Module Four

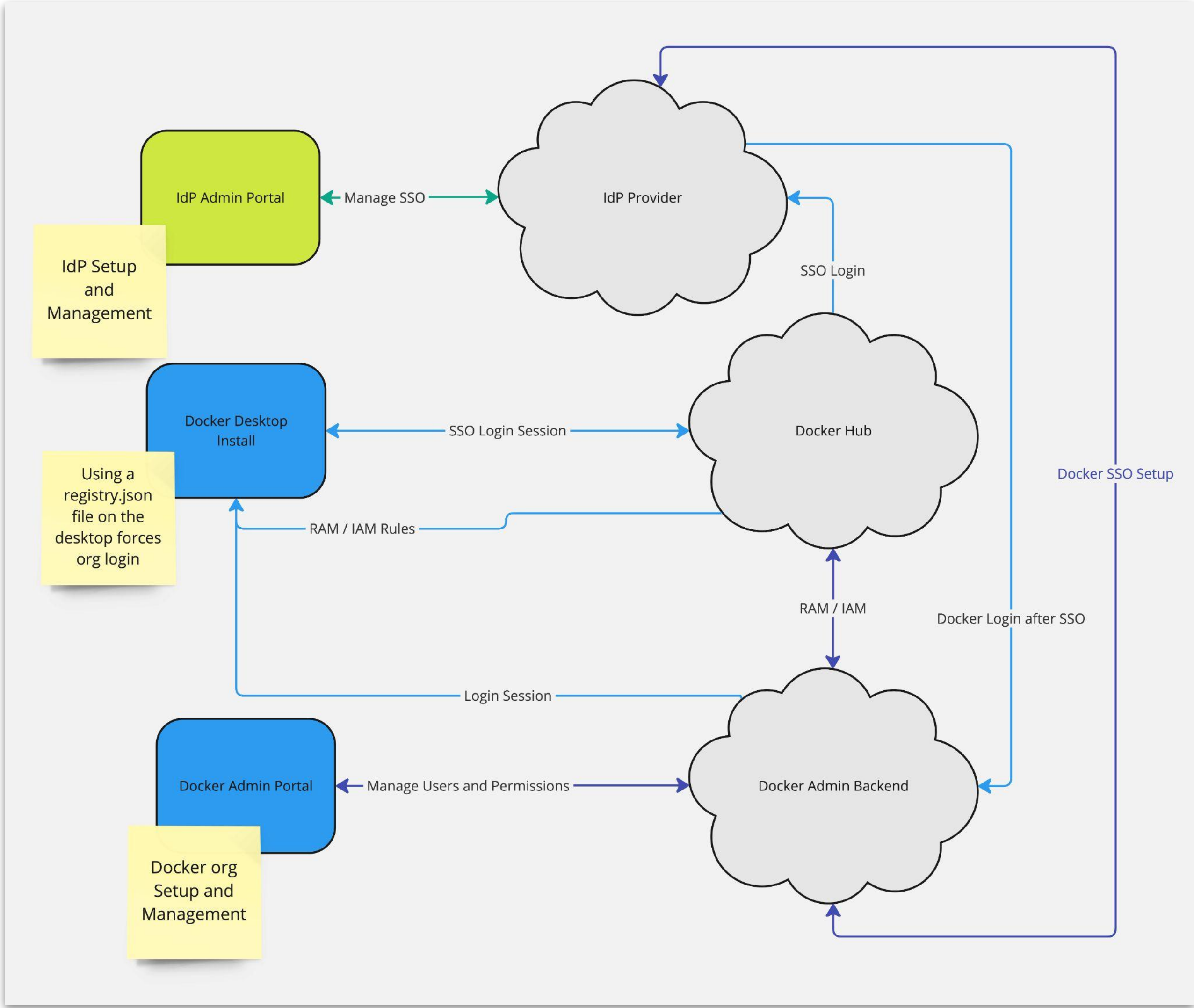
Docker Security

Logging In

Authentication Options

Type	Description	Usage
Username / Email	User is able to log in with either a username/password or email/password; user authenticates to an account on Docker Hub	Most common with personal accounts; can be used with organization accounts
Github	Uses Github to auth/create an account on Docker Hub	Most common with personal accounts; can be used with organization accounts
Google	Uses Google to auth/create an account on Docker Hub	Most common with personal accounts; can be used with organization accounts
Corporate Idp SSO	Uses Idp (SAML, OIDC) to authenticate user, creates account on Docker Hub if needed via JIT provisioning; does not reap inactive accounts	If enforced, prevents username / password login and requires email login via Idp
Corporate Idp SCIM	Uses Idp (SAML, OIDC) to authenticate user and keeps licensed user pools in sync between Docker Hub and the Idp	If enforced, prevents username / password login and requires email login via Idp
Allowed Orgs Key	Not an auth method, but a modifier that provides the allowed list of docker organizations that the user can authenticate to	Docker Desktop will not start until the user is logged into the allowed org

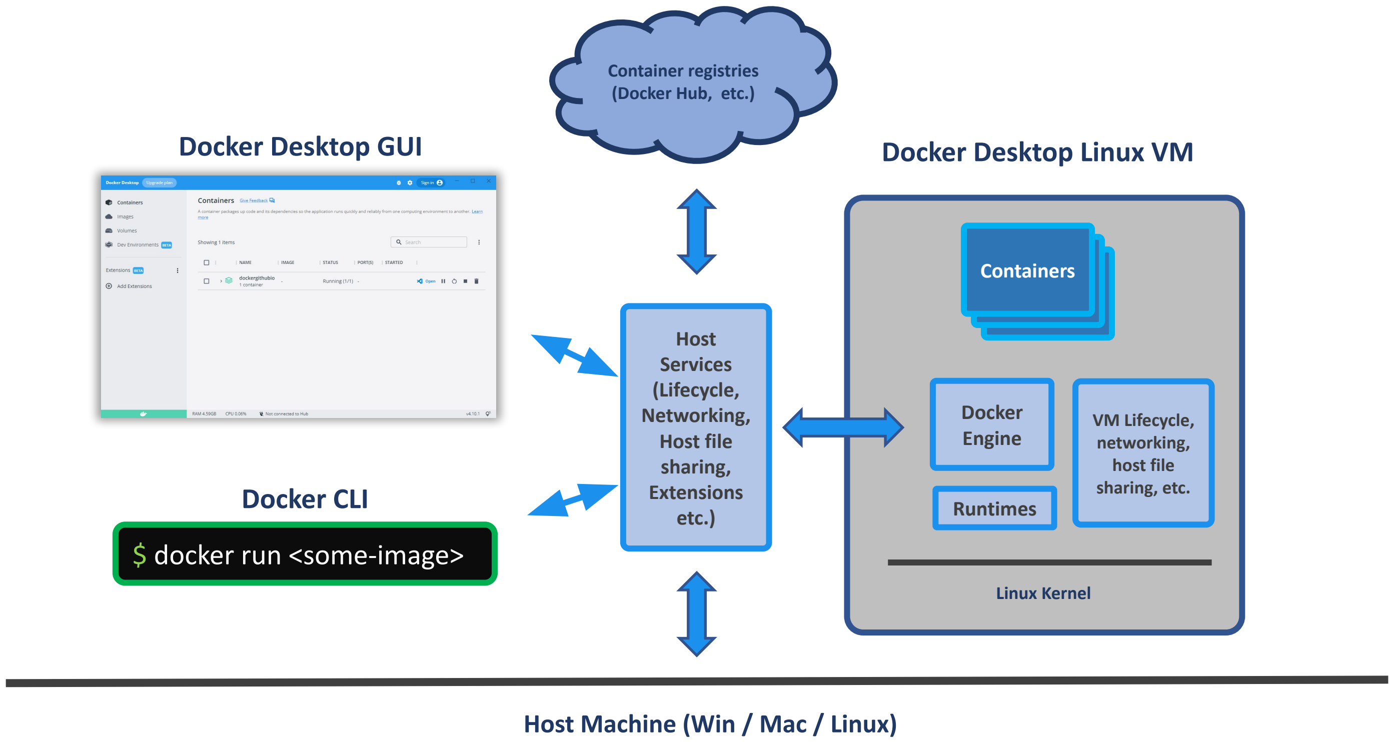
SSO/SCIM FLOW



docker

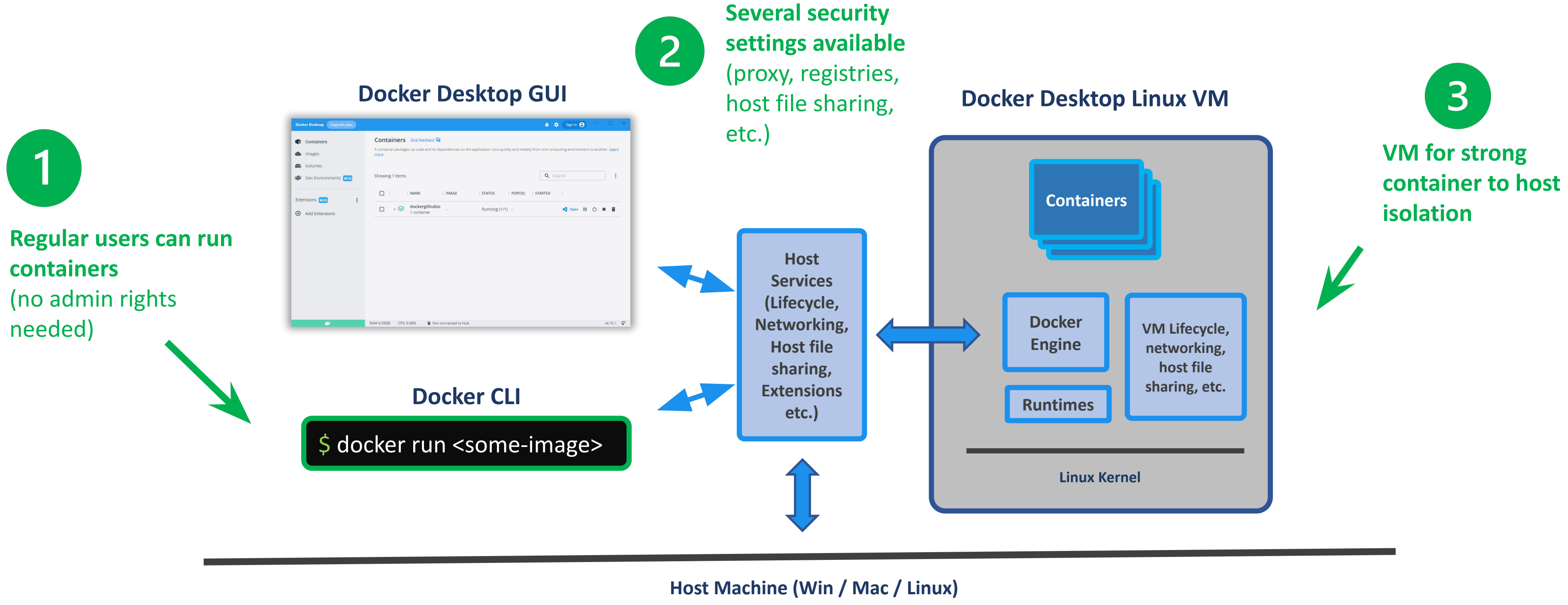
Docker Desktop Security Architecture

Docker Desktop Architecture


Container registries (Docker Hub, etc.)
Docker Desktop GUI
Docker Desktop Linux VM
Host Services (Lifecycle, Networking, Host file sharing, Extensions etc.)
Containers
Docker Engine
Runtimes
VM Lifecycle, networking, host file sharing, etc.
Linux Kernel
Docker CLI
Host Machine (Win / Mac / Linux)

```
$ docker run <some-image>
```

Baseline Security Features



Containers

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Showing 1 items

NAME	IMAGE	STATUS	PORT(S)	STARTED
dockergithubio 1 container	-	Running (1/1)	-	

RAM 4.59GB CPU 0.06% Not connected to Hub v4.10.1

```
$ docker run <some-image>
```

Regular users can run containers
(no admin rights needed)

Several security settings available
(proxy, registries, host file sharing, etc.)

VM for strong container to host isolation

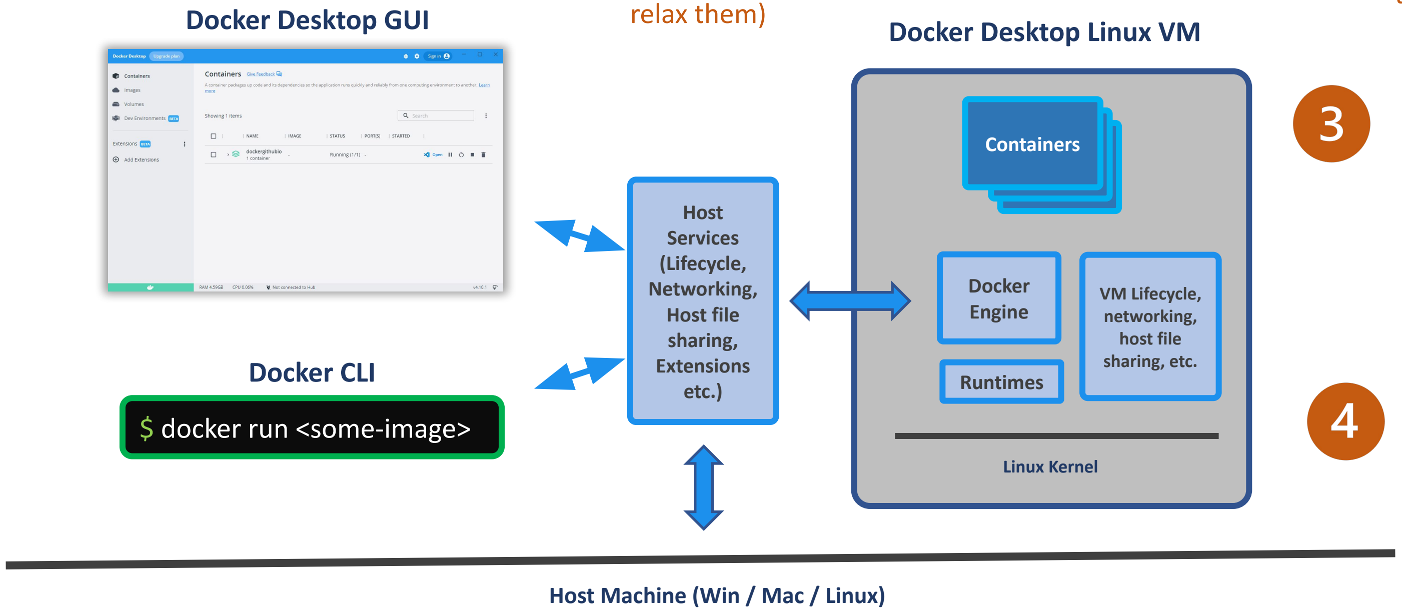
Vulnerable Areas

1 No way for IT admins to lock security settings (developers can relax them)

2 Containers run as <u>root</u> inside the VM.

3 Containers may access VM internals (Docker Engine, Linux kernel, VM services, etc.)

4 VM acts as a black box (host antimalware can't see inside)


Docker Desktop GUI
Docker CLI
Host Services (Lifecycle, Networking, Host file sharing, Extensions etc.)
Docker Desktop Linux VM
Containers
Docker Engine
VM Lifecycle, networking, host file sharing, etc.
Runtimes
Linux Kernel
Host Machine (Win / Mac / Linux)

```
$ docker run <some-image>
```

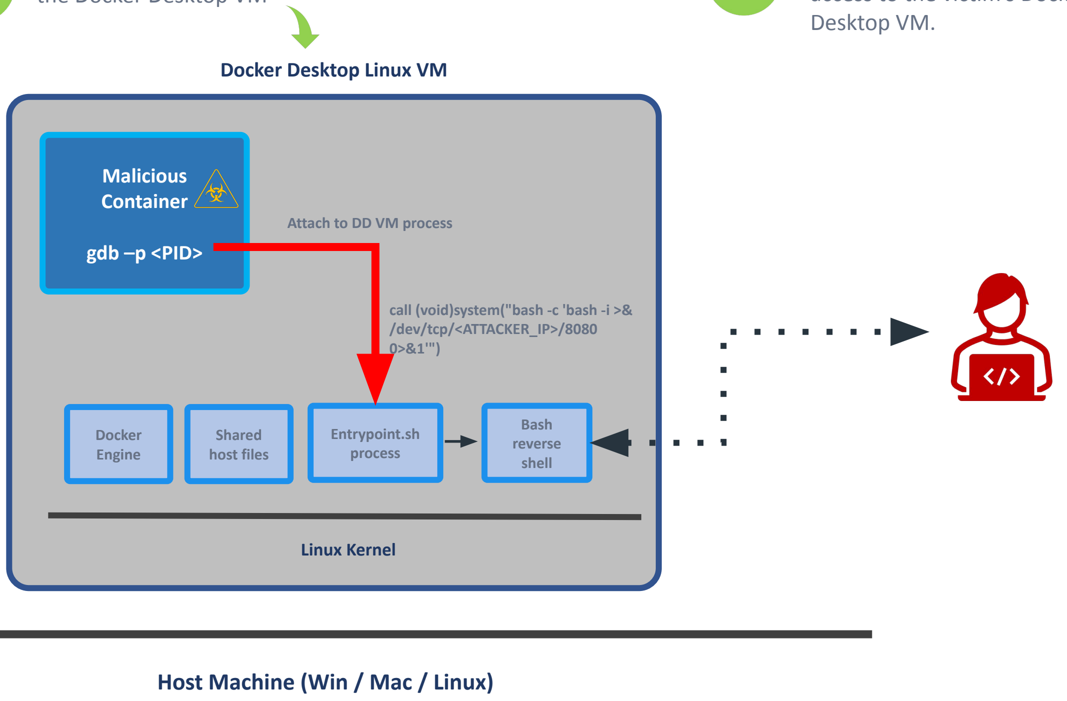
Attacker Reverse Shell on Developer Laptop

1 User inadvertently runs malicious container image with elevated privileges.

```
$ docker run --privileged –pid=host bad-image
```

2 Malicious container runs inside the Docker Desktop VM

3 Attacker gets a shell with ***root*** access to the victim's Docker Desktop VM.



docker

Threat Landscape

Threats & Actors

Threats

- Malware in containers
- Supply chain attacks
    - Corrupt package, corrupt image, etc.
- Misconfiguration by developers

Threat Actors

- Malicious container images
- Malicious software packages in containers
- Unaware / careless developers

Example: Malware in Containers

Malicious Images	Posing as popular images such as Alpine, OpenJDK, Golang.
Malicious Packages	Developers may inadvertently insert these into containers at buildtime
Attacker Techniques	Typo Squatting, Dependency Confusion, etc.

Other Attack Mechanisms Possible

- **Container breakout via CVE:**

  CVE 2019-5736: escape from container (no "—privileged" required)

- **Container breakout via sensitive container mounts:**

  docker run –v /var/run/docker.sock:/var/run/docker.sock

  docker run –v /:/mnt

  docker run –v /bin:/mnt

- **Container breakout via elevated container privileges:**

  docker run –privileged

  docker run –pid=host

  docker run –cap-add=SYS_ADMIN

  docker run –security-opt=seccomp-unconfined

docker

Hardened Docker Desktop

Enhance Docker security on developer workstations, **without impacting** developer experience & productivity

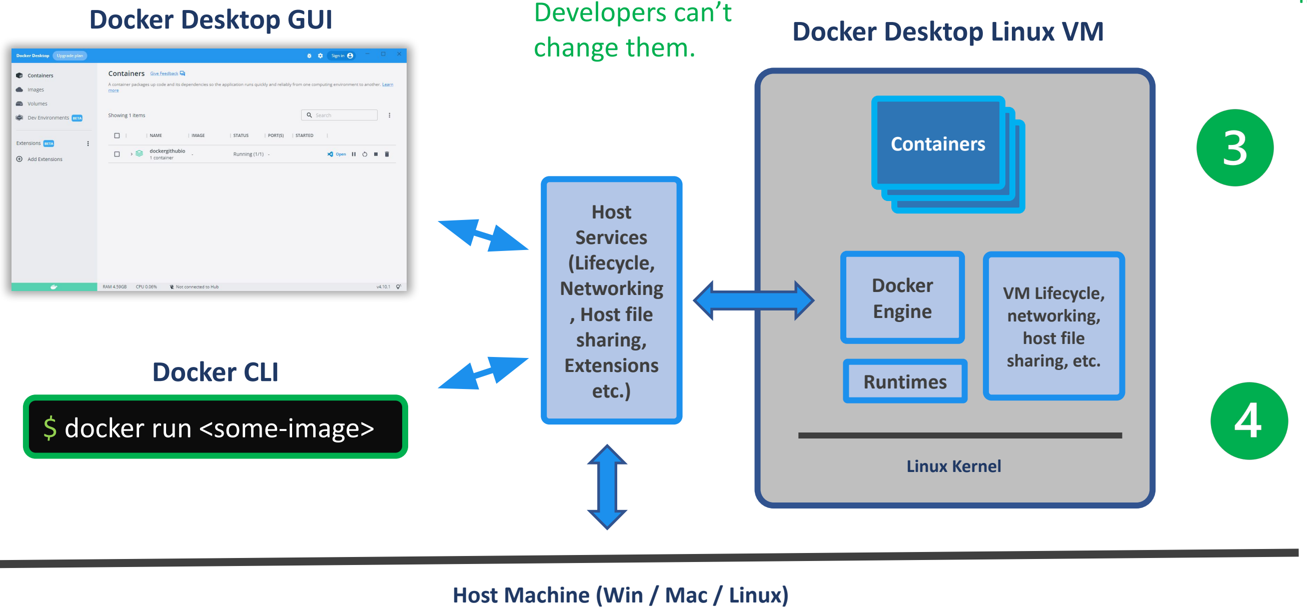
Hardened Desktop Threat Mitigation

1 Enables IT admins to lock security settings of their choice; Developers can't change them.

2 Containers run **unprivileged** inside the VM

3 **Containers blocked from accessing VM internals** (Docker Engine, Linux kernel, VM services, etc.)

4 Blocks developers from accessing the VM.


Docker Desktop GUI
Docker CLI
Host Services (Lifecycle, Networking, Host file sharing, Extensions etc.)
Docker Desktop Linux VM
Containers
Docker Engine
Runtimes
VM Lifecycle, networking, host file sharing, etc.
Linux Kernel
Host Machine (Win / Mac / Linux)

```
$ docker run <some-image>
```

Hardened Docker Desktop Features

Feature	Description	Mitigated Threats
Settings Management	IT admins can preset & lock security settings on Docker Desktop	Misconfiguration
Enhanced Container Isolation (ECI)	• Runs containers unprivileged (always) • Prevents breaches to Docker Desktop Linux VM • Transparent to developers (use containers as usual)	Supply chain attacks Malware Misconfiguration
Registry Access Management	Restrict container registries accessible by developers	Supply chain attacks Malware
Image Access Management	Restricts container image types (official, verified, etc.)	
Air Gapped Containers	Restricts containers from accessing network resources (e.g., limiting where data can be uploaded to or downloaded from).	Malware Lateral movement

docker

Settings Management

Settings Management (Admin)

Allows IT admins to preset & lock security settings on Docker Desktop.

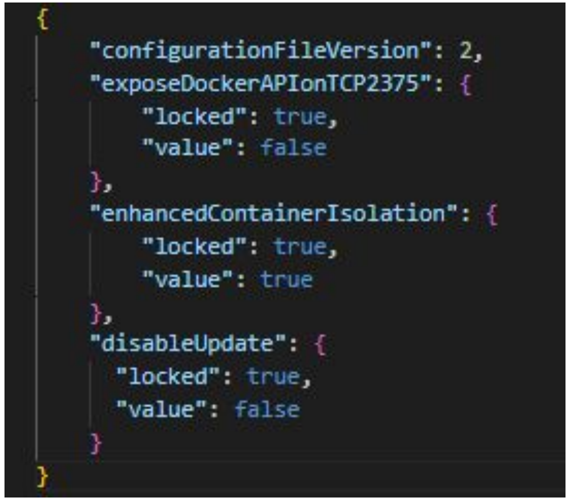
IT Admin

1) Configures locked settings via **admin-settings.json** file.

1) This file is in a restricted folder in the developer's machine.

1) Accessing it requires admin privileges on the machine (developer must not have admin privileges)

Key Requirement



```
{
    "configurationFileVersion": 2,
    "exposeDockerAPIOnTCP2375": {
        "locked": true,
        "value": false
    },
    "enhancedContainerIsolation": {
        "locked": true,
        "value": true
    },
    "disableUpdate": {
      "locked": true,
      "value": false
    }
}
```

Mac: /Library/Application\ Support/com.docker.docker/admin-settings.json
Windows: C:\ProgramData\DockerDesktop\admin-settings.json

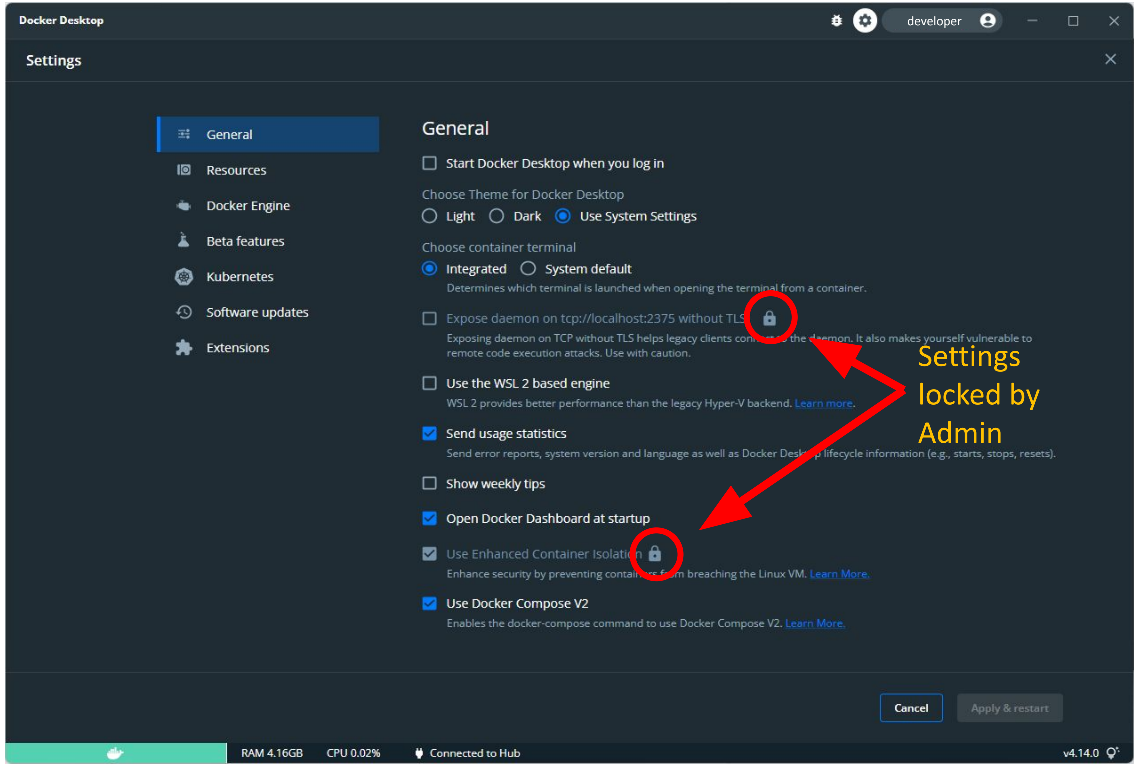
Linux: /usr/share/docker-desktop/admin-settings.json

Settings Management (Developer)

Developer

Upon starting Docker Desktop, admin settings are locked and can't be changed.

(Settings not configured by admin are not affected).



Docker Desktop — developer

Settings

- General
- Resources
- Docker Engine
- Beta features
- Kubernetes
- Software updates
- Extensions

General

[ ] Start Docker Desktop when you log in

Choose Theme for Docker Desktop
Light Dark [✓ Use System Settings]

Choose container terminal
[✓ Integrated] System default
Determines which terminal is launched when opening the terminal from a container.

[ ] Expose daemon on tcp://localhost:2375 without TLS 🔒
Exposing daemon on TCP without TLS helps legacy clients connect to the daemon. It also makes yourself vulnerable to remote code execution attacks. Use with caution.

[ ] Use the WSL 2 based engine
WSL 2 provides better performance than the legacy Hyper-V backend. Learn more.

[✓] Send usage statistics
Send error reports, system version and language as well as Docker Desktop lifecycle information (e.g., starts, stops, resets).

[ ] Show weekly tips

[✓] Open Docker Dashboard at startup

[✓] Use Enhanced Container Isolation 🔒
Enhance security by preventing containers from breaching the Linux VM. Learn More.

[✓] Use Docker Compose V2
Enables the docker-compose command to use Docker Compose V2. Learn More.

Cancel   Apply & restart

RAM 4.16GB   CPU 0.02%   Connected to Hub   v4.14.0

docker

Registry and Image Access Management

Registry and Image Access Management

- Default Registry Behavior
  - Can push/pull from any registry
- Default Image Pull Behavior
  - Can pull all images from Docker Hub

- Registry Access Management Enabled
  - Only able to access named registries
  - Can exclude Docker Hub
- Image Access Management Enabled
  - Restricted to classes of images from Docker Hub

Registry and Image Access Management

NOTE: Relies on Docker Hub as a control plane.

IT Admin

1) Signs-in to Docker Hub as "Org Owner".

1) Configures allowed registries and image types.

1) Configures Docker Desktop on developer machines to force sign-in to Docker Hub (registry.json)

Developer

1) Starts Docker Desktop and signs-in to Docker Hub

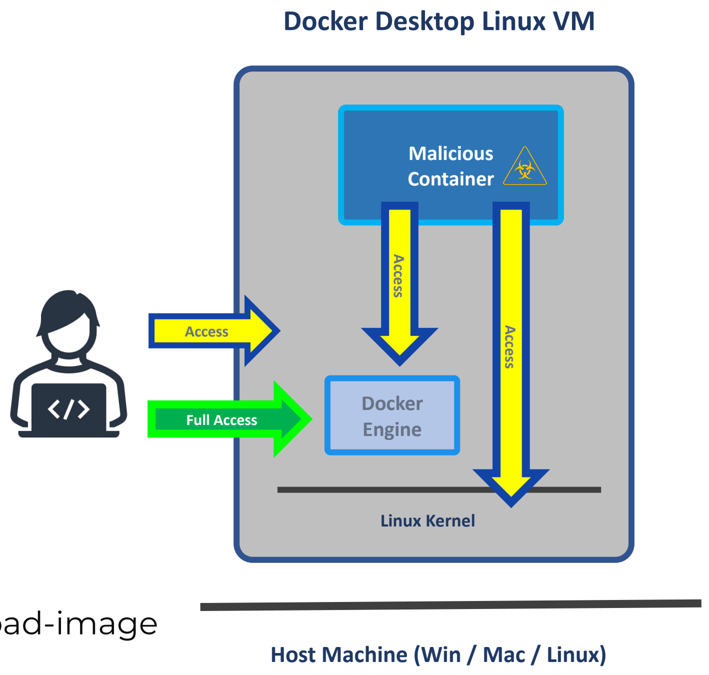
1) Docker Desktop now restricted per registry & image access policy.

1) Developer can't change this locally or on Docker Hub.

Enhanced Container Isolation (ECI)

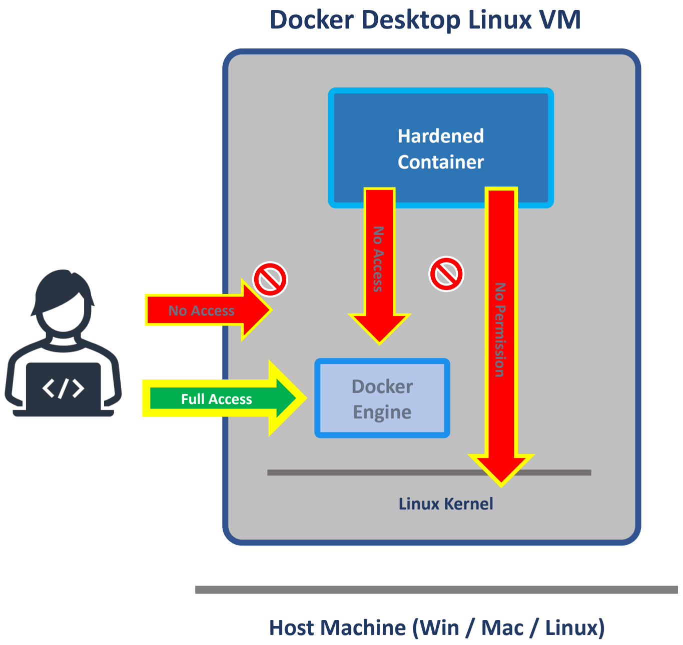
Why ECI?

- Docker Desktop uses a VM
  - Isolates Linux env from host
  - Runs Docker Engine and Containers
- Baseline security is open
  - Containers can run as root
  - Containers can access the vm
    - Kernel
    - Filesystem
    - Docker Engine
- This can cause issues:
  - docker run --privileged bad-image
  - docker run –v /var/run/docker.sock:/var/run/docker.sock bad-image



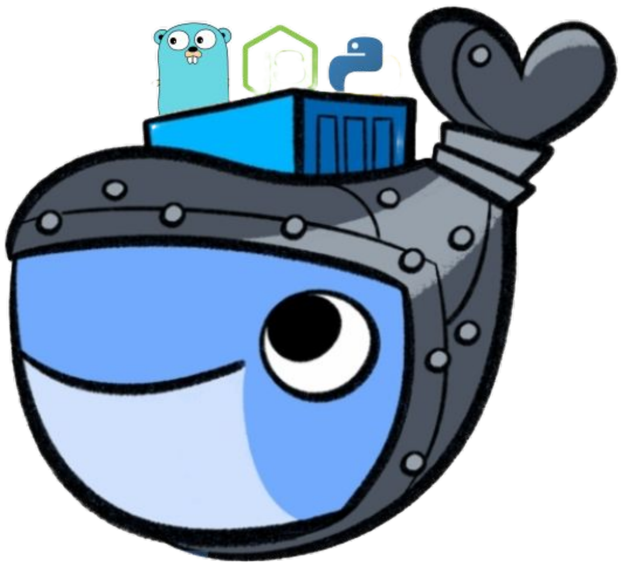
With ECI

- Per-container Linux User Namespace
- Restricts sensitive VM mounts
  - Can “allowlist” problematic mounts
  - Such as the Docker socket
- Sensitive syscall trapping/vetting
- Filesystem ID remapping
- Emulation of sysfs and procfs in container
- All containers run “rootless”
  - Use of “sysbox” runtime
  - Can run most “privileged” workloads
- Engine runs rootful in VM
- VM is hardened
  - Console is protected
- Developers continue work as usual
  - No special commands, processes, etc



ECI at a glance

- All containers run unprivileged (**Linux user-namespace**).
- Even "—privileged" containers are protected.
- Can't mount sensitive VM files into the container.
- Can't access Docker Engine from inside a container.
- Stronger cross-container isolation (per-container user-namespace)
- Sensitive syscalls by containers are trapped and vetted (e.g., mount).
- /proc and /sys inside container are partially emulated for extra isolation.



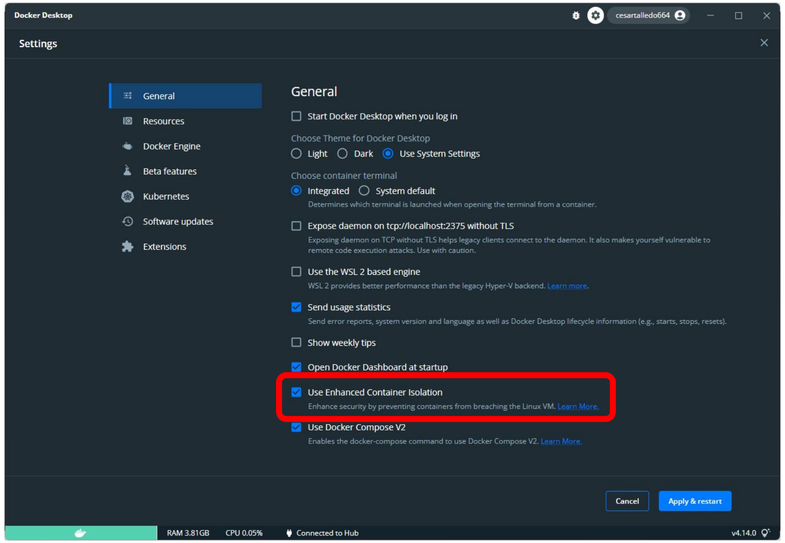
ECI Limitations

Restriction / Limitation	Status
Docker “--pid=host” and “--net=host” disallowed	No plan to change.
Most privileged containers will work (even though they run rootless), but some won’t (e.g., containers that change kernel configs).	No plan to change.
On Windows WSL, ECI hardens containers but does not prevent developers from accessing the Docker Desktop VM internals.	No plan to change (it’s a WSL limitation).
Docker Desktop Extension containers are not yet protected.	TBD.

Enabling ECI

ECI can be enabled by Developers or Admins. Admins can also lock it.

Developer



General

- [ ] Start Docker Desktop when you log in

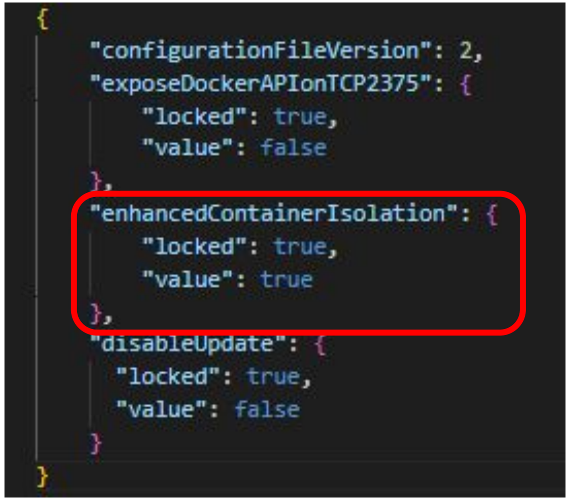
Choose Theme for Docker Desktop
Light Dark [✓ Use System Settings]

Choose container terminal
[✓ Integrated] System default
Determines which terminal is launched when opening the terminal from a container.

- [ ] Expose daemon on tcp://localhost:2375 without TLS
  Exposing daemon on TCP without TLS helps legacy clients connect to the daemon. It also makes yourself vulnerable to remote code execution attacks. Use with caution.
- [ ] Use the WSL 2 based engine
  WSL 2 provides better performance than the legacy Hyper-V backend. Learn more.
- [x] Send usage statistics
  Send error reports, system version and language as well as Docker Desktop lifecycle information (e.g., starts, stops, resets).
- [ ] Show weekly tips
- [x] Open Docker Dashboard at startup
- [x] Use Enhanced Container Isolation
  Enhance security by preventing containers from breaching the Linux VM. Learn More.
- [x] Use Docker Compose V2
  Enables the docker-compose command to use Docker Compose V2. Learn More.

IT Admin

admin-settings.json



```
{
    "configurationFileVersion": 2,
    "exposeDockerAPIOnTCP2375": {
        "locked": true,
        "value": false
    },
    "enhancedContainerIsolation": {
        "locked": true,
        "value": true
    },
    "disableUpdate": {
      "locked": true,
      "value": false
    }
}
```

Docker ECI vs Other OCI Dev Tooling

Malware Threat	Feature that mitigates it	Other Container Dev Tools	Docker Desktop	Hardened Docker Desktop
Host Attack	• Run containers in a Linux VM • Can restrict host file sharing	✔	✔	✔
Linux VM Attack	• Linux user-namespace on all containers • /proc and /sys partial emulation • Sensitive syscalls trapping • Can't mount VM dirs into container			✔
Container Engine Attack	• Can't mount Docker socket in container • Can't mount VM dirs into container			✔
Cross-container attack	• Per-container Linux User Namespace mappings			✔

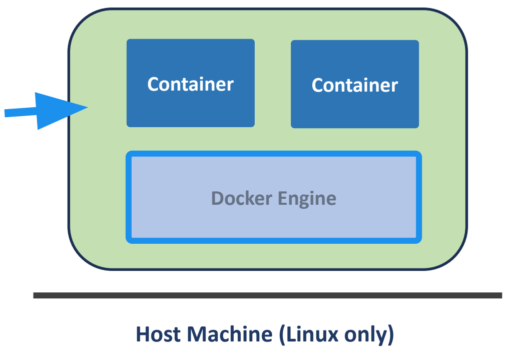
Docker Desktop ECI vs Rootless Docker

Category	Docker Engine	Rootless Docker	Docker Desktop with ECI
Supported Hosts	Linux	Linux	Mac/Win/Linux
Docker Engine isolation from host	None	User-Namespace	Virtual Machine
Container isolation from host	Namespaces (except user-ns)	User-Namespace (shared with all other containers and Docker Engine)	User-Namespace (per each container)
Can mount Docker Socket to container	Yes	Yes	Trusted images only.
Vets sensitive syscalls in container (e.g., mount, unmount)	No	No	Yes

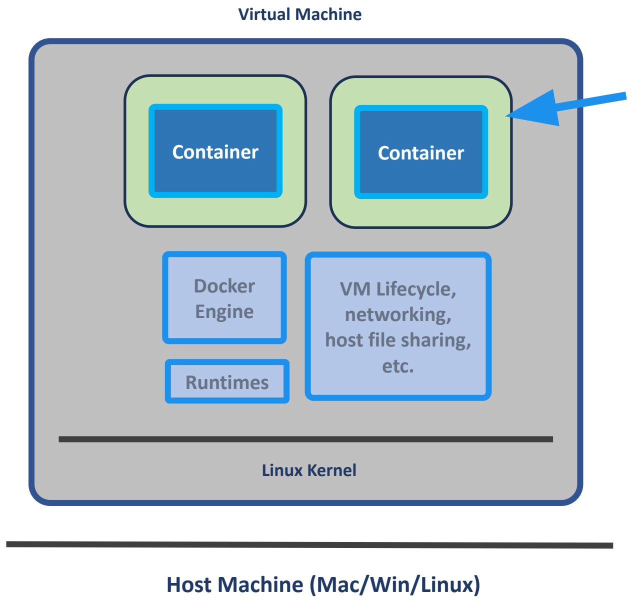
ECI and Rootless Docker

Rootless Docker

User-namespace is shared by containers and engine.



Docker Desktop + ECI

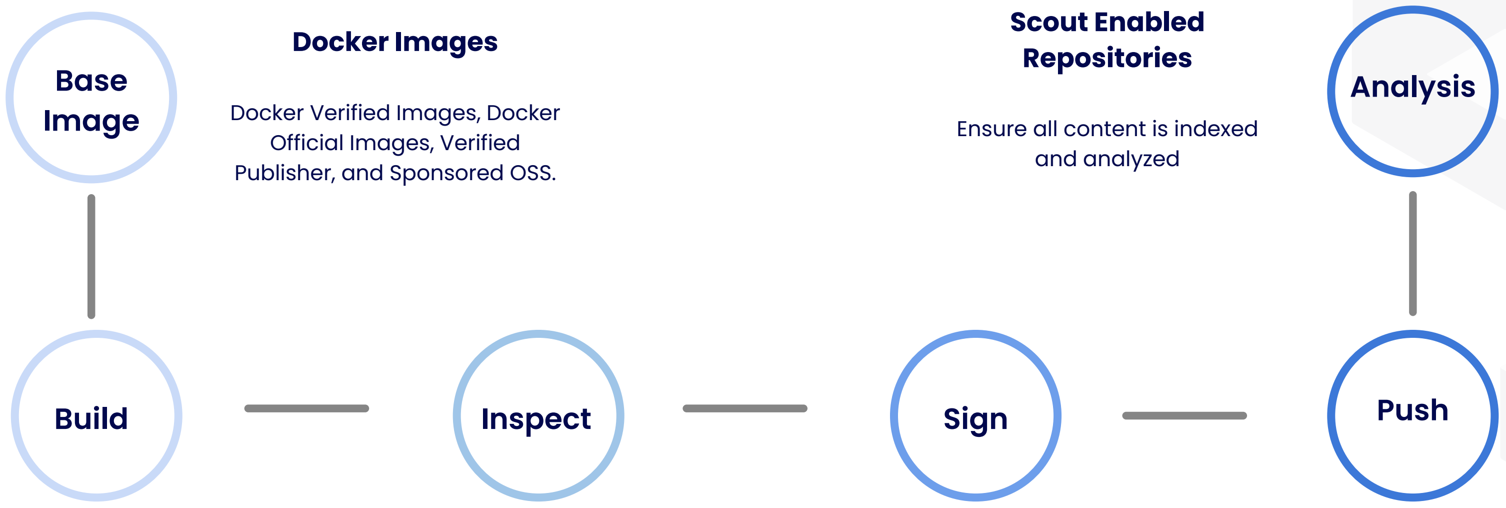

Virtual Machine
Container
Container
Docker Engine
Runtimes
VM Lifecycle, networking, host file sharing, etc.
Linux Kernel
Host Machine (Mac/Win/Linux)

Per container user-namespace (more isolation)

docker

Docker Scout
“Supply chain and policies”

Docker Secure Supply Chain



Docker Images

Docker Verified Images, Docker Official Images, Verified Publisher, and Sponsored OSS.

Scout Enabled Repositories

Ensure all content is indexed and analyzed

Docker Buildx

HIgh fidelity SBOM and attestation, marrying build to SBOM and related data

Docker Scout

SBOM generation, image analysis, policy checking, and remediation recommendations

Docker Trust

Signing of the image and all related metadata

Docker Push

Making the image available to users

What's in this software artifact?

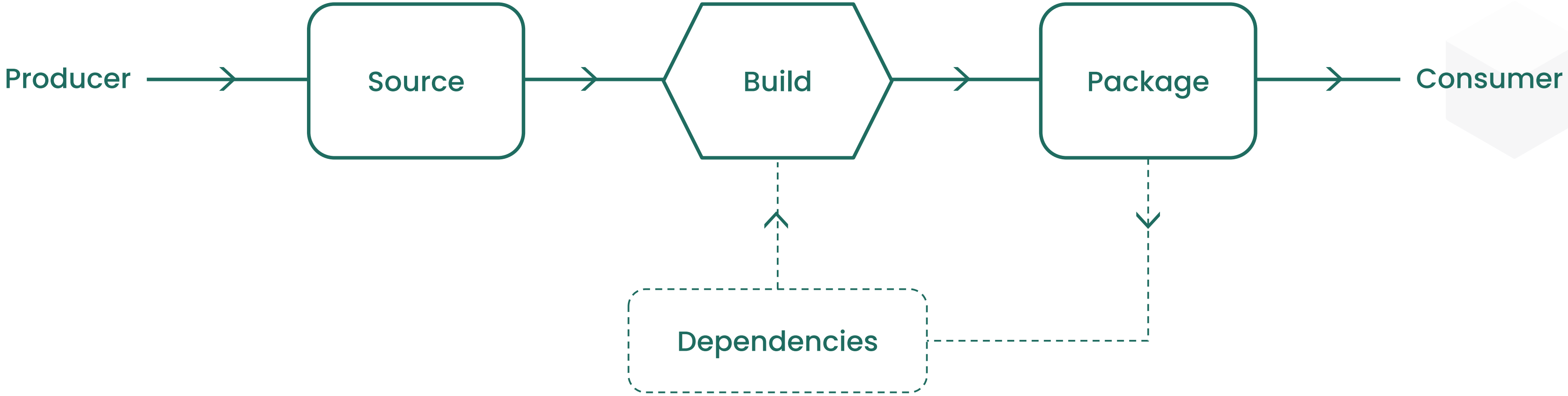
A software bill of materials – **SBOM** – is an *attestation* describing the contents of a software artifact

Where has it come from?

Provenance is an *attestation* about the history of an artifact – where it came from, who produced it, and how

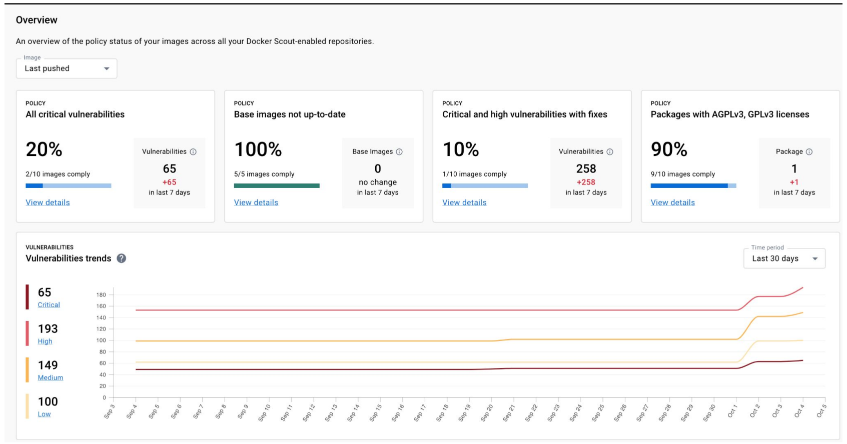
Can I verify the attestation source?

An *attestation* that has been **signed** with a digital signature verifies the source and helps assess trustworthiness



Adapted from slsa.dev/spec/v1.0/threatsoverview

Scout Dashboard


Overview

An overview of the policy status of your images across all your Docker Scout-enabled repositories.

Image: Last pushed

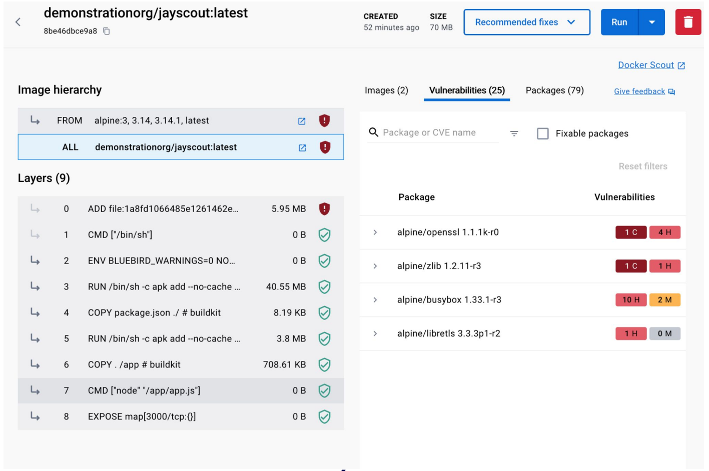
POLICY			
All critical vulnerabilities	20% — 2/10 images comply	Vulnerabilities 65, +65 in last 7 days	View details
Base images not up-to-date	100% — 5/5 images comply	Base Images 0, no change in last 7 days	View details
Critical and high vulnerabilities with fixes	10% — 1/10 images comply	Vulnerabilities 258, +258 in last 7 days	View details
Packages with AGPLv3, GPLv3 licenses	90% — 9/10 images comply	Package 1, +1 in last 7 days	View details

VULNERABILITIES
Vulnerabilities trends

Time period: Last 30 days

65	Critical
193	High
149	Medium
100	Low

Docker Desktop


demonstrationorg/jayscout:latest
8be46dbce9a8

CREATED 52 minutes ago SIZE 70 MB

Image hierarchy

FROM	alpine:3, 3.14, 3.14.1, latest	
ALL	demonstrationorg/jayscout:latest	

Layers (9)

0	ADD file:1a8fd1066485e1261462e...	5.95 MB
1	CMD ["/bin/sh"]	0 B
2	ENV BLUEBIRD_WARNINGS=0 NO...	0 B
3	RUN /bin/sh -c apk add --no-cache ...	40.55 MB
4	COPY package.json ./ # buildkit	8.19 KB
5	RUN /bin/sh -c apk add --no-cache ...	3.8 MB
6	COPY . /app # buildkit	708.61 KB
7	CMD ["node" "/app/app.js"]	0 B
8	EXPOSE map[3000/tcp:{}]	0 B

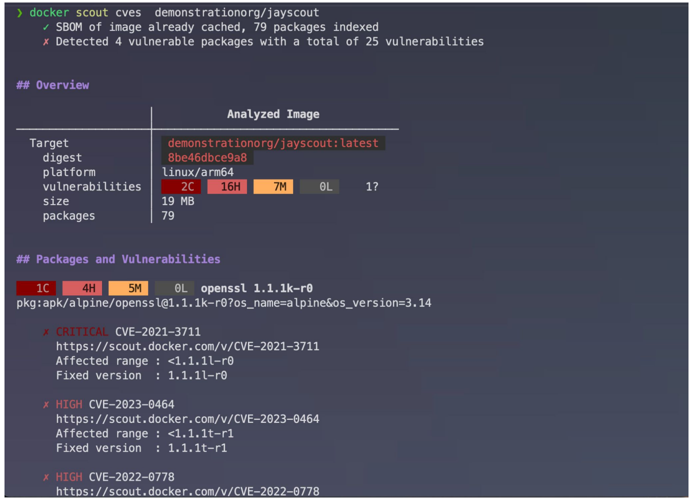
Images (2) Vulnerabilities (25) Packages (79)

Package or CVE name ☐ Fixable packages

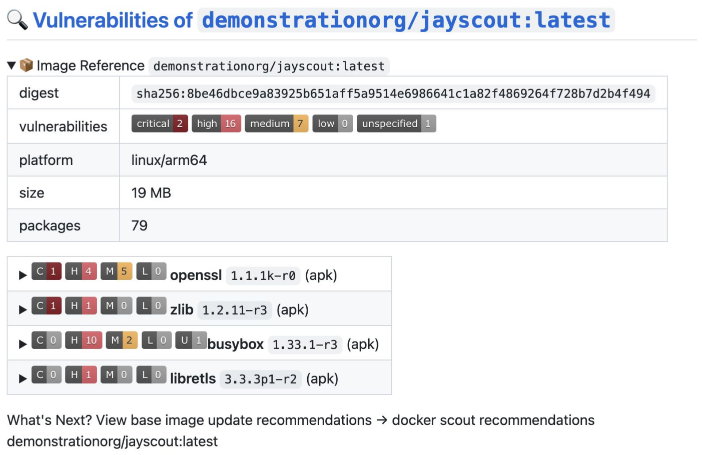
Reset filters

Package	Vulnerabilities
alpine/openssl 1.1.1k-r0	1 C 4 H
alpine/zlib 1.2.11-r3	1 C 1 H
alpine/busybox 1.33.1-r3	10 H 2 M
alpine/libretls 3.3.3p1-r2	1 H 0 M

Scout CLI


```
❯ docker scout cves  demonstrationorg/jayscout
    ✓ SBOM of image already cached, 79 packages indexed
    ✗ Detected 4 vulnerable packages with a total of 25 vulnerabilities


## Overview

                    │          Analyzed Image
────────────────────┼──────────────────────────────
  Target            │  demonstrationorg/jayscout:latest
    digest          │  8be46dbce9a8
    platform        │ linux/arm64
    vulnerabilities │    2C    16H    7M    0L    1?
    size            │ 19 MB
    packages        │ 79


## Packages and Vulnerabilities

   1C    4H    5M    0L  openssl 1.1.1k-r0
pkg:apk/alpine/openssl@1.1.1k-r0?os_name=alpine&os_version=3.14

    ✗ CRITICAL CVE-2021-3711
      https://scout.docker.com/v/CVE-2021-3711
      Affected range : <1.1.1l-r0
      Fixed version  : 1.1.1l-r0

    ✗ HIGH CVE-2023-0464
      https://scout.docker.com/v/CVE-2023-0464
      Affected range : <1.1.1t-r1
      Fixed version  : 1.1.1t-r1

    ✗ HIGH CVE-2022-0778
      https://scout.docker.com/v/CVE-2022-0778
```

CI/CD


🔍 Vulnerabilities of `demonstrationorg/jayscout:latest`

▼ 📦 Image Reference `demonstrationorg/jayscout:latest`

digest	`sha256:8be46dbce9a83925b651aff5a9514e6986641c1a82f4869264f728b7d2b4f494`
vulnerabilities	critical 2 high 16 medium 7 low 0 unspecified 1
platform	linux/arm64
size	19 MB
packages	79

▶ C 1 H 4 M 5 L 0 **openssl** `1.1.1k-r0` (apk)

▶ C 1 H 1 M 0 L 0 **zlib** `1.2.11-r3` (apk)

▶ C 0 H 10 M 2 L 0 U 1 **busybox** `1.33.1-r3` (apk)

▶ C 0 H 1 M 0 L 0 **libretls** `3.3.3p1-r2` (apk)

What's Next? View base image update recommendations → docker scout recommendations demonstrationorg/jayscout:latest

Airgap / Highly Regulated Considerations and Limitations

Current Limitations: Authentication

- Authentication currently requires connectivity to Docker Hub
  - Project underway to address this
  - Will decouple Docker Hub from auth process
    - License Server
    - Auth Once model

- Usage insights requires authentication and data transfer
  - Will not be available to airgapped/regulated customers
  - Potential for local data gathering in future

- Other Features Requiring Authentication
  - Docker Debug
  - Docker Compose Bridge
  - Docker Hardened Desktop

Current Limitations: Docker Build Cloud

- Docker Build Cloud Implemented as SaaS
  - Requires Auth
  - Requires External Access

- Potential Future State
  - Self-hosted model
  - GOV Cloud model

- Current Workarounds
  - Docker buildkit remote builders (self-managed)

Current Limitations: Docker Scout

- Docker Scout Implemented as SaaS
  - Requires Auth
  - Requires External Access

- Potential Future State
  - Self-hosted model
  - GOV Cloud model

- Current Workarounds
  - None

Current Limitations: Docker Harmonia

- Docker Harmonia Implemented as SaaS
  - Requires Auth
  - Requires External Access

- Potential Future State
  - Self-hosted model
  - GOV Cloud model

- Current Workarounds
  - Use remote docker contexts (self-managed)

docker

Questions and Answers