



# DevSecOps in a Cloud Native World

<http://bit.ly/cloudnative-security>

Karthik Gaekwad

@iteration1

Devnet Create 2019

# Hello

- I'm Karthik Gaekwad
- NOT a DBA



- Cloud Native Evangelist at Oracle Cloud
- <https://cloudnative.oracle.com/>
- Past: Developer on the Oracle Managed Kubernetes Team



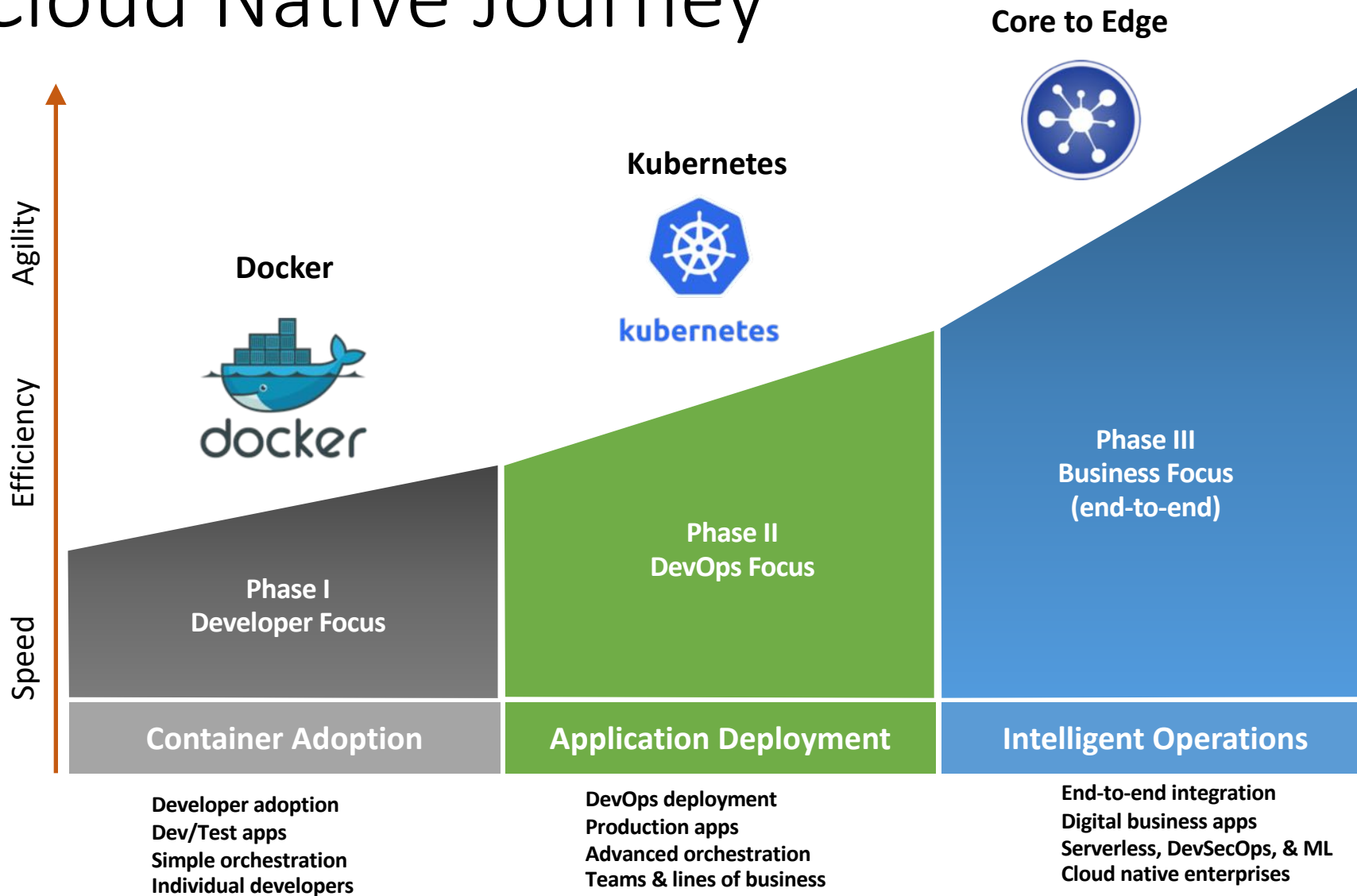
**@iteration1**

# Hello



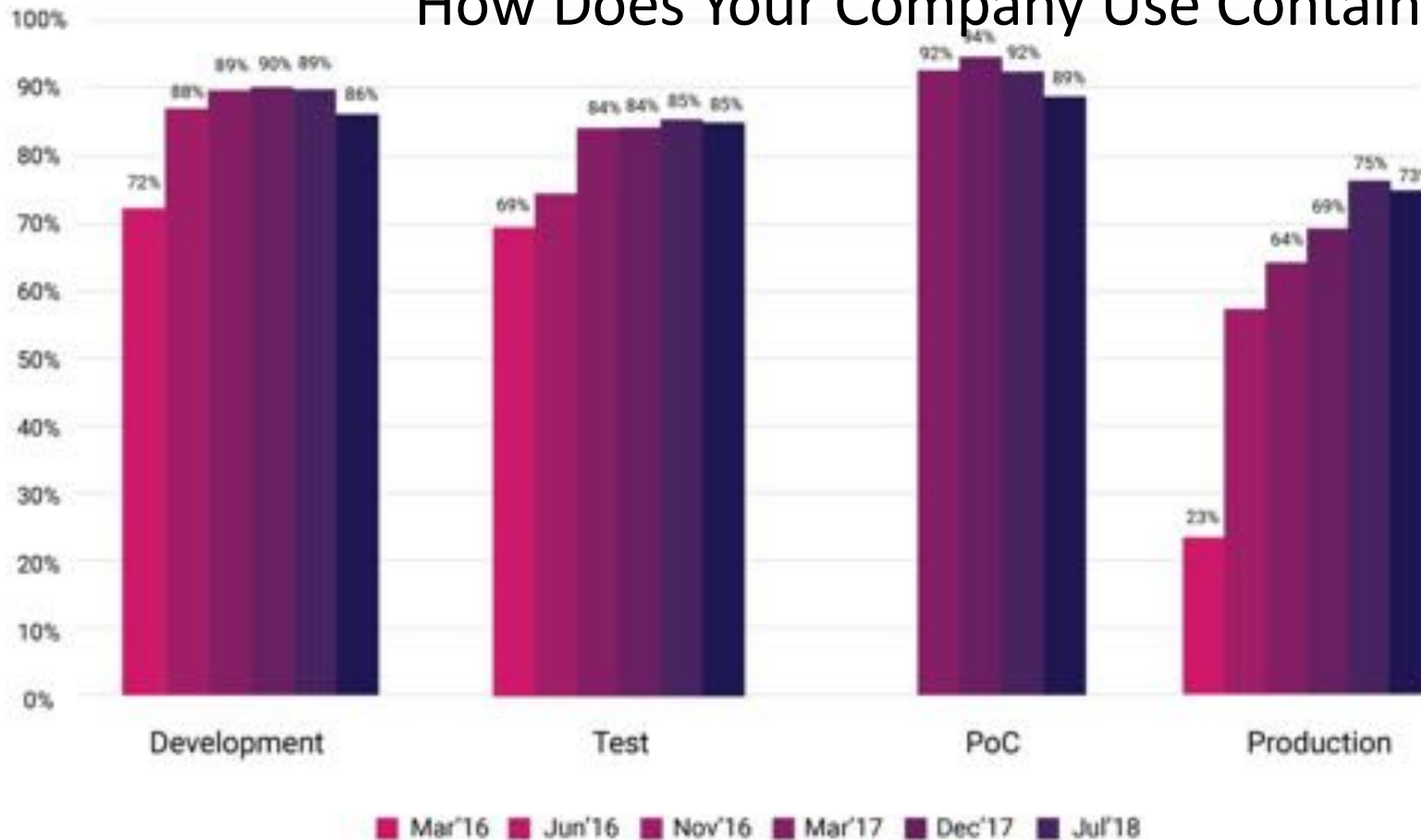
- Been in Industry 15 years.
- In general, I like building stuff with friends.
  - Maintainer for Gauntlt- Open source security scanner.
- Love Teaching and building community.
  - Run Devopsdays Austin, Container Days, Cloud Austin.
  - Chair All Day Devops Cloud Native track.
  - LinkedIn Learning Author for Learning Kubernetes (and more).

# The Cloud Native Journey



# CNCF Survey: August 2018

## How Does Your Company Use Containers and Where?

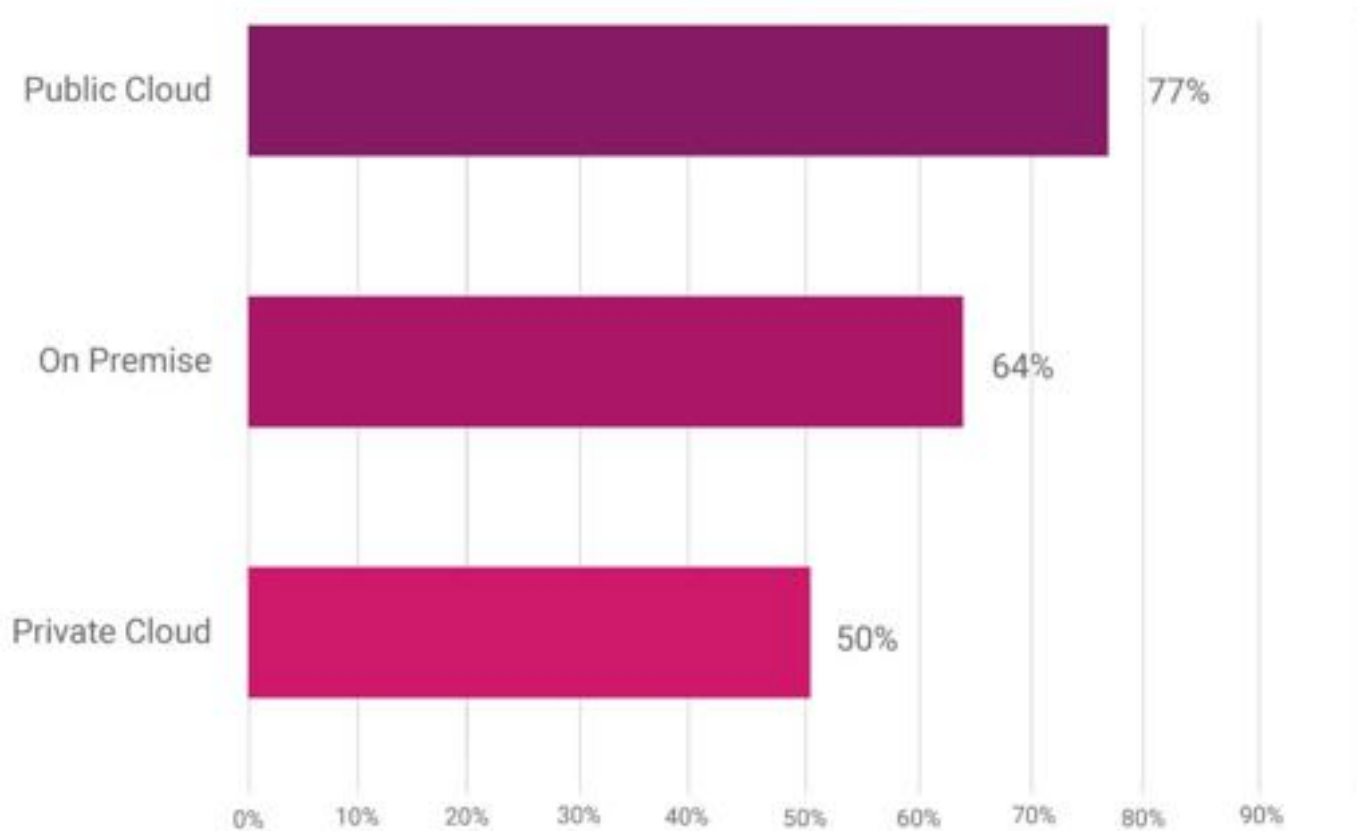


Lots of adoption on dev/staging

Continued production increase

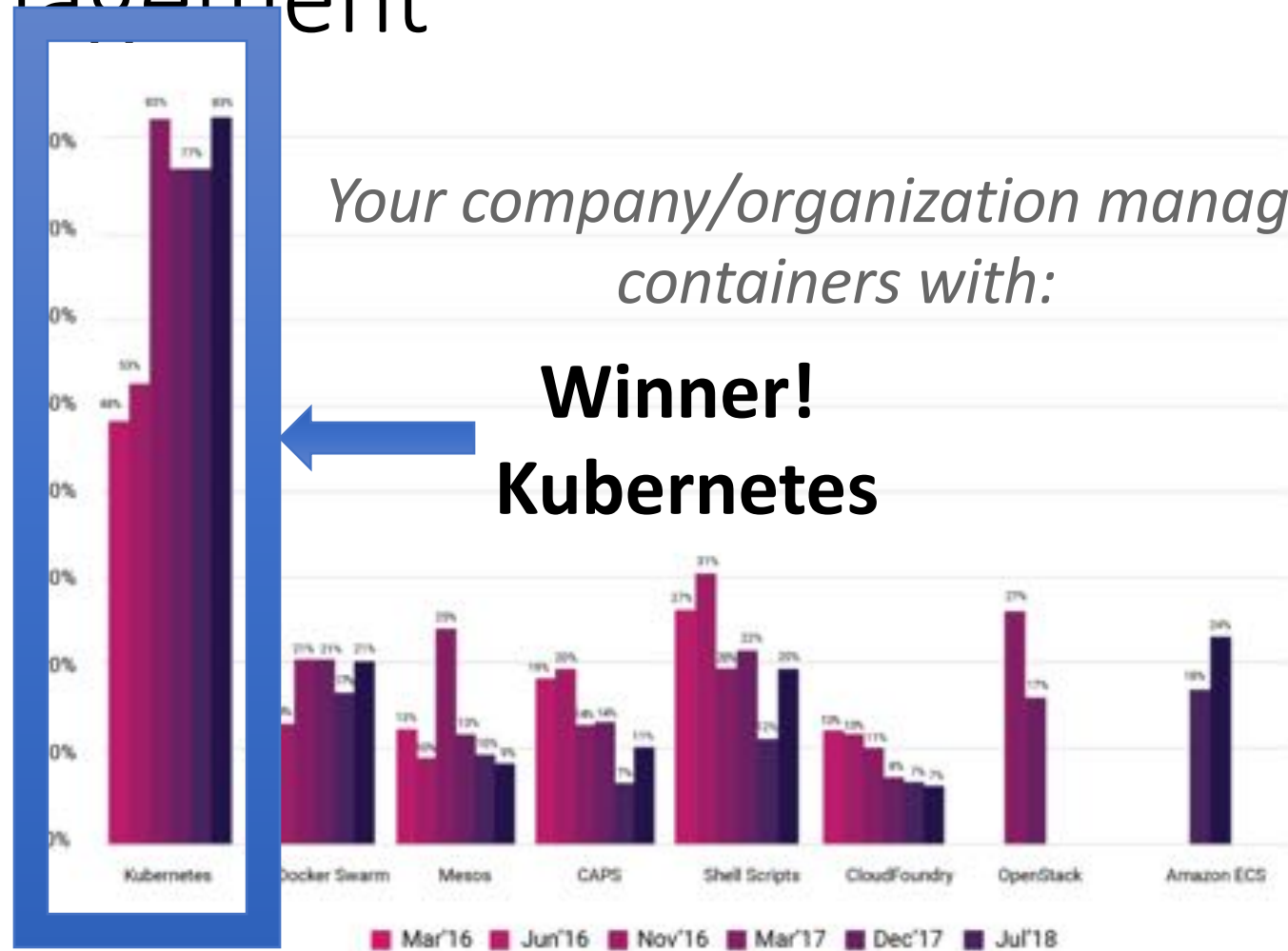
# CNCF Survey: August 2018

How Does Your Company Use Containers and Where?

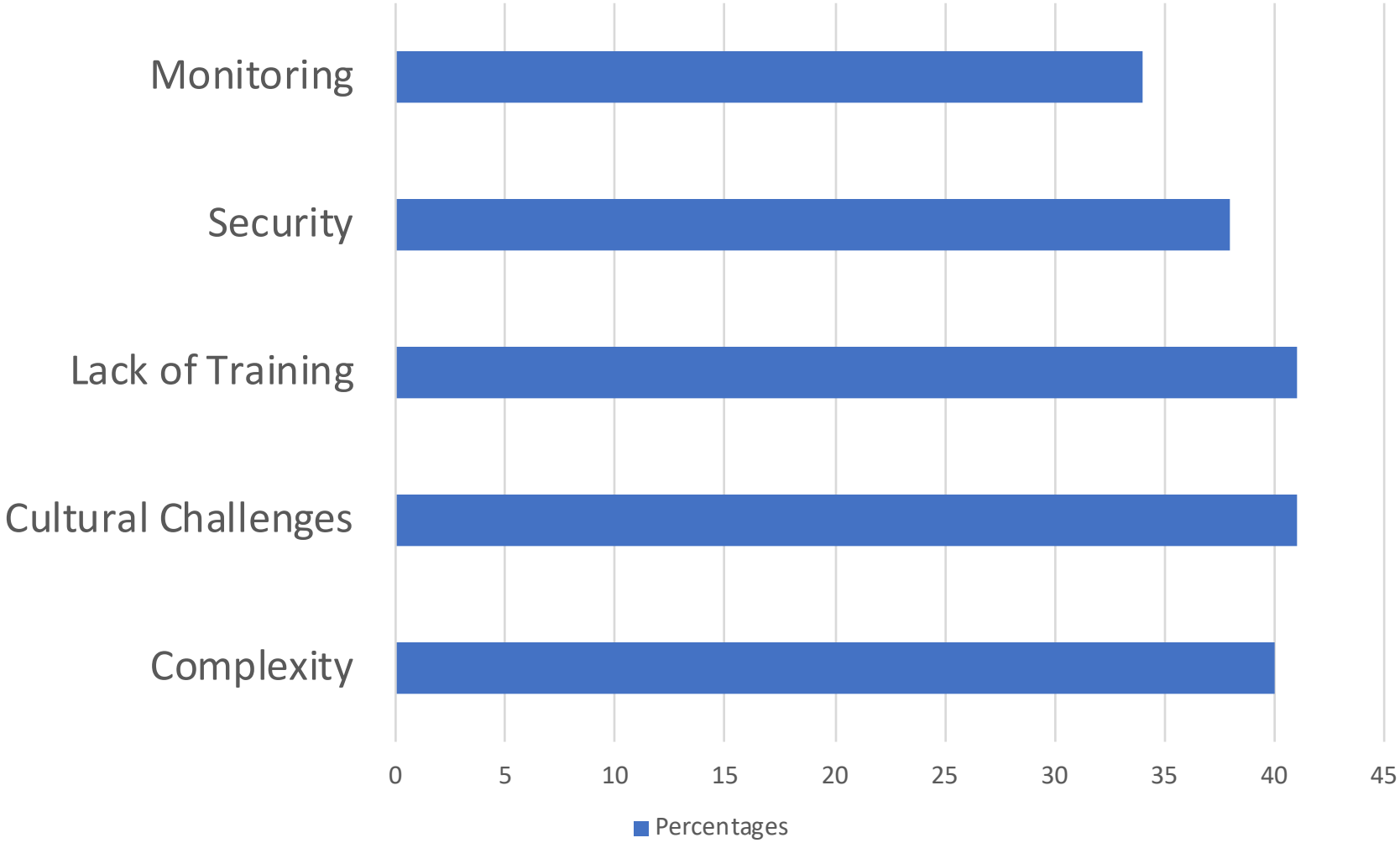


Adoption over public and on-prem

# Kubernetes Dominates Container Management



# Top 5 challenges to cloud native adoption...

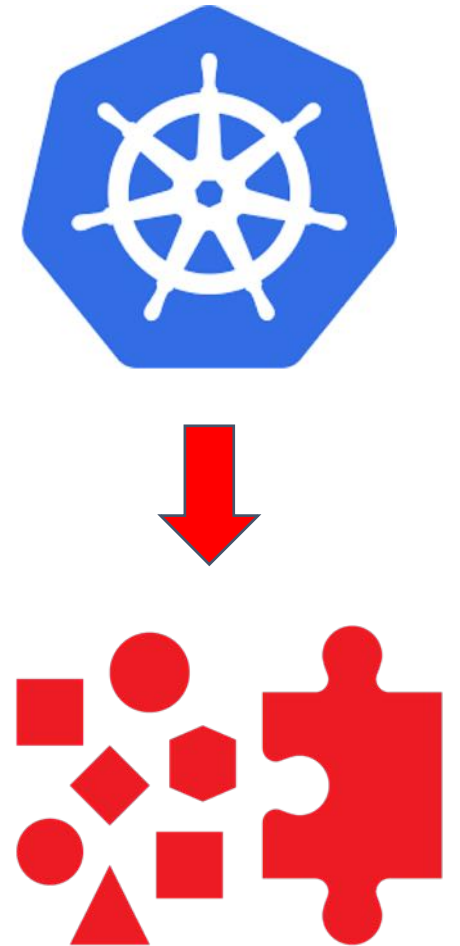




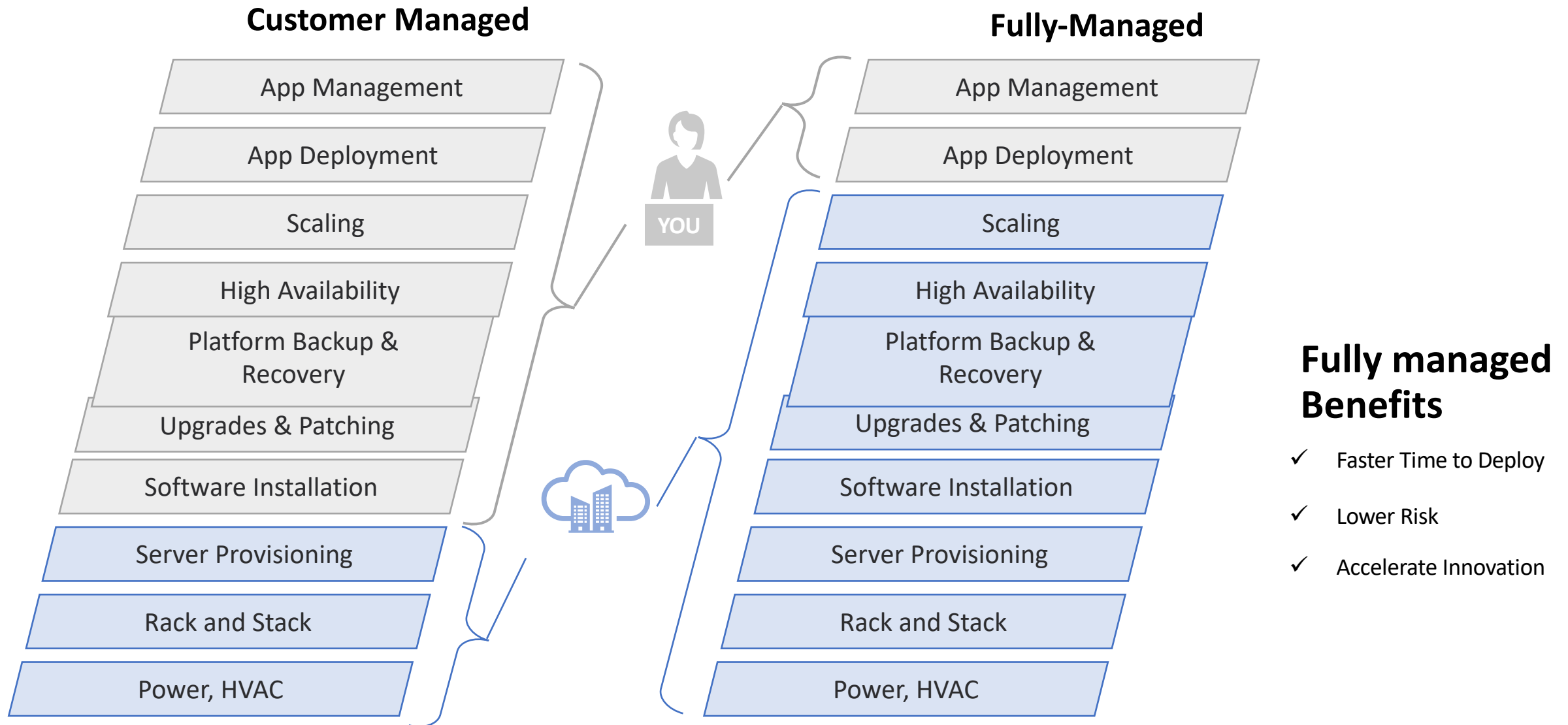
# Kubernetes & Cloud Native Challenges

- Managing, maintaining, upgrading Kubernetes Control Plane
  - API Server, etcd, scheduler etc....
- Managing, maintaining, upgrading Kubernetes Data Plane
  - In place upgrades, deploy parallel cluster etc....
- Figuring out container networking & storage
  - Overlays, persistent storage etc... - it should just work
- Managing Teams
  - How do I manage & control team access to my clusters?
- Security, security, security

Source: Oracle Customer Survey 2018



# How Are Teams Addressing Complexity, Training Issues?



A close-up shot of Gene Wilder as Charlie Bucket from the 1971 film "Willy Wonka & the Chocolate Factory". He is wearing a purple suit jacket, a white shirt, a tan bow tie, and a brown top hat. He has a joyful expression, with his eyes wide and a slight smile. His right hand is resting against his cheek, with his fingers slightly curled. The background is dark and out of focus, showing some indistinct shapes and colors.

Which brings us to security...



Where no news, is good news!



# Unsecured K8s dashboards



< Back

Research

## Lessons from the Cryptojacking Attack at Tesla

by RedLock CSI Team | 02.20.18, 6:00 AM

- Unsecured Kubernetes Dashboard with account creds.
  - Used this to mine cryptocurrency.
  - 2017: Aviva
  - 2018: Tesla, Weight Watchers
- 
- <https://redlock.io/blog/cryptojacking-tesla>

# Kubelet credentials hack

The screenshot shows a Hackerone report for issue #341876. The report is titled "SSRF in Exchange leads to ROOT access in all instances" and is categorized as a "Resolved (Closed)" issue with a "Medium (6.9)" severity. It was reported to Shopify on May 23, 2018, at 4:09pm. The asset is "exchange.shopify.com (Domain)" and the weakness is "Server-Side Request Forgery (SSRF)". A bounty of \$25,000 was awarded. The summary states that Shopify infrastructure is isolated into subsets, and a server-side request forgery bug in the screenshotting functionality of Shopify Exchange was exploited to gain root access to any container in one particular subset. The report also mentions that the bug was fixed by deploying a metadata concealment proxy and that the \$25,000 bounty was awarded as a Shopify Core RCE. The timeline shows that the report was submitted to Shopify on April 22nd (5 months ago) and includes a link to "The Exploit Chain - How to get root access on all Shopify instances". The exploit chain includes two steps: 1. Access Google Cloud Metadata, which involves creating a store (partners.shopify.com) and editing the template "password.liquid" to add a script that accesses the metadata.

hackerone FOR BUSINESS FOR HACKERS HACKTIVITY COMPANY TRY HACKERONE

350 #341876 SSRF in Exchange leads to ROOT access in all instances

State: Resolved (Closed) Severity: Medium (6.9)

Disclosed publicly: May 23, 2018 4:09pm -0500

Reported To: Shopify

Asset: exchange.shopify.com (Domain)

Weakness: Server-Side Request Forgery (SSRF)

Bounty: \$25,000

SUMMARY BY SHOPIFY

Shopify infrastructure is isolated into subsets of infrastructure. @Oxacb reported it was possible to gain root access to any container in one particular subset by exploiting a server side request forgery bug in the screenshotting functionality of Shopify Exchange. Within an hour of receiving the report, we disabled the vulnerable service, began auditing applications in all subsets and remediating across all our infrastructure. The vulnerable subset did not include Shopify core.

After auditing all services, we fixed the bug by deploying a metadata concealment proxy to disable access to metadata information. We also disabled access to internal IPs on all infrastructure subsets. We awarded this \$25,000 as a Shopify Core RCE since some applications in this subset do have access to some Shopify core data and systems.

TIMELINE

Oxacb submitted a report to Shopify.

**The Exploit Chain - How to get root access on all Shopify instances**

1 - Access Google Cloud Metadata

- 1: Create a store (partners.shopify.com)
- 2: Edit the template "password.liquid" and add the following content:

```
<script>
window.location="http://metadata.google.internal/computeMetadata/v1beta1/instance/service-accounts/default
// iframes don't work here because Google Cloud sets the 'X-Frame-Options: SAMEORIGIN' header.
</script>
```

- Shopify: Server Side request Forgery
- Get kubelet certs/private key
- Root access to any container in part of infrastructure.
- <https://hackerone.com/reports/341876>

# CVE's Happen...

## In a Nutshell, Kubernetes...

... has had 70,892 commits made by 2,241 contributors representing 1,537,561 lines of code

## In a Nutshell, docker...

... has had 257,984 commits made by 11,382 contributors representing 9,236,254 lines of code

Even more relevant with increased production usage of containers...

# CVE's Happen...

## In a Nutshell, Kubernetes...

... has had 70,892 commits made by 2,241 contributors representing 1,537,561 lines of code

## In a Nutshell, docker...

... has had 257,984 commits made by 11,382 contributors representing 9,236,254 lines of code



Privilege  
Escalation

**CVE-2018-1002105: proxy request handling in kube-apiserver can leave vulnerable TCP connections #71411**

Container  
Escaping

**DOCKER SECURITY UPDATE: CVE-2019-5736 AND CONTAINER SECURITY BEST PRACTICES**



TOTAL RESULTS

2,367

TOP COUNTRIES



China	833
United States	602
Germany	153
France	110
Singapore	47

TOP ORGANIZATIONS

Amazon.com	323
Hangzhou Alibaba Advertising Co.,Ltd.	276
Tencent cloud computing	207
Hetzner Online GmbH	70
Digital Ocean	70

TOP OPERATING SYSTEMS

Linux 3.x	1
-----------	---

TOP VERSIONS

3.3.2	299
3.2.18	138
3.3.8	128
3.2.22	123
3.2.17	82

**Tencent cloud computing**  
 Added on 2018-09-26 19:13:25 GMT  
 China  
[Details](#)

etcd  
 Name: etcd\_18.8.128.13  
 Version: 3.3.2  
 Uptime: 79h36m33.232183855s  
 Peers: http://18.8.128.13:2380

**Amazon Corporate Services Pty**  
 Added on 2018-09-26 19:08:40 GMT  
 Australia, Sydney  
[Details](#)

etcd  
 Name: MFR-54nodeap-southeast-2002  
 Version: 3.3.2  
 Uptime: 54h22m1.357953836s  
 Peers: http://13.236.207.66:2380

**Spectrum Business**  
 Added on 2018-09-26 18:52:54 GMT  
 United States, Flower Mound  
[Details](#)

etcd  
 Name: core2  
 Version: 3.2.5  
 Uptime: 5h38m55.482175596s  
 Peers: http://18.11.36.2:2380, http://18.11.36.2:7001

**Red Hat**  
 Added on 2018-09-26 18:48:53 GMT  
 United States  
[Details](#)

etcd  
 Name: rdocloud-devstack4.rdocloud  
 Version: 3.2.17  
 Uptime: 635h2m38.382859797s  
 Peers: http://192.168.4.6:2380

**Amazon.com**  
 Added on 2018-09-26 18:45:20 GMT  
 United States, Ashburn  
[Details](#)


etcd  
 Name: master-8  
 Version: 3.2.18  
 Uptime: 78h16.45881983s

SHODAN Search Results for query: port:2379 product:etcd

Navigation: Explore, Downloads, Reports, Developer Pricing, Enterprise Access

Actions: Maps, Share Search, Download Results, Create Report

**TOTAL RESULTS: 2,367**



Country	Count
China	933
United States	662
Germany	153
France	110
Singapore	67

Organization	Count
Amazon.com	323
Hangzhou Alibaba Advertising Co.,Ltd.	276
Tencent cloud computing	207
Hetzner Online GmbH	70
Digital Ocean	70

**OPERATING SYSTEMS**

Linux 3.x	1
-----------	---

**TOP VERSIONS**

3.3.2	299
3.2.18	138
3.3.8	128
3.2.22	123
3.2.17	82

**Sample Results:**

- Tencent cloud computing** (China)
  - Name: etcd\_18.8.128.13
  - Version: 3.3.2
  - Uptime: 79h38m33.232183855s
  - Peers: http://18.8.128.13:2380
- Amazon.com** (Sydney)
  - Name: MFR-54nodeap-southeast-2002
  - Version: 3.3.2
  - Uptime: 54h22m1.357953836s
  - Peers: http://13.236.207.66:2380
- Spectrum Business** (United States, Flower Mound)
  - Name: core2
  - Version: 3.2.5
  - Uptime: 5h38m55.482175596s
  - Peers: http://18.11.36.2:2380, http://18.11.36.2:7001
- Red Hat** (United States)
  - Name: rdocloud-devstack4.rdocloud
  - Version: 3.2.17
  - Uptime: 635h2m38.382859797s
  - Peers: http://192.168.4.6:2380
- Amazon.com** (Ashburn)
  - Name: master-8
  - Version: 3.2.18
  - Uptime: 78h16.45881983s

SHODAN search query: port:2379 product:etcd

Explore Downloads Reports Developer Pricing Enterprise Access

Maps Share Search Download Results Create Report

TOTAL RESULTS: 2,367

World map showing search results distribution by country.

Country	Count
China	933
United States	602
Germany	153
France	110
Singapore	87
Amazon.com	70

Operating Systems:

OS	Count
Linux 3.x	1

TOP VERSIONS:

Version	Count
3.3.2	299
3.2.18	138
3.3.8	128
3.2.20	123
3.2.17	82

Sample Results:

- Tencent cloud computing**  
Added on: 2018-09-26 18:13:25 GMT  
China  
etcd  
Name: etcd\_18.8.128.13  
Version: 3.3.2  
Uptime: 79h38m33.232183855s  
Peers: http://18.8.128.13:2380
- Amazon.com**  
Added on: 2018-09-26 18:02:54 GMT  
United States, Flower Mound  
etcd  
Name: rdocloud-devstack4.rdocloud  
Version: 3.2.17  
Uptime: 633h2m38.382859797s  
Peers: http://192.168.4.6:2380
- Amazon.com**  
Added on: 2018-09-26 18:40:20 GMT  
United States, Ashburn  
etcd  
Name: master-8  
Version: 3.2.18  
Uptime: 78h16.48881983s

W W A T ?

A close-up shot of Gene Wilder as Charlie Bucket from the 1971 film 'Chocolat'. He is wearing a brown top hat, a purple jacket, and a light-colored bow tie. He has a surprised or questioning expression on his face, with wide blue eyes and a slightly open mouth. The background is blurred, showing what appears to be a festive or outdoor setting with lights.

How did we get here?



**“Kubernetes is too complicated”**



A person is seen from the side, wearing a light-colored flight suit, looking at a complex instrument panel. The panel is filled with numerous circular gauges, dials, and digital displays, some of which are illuminated with green and red lights. The person's right hand is visible, resting on a control knob or switch. The overall scene is dimly lit, emphasizing the glowing elements of the cockpit.

**“Kubernetes is too complicated”**

**“We hope it’ll get easier”**





**I want to get better!  
But where to start?**



Let's look at:

- Attack Surface
  - More importantly, how to limit damage
- Security related features in K8s
  - The more you know, the better you build
- Opensource Tooling to help
  - Because we all need help



**Attack Surface**

# Attack Surface

**Goal: Reduce the attack surface**

- Analysis for:
  - Host(s)
  - Container (Images and running)
  - Kubernetes Cluster

# Attack Surface: Host

- These are the machines you're running Kubernetes on.
- Age old principles of Linux still apply:
  - Enable SELinux
  - AppArmor
  - Seccomp
  - Hardened Images
- Goal: Minimize privilege to applications running on the host
- Good news: Already a wealth of information on this subject!
  - <http://imgtfy.com/?q=how+to+reduce+attack+surface+linux>


# Attack Surface: Container Images

**GOAL: Know your base image when building containers**

# Attack Surface: Container Images

**GOAL: Know your base image when building containers**

The screenshot shows the Docker Hub page for the repository `karthequian/ruby`. The repository is public and has over 500,000 pulls. The short description is "A simple sinatra example". The full description is empty. The Docker pull command is `docker pull karthequian/ruby`. The owner is `karthequian`.

Short Description	Docker Pull Command
A simple sinatra example	<code>docker pull karthequian/ruby</code>
Full Description	Owner
Full description is empty for this repo.	 karthequian

# Attack Surface: Container Images

**GOAL: Know your base image when building containers**

The screenshot shows the Docker Hub page for the repository `karthequian/ruby`. The repository is public and has over 500,000 pulls. The page includes a short description, a full description (which is empty), a Docker pull command, and the owner's information.

Key elements highlighted with blue circles:

- The repository name `karthequian/ruby` in the top left.
- The pull count `500K+` in the top right.
- The repository name `karthequian/ruby` in the main header.
- The short description: `A simple Sinatra example`.
- The full description: `Full description is empty for this repo.`
- The Docker pull command: `docker pull karthequian/ruby`.
- The owner's name: `karthequian`.

# Attack Surface: Container Images

**GOAL: Know your base image when building containers**

- When in doubt, stick to an official images!



- Or start from a sane base image (example: alpine linux)

# Attack Surface: Container Images

## **GOAL: Smaller the image, the better**

- Less things for an attacker to exploit.
- Quicker to push, quicker to pull.



# Attack Surface: Container Images

## **GOAL: Don't rely on :latest tag**

- :latest image yesterday might not be :latest image tomorrow
- Instead, you'd want to know what specific version you're operating with.
- Side benefit: If there is a new vulnerability announced for OS version x.y.z, you know immediately whether you're running that version!

# Attack Surface: Container Images

## **GOAL: Check for vulnerabilities periodically**

- Plenty of ways to do this in registries. We'll cover more in the tooling section

# Attack Surface: Running Containers

## **GOAL: Don't run as root**

- Containers running as root might be completely unnecessary for the actual application.
- If compromised, attacker can do a lot more things..
- Pod security policies can help (we'll see how later).

# Attack Surface: Running Containers

## **GOAL: Limit host mounts**

- Be wary of images that require broad access to paths on the host.
- Limit your host mount to a smaller subset of directories.
- Reduces blast radius on compromise.

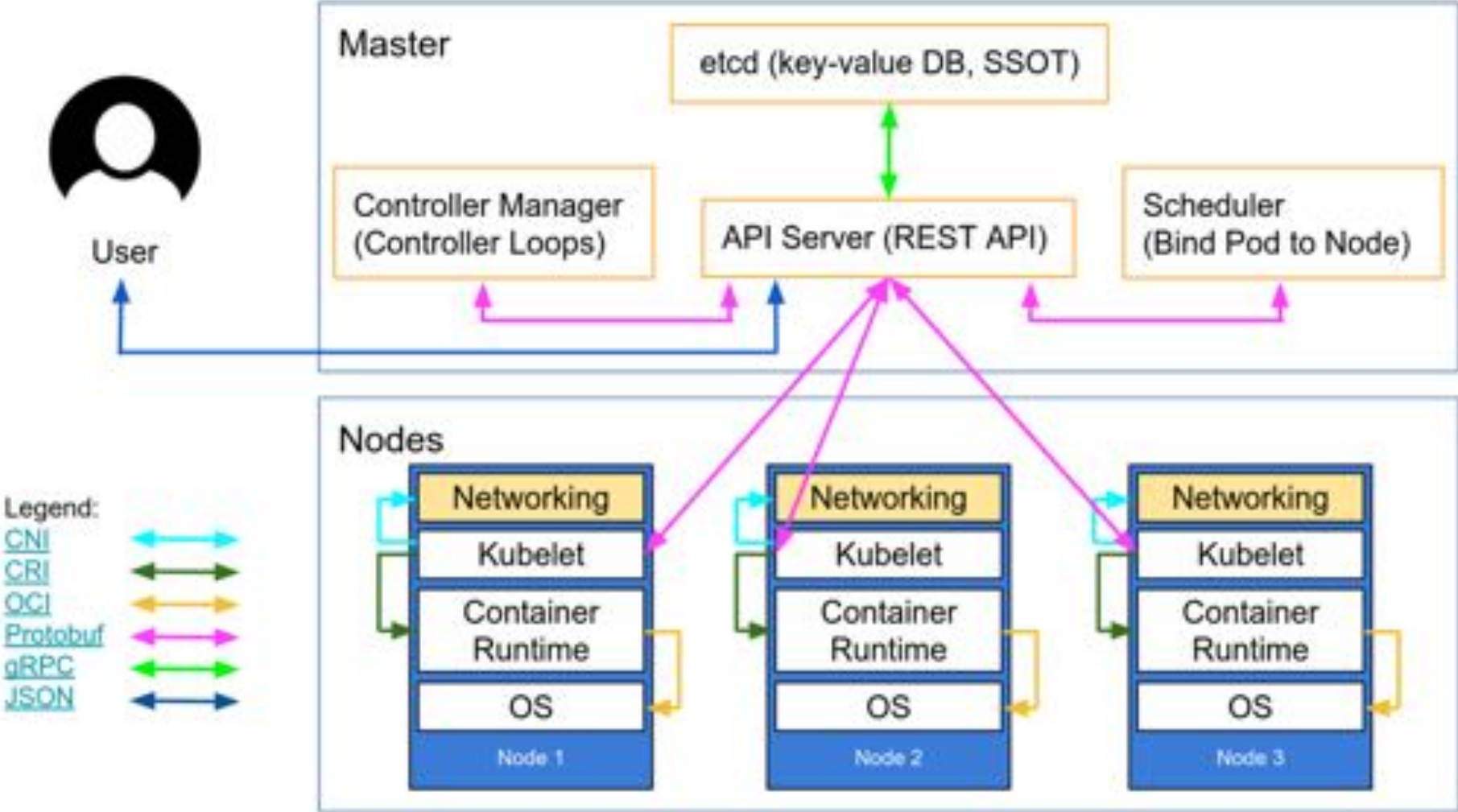
# **Attack Surface: Kubernetes Cluster**

# Kubernetes Cluster- TLS

## TLS ALL THE THINGS



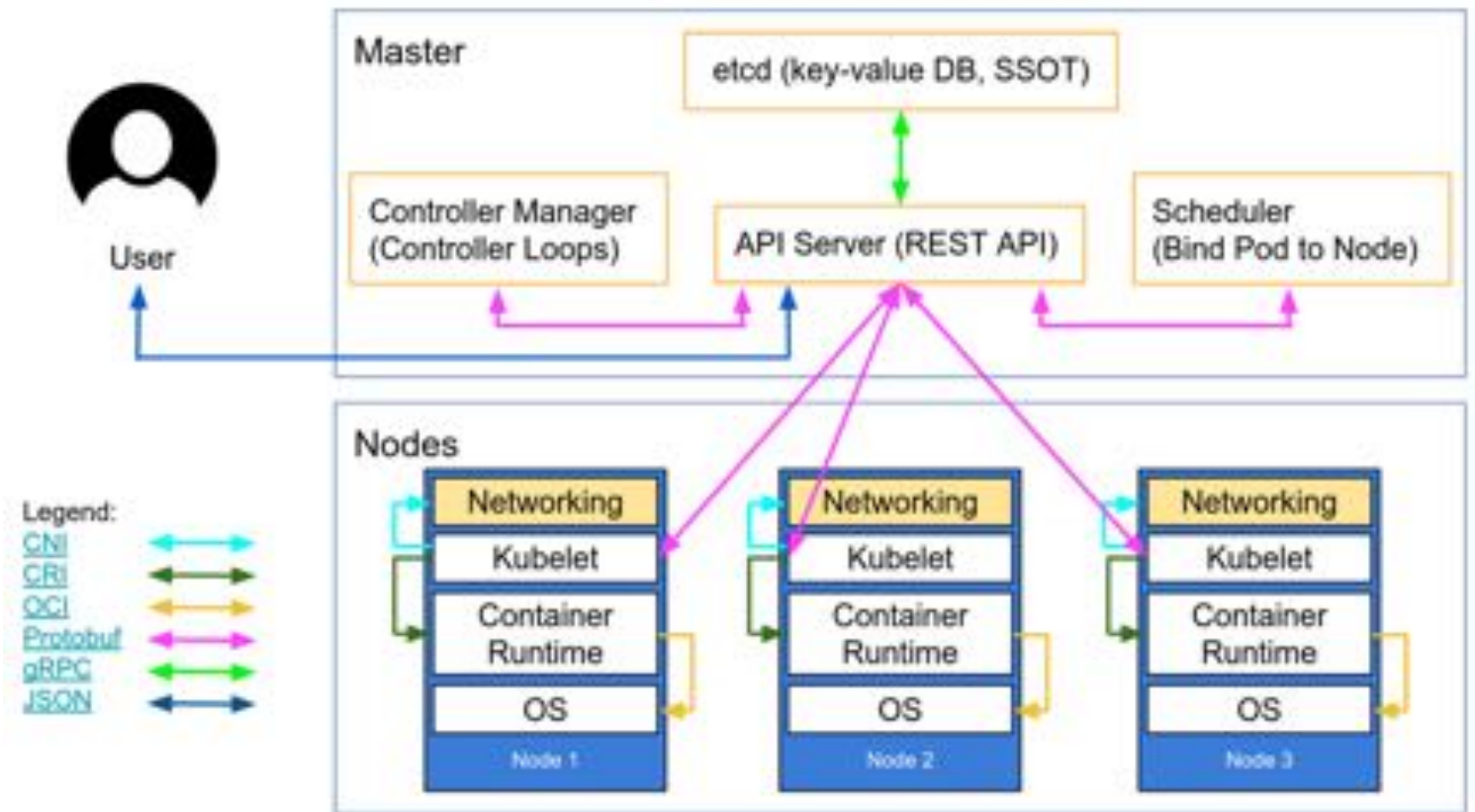
# Kubernetes Cluster- TLS



# Kubernetes Cluster- TLS

- TLS Checklist:

1. User and Master
2. Nodes and Master
3. Everything etcd






# CVE's

## **GOAL: Have an upgrade strategy**

- Because...CVE's are fixed in new minor versions.
- Don't treat K8s as "install once, run all the time".
- Make your K8s install repeatable for different versions.
  
- ..Or use a Managed Provider.
  - Either automatically patch for you, or tell you what to do.

A close-up shot of Gene Wilder as Willy Wonka. He is wearing a brown top hat, a purple suit jacket, a white shirt, and a patterned bow tie. He has a wide-eyed, slightly mischievous expression. He is holding a small, multi-colored ring on his right index finger. The background is dark and out of focus.

We're a little  
better off now.

But what else to do?

# K8s Features

How can the platform help me make secure choices?



# K8s Features

- Kubernetes Secrets
- Authentication
- Authorization
- Audit Logging
- Network Policies
- Pod security policies
- Open Policy Agent



# Kubernetes Secrets

- **GOAL: Use Kubernetes secrets to store sensitive data instead of config maps.**
- Also look at: secrets encryption provider.
  - Controls how etcd encrypts API data
  - **--experimental-encryption-provider-config**
- <https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/>

# Authentication and Authorization


- Do you know how you are authenticating with Kubernetes?
- Many ways to Authenticate
  - Client Certs
  - Static token file
  - Service Account tokens
  - OpenID
  - Webhook Mode
  - And more (<https://kubernetes.io/docs/reference/access-authn-authz/authentication/>)



A close-up photograph of a small, light-colored kitten with dark stripes, peeking its face out from under a thick, textured blanket. The kitten has large, wide, yellowish-brown eyes and a slightly open mouth, looking directly at the camera with a curious expression. The background is dark and out of focus, showing a white circular vent on a wall.

Goal: Pick a strategy that fits  
your use case

Whatever you do,  
DO NOT YOLO!

A close-up photograph of a small, light-colored tabby kitten with large, bright orange eyes. The kitten is lying under a thick, textured, brownish-grey blanket, with only its head and front paws visible. The kitten has a curious and slightly nervous expression, looking directly at the camera. The background is dark and out of focus, suggesting an indoor setting. The overall lighting is soft and warm, highlighting the kitten's fur and the texture of the blanket.

If you DO NOT YOLO...

You can pick an authz strategy..

# Authentication and Authorization

## Authorization Modules

---

- **Node** - A special-purpose authorizer that grants permissions to kubelets based on the pods they are scheduled to run. To learn more about using the Node authorization mode, see [Node Authorization](#).
- **ABAC** - Attribute-based access control (ABAC) defines an access control paradigm whereby access rights are granted to users through the use of policies which combine attributes together. The policies can use any type of attributes (user attributes, resource attributes, object, environment attributes, etc). To learn more about using the ABAC mode, see [ABAC Mode](#).
- **RBAC** - Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within an enterprise. In this context, access is the ability of an individual user to perform a specific task, such as view, create, or modify a file. To learn more about using the RBAC mode, see [RBAC Mode](#)
  - When specified RBAC (Role-Based Access Control) uses the `rbac.authorization.k8s.io` API group to drive authorization decisions, allowing admins to dynamically configure permission policies through the Kubernetes API.
  - To enable RBAC, start the apiserver with `--authorization-mode=RBAC`.
- **Webhook** - A Webhook is an HTTP callback: an HTTP POST that occurs when something happens; a simple event-notification via HTTP POST. A web application implementing WebHooks will POST a message to a URL when certain things happen. To learn more about using the Webhook mode, see [Webhook Mode](#).

<https://kubernetes.io/docs/reference/access-authn-authz/authorization/>



# Authentication and Authorization

- Pro tip: Nobody uses ABAC anymore. Don't be that guy....
- RBAC is the defacto standard
  - Based on roles and role bindings
  - Good set of defaults: <https://github.com/uruddarraju/kubernetes-rbac-policies>
- Can use multiple authorizers together, but can get confusing.
  - 1<sup>st</sup> authorizer to authorize passes authz

# Kubernetes Cluster- Audit Logs

- Wat?
- “Kubernetes auditing provides a security-relevant chronological set of records documenting the sequence of activities that have affected system by individual users, administrators or other components of the system.”
- Answers: What/when/who/where information on security events.
- **Your job:** Periodically watch Kubernetes Audit logs
- <https://kubernetes.io/docs/tasks/debug-application-cluster/audit/>



See, you know how to take the reservation, you just don't know how to hold the reservation and that's really the most important part of the reservation, the holding. Anybody can just take them.

— *Jerry Seinfeld* —

**AZ QUOTES**



# Kubernetes Cluster- Network Policies

- Consider adding a network policy to the cluster...
- Default Policy: All pods can talk to all other pods.
- Consider limiting this with a Network Policy
- <https://kubernetes.io/docs/concepts/services-networking/network-policies/>

# Kubernetes Cluster- Pod Security Policies

- Consider adding Pod Security policies
- PodSecurityPolicy: A Defined set of conditions a pod must run with.
- Think of this as authorization for pods.

# Kubernetes Cluster: Pod Security Policies

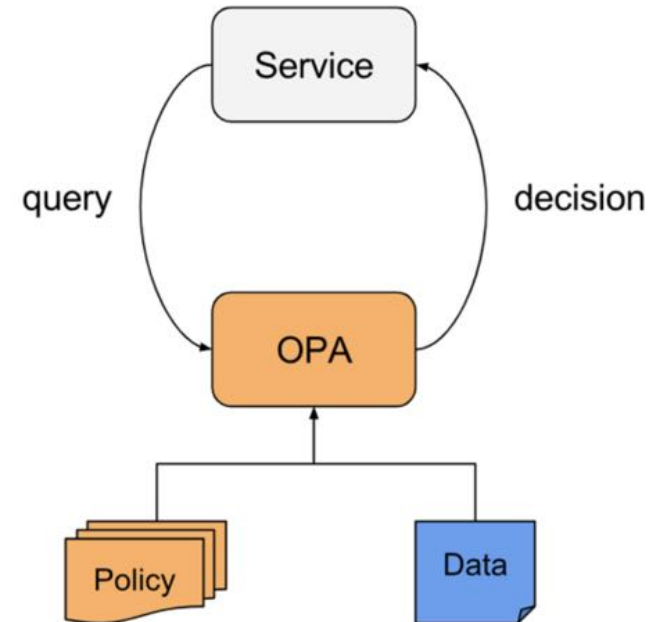
Capability for an admin to control specific actions

Control Aspect	Field Names
Running of privileged containers	<a href="#">privileged</a>
Usage of host namespaces	<a href="#">hostPID</a> , <a href="#">hostIPC</a>
Usage of host networking and ports	<a href="#">hostNetwork</a> , <a href="#">hostPorts</a>
Usage of volume types	<a href="#">volumes</a>
Usage of the host filesystem	<a href="#">allowedHostPaths</a>
White list of Flexvolume drivers	<a href="#">allowedFlexVolumes</a>
Allocating an FSGroup that owns the pod's volumes	<a href="#">fsGroup</a>
Requiring the use of a read only root file system	<a href="#">readOnlyRootFilesystem</a>
The user and group IDs of the container	<a href="#">runAsUser</a> , <a href="#">supplementalGroups</a>
Restricting escalation to root privileges	<a href="#">allowPrivilegeEscalation</a> , <a href="#">defaultAllowPrivilegeEscalation</a>
Linux capabilities	<a href="#">defaultAddCapabilities</a> , <a href="#">requiredDropCapabilities</a> , <a href="#">allowedCapabilities</a>
The SELinux context of the container	<a href="#">seLinux</a>
The AppArmor profile used by containers	<a href="#">annotations</a>
The seccomp profile used by containers	<a href="#">annotations</a>
The sysctl profile used by containers	<a href="#">annotations</a>

<https://kubernetes.io/docs/concepts/policy/pod-security-policy/#what-is-a-pod-security-policy>

# Open Policy Agent

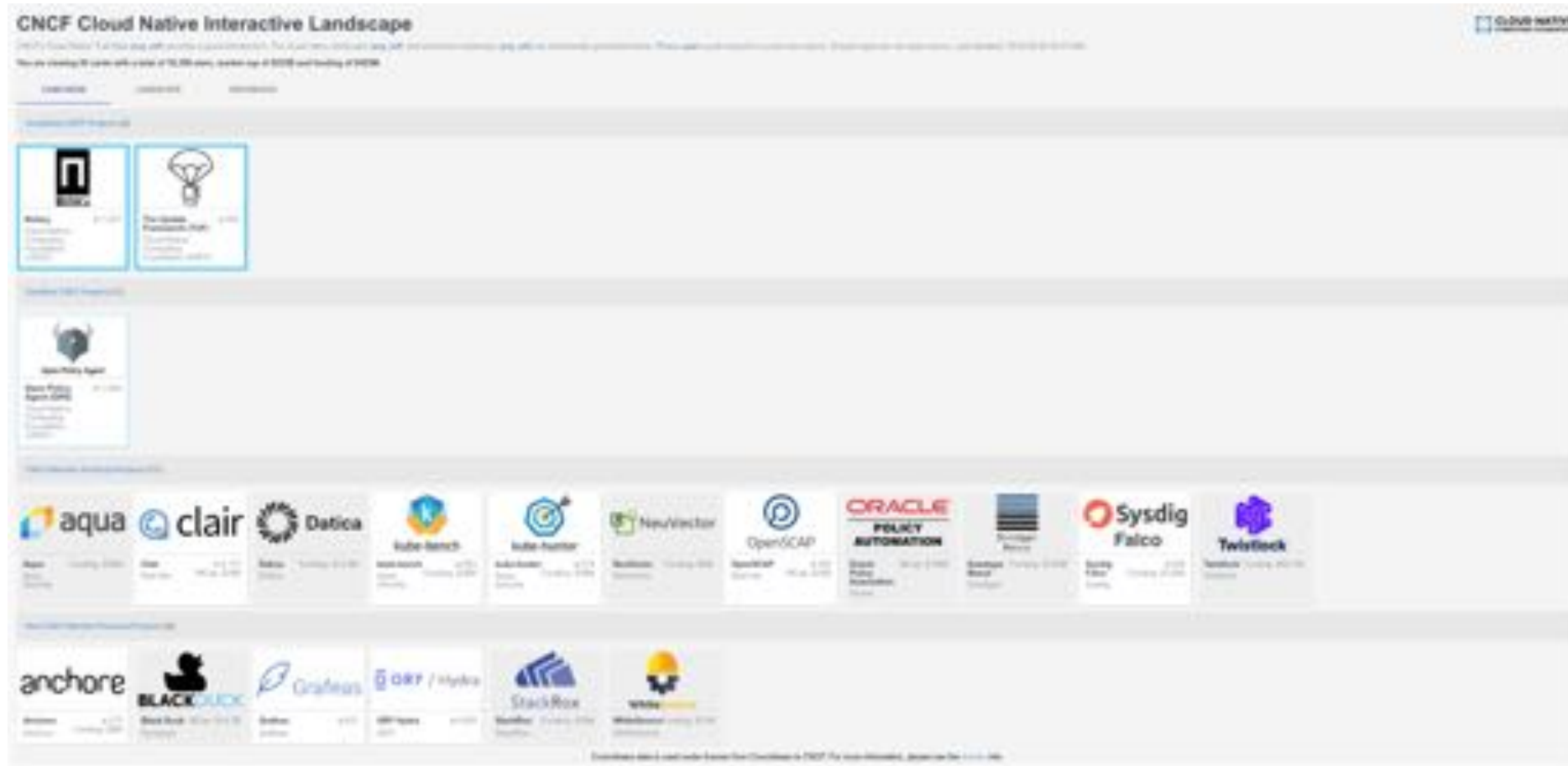
- Policy based control for your whole environment.
- Full featured Policy Engine to offload policy decisions from each application/service.
  - Deploy OPA alongside your service
  - Add policy data to OPA's store
  - Query OPA on decisions.
- Great idea, still early, watch this space...
- Standardize policies for all clusters
- <https://www.openpolicyagent.org/>



# Opensource Tooling



# Keep tabs on the CNCF Security landscape



<https://landscape.cncf.io/landscape=security-compliance>



# CNCF Projects



- “The Update Framework”
- Is a framework or a methodology.
- Used for secure software updates.
- Based on ideas surrounding trust and integrity.



- Is a project.
- Based on TUF.
- A solution to secure software updates and distribution.
- Used in Docker Trusted Registry.

# Clair



- Open source project for the static analysis of vulnerabilities in containers.
- Find vulnerable images in your repo.
- Built into quay.io, but you can add to your own repo.
- <https://github.com/coreos/clair>



Quay Security Scanner has detected **13** vulnerabilities.

Patches are available for **4** vulnerabilities.

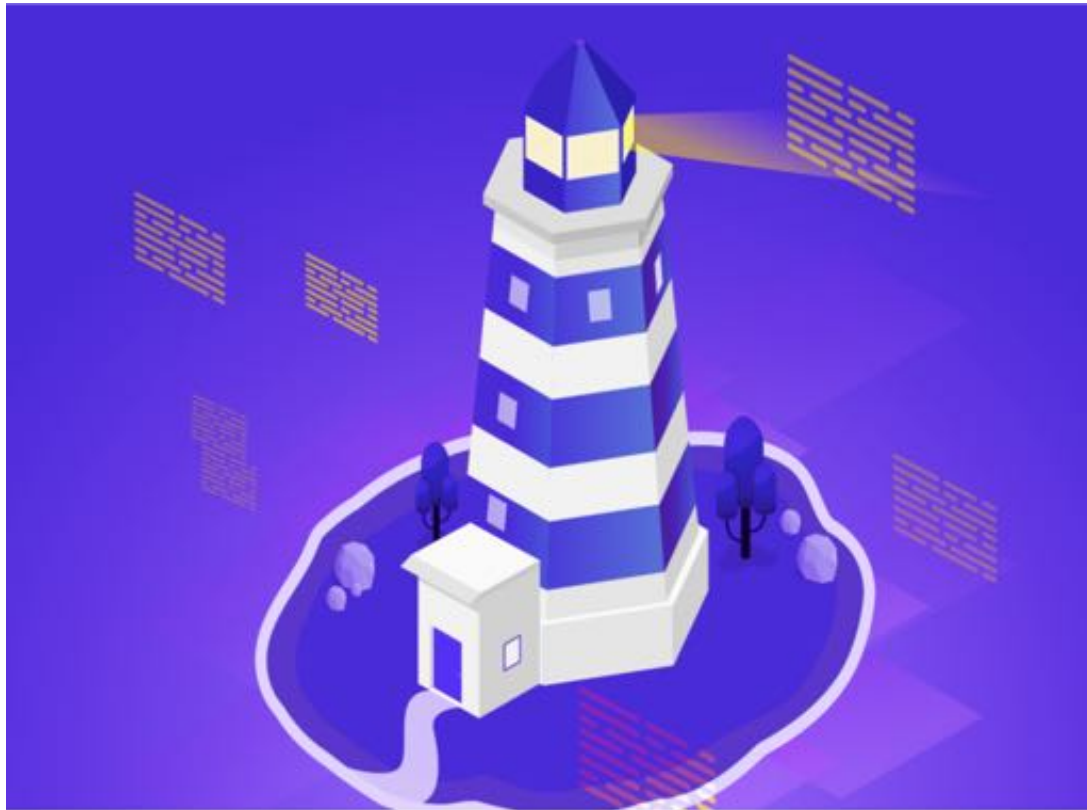
- 1 High-level vulnerabilities.
- 1 Medium-level vulnerabilities.
- 2 Low-level vulnerabilities.
- 5 Negligible-level vulnerabilities.
- 4 Unknown-level vulnerabilities.

### Image Vulnerabilities

Filter vulnerabilities...  Only show fixable

CVE	SEVERITY	PACKAGE	CURRENT VERSION	FRESH VERSION	INTRODUCED IN IMAGE
CVE-2013-7445	High	linux	3.16.7-ckt20-1+deb8u3		<input type="button" value="RUN"/> apt-get up.
CVE-2015-5276	Medium	gcc-4.9	4.9.2-10		<input type="button" value="ADD"/> file:b5391.
CVE-2016-2856	Low	glibc	2.19-18+deb8u3		<input type="button" value="ADD"/> file:b5391.
CVE-2016-0823	Low	linux	3.16.7-ckt20-1+deb8u3		<input type="button" value="RUN"/> apt-get up.
CVE-2005-3660	Negligible *	linux	3.16.7-ckt20-1+deb8u3		<input type="button" value="RUN"/> apt-get up.
CVE-2015-4009	Negligible *	linux	3.16.7-ckt20-1+deb8u3		<input type="button" value="RUN"/> apt-get up.
CVE-2008-4206	Negligible *	python-defaults	2.7.9-1		<input type="button" value="RUN"/> apt-get up.
CVE-2015-8830	Negligible	linux	3.16.7-ckt20-1+deb8u3	<a href="#">3.16.7-ckt20-1+deb8u4</a>	<input type="button" value="RUN"/> apt-get up.
CVE-2013-4392	Negligible *	systemd	215-17+deb8u3		<input type="button" value="ADD"/> file:b5391.
CVE-2015-7525	Unknown	linux	3.16.7-ckt20-1+deb8u3		<input type="button" value="RUN"/> apt-get up.
CVE-2015-8806	Unknown	linux	3.16.7-ckt20-1+deb8u3	<a href="#">3.16.7-ckt20-1+deb8u4</a>	<input type="button" value="RUN"/> apt-get up.
CVE-2016-2547	Unknown	linux	3.16.7-ckt20-1+deb8u3	<a href="#">3.16.7-ckt20-1+deb8u4</a>	<input type="button" value="RUN"/> apt-get up.
CVE-2016-2545	Unknown	linux	3.16.7-ckt20-1+deb8u3	<a href="#">3.16.7-ckt20-1+deb8u4</a>	<input type="button" value="RUN"/> apt-get up.

# Harbor



- Newer! CNCF Project
- Registry product
- Supports vulnerability scanning, image signing and identity control
- Scope is larger than clair

# Harbor

< Projects < Repositories < thomas/postgres  
**thomas/postgres:alpine**

Author: anonymity  
Architecture: amd64  
OS: linux  
Docker Version: 17.06.2-ce  
Scan Completed: Nov 2, 2018

 0 High Level Vulnerabilities  
1 Medium Level Vulnerabilities  
0 Low Level Vulnerabilities  
0 Unknown Level Vulnerabilities

SCAN

Q | C

Vulnerability	Severity	Package	Current version	Fixed in version
> CVE-2018-9251	Low	libcni2	2.9.8-r0	2.9.8-r1
> CVE-2018-14404	Medium	libcni2	2.9.8-r0	2.9.8-r1
> CVE-2018-14567	Medium	libcni2	2.9.8-r0	2.9.8-r1

# Kube-bench

- Checks whether a Kubernetes cluster is deployed according to security best practices.
- Run this after creating your K8s cluster.
- <https://github.com/aquasecurity/kube-bench>
- Defined by the CIS Benchmarks Docs: <https://www.cisecurity.org/benchmarks/>
- Run it against your Kubernetes Master, or Kubernetes node.



kube-bench



# Kube-bench example

```
➤ ~$ kubectl logs kube-bench-node
[INFO] 2 Worker Node Security Configuration
[INFO] 2.1 Kubelet
[FAIL] 2.1.1 Ensure that the --allow-privileged argument is set to false (Scored)
[PASS] 2.1.2 Ensure that the --anonymous-auth argument is set to false (Scored)
[PASS] 2.1.3 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)
[PASS] 2.1.4 Ensure that the --client-ca-file argument is set as appropriate (Scored)
[PASS] 2.1.5 Ensure that the --read-only-port argument is set to 0 (Scored)
[FAIL] 2.1.6 Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Scored)
[FAIL] 2.1.7 Ensure that the --protect-kernel-defaults argument is set to true (Scored)
[FAIL] 2.1.8 Ensure that the --make-iptables-util-chains argument is set to true (Scored)
[FAIL] 2.1.9 Ensure that the --keep-terminated-pod-volumes argument is set to false (Scored)
[FAIL] 2.1.10 Ensure that the --hostname-override argument is not set (Scored)
[FAIL] 2.1.11 Ensure that the --event-qps argument is set to 0 (Scored)
[PASS] 2.1.12 Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Scored)
[PASS] 2.1.13 Ensure that the --cadvisor-port argument is set to 0 (Scored)
[FAIL] 2.1.14 Ensure that the RotateKubeletClientCertificate argument is set to true
[FAIL] 2.1.15 Ensure that the RotateKubeletServerCertificate argument is set to true
[INFO] 2.2 Configuration Files
[FAIL] 2.2.1 Ensure that the kubelet.conf file permissions are set to 644 or more restrictive (Scored)
[FAIL] 2.2.2 Ensure that the kubelet.conf file ownership is set to root:root (Scored)
[FAIL] 2.2.3 Ensure that the kubelet service file permissions are set to 644 or more restrictive (Scored)
[FAIL] 2.2.4 2.2.4 Ensure that the kubelet service file ownership is set to root:root (Scored)
[FAIL] 2.2.5 Ensure that the proxy kubeconfig file permissions are set to 644 or more restrictive (Scored)
[FAIL] 2.2.6 Ensure that the proxy kubeconfig file ownership is set to root:root (Scored)
[WARN] 2.2.7 Ensure that the certificate authorities file permissions are set to 644 or more restrictive (Scored)
[WARN] 2.2.8 Ensure that the client certificate authorities file ownership is set to root:root
```

# Kubesecc

- Helps you quantify risk for Kubernetes resources.
- Run against your K8s applications (deployments/pods/daemonsets etc)
- <https://kubesecc.io/> from controlplane
- Can be used standalone, or as a kubectl plugin (<https://github.com/stefanprodan/kubectl-kubesecc>)

# Kubesecc example

```
~$ kubectl -n kube-system plugin scan deployment/kubernetes-dashboard
scanning deployment/kubernetes-dashboard
deployment/kubernetes-dashboard kubesecc.io score 3
-----
Advise
1. containers[] .securityContext .runAsNonRoot == true
Force the running image to run as a non-root user to ensure least privilege
2. containers[] .securityContext .capabilities .drop
Reducing kernel capabilities available to a container limits its attack surface
3. containers[] .securityContext .readOnlyRootFilesystem == true
An immutable root filesystem can prevent malicious binaries being added to PATH and increase attack cost
4. containers[] .securityContext .runAsUser > 10000
Run as a high-UID user to avoid conflicts with the host's user table
5. containers[] .securityContext .capabilities .drop | index("ALL")
Drop all capabilities and add only those required to reduce syscall attack surface
~$ █
```



# Kubeaudit

- Opensourced from Shopify.
- Auditing your applications in your K8s cluster.
- <https://github.com/Shopify/kubeaudit>
- Little more targeted than Kubesec.



kubeaudit is a program that will help you audit your Kubernetes clusters. Specify `-l` to run kubeaudit using `~/.kube/config` otherwise it will attempt to create an in-cluster client.

#patcheswelcome

Usage:

kubeaudit [command]

Available Commands:

allowpe	Audit containers that allow privilege escalation
caps	Audit container for capabilities
help	Help about any <a href="#">command</a>
image	Audit container images
nonroot	Audit containers running as root
np	Audit namespace network policies
priv	Audit containers running as root
rootfs	Audit containers with read only root filesystems
sat	Audit automountServiceAccountToken = true pods against an empty (default) service account
version	Print the version number of kubeaudit

Flags:

<code>-a, --allPods</code>	Audit againsts pods in all the phases (default Running Phase)
<code>-h, --help</code>	help for kubeaudit
<code>-j, --json</code>	Enable json logging
<code>-c, --kubeconfig string</code>	config file (default is <code>\$HOME/.kube/config</code> )
<code>-l, --local</code>	Local mode, uses <code>~/.kube/config</code> as configuration
<code>-f, --manifest string</code>	yaml configuration to audit
<code>-v, --verbose string</code>	Set the debug level (default "INFO")

# Kubeaudit example

```
- $ /Users/karthik/Downloads/kubeaudit_0.2.0_darwin_amd64/kubeaudit allowpe -c /Users/karthik/.kube/config
ERR0[0003] SecurityContext not set, please set it! KubeType=deployment Name=dotnetworld Namespace=default
ERR0[0003] SecurityContext not set, please set it! KubeType=deployment Name=javademo Namespace=default
ERR0[0003] SecurityContext not set, please set it! KubeType=deployment Name=pran-demo-prometheus-alertmanager Namespace=default
ERR0[0003] SecurityContext not set, please set it! KubeType=deployment Name=pran-demo-prometheus-kube-state-metrics Namespace=default
ERR0[0003] SecurityContext not set, please set it! KubeType=deployment Name=pran-demo-prometheus-pushgateway Namespace=default
ERR0[0003] SecurityContext not set, please set it! KubeType=deployment Name=pran-demo-prometheus-server Namespace=default
ERR0[0003] SecurityContext not set, please set it! KubeType=deployment Name=wishlist-deployment Namespace=default
ERR0[0003] SecurityContext not set, please set it! KubeType=deployment Name=contour Namespace=heptio-contour
ERR0[0003] SecurityContext not set, please set it! KubeType=deployment Name=kube-dns Namespace=kube-system
ERR0[0003] SecurityContext not set, please set it! KubeType=deployment Name=kube-dns-autoscaler Namespace=kube-system
ERR0[0003] SecurityContext not set, please set it! KubeType=deployment Name=kubernetes-dashboard Namespace=kube-system
ERR0[0003] SecurityContext not set, please set it! KubeType=deployment Name=oci-volume-provisioner Namespace=kube-system
ERR0[0003] SecurityContext not set, please set it! KubeType=deployment Name=tiller-deploy Namespace=kube-system
ERR0[0003] SecurityContext not set, please set it! KubeType=daemonSet Name=pran-demo-prometheus-node-exporter Namespace=default
ERR0[0003] AllowPrivilegeEscalation not set which allows privilege escalation, please set to false KubeType=daemonSet Name=kube-flannel-ds Namespace=kube-system
ERR0[0003] AllowPrivilegeEscalation not set which allows privilege escalation, please set to false KubeType=daemonSet Name=kube-proxy Namespace=kube-system
```



**Put it all together...**



# Apply It!

- **Day 1:**
- Know what version of Docker and Kubernetes you use.
- Understand if your control and data plane nodes are hardened.
- Understand how your Docker containers are built.
- Find out how you can optimize for your clusters.



**KNOWING**  
IS HALF THE BATTLE

# Apply It!

- **Week 1:**
- **Build an Automation Pipeline:**
  - To build Docker images on code pushes
  - Versioning strategy for code
  - To build your Kubernetes clusters

# Apply It!

- **1<sup>st</sup> Month**
- Sanitize your code:
  - Know your base images
  - Implement versioning for your containers
  - Invest in a registry (or tooling) that does vulnerability scanning
- Kubernetes:
  - Have an upgrade strategy in place
  - Analyze secrets/sensitive cluster data
  - Turn on audit logging

# Apply It!

- **3 Months:**
- Continuously Monitor
  - Tooling like Kubesec/Kube-audit
- Plan how to address vulnerabilities/CVE's
- K8s:
  - Strategy for Pod Security Policies
  - Strategy for Network Policies
  - Run scans (like kube-bench) on cluster creation



# Apply It!

- **6 Months:**
- Re-ask day 1 questions.
- Review strategies- is it working? What needs tweaking?
- Review tooling- are there new tools that help? Are existing tools working?
- Review CVE's



# Couple more resources to look at:

- 11 ways not to get hacked:  
<https://kubernetes.io/blog/2018/07/18/11-ways-not-to-get-hacked>
- K8s security (from Image Hygiene to Network Policy):  
<https://speakerdeck.com/mhausenblas/kubernetes-security-from-image-hygiene-to-network-policies>

**KEEP CALM  
AND  
KUBE ON**  
*@iteration1*

