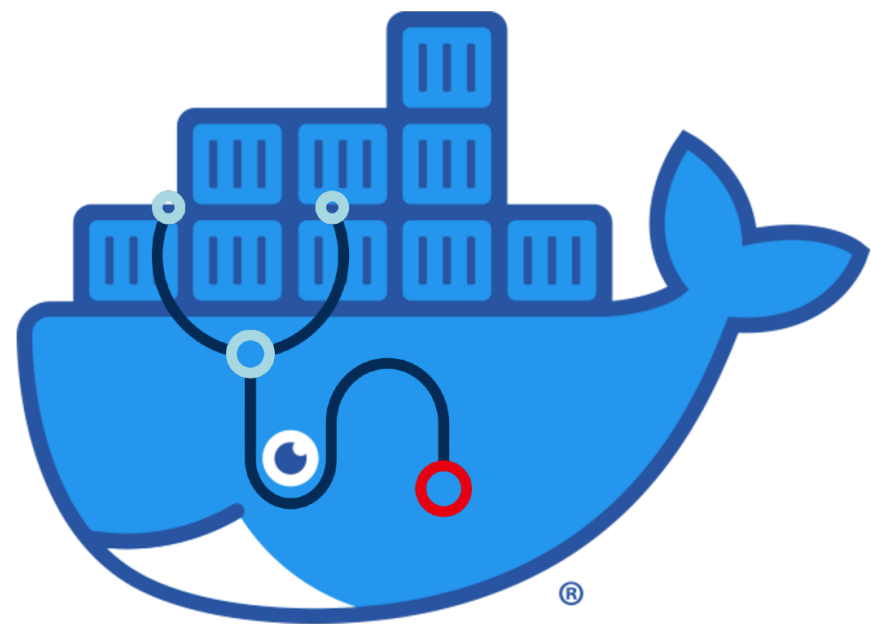


# DockerENT

Open Source tool & framework to Analyse security issues with running containers.

**Rohit Sehgal**



# DockerENT

Open Source tool to Analyse running container.

00

**Who-am-I**

01

**What is DockerENT**

A quick overview of tool, and why do we need it.

02

**Checks**

List of checks which DockerENT can perform

03

**Technical Details**

Create your own plugins.

04

**Demo**

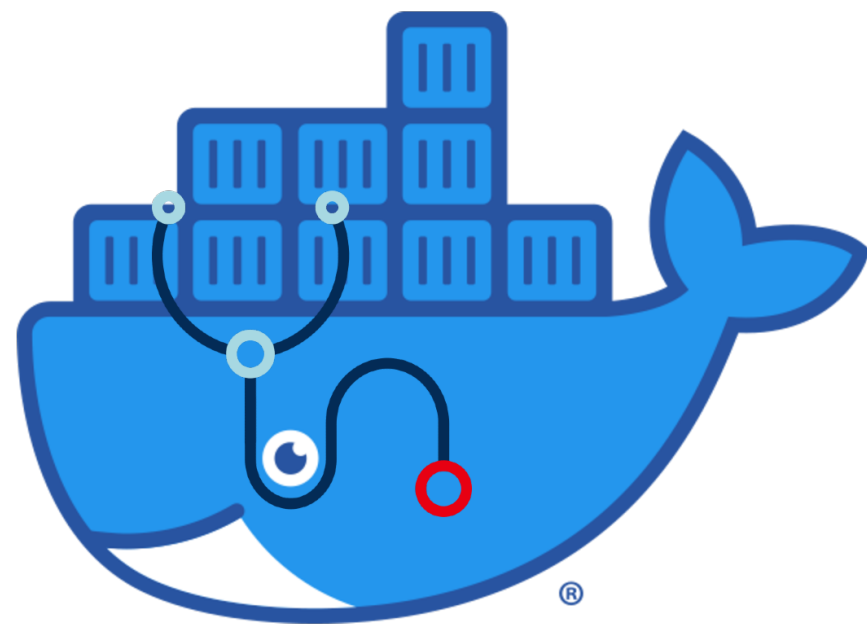
05

**Future Work**

06

**Q&A**

It's not the end.



## Analyse running container.

- Security Engineer with ~4 years of Experience, Master from IITK
- Started with Bug Hunting, but now I am on the other side of the table.
- OSCP
- Into : PenTesting, Secure Code Review, Design Review etc.
- Into Development :
  - Open Source
    - TrashEmail - Disposable Email Solution for Telegram.
    - DoT-Proxy - DNS over TLS proxy (WIP)
    - Cloudmarker
    - BinExp
    - k8s-in-30-mins, etc

00

**Who-am-I**

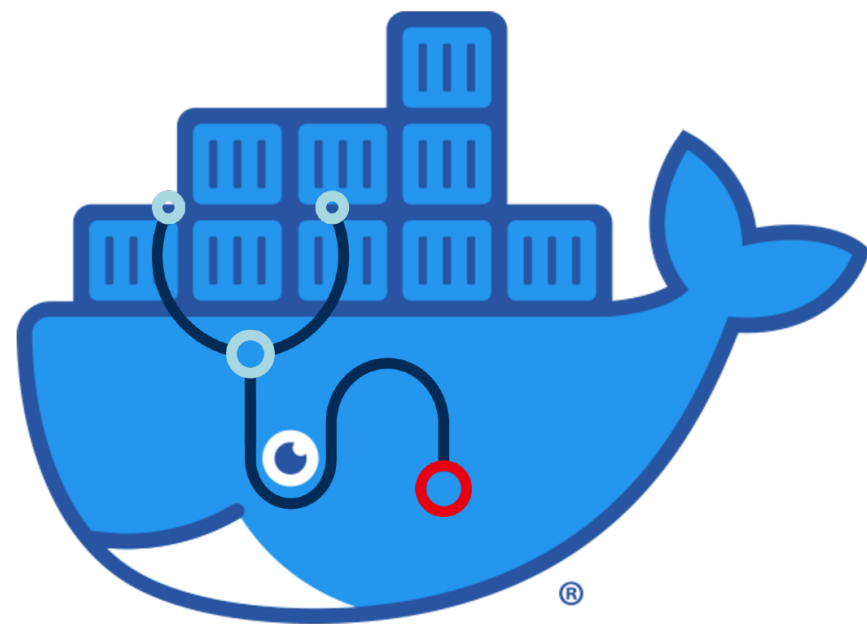
GitHub : [r0hi7](#)

LinkedIn : [r0hitsehgal](#)


Twitter : [sehgal\\_rohit](#)

More @ [rsehgal.in](#)

**DockerENT**



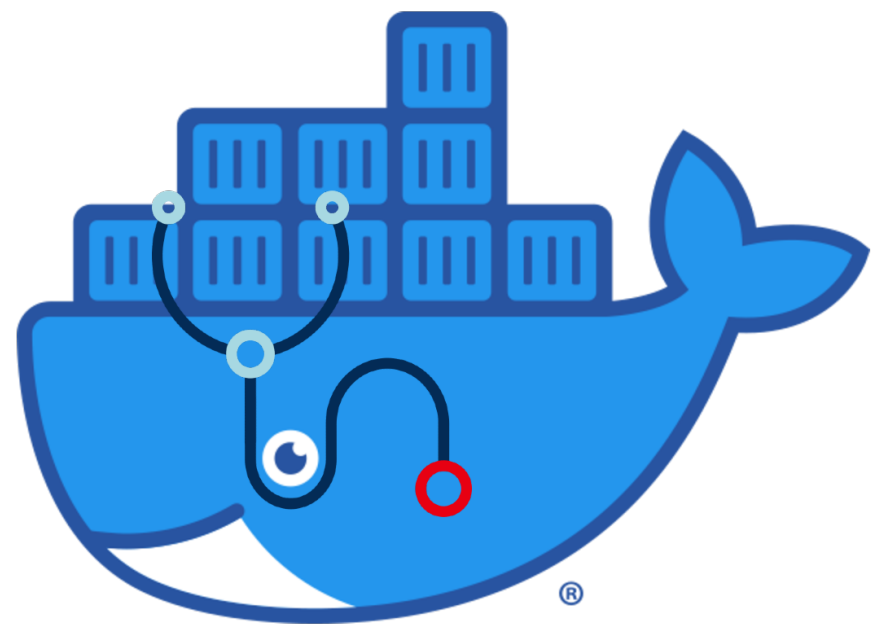
## Analyse running container.

- **Docker** runtime security scanning Tool.
- Framework written in 
  - It's extremely simple to create custom plugins.
  - Use it quickly with exiting plugins.
- Tasks
  - Scan containers running in System with security misconfiguration.
  - Runtime docker level security misconfigurations.
  - Container Namespace level check and misconfigurations.
  - Audit results.
  - Customisable.
- A DevSecOps friendly tool for quick and repetitive checks.
- And Yes there is a Web App along with fully functional CLI.

01

**What is  
DockerENT**

**DockerENT**

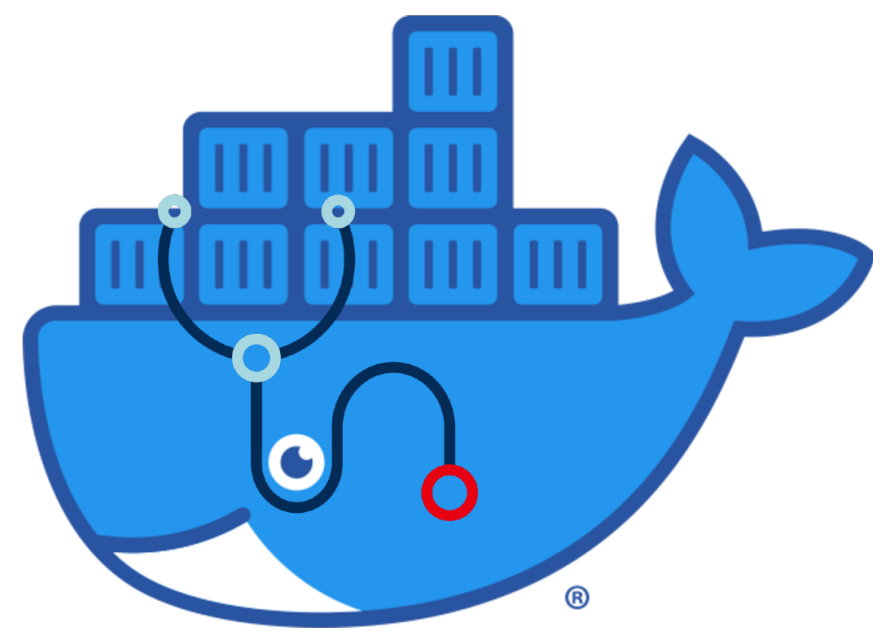


Analyse running container.

01

## What is DockerENT

- **Docker** runtime security scanning Tool.
- Be it K8S or Swarm Cluster, at the end of the day, they are running docker.
  - Fine grain and docker level checks.
- Framework:
  - Similar to but powerful than DockerBench
  - I will talk about this in next slide.



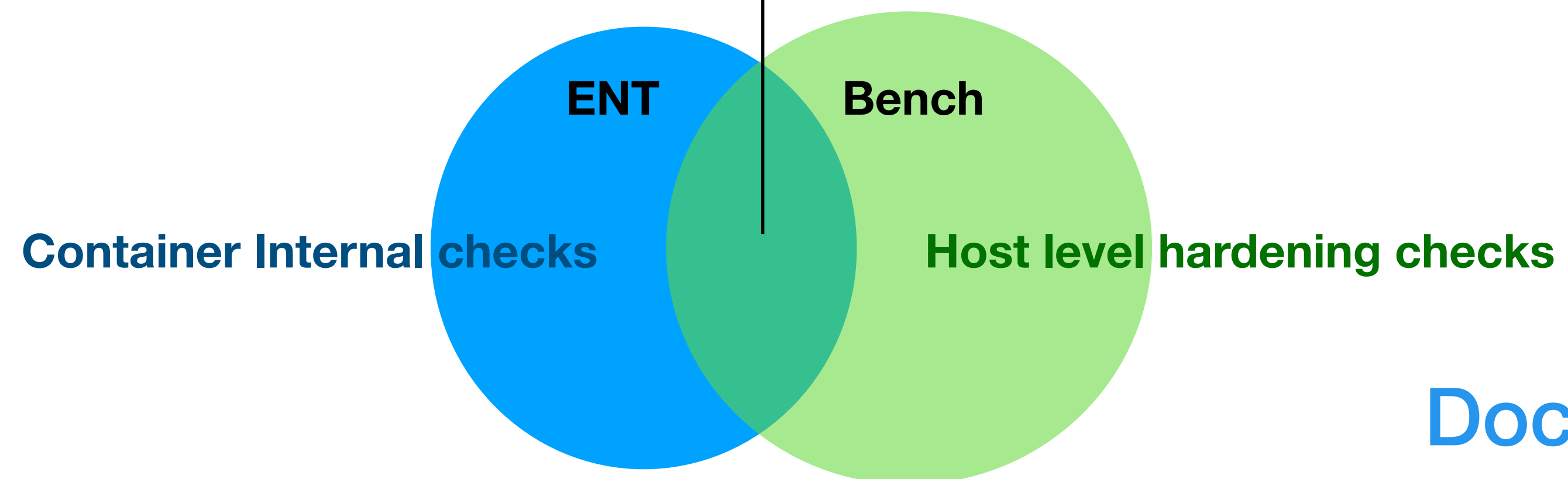
## Analyse running container.

### DockerENT

- \* Performs Runtime configuration checks.
- \* Only performs checks for running containers.
- \* Overlapping scan set with DockerBench
- \* Also Performs checks by executing commands inside docker.

### DockerBench

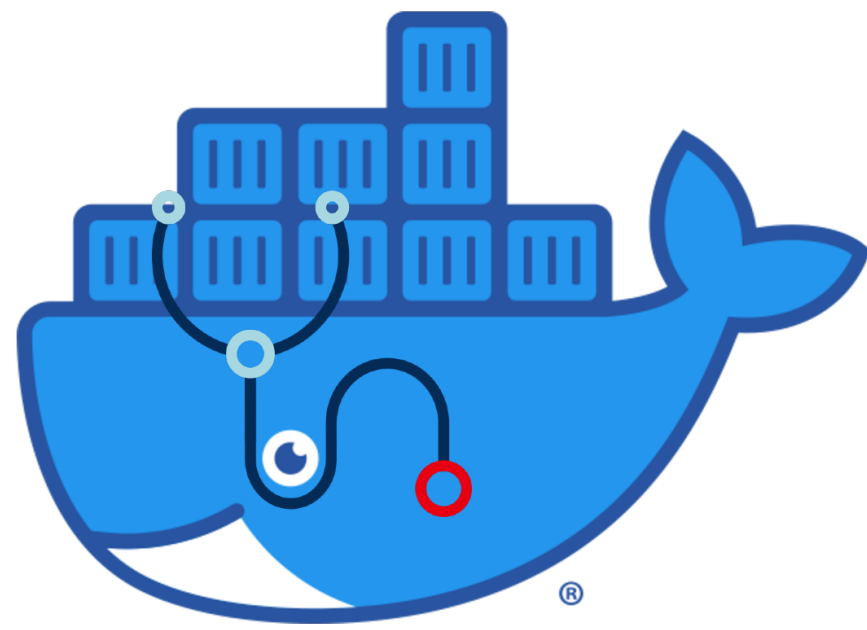
- \* Perform CIS Benchmarks checks.
- \* Performs Container and host level checks.
- \* Cant identify security issues inside running containers.



01

**What is  
DockerENT**

**DockerENT**



# Analyse running container.

## DockerENT

### Install with PIP

```
pip install DockerENT  
  
DockerENT -w  
  
DockerENT --help
```

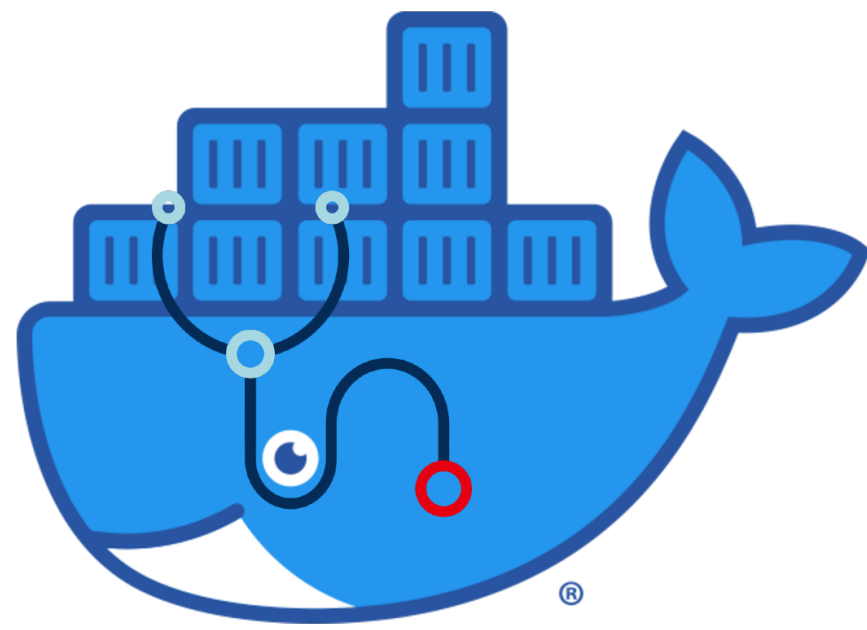
01

## What is DockerENT

### Run From Source

```
git clone https://github.com/r0hi7/DockerENT.git  
  
make deps  
  
make venv  
  
python -m DockerENT --help
```





# Analyse running container.

Most updated list can be found here : <https://github.com/r0hi7/DockerENT#plugins-features>

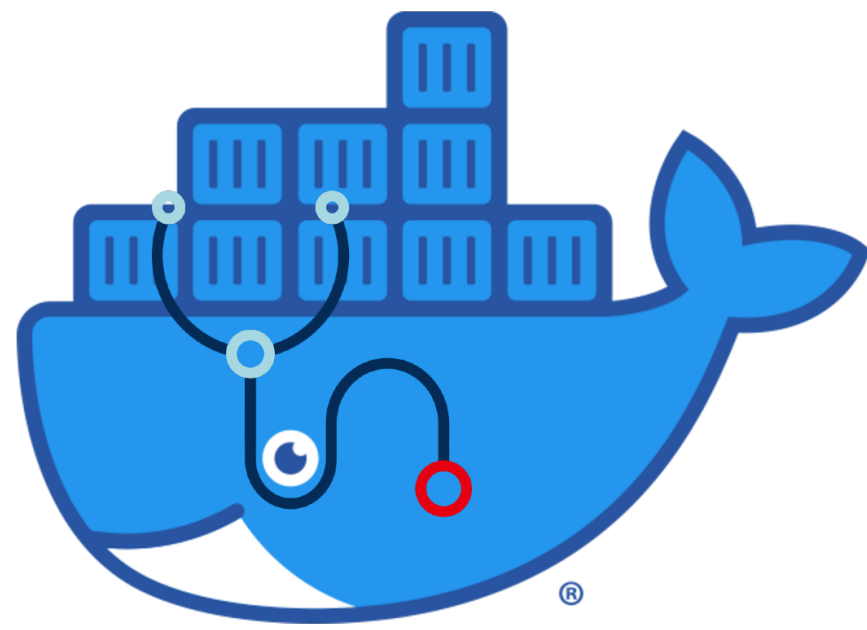
02

## Checks

List of checks which DockerENT can perform

Plugin Name	Plugin File	Feature	Audit
CMD_HISTORY	<a href="#">File</a>	Identify shell history	Root history and User shell history
FILESYSTEM	<a href="#">File</a>	Identify RW File Systems	If RW file systems are present.
NETWORK	<a href="#">File</a>	Identify Network state	Identifies All mapped ports.
PLAINTEST_PASSWORD	<a href="#">File</a>	Identify password in different files	
SECURITY_PROFILES	<a href="#">File</a>	Identify Weak Security Profiles	List Weak security profiles.
USER_INFO	<a href="#">File</a>	Identify user info	List permissions in passwd and other sensitive files
SYSTEM_INFO	<a href="#">File</a>	Identify docker system info	No Audit
FILES_INFO	<a href="#">File</a>	Identify world writeable directories and files	List all such files.





# Analyse running container.

03

## Technical Details

Create your own plugins.

```
_plugin_name = 'demo_plugin'

def scan(container, output_queue, audit=False, audit_queue=None):
    _log.info('Starting {} Plugin ...'.format(_plugin_name))

    res = {}

    result = {
        'test_class': {
            'TEST_NAME': ['good']
        }
    }

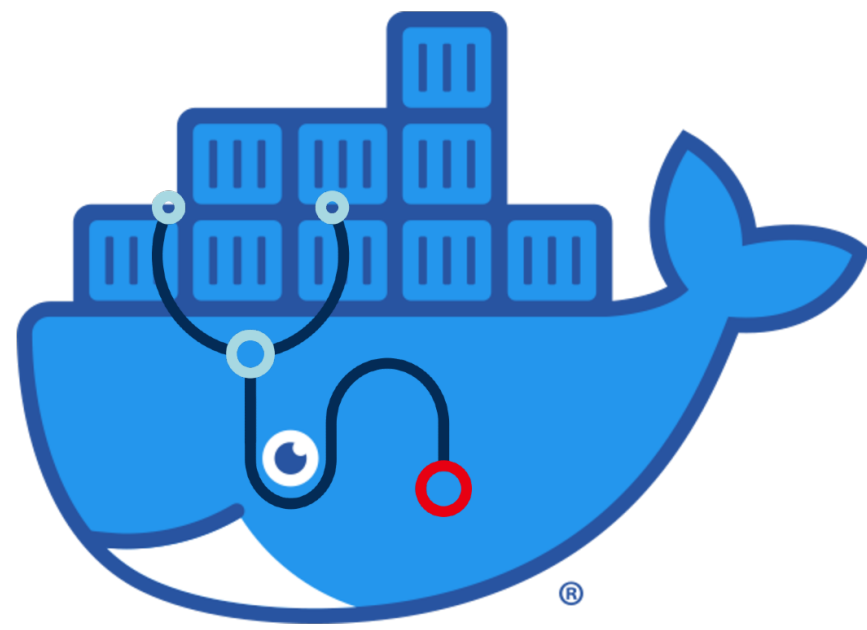
    res[container.short_id] = {
        _plugin_name: result
    }

    # Do something magical.

    _log.info('Completed execution of {} Plugin.'.format(_plugin_name))

    '''Make Sure you put dict of following structure in Q.
    {
        'container_id': {
            'plugin_name': {
                'test_name_demo1': {
                    resultss: []
                },
                'test_name_demo2': {
                    results: []
                }
            }
        }
    }
    ...
    output_queue.put(res)

    if audit:
        _audit(container, res, audit_queue)
```



# Analyse running container.

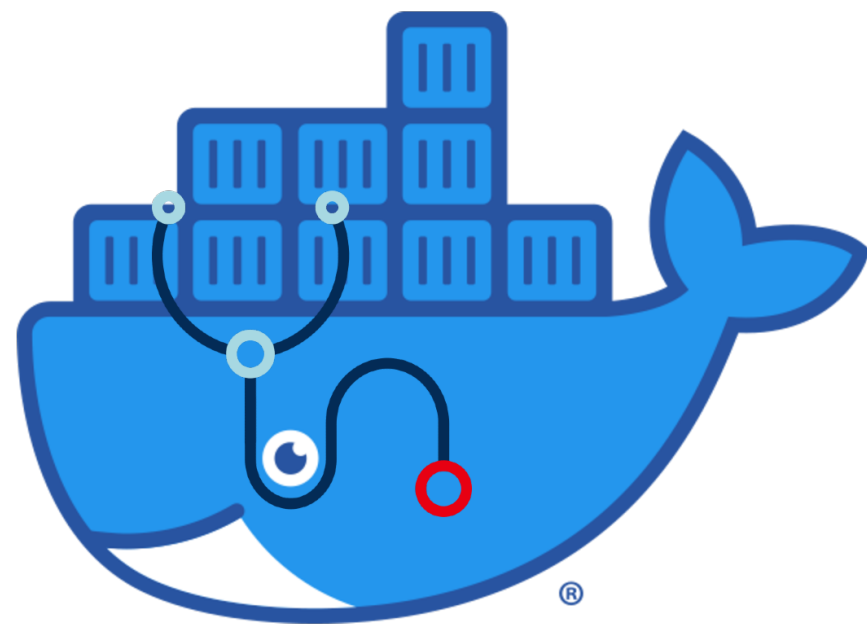
03

## Technical Details

Create your own plugins.

```
def _audit(container, results, audit_queue):  
    '''Make Sure to add dict of following structure to Audit Q  
    res = {  
        "container_id": [  
            "_plugin_name_", WARN/INFO/ERROR, details"  
        ]  
    }  
    ...  
    # Magical logic to perform Audit.  
    audit_queue.put(res)
```

```
_plugin_name = 'demo_plugin'  
  
def scan(container, output_queue, audit=False, audit_queue=None):  
    _log.info('Starting {} Plugin ...'.format(_plugin_name))  
  
    res = {}  
  
    result = {  
        'test_class': {  
            'TEST_NAME': ['good']  
        }  
    }  
  
    res[container.short_id] = {  
        _plugin_name_: result  
    }  
  
    # Do something magical.  
  
    _log.info('Completed execution of {} Plugin.'.format(_plugin_name))  
  
    '''Make Sure you put dict of following structure in Q.  
    {  
        'container_id': {  
            'plugin_name': {  
                'test_name_demo1': {  
                    resultss:[]  
                },  
                'test_name_demo2': {  
                    results: []  
                }  
            }  
        }  
    }  
    ...  
    output_queue.put(res)  
  
    if audit:  
        _audit(container, res, audit_queue)
```



# Analyse running container.

03

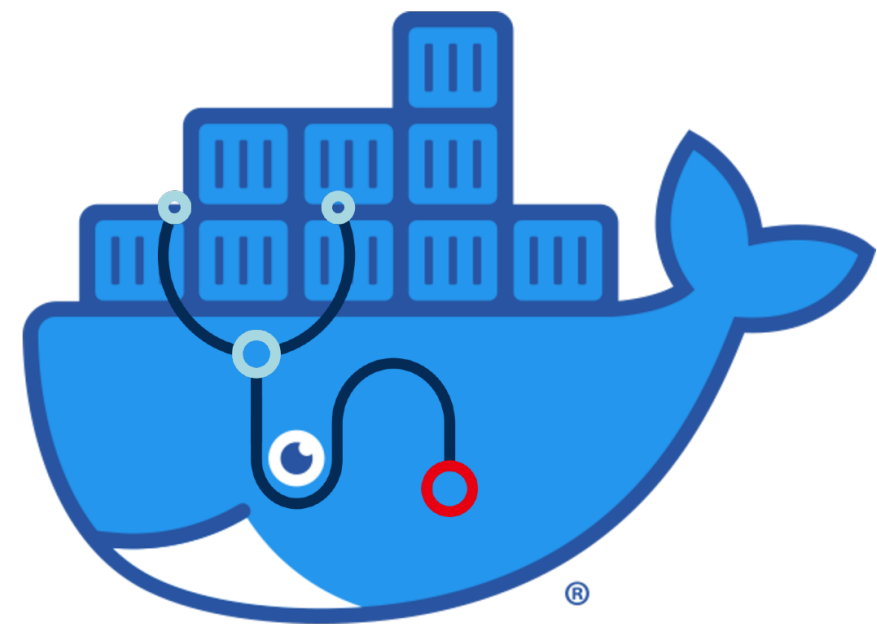
## Technical Details

Create your own plugins.

Scanning & Working: <https://asciinema.org/a/357544>

```
def _audit(container, results, audit_queue):  
    '''Make Sure to add dict of following structure to Audit Q  
    res = {  
        "container_id": [  
            "_plugin_name_", WARN/INFO/ERROR, details"  
        ]  
    }  
    ...  
    # Magical logic to perform Audit.  
    audit_queue.put(res)
```

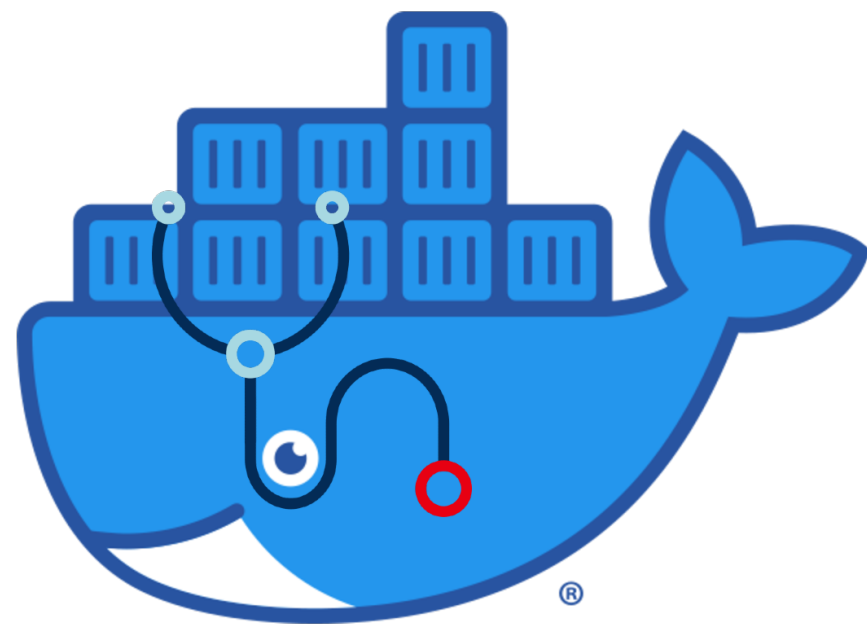
```
_plugin_name = 'demo_plugin'  
  
def scan(container, output_queue, audit=False, audit_queue=None):  
    _log.info('Starting {} Plugin ...'.format(_plugin_name))  
  
    res = {}  
  
    result = {  
        'test_class': {  
            'TEST_NAME': ['good']  
        }  
    }  
  
    res[container.short_id] = {  
        _plugin_name_: result  
    }  
  
    # Do something magical.  
  
    _log.info('Completed execution of {} Plugin.'.format(_plugin_name))  
  
    '''Make Sure you put dict of following structure in Q.  
    {  
        'container_id': {  
            'plugin_name': {  
                'test_name_demo1': {  
                    resultss:[]  
                },  
                'test_name_demo2': {  
                    results: []  
                }  
            }  
        }  
    }  
    ...  
    output_queue.put(res)  
  
    if audit:  
        _audit(container, res, audit_queue)
```



Analyse running container.

04

**Demo**



## Analyse running container.

04

**Demo**

3:"security-profiles, WARN, No AppArmorProfile Found"

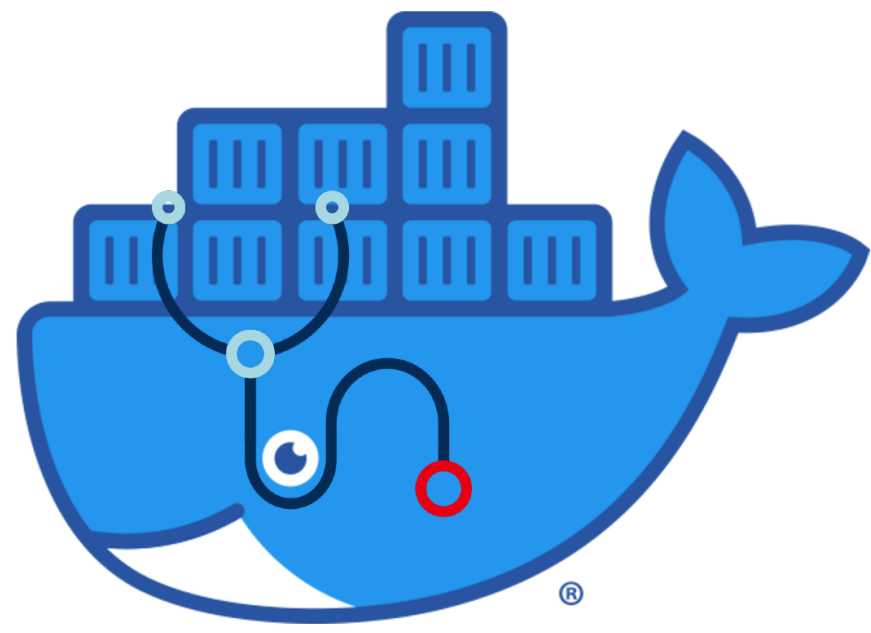
4:"security-profiles, WARN, SELinux disabled"

5:"user-info, WARN, Current USER is : root"

↓  
**Plugin  
Name**

↓  
**Alert Type**

↓  
**Description**

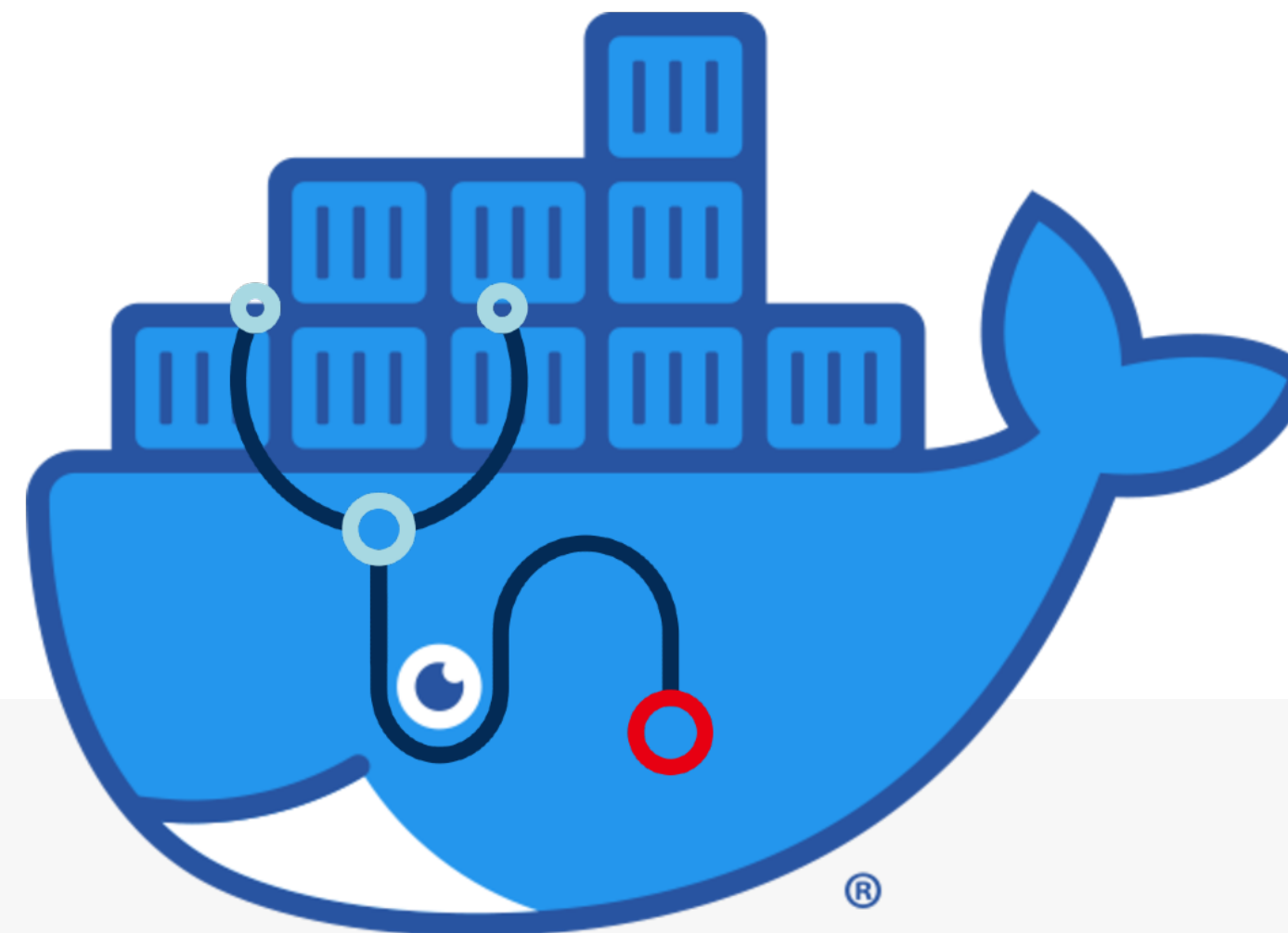


Analyse running container.

05

**Future**

- \* Suggestions for remediation.
- \* Better UI.
- \* Support from community.
- \* Simplified way of plugin creation.
- \* Docker Network Monitoring.
- \* Currently there is no Parallel processing when it come to UI.
- \* Popularise it !!



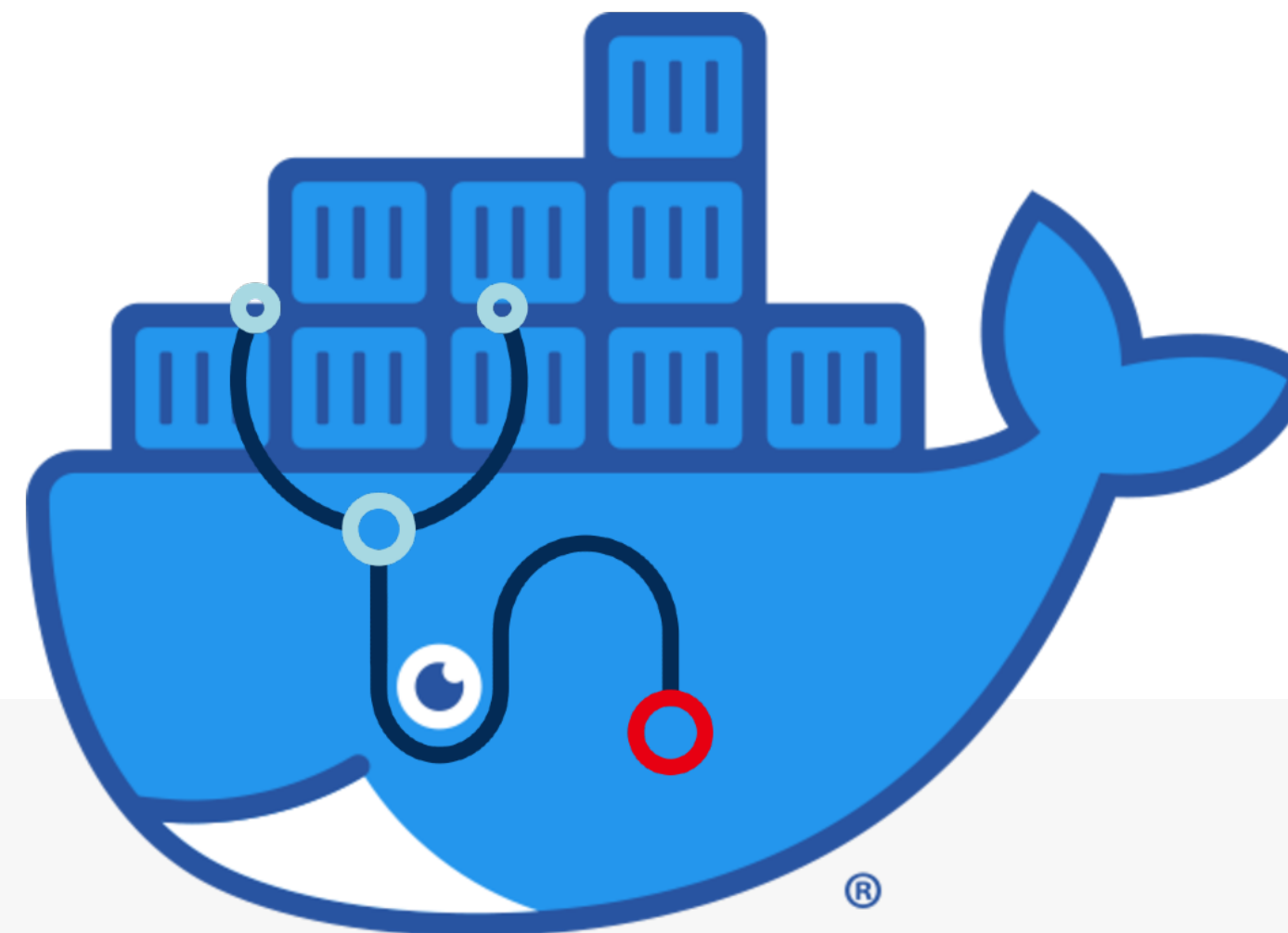
# DockerENT

**Analyse running container.**

GitHub Link : <https://github.com/r0hi7/DockerENT>



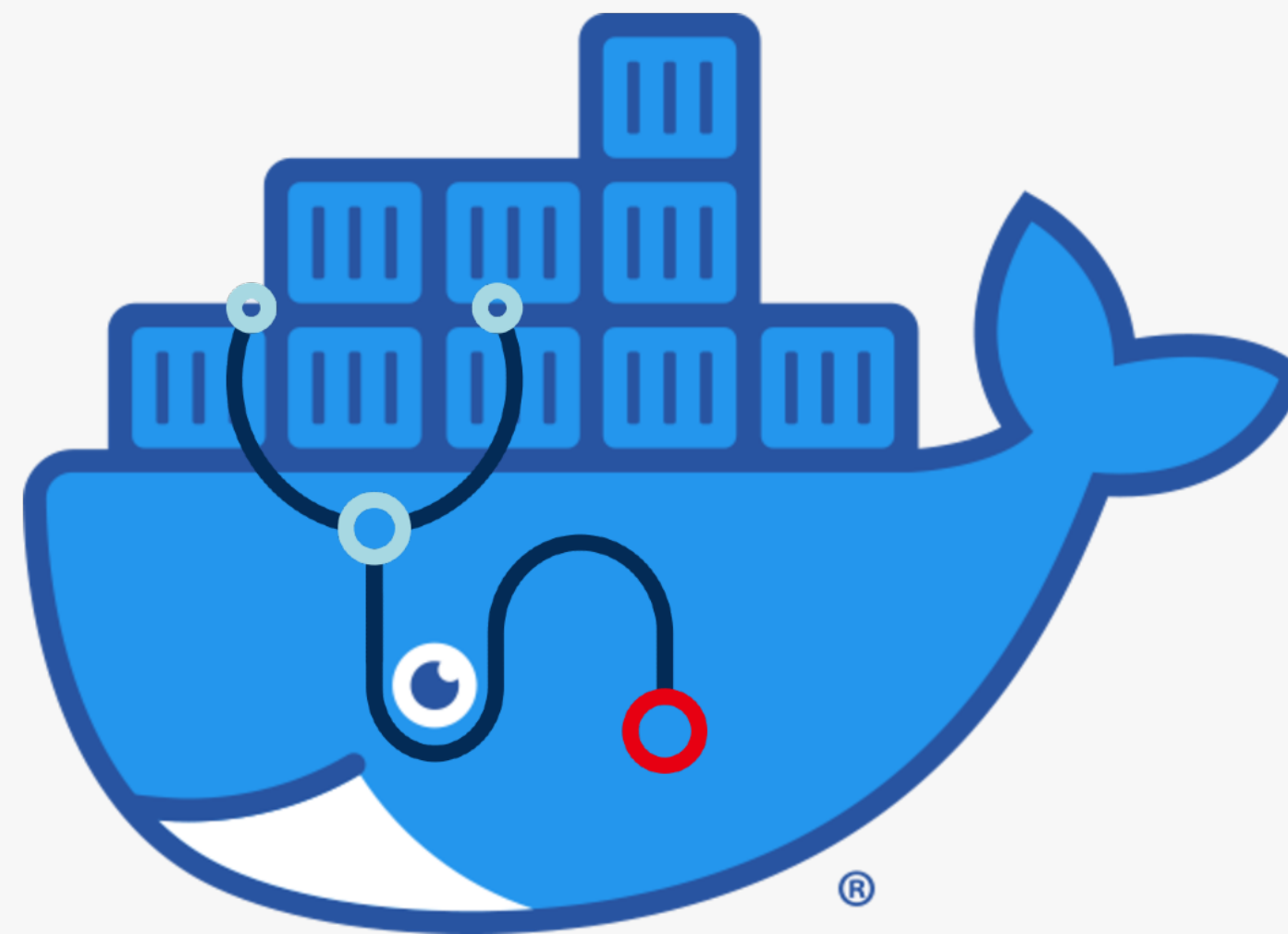




# DockerENT

## Q & A

# DockerENT



# Thank You !