

# Continuous Kubernetes Security

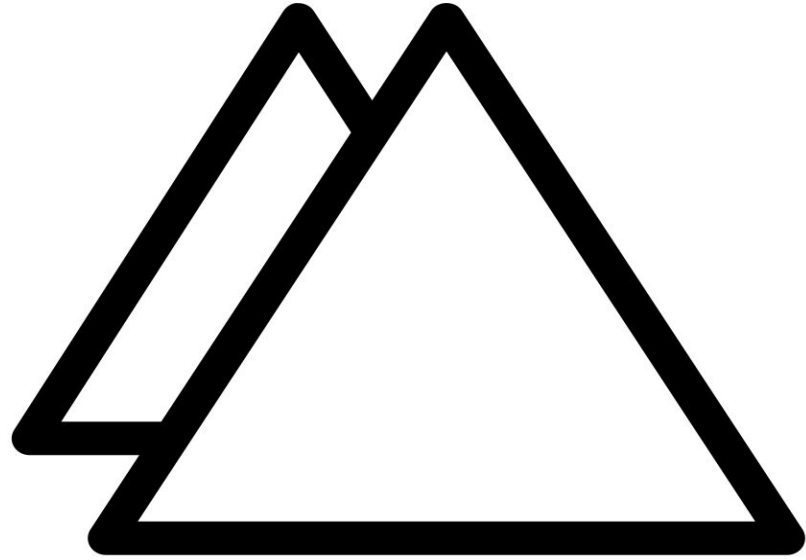
@sublimino and @controlplaneio





I'm:

- Andy
- Dev-like
- Sec-ish
- Ops-y



control plane

Is this Kubernetes cluster secure?







**LIVE**

**BREAKING NEWS**

# LOCAL FISH IN GREAT MOOD TODAY

**19:41**

THE WATER TASTES GREAT AND THERE'S NO PREDATORS AROUND AT THE MOMENT











*Lake Berryessa Is 1 Foot Over the  
Glory Hole Spillway  
& Hwy 128 Is Closed on Saturday!  
2-18-17 Lake Berryessa News Drone Report  
Napa County, California*

# How secure is Kubernetes?



# What this Kubernetes talk is about

- Common Pwns
- Hardening the Control Plane
- Securing Workloads and Networks
- Hard and Soft Multi Tenancy
- Continuous Security



# Common Pwns



# kubelet-exploit

There were discussions (<https://github.com/kubernetes/kubernetes/issues/11816>, <https://github.com/kubernetes/kubernetes/issues/3168>, <https://github.com/kubernetes/kubernetes/issues/7965>), but looks like nobody cares.

Everybody who has access to the service kubelet port (10250), even without a certificate, can execute any command inside the container.

```
# /run/%namespace%/%pod_name%/%container_name%
$ curl -k -XPOST "https://k8s-node-1:10250/run/kube-system/node-exporter-iuwg7/node-exporter" -d "cmd=ls -l
total 12
drwxr-xr-x 13 root    root          148 Aug 26 11:31 .
drwxr-xr-x 13 root    root          148 Aug 26 11:31 ..
-rwxr-xr-x  1 root    root           0 Aug 26 11:31 .dockerenv
drwxr-xr-x  2 root    root        8192 May  5 22:22 bin
drwxr-xr-x  5 root    root        380 Aug 26 11:31 dev
drwxr-xr-x  3 root    root        135 Aug 26 11:31 etc
drwxr-xr-x  2 nobody  nogroup       6 Mar 18 16:38 home
drwxr-xr-x  2 root    root         6 Apr 23 11:17 lib
dr-xr-xr-x 353 root    root         0 Aug 26 07:14 proc
drwxr-xr-x  2 root    root         6 Mar 18 16:38 root
dr-xr-xr-x 13 root    root         0 Aug 26 15:12 sys
drwxrwxrwt  2 root    root         6 Mar 18 16:38 tmp
drwxr-xr-x  4 root    root        31 Apr 23 11:17 usr
drwxr-xr-x  5 root    root        41 Aug 26 11:31 var
```

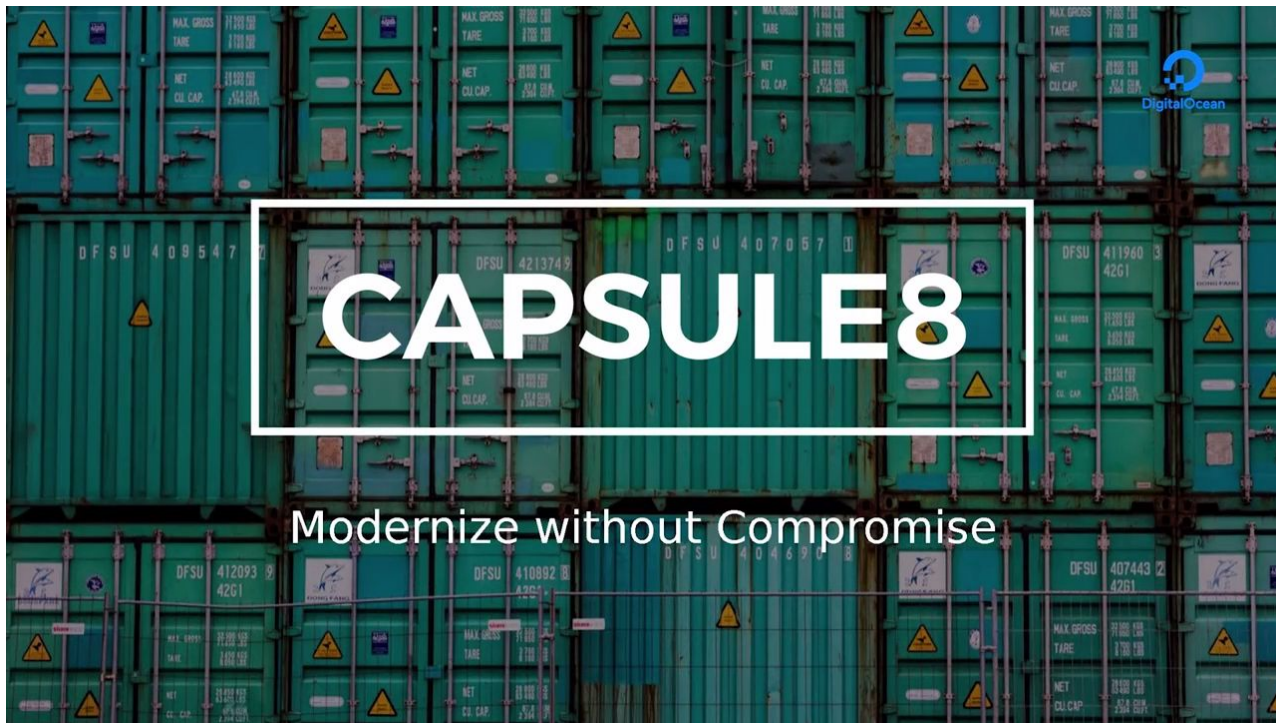
This makes namespaces/authentication and other security implementations in Kubernetes useless because by default any app inside the scheduled pod can access this port.



**Security vs Features**



# No RBAC



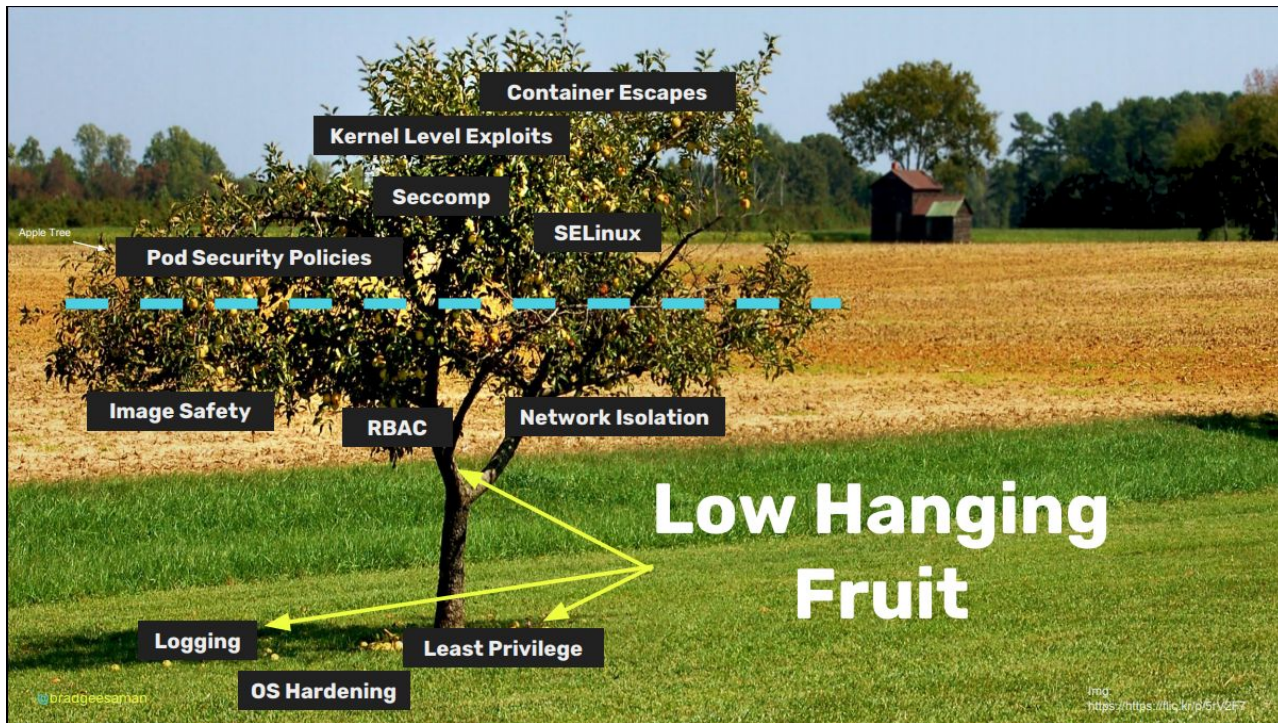
# No Workload Security



[Building for Trust: How to Secure Your Kubernetes Cluster \[I\] - Alexander Mohr & Jess Frazelle](#)



# No Security - Cluster Edition

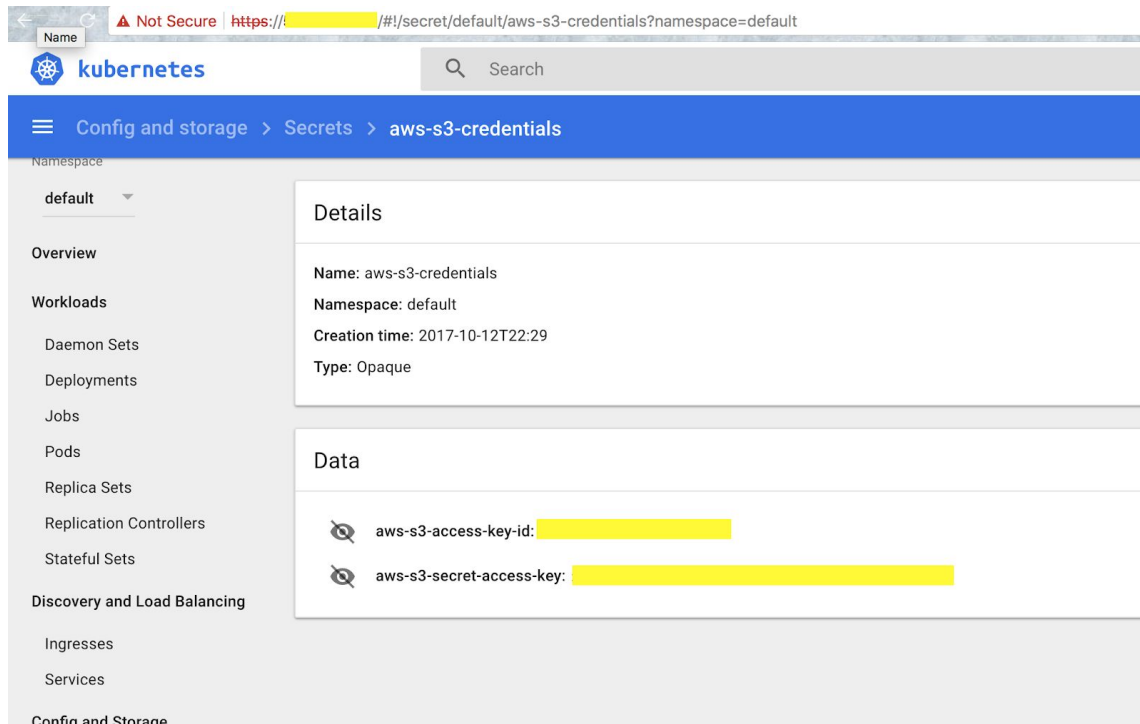




# Helm



# Unsecured Dashboard - Tesla



[Lessons from the Cryptojacking Attack at Tesla - RedLock CSI Team](#)



# CVE-2017-1002101 - subpath volume mount handling allows arbitrary file access in host filesystem #60813

New issue



liggitt opened this issue 7 days ago · 2 comments



liggitt commented 7 days ago · edited ▾

Member

CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

This vulnerability allows containers using [subpath volume mounts](#) with any volume type (including non-privileged pods, subject to file permissions) to access files/directories outside of the volume, including the host's filesystem.

Thanks to Maxim Ivanov for reporting this problem.

## Vulnerable versions:

- Kubernetes 1.3.x-1.6.x
- Kubernetes 1.7.0-1.7.13
- Kubernetes 1.8.0-1.8.8
- Kubernetes 1.9.0-1.9.3

## Vulnerable configurations:

- Clusters that allow untrusted users to control pod spec content, and prevent host filesystem access via hostPath volumes (or other volume types) using PodSecurityPolicy (or custom admission plugins)
- Clusters that make use of [subpath volume mounts](#) with untrusted containers or containers that can be compromised

## Vulnerability impact:

A specially crafted pod spec combined with malicious container behavior can allow read/write access to arbitrary files outside volumes specified in the pod, including the host's filesystem. This can be accomplished with any volume type, including emptyDir, and can be accomplished with a non-privileged pod (subject to file permissions).

## Assignees

- liggitt
- jsafrane
- msau42

## Labels

- area/security
- kind/bug
- priority/critical-urgent
- sig/storage
- status/approved-for-milestone
- status/in-progress

## Projects

None yet

## Milestone

v1.10

## 5 participants



controlplane

[CVE-2017-1002101 - subpath volume mount handling allows arbitrary file access in host filesystem #60813](#)



even with authentication enabled on Kubelet, it only applies to the HTTPS port (10250). Meaning the read-only HTTP port (10255) still stays open without any means to protect besides network ACL's.

<https://medium.com/handy-tech/analysis-of-a-kubernetes-hack-backdooring-through-kubelet-823be5c3d67c>



# What is Continuous Security?

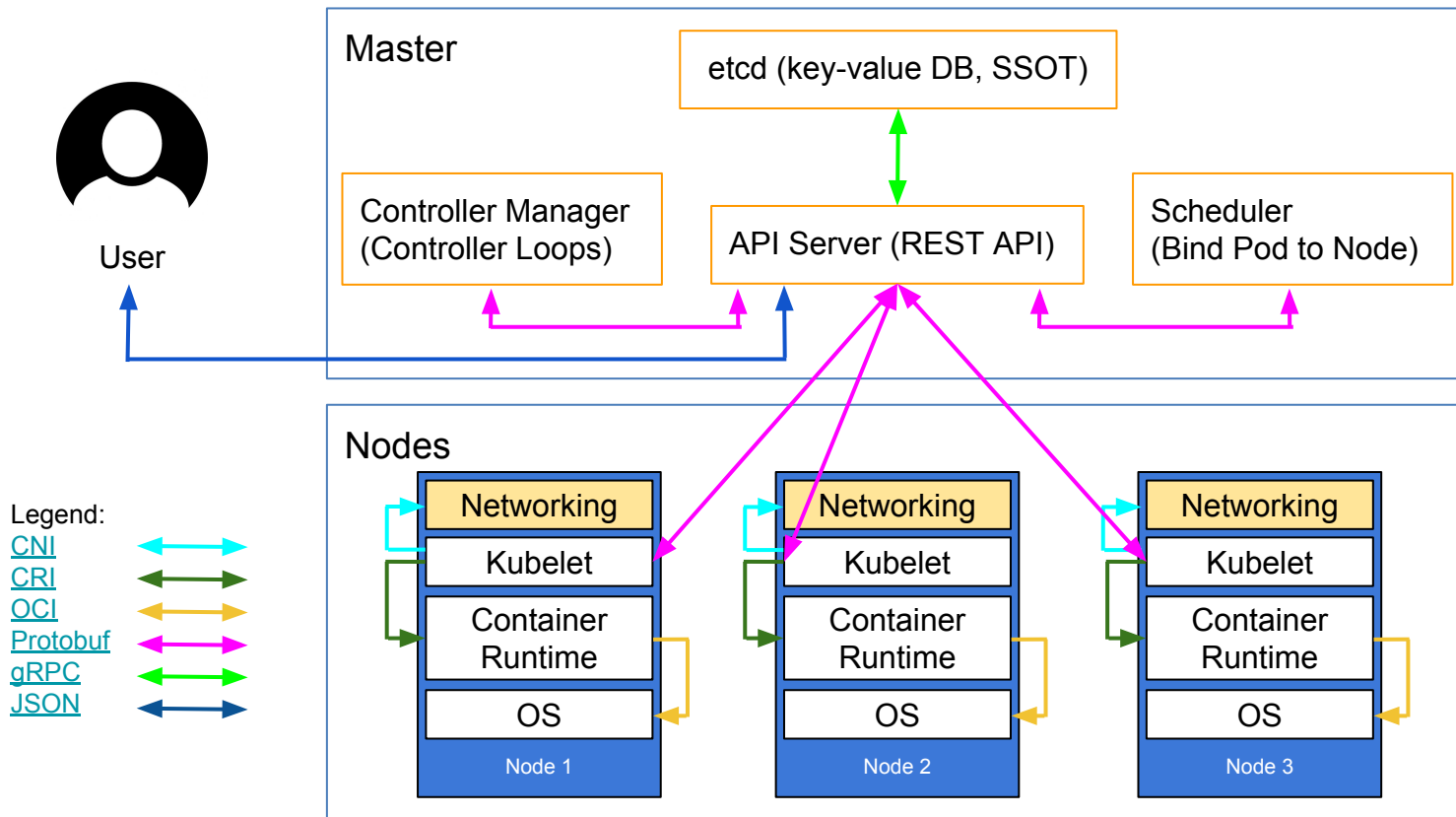
- Infrastructure as Code
- Security as Code
- Continuous Delivery





# Hardening the Kubernetes Control Plane





# Minimum Viable Security

## TLS Everywhere

Note that **some components and installation methods may enable local ports over HTTP** and administrators should familiarize themselves with the settings of each component to identify potentially unsecured traffic.

<https://kubernetes.io/docs/tasks/administer-cluster/securing-a-cluster/#use-transport-level-security-tls-for-all-api-traffic>





# Bootstrapping TLS

## Kubernetes the Hard Way

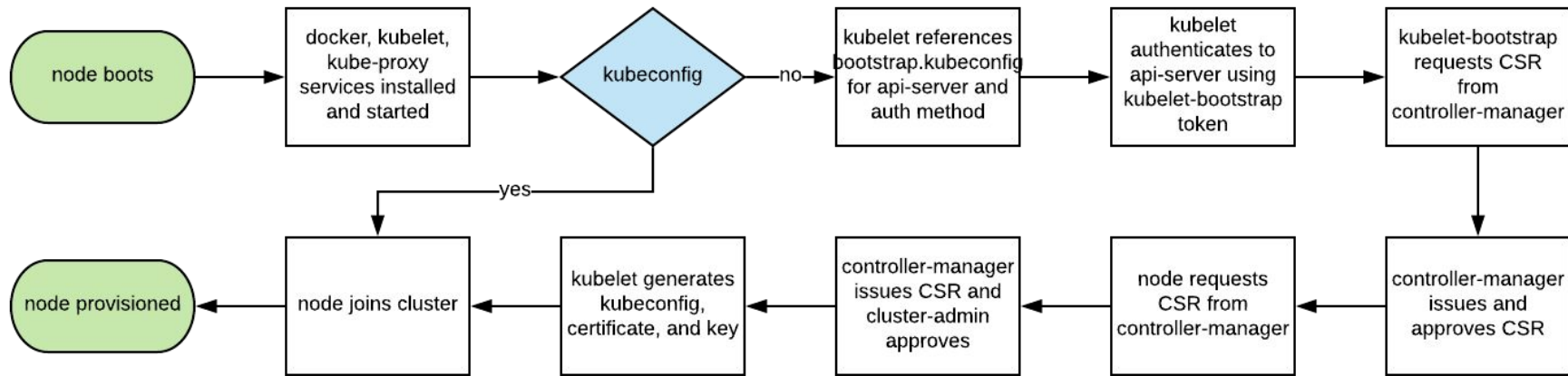
- <https://github.com/kelseyhightower/kubernetes-the-hard-way/blob/master/docs/04-certificate-authority.md>

## Kubelet TLS Bootstrap (still beta, stable v1.11?)

- <https://kubernetes.io/docs/admin/kubelet-tls-bootstrapping/>
- <https://github.com/kubernetes/features/issues/43>



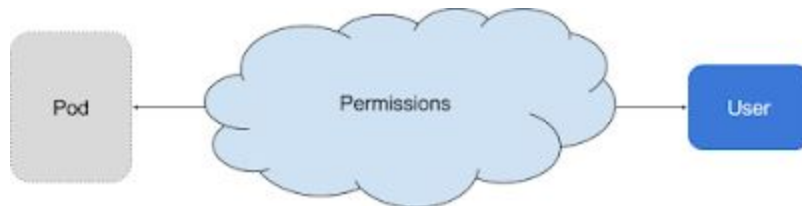
# Bootstrapping TLS



<https://medium.com/@todrosner/kubernetes-tls-bootstrapping-cf203776abc7>

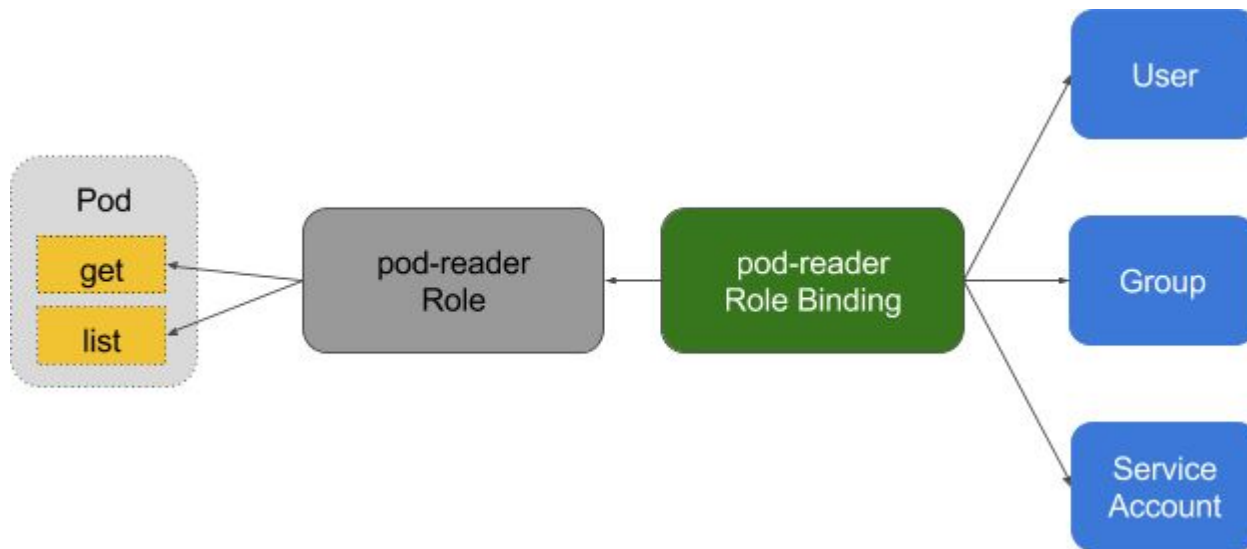


# Enable RBAC



[RBAC Support in Kubernetes](#) (stable v1.8)

# Enable RBAC



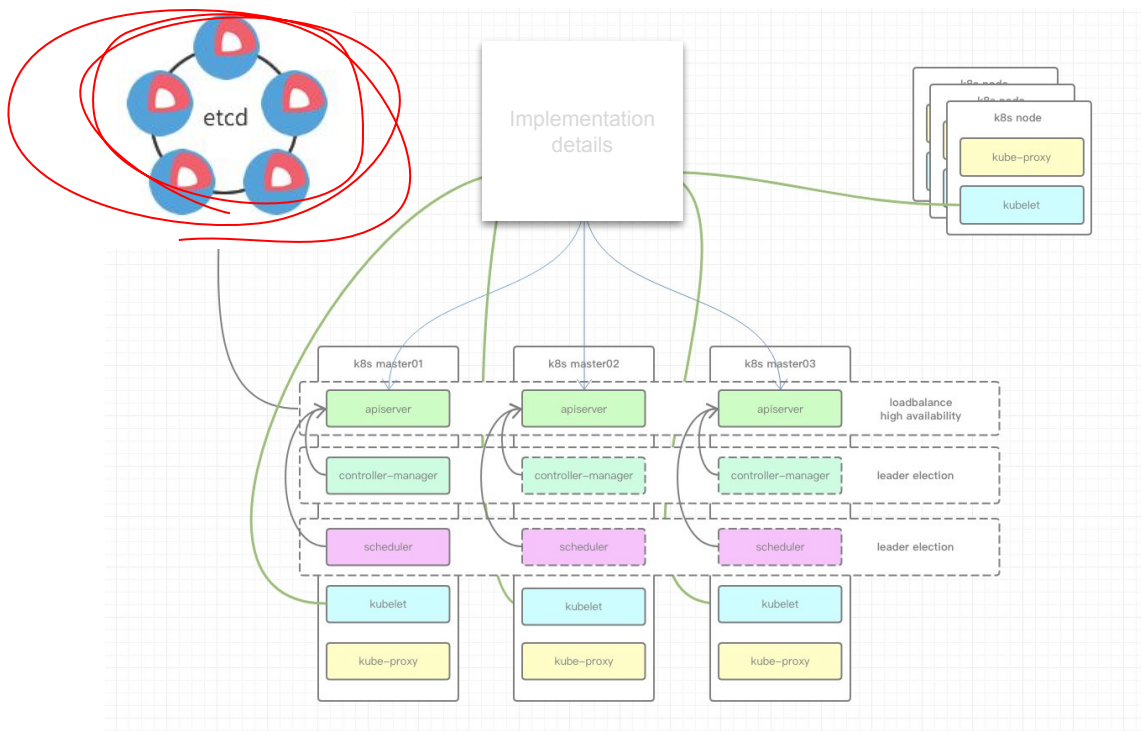
[RBAC Support in Kubernetes](#) (stable v1.8)

# External Auth to API Server (e.g. via kubectl)

- <https://thenewstack.io/kubernetes-single-sign-on-less-identity/>
- <https://github.com/coreos/dex> - OpenID Connect Identity (OIDC) and OAuth 2.0 provider with pluggable connectors
- <https://github.com/negz/kuberos> - OIDC authentication helper for kubectl (also <https://cloud.google.com/community/tutorials/kubernetes-auth-openid-rbac>)
- <https://github.com/micahhausler/k8s-oidc-helper> - helper tool for authenticating to Kubernetes using Google's OpenID Connect



# Separate, Firewalled etcd Cluster





# Disable legacy authorization

```
--no-enable-legacy-authorization
```



# Rotate keys



# Securing Workloads



# Containers



# Containers

- Namespaces
- cgroups
- seccomp-bpf
- AppArmor / SELinux
- Users
- Capabilities



# Pods





# Pods

```
apiVersion: v1
kind: Pod
metadata:
  name: nfs-server
  labels:
    role: nfs-server
spec:
  containers:
    - name: nfs-server
      image: jsafrane/nfs-data
      securityContext:
        privileged: true
```



# kubesecc.io - risk score for K8S YAML



# index

```
.metadata .annotations
'container.seccomp.security.alpha.kubernetes.io/pod'
.metadata .annotations
'seccomp.security.alpha.kubernetes.io/pod'
.spec.template.spec.hostIPC
.spec.template.spec.hostNetwork
.spec.template.spec.hostPID
Service Accounts
containers[] .resources.limits.cpu
containers[] .resources.limits.memory
containers[] .resources.requests.cpu
containers[] .resources.requests.memory
containers[] .securityContext.capabilities.add | index("SYS_ADMIN")
containers[] .securityContext.capabilities.drop | index("ALL")
containers[] .securityContext.privileged == true
containers[] .securityContext.readOnlyRootFilesystem == true
containers[] .securityContext.runAsNonRoot == true
```

kubesecc.io > index

[Edit this page](#)

## KUBESEC.IO

- .metadata .annotations "container.seccomp.security.alpha.kubernetes.io/pod"
- .metadata .annotations "seccomp.security.alpha.kubernetes.io/pod"
- .spec.template.spec.hostIPC
- .spec.template.spec.hostNetwork
- .spec.template.spec.hostPID
- Service Accounts
- containers[] .resources.limits.cpu
- containers[] .resources.limits.memory
- containers[] .resources.requests.cpu
- containers[] .resources.requests.memory
- containers[] .securityContext.capabilities.add | index("SYS\_ADMIN")
- containers[] .securityContext.capabilities.drop | index("ALL")
- containers[] .securityContext.privileged == true
- containers[] .securityContext.readOnlyRootFilesystem == true
- containers[] .securityContext.runAsNonRoot == true
- containers[] .securityContext.runAsUser > 10000
- securityContext.capabilities
- .metadata .annotations "container.apparmor.security.beta.kubernetes.io/nginx"
- .spec.volumeClaimTemplates[] .spec.accessModes | index("ReadWriteOnce")
- .spec.volumeClaimTemplates[] .spec.resources.requests.storage

## FUTHER READING

- <http://blog.kubernetes.io/2016/08/security-best-practices-kubernetes-deployment.html>



# kubesecc.io - example insecure pod

```
{
  "score": -30,
  "scoring": {
    "critical": [{
      "selector": "containers[] .securityContext .privileged == true",
      "reason": "Privileged containers can allow almost completely unrestricted host access"
    }],
    "advise": [{
      "selector": "containers[] .securityContext .runAsNonRoot == true",
      "reason": "Force the running image to run as a non-root user to ensure least privilege"
    }, {
      "selector": "containers[] .securityContext .capabilities .drop",
      "reason": "Reducing kernel capabilities available to a container limits its attack surface",
      "href": "https://kubernetes.io/docs/tasks/configure-pod-container/security-context/"
    }],
    ...
  }
}
```

# PodSecurityPolicies

```
apiVersion: extensions/v1beta1
kind: PodSecurityPolicy
metadata:
  name: restricted
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: 'docker/default'
    apparmor.security.beta.kubernetes.io/allowedProfileNames: 'runtime/default'
    seccomp.security.alpha.kubernetes.io/defaultProfileName: 'docker/default'
    apparmor.security.beta.kubernetes.io/defaultProfileName: 'runtime/default'
spec:
  privileged: false
  allowPrivilegeEscalation: false # Required to prevent escalations to root.
  # 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'
    ...
  hostNetwork: false
  hostIPC: false
  hostPID: false
  runAsUser:
    rule: 'MustRunAsNonRoot' # Require the container to run without root privileges.
  ...
```



controlplane

# Resource Linting

- <https://kubesec.io/> - calculate “risk” of Kubernetes resource YAML by use of security features
- <https://github.com/garethr/kubetest> - unit tests for your Kubernetes configurations



# Deployments







**thockin** @thockin · Mar 1

Omega turned out to be a bunch of useful technologies that were back-integrated into Borg, rather than a full reboot.



**Joe Beda** @jbeda · Mar 1

It heavily influenced k8s too. For instance, the way that labels work to relate thing is based on a lot of ideas that were designed/prototyped/explored with both Omega and GCP.



**Brian Grant**

@bgrant0607

Following

Replying to @jbeda @thockin and 3 others

Yes, labels, pods, watch, async. controllers, to name a few things.

1:11 AM - 2 Mar 2018



controlplane

# Services



# Services

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 443
      targetPort: 8443
```



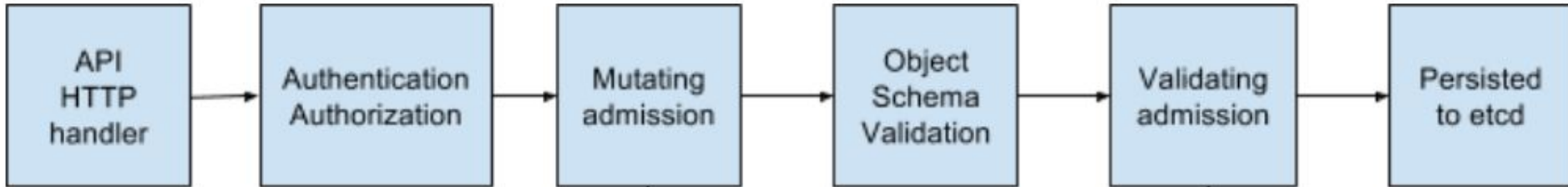
# ServiceAccounts

“We recommend you create and use a minimally privileged service account to run your Kubernetes Engine Cluster”

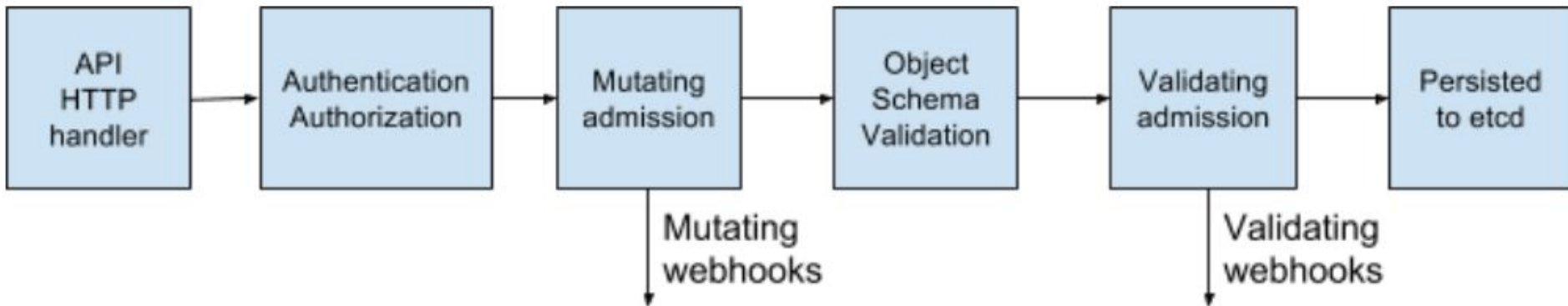
<https://cloudplatform.googleblog.com/2017/11/precious-cargo-securing-containers-with-Kubernetes-Engine-18.html>



# API Admission Controllers



# Extensible Admission Controllers



<http://blog.kubernetes.io/2018/01/extensible-admission-is-beta.html>





# Docs: Recommended Admission Controllers

```
--admission-control=${CONTROLLERS}
```

*# ORDER MATTERS. For versions >= v1.9.0*

- NamespaceLifecycle
- LimitRanger
- ServiceAccount
- PersistentVolumeLabel
- DefaultStorageClass
- DefaultTolerationSeconds
- MutatingAdmissionWebhook
- ValidatingAdmissionWebhook
- ResourceQuota

<https://kubernetes.io/docs/admin/admission-controllers/#is-there-a-recommended-set-of-admission-controllers-to-use>



# Admission Controllers: ImagePolicyWebhook

allows a **backend webhook** to make admission decisions



# Admission Controllers: DenyEscalatingExec

**deny exec and attach** commands to  
pods that run with **escalated privileges**  
that allow host access  
(privileged, access to host IPC/PID namespaces)



# Admission Controllers: LimitRanger

observe the incoming request and ensure that it does not  
**violate any of the LimitRange constraints**



# Admission Controllers: ResourceQuota

observe the incoming request and ensure that it does not  
**violate any of the ResourceQuota constraints**



# Admission Controllers: NodeRestriction

**limits** the Node and Pod **objects** a kubelet can modify

kubelets must use credentials in the `system:nodes` group,

with a username in the form `system:node:<nodeName>`

**n.b. Node Authorizer authorization mode required**

<https://kubernetes.io/docs/admin/authorization/node/>



```
$ kubectl describe clusterrole system:node
Name:         system:node
Labels:       kubernetes.io/bootstrapping=rbac-defaults
Annotations:  rbac.authorization.kubernetes.io/autoupdate=true
PolicyRule:
  Resources                                Non-Resource URLs  Resource Names      Verbs
  -----                                -
certificatesigningrequests.certificates.k8s.io  []                []                  [create get list watch]
configmaps                                     []                []                  [get]
endpoints                                     []                []                  [get]
events                                         []                []                  [create patch update]
localsubjectaccessreviews.authorization.k8s.io  []                []                  [create]
nodes                                          []                []                  [create get list watch delete patch update]
nodes/status                                  []                []                  [patch update]
persistentvolumeclaims                       []                []                  [get]
persistentvolumes                           []                []                  [get]
pods                                           []                []                  [get list watch create delete]
pods/eviction                                []                []                  [create]
pods/status                                  []                []                  [update]
secrets                                       []                []                  [get]
services                                     []                []                  [get list watch]
subjectaccessreviews.authorization.k8s.io      []                []                  [create]
tokenreviews.authentication.k8s.io             []                []                  [create]
```

clusterrole system:node



# Admission Controllers: NodeRestriction

`--authorization-mode=Node`

A kubelet can not:

- alter the state of resources of any Pod it does not manage
- access Secrets, ConfigMaps or Persistent Volumes / PVCs, unless they are bound to a Pod managed by itself
- alter the state of any Node but the one it is running on

<https://kubernetes.io/docs/admin/authorization/node/>





# Admission Controllers: PodSecurityPolicy

determines if it should be admitted based on the **requested security context** and available Pod Security Policies

<https://github.com/kubernetes/examples/tree/master/staging/podsecuritypolicy/rbac>



# Admission Controllers: ServiceAccount

automation for serviceAccounts

if not exist, set:

ServiceAccount, ImagePullSecrets,  
/var/run/secrets/kubernetes.io/serviceaccount volume



# Admission Controllers in GKE



# Admission Controllers:

## ValidatingAdmissionWebhook (v1.9 beta)

calls validating webhooks in parallel,

rejects pod if any fail



# Admission Controllers:

## ValidatingAdmissionWebhook (v1.9 beta)

<https://github.com/kelseyhightower/denyenv-validating-admission-webhook#validating-admission-webhook-configuration>

<https://github.com/openshift/generic-admission-server>



# Secrets and Configmaps

--experimental-encryption-provider-config

- Secrets and configmaps are encrypted at rest with 'aescbc'
  - If 'aesgcm' encryption is used, encryption keys should be rotated frequently
- Secure connection is set between apiserver and etcd
- Only apiserver user can read / edit EncryptionConfig file

<https://www.twistlock.com/2017/08/02/kubernetes-secrets-encryption/>



# Secrets and Configmaps

- <https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/>
- Secure Secret management for Kubernetes (with gpg, Google Cloud KMS and AWS KMS backends) - <https://github.com/shyiko/kubesecc>
- Encryption at rest KMS integration - <https://github.com/kubernetes/features/issues/460>
- <https://medium.com/@mtreacher/using-aws-kms-for-application-secrets-in-kubernetes-149ffb6b4073>
- Sealed Secrets - a Kubernetes controller and tool for one-way encrypted Secrets <https://github.com/bitnami-labs/sealed-secrets>



# TokenRequest API (v1.10 alpha)

The TokenRequest API enables creation of tokens that:

- aren't persisted in the Secrets API
- targeted for specific audiences (such as external secret stores)
- have configurable expiries
- bindable to specific pods.





# Compliance Scanning

- <https://github.com/nccgroup/kube-auto-analyzer> - review Kubernetes installations against the CIS Kubernetes 1.8 Benchmark
- <https://github.com/aquasecurity/kube-bench> - test versions of Kubernetes (1.6, 1.7 and 1.8) against CIS Kubernetes 1.0.0, 1.1.0 and 1.2.0
- <https://github.com/heptio/sonobuoy> - running a set of Kubernetes conformance tests in an accessible and non-destructive manner
- <https://github.com/bgeesaman/sonobuoy-plugin-bulkhead> - kube-bench for sonobuoy
- <https://github.com/bgeesaman/kubeatf> - spin up, test, and destroy Kubernetes clusters in a human and CI/CD friendly way

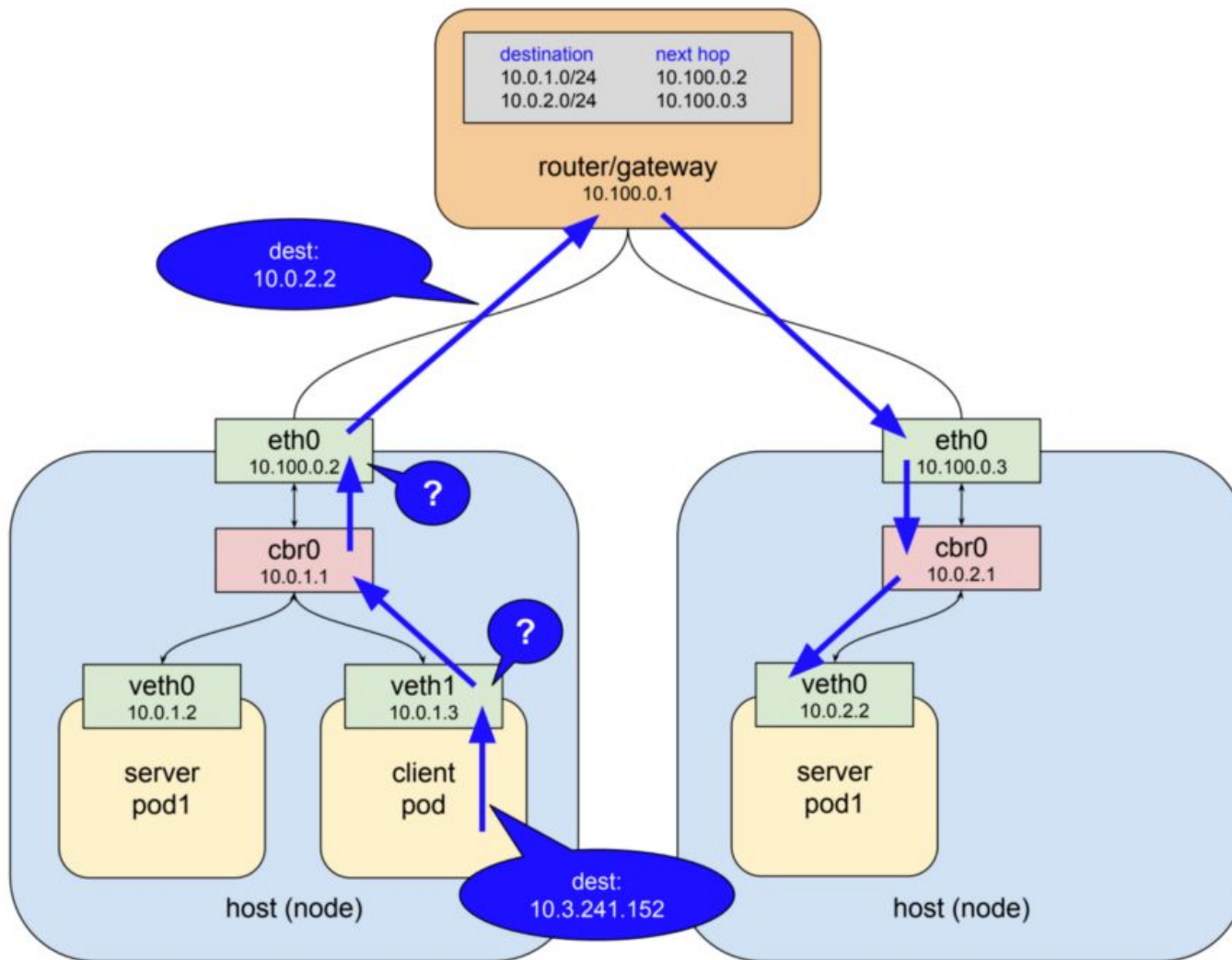


# Image Scanning

- <https://github.com/coreos/clair>
- <https://github.com/arminc/clair-local-scan>
- <https://github.com/optiopay/klar> - integration of Clair and Docker Registry
- <https://github.com/banyanops/collector>
- <https://github.com/anchore/anchore-engine>

# Securing Kubernetes Networking



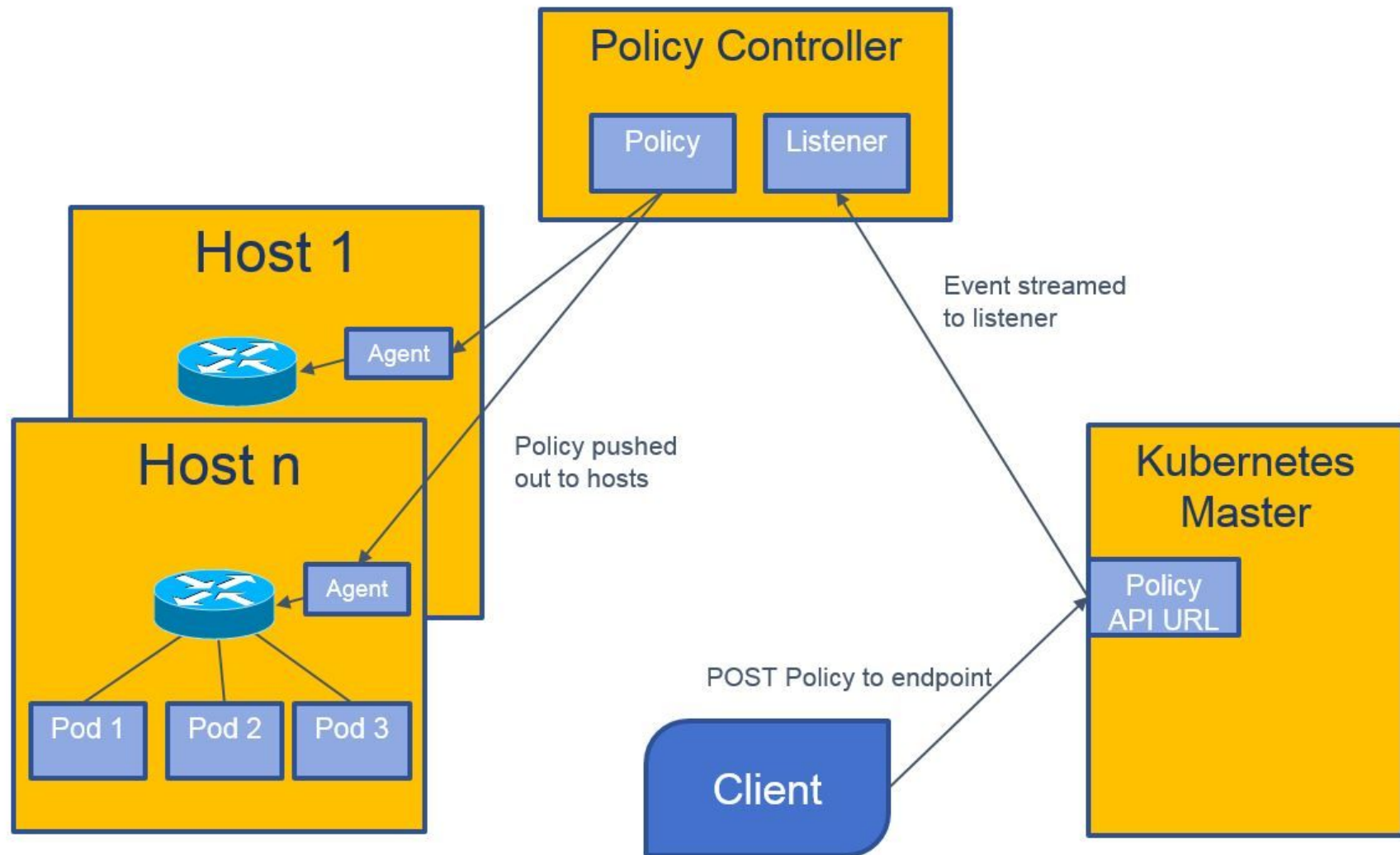


<https://medium.com/google-cloud/understanding-kubernetes-networking-services-f0cb48e4cc82>



# NetworkPolicy

- [Calico](#)
- [Cilium](#) ([Learn more about eBPF](#))
- [Kube-router](#)
- [Romana](#)
- [Weave Net](#)



```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector:
```

<https://github.com/ahmetb/kubernetes-network-policy-recipes>

Kubernetes NetworkPolicy: default deny



```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector:
    - "*"

```

Illegal syntax, but  
represents what it  
actually does  
(effectively a wildcard)

<https://github.com/ahmetb/kubernet-network-policy-recipes>

Kubernetes NetworkPolicy: default deny





```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: foo-deny-external-egress
spec:
  podSelector:
    matchLabels:
      app: foo
  policyTypes:
  - Egress
  egress:
  - ports:
    - port: 53
      protocol: UDP
    - port: 53
      protocol: TCP
  - to:
    - namespaceSelector: {}
```

<https://github.com/ahmetb/kubernetesis-network-policy-recipes>

Kubernetes NetworkPolicy





**thockin** (Tim Hockin) 27 days ago <>

Owner



I really don't think we want to impose DNS refreshing on implementations of NetworkPolicy without a bunch of REALLY REALLY good use cases that just CAN NOT be solved any other way. Do we have such use cases?



**thockin** (Tim Hockin) closed this 27 days ago

<https://github.com/kubernetes/kubernetes/issues/56901>

# Kubernetes NetworkPolicy - NO DNS NAMES



```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: foo-deny-external-egress
spec:
  podSelector:
    dnsName: control-plane.io
  policyTypes:
  - Egress
  egress:
  - ports:
    - port: 53
      protocol: UDP
    - port: 53
      protocol: TCP
  - to:
    - namespaceSelector: {}
```

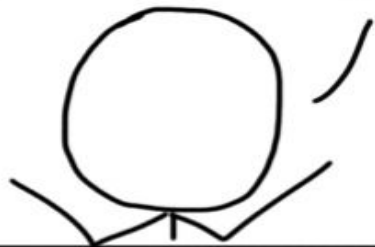
ILLEGAL! NOT ALLOWED!

<https://github.com/ahmetb/kubernetes-network-policy-recipes>

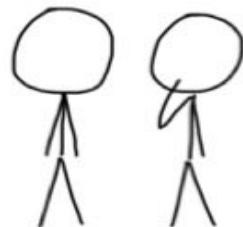
Kubernetes NetworkPolicy - ILLEGAL!



SERVICE MESH!  
SERVICE MESH!

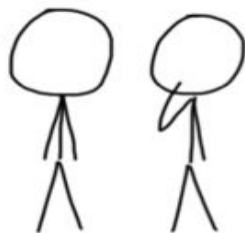


SERVICE MESH!  
SERVICE MESH!



SERVICE MESH!  
SERVICE MESH!  
SERVICE MESH!

HE'S BROKEN.  
NOT THIS AGAIN.  
PUT HIM WITH THE REST.



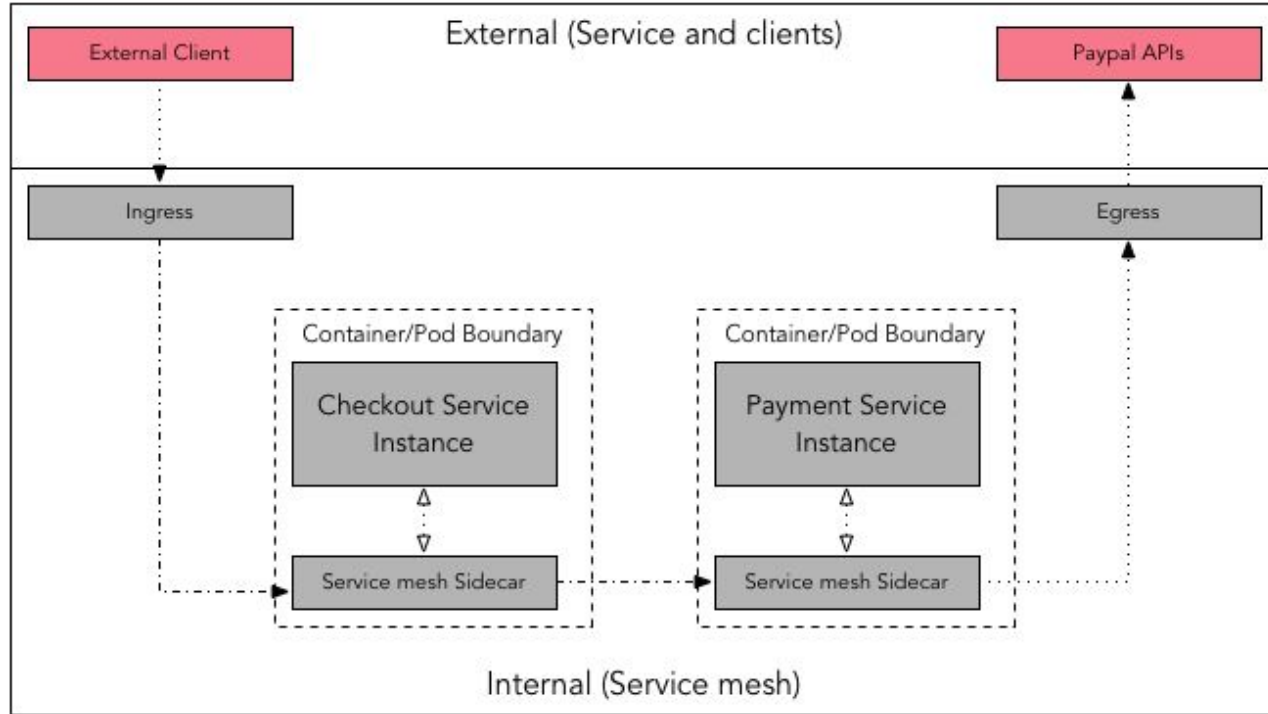
SERVICE MESH?

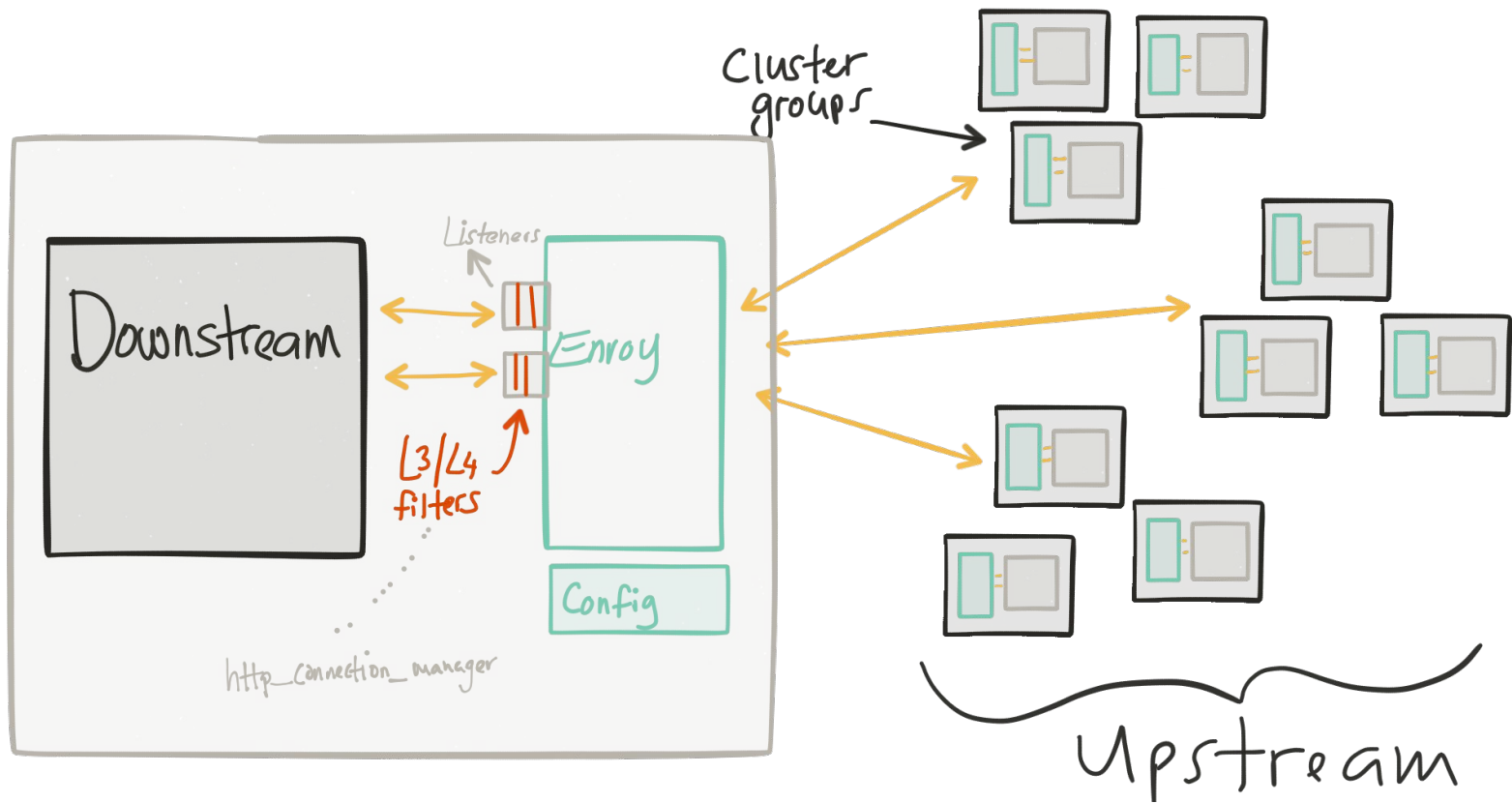


SERVERLESS!  
BLOCKCHAIN!  
INFRA AS CODE!

@sebiwicb

# What is a Service Mesh?





<http://blog.christianposta.com/istio-workshop/>

# Service Meshes - Istio

- Automatic mutual TLS between services
- Service-level RBAC
- External identity provider integration
- Policy and quota enforcement, dynamic per-request routing
- Deployment strategies such as red/black, canary, dark/mirrored
- Distributed tracing
- Network policy between apps/services, and on ingress/egress



```
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: reviews-test-v2
  namespace: default
  ...
spec:
  destination:
    name: reviews
  match:
    request:
      headers:
        cookie:
          regex: ^(..*?;)?(user=jason)(;.*)?$
  precedence: 2
  route:
    - labels:
        version: v2
```

<https://istio.io/docs/tasks/traffic-management/request-routing.html>

Istio networking





# netassert - cloud native network testing

- netassert - network security testing for DevSecOps workflows  
<https://github.com/controlplaneio/netassert>

```
host:
  localhost:
    bitbucket.com:
      - 22
  control-plane.io:
    github.com:
      - 22
```



# netassert - cloud native network testing

```
k8s: # used for Kubernetes pods
deployment: # only deployments currently supported
  test-frontend: # pod name, defaults to `default` namespace
    test-microservice: 80 # `test-microservice` is the DNS name of the target service
    test-database: -80    # should not be able to access port 80 of `test-database`

new-namespace:test-microservice: # `new-namespace` is the namespace name
  test-database.new-namespace: 80 # longer DNS names can be used for other namespaces
  test-frontend.default: 80

default:test-database:
  test-frontend.default.svc.cluster.local: 80 # full DNS names can be used
  test-microservice.default.svc.cluster.local: -80
  control-plane.io: 443 # we can check remote services too
```

<https://github.com/controlplaneio/netassert>



```
[2018-02-02T16:06:49.124+0000] ./netassert: Results: localhost
TAP version 13
# localhost TCP:30731 closed
ok 1 - localhost TCP:30731 closed
# localhost UDP:1234 closed
ok 2 - localhost UDP:1234 closed
# localhost TCP:22 open
ok 3 - localhost TCP:22 open
# binarysludge.com TCP:443 open
ok 4 - binarysludge.com TCP:443 open
# localhost TCP:999 closed
ok 5 - localhost TCP:999 closed
# control-plane.io TCP:443 open
ok 6 - control-plane.io TCP:443 open
# localhost UDP:555 closed
ok 7 - localhost UDP:555 closed
# control-plane.io TCP:80 open
ok 8 - control-plane.io TCP:80 open
# binarysludge.com TCP:22 open
ok 9 - binarysludge.com TCP:22 open
# binarysludge.com TCP:80 open
ok 10 - binarysludge.com TCP:80 open
# 8.8.8.8 UDP:53 open
ok 11 - 8.8.8.8 UDP:53 open
# google.co.uk TCP:443 open
ok 12 - google.co.uk TCP:443 open
# binarysludge.com TCP:81 open
ok 13 - binarysludge.com TCP:81 open
# 8.8.4.4 UDP:53 open
ok 14 - 8.8.4.4 UDP:53 open

1..14
# tests 14
# pass 14
# fail 0

[2018-02-02T16:06:49.129+0000] ./netassert: localhost pass
```



controlplane

# Cloud Native Dynamic Firewalls

- Network Policy recipes - <https://github.com/ahmetb/kubernetes-network-policy-recipes>
- WeaveNet Network Policy - <https://kubernetes.io/docs/tasks/administer-cluster/weave-network-policy/>
- NeuVector Container Firewall - <https://neuvector.com/products/>
- Tesla Compromise mitigation - <https://www.tigera.io/tesla-compromise-network-policy/>



# Recap



# Recap

- Hardened control plane
- Secured application runtimes
- Locked down network traffic
- ....so with all those things configured, we can consider multitenancy



# Multi Tenancy Principles



# Secure Hosts

- Minimal attack surface
  - CoreOS (RIP), forked as FlatCar Linux- <https://coreos.com/> and <https://kinvolk.io/>
  - Red Hat Atomic - <https://www.redhat.com/en/resources/enterprise-linux-atomic-host-datasheet>
  - Ubuntu Core - <https://www.ubuntu.com/core>
  - Container-Optimized OS from Google - <https://cloud.google.com/container-optimized-os/docs/>
- Security extensions enabled, configured, and monitored
- Immutable infrastructure
- Group nodes by type, usage, and security level





# No Routes To:

- cadvisor
- heapster
- kubelet
- kubernetes dashboard
- etcd

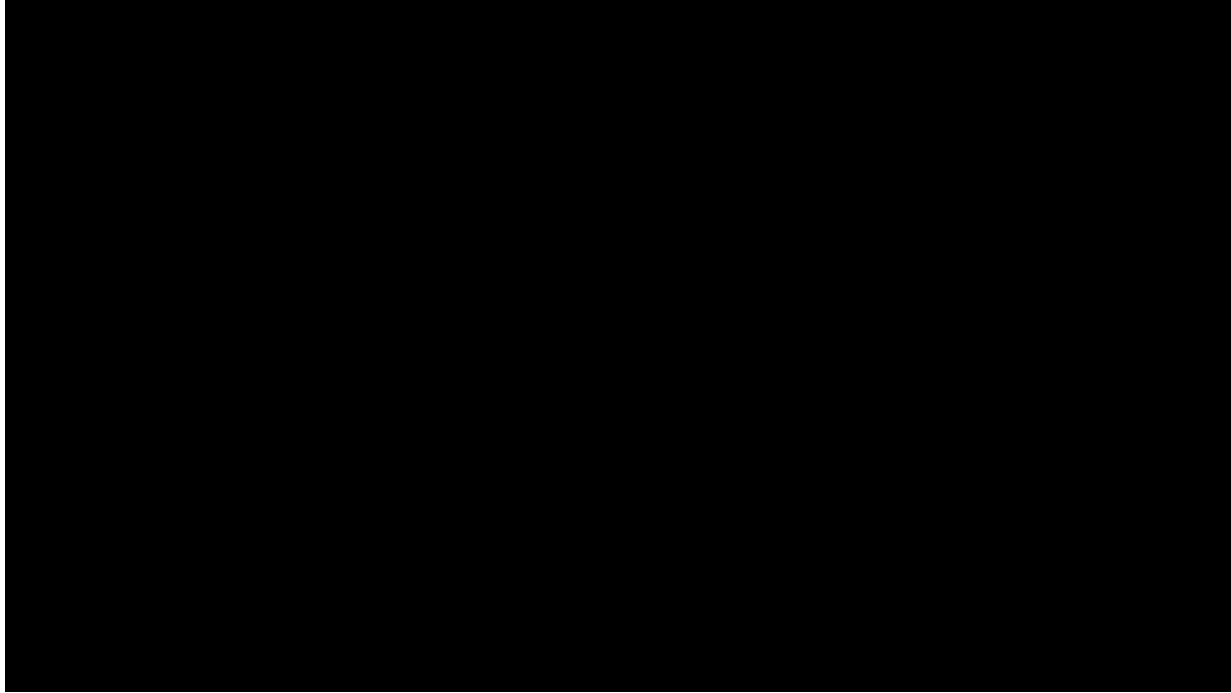


# Proxy to Metadata APIs

- <https://github.com/jtblin/kube2iam> - provides different AWS IAM roles for pods running on Kubernetes
- <https://github.com/uswitch/kiam> - allows cluster users to associate IAM roles to Pods
- <https://github.com/heptio/authenticator> - allow AWS IAM credentials to authenticate to a Kubernetes cluster
- <https://github.com/GoogleCloudPlatform/k8s-metadata-proxy> - a simple proxy for serving concealed metadata to container workloads



# MULTI TENANCY: Soft



# MULTI TENANCY: Soft

- Isolate by namespace
  - don't forget the default networkpolicy and podsecuritypolicy
  - assign limits to the namespace with LimitRanges  
<https://kubernetes.io/docs/tasks/administer-cluster/memory-default-namespace/>
- Separate dev/test from production
- Image scanning
  - private registry and build artefacts/supply chain



# MULTI TENANCY: Soft

- Policed, scanned, compliant base images
  - minimal attack surface
  - FROM scratch if possible
- Deploy admission controllers, pod security policies, etc
- Everything as code
  - <https://www.weave.works/blog/gitops-operations-by-pull-request>



# MULTI TENANCY: Hard



# MULTI TENANCY: Hard

- All users untrusted, potentially malicious
  - comfortable running code from multiple third parties, with the potential for malice that implies, in the same cluster
- Only co-tenant along your existing security boundaries
- Segregate logically by application type, security level, and/or physically by project/account
- Separate node pools for different tenants



# Container Runtimes

- **runc** - CLI tool for spawning and running containers according to the OCI specification <https://github.com/opencontainers/runc>
- **cri-o** - Open Container Initiative-based implementation of Kubernetes Container Runtime Interface <https://github.com/kubernetes-incubator/cri-o>
- **Kata Containers** - hardware virtualized containers <https://katacontainers.io/>
- **VirtualKubelet** - a Kubernetes kubelet implementation <https://github.com/virtual-kubelet/virtual-kubelet>
- **LXC/LXD, rkt, systemd-nspawn** - <https://coreos.com/rkt/docs/latest/rkt-vs-other-projects.html>





# MULTI TENANCY: Hard

- this may not look a lot like hard multitenancy?
  - it's still running a centralised control plane
- run kubedns in a sidecar to restrict DNS leakage
- mixed vm and container workload
  - Dan Walsh nailed it
  - "glasshouse VMs"
- Defence in depth
- Remote logging



# MULTI TENANCY: Hard

**TREAT CONTAINER SERVICES JUST LIKE  
REGULAR SERVICES**

Drop privileges as quickly as possible

Run your services as non Root whenever possible

Treat root within a container the same as root outside of the  
container

**"Docker is about running random crap from the internet as  
root on your host"**

Only run containers from trusted parties

<https://www.weave.works/blog/container-security-with-dan-walsh-redhat>





IDS: Not a problem while undetected

# IDS Vendors

- <https://www.twistlock.com/>
- <https://www.aquasec.com/>
- <https://www.blackducksoftware.com/>
- <https://github.com/capsule8/capsule8>
- <https://sysdig.com/>

# RBAC

- <https://github.com/uruddarraju/kubernetes-rbac-policies> - RBAC policies for cluster services
- <https://github.com/liggitt/audit2rbac> - autogenerate RBAC policies based on Kubernetes audit logs



# Audit Logs in GKE

```
{
  insertId: "1yr52hqdv1hr"
  labels: {...}
  logName: "projects/dev/logs/cloudaudit.googleapis.com%2Factivity"
  operation: {...}
  protoPayload: {...}
  receiveTimestamp: "2018-03-12T20:45:04.497610612Z"
  resource: {...}
  severity: "NOTICE"
  timestamp: "2018-03-12T20:44:45.213721Z"
}
```



```
kubectl delete configmap myconfig --ignore-not-found
failed, retrying...
```

```
audit2rbac.liggitt.net/user: system-serviceaccount-ns1-sa1
name: audit2rbac:sa1
namespace: ns1
rules:
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - get
  - list
  - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    audit2rbac.liggitt.net/version: v0.4.0
  creationTimestamp: null
  labels:
    audit2rbac.liggitt.net/generated: "true"
    audit2rbac.liggitt.net/user: system-serviceaccount-ns1-sa1
  name: audit2rbac:sa1
  namespace: ns1
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: audit2rbac:sa1
subjects:
- kind: ServiceAccount
  name: sa1
  namespace: ns1
Complete!
$ audit2rbac -f /tmp/kube-apiserver-audit.log --serviceaccount=ns1:sa1 | kubectl apply -f -
-
Opening audit source...
Loading events...
Evaluating API calls...
Generating roles...
Complete!
role "audit2rbac:sa1" created
rolebinding "audit2rbac:sa1" created
$
```

# Docker

- <https://www.youtube.com/watch?v=7mzblOtclaQ> - Jessie Frazelle's History of Containers keynote
- <https://github.com/openSUSE/umoci> - a complete manipulation tool for [OCI images](#)
- <https://github.com/projectatomic/skopeo> - work with remote images registries to retrieve information and images, and sign content
- <https://contained.af> - Docker/Kubernetes CTF  
(<https://github.com/jessfraz/contained.af>)





# Persisting Configuration: Continuous Security



# Continuous Security



# Continuous Infra Security

- The system can continually self-validate
- Test pipelines are more robust
- Highly skilled penetration testers are free to focus on the “high-hanging fruit”





# Conclusion

- The brave new world of Kubernetes increases attack surface and potential for misconfiguration
- Lots of new security primitives are landing
- The only way to iterate quickly is: supported by a test suite
- Security testing keeps you young

