

CONTAINERS & DOCKER

HOMELAB CLUB AT UMD

10/07/2025

“WELL, IT WORKS ON
MY MACHINE”

THE PROBLEM

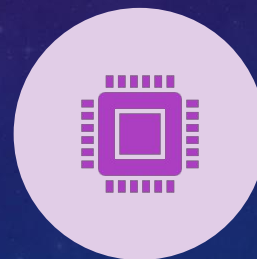
Development and production environments vary greatly



OPERATING
SYSTEM



ENVIRONMENT
VARIABLES



INSTALLED
APPLICATIONS

THE SOLUTION



CONTAINERS



AND VIRTUAL MACHINES

WHAT DOES A CONTAINER DO?

- Provides a separate environment for an application or group of applications that is lightweight and portable
- Isolated (contained haha)
- Scalable
- Consistent

HOW DO I START USING CONTAINERS?

Many providers of container services, each with their own pros and cons

- Docker
- Podman
- Kubernetes
- Portainer
- ...

Docker is the simplest and most widely used, so we will show how to use that

IMAGES

The building block for
containers

Contains:

- Files
- Binaries
- Libraries
- Configurations

Essentially a “snapshot”
of an environment

CREATING AN IMAGE IN DOCKER

Use a Dockerfile!

Create a file named
“Dockerfile” and
put in whatever
commands you
need

COMMON COMMANDS

- **FROM** `<image>` - specify the base image.
- **WORKDIR** `<path>` - sets the working directory in the image.
- **COPY** `<host-path>` `<image-path>` - copies files from the host to the image.
- **RUN** `<command>` - run the specified command in the default shell.
- **ENV** `<name>` `<value>` - set an environment variable in the image.
- **EXPOSE** `<port-number>` - expose a port on the image and set it to use a host's port.
- **USER** `<user-or-uid>` - set the user for all following instructions.
- **CMD** `["<command>", "<arg1>"]` - run the given command when the container is started.

BUILD AND RUN

To build an image from a Dockerfile,
simply type into a terminal

```
docker build
```

This will create an image in the
current directory.

To run it from there,
type in

```
docker run
```

COMPARISONS

CONTAINERS VS. VIRTUAL MACHINES (VM)

Containers:

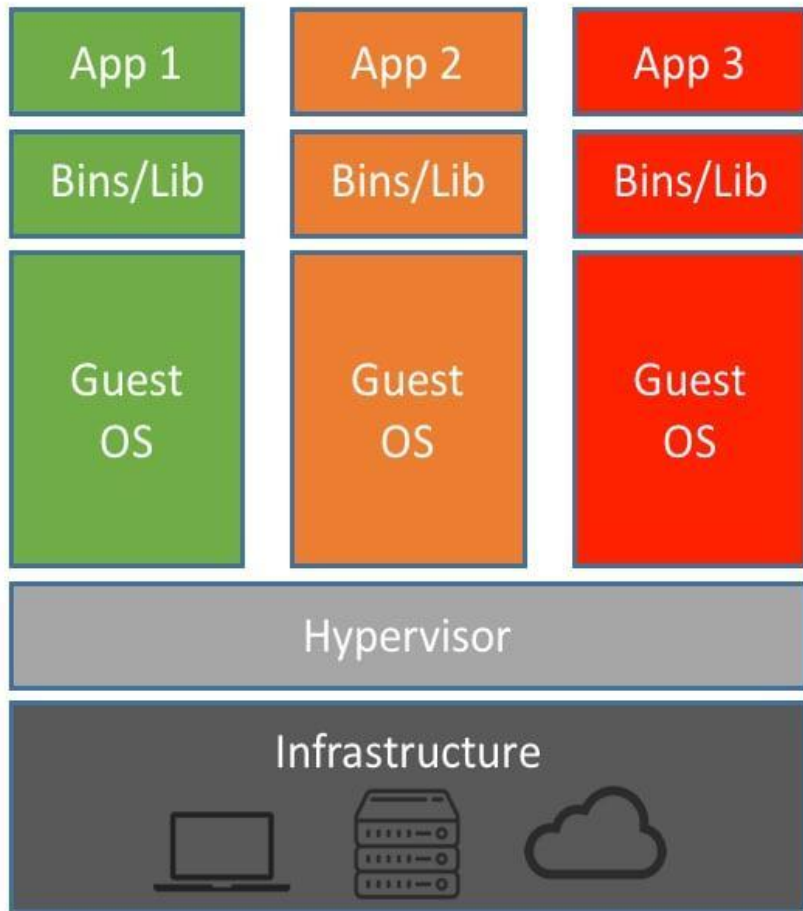
Share kernel space with the host machine

- Pros
 - Lightweight
 - More portable and scalable
- Cons
 - Less secure

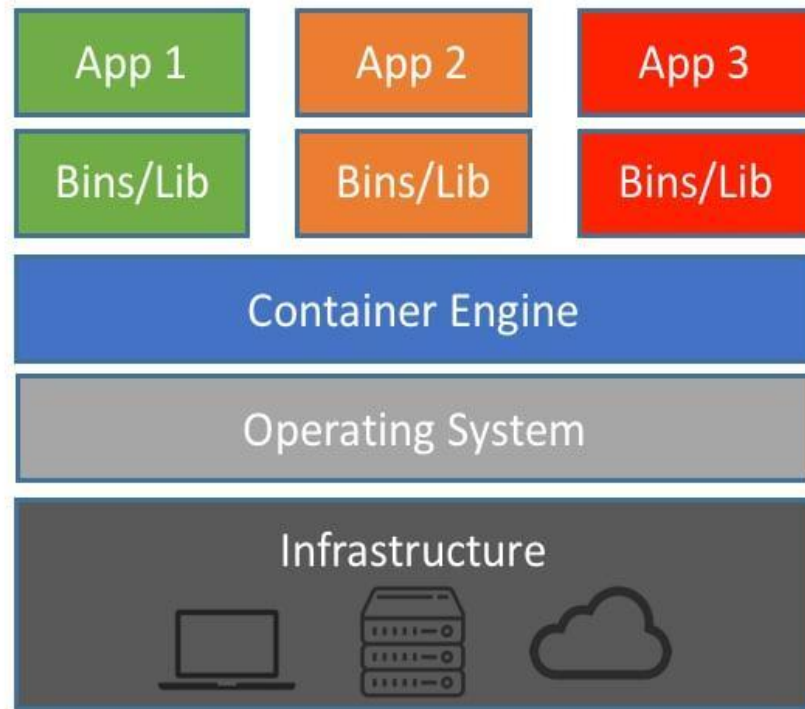
VMs:

Ship with their own Operating System and kernel space

- Pros
 - Better segmentation = more secure
- Cons
 - Require more resources than a container



Machine Virtualization



Containers

CONTAINER RUNTIMES

Docker

- Pros
 - All-in-one, can build and execute containers
 - Very widespread
- Cons
 - Typically requires root privileges

Podman

- Pros
 - Daemon-less
 - Root-less
 - Compatible with docker images
- Cons
 - Doesn't have its own image creation tool
 - Requires more setup

CONTAINER DEEP DIVE

DEPENDENCIES

Containers rely* on three different linux technologies in order to function

- Namespaces
- Chroot
- Cgroups

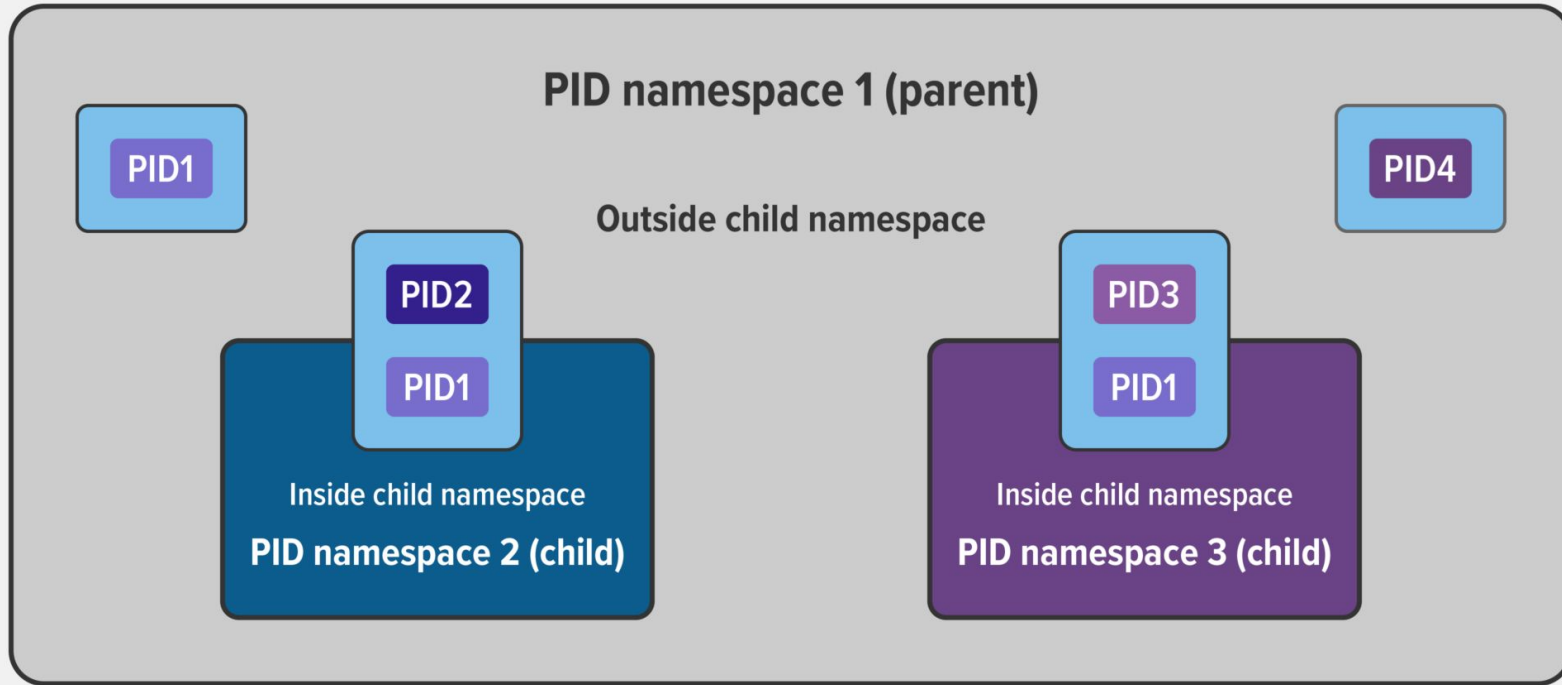
NAMESPACES

Namespaces are a tool that is useful in isolating processes on a linux machine.

They partition software resources that limit what resources certain processes can see.

There are different kinds of namespaces in linux

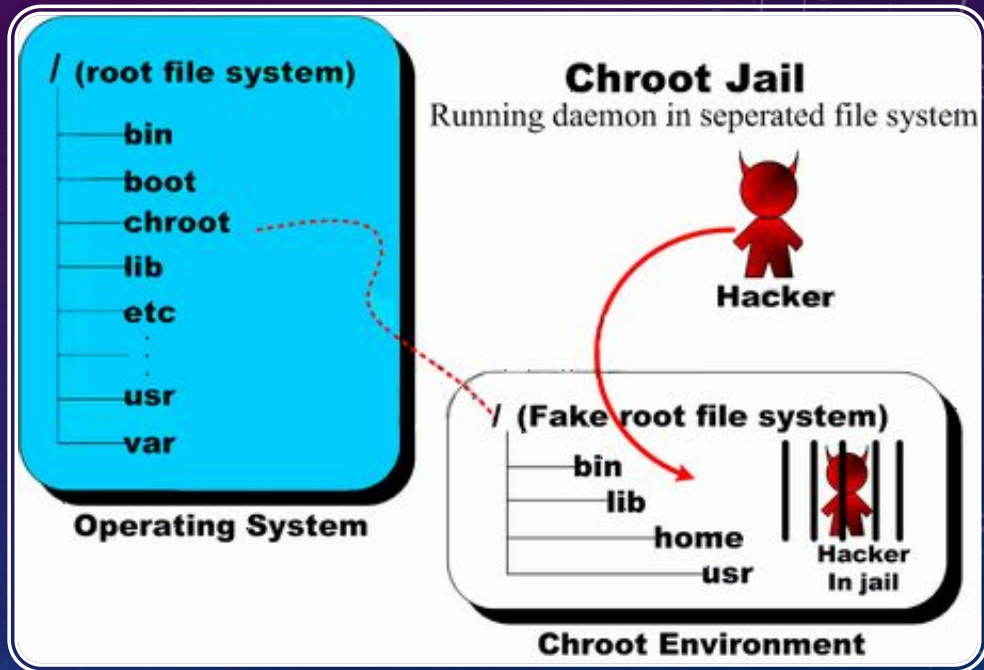
- User namespaces
- Process ID(PID) namespaces
- Network namespaces
- Mount namespaces
- ...



Example of a PID namespace

CHROOT

This is a Linux command that will change the root filesystem for a process



CGROUPS

Cgroups are useful for managing **hardware** resources in sets of processes

Resource limits

set a hard limit for the amount of resources a cgroup can use

Prioritization

You can give one cgroup a higher proportion of resources compared to others when they contend for resources

Accounting

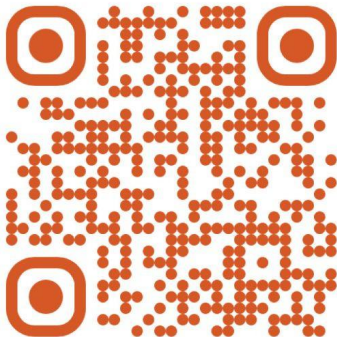
resource usage is monitored at the cgroup level

Control

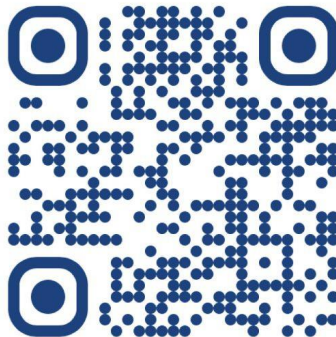
processes in a cgroup can be frozen, stopped, and restarted all at once

QUESTIONS, COMMENTS, CONCERNS?

Website



Discord



Terplink

