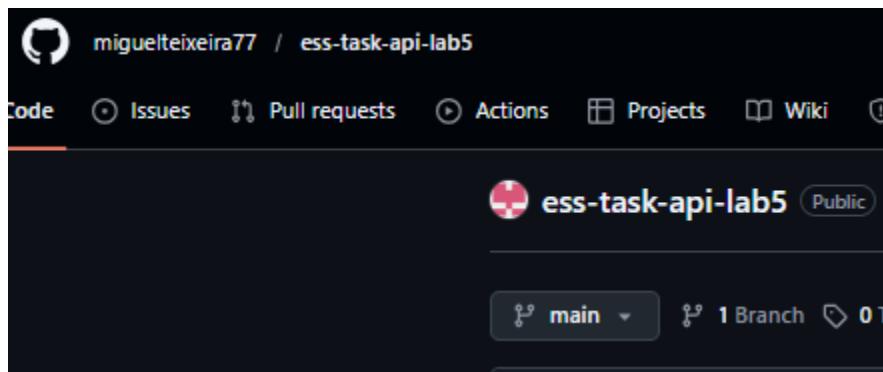
```
routes/search.ts:23
20   return (req: Request, res: Response, next: NextFunction) => {
21     let criteria: any = req.query.q === 'undefined' ? '' : req.query.q ?? ''
22     criteria = (criteria.length <= 200) ? criteria : criteria.substring(0, 200)
23     models.sequelize.query(`SELECT * FROM Products WHERE ((name LIKE '${criteria}%') OR description LIKE '${criteria}%') AND deletedAt = false`)
```

An error message indicates that the database query depends on a user-provided value ('req.query.q'). A 'Generate fix' button is available. The alert also lists affected branches (master, default), development status, tags (security), and weaknesses (CWE-89, CWE-90, CWE-943).

2. GitHub Actions – API de Tarefas

2.a Criação do Repositório

Foi criado um novo repositório público na conta GitHub do grupo para alojar a API de tarefas.



2.b Adição do Código Fonte

Foi adicionado ao repositório o código fonte da API de tarefas desenvolvida anteriormente.

2.c Criação do Workflow Hello World

Foi criado o ficheiro '.github/workflows/hello-world.yml' com trigger 'workflow_dispatch', executando o comando 'echo Hello, world!'.

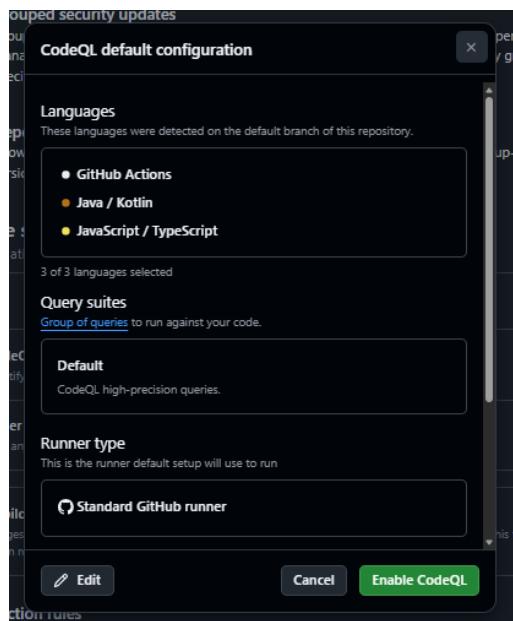
This screenshot shows the GitHub Actions interface. At the top, it says "3 workflow runs". Below that, it states "This workflow has a `workflow_dispatch` event trigger." A button labeled "Run workflow" is visible. The main list contains one item: "Hello world" (green checkmark), which was manually run by [miguelteixeira77](#). It is associated with the "main" branch and was run 1 minute ago. There is also a "..." button.

2.d Localização do Workflow

Após realizar pull do repositório para o ambiente local, o ficheiro encontra-se na diretoria: `.github/workflows/hello-world.yml`.

2.e Pipeline de Build com Maven e CodeQL

Foi implementado um pipeline de CI composto pelas etapas de compilação (`mvn clean install`), execução de testes unitários (`mvn test`) e análise estática com CodeQL.

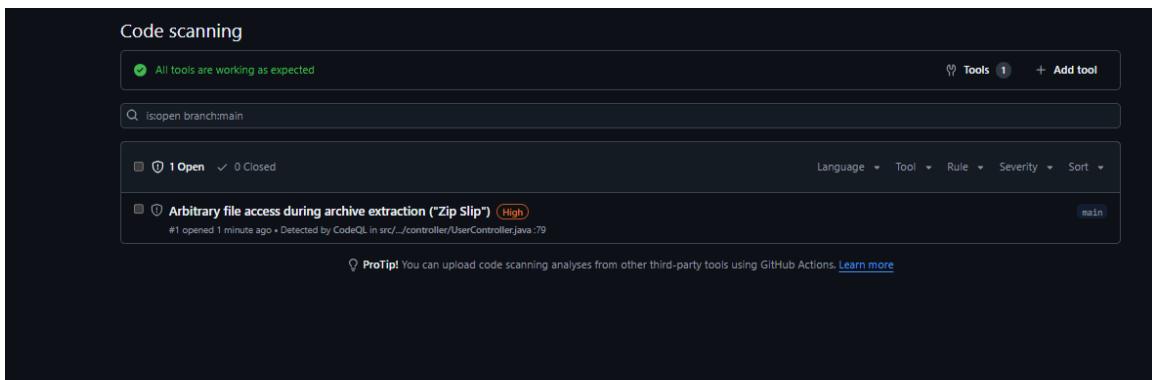


This screenshot shows the GitHub Actions interface with "3 workflow runs". The first run is "Merge branch 'main' of https://github.com/miguelteixeira77/ess-task-a... (CI (Build, Test, CodeQL) #3: Commit 918d02a pushed by miguelteixeira77)" on the "main" branch, completed 2 minutes ago. The second run is "CI (Build, Test, CodeQL) (CI (Build, Test, CodeQL) #2: Manually run by miguelteixeira77)" on the "main" branch, completed 22 minutes ago. The third run is "Create CI workflow for build and CodeQL analysis (CI (Build, Test, CodeQL) #1: Commit 19802dc pushed by miguelteixeira77)" on the "main" branch, completed 24 minutes ago. Each run has a "..." button.

2.f Vulnerabilidade Zip Slip (CWE-22)

Foi introduzida uma vulnerabilidade do tipo Zip Slip através de um método que extrai ficheiros ZIP sem validação do caminho das entradas. Esta vulnerabilidade permite Path Traversal, possibilitando a escrita de ficheiros fora do diretório pretendido.

A vulnerabilidade Zip Slip está relacionada com a CWE-22 (Path Traversal), uma vez que permite manipulação de caminhos no sistema de ficheiros.



The screenshot shows the GitHub Code scanning interface. At the top, there's a green checkmark indicating "All tools are working as expected". Below that is a search bar containing the query "isopen branch:main". Under the search bar, there are filters for "Open" (1) and "Closed" (0) issues. A single issue is listed: "Arbitrary file access during archive extraction ("Zip Slip") [High]". The issue details show it was opened 1 minute ago, detected by CodeQL in src./controller/UserController.java:79. There are dropdown menus for "Language", "Tool", "Rule", "Severity", and "Sort". A "ProTip!" message at the bottom suggests uploading code scanning analyses from other third-party tools using GitHub Actions.

3. Containerização da Aplicação

3.1 Criação do Dockerfile (Multi-stage)

Foi criado um Dockerfile multi-stage, permitindo separar a fase de build (Maven) da fase de runtime (JRE). Esta abordagem reduz o tamanho final da imagem e melhora a segurança.

- Stage 1: Build com Maven
- Stage 2: Runtime com eclipse-temurin JRE

3.2 Build da Imagem Docker

A imagem foi construída localmente utilizando o comando docker build.

```
task-api-lab5> docker build -t todo-service .
[+] Building 16.7s (7/17)
=> [internal] load build definition from Dockerfile
=> [internal] load metadata for docker.io/library/eclipse-temurin:21-jre-alpine
=> [internal] load metadata for docker.io/library/maven:3.9-eclipse-temurin-21
=> [auth] library/maven:pull token for registry-1.docker.io
=> [auth] library/eclipse-temurin:pull token for registry-1.docker.io
=> [internal] load .dockerrcignore
=> [internal] load context
=> [internal] transfer context: 2B
=> [build 1/5] FROM docker.io/library/maven:3.9-eclipse-temurin-21@sha256:67dd489908ce4a87c1bea43c589b00c1121a6
=> => resolve docker.io/library/maven:3.9-eclipse-temurin-21@sha256:67dd489908ce4a87c1bea43c589b00c1121a64c1b0ca
=> => sha256:333c5b030538510cc249a075dcffa38d6ee7392c4dca64fe06b302084558e7e 156B / 156B
=> => sha256:4f4fb700ef54461cf02571ae0db9a0dc1e0cd5577484a6d75e68d38e8acc1 32B / 32B
=> => sha256:eb0b4b738c2eba69dec10d2b3d07b8558a1729f0df36f6ccde2c6a104f9f50633 851B / 851B
=> => sha256:6fe9d8f3eebe07f057d9e829a030f1d7aeeb5b485f96a27ac0ca87ac94e7d54 2.10MB / 9.31MB
=> [internal] load dockerrcignore
=> [internal] load context
=> [internal] transfer context: 60.68kB
=> [stage-1 1/5] FROM docker.io/library/eclipse-temurin:21-jre-alpine@sha256:6ad8ed080d9be96b61438ec3ce99388e29
=> => resolve docker.io/library/eclipse-temurin:21-jre-alpine@sha256:6ad8ed080d9be96b61438ec3ce99388e294af216ed5
=> => sha256:3655cb26ed1af33f0fe5361c725d9b419f8f7e57bb595528b27cc387059adaa2 2.28kB / 2.28kB
=> => sha256:1a7a516dcbb69b4973083542d2549433254cc864cec8da3251f7d889b69099e2 128B / 128B
=> => sha256:5df9c4c4dcf56dc89e6ef5eb334bec79efafa46ced3dd8fb4b4a2cb7d1985e1ab 5.24MB / 53.17MB
=> => sha256:71075870ab4982d45fa235ee69bf94cf05112736cabcd28cb26fe512a6cc2c15 2.10MB / 16.84MB
=> => sha256:589002ba0eaed121a1dbf42f66448f29e5be55d5c8a6ee0f8eaa0285cc21ac153 2.10MB / 3.86MB
|
```

3.3 Execução da Imagem

A imagem foi executada com mapeamento de portas e validação do funcionamento da API.

	Name	Container ID	Image	Port(s)
<input type="checkbox"/>	<input checked="" type="radio"/> postgres-academy	bc70cdc0fae4	postgres	5432:5432
<input type="checkbox"/>	<input checked="" type="radio"/> agitated_volhard	2e11dee7cbac	todo-service	443:443 Show all ports (2)

The screenshot shows the Postman interface with the following details:

- Collection:** C-Academy ESS - Todo Lists - Phase1
- Request:** POST /Login
- Method:** POST
- URL:** http://localhost/login
- Body:** JSON (raw) content: {"username": "admin", "password": "cncts**2024"}
- Response Status:** 200 OK
- Response Body:** Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJjbmNzLmFjYWRlbXkuZXNzIiwic3ViIjoiYWRtaW4iLCJ1c2VySWQiOjMsInVzZXJuYW1lIjoiYWRtaW4iLCJpYXQiOjE3NzE3NTYyMDAsImV4cCI6MTc3MTC10TgwMH0.xpd0sUiifl9ZLWDhj5gRLseoA9CRW3SJgoU90IA41bA

```

[INFO] [JettyServerThreadpool-42] cncs.academy.ess.controller.UserController - Create user: admin
[INFO] [JettyServerThreadpool-29] cncs.academy.ess.controller.UserController - Create user: admin
[INFO] [JettyServerThreadpool-47] cncs.academy.ess.controller.UserController - Login user: userhash
[INFO] [JettyServerThreadpool-87] cncs.academy.ess.controller.UserController - Login user: admin
  
```

3.4 Publicação no DockerHub

Após autenticação com docker login, a imagem foi publicada no DockerHub.

Comandos utilizados:

- docker tag
- docker push

```
k-api-lab5\ess-task-api-lab5> docker tag todo-service miguelteixeira77/todo-service
```

```
k-api-lab5\ess-task-api-lab5> docker login
Authenticating with existing credentials... [Username: miguelteixeira77]

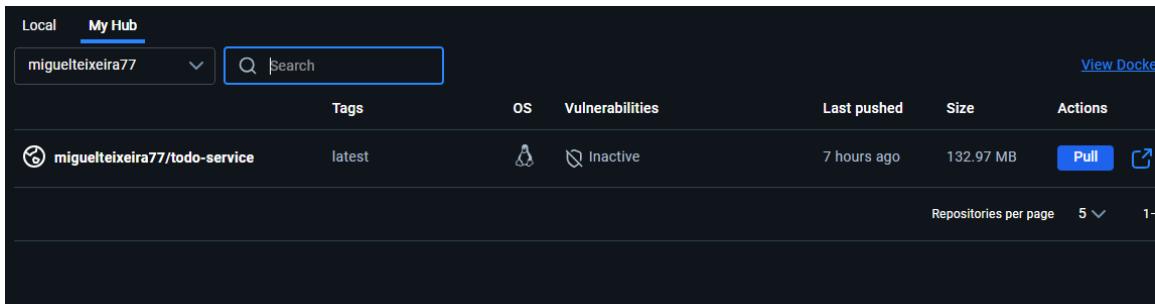
Info → To login with a different account, run 'docker logout' followed by 'docker login'

Login Succeeded
```

Using default tag: latest
The push refers to repository [docker.io/miguelteixeira77/todo-service]
51024d09d55a: Pushed
c4247047ab40: Pushed
f1389b9b70dd: Pushed
1a005a500401: Pushed
d23efddf26cd: Pushed
4222d53aeb55: Pushed
01d7766a2e4a: Pushed
f3b2661a0534: Pushed
5231ace86ec4: Pushed
05ec7c6edf1e: Pushed
latest: digest: sha256:671ef24c66056b8d1d0dfadbc360765b4db9fe8fac98b69834d9fb8f4ffd5e3b size: 856

3.5 Commit do Dockerfile

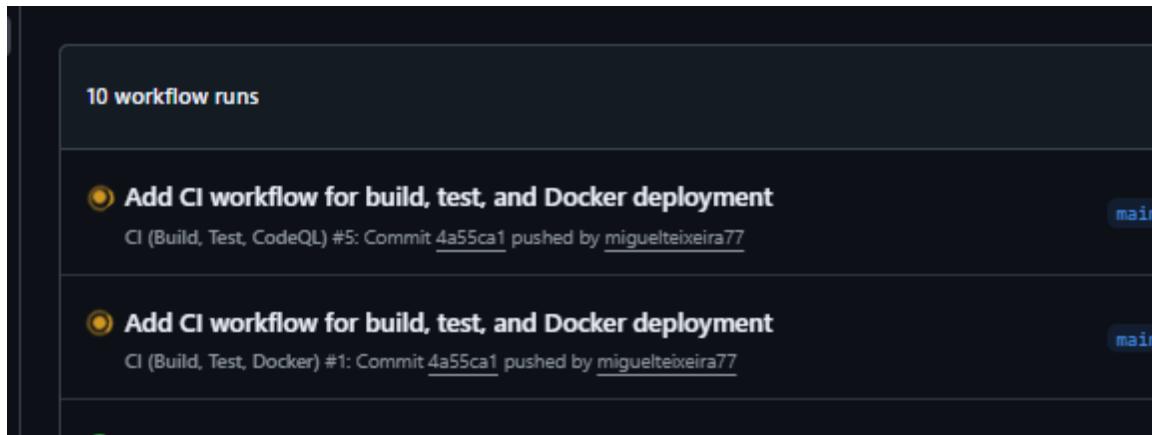
O Dockerfile foi versionado e enviado para o repositório GitHub.



4. Pipeline CI/CD Seguro

4.1 Criação do Workflow Base (cicd-base.yml)

Foi criado um workflow no GitHub Actions para execução automática em push e pull request. Este inclui build Maven, testes e publicação da imagem no GitHub Container Registry (GHCR).



4.2 Detecção de Segredos com Trivy

Foi integrado o Trivy para detetar segredos no código-fonte (filesystem scan). Os resultados foram exportados em formato SARIF e publicados na secção Security → Code Scanning.

A screenshot of the GitHub Security Code scanning interface. It shows a summary message 'All tools are working as expected' with a green checkmark icon. Below it, there is a search bar with the query 'is:open branch:main'. Under the search bar, there is a section with a grey square icon, a warning icon (!), the text '2 Open', and '0 Closed'. Below this, there is a detailed view of a single issue: 'Asymmetric Private Key [High]'. The issue details state '#2 opened 1 minute ago • Detected by Trivy in key.pem :'. There is also another partially visible issue below it.

4.3 Separação do Pipeline em Dois Jobs

O pipeline foi dividido em dois jobs distintos: 1) SAST + Testes + Secret Scan 2) Build & Push da imagem Docker (dependente do primeiro através de 'needs').

All workflows
Showing runs from all workflows

16 workflow runs

- ponto 2f**
CI/CD Base (Tests + SAST + Build & Push) #4: Commit [c2e8091](#) pushed by [miguelteixeira77](#) main
- ponto 2f**
CI (Build, Test, CodeQL) #8: Commit [c2e8091](#) pushed by [miguelteixeira77](#) main
- Integrate Trivy secret scan into CI/CD workflow**
CI/CD Base (Tests + SAST + Build & Push) #3: Commit [9056359](#) pushed by [miguelteixeira77](#) main
- Integrate Trivy secret scan into CI/CD workflow**
CI (Build, Test, CodeQL) #7: Commit [9056359](#) pushed by [miguelteixeira77](#) main

ponto 2f #4

Summary
Triggered via push 5 minutes ago
[miguelteixeira77](#) pushed [→ c2e8091](#) main
Status: Success | Total duration: 2m 34s | Artifacts: 2

cicd-base.yml
on: push

```

graph LR
    A[SAST + Tests + Secret scan] --> B[Build & Push Docker im...]

```

Build & Push Docker image (GHCR) summary
Docker Build summary
For a detailed look at the build, download the following build record archive and import it into Docker Desktop's Builds view. Build records include details such as timing, dependencies, results, logs, traces, and other information about a build. [Learn more](#)
[miguelteixeira77-ess-task-api-lab5-RFP9I9.dockerbuild](#) (193.08 KB - includes 1 build record)
Find this useful? [Let us know](#)

ID	Name	Status	Cached	Duration
RFP9I9	ess-task-api-lab5	completed	0%	59s

► Build inputs
Job summary generated at run-time

4.4 Scan de Vulnerabilidades da Imagem Docker

Foi realizado um scan de vulnerabilidades à imagem Docker utilizando Trivy. Os resultados foram exportados em SARIF e disponibilizados como artifact.

The screenshot shows the GitHub Actions interface for a workflow named 'miguelteixeira77-ess-task-api-labs-389RWD.dockerbuild'. The summary table includes columns for ID, Name, Status, Cached, and Duration. There are two annotations listed under 'Annotations' (SAST + Tests + Secret scan and Build & Push Docker Image (GHCR) warnings) and three artifacts listed under 'Artifacts' (miguelteixeira77-ess-task-api-labs-389RWD.dockerbuild, trivy-image-results, and trivy-secret-results).

4.5 Assinatura da Imagem com Cosign

A imagem Docker foi assinada utilizando Cosign (modo keyless) com suporte OIDC do GitHub Actions. As assinaturas foram armazenadas no GHCR associadas à imagem publicada.

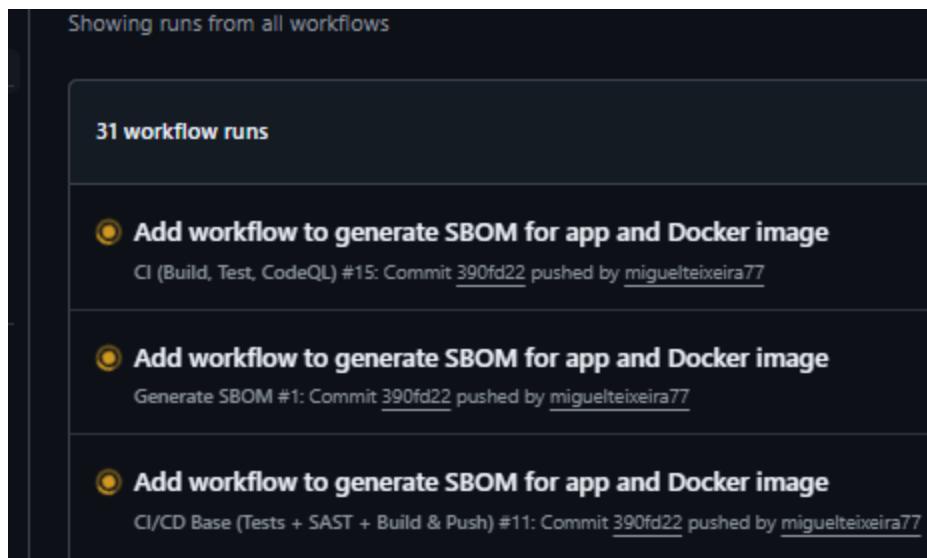
The screenshot shows the 'All workflows' page with four workflow runs listed. Each run is associated with a green checkmark icon and the text 'Enhance CI/CD workflow with Cosign integration'. The runs are identified by their names: 'CI/CD Base (Tests + SAST + Build & Push) #6', 'CI (Build, Test, CodeQL) #10', 'CI (Build, Test, CodeQL) #9', and 'CI (Build, Test, CodeQL) #8'. All runs were pushed by the user 'miguelteixeira77'.

4.6 Geração de SBOM

Foi gerado um SBOM no formato CycloneDX (sbom.cdx.json), permitindo listar todas as dependências e componentes da imagem. O ficheiro foi disponibilizado como artifact do workflow.

The screenshot shows the GitHub Actions job summary for a workflow named "ess-task-api-lab5". The job has completed successfully (Duration: 51s) and produced several artifacts:

- Annotations:** 2 warnings:
 - SAST + Tests + Secret scan: CodeQL Action v3 will be deprecated in December 2026. Please update all occurrences of [Show more](#)
 - Build, Scan & Push Docker image (GHCR) + Cosign + SBOM: CodeQL Action v3 will be deprecated in December 2026. Please update all occurrences of [Show more](#)
- Artifacts:** Produced during runtime
 - miguelteixeira77~ess-task-api-lab5~JVQS2N.dockerbuild (193 KB)
 - sbom (16.7 KB)
 - trivy-image-results (10.3 KB)
 - trivy-secret-results (933 Bytes)



Os ficheiros SBOM gerados no ponto 2.g ficam disponíveis como artefactos do workflow no GitHub Actions, podendo ser descarregados através do separador *Actions* após a execução do pipeline.

As assinaturas da imagem Docker, geradas com *cosign*, ficam armazenadas no GitHub Container Registry (GHCR) como artefactos associados à imagem publicada.

Tanto os SBOM como as assinaturas podem ser descarregados ou verificados localmente, garantindo a integridade e rastreabilidade da imagem construída.

5. Conclusão

A Ficha 5 permitiu implementar um pipeline CI/CD completo com integração de práticas DevSecOps. Foram aplicadas técnicas de deteção de segredos, análise de vulnerabilidades, assinatura de imagens Docker e geração de SBOM.

O resultado é um fluxo automatizado seguro que reforça a segurança da cadeia de fornecimento de software.

Este trabalho demonstra a aplicação prática de conceitos modernos de segurança em pipelines de integração e entrega contínua.