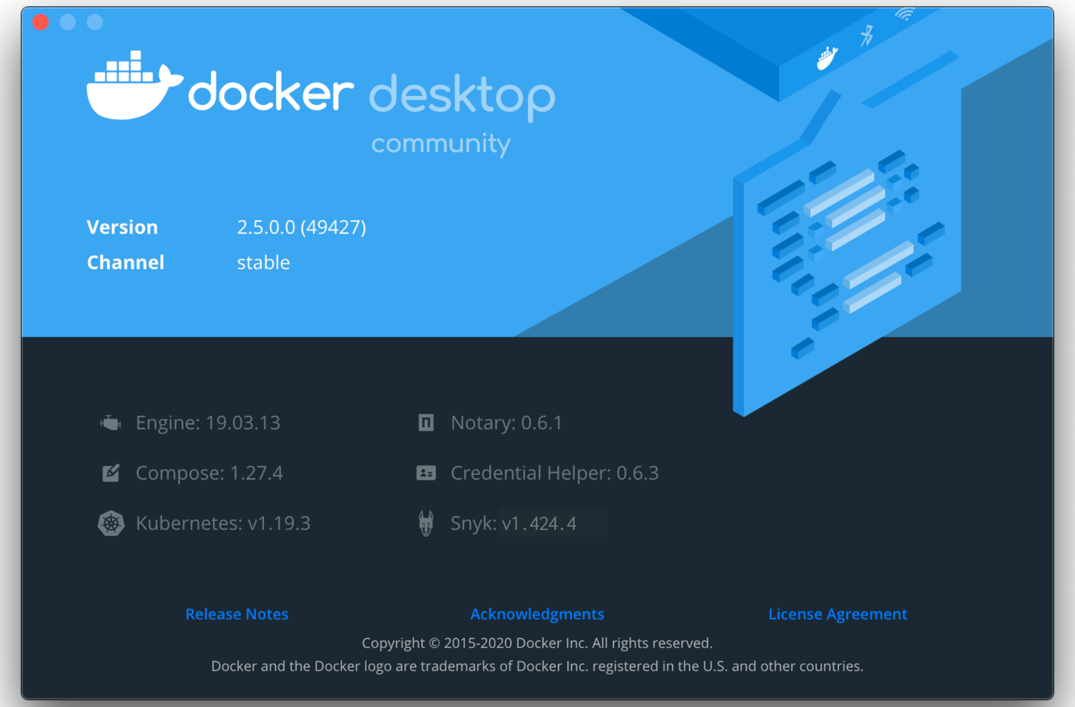
Docker Vulnerability Scanning CLI Cheat Sheet

Getting Started

Docker Desktop now includes container vulnerability scanning, powered by Snyk!

Download and install the latest version of Docker Desktop.

Once installed, login to your Docker Hub account, and you can verify the vulnerability scanning is installed two ways. First, the "About Docker Desktop" screen will now show the Snyk version:



```
Version    2.5.0.0 (49427)
Channel    stable

Engine: 19.03.13        Notary: 0.6.1
Compose: 1.27.4         Credential Helper: 0.6.3
Kubernetes: v1.19.3     Snyk: v1.424.4
```

Release Notes   Acknowledgments   License Agreement

Copyright © 2015-2020 Docker Inc. All rights reserved.
Docker and the Docker logo are trademarks of Docker Inc. registered in the U.S. and other countries.

You can also check the Snyk version at the command line:

```
$ docker scan --version
Version:    v0.3.3
Git commit: 1bab774
Provider:   Snyk (1.424.4)
```

You can run up to ***10 tests per month*** without any additional configuration. To unlock additional free monthly tests, sign up for a free Snyk account, if you do not already have one, and authenticate in the Docker client.

Once you have a Snyk account you can authenticate in the Docker CLI using either of the commands below.

If you want to use the --token method, you can manage Snyk API tokens in the Snyk console under "Settings --> Service accounts".

```
# opens a browser window to authenticate:
$ docker scan --login

# Login directly using your Snyk API token:
$ docker scan --login --token 1234567-abcd-456
```

Common Docker Scanning Options

Run a single test on an image tagged myapp:mytag:

```
$ docker scan myapp:mytag
```

Use the Dockerfile to generate a more detailed analysis:

```
$ docker scan myapp:mytag --file path/to/Dockerfile
```

For popular official images on Docker Hub, this will provide base image recommendations and alternate base images that can help reduce vulnerabilities.

In addition, if `RUN` commands in the Dockerfile install packages that introduce vulnerabilities, that Dockerfile command is provided as part of the vulnerability details.

Ignore any vulnerabilities from the base image. The `--exclude-base` option requires the `--file` option.

```
$ docker scan myapp:mytag --exclude-base \
  --file path/to/Dockerfile
```

Provides a package dependency tree for the image, in addition to the vulnerability findings.

```
$ docker scan myapp:mytag --dependency-tree
```

Output scan results in JSON format. See the *Vulnerability Data* section below for more examples:

```
$ docker scan myapp:mytag --json
```

Vulnerability Data & Advanced CLI Usage

Using the --json output is a powerful way to filter and display scan results. A subset of the vulnerability output is shown below with some examples using the `jq` utility to filter results.

```
{
   "title": "Out-of-bounds Read",
   "packageName": "curl",
   "language": "linux",
   "packageManager": "alpine:3.7",
   "description": "## Overview\nlibcurl versions from...
   "identifiers": {
     "ALTERNATIVE": [],
     "CVE": [
       "CVE-2019-3823"
     ],
     "CWE": [
       "CWE-125"
     ]
   },
   "severity": "high",
   "cvssScore": 7.5,
   "CVSSv3": "CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H",
   "creationTime": "2020-07-21T16:54:36.291584Z",
   "modificationTime": "2020-07-23T10:26:06.499683Z",
   "publicationTime": "2019-02-06T20:29:00Z",
   "disclosureTime": "2019-02-06T20:29:00Z",
   "id": "SNYK-ALPINE37-CURL-343582",
   "nvdSeverity": "high",
   "semver": {
     "vulnerable": [
       "<7.61.1-r2"
     ]
   },
   "from": [
     "docker-image|purpledobie/utilities@curl.alp37",
     "curl/libcurl@7.60.0-r1"
   ],
   "name": "curl/libcurl",
   "version": "7.60.0-r1",
   "nearestFixedInVersion": "7.61.1-r2",
   "dockerfileInstruction": "RUN apk add --no-cache curl",
   "dockerBaseImage": "alpine:3.7"
 }
```

Vulnerability Key	Description
`packageName`	Simple name of the top-level package
`severity`	Severity rating based on CVSS score
`id`	Snyk-specific vulnerability to look up vulnerabilties in Snyk's database
`name`	Specific name of the vulnerable binary
`version`	Version installed in the container image
`nearestFixedInVersion`	Minimum version required to fix the vulnerability
`dockerfileInstruction`	Line in the Dockerfile that installed the vulnerable package (requires --file)
`dockerBaseImage`	Parent image detected in the scan (requires --file)

Shown only high severity vulnerabilities from layers *other than* the base image:

```
$ docker scan myapp:mytag --exclude-base   \
  --file path/to/Dockerfile --json       | \
  jq '[.vulnerabilities[] | select(.severity=="high")]'
```

High severity vulnerabilities with an CVSSv3 network attack vector:

```
$ docker scan myapp:mytag --json              |  \
  jq '[.vulnerabilities[]                     |
  select(.CVSSv3 | contains("AV:N")))         |
  select(.severity=="high")]'
```

High severity vulnerabilities with a fix available:

```
$ docker scan myapp:mytag --json     |  \
  jq '[.vulnerabilities[]            |
  select(.nearestFixedInVersion)     |
  select(.severity=="high")]'
```

The über example! Instead of listing each vulnerability, this will show only one result per *vulnerable package* and that entry will be the one requiring the most recent fix:

```
$ docker scan myapp:mytag --file Dockerfile --json | \
  jq '[.vulnerabilities[]               |
  select(.severity == "high")           | # high severity
  select(.nearestFixedInVersion) ]      | # with a fix
  group_by(.packageName)[]              | # group by affected package
  sort_by(.nearestFixedInVersion)       | # sorted by fix version
  .[0]                                  | # select the first entry
  {packageName, dockerfileInstruction, version,
nearestFixedInVersion}'
```