null Ahmedabad

# Securing Containers from Day One

Container Security

• • •

Kumar **Ashwin**

0xcardinal.com

# About Me

Kumar **Ashwin**

- Security Consultant @ Payatu
- Manages null Study Groups for six different Security Domains and also contributes to other communities

- 0xcardinal.com
- 0xCardinal on social platforms

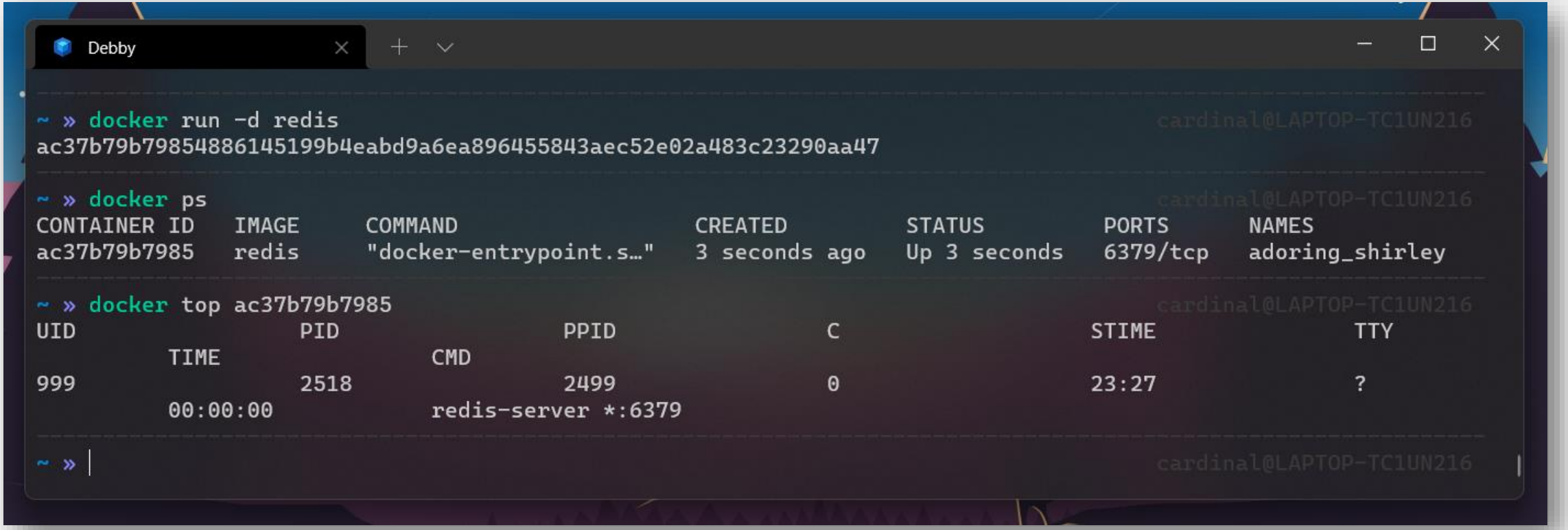
# Agenda

- What are containers?
- Why we need containers?
- VM v/s Containers
- Cgroups & Namespaces
- Docker Primer
- Build Optimization
- Security
- Resources
- QnA

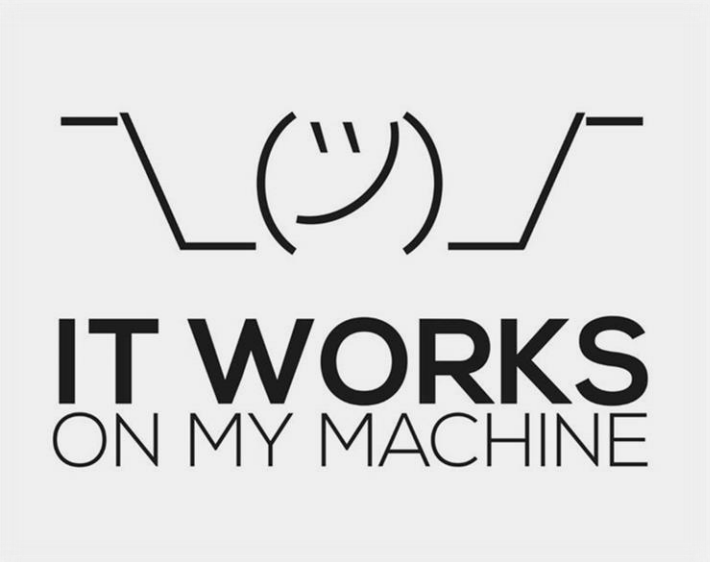

https://giphy.com/gifs/rockstargames-usz0fqhUiVxSs6IUKB

# What are **containers?**

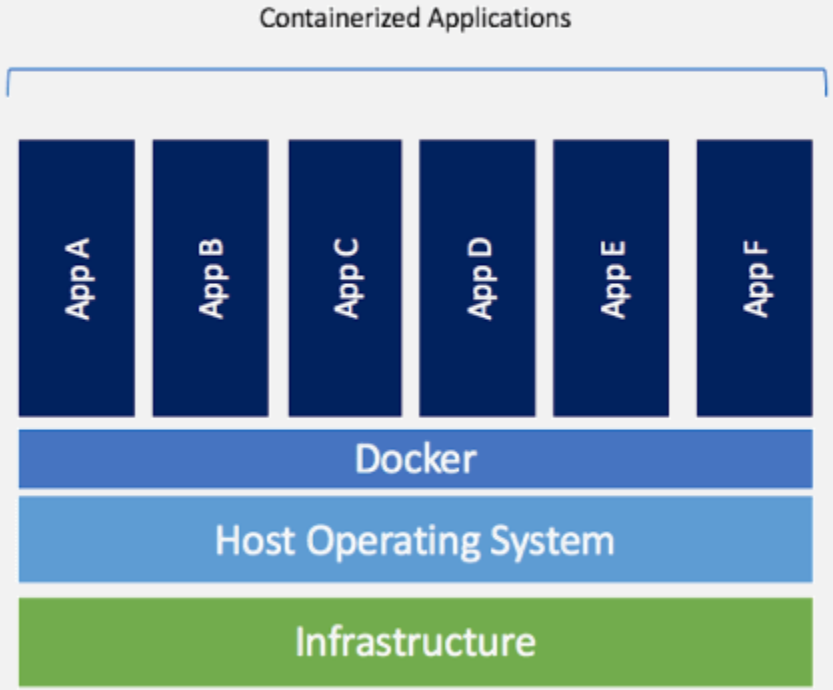
Containers are nothing but **just another Linux process** which is isolated from other processes running on the same host.

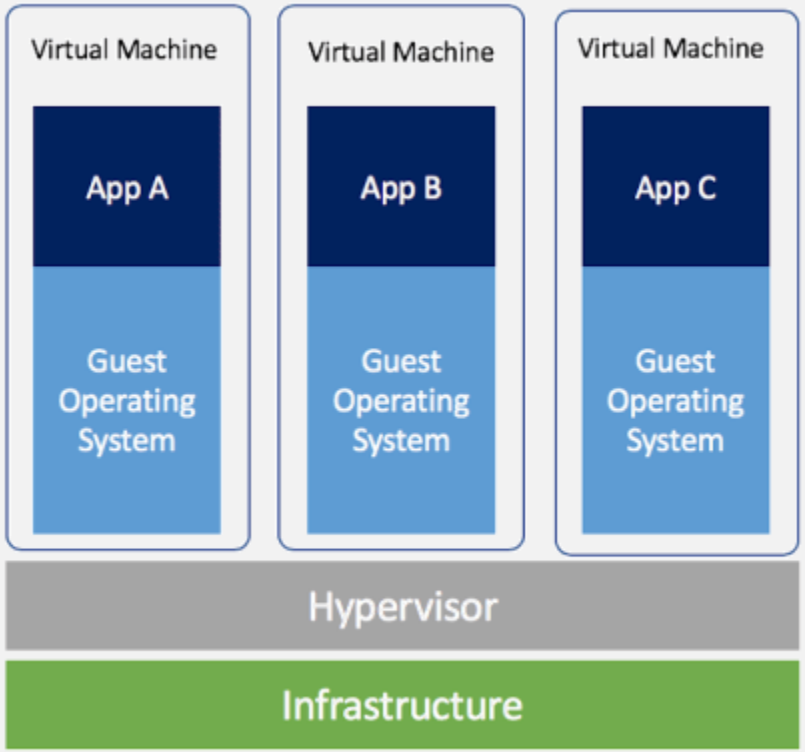


# Why do we need containers?



# Virtual Machines v/s Containers



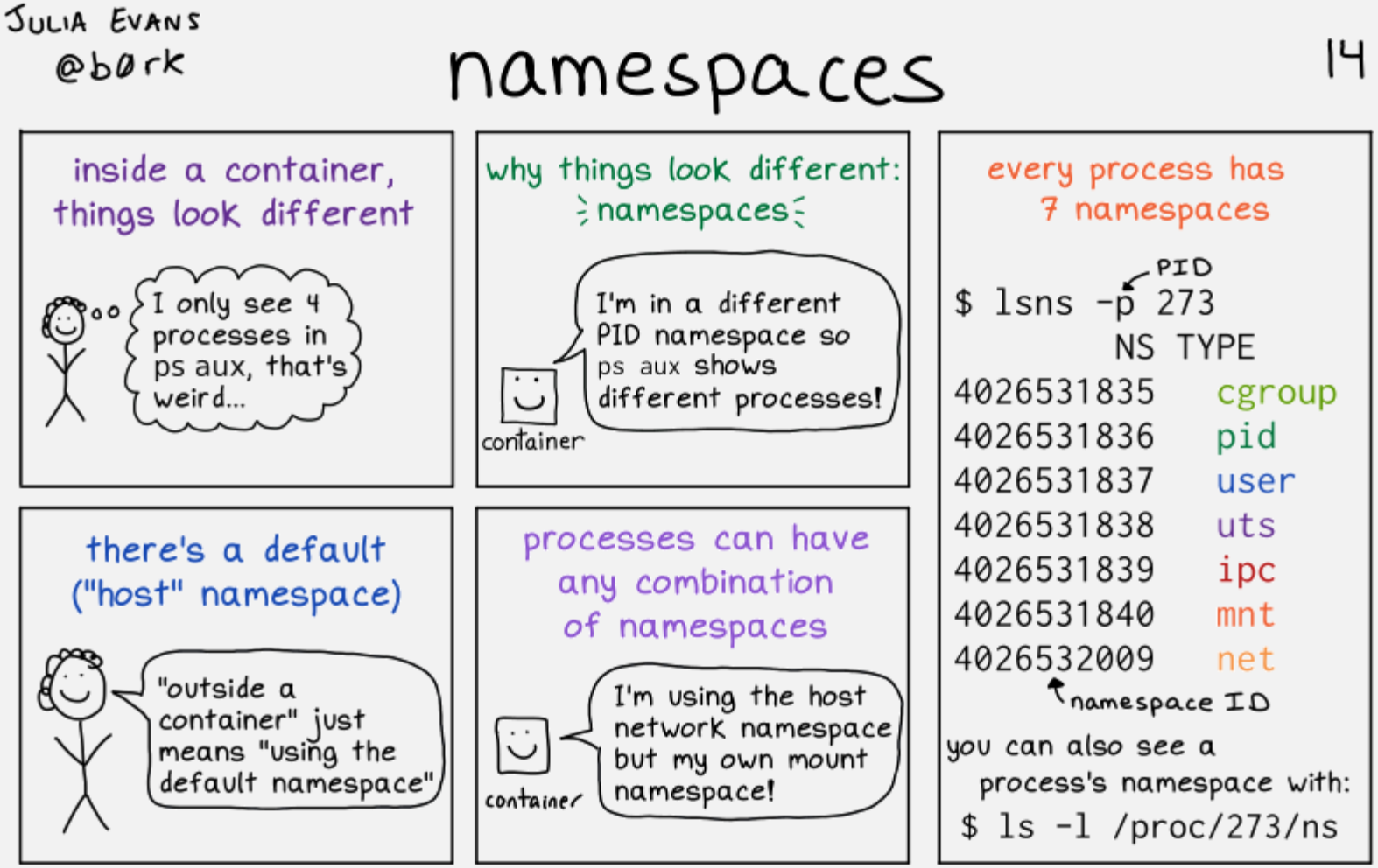




https://www.docker.com/blog/containers-replacing-virtual-machines/

# cgroups & namespaces

# namespaces

It defines what a container can **see**, uses syscalls to do so.



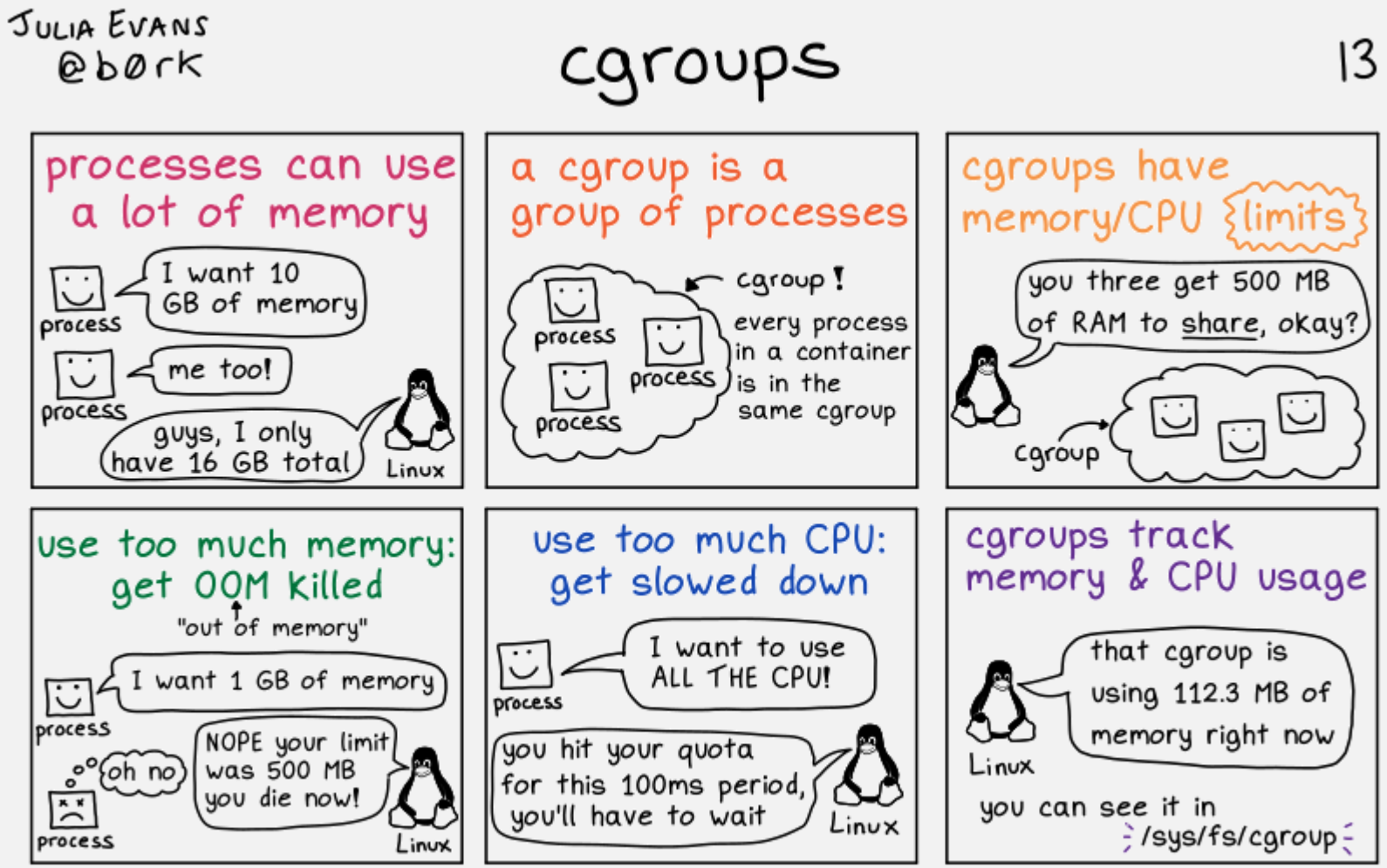

https://wizardzines.com/comics/namespaces/

# **Demo** : namespaces

Will share how the name spaces work using unshare and creating namespaces for user and network, and demonstrating the difference.

# cgroups

It defines what a container can **use or access**, uses syscalls to do so.




https://wizardzines.com/comics/cgroups/

# Docker
# Primer

Myth busting

# Docker itself is not a Container.

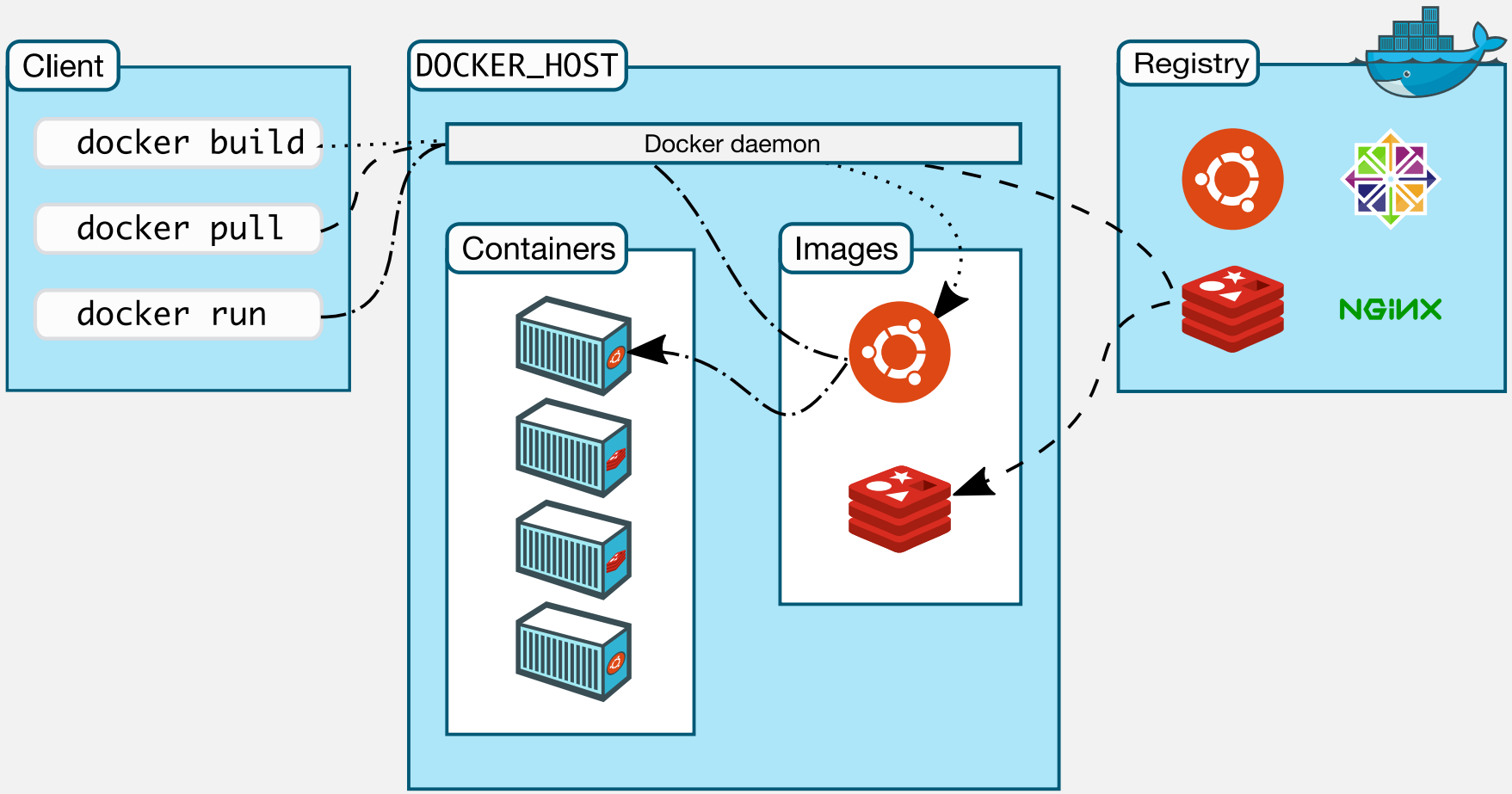
# What is **Docker?**

- Docker is a **container engine**, which

  *is a piece of software that accepts user requests, including command line options, pulls images, and from the end user's perspective runs the container.*

- Other than docker there are other container engines as well, like - **RKT, CRI-O, and LXD**

# Docker **Architecture**




https://docs.docker.com/get-started/overview/

# Docker **Basics**

- There are different **images**, which are stored in the **registry**, from where it **pull** the image, to create different containers.
- Generally, a `Dockerfile` (contains commands/instructions) is used to build a container.
- If you want to run a multi-container Docker application – **Docker Compose** is your go to tool.
- Common docker/docker compose commands, that are generally used -
  - `docker pull <image-name>` – used to pull images from registry
  - `docker run [args] <image-name>` – used to run a container from the image defined
  - `docker build [args]` – used to build a container out of a `Dockerfile`
  - `docker-compose up` – used to build a multi-container setup from `docker-compose.yml`
  - `docker ps` – used to list down the active containers
- For every `RUN`, `COPY`, `ADD` instruction in a `Dockerfile`, a layer is created.

# Dockerfile

Dockerfile

```
1   # Pulling the ngnix image from the dockerhub
2   FROM nginx:latest
3
4   # Copying the required files inside the container
5   ADD index.html /usr/share/nginx/html
6   ADD linux.png /usr/share/nginx/html
7
8   # Exposing the Port to interact with the server outside the container
9   EXPOSE 80 443
10
11  # Running some command
12  CMD ["nginx", "-g", "daemon off;"]
13
```

*** insecure**

# Build Optimization

- Minimize number of layers. Improves performance.
- Multi-Stage Builds
- Do not install unnecessary packages
- Decouple the application
- Slim down the image – using `docker-slim`
  - It promises to slim down the image by 30x - https://github.com/docker-slim/docker-slim

# Why Dockerfile **Security?**

- Here we will be talking about securing the images pre-build and what are the practices that we can follow.
- A great start point to look for security issues are Dockerfile.

- Dockerfiles?
  - These are the blueprints of the system/container that is to be created.
  - Infrastructure as a Code (IaaC)
  - One of the main components for the entire supply chain security.

# Security **Best-Practices**

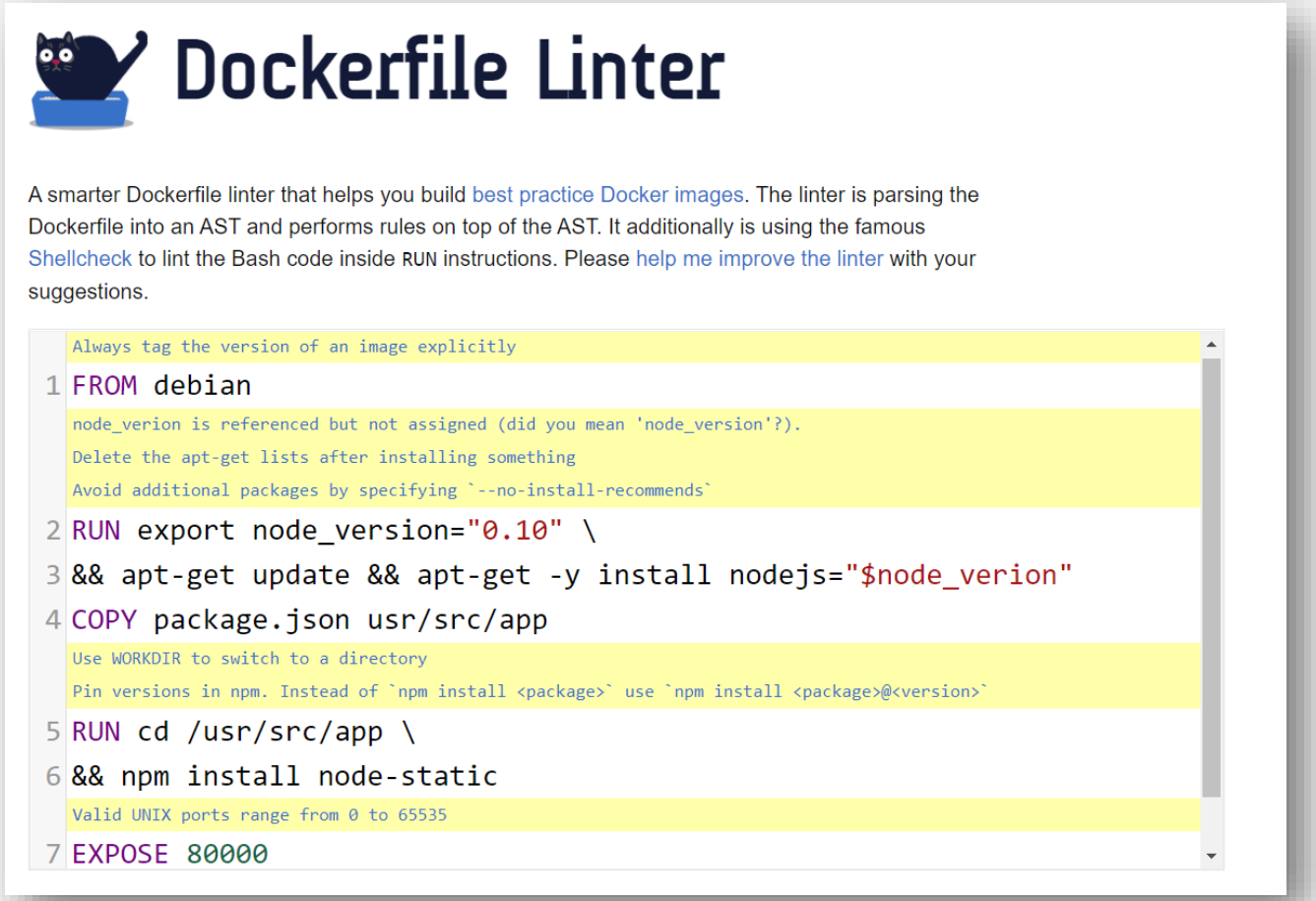
- Prefer minimal base images
- Use .dockerignore file to exclude files from build
- Create Golden Images
  - Golden Images are hardened base images than can be used further development
- Do not run containers as root
  - Adding this in the Dockerfile will help in chainging the user

```
FROM alpine:latest
RUN useradd –u 1234 non-root-user
USER non-root-user
```

- Do not commit secrets in Dockerfile or Containers
  - Can use BuildKit to pass secrets to use in containers securely
- Use COPY instead of ADD, wherever possible

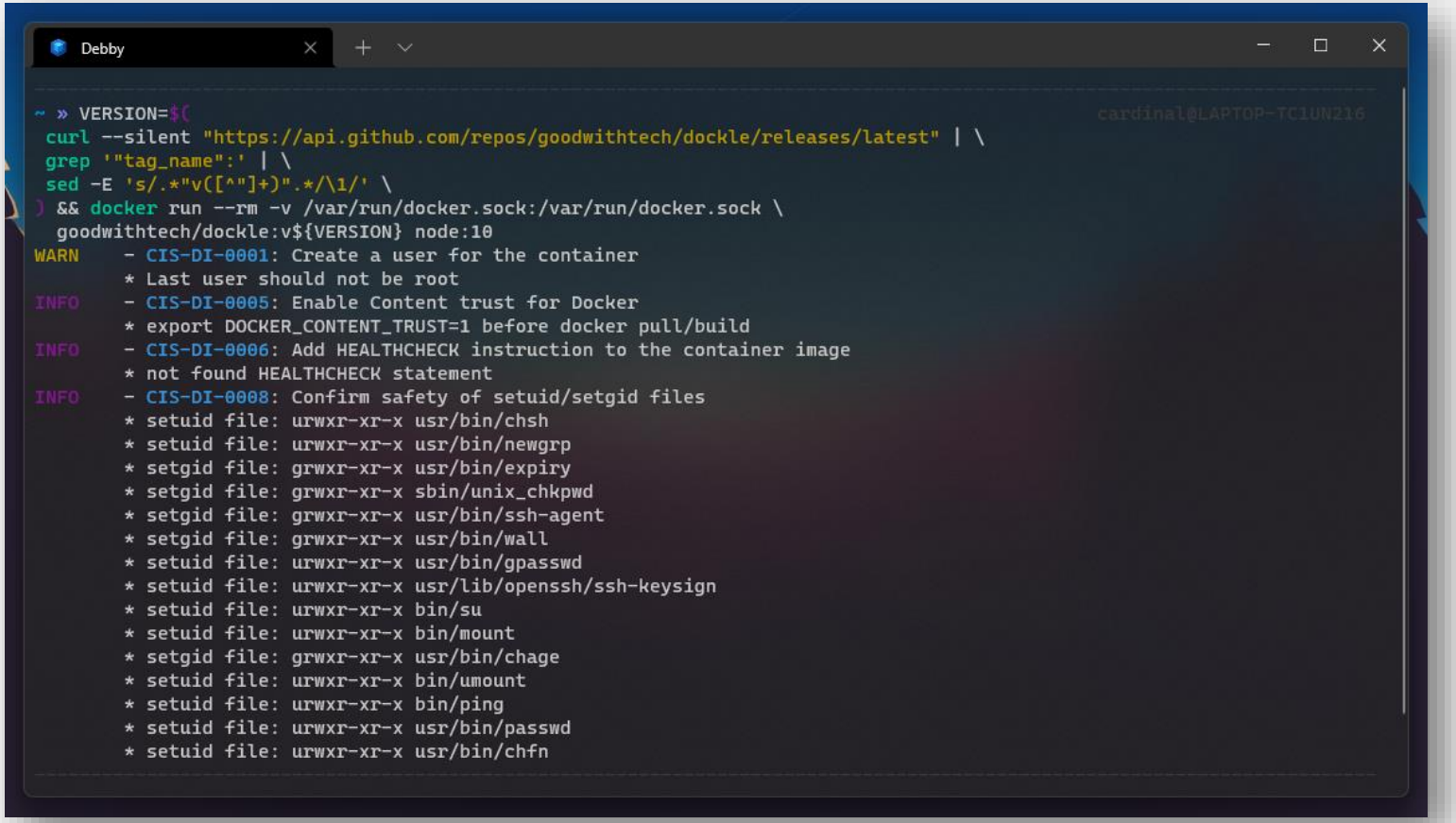
# More **Security** Stuff

- Use linters like [hadolint](), that will help to build best practice Docker Images.



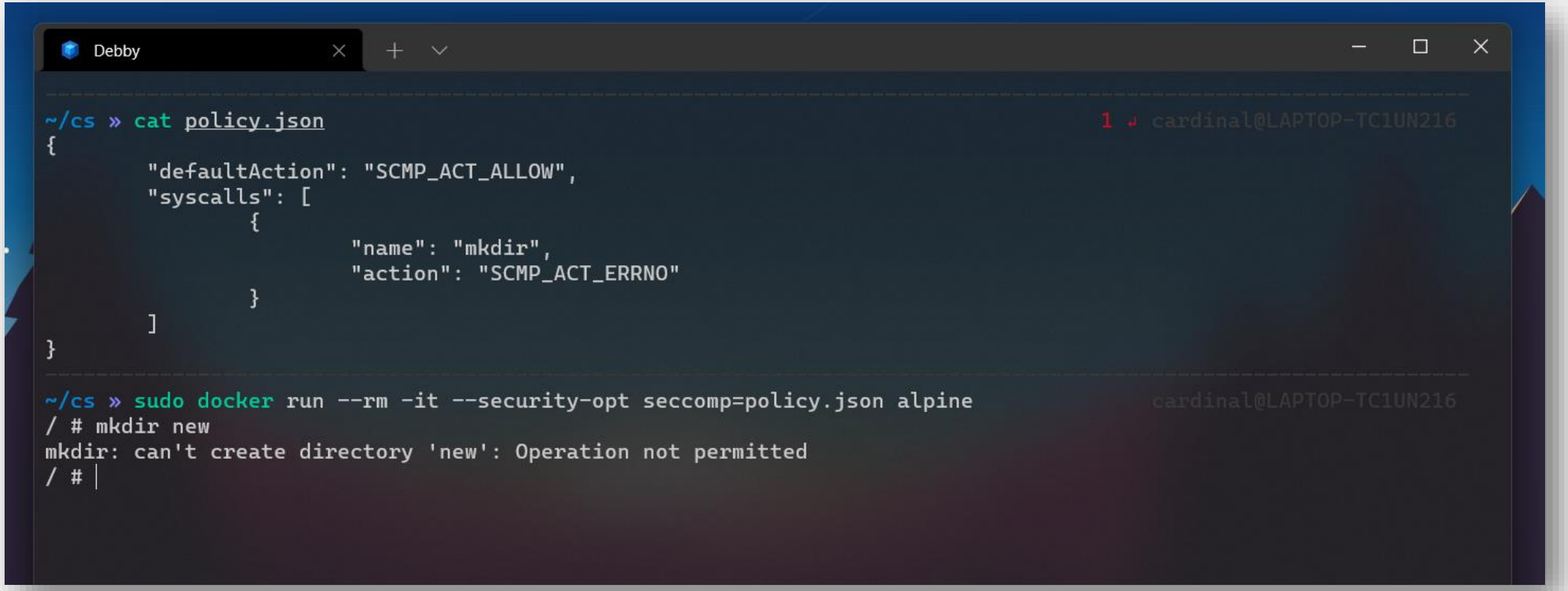
# More **Security** Stuff

- Use `dockle` to scan images against CIS benchmarks.
- [CIS Benchmark security Comparision]



# Seccomp

- It is a security feature in the Linux kernel. It can be used to restrict any system calls in the docker container.
- E.g.,



# Some **Resources**

- Containers from Scratch [YouTube]
- Dockerfile Best Practices
- Multi-stage builds
- dockle checkpoint comparison
- Tips for optimizing builds
- Dockerfile Tutorial
- Practical guide to write Dockerfile

# QnA

## Securing Containers from Day One

# Thank You!

Kumar **Ashwin**

twitter.com/0xcardinal