Kubernetes for InfoSec

Why does Kubernetes Make Me Feel Like a Newbie?

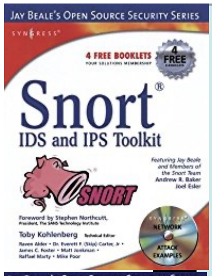
Wild West Hackin' Fest

June 2021

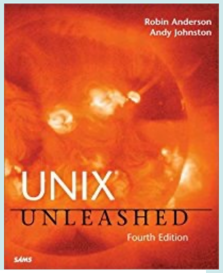
Jay Beale

InGuardians

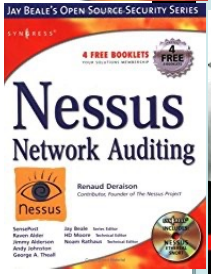
Copyright 2000-2021 Jay Beale


Jay Beale's Open Source Security Series

Snort® IDS and IPS Toolkit


Robin Anderson
Andy Johnston

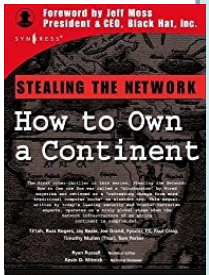
UNIX UNLEASHED
Fourth Edition


Jay Beale's Open Source Security Series

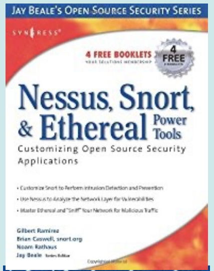
Nessus Network Auditing

Red Hat Linux Internet Server

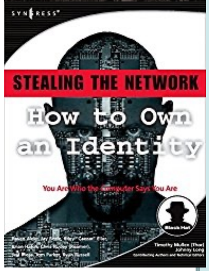
redhat. PRESS

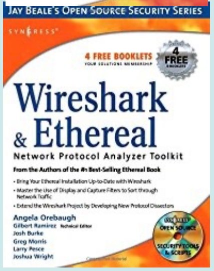

Foreword by Jeff Moss
President & CEO, Black Hat, Inc.

STEALING THE NETWORK
How to Own a Continent


Jay Beale's Open Source Security Series

Nessus, Snort, & Ethereal Power Tools
Customizing Open Source Security Applications


STEALING THE NETWORK
How to Own an Identity


Jay Beale's Open Source Security Series

Wireshark & Ethereal
Network Protocol Analyzer Toolkit

INFORMATION SECURITY

Going the Distance

Graphical Bio

black hat

A PURPLE TEAM VIEW: ATTACKING AND DEFENDING LINUX, DOCKER AND KUBERNETES

JAY BEALE, INGUARDIANS | JULY 31-AUGUST 3

BASTILLE

http://bastille-linux.sourceforge.net/

BUST-A-KUBE

PEIRATES

CAN SEC WEST

INGUARDIANS™ RED TEAM

CIS. Center for Internet Security®

DEFCON

toorcon

SecurityFocus™

OVAL

SANS

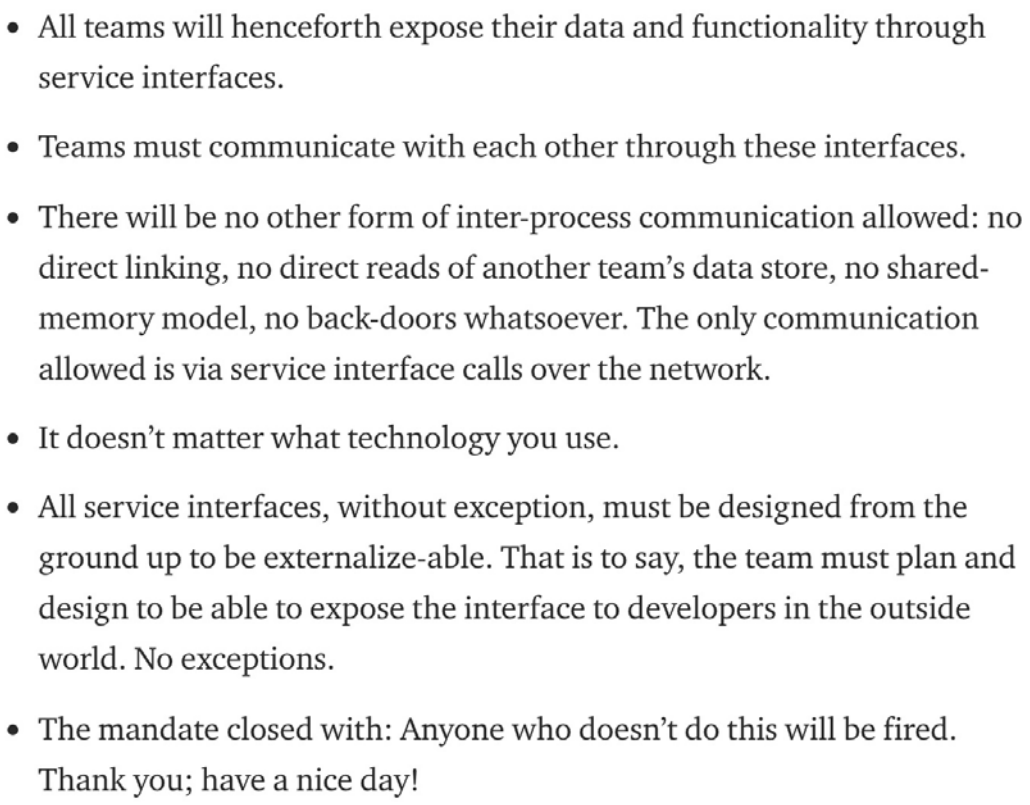
INGUARDIANS™

Kubernetes

Let’s talk about container orchestration!

Then let’s turn into peiratés!

Cloud Native’s Birth: the API (Service) Moment



- All teams will henceforth expose their data and functionality through service interfaces.
- Teams must communicate with each other through these interfaces.
- There will be no other form of inter-process communication allowed: no direct linking, no direct reads of another team’s data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.
- It doesn’t matter what technology you use.
- All service interfaces, without exception, must be designed from the ground up to be externalize-able. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.
- The mandate closed with: Anyone who doesn’t do this will be fired. Thank you; have a nice day!

Jeff Bezos' 2002 API Mandate Memo

Amazon Web Services

- The memo forced every single connectable software project at Amazon to function as a product.
- In 2002, the same year as the memo, Amazon went from an online retailer to the cloud service provider that also operated a retail business.
- Amazon's market share in cloud services is 34%, which is larger than the next three players put together (as of 2017).

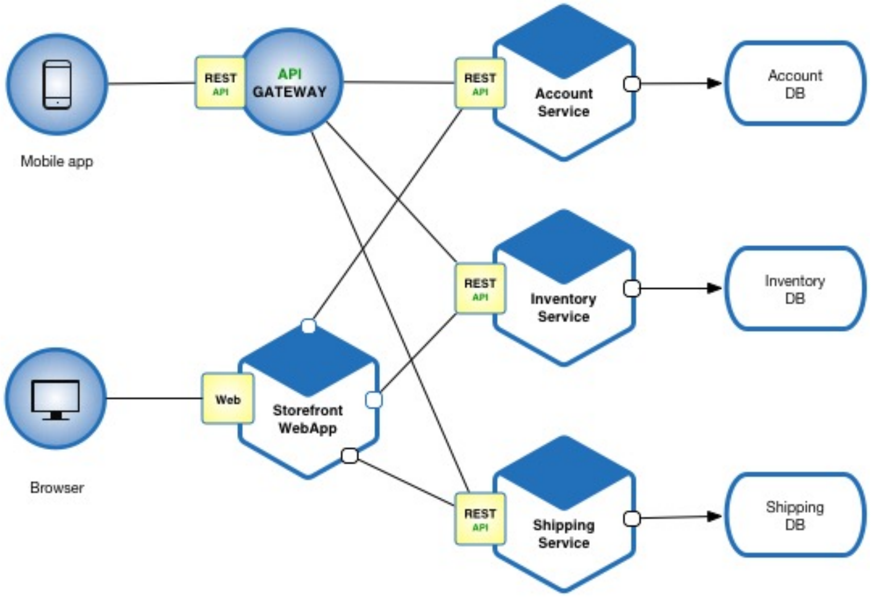
Microsoft: 11%

Google: 8%

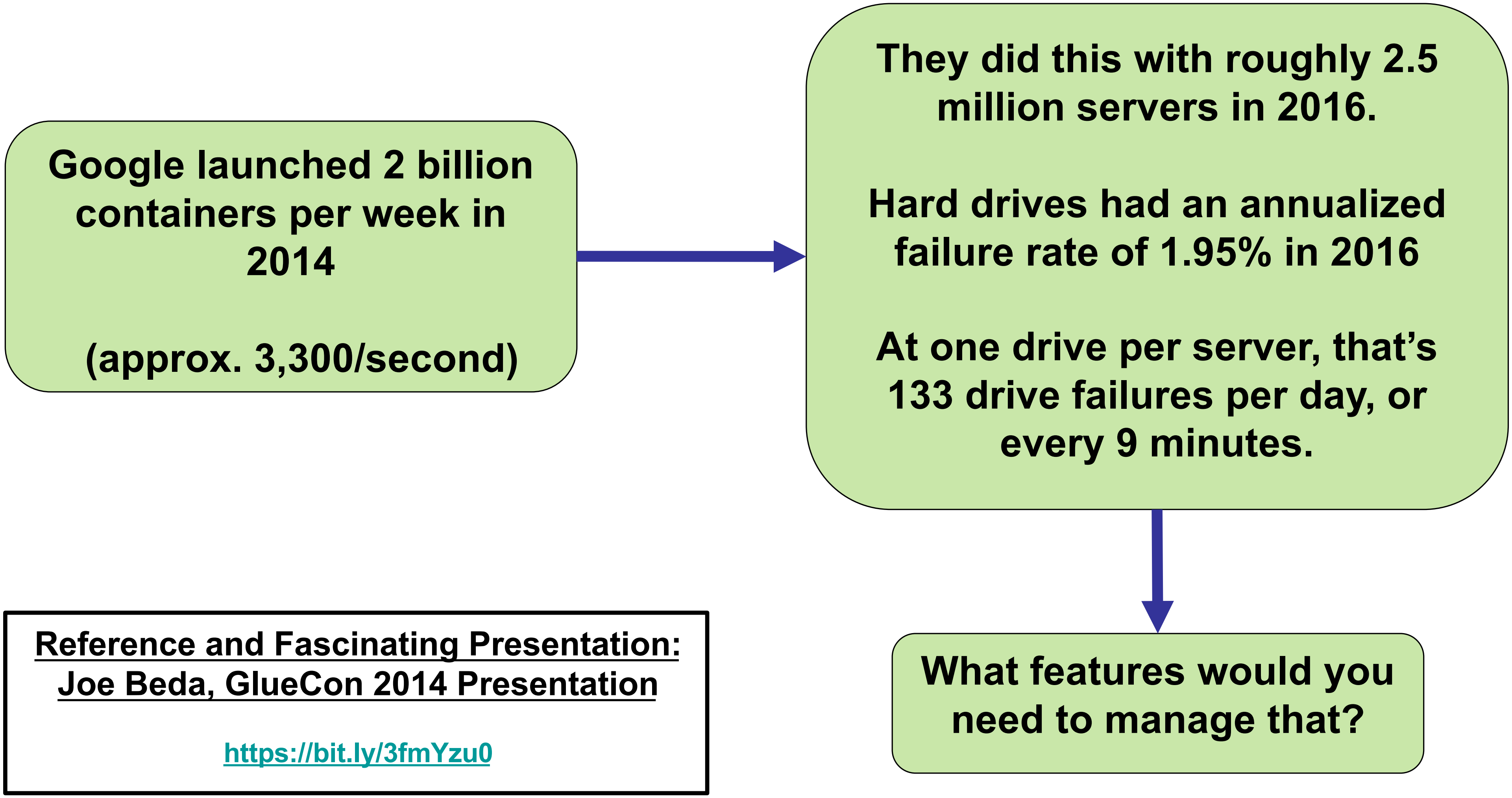
IBM: 6%

2021 Update
AWS: 32%
Azure: 19%
Google: 7%

Microservice Architecture



Credit: https://microservices.io/patterns/microservices.html



Google launched 2 billion containers per week in 2014

(approx. 3,300/second)

They did this with roughly 2.5 million servers in 2016.

Hard drives had an annualized failure rate of 1.95% in 2016

At one drive per server, that’s 133 drive failures per day, or every 9 minutes.

What features would you need to manage that?

Reference and Fascinating Presentation:
Joe Beda, GlueCon 2014 Presentation

https://bit.ly/3fmYzu0

What Does Kubernetes Do?

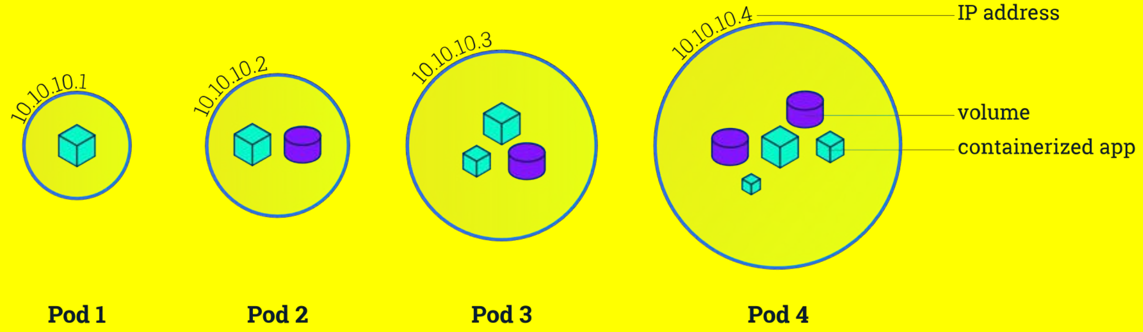
- Bin Packing (Assigning workloads to machines)
- Self Healing
- Horizontal Scaling
- Service Discovery and Load Balancing
- Secret and Configuration Management
- Storage Orchestration
- Automated Rollouts and Rollbacks
- A/B Testing

Software-defined Datacenter via Container Orchestration

Kubernetes Concepts and Terms

- Pods and Volumes
- Nodes
- Services
- Deployments
- Namespaces

Pods: Containers and Volumes



Pods are the smallest unit of compute in Kubernetes

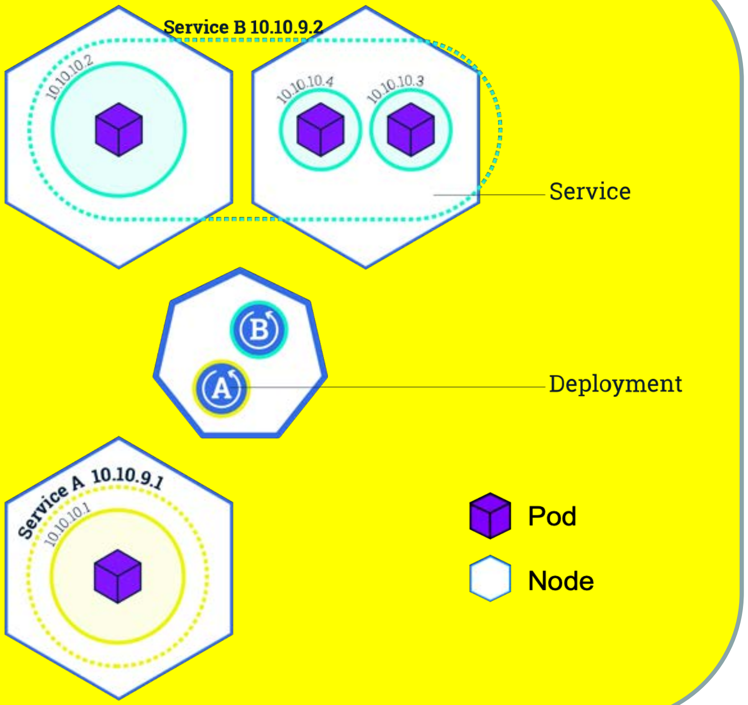
All containers in a pod share an IP address and may share the volumes defined in that pod.

Deployment: Creating and Maintaining Pods

Deployment: a container version

A deployment creates pods from the image you specify.

It maintains and scales the right number of pods, through both crashes and load increases/decreases.



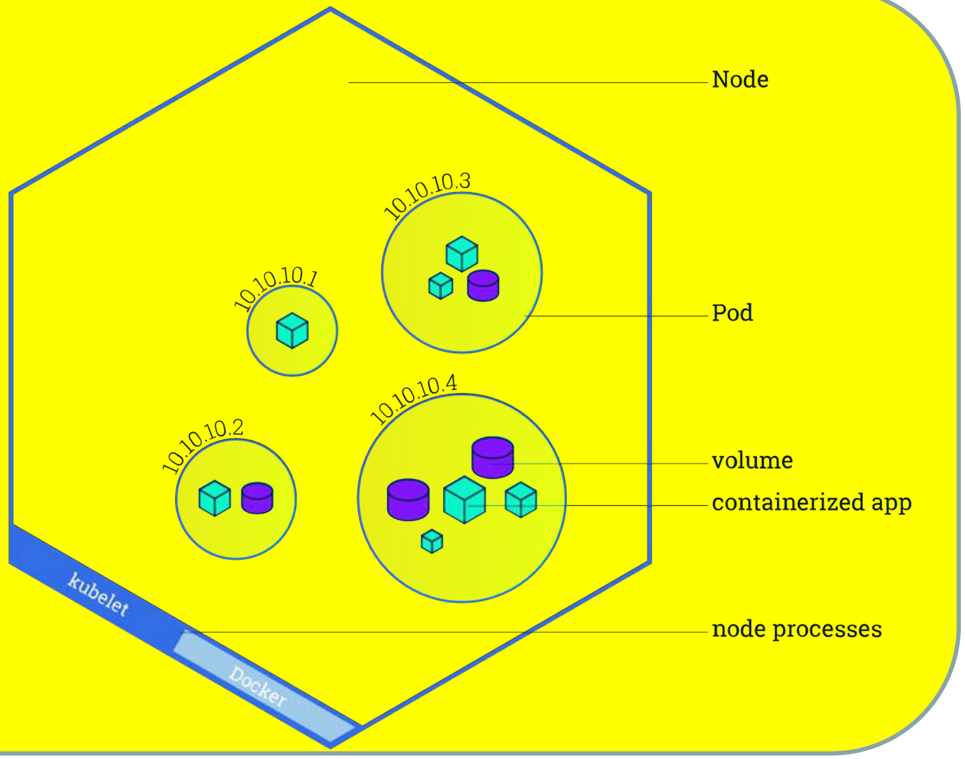
Nodes: Hosts in the Cluster

Nodes run:

- **Kubelet**
- **Container Runtime**

(Docker, containerd, ...)

- **Kube-Proxy**



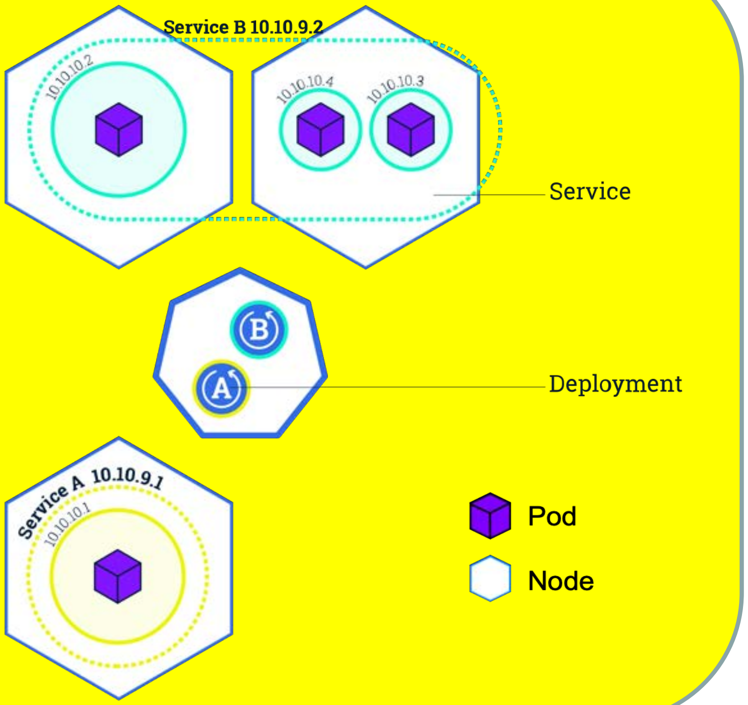
Services: Load Balancers

Service: a load balancer

A service creates:

- **a DNS name**
- **IP address**
- **port**

These redirect traffic they receive to the pods that match the labels specified by the Service's description.



Namespaces: Organize Objects

Namespace

- A logical grouping for Kubernetes objects (pods, roles, …)
- Namespaces might separate:
    - departments
    - development groups
    - companies (tenants)
- Every cluster starts with two namespaces:
    - **kube-system**: Kubernetes' default control plane components are here.
    - **default**: resources are deployed here when namespace isn't specified

Kubernetes Glossary

- Containers: Linux namespace and control group-based lightweight VMs
- Pods: collections of containers, the smallest unit of work in K8S
- Nodes: hosts on which the containers/pods run
- Services: load balancers, allowing pods to fail and scale
- Deployments: method for creating pods and handling failure and scaling
- Namespaces: logical groupings of resources, possibly by tenant, department or application

Control Loops

- Kubernetes is a "declarative" system, rather than an "imperative" one.
- You tell Kubernetes that you’d like five (5) copies of this application running.
- Kubernetes takes responsibility for keeping five containers staged, spread out to as many as five nodes, watching for container or node failures.
- You build YAML files or JSON objects describing what you want, pass these to the API server, and let it take responsibility for effecting that declaration.

```
kubectl create -f file.yaml
```

Kubernetes Target Components: API Nodes

- Kubernetes API Server
  - Accepts the declarative configurations.
  - Serves as the first point of contact.
- etcd Server
  - Retains the state of every object in the cluster.
  - Allows "is the answer different from the last time I asked" queries.
- Controller Manager
  - Runs control loops to bring the cluster's state to parity with etcd's contents
  - Contains multiple controllers, all compiled into one binary.

Kubernetes Target Components: API Nodes

- Scheduler
  - Chooses a node for each new pod, subject to constraints. (i.e., "bin packs workloads")
- Kube-DNS (or CoreDNS)
  - Gives every endpoint a DNS name, like postgres.mktg.svc.cluster.local

Vital Kubernetes Target Components: All Nodes

- Kubelet
  - Bridges the Kubernetes infrastructure to the container runtime (e.g., containerd, CRI-O, Docker,...)
- Container Runtime
  - Pulls container images and instructs the kernel to create/destroy containers, as well as other functionality.
- Kube-Proxy
  - Proxies traffic and configures iptables and ipvs.
- Pods
  - Control plane components
  - Workloads.

Attacking Kubernetes Clusters

- An attack on Kubernetes generally starts from the perspective of a compromised pod.
- The threat actor reaches this point via a scenario similar to these:
  - Actor compromised the application running in one container in the pod.
  - Actor phished/compromised a person who had access to the pod.
  - Actor was authorized and wants to escalate their privileges.

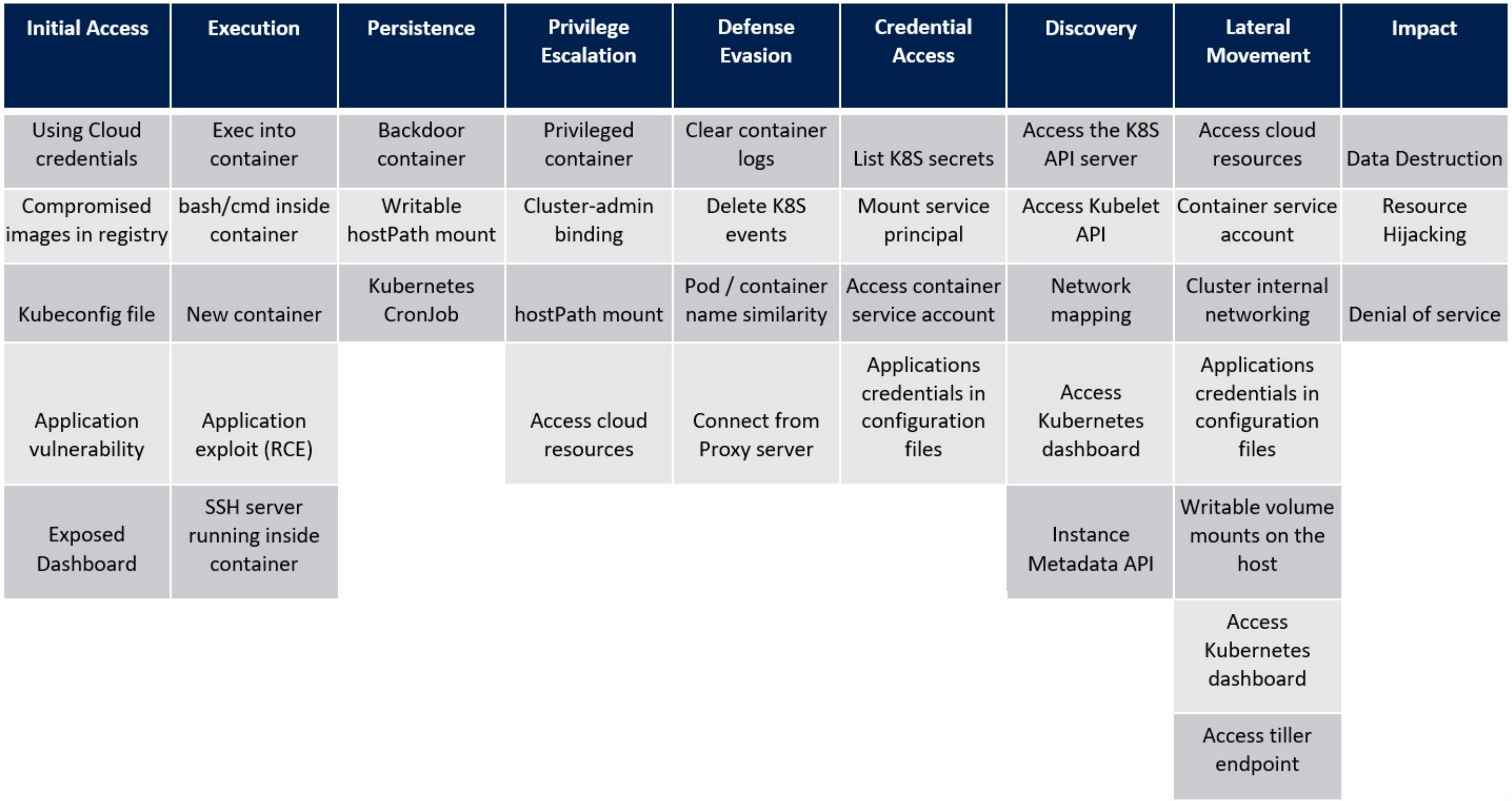
- As a defender, once you can handle the compromised pod scenario, it's time to gain the ability to handle a compromised node.
  - Nodes are compromised either directly, through phishing/social engineering attacks, or through container breakouts.

Attacks from within a Compromised Pod

An attacker in a pod may, among other things:

- Use the access provided by the pod to access other services`
- Attack other containers in their pod
- Make requests to the API server or a Kubelet to:
  - Run commands (possibly interactively) in a different pod
  - Start a new pod with privilege and node filesystem/resource access
  - Gather secrets that Kubernetes provides to pods
- Connect to the Kubernetes dashboard to perform actions
- Interact with the etcd server to change the cluster state
- Interact with the cloud service provider using a cluster account.

Microsoft's Threat Matrix for Kubernetes



Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Impact
Using Cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8S secrets	Access the K8S API server	Access cloud resources	Data Destruction
Compromised images in registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account	Resource Hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Access container service account	Network mapping	Cluster internal networking	Denial of service
Application vulnerability	Application exploit (RCE)		Access cloud resources	Connect from Proxy server	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files	
Exposed Dashboard	SSH server running inside container					Instance Metadata API	Writable volume mounts on the host	
							Access Kubernetes dashboard	
							Access tiller endpoint	

Shameless Plug

Jay teaches a class at Black Hat each year on Linux and Kubernetes security:

https://tinyurl.com/r4wwjbd5