



Image Scanning - Build Secure Images

Key Takeaways

All rights reserved to nnSoftware GmbH

No part of this publication may be reproduced, copied, transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of nnSoftware GmbH

About TechWorld with Nana

TechWorld with Nana is an established name in the DevOps and Cloud industry, and it stands for the quality trainings helping 1,000s of engineers acquire the most in-demand skills in this field.

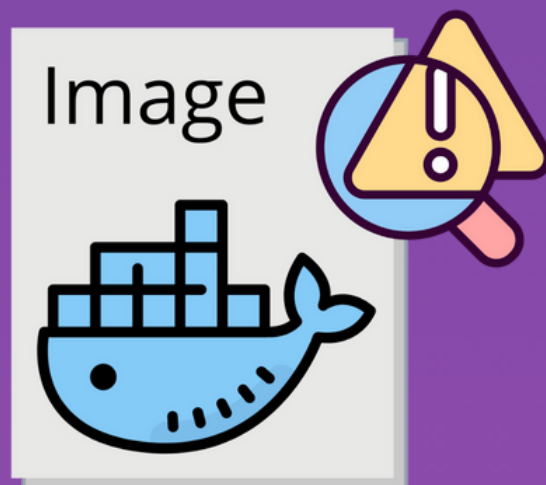


Our mission is enable individual engineers as well as companies to take advantage of the recent developments in Cloud and DevOps fields, to use technologies and concepts in order to create efficient, automated, streamlined DevSecOps processes in organisations.

CD Part opens up a whole new threat landscape



We will secure step by step



And start with **Docker Image Security**



Automated Security Scanning of Docker Image

Why scan Container images?

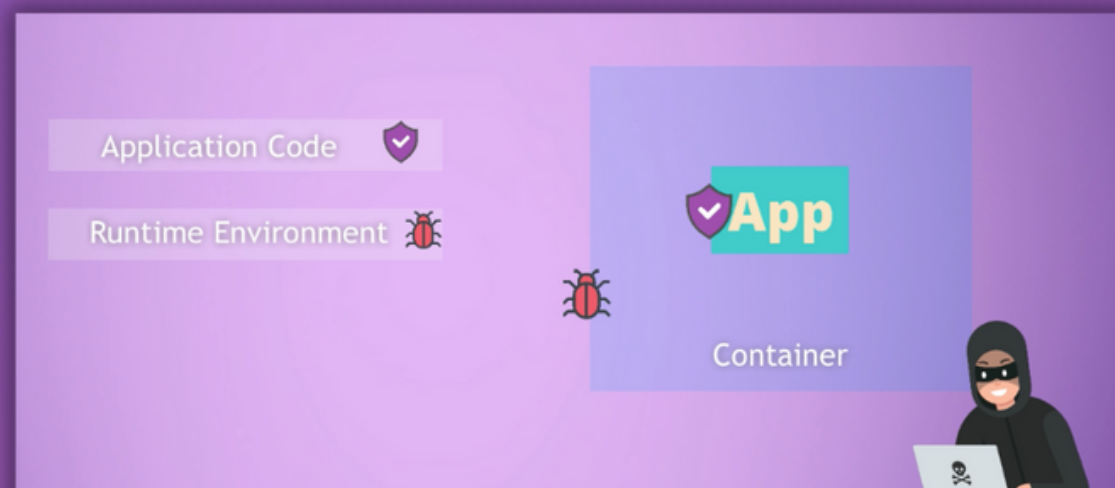
Mini Virtual Server Environment

- Docker images consist of an **underlying operating system, runtime, dependencies**
- Images rely on **base images**
- **Tools and packages installed** in the image



Security Threat

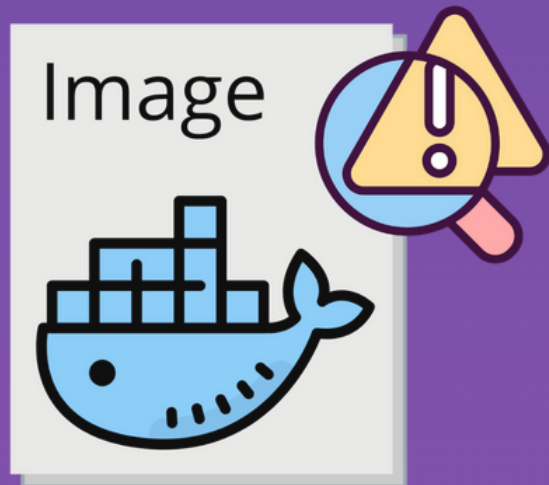
- This means we can have a secure application, but still an insecure runtime environment, which **hackers can exploit**



Dockerfile 666 B

```
1 FROM node:18 as installer
2 COPY . /juice-shop
3 WORKDIR /juice-shop
4 RUN npm i -g typescript ts-node
5 RUN npm install --omit=dev --unsafe-perm
6 RUN npm dedupe
7 RUN rm -rf frontend/node_modules
8 RUN rm -rf frontend/.angular
9 RUN rm -rf frontend/src/assets
10 RUN mkdir logs
11 RUN chown -R 65532 logs
12 RUN chgrp -R 0 ftp/ frontend/dist/ logs/ data/ i18n/
13 RUN chmod -R g=u ftp/ frontend/dist/ logs/ data/ i18n/
14 RUN rm data/chatbot/botDefaultTrainingData.json || true
15 RUN rm ftp/legal.md || true
16 RUN rm i18n/*.json || true
17
18 FROM gcr.io/distroless/nodejs:18
```

Container Image Scanning Tools



We can again leverage **security tools**

- ✓ To scan every image layer for known vulnerabilities
- ✓ Open source and commercial tools available
- ✓ More and more tools **natively support security features**

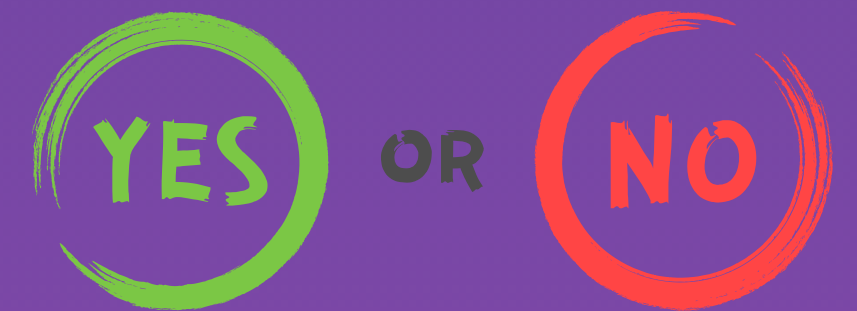
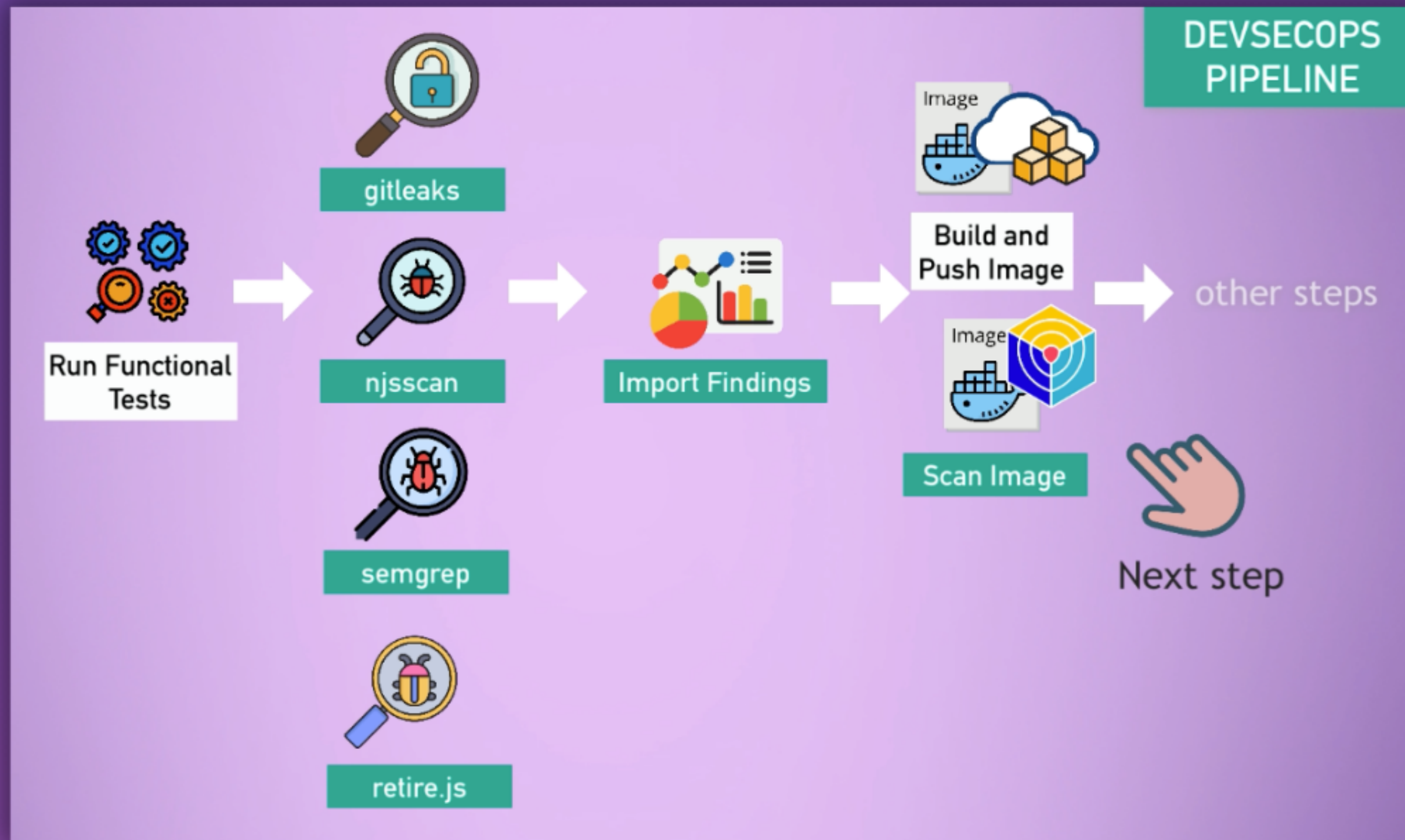


Indicator that **security is becoming more important** within the developer workflow



We use Trivy to scan our image

Integrate Scanning step into our CI/CD



Based on the result, we can decide whether we want to deploy the application or not

Integrate Scanning step into our CI/CD

Add Job to Pipeline Configuration

```
133   trivy:
134     stage: build
135     needs: ["build_image"]
136     image: docker:24
137     services:
138       - docker:24-dind
139     before_script:
140       - apk --no-cache add curl python3 py3-pip
141       - pip3 install --no-cache-dir awscli
142       - curl -sL https://raw.githubusercontent.com/aquasecurity/trivy/main/contrib/install.sh | sh -s -- -b /usr/local/bin
143       - aws ecr get-login-password | docker login --username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com
144     script:
145       - docker pull $IMAGE_NAME:$CI_COMMIT_SHA
146       - trivy image $IMAGE_NAME:$CI_COMMIT_SHA
147
```

Trivy Security Scanning

Trivy can **scan files** inside container images for

- ✓ Vulnerabilities
- ✓ Exposed Secrets
- ✓ Misconfigurations, useful only if your image includes IaC files
- ✓ Licenses

By default, vulnerability and secret scanning are enabled

```
es for 66359313666.dkr.ecr.eu-west-3.amazonaws.com/juice-shop:539894ce2c4d160eaa17c25350d7f3cc23e05646
148 $ trivy image $IMAGE_NAME:$CI_COMMIT_SHA
149 2023-08-02T16:51:01.988Z      INFO      Need to update DB
150 2023-08-02T16:51:01.988Z      INFO      DB Repository: ghcr.io/aquasecurity/trivy-db
151 2023-08-02T16:51:01.988Z      INFO      Downloading DB...
152 19.53 MiB / 38.68 MiB [----->] 50.50% ? p/s
    / 38.68 MiB [----->] 100.00% ? p/s ?38.68 MiB
    B [----->] 100.00% ? p/s ?38.68 MiB / 38.68 M
    ----->] 100.00% 31.88 MiB p/s ETA 0s38.68 MiB / 38.68 MiB [-----
    ----->] 100.00% 31.88 MiB p/s ETA 0s38.68 MiB / 38.68 MiB [-----
    ----->] 100.00% 31.88 MiB p/s ETA 0s38.68 MiB / 38.68 MiB [-----
```



Trivy utilizes a database containing vulnerability information

It downloads the database every 6 hours and is cached and updated as needed

Trivy Security Scanning

What Trivy can find

- ✓ OS packages and software dependencies
- ✓ Known vulnerabilities (CVEs)
- ✓ Trivy scans vulnerabilities in the application dependencies

```
INFO Detected OS: debian
INFO Detecting Debian vulnerabilities...
INFO Number of language-specific files: 1
INFO Detecting node-pkg vulnerabilities...
-3.amazonaws.com/juice-shop:539894ce2c4d160eaa17c25350d7f3cc23e05646 (deb
=====
11, MEDIUM: 8, HIGH: 4, CRITICAL: 0)
```

Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
base64url (package.json)	NSWG-ECO-428	HIGH	fixed	0.0.6	>=3.0.0	Out-of-bounds Read https://hackerone.com/reports/321687
	GHSA-rvg8-pwq2-xj7q	MEDIUM			3.0.0	Out-of-bounds Read in base64url https://github.com/advisories/GHSA-rvg8-pwq2-xj7q
ecstatic (package.json)	CVE-2019-10775			3.3.2	4.1.3	Denial of Service in ecstatic https://avd.aquasec.com/nvd/cve-2019-10775



Following packages are supported

- ▶ OS packages
- ▶ Language-specific packages




Trivy automatically detects the dependencies file specific to the programming language

Remediation Example in Dockerfile

Reported vulnerability in Trivy

- Vulnerability of Debian base image
- It uses the library libssl1.1, which has known vulnerabilities



Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
libssl1.1	CVE-2023-0464	HIGH	fixed	1.1.1n-0+deb11u4	1.1.1n-0+deb11u5	Denial of service by excessive resource usage in verifying X509 policy constraints... https://avd.aquasec.com/nvd/cve-2023-0464
	CVE-2023-2650					Possible DoS translating ASN.1 object identifiers https://avd.aquasec.com/nvd/cve-2023-2650

Suggested solution is to update to the fixed version



Debian is used under the hood. It's an outdated image

```
18 FROM gcr.io/distroless/nodejs:18
```



Change to image with secure debian base image

```
18 FROM node:18-bookworm-slim
```




Fixing Security Issues is not a zero-sum game



- Sometimes when you upgrade to a newer version to fix a known vulnerability in a previous version, you **introduce other, newer vulnerabilities**
- Some libraries may have not fixed the issue yet
- It's hard to get to zero vulnerabilities, but the goal is to **get to the closest secure state** possible





















Some steps you can take in such cases

- ✓ Assess the **severity**
- ✓ **Alternative solutions**
- ✓ **Monitor** updates
- ✓ Research and implement **mitigation strategies** to reduce the risk
- ✓ **Risk acceptance** after thorough understanding of potential risks

GitLab CI/CD Status after this chapter

Pipeline Needs Jobs 10 Failed Jobs 4 Tests 0

Group jobs by Stage Job dependencies

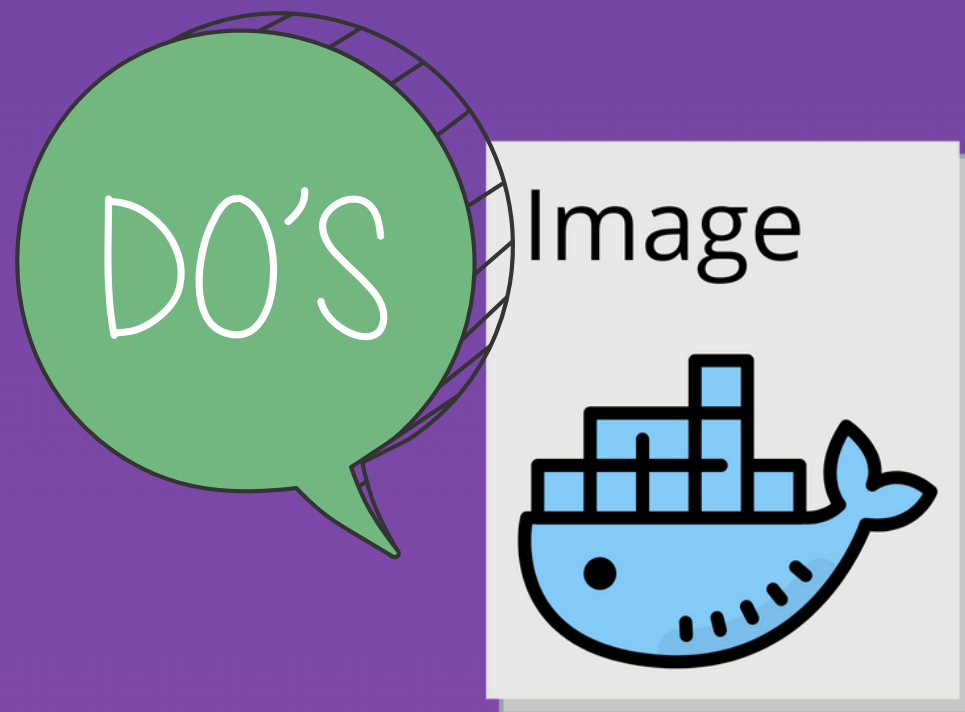
cache	test	build	upload-reports	deploy
 create_cache 	 gitleaks 	 build_image 	 upload_reports 	 deploy_image 
	 njsscan 	 trivy 		
	 retire 			
	 semgrep 			
	 yarn_test 			





Docker Security Best Practices

Why learn Security Best Practices



Learn about Security Best Practices, that we can follow

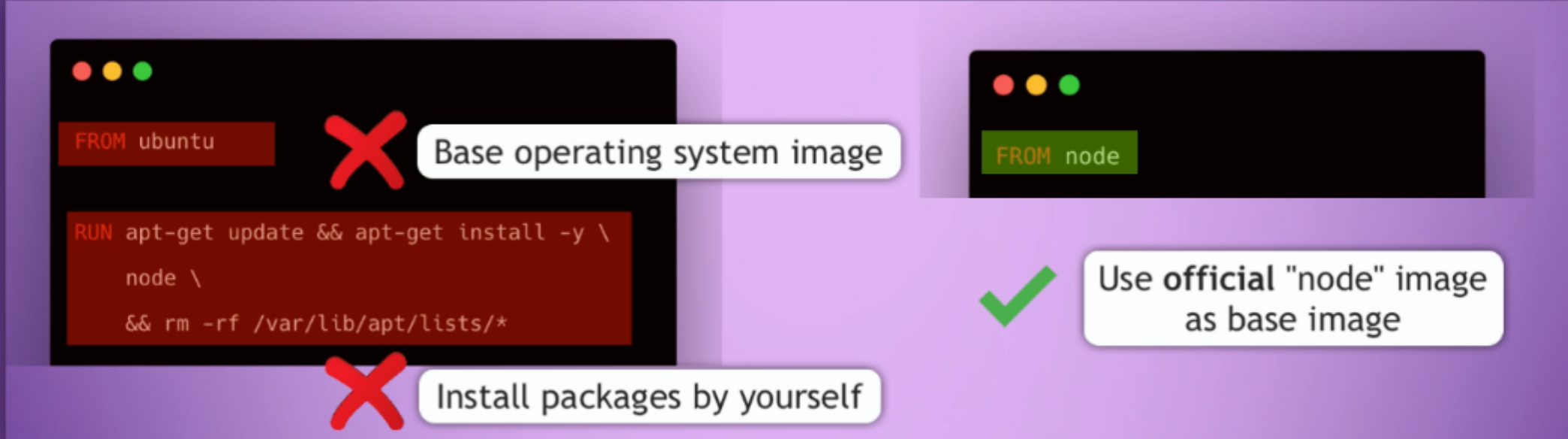
- ✓ To be **less exposed** to security threats
- ✓ Have a **smaller attack surface**

Security Best Practices

DO'S

1 - Use Official Docker Images as Base Image

- Verified and already built with best practices

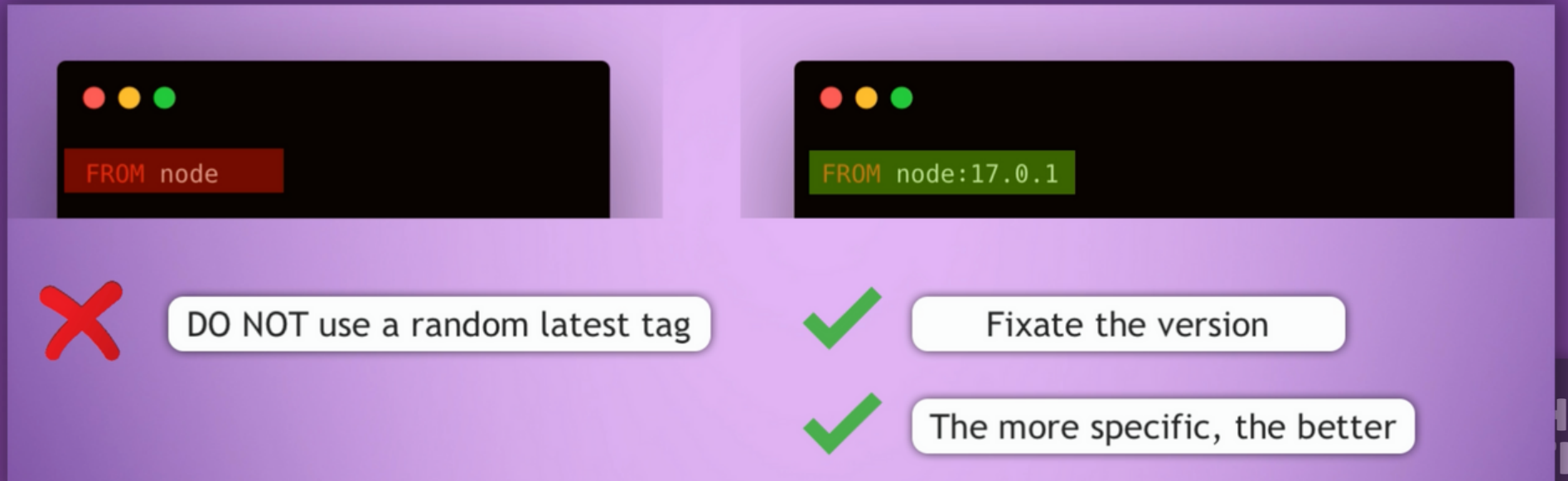


The diagram compares two Dockerfile snippets. On the left, a snippet starting with `FROM ubuntu` is marked with a red 'X' and labeled 'Base operating system image'. Below it, a `RUN` command is shown: `apt-get update && apt-get install -y \ node \ && rm -rf /var/lib/apt/lists/*`, which is also marked with a red 'X' and labeled 'Install packages by yourself'. On the right, a snippet starting with `FROM node` is marked with a green checkmark and labeled 'Use official "node" image as base image'.

DO'S

2 - Use Specific Image Versions

- You might get different image versions that break stuff
- Latest tag is unpredictable



The diagram compares two Dockerfile snippets. On the left, a snippet starting with `FROM node` is marked with a red 'X' and labeled 'DO NOT use a random latest tag'. On the right, a snippet starting with `FROM node:17.0.1` is marked with a green checkmark and labeled 'Fixate the version'. Below this, another green checkmark is labeled 'The more specific, the better'.

Security Best Practices

DO'S

3 -Use Small-Sized Official Images

- Avoid introducing unnecessary security issues from the beginning
- Reduced attack surface



The image shows two terminal windows side-by-side. The left window has a red background and displays 'FROM node:17.0.1'. Below it is a red 'X' icon and a text box that says 'Could be based on full-blown OS'. The right window has a green background and displays 'FROM node:17.0.1-alpine'. Below it is a green checkmark icon and a text box that says 'Use Image based on a leaner and smaller OS distro'.

DO'S

4 - Use .dockerignore to explicitly exclude unneeded files and folders

- List files and folders you want to ignore
- Prevent unintended secrets exposure

```
# ignore .git and .cache folders
.git
.cache

# ignore all markdown files (md)
*.md

# ignore sensitive files
private.key
settings.json
```


Security Best Practices

DO'S

5 - Make use of Multi-Stage Builds

- “From” instruction starts a new build stage, leaving everything you don’t want in the final image behind
- Selectively copy artifacts from one stage to another
- Only the last Dockerfile commands are the image layers
- Reduces security attack surface

Image



Image



final image

Dockerfile with 2 Build Stages

```
# Build stage
FROM maven AS build
WORKDIR /app
COPY myapp /app
RUN mvn package
```

```
#Run stage
FROM tomcat
COPY --from=build /app/target/file.war /usr/local/tomcat/webapps/
...
```

Security Best Practices



6 - Use the Least Privileged User

- **Bad Practice:**

- Using root or user with high privilege
- Easier privilege escalation for an attacker

- **Best Practice**

- Create a dedicated user and group
- Set required permissions
- Change to non-root user



Some base images have a generic user bundled in, which we can use



```
...  
  
# create group and user  
RUN groupadd -r tom && useradd -g tom tom  
  
# set ownership and permissions  
RUN chown -R tom:tom /app  
  
# switch to user  
USER tom  
  
CMD node index.js
```



Continuously Scan Images in Container Registry

Why scanning the image once is not enough

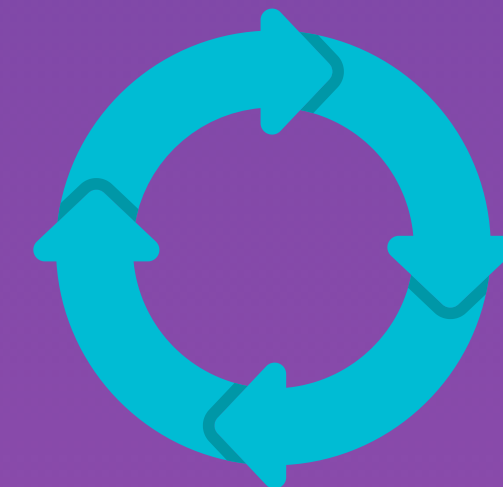


Image can be **safe at the point of scanning**



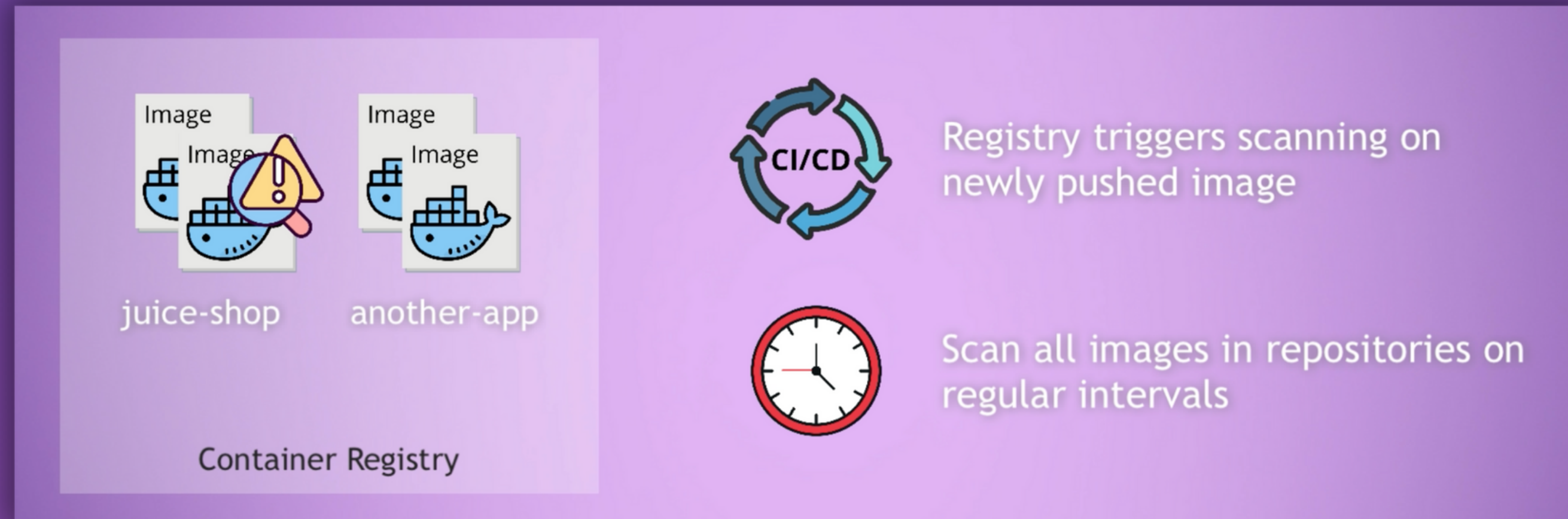
But can be **unsafe a few days later**

- We could use a Docker image in production and don't even know it
- That's why we need to **continuously scan** the Docker images in the container image repository

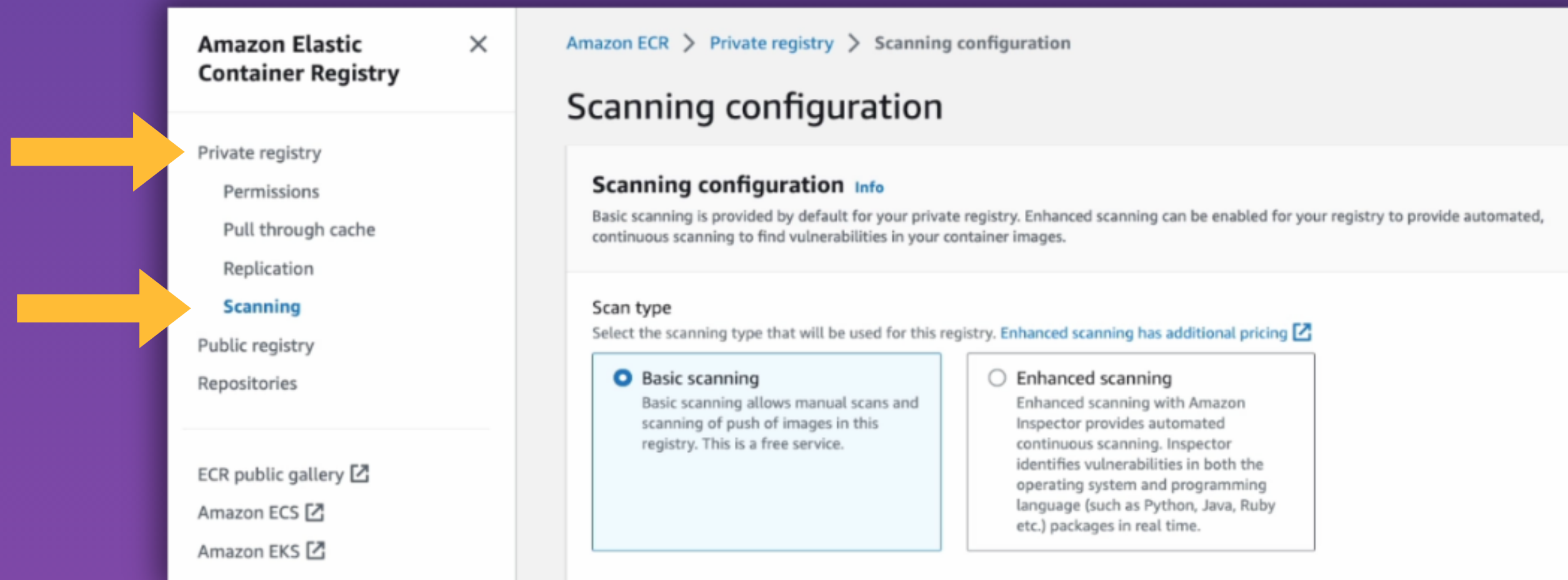


How to continuously scan

Container registries offer such functionality



Configure Security Scanning on AWS ECR



Basic Scanning

- **Manually or Scanning on Push:**
 - Vulnerability detected when image is pushed
- Free of charge

Enhanced Scanning

- **Continuous Scanning:**
 - Detection when vulnerabilities occur
- **Deeper, more comprehensive** security analysis, more extensive set of vulnerability databases
- Comes at an additional cost

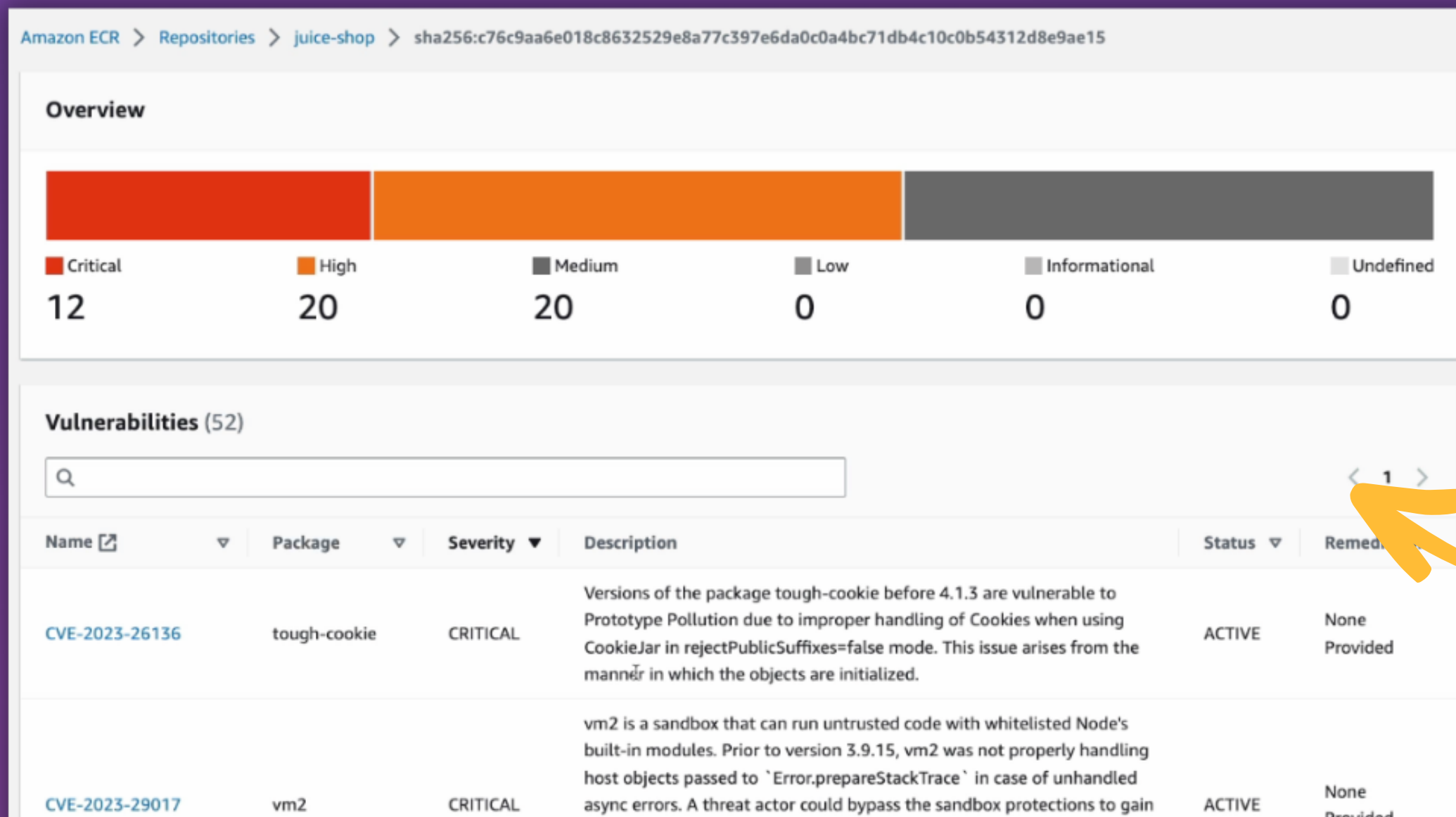
Configure Security Scanning on AWS ECR

Repository with continuous scanning enabled

Private repositories (1)

Find repositories

	Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	
<input type="checkbox"/>	juice-shop	066359313666.dkr.ecr.eu-west-3.amazonaws.com/juice-shop	01 August 2023, 21:44:15 (UTC+02)	Disabled	Continuous	AES-256	

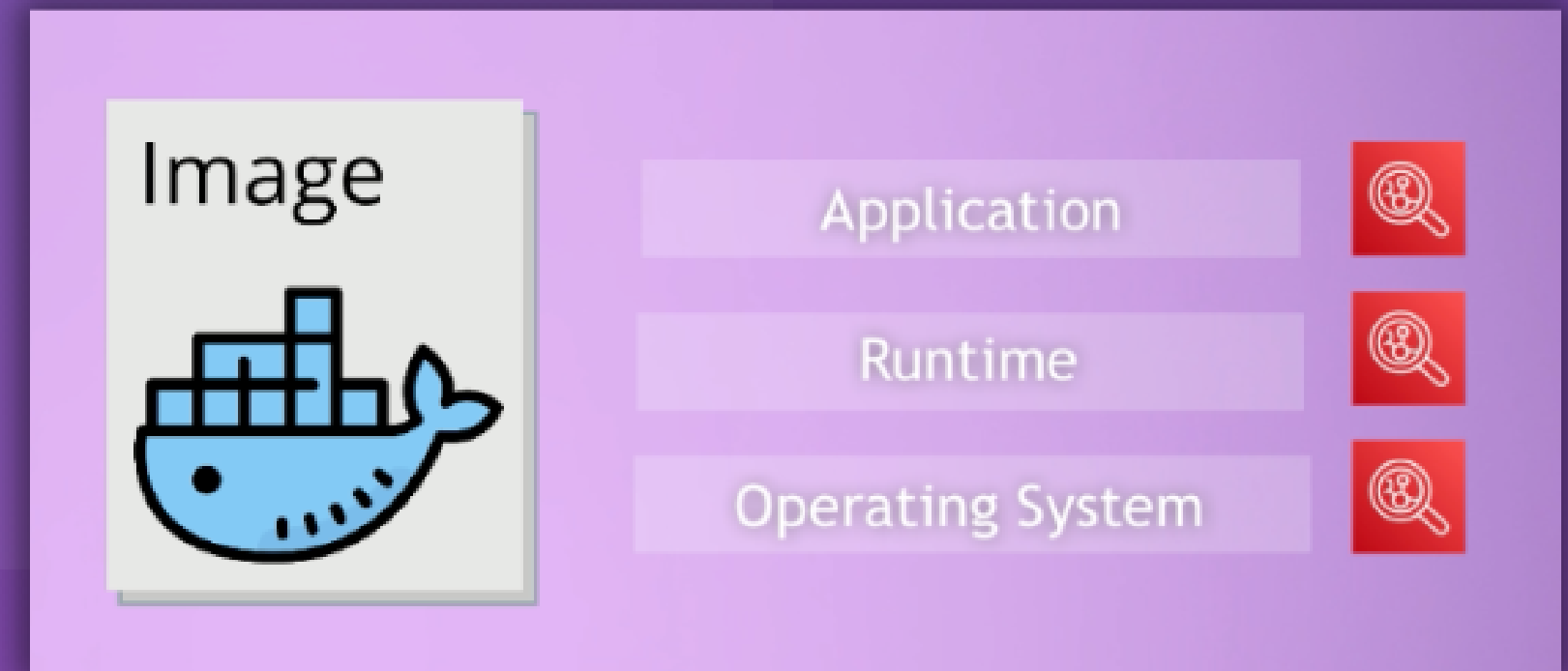


Vulnerability Findings

rights reserved.

What Amazon Inspector Scans

- ✓ Scans application runtime environment
- ✓ Scans whole application and application dependencies



Wrap Up and Next Steps



- We have secured our **application** in the last chapters
- We have secured the **container runtime** in this chapter

TO DO

- We will secure the **continuous deployment**
- And the **AWS cloud infrastructure**

