KubeCon

North America 2017

Multitenancy Deep Dive

Thursday, December 7 • 2:00pm - 3:20pm

David Oppenheimer (Google) *davidopp@google.com*
Quinton Hoole (Huawei) *quinton.hoole@huawei.com*

Agenda

- Presentations
- Discussion of topics of interest
- Ideas for 2018 (including whether we should create a Working Group)

Presentations

- Quinton Hoole, Huawei
- Jessica Frazelle, Microsoft
- Harry Zhang, Hyper
- David Oppenheimer, Google
- Tim Allclair, Google

Multi-Tenancy Models

Jessie Frazelle - Microsoft

refer to original doc

Soft Multi-Tenancy

- multiple users _within the same organization_ in the same cluster.
- could have possible bad actors such as people leaving the company, etc.
- Users are not thought to be actively malicious since they are within the same organization, but potential for accidents or “evil leaving employees.”
- A large focus is to prevent accidents.

Hard Multi-Tenancy

- multiple users, from various places, in the same cluster.
- means that anyone on the cluster is thought to be potentially malicious and therefore should not have access to any other tenants resources.

Access to k8s API

For our purposes, we only run untrusted workloads, but we have our own trusted API on top of the kubernetes API

(seems like SaaS from davidopp's doc.)

A different multi-tenancy models would also restrict access to the API and create roles, etc. Refer to that doc for more details.

Host OS

Container Runtime

Network

DNS

AuthN/AuthZ

Isolation of Master and System nodes.

Isolation of system services.

Restricting access to host resources.

Environment Variables

Thoughts about Hard Multi-tenancy in Kubernetes with Hypervisor based Container Runtimes

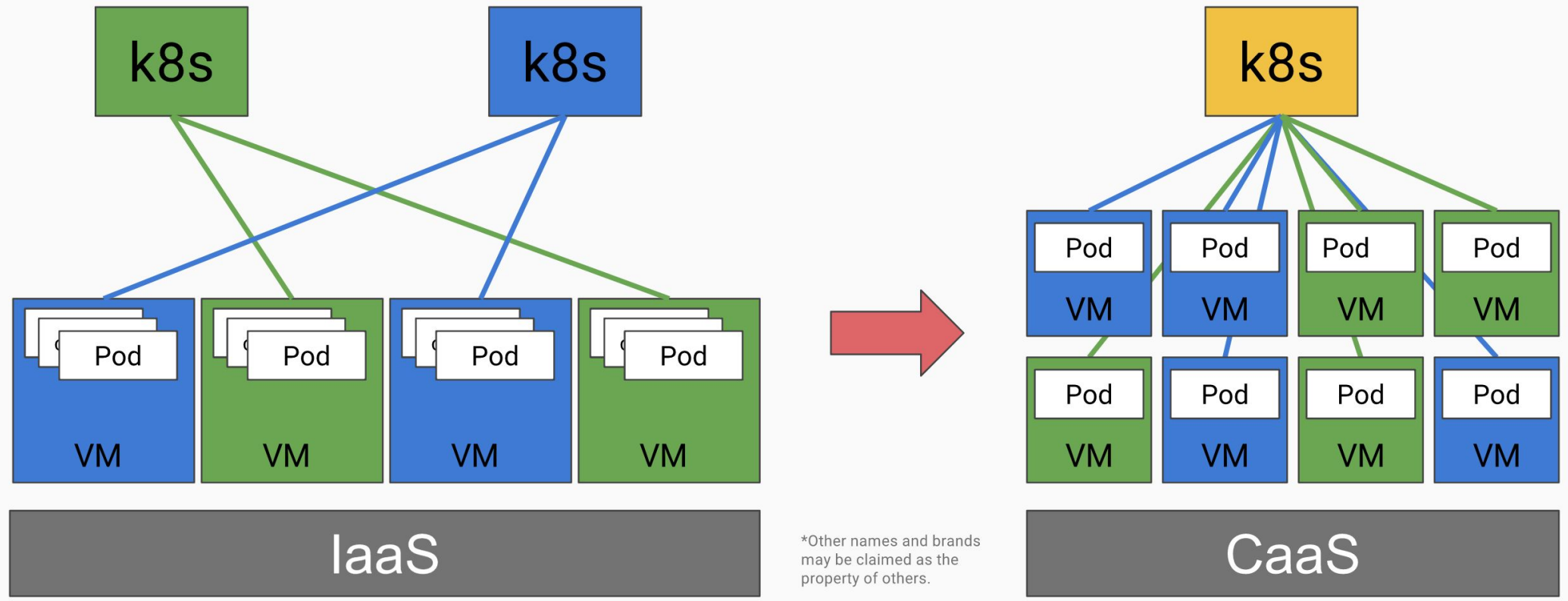
Harry (Lei) Zhang @resouer

Background

- Hypernetes (Stackube):
  - A multi-tenant Kubernetes distro with hypervisor based container runtime
    - runV, now upgrading to KataContainers
  - https://github.com/openstack/stackube
    - Upstream Kubernetes + customized plugins
    - The core system behind https://hyper.sh/
    - Passed 100% conformance e2e tests

Container Runtime: Isolation & Security

- KataContainers



Container Runtime: OS Multi-Tenancy

BYOK (Bring Your Own Kernel):

```
annotations:
  com.github.katacontainers.KernelPath: "/boot/vmlinuz-custom-myversion”
```

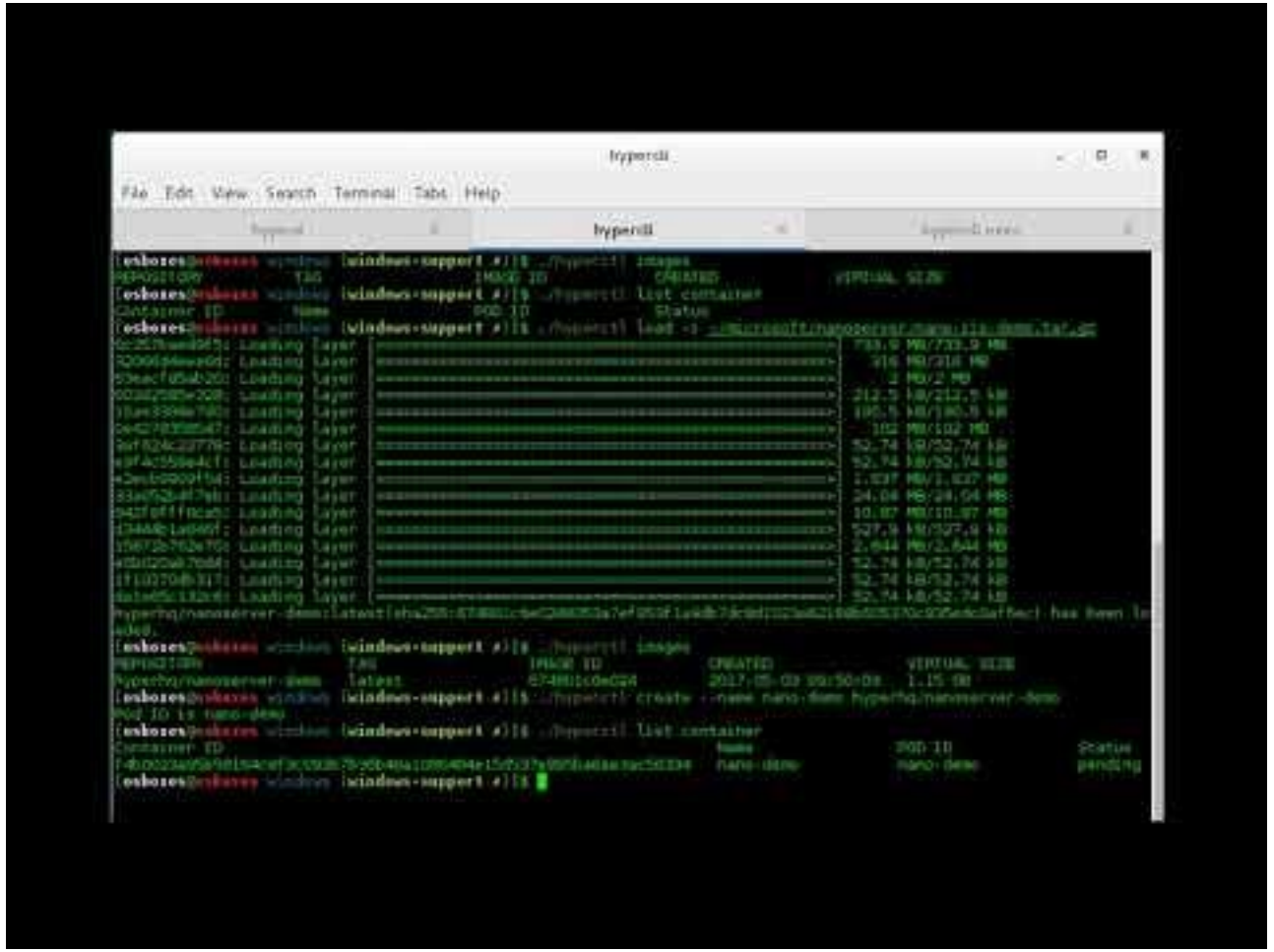
(This has already been implemented)

Or even:

```
annotations:
  com.github.katacontainers.KernelPath: "/boot/windows-nano-server-myversion”
```

(This has also been concept proved)

Container Runtime: OS Multi-Tenancy



Tenant

- Stackube:
  - Tenant == Namespace
    - CRD
    - tenant controller with RBAC
    - Keystone
- Q:
  - Do we need nested namespace? (One Tenant with multiple namespaces?)
    - Tenant == Namespace, or Tenant 1:N Namespace

Network

- Stackube:
  - One Network per Tenant
    - CRD
    - network controller
    - Neutron CNI plugin
  - L2 network isolation
    - Pods of same Tenant belong to same subnet
- Q:
  - Do we really need Network API object? Or Network Policy should be the plan?.
    - And what about multiple networks?
  - Is L2 isolation specially preferred for hard multi-tenancy?
  - Do we need to isolate Nodes and Pods by different subnets?

DNS

- Stackube:
  - Per kube-dns per namespace (tenant)
- Q:
  - Discussion: https://github.com/kubernetes/dns/issues/132
  - Other approach:
    - Sidecar, we use this in old version of Hypernetes
    - Enforce by CoreDNS (+RBAC)

Summary

- KataContainers can play an important role in hard-multitenant Kubernetes
  - Thanks to CRI
- While other aspects like Tenant, Network, DNS etc still need to be clearly defined or updated to build the whole stack up.
- Then what is the Kubernetes/Cloud Native way to solve them?

KubeCon

North America 2017

Multitenancy taxonomies

David Oppenheimer, *Google*

December 7, 2017

Control plane vs. node multitenancy

All policies are specified through the control plane.

Distinction is whether policy controls sharing of control plane or nodes.

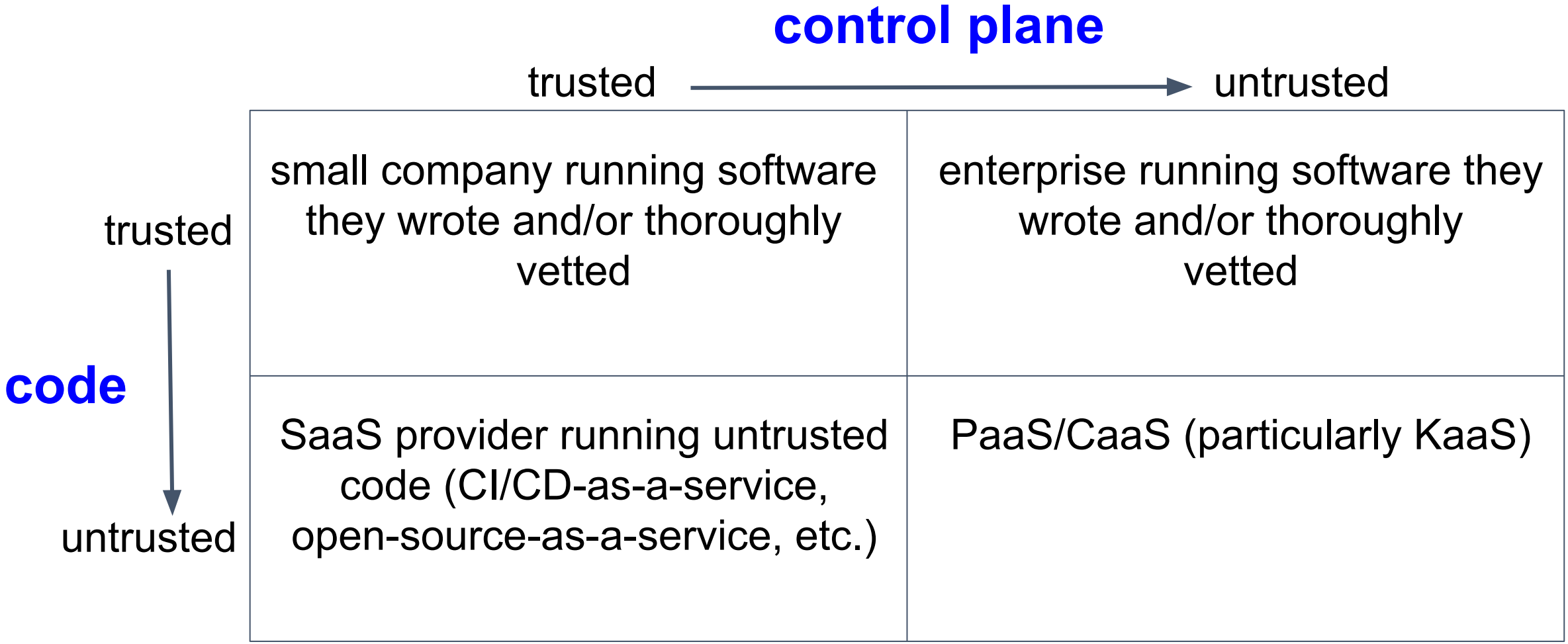
Control plane multitenancy

- RBAC
- EventRateLimit admission controller

Many for node multitenancy

- ResourceQuota / LimitRange / request / limit / priority
- node affinity, pod affinity, taints/tolerations
- PodSecurityPolicy
- NetworkPolicy

How control plane and node are used



Other axes

- What do users see?
  - objects (or subset) in user's namespace(s) or all namespace(s)?
  - nodes?
  - metadata about other tenants (namespace collision, service names in DNS, etc.)?
- Node-level isolation mechanism
  - containers + PodSecurityPolicy, seccomp, AppArmor, SELinux, ...
  - container + hypervisor (nested virtualization)
  - dedicated nodes (taints/tolerations or anti-affinity)

KubeCon

North America 2017

Secure Containers

Tim Allclair, *Google*

Secure Containers

Stronger Isolation

- Sandboxing untrusted code
- VM strength isolation

Work in Progress

- CRI-O with Clear Containers
- Frakti with runV
- (soon!) Kata Containers
- Cloud providers exploring CaaS

It's time to agree on the abstractions, before we diverge too much.

Open Questions

We're kicking off the discussions now.

What are the properties of a sandbox?

- Must it employ full virtualization technology?
  - Or could a sandbox be a very restrictive seccomp profile?
- What does sandboxing imply about networking?
- What does sandboxing imply about auth[nz]?
- What features is it OK to break with a sandbox?
  - E.g. cross-container IPC? host namespaces? etc.

Where is the sandbox boundary?

Pods?

- Easier resource sharing & communication between containers.
- Better for a serverless (nodeless) model
- Much simpler networking

Containers?

- Finer grained control allows for models like trusted sidecars

Or should we consider something else?

Namespaces? Sandbox resource? A combination of pods + containers?

API Design

How do we surface sandboxes to the user?

- Explicit, without choice of backend?
  - `Sandbox *bool`
- Explicit, with choice of backend?
  - `Sandbox string`
- Implicit, derived from security attributes?
  - See: Entitlements

Implementation details

Sandboxed & unsandboxed containers should live side-by-side.

Should sandboxing be enforced by the runtime (CRI), or the Kubelet?

Does the kubelet decide which CRI server to talk to, or just pass the sandbox bit on the CreateContainer request?

Stay tuned!

Look for a design proposal soon...

Expect more discussions in sig-node meetings

Thoughts? Questions? Get in touch!

- tallclair@google.com
- **@tallclair** (github, slack, twitter)

Possible group discussion topics (1)

- How are you (and/or your customers) using the existing Kubernetes multitenancy features? What problems/use cases are you solving?
- What would you (and/or your customers) like to do, but can't (or are rolling your own, and would like it supported in upstream)? What problems/use cases/pain points would this address?

(Consider both control plane and node support for multitenancy)

Possible group discussion topics (2)

- Hierararchy vs. labels vs. good enough how it is
  - policies that span namespaces and/or apply within a namespace?
- Need better hiding of tenants from one another?
- Issues with tenants DoSing each other via the control plane?
- Uses cases and missing features for “hard” multitenancy
  - need more isolation in the control plane?
  - need “secure containers”?

2018 planning

- Should we create a multitenancy Working Group?
- Specific multitenancy features you want and/or are interested in working on?

Note: a mailing list has been set up -- please join it!

https://groups.google.com/forum/#!forum/kubernetes-wg-multitenancy