CyberCamp 2019

Cloud Security – Container Edition

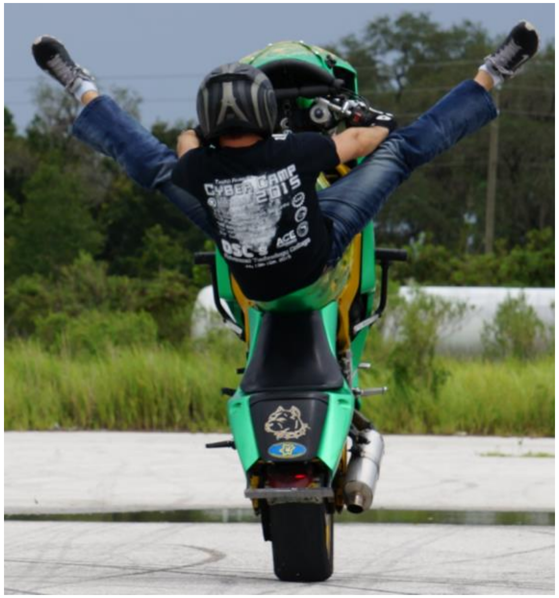
Ricky Payne CISSP, AWS CSA, RHCE, RHCSA, Security+ , BS-IST, AS-CET

About Me

- Over 13.5 years of progressive DevSecOps experience from Intern to Security Analyst -> Architect -> CSO -> Sr. Eng
- Built and operationalized vuln mgmt process for 1000s of globally distributed machines
- Built “Gold Standard” federal security programs that produced 10+ federally certified systems.
- Expert generalist: from pre-sales proposal work, policy and reference architecture development, requirements decomposition into agile sprints, proof of concepts, implementation, operations, and technical training to incident response.

- Mentored/taught Windows and Linux security at CyberPatriot/CyberCamps since 2013. Accomplishments include 1st and 2nd in State and 1st in Regionals.

- As a Sr. SecEng for a global, Silicon Valley InsureTech firm, engaged in all SecOps activities for 1000s of heterogeneous machines across multiple cloud technologies



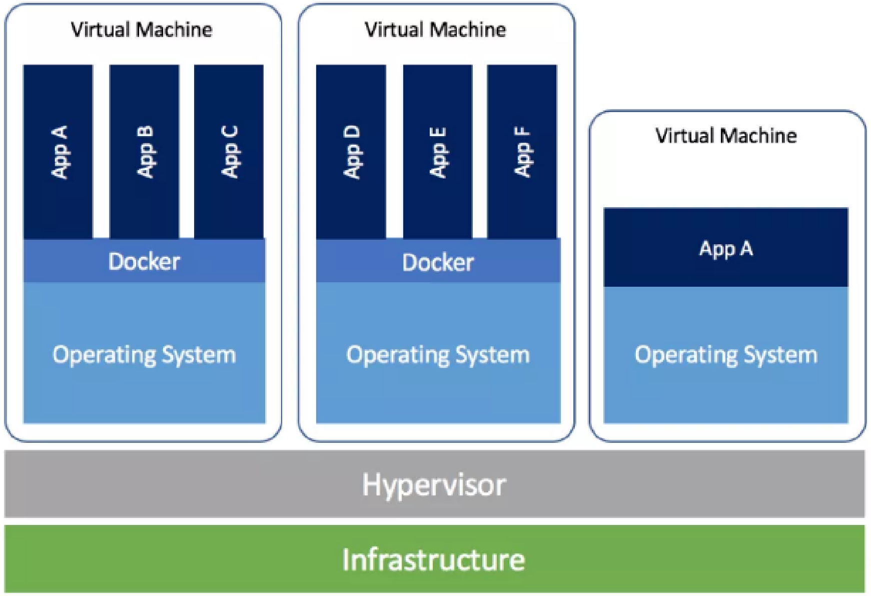
Ricky Payne
CISSP, AWS CSA, RHCE, RHCSA, Security+, BS-IST, AS-CET
grep.rickp@gmail.com
@RickPayne929

Agenda

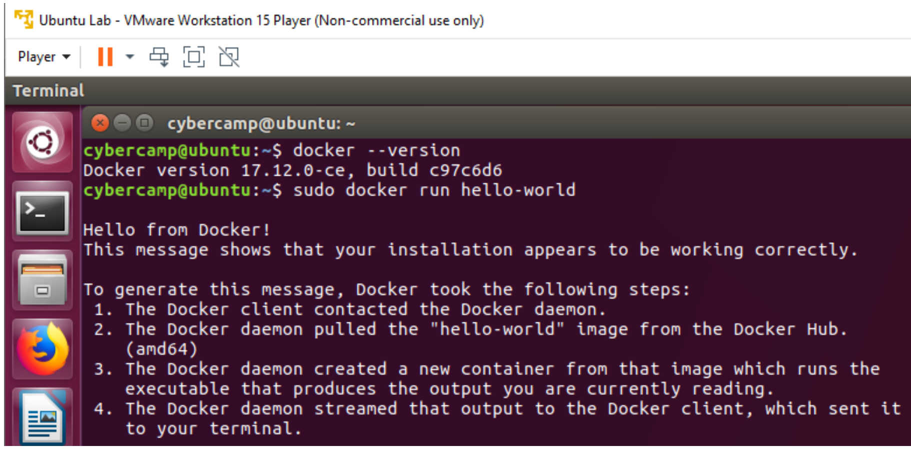
- Presentation scope / goal
- Container ecosystem
- Cloud native ecosystem
- What are containers?
- Orchestration service example
- What companies support them?
- What’s the fuss about?
- Deployment paradigm shift
- Usage: At what scale?
- Container threats
- Container vulnerabilities
- Container incident
- Container risks
- Securing containers
- What we learned
- Hands-on session in room Eng1-187

Presentation scope / goal

Scope – Focusing on running a Docker container inside a VM.



Goal – Prepare students for a hands-on Docker session.


Ubuntu Lab - VMware Workstation 15 Player (Non-commercial use only)
Player
Terminal
cybercamp@ubuntu: ~

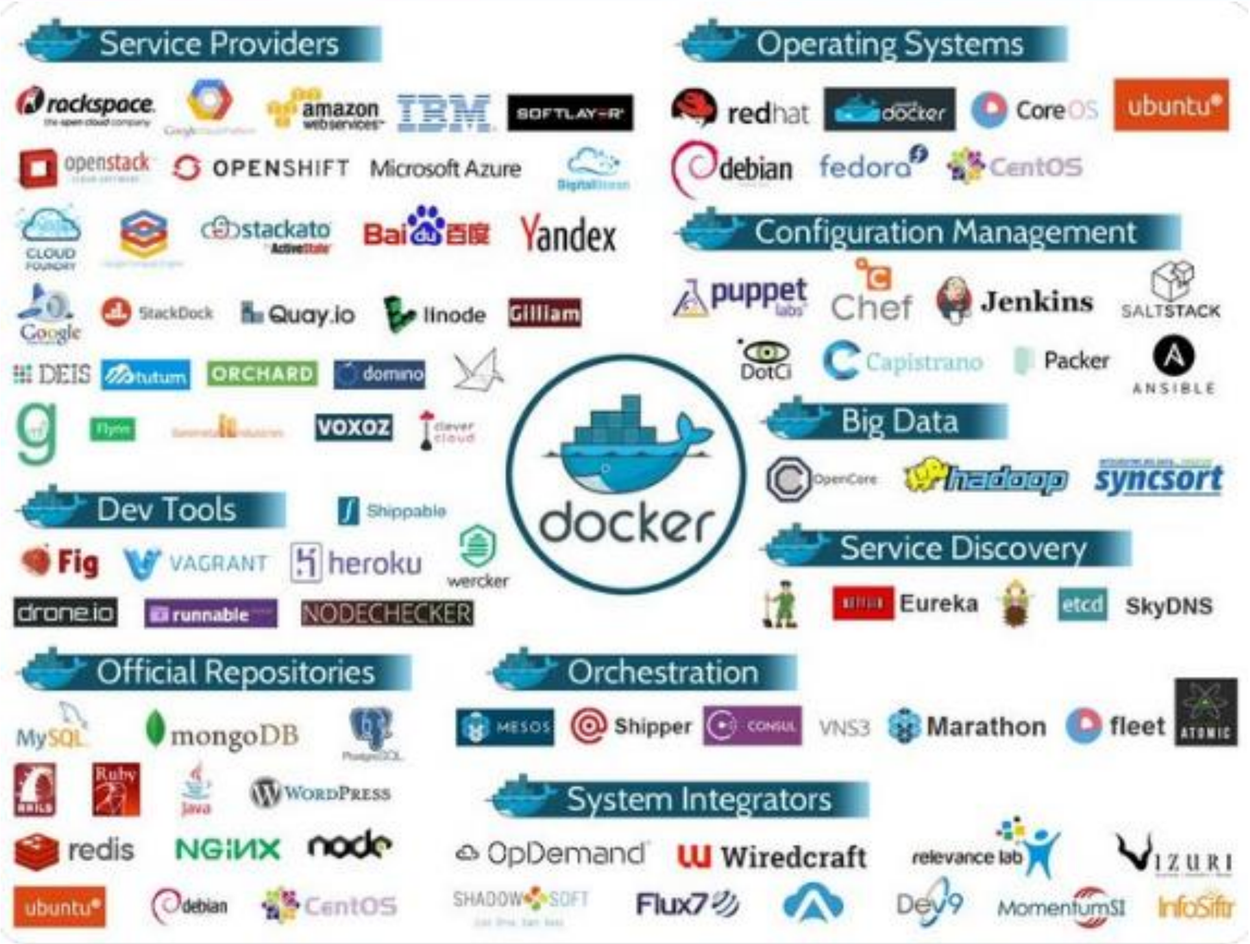
```
cybercamp@ubuntu:~$ docker --version
Docker version 17.12.0-ce, build c97c6d6
cybercamp@ubuntu:~$ sudo docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
```

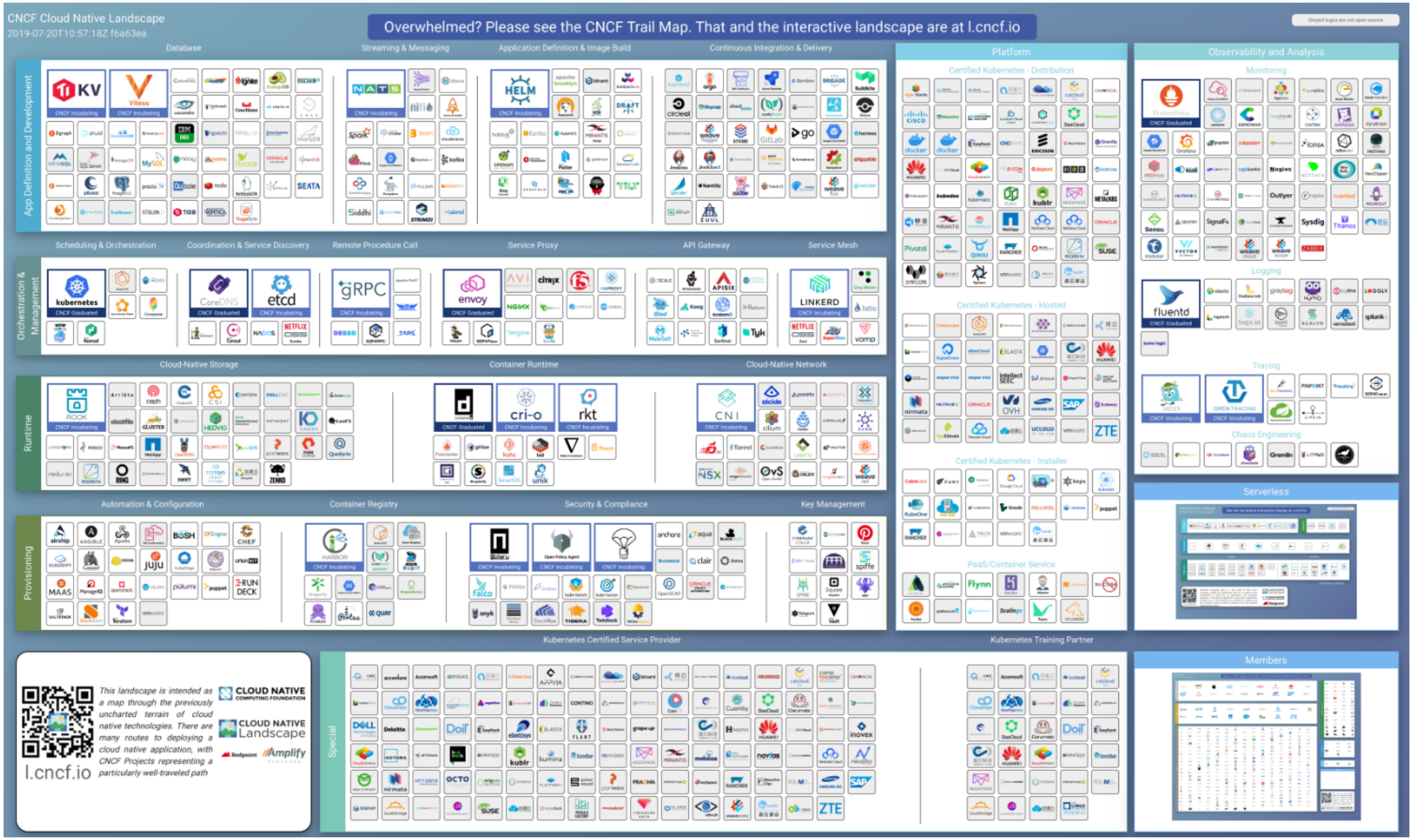
Container ecosystem

Docker's partner ecosystem.



Cloud native ecosystem

Cloud native ecosystem



CNCF Cloud Native Landscape
2019-07-20T10:57:18Z f6a63ea

Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path

l.cncf.io

What are containers?

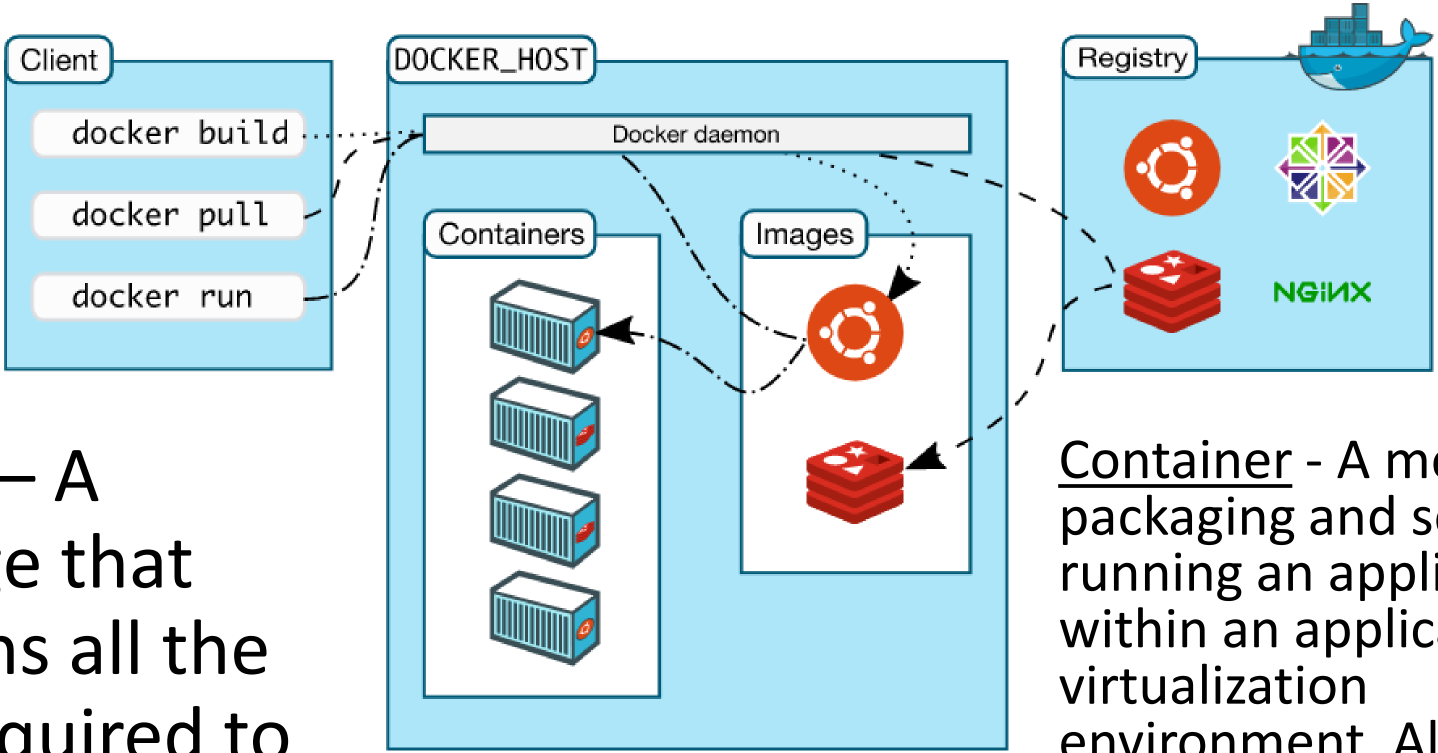
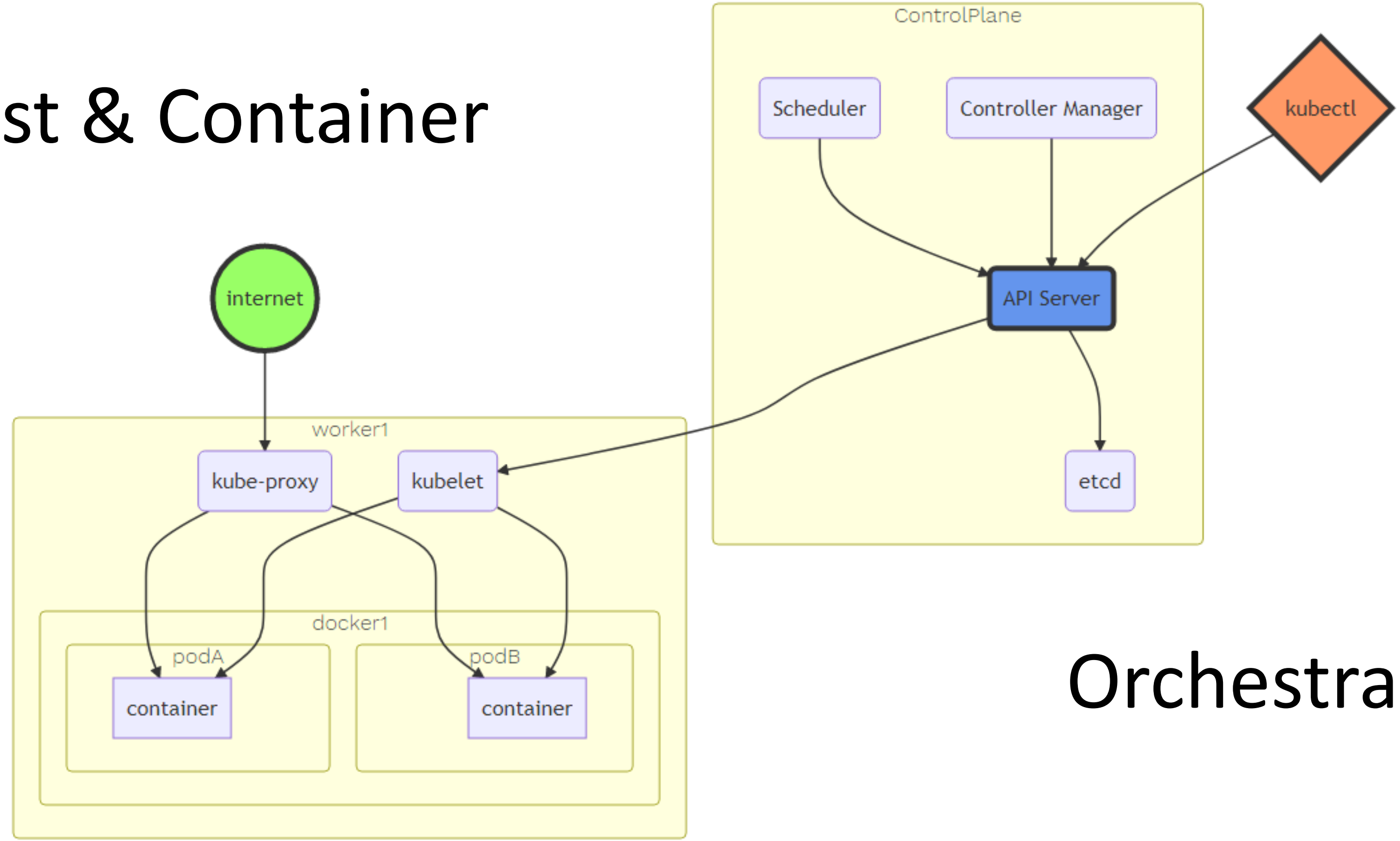


Image – A package that contains all the files required to run a container.[1]

Container - A method for packaging and securely running an application within an application virtualization environment. Also known as an application container or a server application container.[1]

Orchestration service example

Host & Container



Orchestration

What companies support them?

RedHat, Google, AWS, VMware, Microsoft, Dell have been partnered, teamed, integrated, supporting…since 2014[3].

What’s the fuss about?

VISA

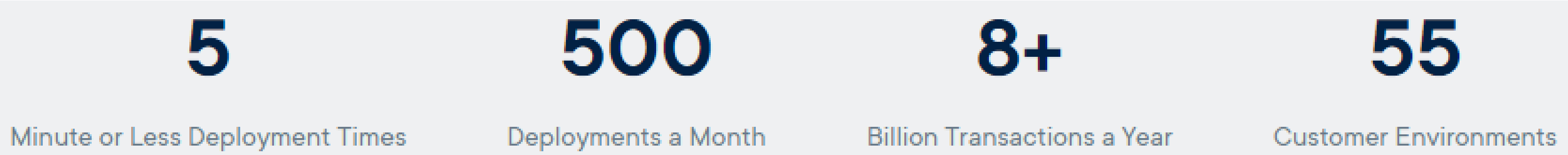
10x increase in scalability

PayPal

+50% app deploy productivity increase

kadaster

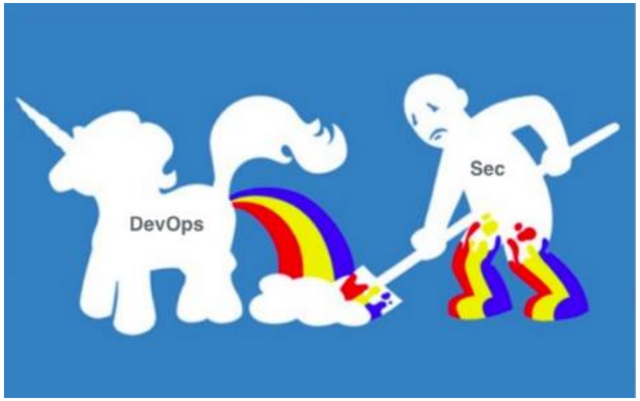
Netherlands government realized 100x faster deployments 5 -> 500/mo

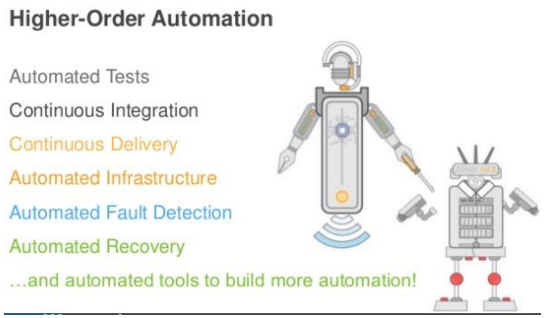


5	500	8+	55
Minute or Less Deployment Times	Deployments a Month	Billion Transactions a Year	Customer Environments

Deployment paradigm shift

Now a developer must become fluent in software testing, deployment, telemetry and even security. Developers will be responsible for securing their own work![4]



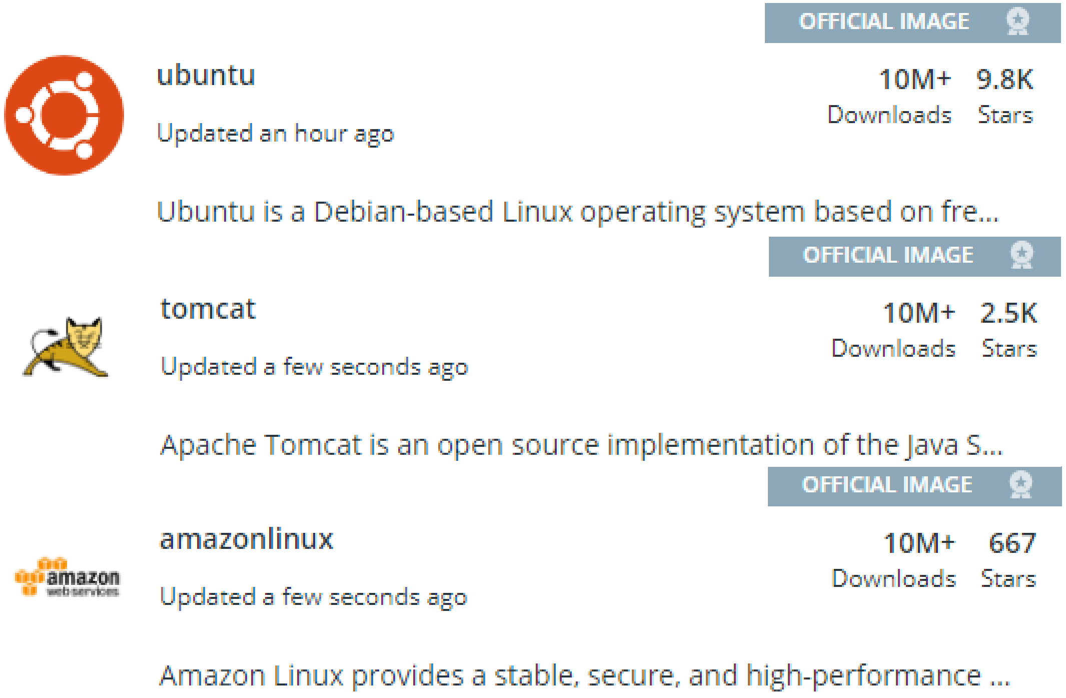


Higher-Order Automation

Automated Tests
Continuous Integration
Continuous Delivery
Automated Infrastructure
Automated Fault Detection
Automated Recovery
...and automated tools to build more automation!

Usage: At what scale?

A quick browse of Docker Hub’s image repo shows the popularity with well over 1 BILLION downloads.



OFFICIAL IMAGE

ubuntu

Updated an hour ago

10M+ Downloads

9.8K Stars

Ubuntu is a Debian-based Linux operating system based on fre...

OFFICIAL IMAGE

tomcat

Updated a few seconds ago

10M+ Downloads

2.5K Stars

Apache Tomcat is an open source implementation of the Java S...

OFFICIAL IMAGE

amazonlinux

Updated a few seconds ago

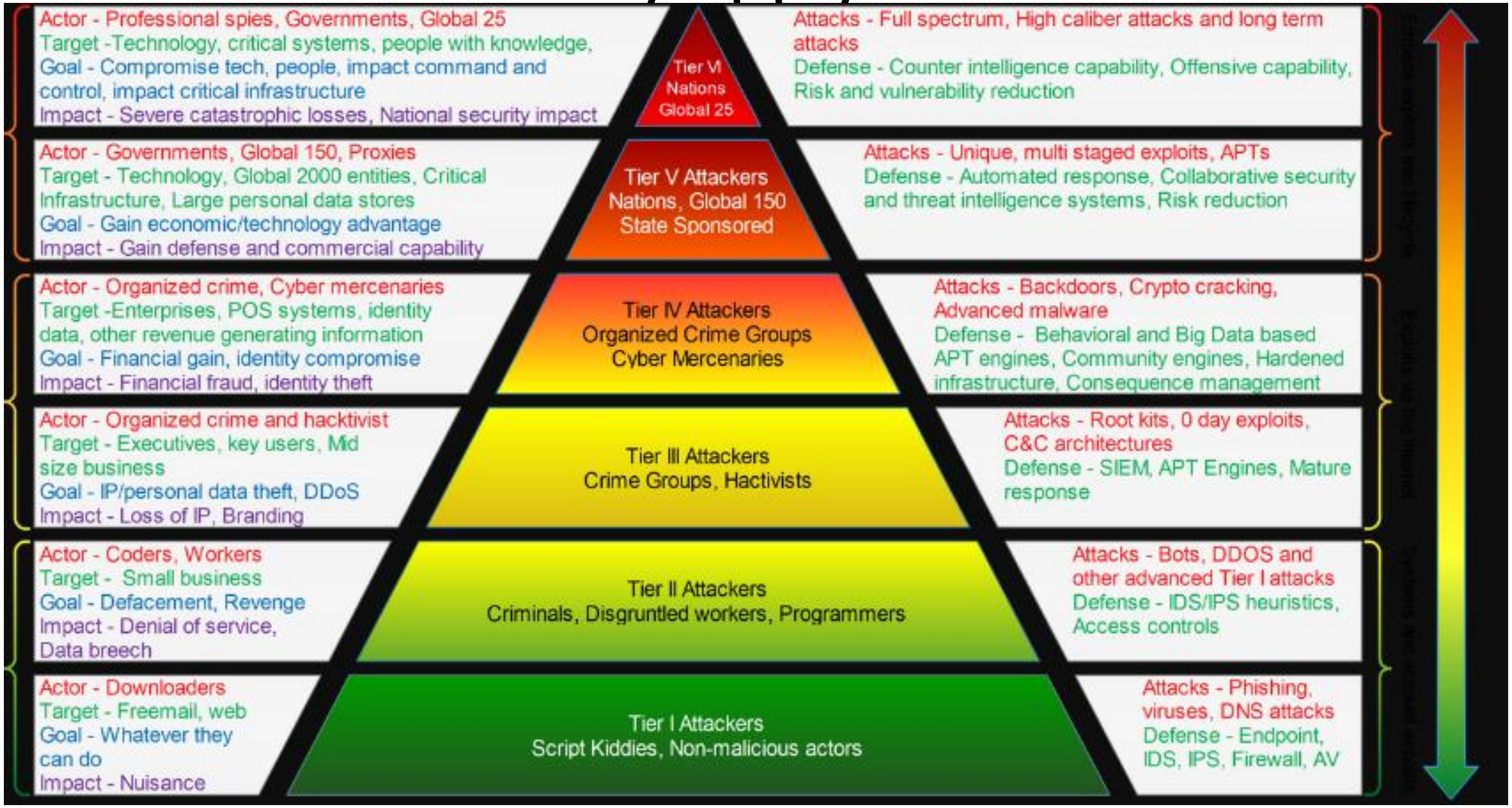
10M+ Downloads

667 Stars

Amazon Linux provides a stable, secure, and high-performance ...

Container Threats?

Do and will they apply to containers?



Actor - Professional spies, Governments, Global 25
Target -Technology, critical systems, people with knowledge,
Goal - Compromise tech, people, impact command and control, impact critical infrastructure
Impact - Severe catastrophic losses, National security impact

Tier VI
Nations
Global 25

Attacks - Full spectrum, High caliber attacks and long term attacks
Defense - Counter intelligence capability, Offensive capability, Risk and vulnerability reduction

Actor - Governments, Global 150, Proxies
Target - Technology, Global 2000 entities, Critical Infrastructure, Large personal data stores
Goal - Gain economic/technology advantage
Impact - Gain defense and commercial capability

Tier V Attackers
Nations, Global 150
State Sponsored

Attacks - Unique, multi staged exploits, APTs
Defense - Automated response, Collaborative security and threat intelligence systems, Risk reduction

Actor - Organized crime, Cyber mercenaries
Target -Enterprises, POS systems, identity data, other revenue generating information
Goal - Financial gain, identity compromise
Impact - Financial fraud, identity theft

Tier IV Attackers
Organized Crime Groups
Cyber Mercenaries

Attacks - Backdoors, Crypto cracking, Advanced malware
Defense - Behavioral and Big Data based APT engines, Community engines, Hardened infrastructure, Consequence management

Actor - Organized crime and hacktivist
Target - Executives, key users, Mid size business
Goal - IP/personal data theft, DDoS
Impact - Loss of IP, Branding

Tier III Attackers
Crime Groups, Hactivists

Attacks - Root kits, 0 day exploits, C&C architectures
Defense - SIEM, APT Engines, Mature response

Actor - Coders, Workers
Target - Small business
Goal - Defacement, Revenge
Impact - Denial of service, Data breech

Tier II Attackers
Criminals, Disgruntled workers, Programmers

Attacks - Bots, DDOS and other advanced Tier I attacks
Defense - IDS/IPS heuristics, Access controls

Actor - Downloaders
Target - Freemail, web
Goal - Whatever they can do
Impact - Nuisance

Tier I Attackers
Script Kiddies, Non-malicious actors

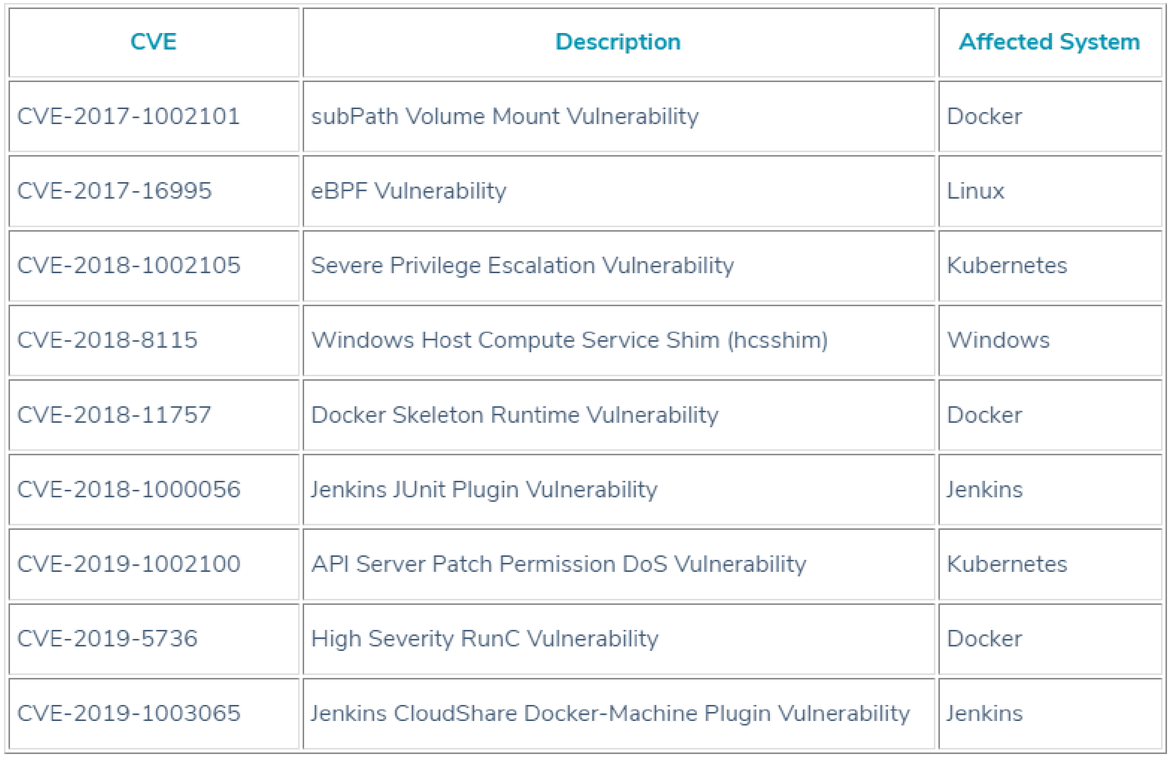
Attacks - Phishing, viruses, DNS attacks
Defense - Endpoint, IDS, IPS, Firewall, AV

A threat is defined in NIST Special Publication (SP) 800-30 as “any circumstance or event with the potential to adversely impact organizational operations and assets, individuals, other organizations, or the Nation through an information system via unauthorized access, destruction, disclosure, or modification of information, and/or denial of service.”

Container Vulnerabilities

CVE-2019-5736: High Severity RunC Vulnerability

This vulnerability allows an attacker to potentially compromise the container host. The vulnerability allows a malicious container to overwrite the host runc binary and gain root-level code execution capability on the host.



CVE	Description	Affected System
CVE-2017-1002101	subPath Volume Mount Vulnerability	Docker
CVE-2017-16995	eBPF Vulnerability	Linux
CVE-2018-1002105	Severe Privilege Escalation Vulnerability	Kubernetes
CVE-2018-8115	Windows Host Compute Service Shim (hcsshim)	Windows
CVE-2018-11757	Docker Skeleton Runtime Vulnerability	Docker
CVE-2018-1000056	Jenkins JUnit Plugin Vulnerability	Jenkins
CVE-2019-1002100	API Server Patch Permission DoS Vulnerability	Kubernetes
CVE-2019-5736	High Severity RunC Vulnerability	Docker
CVE-2019-1003065	Jenkins CloudShare Docker-Machine Plugin Vulnerability	Jenkins

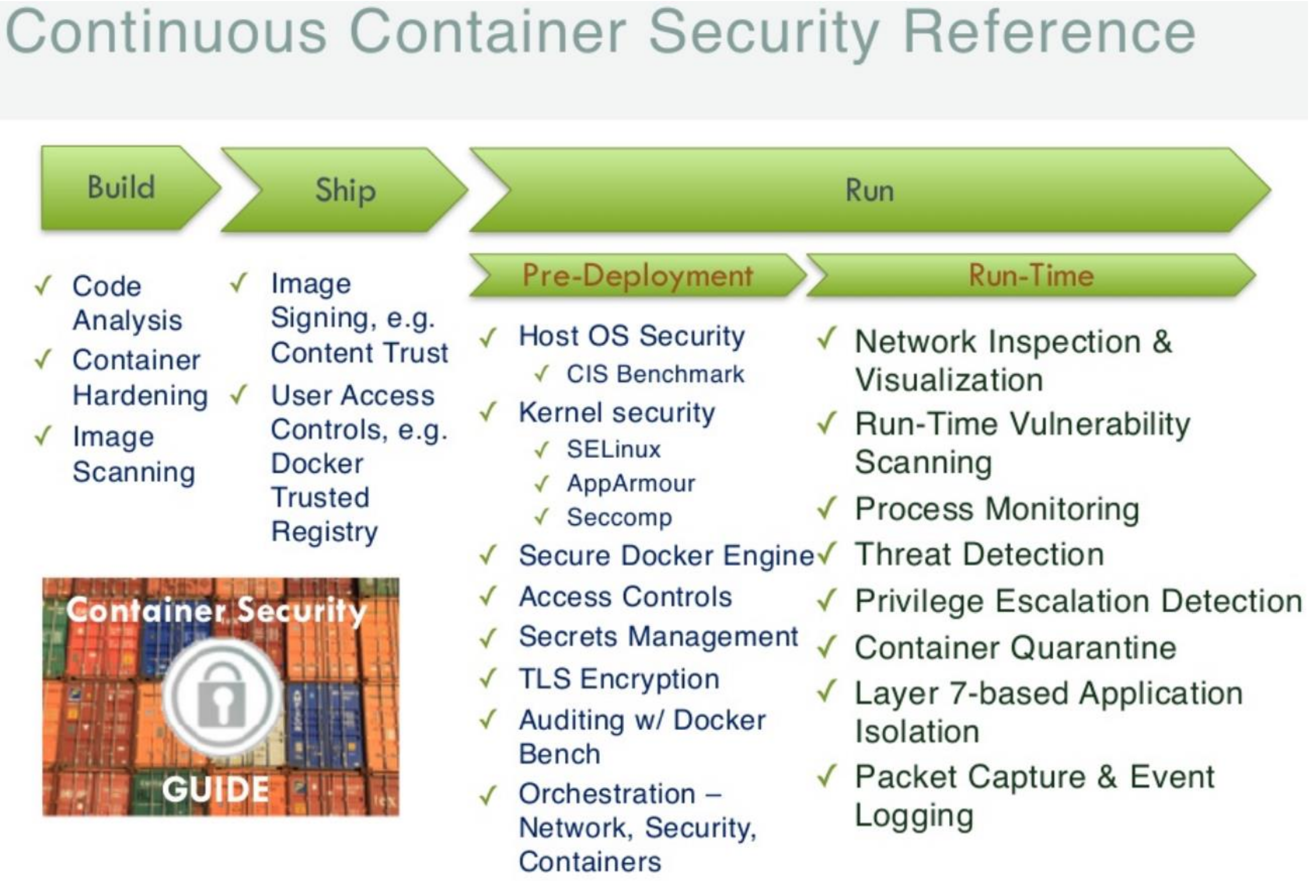
Container incident

Docker Hub – 17 Cryptomining containers. 5M downloads

Container risks

- Awareness & Training – Image source integrity
- Access Control – Container backdoors?
- Auditing – Visibility across orchestration service, Host, Container
- Identification and Authentication – Embedded secrets
- Least Functionality – Limit services & exposed ports
- Permissions – PoLP across orchestration service, Host, Container
- Patching – Vuln mgmt across orchestration service, Host, Container

Securing containers



Continuous Container Security Reference

Build

- ✓ Code Analysis
- ✓ Container Hardening
- ✓ Image Scanning

Ship

- ✓ Image Signing, e.g. Content Trust
- ✓ User Access Controls, e.g. Docker Trusted Registry

Container Security

GUIDE

Run

Pre-Deployment

- ✓ Host OS Security
  - ✓ CIS Benchmark
- ✓ Kernel security
  - ✓ SELinux
  - ✓ AppArmour
  - ✓ Seccomp
- ✓ Secure Docker Engine
- ✓ Access Controls
- ✓ Secrets Management
- ✓ TLS Encryption
- ✓ Auditing w/ Docker Bench
- ✓ Orchestration – Network, Security, Containers

Run-Time

- ✓ Network Inspection & Visualization
- ✓ Run-Time Vulnerability Scanning
- ✓ Process Monitoring
- ✓ Threat Detection
- ✓ Privilege Escalation Detection
- ✓ Container Quarantine
- ✓ Layer 7-based Application Isolation
- ✓ Packet Capture & Event Logging

What we learned…

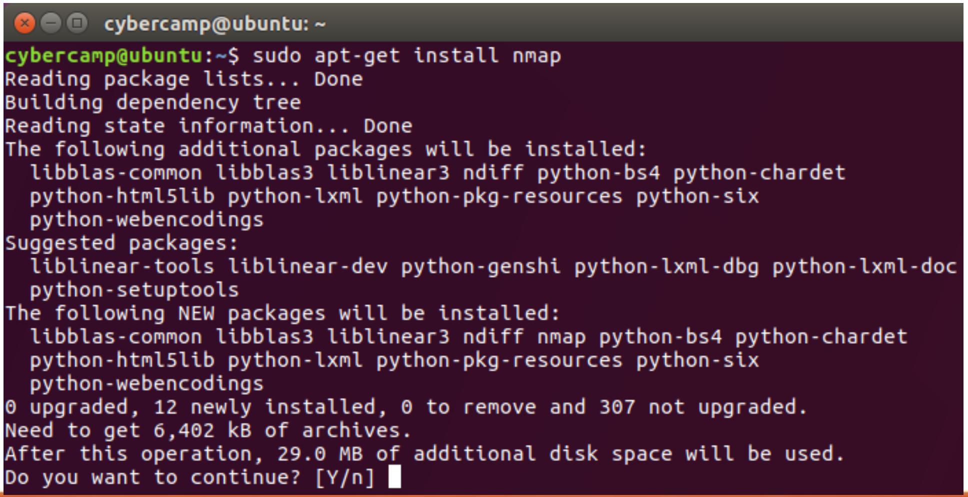
- If you haven't noticed, they're here!
- Massive support system
- Widely used
- Rapid adoption
- Container security conceptually parallels standard practices

Hands-on session

- Host: Update repo list
- Host: Install Docker
- Host: Find secure image
- Host: Pull secure image
- Host, container: Run image
- Container: Run images as ?
- Container: What’s running?
- Host, container: Persistence
- Host, container: Who can run images?
- Host, container: Who can run images? (cont)
- Challenge: PoLP Image build

Update repo list

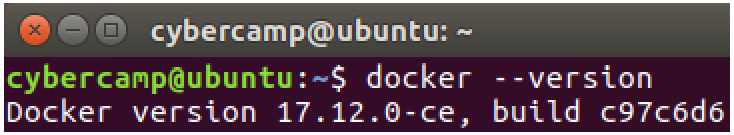
- $ sudo apt-get update ??
- $ wget https://github.com/rickpayne929
- $ chmod +x ubuntu17_archiverepo.sh
- $ ./ubuntu17_archiverepo.sh



```
cybercamp@ubuntu:~$ sudo apt-get install nmap
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libblas-common libblas3 liblinear3 ndiff python-bs4 python-chardet
  python-html5lib python-lxml python-pkg-resources python-six
  python-webencodings
Suggested packages:
  liblinear-tools liblinear-dev python-genshi python-lxml-dbg python-lxml-doc
  python-setuptools
The following NEW packages will be installed:
  libblas-common libblas3 liblinear3 ndiff nmap python-bs4 python-chardet
  python-html5lib python-lxml python-pkg-resources python-six
  python-webencodings
0 upgraded, 12 newly installed, 0 to remove and 307 not upgraded.
Need to get 6,402 kB of archives.
After this operation, 29.0 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Install Docker

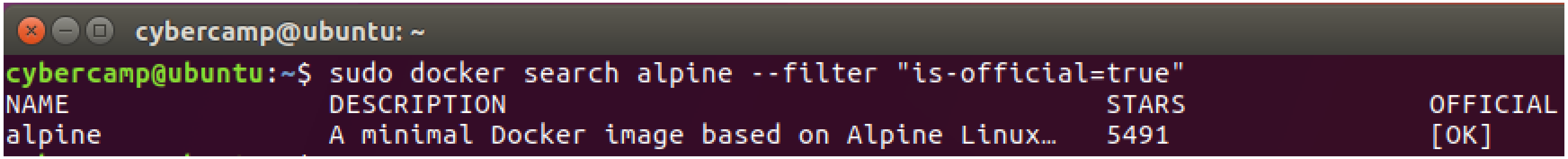
- Connect to Archive Repos
- $ sudo apt-get update   ??
- $ wget https://github.com/rickpayne929
- $ chmod +x ubuntu_install_docker.sh
- $ ./ubuntu_install_docker.sh
- $ docker –version



```
cybercamp@ubuntu:~$ docker --version
Docker version 17.12.0-ce, build c97c6d6
```

Find secure image

- https://hub.docker.com/
  - Official images
  - Alpine vulnerability
- $ sudo docker search <image>
- $ sudo docker search alpine --filter "is-official=true"

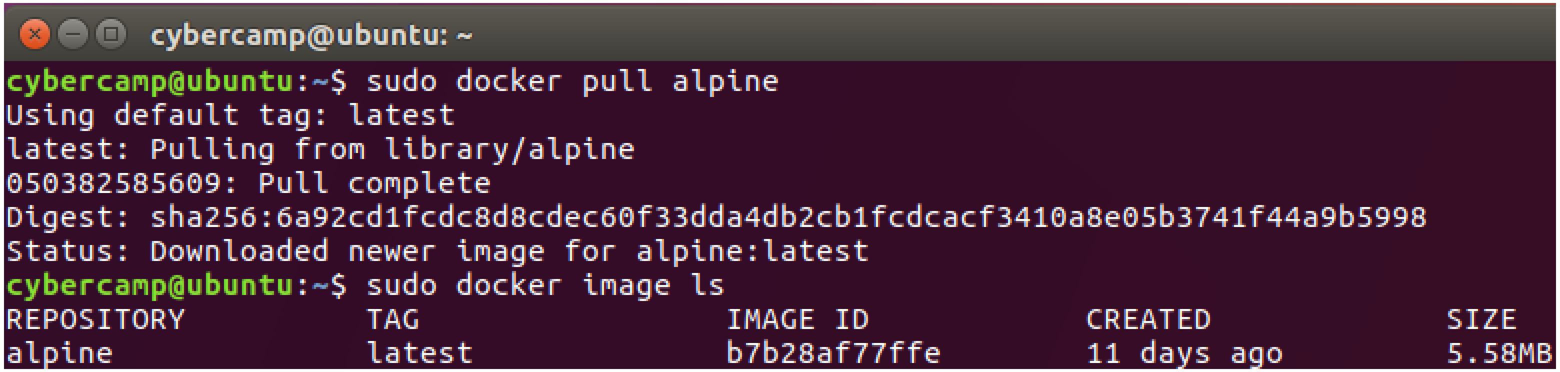


cybercamp@ubuntu: ~

```
cybercamp@ubuntu:~$ sudo docker search alpine --filter "is-official=true"
NAME                DESCRIPTION                                     STARS               OFFICIAL
alpine              A minimal Docker image based on Alpine Linux…   5491                [OK]
```

Pull secure image

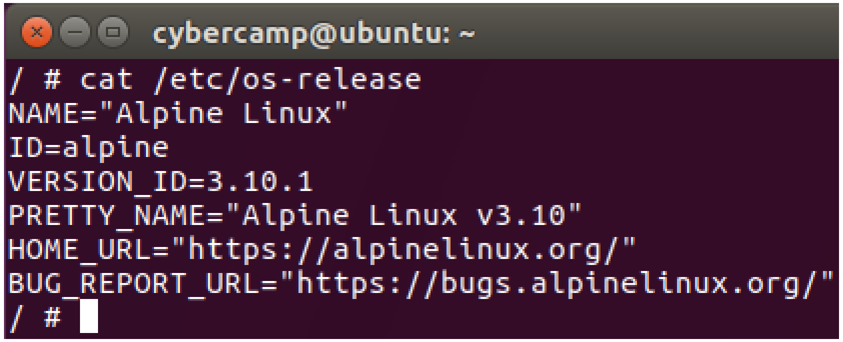
- https://hub.docker.com/
  - Official images – Version tag
  - Alpine vulnerability – What version?
- $ sudo docker pull <image>
- $ sudo docker image ls



```
cybercamp@ubuntu:~$ sudo docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
050382585609: Pull complete
Digest: sha256:6a92cd1fcdc8d8cdec60f33dda4db2cb1fcdcacf3410a8e05b3741f44a9b5998
Status: Downloaded newer image for alpine:latest
cybercamp@ubuntu:~$ sudo docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
alpine              latest              b7b28af77ffe        11 days ago         5.58MB
```

Run image

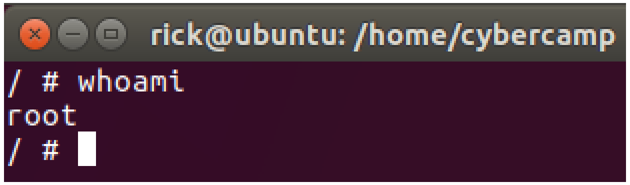
- $ sudo docker run --help | egrep 'interactive|tty'
- $ sudo docker run -it alpine
- / # cat /etc/os-release



```
/ # cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.10.1
PRETTY_NAME="Alpine Linux v3.10"
HOME_URL="https://alpinelinux.org/"
BUG_REPORT_URL="https://bugs.alpinelinux.org/"
/ #
```

Run image as ?

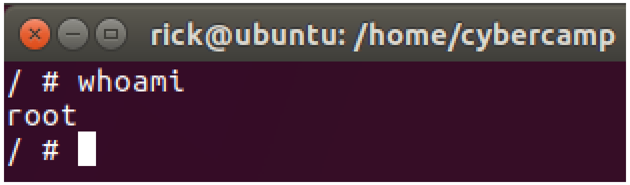
- / # whoami
- / # exit
- $ vi Dockerfile
- Tomorrow → We'll create a Dockerfile, add a standard user, build a container, and run a process as a non-root user.



```
/ # whoami
root
/ # 
```

Run image as ? (cont)

- / # whoami

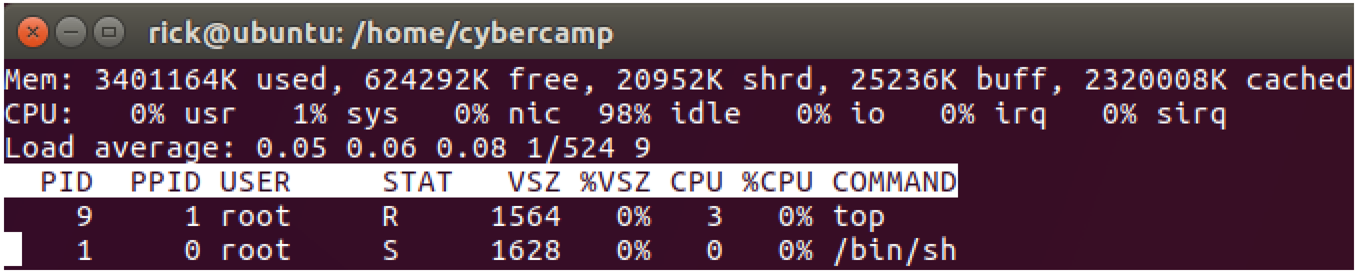


```
/ # whoami
root
/ #
```

- Tomorrow → We'll create a Dockerfile, add a standard user, build a container, and run a process as a non-root user.

What's running?

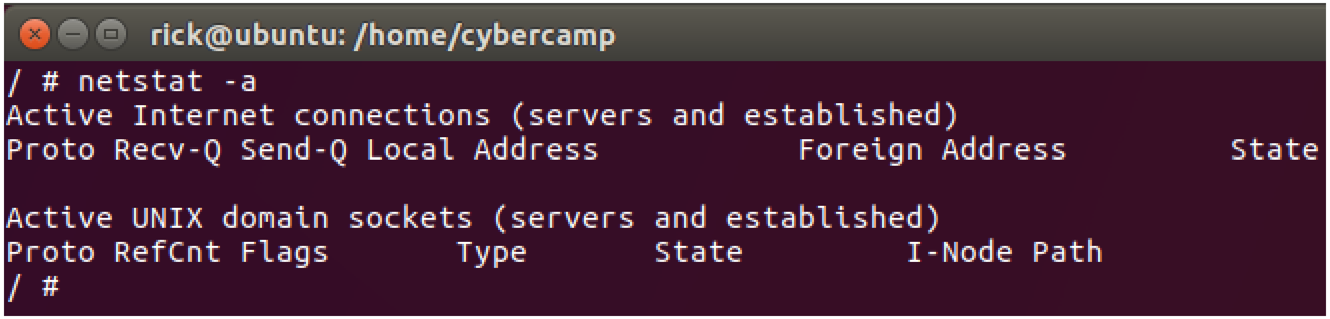
- Default processes
- / # top



rick@ubuntu: /home/cybercamp

```
Mem: 3401164K used, 624292K free, 20952K shrd, 25236K buff, 2320008K cached
CPU:   0% usr   1% sys   0% nic  98% idle   0% io   0% irq   0% sirq
Load average: 0.05 0.06 0.08 1/524 9
  PID  PPID USER     STAT   VSZ %VSZ CPU %CPU COMMAND
    9     1 root     R     1564   0%   3   0% top
    1     0 root     S     1628   0%   0   0% /bin/sh
```

- Open ports?
- /# netstat -a



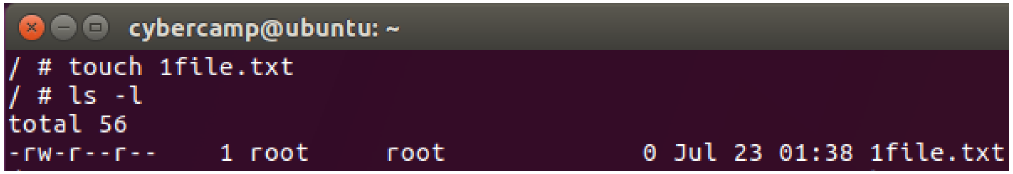
rick@ubuntu: /home/cybercamp

```
/ # netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
                                                                                
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
/ #
```

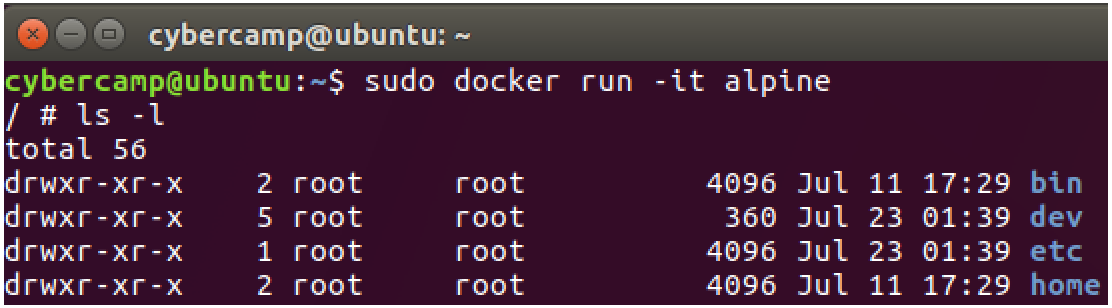
CyberCamp
2019

Persistence

- Image vs container
- / # touch 1file.txt
- / # ls -l
- / # exit
- $ sudo docker run -it alpine
- / # ls -l



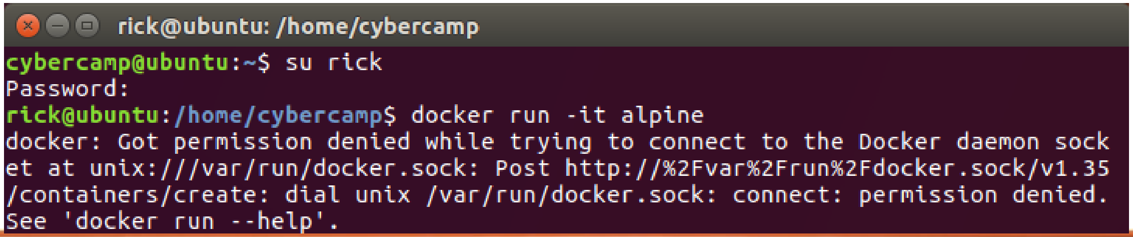
```
/ # touch 1file.txt
/ # ls -l
total 56
-rw-r--r--    1 root     root             0 Jul 23 01:38 1file.txt
```



```
cybercamp@ubuntu:~$ sudo docker run -it alpine
/ # ls -l
total 56
drwxr-xr-x    2 root     root          4096 Jul 11 17:29 bin
drwxr-xr-x    5 root     root           360 Jul 23 01:39 dev
drwxr-xr-x    1 root     root          4096 Jul 23 01:39 etc
drwxr-xr-x    2 root     root          4096 Jul 11 17:29 home
```

Who can run images?

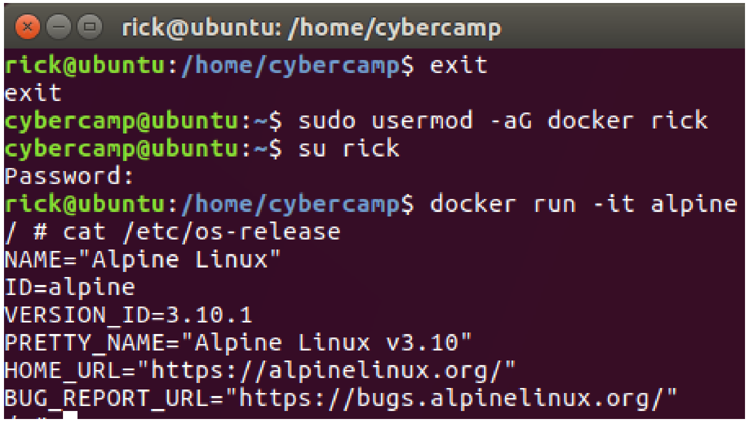
- Root users of course
- Let’s take a PoLP approach on the host
- $ getent group docker, less /etc/group
- $ sudo adduser rick →
- $ su rick
- $ sudo docker run -it alpine



```
cybercamp@ubuntu:~$ su rick
Password:
rick@ubuntu:/home/cybercamp$ docker run -it alpine
docker: Got permission denied while trying to connect to the Docker daemon sock
et at unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.35
/containers/create: dial unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
```

Who can run images? (cont)

- $ exit
- $ sudo usermod -aG docker rick
- $ sudo docker run -it alpine
- $ su rick
- $ docker run -it alpine
- / # cat /etc/os-release



```
rick@ubuntu:/home/cybercamp$ exit
exit
cybercamp@ubuntu:~$ sudo usermod -aG docker rick
cybercamp@ubuntu:~$ su rick
Password:
rick@ubuntu:/home/cybercamp$ docker run -it alpine
/ # cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.10.1
PRETTY_NAME="Alpine Linux v3.10"
HOME_URL="https://alpinelinux.org/"
BUG_REPORT_URL="https://bugs.alpinelinux.org/"
```

Challenge

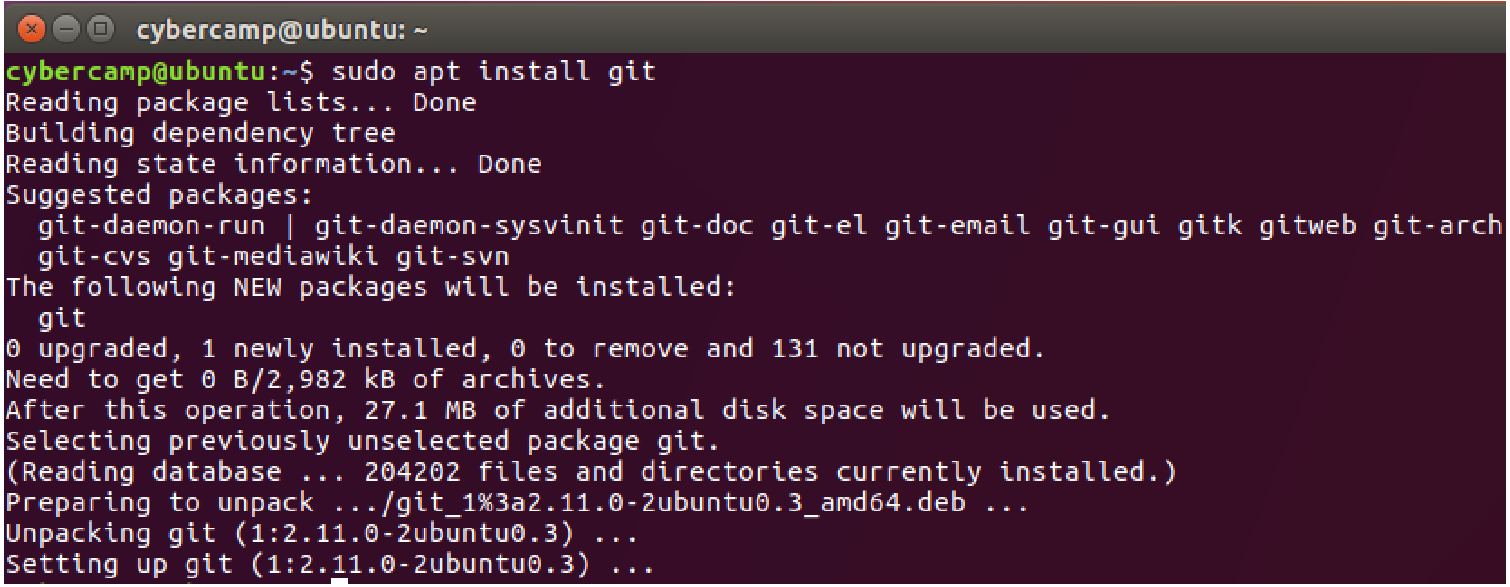
- Time permitting, let’s
  - create a Dockerfile
  - add a standard user
  - build the container
  - run a process as a non-root user
  - confirm the process is running non-root

Hands-on session #2

- Host: Install Git
- Host: CIS Scan
- Host: CIS Audit Finding Remediation
- Host: CIS Remediation Scan
- Host: Build nmap container
- Host, Container: Test nmap container
- Host, Container: Test nmap container #2
- Challenge: Remediate CIS Finding

Install Git

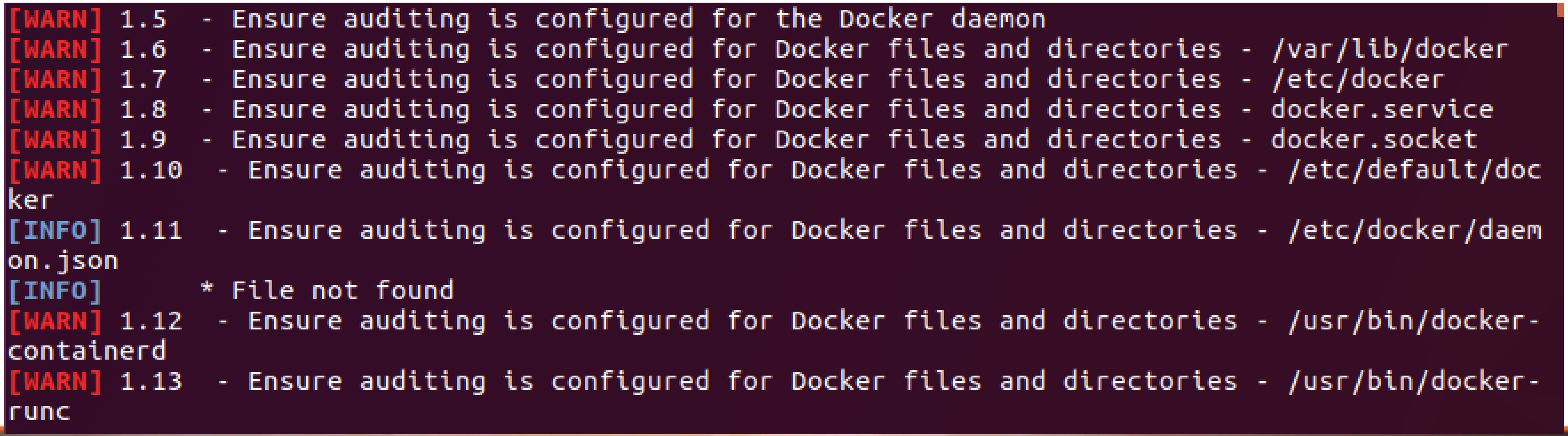
- $ sudo apt-get update
- $ sudo apt install git
- $ git --version



```
cybercamp@ubuntu:~$ sudo apt install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-arch
  git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git
0 upgraded, 1 newly installed, 0 to remove and 131 not upgraded.
Need to get 0 B/2,982 kB of archives.
After this operation, 27.1 MB of additional disk space will be used.
Selecting previously unselected package git.
(Reading database ... 204202 files and directories currently installed.)
Preparing to unpack .../git_1%3a2.11.0-2ubuntu0.3_amd64.deb ...
Unpacking git (1:2.11.0-2ubuntu0.3) ...
Setting up git (1:2.11.0-2ubuntu0.3) ...
```

CIS Scan

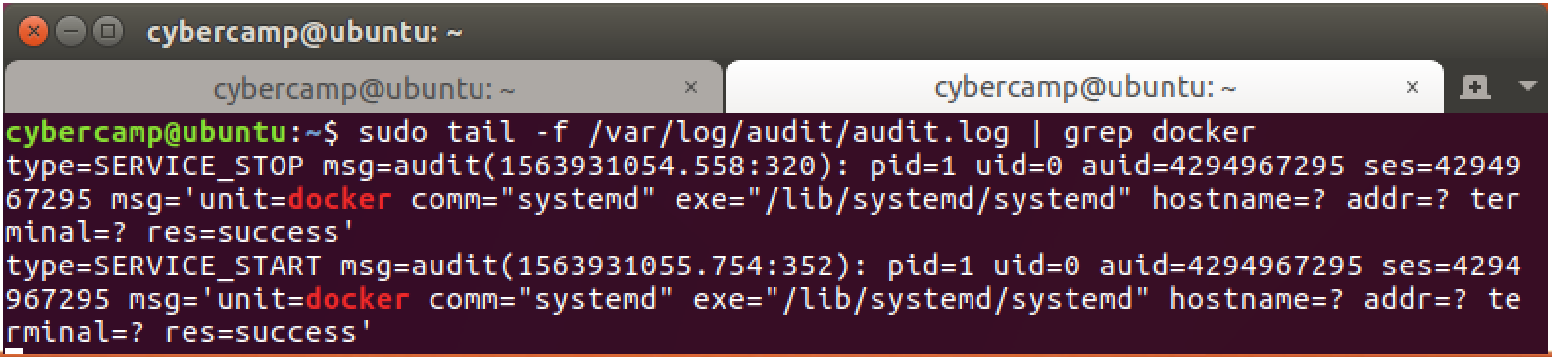
- $ git clone https://github.com/docker/docker-bench-security.git
- $ cd docker-bench-security
- $ sudo sh docker-bench-security.sh



```
[WARN] 1.5  - Ensure auditing is configured for the Docker daemon
[WARN] 1.6  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.7  - Ensure auditing is configured for Docker files and directories - /etc/docker
[WARN] 1.8  - Ensure auditing is configured for Docker files and directories - docker.service
[WARN] 1.9  - Ensure auditing is configured for Docker files and directories - docker.socket
[WARN] 1.10  - Ensure auditing is configured for Docker files and directories - /etc/default/doc
ker
[INFO] 1.11  - Ensure auditing is configured for Docker files and directories - /etc/docker/daem
on.json
[INFO]      * File not found
[WARN] 1.12  - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-
containerd
[WARN] 1.13  - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-
runc
```

CIS Audit Finding Remediation

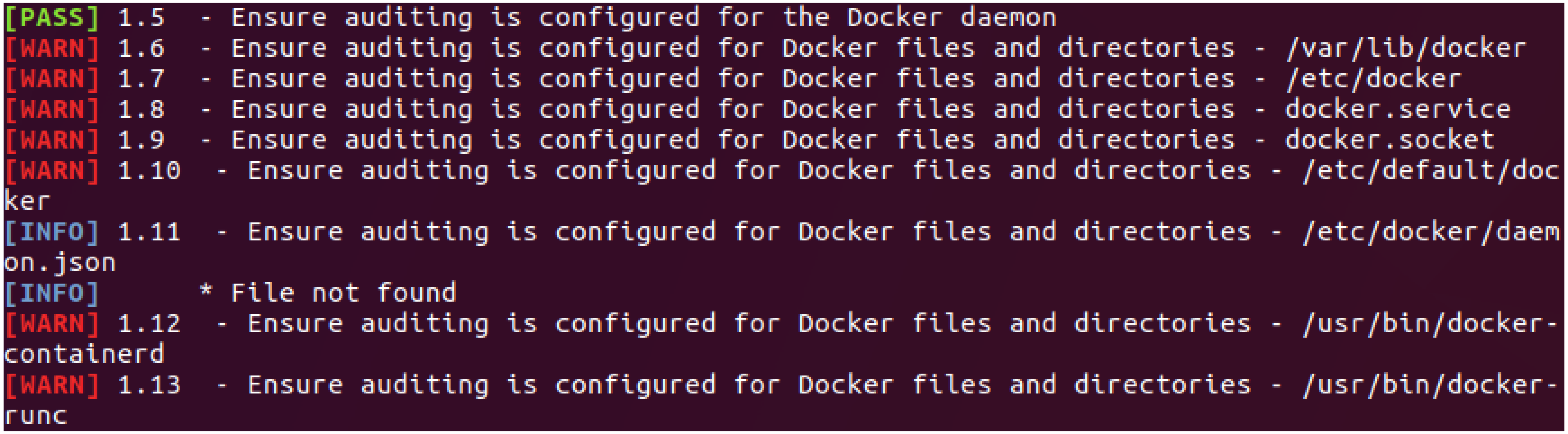
- $ sudo apt install auditd
- $ echo "-w /usr/bin/docker -p wa" | sudo tee -a /etc/audit/rules.d/audit.rules
- $ sudo service auditd restart
- $ sudo grep log_file /etc/audit/auditd.conf
- Ctrl + shift + t, sudo tail -f /var/log/audit/audit.log | grep docker
- Ctrl + PgUp, $ sudo service docker restart



```
cybercamp@ubuntu:~$ sudo tail -f /var/log/audit/audit.log | grep docker
type=SERVICE_STOP msg=audit(1563931054.558:320): pid=1 uid=0 auid=4294967295 ses=42949
67295 msg='unit=docker comm="systemd" exe="/lib/systemd/systemd" hostname=? addr=? ter
minal=? res=success'
type=SERVICE_START msg=audit(1563931055.754:352): pid=1 uid=0 auid=4294967295 ses=4294
967295 msg='unit=docker comm="systemd" exe="/lib/systemd/systemd" hostname=? addr=? te
rminal=? res=success'
```

Host CIS Remediation Scan

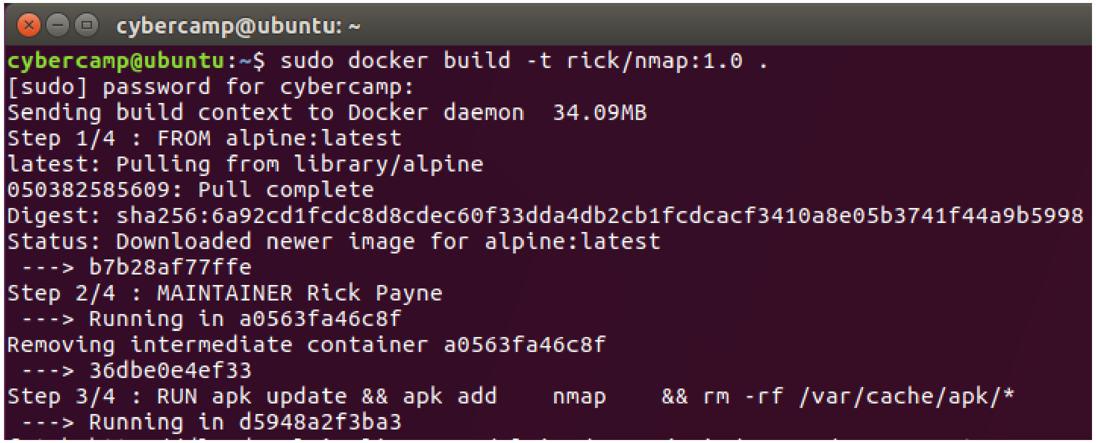
- $ cd docker-bench-security
- $ sudo sh docker-bench-security.sh
- $ cd ~



```
[PASS] 1.5  - Ensure auditing is configured for the Docker daemon
[WARN] 1.6  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.7  - Ensure auditing is configured for Docker files and directories - /etc/docker
[WARN] 1.8  - Ensure auditing is configured for Docker files and directories - docker.service
[WARN] 1.9  - Ensure auditing is configured for Docker files and directories - docker.socket
[WARN] 1.10  - Ensure auditing is configured for Docker files and directories - /etc/default/doc
ker
[INFO] 1.11  - Ensure auditing is configured for Docker files and directories - /etc/docker/daem
on.json
[INFO]      * File not found
[WARN] 1.12  - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-
containerd
[WARN] 1.13  - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-
runc
```

Build nmap container

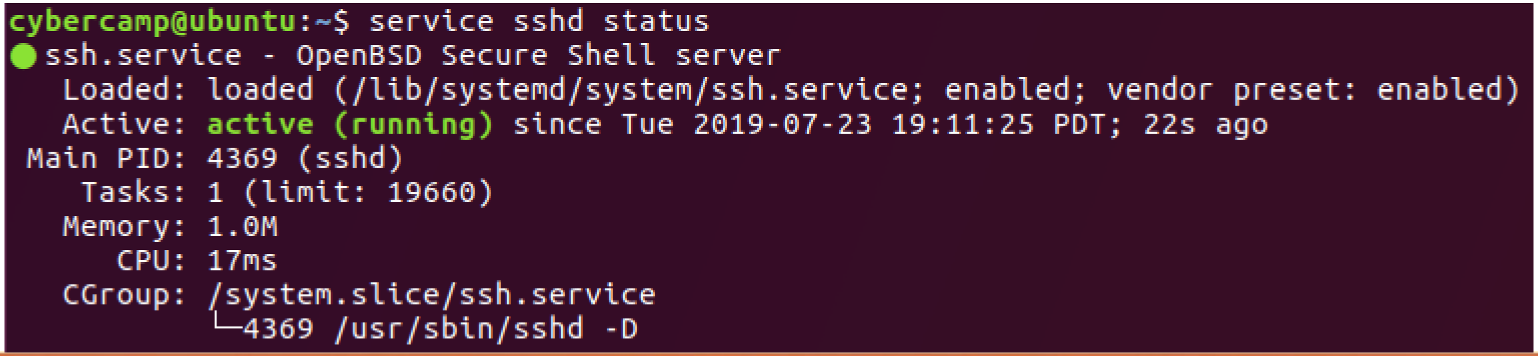
- Paste $ wget https://github.com/rickpayne929
- into $ nano Dockerfile
- $ sudo docker build –t rick/nmap:1.0 .



```
cybercamp@ubuntu:~$ sudo docker build -t rick/nmap:1.0 .
[sudo] password for cybercamp:
Sending build context to Docker daemon  34.09MB
Step 1/4 : FROM alpine:latest
latest: Pulling from library/alpine
050382585609: Pull complete
Digest: sha256:6a92cd1fcdc8d8cdec60f33dda4db2cb1fcdcacf3410a8e05b3741f44a9b5998
Status: Downloaded newer image for alpine:latest
 ---> b7b28af77ffe
Step 2/4 : MAINTAINER Rick Payne
 ---> Running in a0563fa46c8f
Removing intermediate container a0563fa46c8f
 ---> 36dbe0e4ef33
Step 3/4 : RUN apk update && apk add    nmap    && rm -rf /var/cache/apk/*
 ---> Running in d5948a2f3ba3
```

Test nmap container

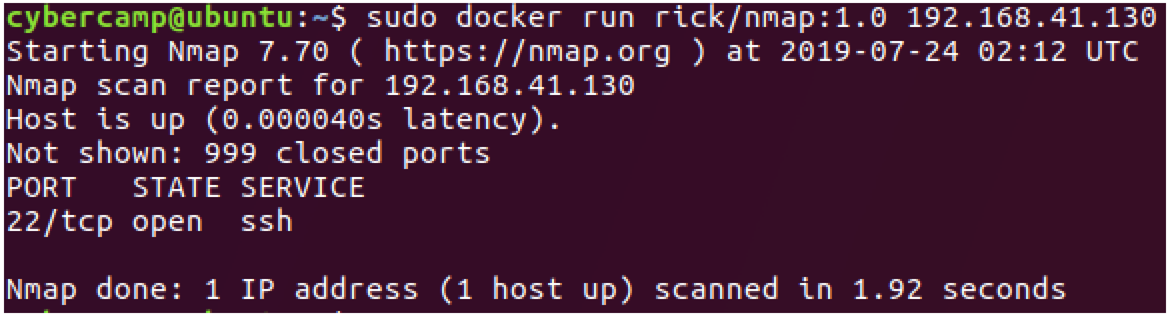
- $ sudo docker images ls
- $ sudo netstat –tulnp
- $ sudo apt install ssh –y
- $ sudo service sshd status
- $ sudo netstat –tulnp



```
cybercamp@ubuntu:~$ service sshd status
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2019-07-23 19:11:25 PDT; 22s ago
 Main PID: 4369 (sshd)
    Tasks: 1 (limit: 19660)
   Memory: 1.0M
      CPU: 17ms
   CGroup: /system.slice/ssh.service
           └─4369 /usr/sbin/sshd -D
```

Test nmap container #2

- $ sudo docker images ls
- $ docker run –it rick/nmap –help
- $ docker run –it rick/nmap:1.0 –help
- $ ipconfig | grep inet
- $ docker run –it rick/nmap:1.0 <inet IP>



```
cybercamp@ubuntu:~$ sudo docker run rick/nmap:1.0 192.168.41.130
Starting Nmap 7.70 ( https://nmap.org ) at 2019-07-24 02:12 UTC
Nmap scan report for 192.168.41.130
Host is up (0.000040s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 1.92 seconds
```

Challenge: Remediate CIS Finding

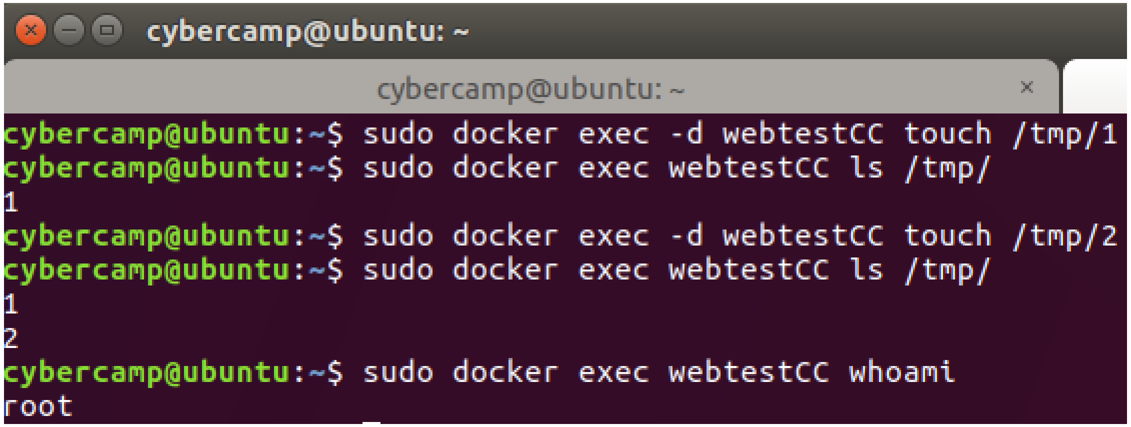
- Time permitting, let’s
  - Identify another CIS finding
  - Remediate
  - Rescan
  - Confirm remediation

Hands-on session #3

- Host: Docker Mgmt Commands
- Host: Docker Mgmt Commands #2
- Host, Container: Cleanup -> Web Server Launch
- Host, Container: Host <-> Container Volume
- Host, Container: Container – root by default?
- Host: Create user-level image
- Challenge: Image Vulnerability Scan

Docker Mgmt Commands

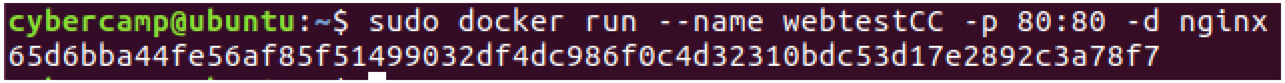
- $ sudo docker –help
- $ sudo docker image –help
- $ sudo docker inspect
- $ sudo docker exec –d <image> touch /tmp/1



```
cybercamp@ubuntu:~$ sudo docker exec -d webtestCC touch /tmp/1
cybercamp@ubuntu:~$ sudo docker exec webtestCC ls /tmp/
1
cybercamp@ubuntu:~$ sudo docker exec -d webtestCC touch /tmp/2
cybercamp@ubuntu:~$ sudo docker exec webtestCC ls /tmp/
1
2
cybercamp@ubuntu:~$ sudo docker exec webtestCC whoami
root
```

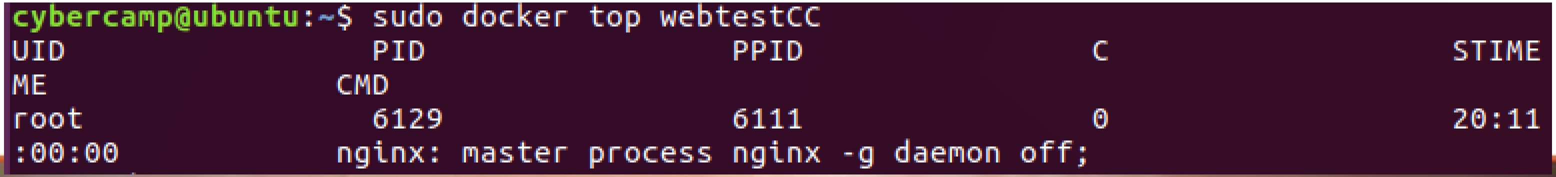
Docker Mgmt Commands #2

- $ docker run --name webtestCC -p 80:80 -d nginx



```
cybercamp@ubuntu:~$ sudo docker run --name webtestCC -p 80:80 -d nginx
65d6bba44fe56af85f51499032df4dc986f0c4d32310bdc53d17e2892c3a78f7
```

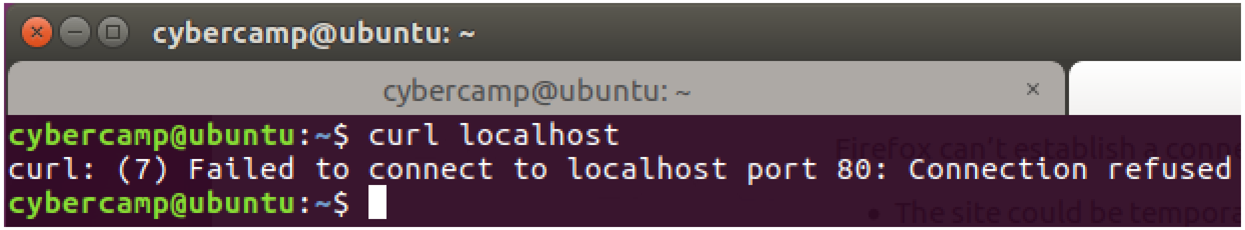
- What’s the container’s resource (CPU, mem) usage?
- Running process? Ports exposed?
- Process running as what user?
- $ sudo docker ps -> top -> stats



```
cybercamp@ubuntu:~$ sudo docker top webtestCC
UID                 PID                 PPID                C                   STIME
ME                  CMD
root                6129                6111                0                   20:11
:00:00              nginx: master process nginx -g daemon off;
```

Cleanup -> Web Server Launch

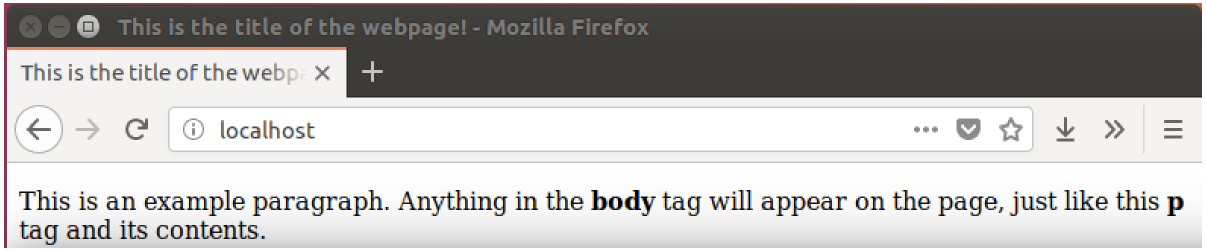
- $ sudo docker container kill webtestCC
- $ sudo docker rm webtestCC
- $ sudo docker run --name nginxtest -v /home/cybercamp/index.html:/usr/share/nginx/html/index.html:rw -d nginx
- $ curl localhost, open Firefox -> goto localhost



```
cybercamp@ubuntu:~$ curl localhost
curl: (7) Failed to connect to localhost port 80: Connection refused
cybercamp@ubuntu:~$
```

Host <-> Container Volume

- $ sudo docker run --name nginxtest -v /home/cybercamp/index.html:/usr/share/nginx/html/index.html:rw -p 80:80 -d nginx
- $ curl localhost, open Firefox -> goto localhost
- $ gedit index.html
- $ sudo docker container nginx restart

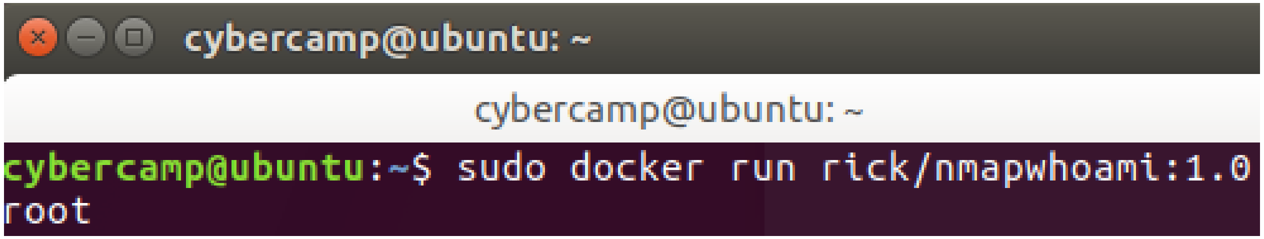


This is the title of the webpage! - Mozilla Firefox
This is the title of the webp
localhost

This is an example paragraph. Anything in the **body** tag will appear on the page, just like this **p** tag and its contents.

Container – root by default?

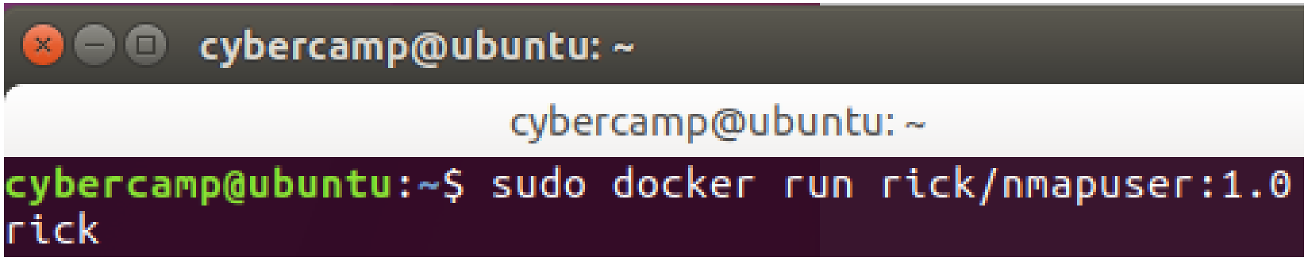
- $ wget https://github.com/rickpayne929
- $ sudo docker build –t rick/nmapwhoami:1.0 .



```
cybercamp@ubuntu:~$ sudo docker run rick/nmapwhoami:1.0
root
```

Create user-level image

- $ wget https://github.com/rickpayne929
- $ sudo docker build –t rick/nmapuser:1.0 .



```
cybercamp@ubuntu:~$ sudo docker run rick/nmapuser:1.0
rick
```

Challenge: Image Vulnerability Scan

- Time permitting, let's
  - Download open source scanner
  - Scan image
  - Generate Report
  - Fix 1 finding
  - Report and compare

References

1. Special Publication 800-190 - Application Container Security Guide. NIST. Retrieved from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-190.pdf
2. Docker overview. Docker. Retrieved from https://docs.docker.com/engine/docker-overview/
3. Infographic: Docker Ecosystem. IT Briefcase. Retrieved from http://www.itbriefcase.net/infographic-docker-ecosystem-2014-year-in-review
4. Docker Threat Modeling. Dr. Wetter. Retrieved from https://www.owasp.org/images/1/17/Dirk_Wetter_-_Docker_Security_Brussels.pdf

References

1. Additional reference citations listed in slide notes