Container Basics

what why secure?

> $ _

n3|{π0s!s

- day time cybersecurity engineer
- night time student
  - perpetual newb

- tweet @n3krosis11
- email n3krosis@protonmail.com

> $ _

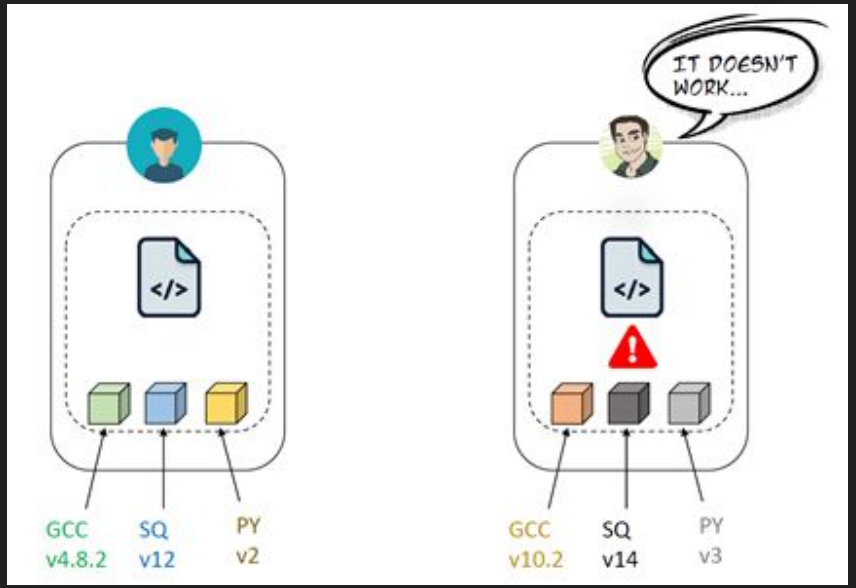
today's lecture

- content adapted from class lecture and blackhat talk
  - Shelby Thomas Ph.D. @realshelbyt on Twitter
  - Brandon Edwards & Nick Freeman
- what are container
- container platforms
- docker demo
- security
- escape demo
- questions

> $ _

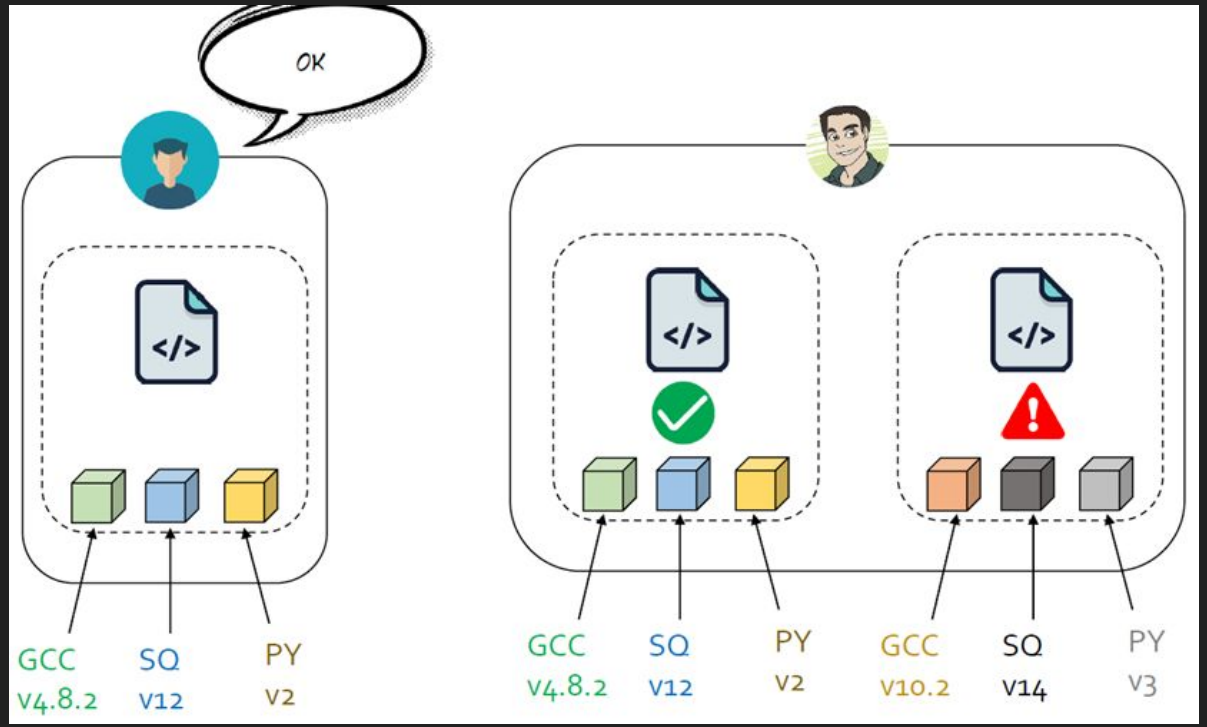
what can containers do?

- send code to another
- different development environments
- prevents everything breaking
- can’t easily send computers



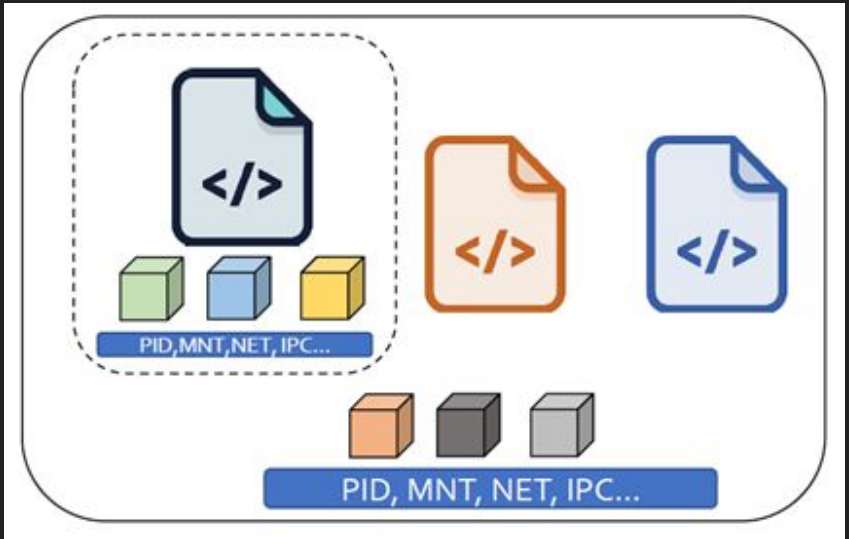
> $ _

multiple versions & dependencies



> $ _

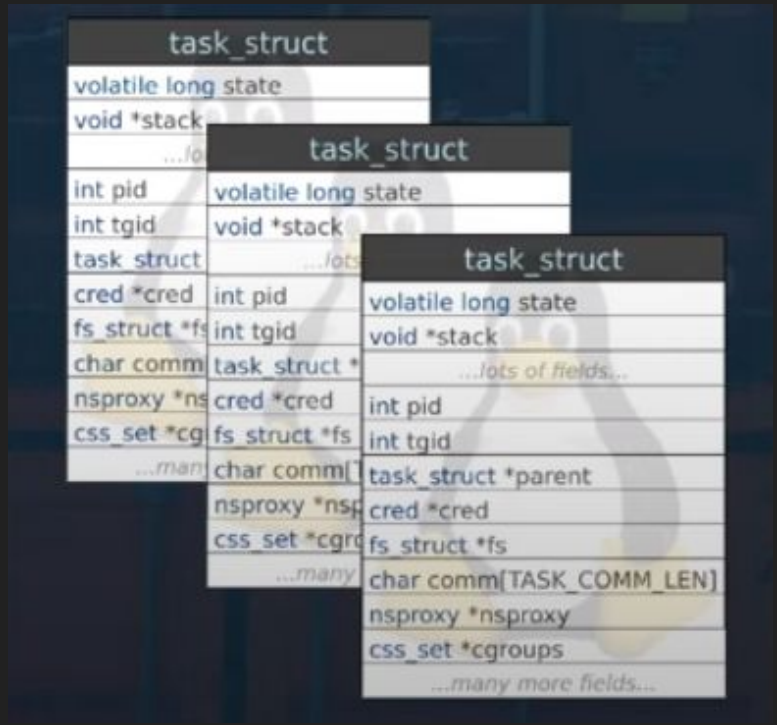
what is a container?



- containers are a type of virtualization for linux kernel
- what kernel capabilities might a process (program) need?
- the capabilities are called namespaces
- new containers mean new namespaces

> $ _

little deeper, what is a container?

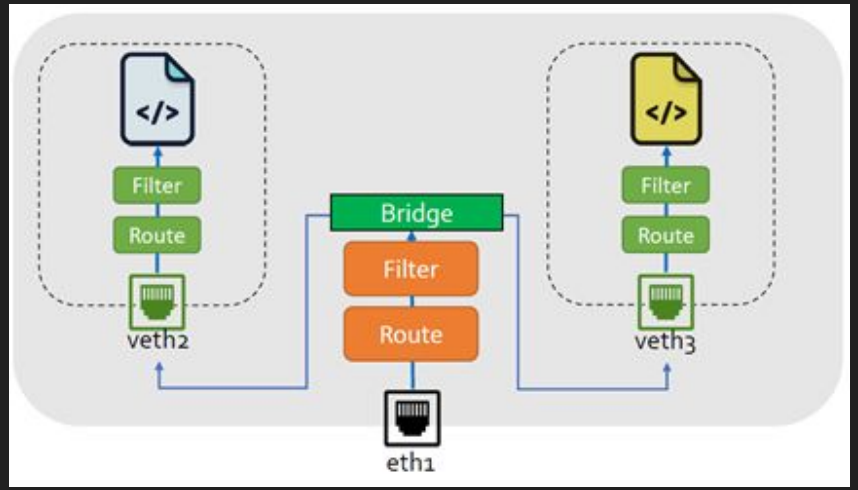


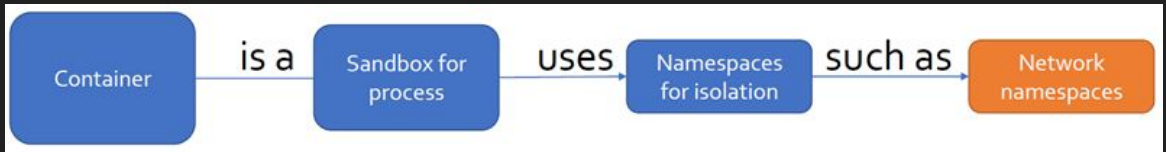
- task struct with key features
  - credentials
    - user permissions
  - capabilities
    - separate root things
  - filesystem
    - container's own fs
  - namespaces
    - define things it can do
  - cgroups
    - resource limitations
  - linux security modules
    - apparmor, selinux
  - seccomp
    - restrict syscalls

> $ _

closer look at the network namespace

- what is a network?
- network namespace isolation
- operational benefit: experiment with rules because they are tied to a container
- often managed through a virtual bridge/switch





> $ _

container platforms

```
$ ip link add veth0 type veth peer name veth1
$ ip link set veth1 netns pa1
$ ip netns exec pa1 ip addr add 10.1.1.1/24 dev veth1
$ ip netns exec pa1 ip link set dev veth1 up
```

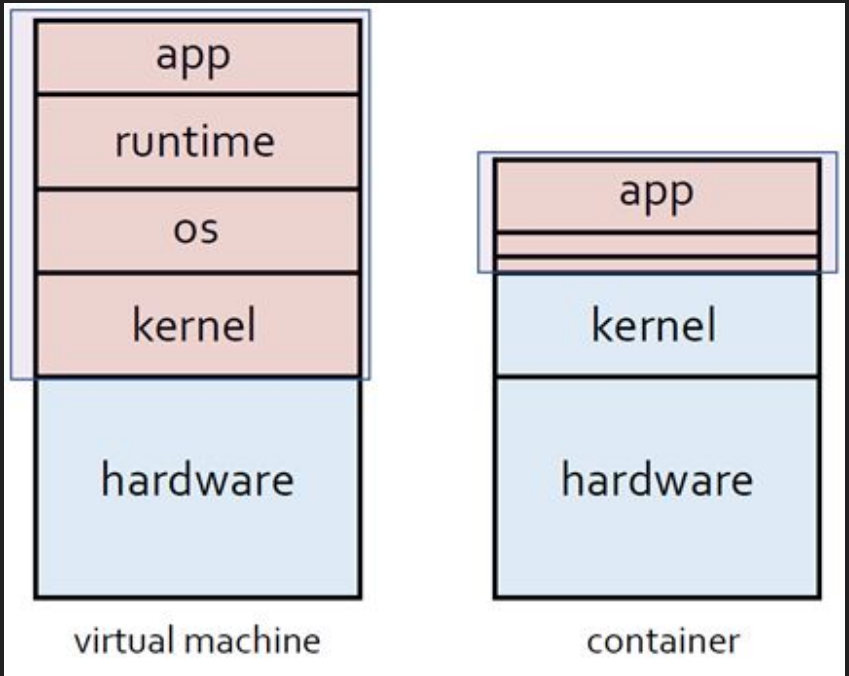
- these are all kernel capabilities (i.e., can build a container without additional software)
- struggles of creating namespaces
- repeat for additional isolation

> $ _

- container software makes it easy to manage containers and namespaces
- docker abstracts underlying namespaces
- isolated shell, no manual configuration of namespaces needed

> $ _

virtual machine vs. containers

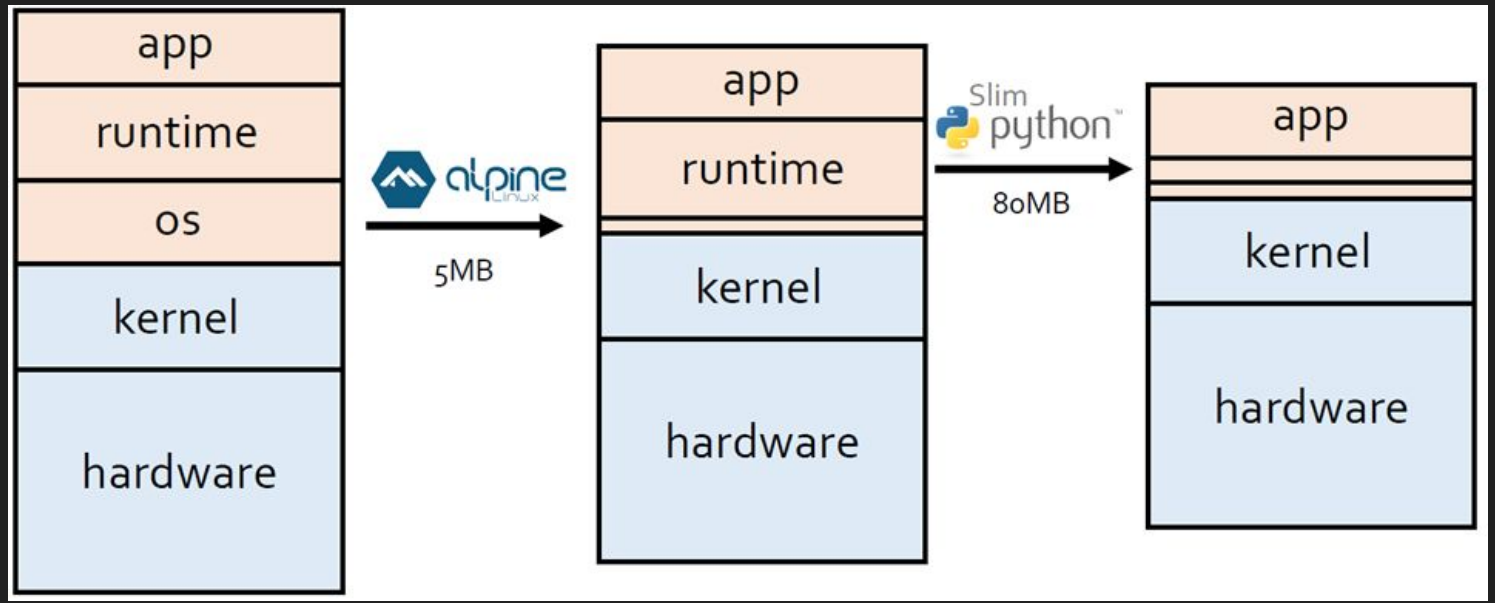


- different isolation points
- container benefits
  - lightweight (megabytes vs gigabytes [10x])
  - fast startup (milliseconds vs. minutes
  - simple to build, deploy, send, & maintain

> $ _

virtual machine vs. containers

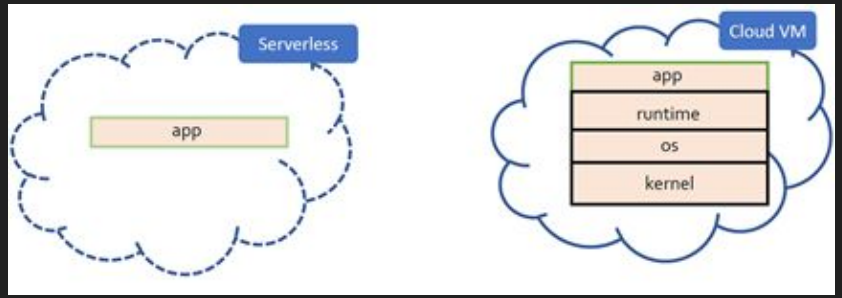
- benefits come from specialization



> $ _

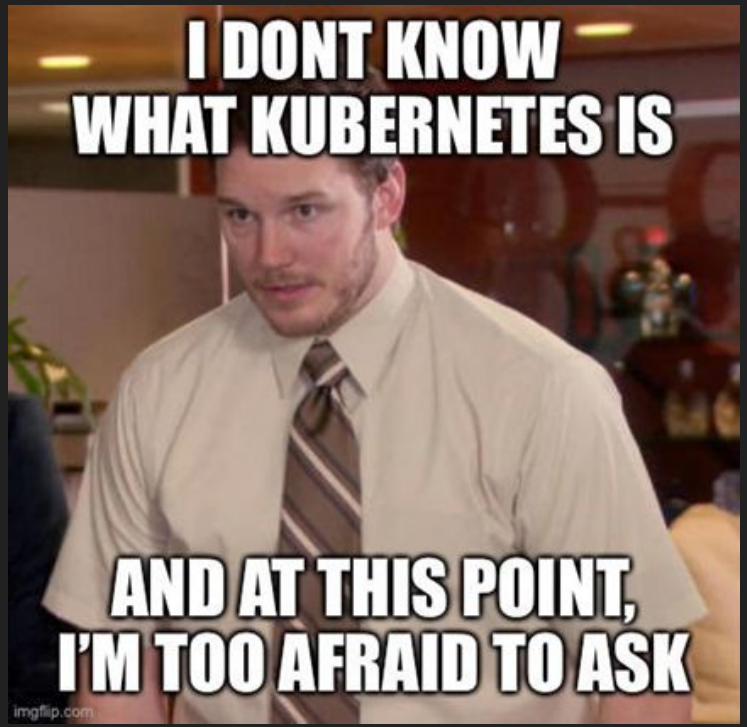
serverless computing

- os and runtime are partially managed by cloud providers
- serverless
  - on demand
  - focus on application
  - subject to fluctuations
  - questionable security
- traditional cloud
  - persistent
  - more control
  - fast and predictable
  - difficult to manage



> $ _

kubernetes aside

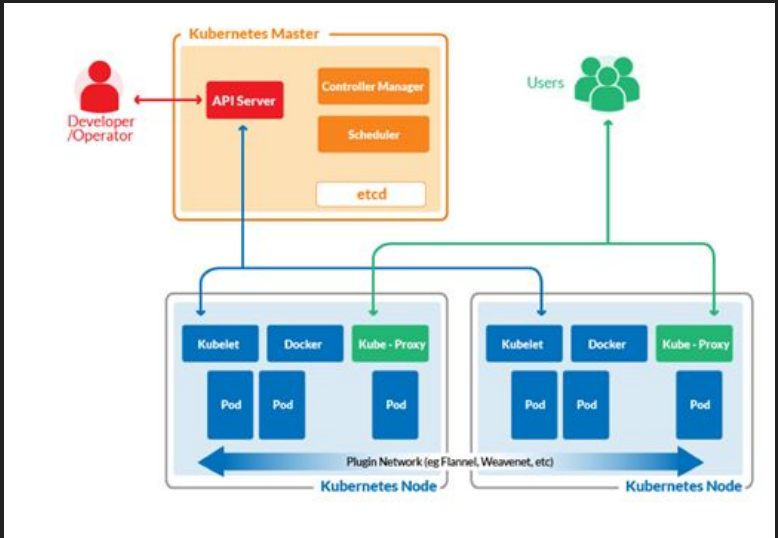


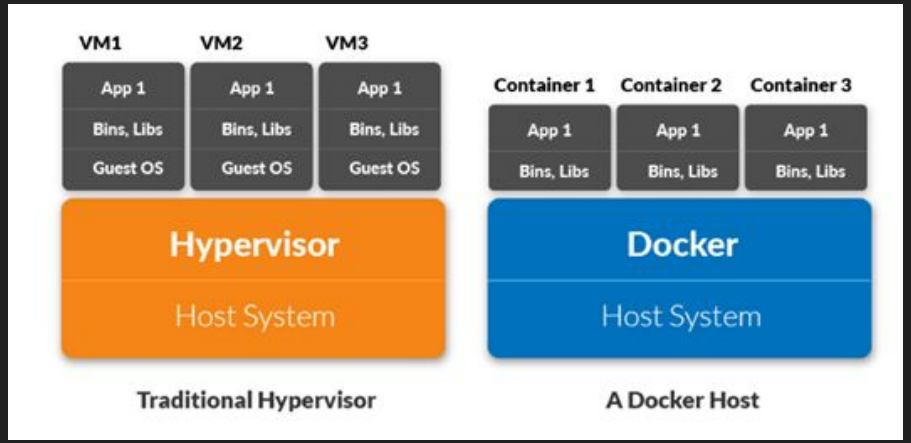
- Automates operational tasks of container management
  - Automated operations
  - Infrastructure abstraction
  - Service health monitoring
- vs. docker
  - kubernetes == architecture
  - docker == host application

> $ _

kubernetes vs. docker

kubernetes





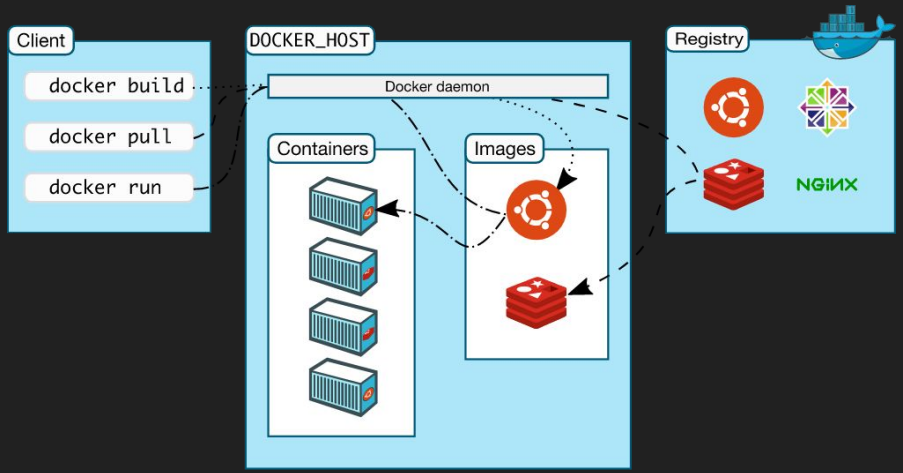
docker

> $ _

setup demo

> $ _

docker basic operation



- format
  - docker cmd [IMAGE_NAME]
  - docker cmd [CONTAINER_NAME]
- popular
  - create / run [image]
  - start / stop [container]
  - restart [container]
  - pause / unpause [container]
  - ps / ps -a
  - rm

> $ _

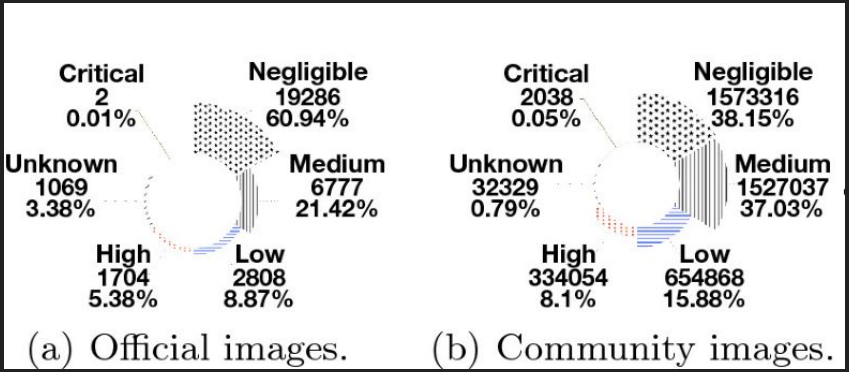
dockerfile + compose

- dockerfile
  - text document that contains all the commands a user could call on the command line to assemble an image
  - ***docker build*** allows for automated build that executes several command-line instructions in succession
- compose
  - tool for defining and running multi-container docker applications
  - e.g. development environments, automated testing environments, single host deployments

```
# syntax=docker/dockerfile:1
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
EXPOSE 5000
COPY . .
CMD ["flask", "run"]
```

```
version: "3.9"
services:
  web:
    build: .
    ports:
      - "5000:5000"
  redis:
    image: "redis:alpine"
```

> $ _

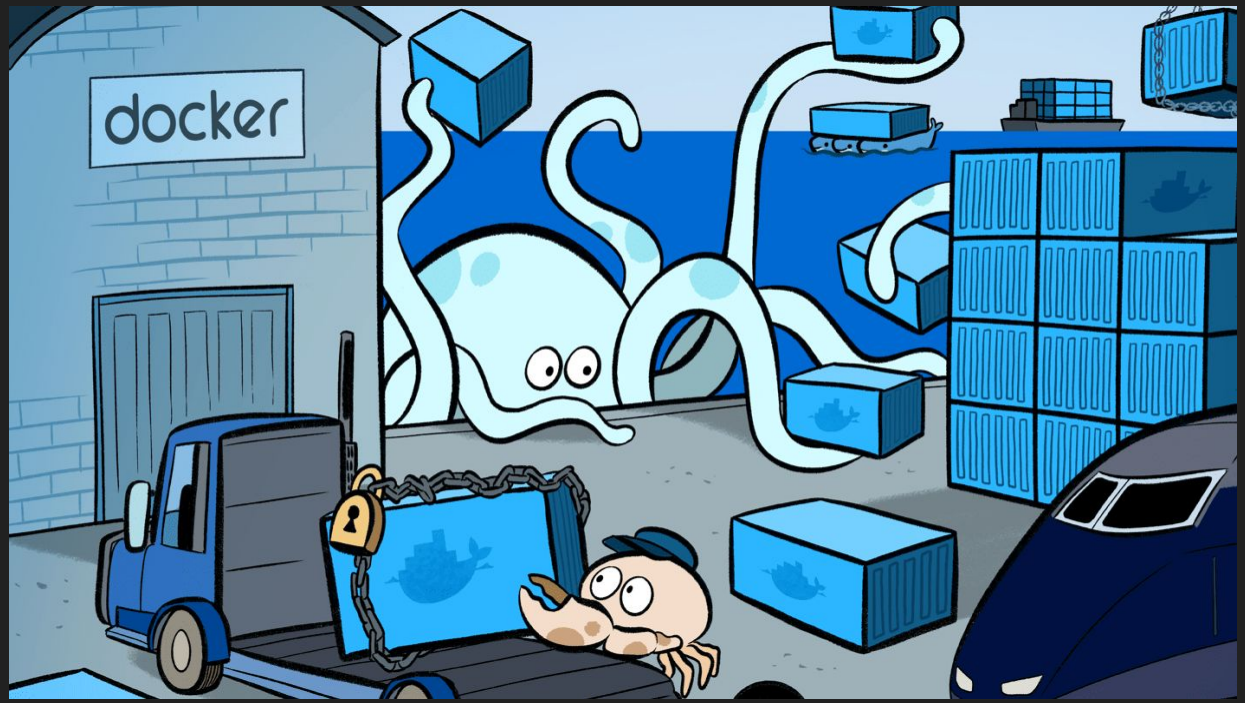


from Understanding the Security Risks of Docker Hub 09-2020

- benefits
  - image repos
  - teams and orgs
  - github bitbucket integration
  - automated builds
  - webhooks
  - official and publisher images
- security issues
  - trust everything you download?
  - images contain patched vulns
  - malicious images
    - ffmpeg -> btc mining
  - Dec 2020 - reported 51% of the Docker Hub images had exploitable vulnerabilities
    - company site offline

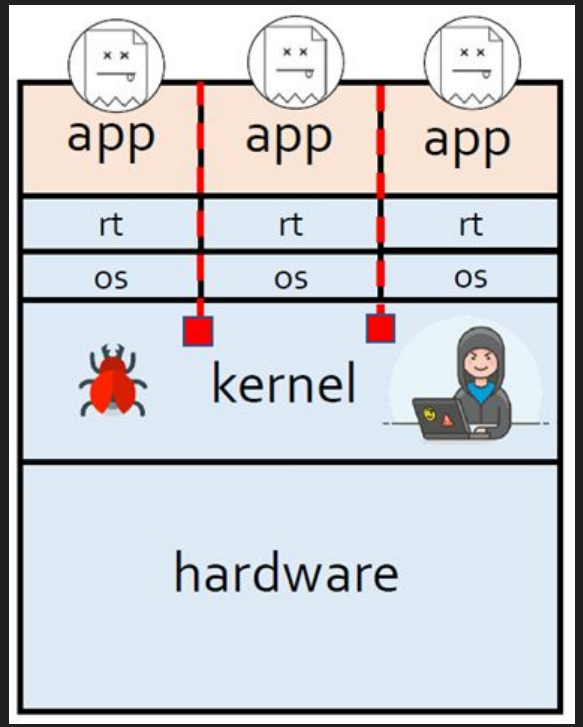
> $ _

but is it safe



> $ _

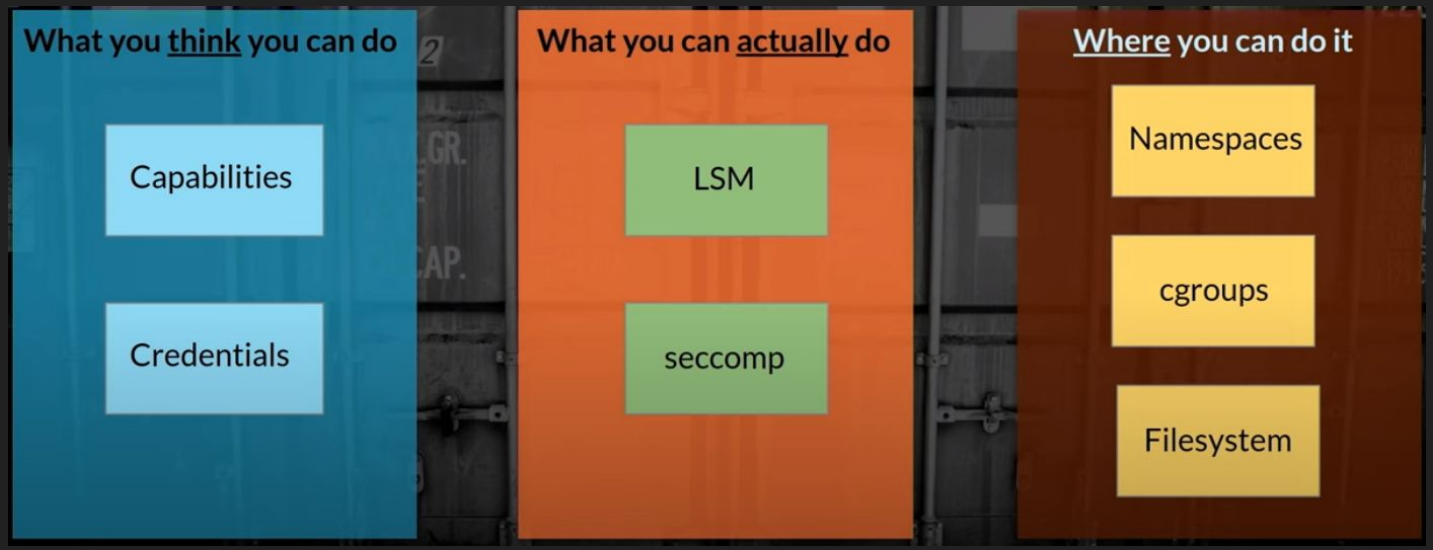
security layers



- container image itself and the software inside
- interaction between container, host os, and other containers on the same host
- host os
- networking and storage
- security at runtime

> $ _

container security model

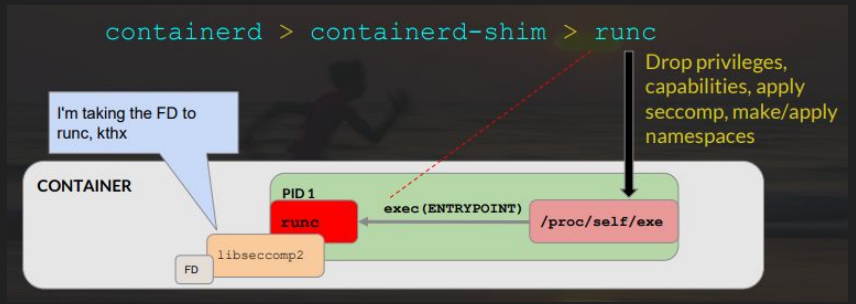


recent cves

- CVE-2019-5736
  - CVSS 3.x -- 8.6 HIGH
  - CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H
  - runc 1.0-rc6, Docker before 18.09.2
  - attackers overwrite host runc binary and consequently obtain host root access
- 9 CVEs since <= 7.5 Medium

regular

escape



> $ _

brandon edwards & nick freeman

> $ _

common weaknesses

- exposed docker sockets
  - docker cmds, use curl too
  - inception (docker in docker)
- privileged containers
  - init with --privileged flag
- excessive capabilities
  - privileged or root
- sensitive mounts
  - access to hosts fs
- kernel exploits
  - dirty cow
  - escape to update struct
    - credentials
    - namespaces
    - fs struct

> $ _

escape patterns

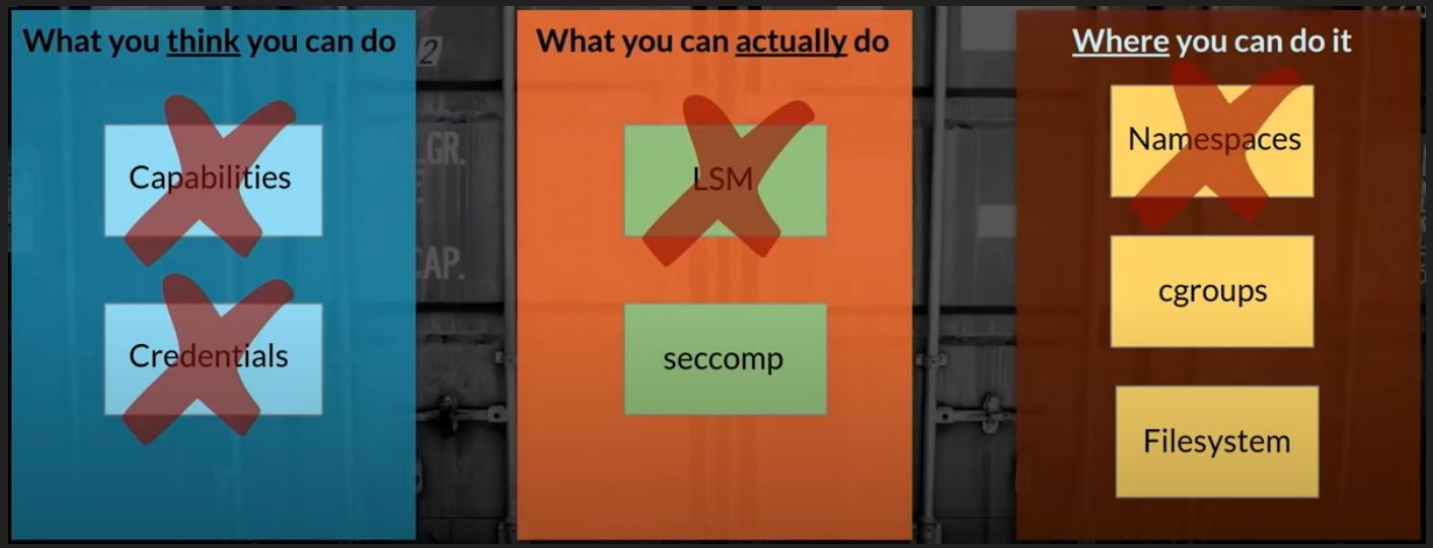
- step 1:
  - memory layout, state grooming, etc.
- step 2:
  - trigger bug for kernel access
- step 3:
  - rop to disable smep/smap
- step 4:
  - return to userland
- step 5:
  - commit_creds(\
    prepare_kernel_creds(0));

> $ _

escape demo

> $ _

revised container security model



> $ _

securing containers -- owasp

- RULE #0 - Keep Host and Docker up to date
- RULE #1 - Do not expose the Docker daemon socket (even to the containers)
- RULE #2 - Set a user
- RULE #3 - Limit capabilities
- RULE #4 - Add -no-new-privileges flag
- RULE #5 - Disable inter-container communication
- RULE #6 - Use Linux Security Module
- RULE #7 - Limit resources
- RULE #8 - Set filesystem and volumes to read-only
- RULE #9 - Use static analysis tools
- RULE #10 - Set the logging level to at least INFO
- Rule #11 - Lint the Dockerfile at build time

> $ _

automate security

- Docker Bench for Security
  - script that checks for dozens of common best-practices around deploying Docker containers in production
  - tests are all automated
  - inspired by the CIS Docker Benchmark v1.2.0
- Ansible from DoD
  - define and enforce system and application configurations
  - configurations that implement most of the Docker Enterprise 2.x Linux/Unix STIG

> $ _

docker bench demo

> $ _

questions


I NEED TO KNOW WHY MOVING OUR APP TO THE CLOUD DIDN'T AUTOMATICALLY SOLVE ALL OUR PROBLEMS.
Dilbert.com @ScottAdamsSays
YOU WOULDN'T LET ME RE-ARCHITECT THE APP TO BE CLOUD-NATIVE.
JUST PUT IT IN CONTAINERS.
© 2017 Scott Adams, Inc./Dist. by Andrews McMeel
YOU CAN'T SOLVE A PROBLEM JUST BY SAYING TECHY THINGS.
KUBERNETES.

> $ _

further reading

- Kubernetes vs. Docker: A Primer
  - https://containerjournal.com/topics/container-ecosystems/kubernetes-vs-docker-a-primer/
- Installing Docker on Debian 10
  - https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-debian-10
- Container Security For Development Teams
  - https://snyk.io/learn/container-security
- A Compendium of Container Escapes (Black Hat 2019)
  - https://youtu.be/BQlqita2D2s
- Docker Security Cheat Sheet
  - https://cheatsheetseries.owasp.org/cheatsheets/Docker_Security_Cheat_Sheet.html

> $ _

further reading

- Understanding Docker container escapes
  - https://blog.trailofbits.com/2019/07/19/understanding-docker-container-escapes/
- Vulnerable Docker containers for testing
  - https://hub.docker.com/u/vulnerables
- Example of Docker escape
  - https://github.com/Swordfish-Security/Pentest-In-Docker
- Understanding the Security Risks of Docker Hub
  - https://link.springer.com/chapter/10.1007/978-3-030-58951-6_13

> $ _