RSAConference2019

San Francisco | March 4–8 | Moscone Center

BETTER.

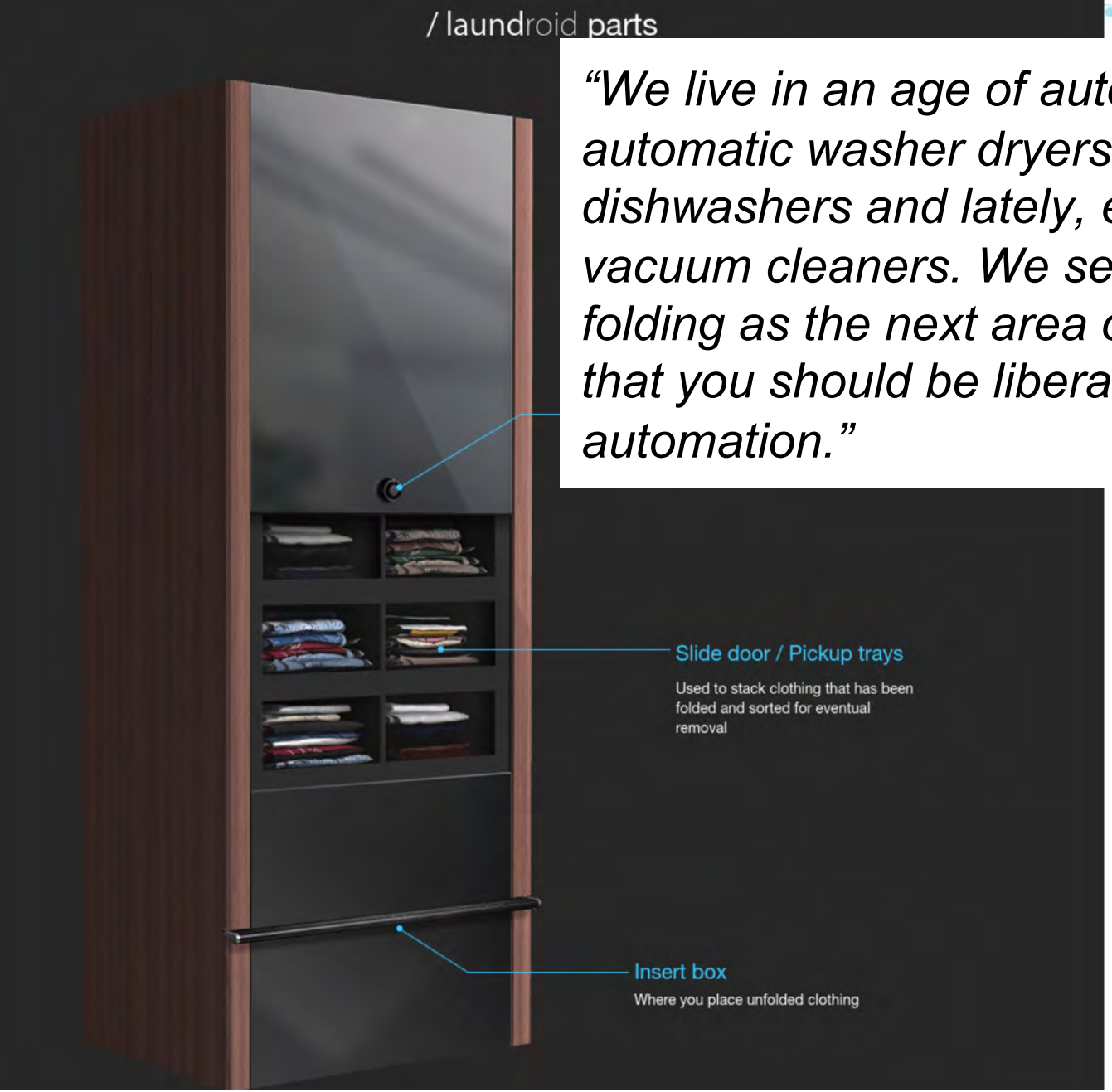
SESSION ID: ASD-W12

How I Learned Docker Security the Hard Way (So You Don't Have To)

Paul Asadoorian

Founder & CTO at Security Weekly

We connect the security community with the security industry through our network of shows and security market validation programs.



"We live in an age of automation — automatic washer dryers, automatic dishwashers and lately, even automatic vacuum cleaners. We see laundry folding as the next area of housework that you should be liberated from by automation."

Me: *"We don't have an artificially intelligent robot that puts away all of our laundry."*

Son: "*We should get one.*"

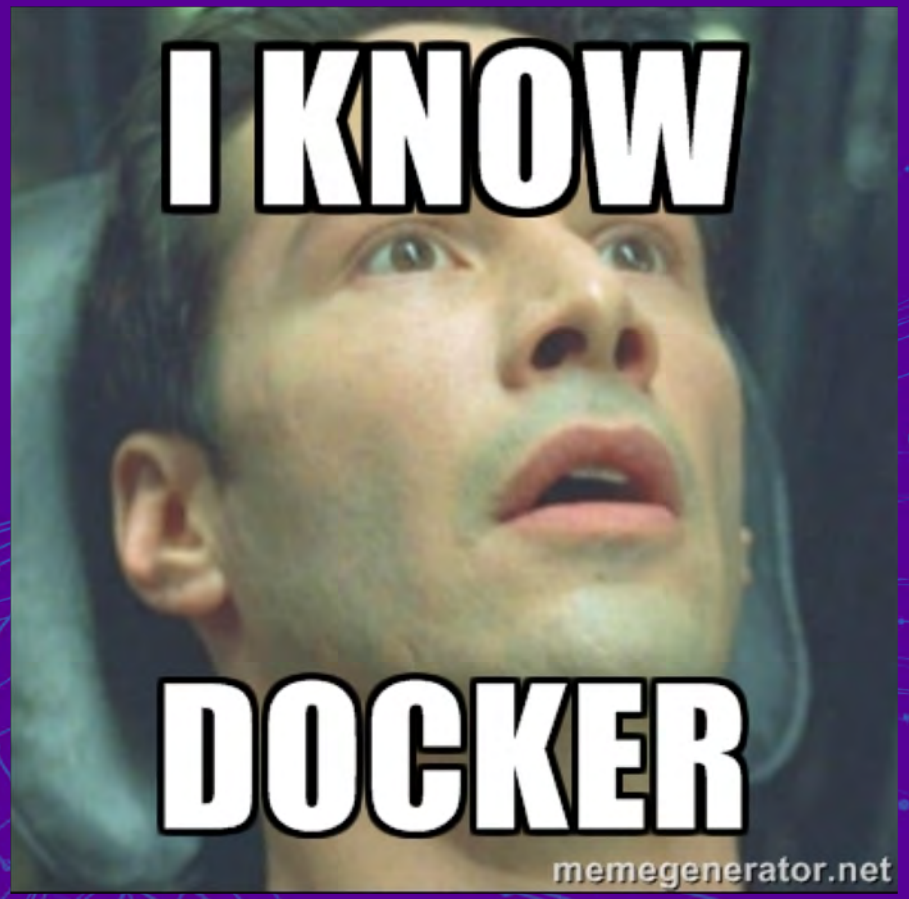
This Is A Journey

1. Unleash your inner curmudgeon - To Hell With Docker
2. Okay, DevOps is on to something, but I'm still skeptical
3. Why is there no good documentation?
4. Are people really using this in production?
5. Docker files are pretty neat
6. Holy crap Docker compose is awesome
7. I will never deploy containers unless I control everything
8. Okay, let's use EC2
9. Ya know what, screw it, let's go completely serverless
10. We need as few containers as possible
11. We don't have enough containers
12. I will never use Kubernetes
13. Crap, why didn't I use Kubernetes sooner?
14. We are releasing 10 new versions a day, and each one is getting tested and deployed automatically

What To Expect From This Talk

- Learn the appropriate uses for Docker implementations and understand the pros and cons of Docker technology.
- See examples of a traditionally deployed application along its journey to full containerization. This will allow you to learn and apply these concepts to your own software projects.
- Review forensic data from a Docker security breach. You will be able to identify the characteristics of these attacks and spot them more easily in your environment(s).
- Run through a Docker security checklist, allowing you to make better security recommendations to your development teams.

Why Docker?



Why Docker?

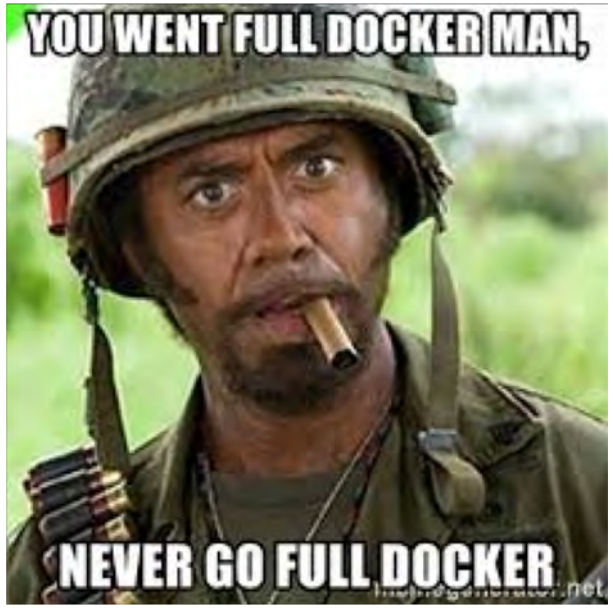
- There is a free version, low barrier to entry (cost-wise)
- There is a big learning curve, I like to learn new things (you should too!)
- VMs are too large, require too much maintenance and are more of a commitment than a container

The Truly Original Fidget Spinners

We want to be cool and fidget spinners are so 2016...

Why Docker? (2)

1. You can combine development, sysadmin and security all in one process
2. Reproducibility and ease of deployment means we can have more environments for specific purposes (like security testing)



Why Docker? (2)

"Containers are immutable."

- An attacker may get shell, but end up stuck inside a container with little functionality
- If you get hacked, you can deploy a new set of containers, and, well, not be hacked for a bit
- OS, application and library updates occur in the natural, more frequently occurring, release cycle
- More easily strip out components you don't need

When I change a line of my code and my docker container dies.

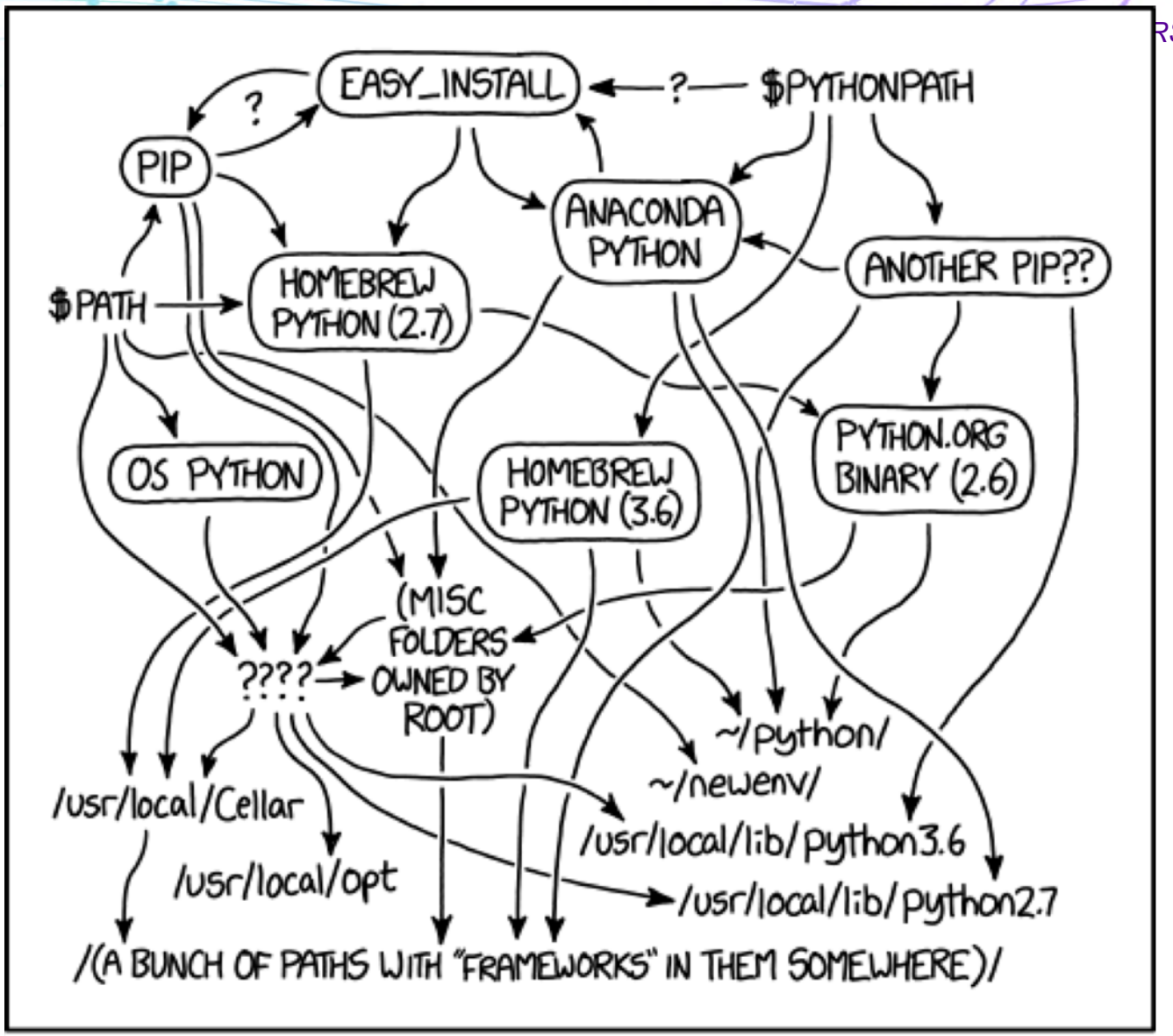
My Initial Reason

Python environments….

pip

virtualenv

And the resulting mess



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://imgs.xkcd.com/comics/python_environment.png

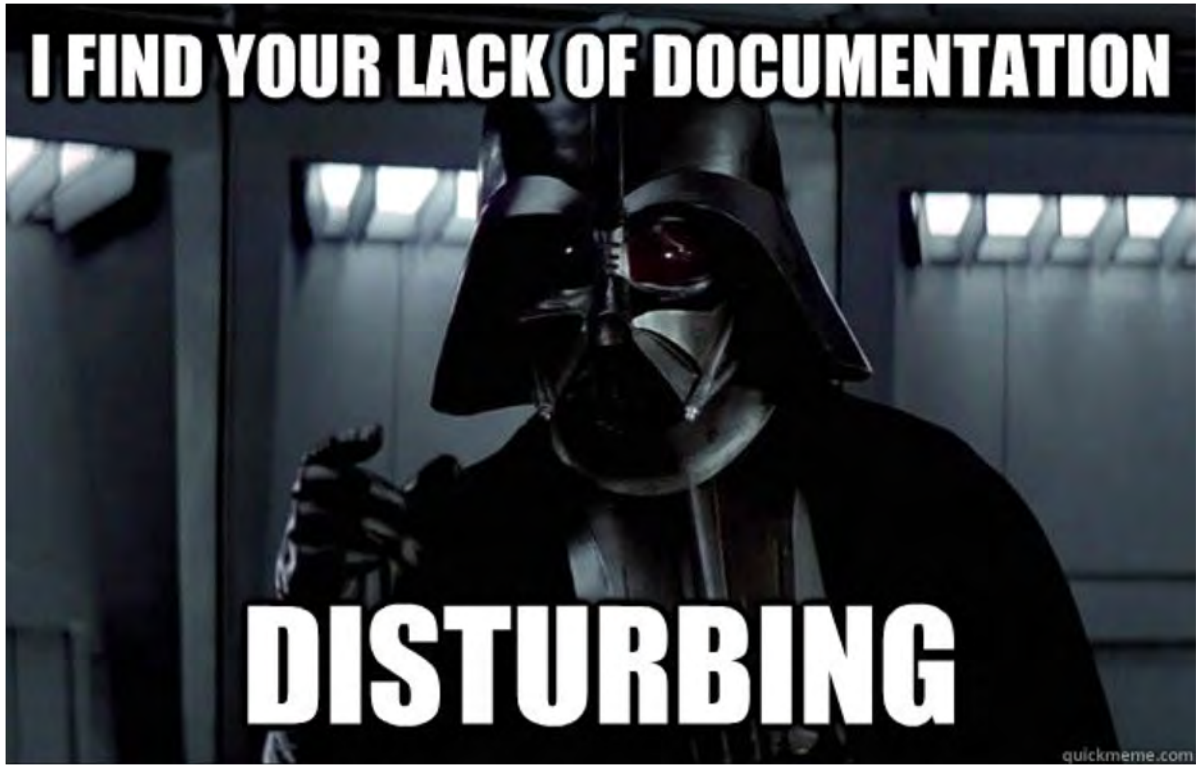
RSAConference2019

Why Not Docker?

Why Not Docker?

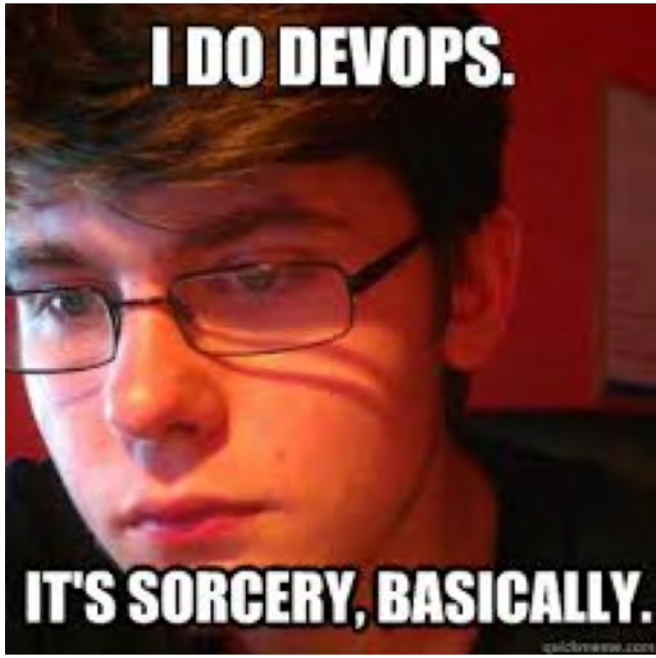
Docker is still pretty new, which means:

- Documentation is pretty rough, not many examples (Sometimes the documentation is someone's blog entry from 2 years ago)
- Most are just "testing it out" and not running in production
- There is a big learning curve



Why Not Docker? (2)

- Security is still largely untested (thankfully I've done some of that already)
- Multi-Platform applications - Containers share the host OS, so may not be portable
- Tool Fatigue - Jenkins, Git, Swarm, Compose, Kubrenetes, Open Shift, CoreOS, Puppet, Chef, Ansible,



RSAConference2019

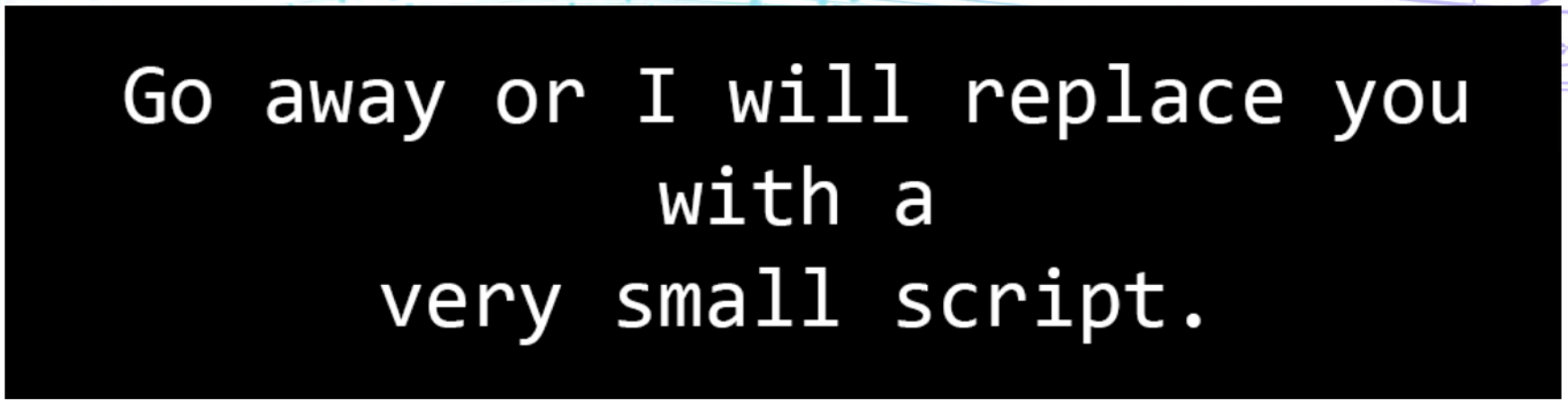
Example: Our Internal Podcasting App

Despite All That I “Dockerized” My Project

- Enter **PPWorks** - Podcast Post-Production Workflow System
- Yes, I named it.And yes, the developers hate the name.
- I still make jokes about the name. The developers do not think its funny.
- There is no logo, and our beta designs are, well, you might not think its funny...

A Little About My Project

- We produce 16 segments every week here at Security Weekly
- Each segment gets pushed to podcast catchers, social media and blogs
- Each segment has at least 3 different formats (Audio, HD video, SD video)
- **Manually uploading and describing all content would require 1 FTE, whose sole (or soul) job would be only this.**



- It used to take the same amount of time to publish as it did to record (e.g. a 2 hour show would take 2 hours to publish)
- We needed a script/application that could automate and cut that time down
- Today we publish in a fraction of the time, about 15 minutes per show to actually publish
- The point: **Our project started out as a script that could automate publishing**

The Beginning

We chose Python Flask to create a web interface, upload HD video, convert it, add your metadata and publish to all destinations

It ran on hardware + Linux + Python+UWSGI

Enter Docker

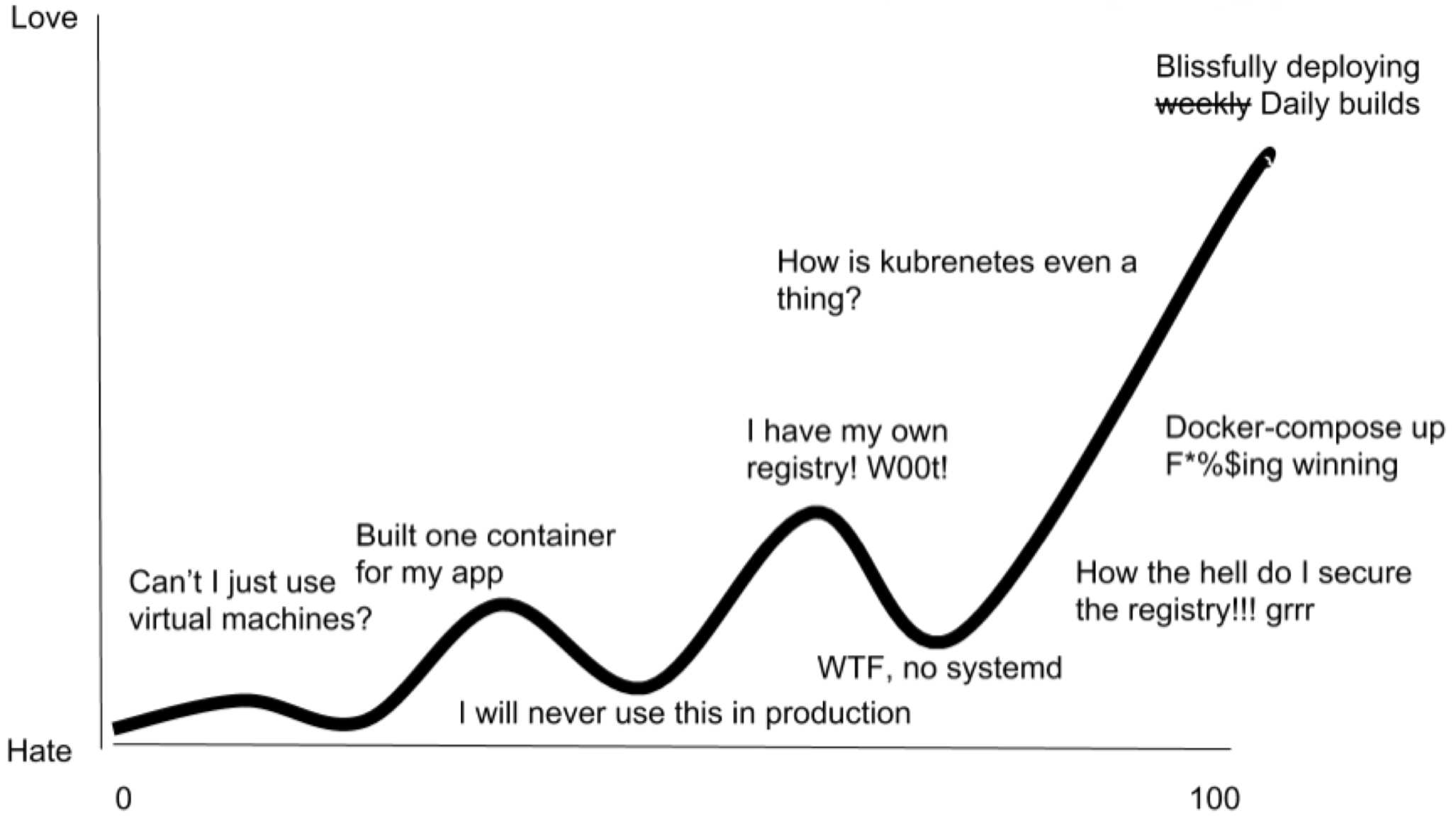
I chose Docker to “containerize” the environment

I learned its really hard to take an application that already exists and implement it in Docker

Defining the environment took time, and really was not complete until several months later

Goals: **Have a repeatable environment that could be replicated to create a Dev and Prod environment**

The Docker Journey - From Hate to Love

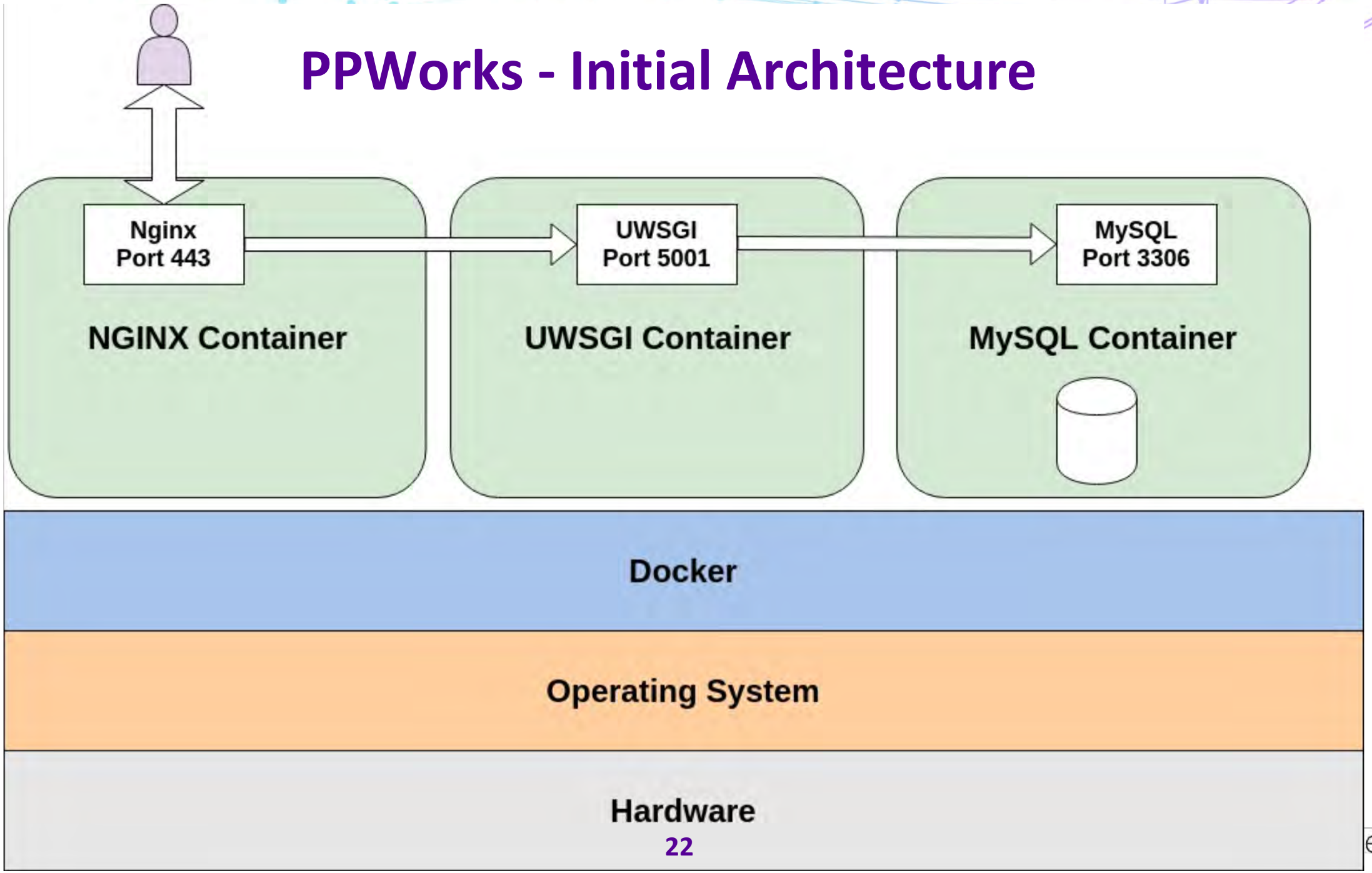


At This Stage Consider A Mindset Change

1. **Containers are immutable** - They get destroyed and re-created often
2. Keep your data outside the container
3. Keep all components in separate containers (Reduces the attack surface!)
4. Don’t image containers, have a repeatable process to build and deploy

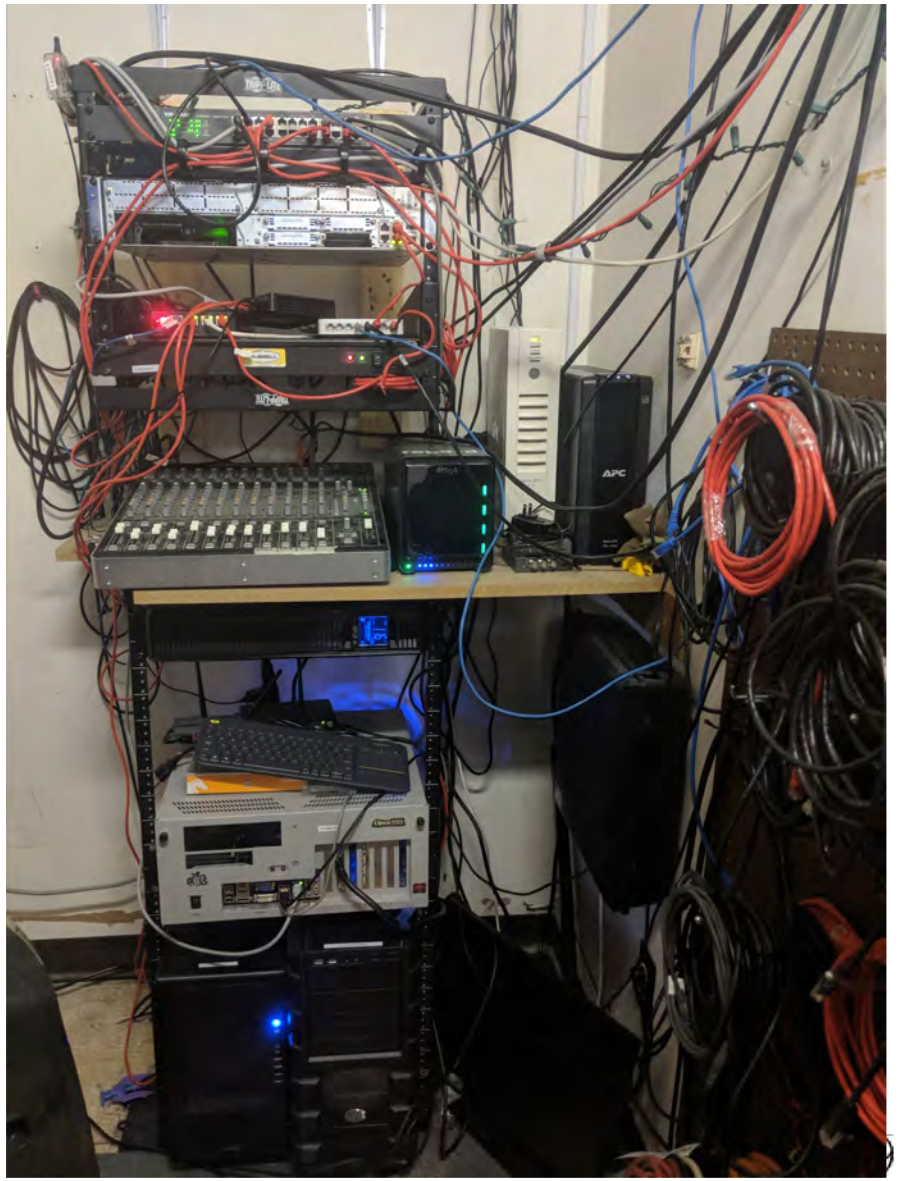
https://developers.redhat.com/blog/2016/02/24/10-things-to-avoid-in-docker-containers/

PPWorks - Initial Architecture



Two Things Pushed Us To The Cloud

1) Ubuntu upgrades and releases were not stable
2) The development server hard drive crashed



Initially we rejected cloud architecture because we are moving multiple large files (3GB) daily. Once to upload to PPWorks and once more to upload to services like YouTube.

Ubuntu Upgrades

System76 came with Ubuntu by default

We ran this in production, until upgrades started really breaking things

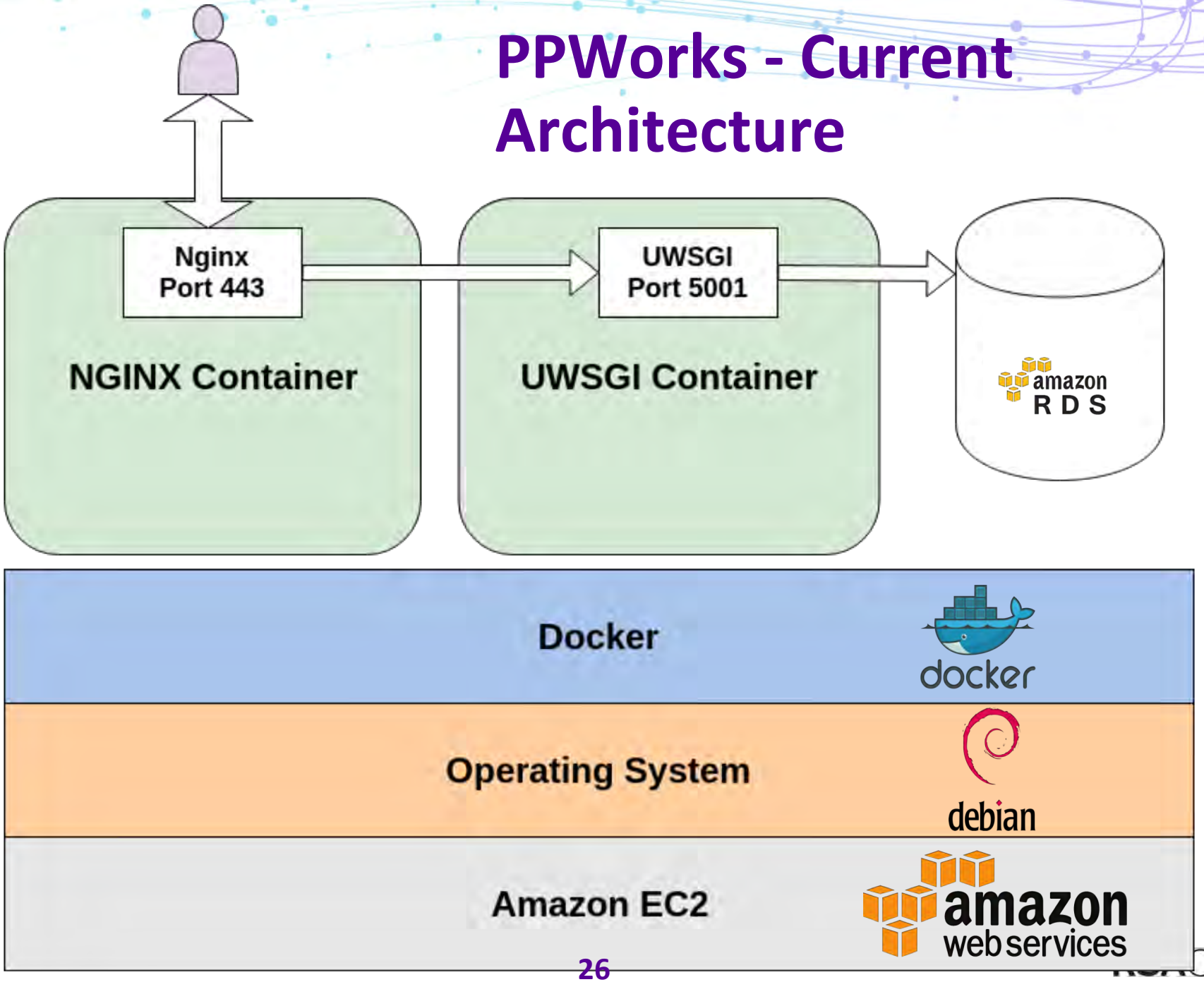
No more Ubuntu on servers (in fact, System76 now has their own Linux Distro called PopOS!)

The One Spinning Drive In The Building

Well, it failed. Which also means so did the development server.

Losses: Time to rebuilding the server, OS and all software (we had backups). Money to purchase new hard drive. Development time that could have been spend on new features.

PPWorks - Current Architecture

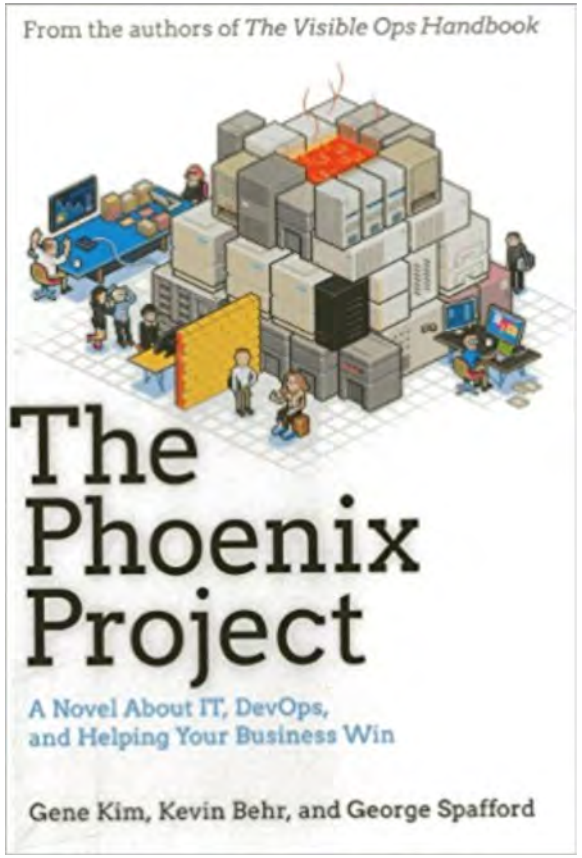


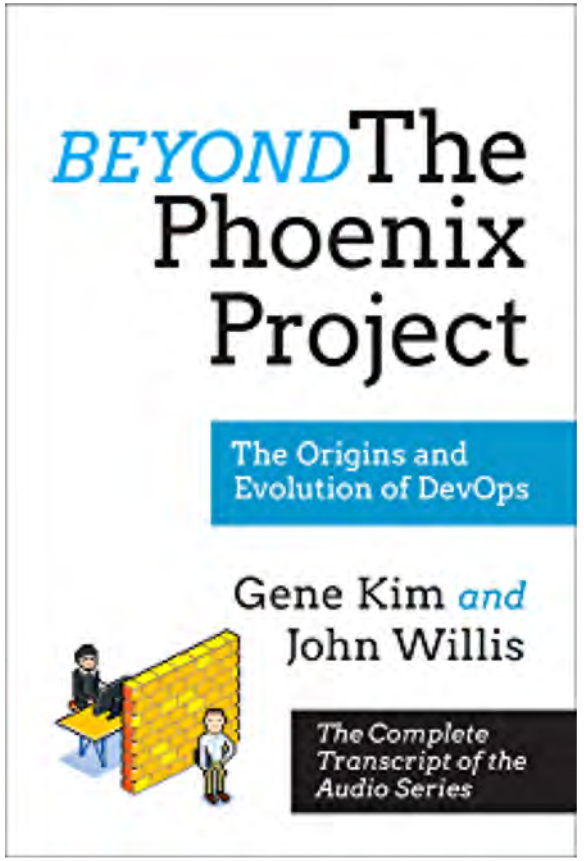
Then I Read The Phoenix Project...

I love Bill

Eric grew on me

I still hate Sarah





Big Takeaways

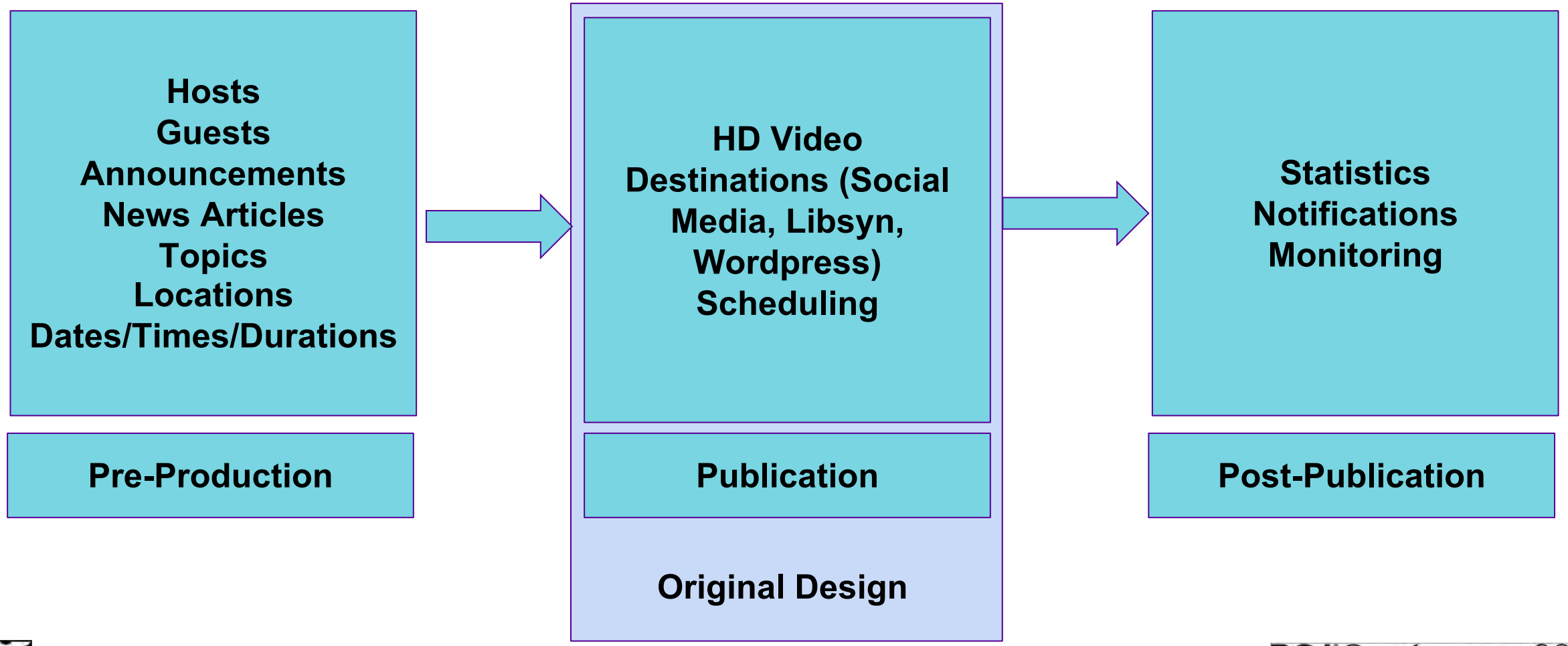
Unifying development, IT operations and security has the potential to solve problems

A continuous and rapid feedback loop is critical to success

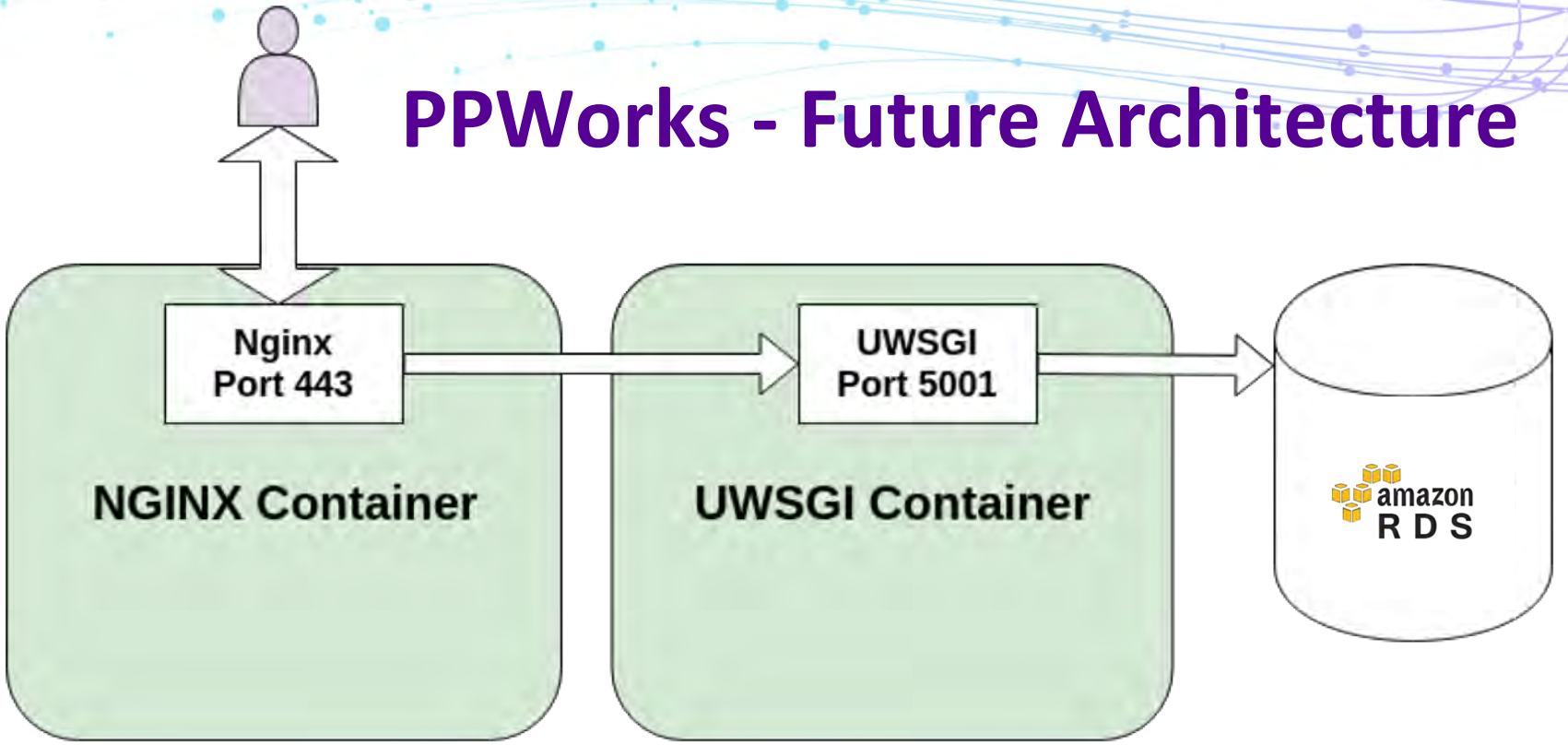
Understanding your constraints, while not always straightforward, is important

Have everything you require before you start a process or push a job through the system

PPWorks Next Major Release - Coming Soon!

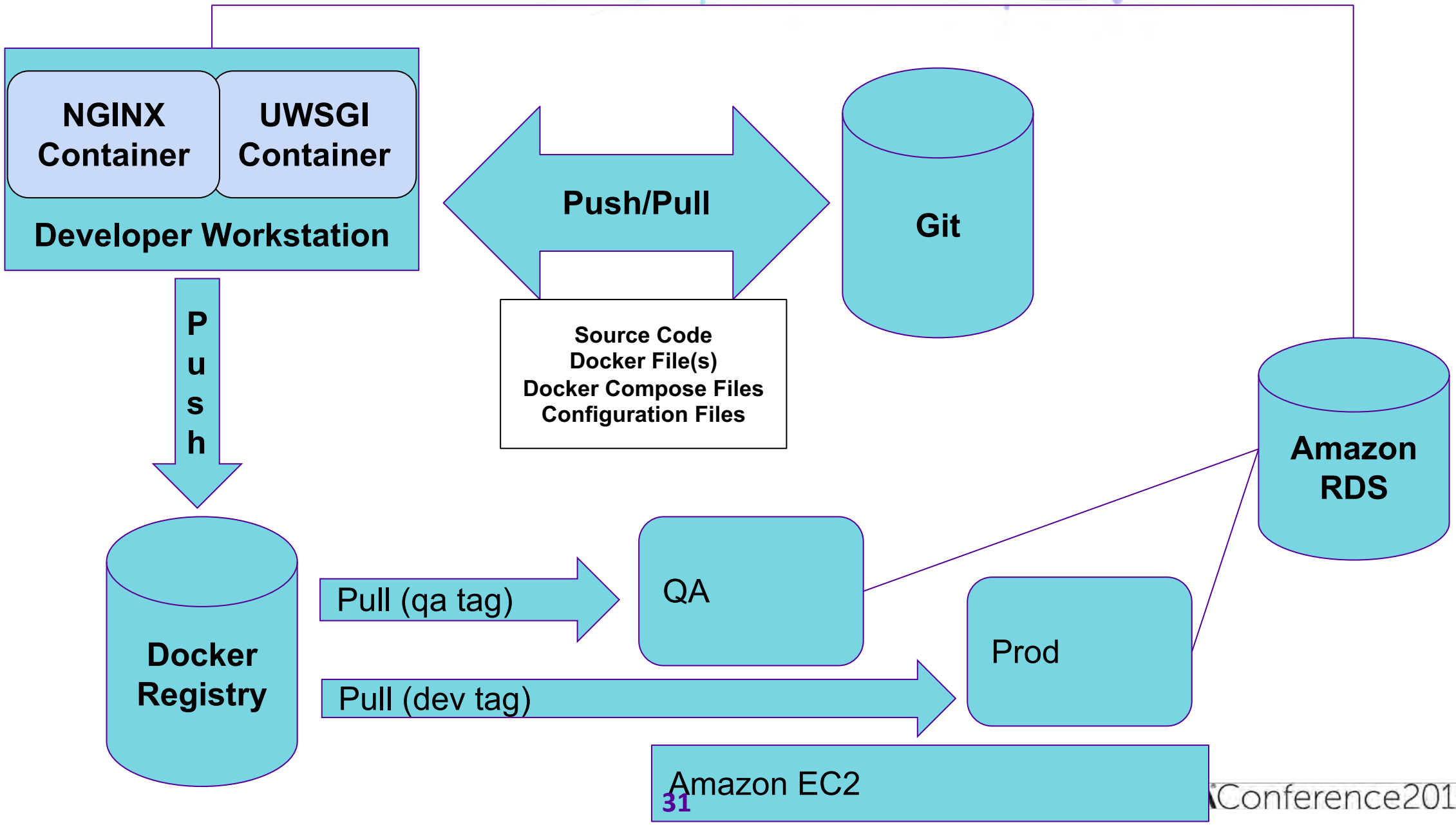


PPWorks - Future Architecture

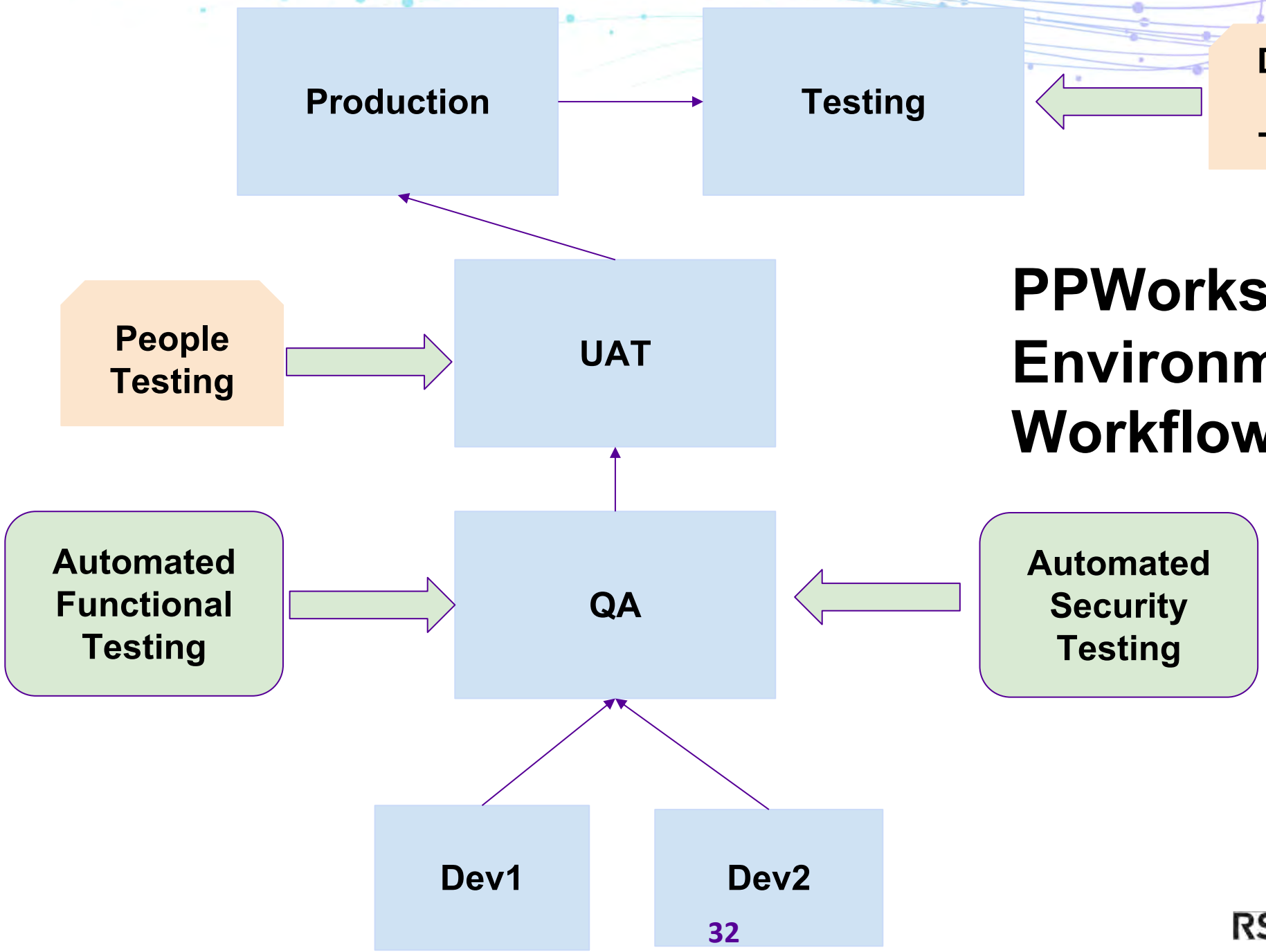


AWS Fargate

The Development Process



PPWorks - Environments and Workflow



RSAConference2019

Docker Security - Lessoned Learned

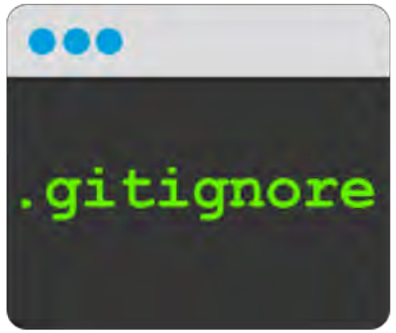
“Network” Security

```
dev_ppworks:
    build: ./ppworks_docker
    image: "registry:5000/dev_ppworks:${TAG}"
    hostname: dev_ppworks
    ports:
      - "8001:8001"
      - "9996:9996"
    links:
      - "dev_ppworks_mysql:mysql"
    volumes:
      - /storage/docker/dev/uploads:/uploads
      - /storage/docker/dev/conf:/ppconf
      - /Users/paulda/docker/ppworks/code:/ppworks
    networks:
      default:
      dev_ppworks_nw:
          ipv4_address: 10.11.1.20
```

Data “Storage”

```
dev_ppworks:
    build: ./ppworks_docker
    image: "registry:5000/dev_ppworks:${TAG}"
    hostname: dev_ppworks
    ports:
      - "8001:8001"
      - "9996:9996"
    links:
      - "dev_ppworks_mysql:mysql"
    volumes:
      - /storage/docker/dev/uploads:/uploads
      - /storage/docker/dev/conf:/ppconf
      - /Users/paulda/docker/ppworks/code:/ppworks
    networks:
      default:
      dev_ppworks_nw:
          ipv4_address: 10.11.1.20
```

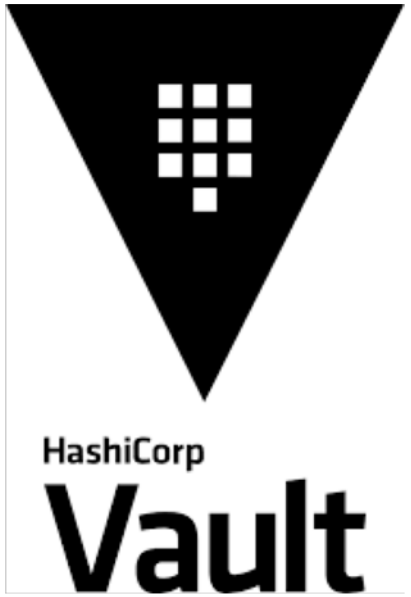
Managing Secrets



Secrets can be: SQL passwords, API keys, app credentials for automated testing

More automation = more stored secrets

More secrets = increased likelihood they are leaked on Github



The Security Incident...

April 1, 2018 - Extreme amounts of traffic leaving our AWS instance

Rogue Docker container was uploaded to one of our Docker servers

In our rush to the cloud, security controls were not in place that should have been!

I Didn’t Know What I Was Doing

syslog:Apr 2 13:31:32 ip-172***** dockerd[14406]: time="2018-04-02T13:31:32.186864242Z" level=error msg="Error setting up exec command in container gogo: No such container: gogo"

syslog:Apr 2 14:14:13 localhost dockerd[539]: time="2018-04-02T14:14:13.538558381Z" level=warning msg="[!] **DON'T BIND ON ANY IP ADDRESS WITHOUT setting --tlsverify IF YOU DON'T KNOW WHAT YOU'RE DOING [!]**"

Uploaded Via The Docker API

{"Repositories":{"19144885/gogo":{"19144885/gogo:latest":"sha256:e9789c17899ef6734f508d513c3f96f3f63eaf41772d05e8b0029fa9141b7a76","19144885/gogo@sha256:c40df2d8de9ac5919b4169cd87c7b4325c097c108fb743957a069248a8445df1":"sha256:e9789c17899ef6734f508d513c3f96f3f63eaf41772d05e8b0029fa9141b7a76"}

https://blog.aquasec.com/cryptocurrency-miners-abusing-containers-anatomy-of-an-attempted-attack

The Security Incident… (2)

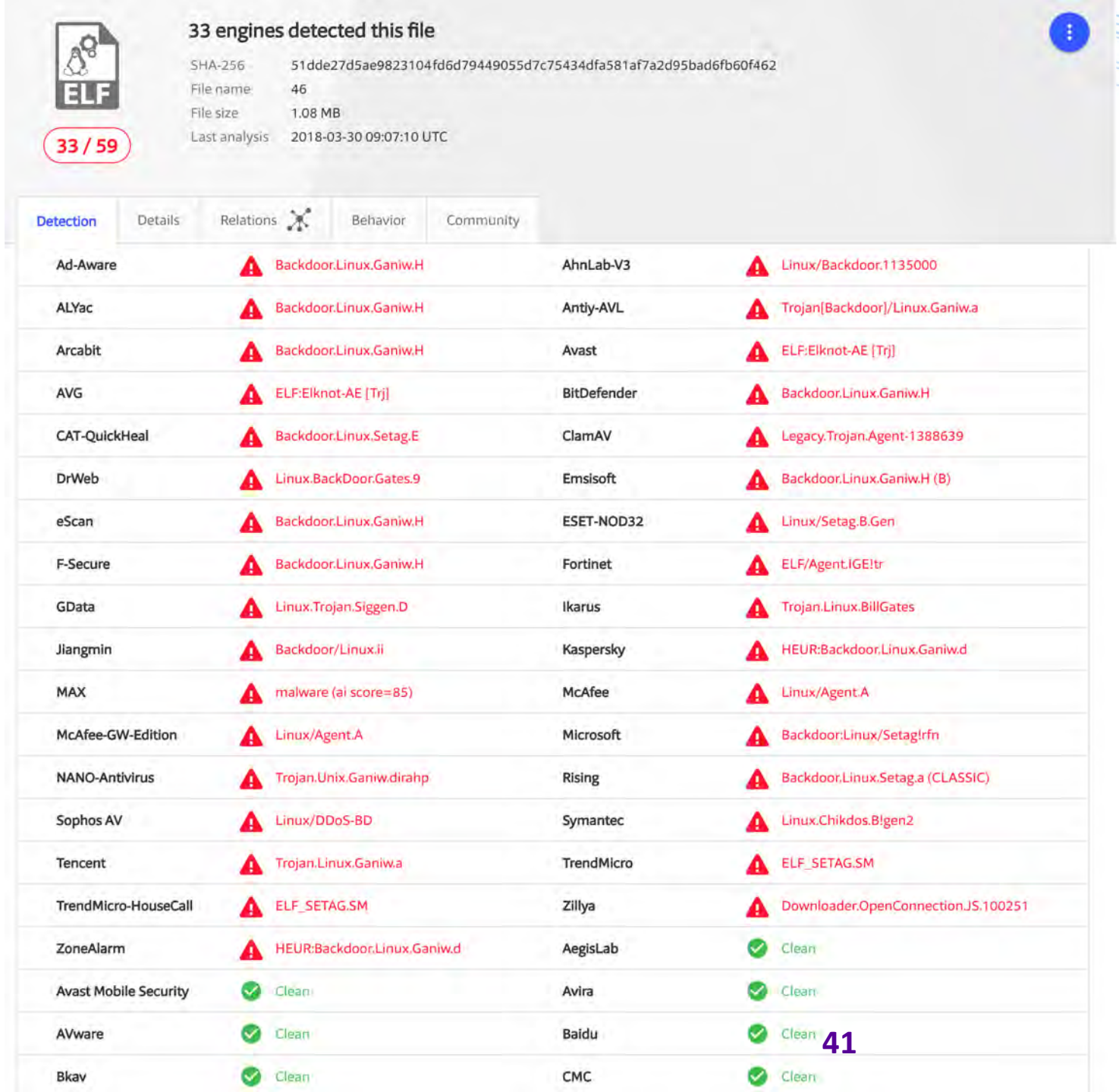
Thankfully, it was not a sophisticated attack

The files inside the container were named “java” and “pythno”

The “java” binary was only 1.1 MB

I caught the “pythno” switching of letters (duh)

Then I uploaded it to Virus Total...



33 engines detected this file

SHA-256 51dde27d5ae9823104fd6d79449055d7c75434dfa581af7a2d95bad6fb60f462
File name 46
File size 1.08 MB
Last analysis 2018-03-30 09:07:10 UTC

33 / 59

Detection | Details | Relations | Behavior | Community

Ad-Aware	Backdoor.Linux.Ganiw.H	AhnLab-V3	Linux/Backdoor.1135000
ALYac	Backdoor.Linux.Ganiw.H	Antiy-AVL	Trojan[Backdoor]/Linux.Ganiw.a
Arcabit	Backdoor.Linux.Ganiw.H	Avast	ELF:Elknot-AE [Trj]
AVG	ELF:Elknot-AE [Trj]	BitDefender	Backdoor.Linux.Ganiw.H
CAT-QuickHeal	Backdoor.Linux.Setag.E	ClamAV	Legacy.Trojan.Agent-1388639
DrWeb	Linux.BackDoor.Gates.9	Emsisoft	Backdoor.Linux.Ganiw.H (B)
eScan	Backdoor.Linux.Ganiw.H	ESET-NOD32	Linux/Setag.B.Gen
F-Secure	Backdoor.Linux.Ganiw.H	Fortinet	ELF/Agent.IGE!tr
GData	Linux.Trojan.Siggen.D	Ikarus	Trojan.Linux.BillGates
Jiangmin	Backdoor/Linux.ii	Kaspersky	HEUR:Backdoor.Linux.Ganiw.d
MAX	malware (ai score=85)	McAfee	Linux/Agent.A
McAfee-GW-Edition	Linux/Agent.A	Microsoft	Backdoor:Linux/Setag!rfn
NANO-Antivirus	Trojan.Unix.Ganiw.dirahp	Rising	Backdoor.Linux.Setag.a (CLASSIC)
Sophos AV	Linux/DDoS-BD	Symantec	Linux.Chikdos.B!gen2
Tencent	Trojan.Linux.Ganiw.a	TrendMicro	ELF_SETAG.SM
TrendMicro-HouseCall	ELF_SETAG.SM	Zillya	Downloader.OpenConnection.JS.100251
ZoneAlarm	HEUR:Backdoor.Linux.Ganiw.d	AegisLab	Clean
Avast Mobile Security	Clean	Avira	Clean
AVware	Clean	Baidu	Clean
Bkav	Clean	CMC	Clean

Sophisticated malware it was not...

Process And Service Actions

Processes Created

/tmp/EB93A6/996E.elf
sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc1.d/S97VsystemsshMdt
sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc2.d/S97VsystemsshMdt
sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc3.d/S97VsystemsshMdt
sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc4.d/S97VsystemsshMdt
sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc5.d/S97VsystemsshMdt

Shell Commands

sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc1.d/S97VsystemsshMdt
sh
sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc2.d/S97VsystemsshMdt
sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc3.d/S97VsystemsshMdt
sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc4.d/S97VsystemsshMdt
sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc5.d/S97VsystemsshMdt

Processes Terminated

/tmp/EB93A6/996E.elf
/lib/systemd/systemd-udevd --daemon
ln -s /etc/init.d/VsystemsshMdt /etc/rc1.d/S97VsystemsshMdt
sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc1.d/S97VsystemsshMdt
ln -s /etc/init.d/VsystemsshMdt /etc/rc2.d/S97VsystemsshMdt
sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc2.d/S97VsystemsshMdt
ln -s /etc/init.d/VsystemsshMdt /etc/rc3.d/S97VsystemsshMdt
sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc3.d/S97VsystemsshMdt
ln -s /etc/init.d/VsystemsshMdt /etc/rc4.d/S97VsystemsshMdt
sh -c ln -s /etc/init.d/VsystemsshMdt /etc/rc4.d/S97VsystemsshMdt

Highlighted Actions

Highlighted Text

11CAttackBase
13CPacketAttack
10CAttackUdp
10CAttackSyn
11CAttackIcmp
10CAttackDns
10CAttackAmp
10CAttackPrx

What is the intention of the malware and the user?

Jenkins Uses The Docker API

https://docs.docker.com/engine/reference/commandline/dockerd/#examples

```
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target docker.socket firewalld.service
Wants=network-online.target
Requires=docker.socket

[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:2375
ExecReload=/bin/kill -s HUP $MAINPID
LimitNOFILE=1048576
```

https://docs.docker.com/engine/security/https/#create-a-ca-server-and-client-keys-with-openssl

RSAConference2019

Docker Security Recommendations

Scan Your Containers

Several commercial and open-source tools available

Consistently scan, send the results to the developers

You decide what holds up a build or release

Docker provides scanning in the public (or private) registry services

Goal: Discover outdated OS updates, applications, code libraries install via the OS, general exposures and mis-configuration

Tip: Don't forget to flag add-ons, plugins or libraries that have been deprecated or abandoned. These have been the source of recent attacks.

Ref: https://www.bleepingcomputer.com/news/security/somebody-tried-to-hide-a-backdoor-in-a-popular-javascript-npm-package/

Scan Your Containers (2)

The following resource will get you started:

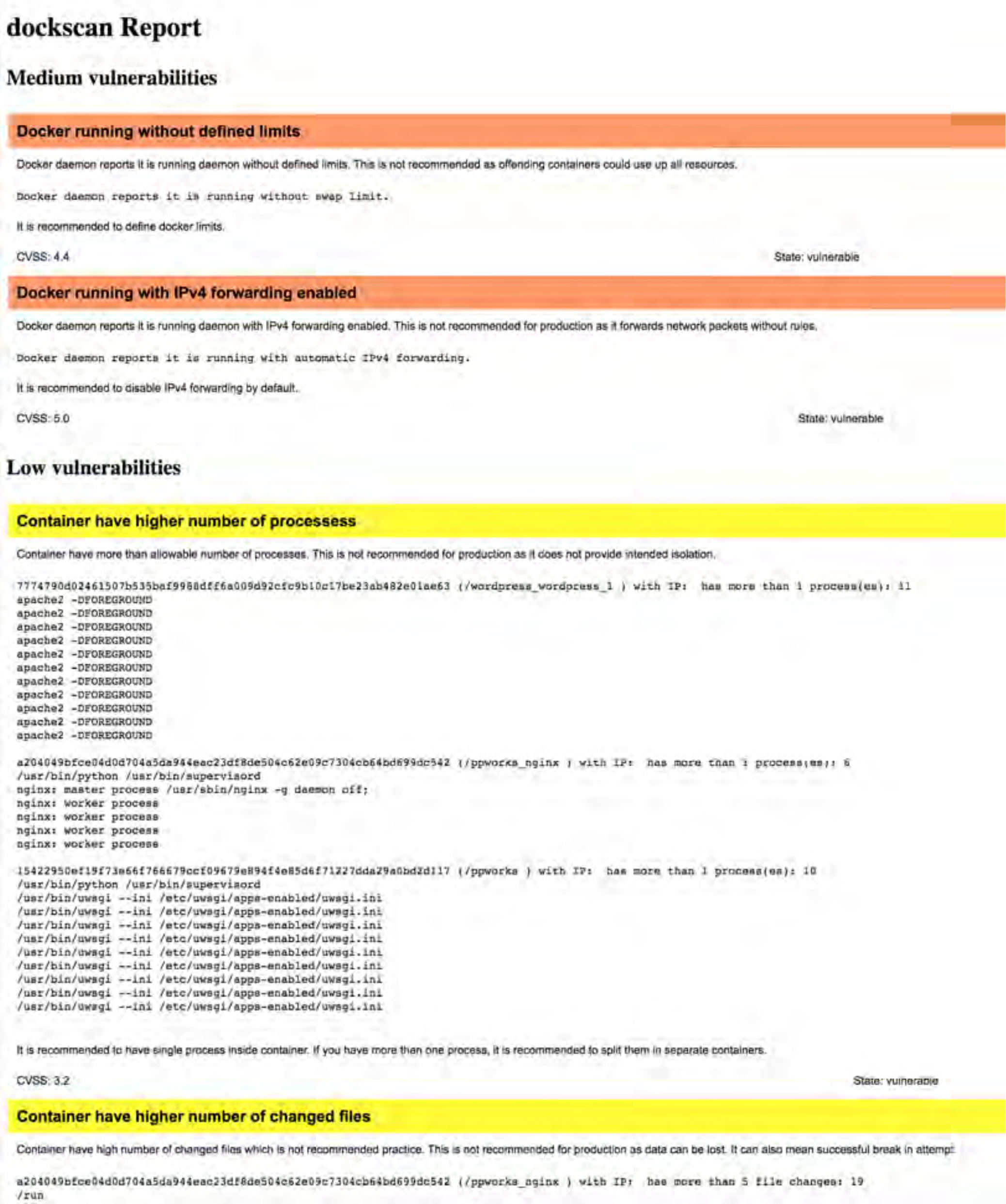
1. https://www.slideshare.net/jlkinsel/a-fun-comparison-of-docker-vulnerability-scanners
2. https://sysdig.com/blog/20-docker-security-tools/
3. https://thenewstack.io/draft-vulnerability-scanners/

Special thanks to Layered Insight (Sponsor), along with John Kinsella and Matt Alderman

Example: Dockscan

```
dockscan -r html -o myreport -v \
tcp://yourhost.com:2375
```

https://github.com/kost/dockscanhttps://github.com/kost/dockscan (Ruby, standalone scanner)



dockscan Report

Medium vulnerabilities

Docker running without defined limits

Docker daemon reports it is running daemon without defined limits. This is not recommended as offending containers could use up all resources.

```
Docker daemon reports it is running without swap limit.
```

It is recommended to define docker limits.

CVSS: 4.4 State: vulnerable

Docker running with IPv4 forwarding enabled

Docker daemon reports it is running daemon with IPv4 forwarding enabled. This is not recommended for production as it forwards network packets without rules.

```
Docker daemon reports it is running with automatic IPv4 forwarding.
```

It is recommended to disable IPv4 forwarding by default.

CVSS: 5.0 State: vulnerable

Low vulnerabilities

Container have higher number of processess

Container have more than allowable number of processes. This is not recommended for production as it does not provide intended isolation.

```
7774790d02461507b535baf9988dff6a009d92cfc9b10c17be23ab482e01ae63 (/wordpress_wordpress_1 ) with IP:  has more than 1 process(es): 11
apache2 -DFOREGROUND
apache2 -DFOREGROUND
apache2 -DFOREGROUND
apache2 -DFOREGROUND
apache2 -DFOREGROUND
apache2 -DFOREGROUND
apache2 -DFOREGROUND
apache2 -DFOREGROUND
apache2 -DFOREGROUND
apache2 -DFOREGROUND
apache2 -DFOREGROUND

a204049bfce04d0d704a5da944eac23df8de504c62e09c7304cb64bd699dc542 (/ppworks_nginx ) with IP:  has more than 1 process(es): 6
/usr/bin/python /usr/bin/supervisord
nginx: master process /usr/sbin/nginx -g daemon off;
nginx: worker process
nginx: worker process
nginx: worker process
nginx: worker process

15422950ef19f73e66f766679ccf09679e894f4e85d6f71227dda29a0bd2d117 (/ppworks ) with IP:  has more than 1 process(es): 10
/usr/bin/python /usr/bin/supervisord
/usr/bin/uwsgi --ini /etc/uwsgi/apps-enabled/uwsgi.ini
/usr/bin/uwsgi --ini /etc/uwsgi/apps-enabled/uwsgi.ini
/usr/bin/uwsgi --ini /etc/uwsgi/apps-enabled/uwsgi.ini
/usr/bin/uwsgi --ini /etc/uwsgi/apps-enabled/uwsgi.ini
/usr/bin/uwsgi --ini /etc/uwsgi/apps-enabled/uwsgi.ini
/usr/bin/uwsgi --ini /etc/uwsgi/apps-enabled/uwsgi.ini
/usr/bin/uwsgi --ini /etc/uwsgi/apps-enabled/uwsgi.ini
/usr/bin/uwsgi --ini /etc/uwsgi/apps-enabled/uwsgi.ini
/usr/bin/uwsgi --ini /etc/uwsgi/apps-enabled/uwsgi.ini
```

It is recommended to have single process inside container. If you have more than one process, it is recommended to split them in separate containers.

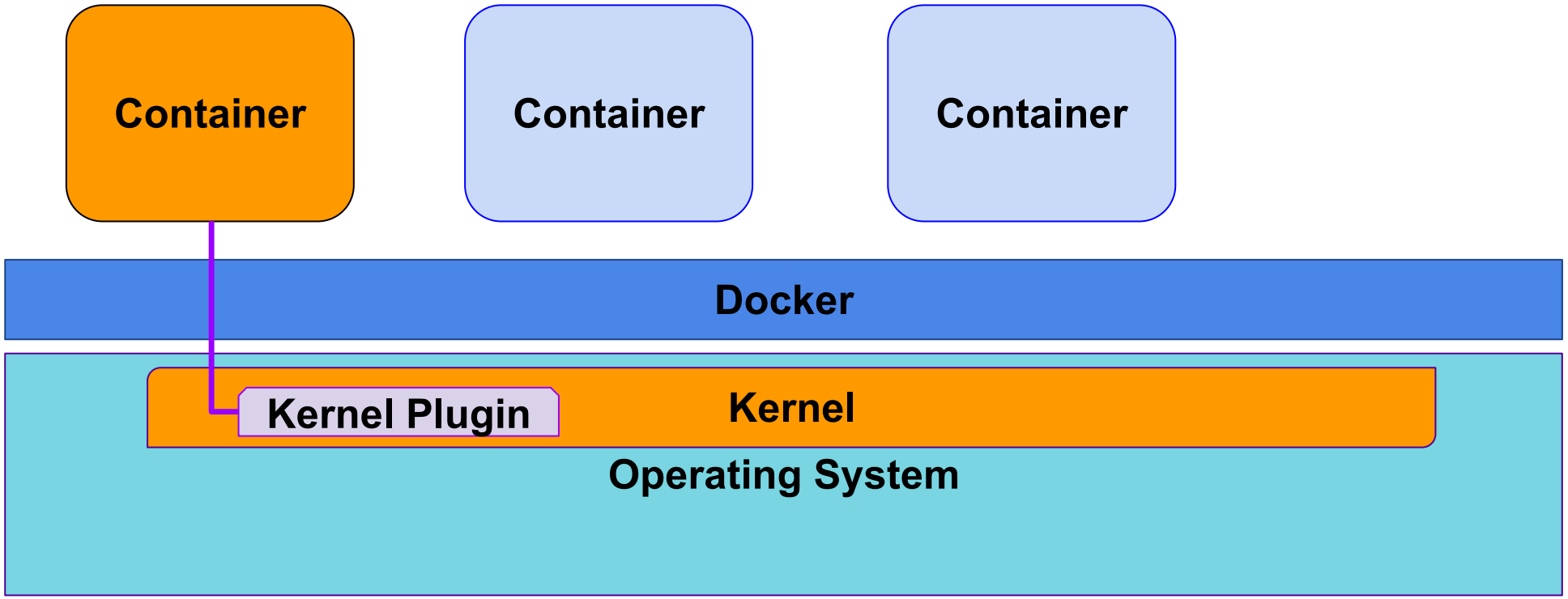
CVSS: 3.2 State: vulnerable

Container have higher number of changed files

Container have high number of changed files which is not recommended practice. This is not recommended for production as data can be lost. It can also mean successful break in attempt

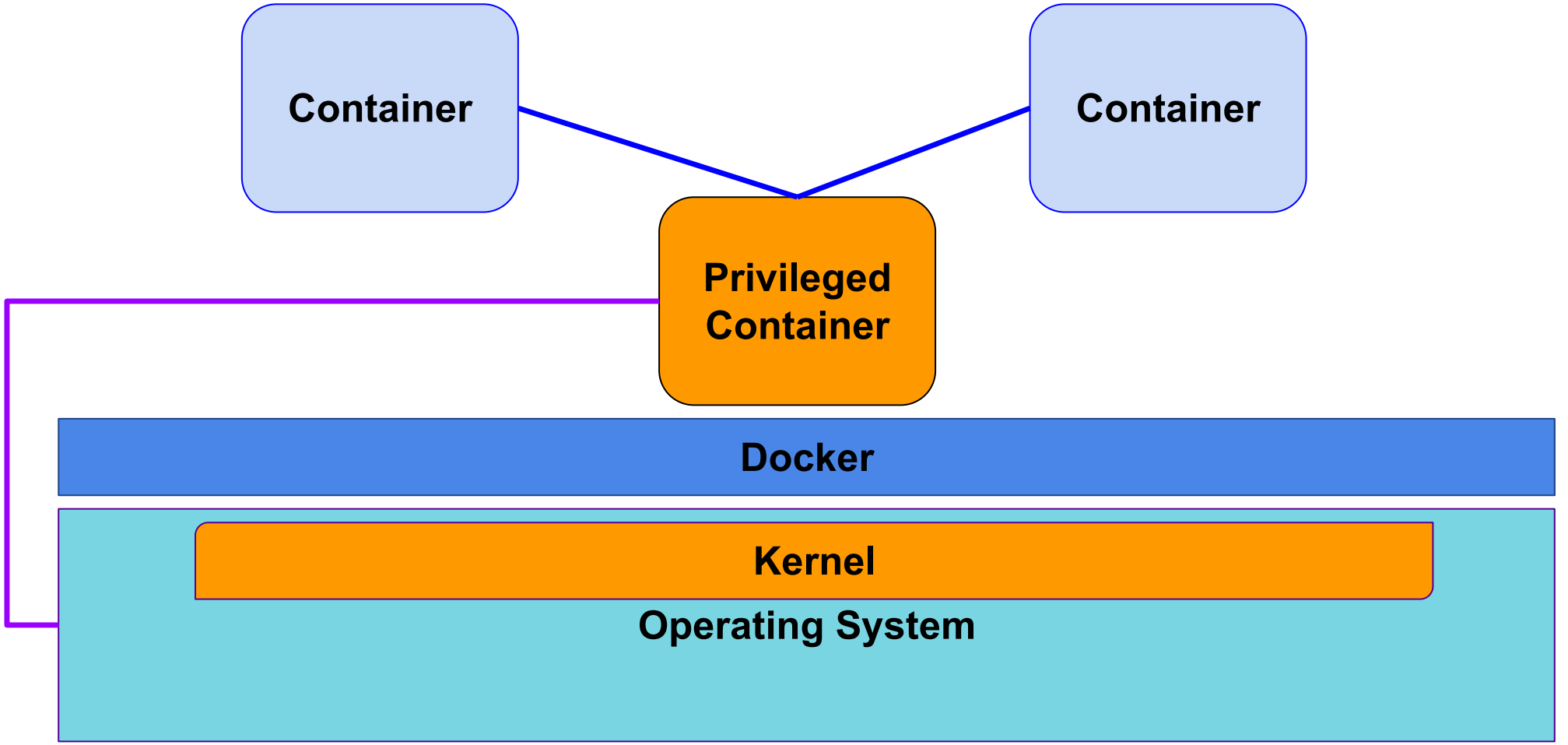
```
a204049bfce04d0d704a5da944eac23df8de504c62e09c7304cb64bd699dc542 (/ppworks_nginx ) with IP:  has more than 5 file changes: 19
/run
```

Kernel Module Protection



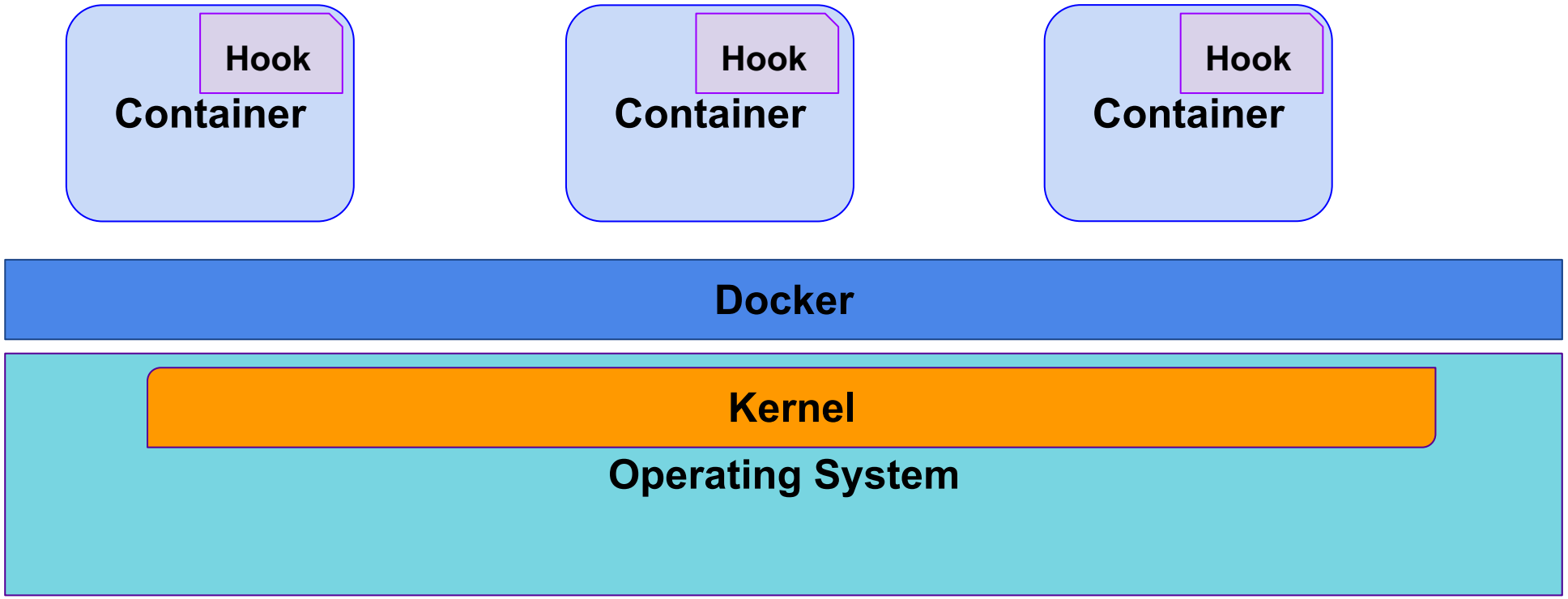
Does not require access to the container service or a hook into the containers.

Privileged Containers



Privileged containers run as root, but still have some limitation as they are not inside the kernel.

Non-Privileged Containers (No Root)



Does not require root and offers good protection and monitoring.

Docker Defensive Recommendations Recap

1. Harden your Docker installations, disable API access
2. Strip out *anything* not required in your Docker containers
3. Scan your Docker containers for vulnerabilities and exposed data
4. Implement a security technology to monitor and respond to container threats
5. Keep all of your secrets in a separate trusted system
6. Consider a serverless architecture for container deployment
7. Work closely with Development and IT Operations to ensure a secure platform for your applications

The End

https://blog.jayway.com/2015/03/21/a-not-very-short-introduction-to-docker/

https://www.humblebundle.com/books/devops-books?partner=indiekings&charity=2030222

Paul Asadoorian, Founder & CTO at Security Weekly
paul@securityweekly.com / @securityweekly