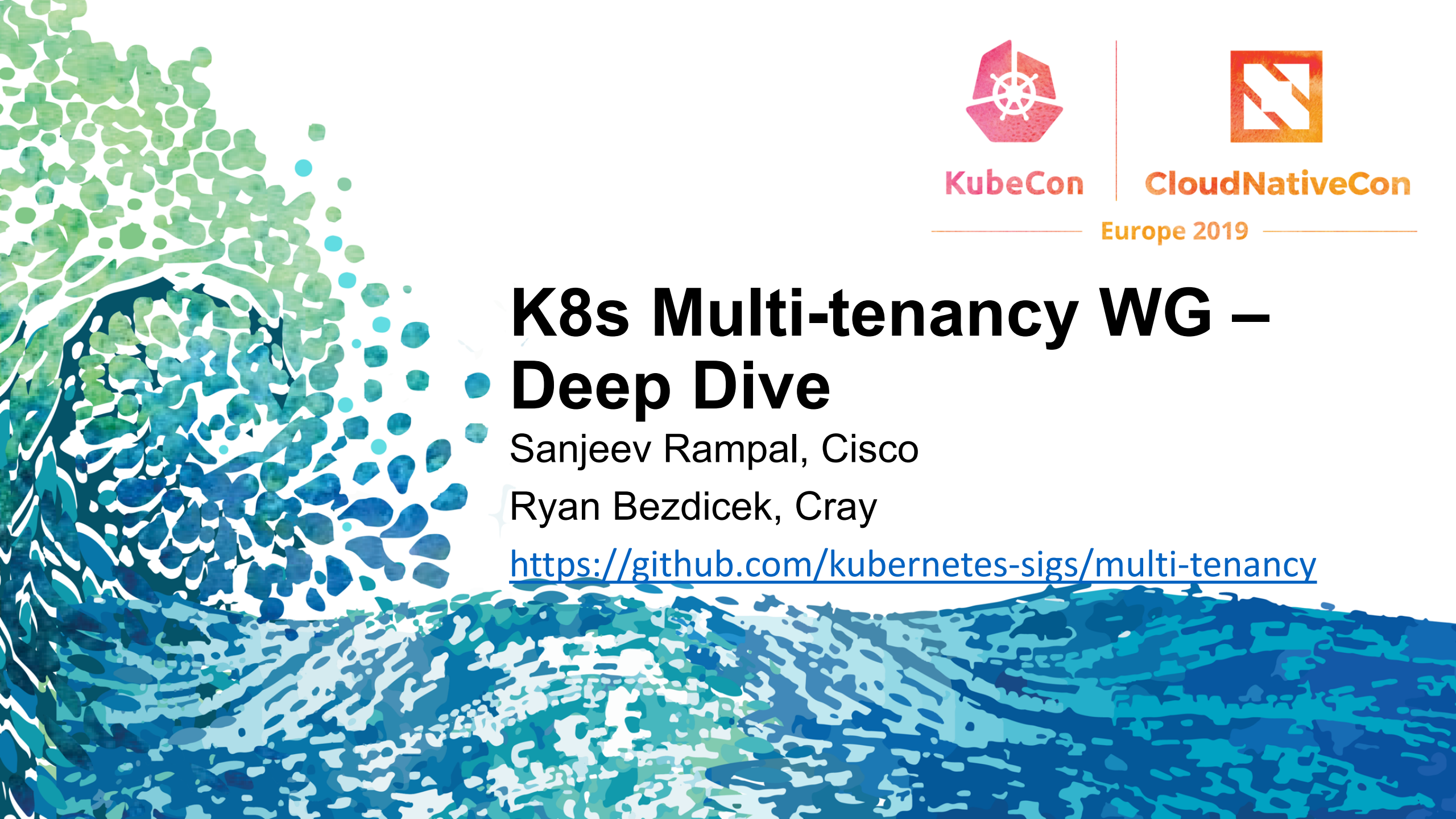


K8s Multi-tenancy WG – Deep Dive

Sanjeev Rampal, Cisco

Ryan Bezdicek, Cray

https://github.com/kubernetes-sigs/multi-tenancy

Agenda

- Introduction and architecture options
- A Sample Reference Multitenancy Architecture
- Demo
- Q&A, open discussion

Kubernetes Multi-tenancy: What is it ?

- Functionality to allow secure sharing of a Kubernetes cluster by multiple “Tenants”

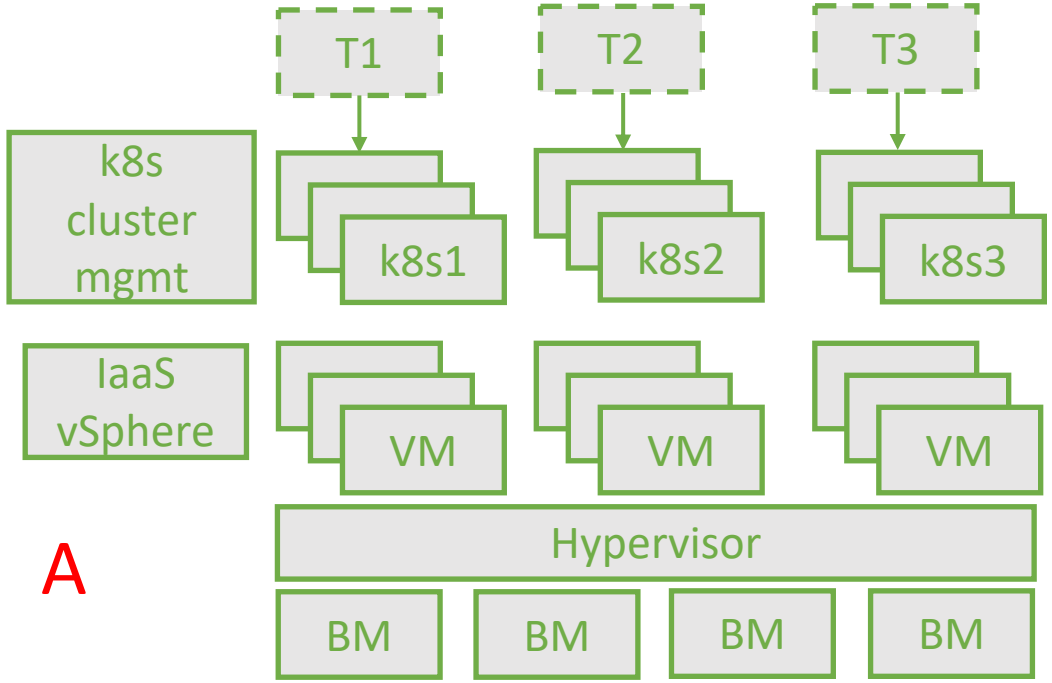
- What is a ***“Tenant”*** ?
  - A subset of resources (compute, network, storage, other) within a single Kubernetes cluster that has “soft” or “hard” isolation from the rest of the cluster and typically setup for use by a single team of users

- Why do this ?
  - Capex and opex efficiency
  - Not doing this can lead to cluster sprawl, stack sprawl, inefficient resource use
  - Specially useful when running containers/ K8S directly on Bare Metal servers

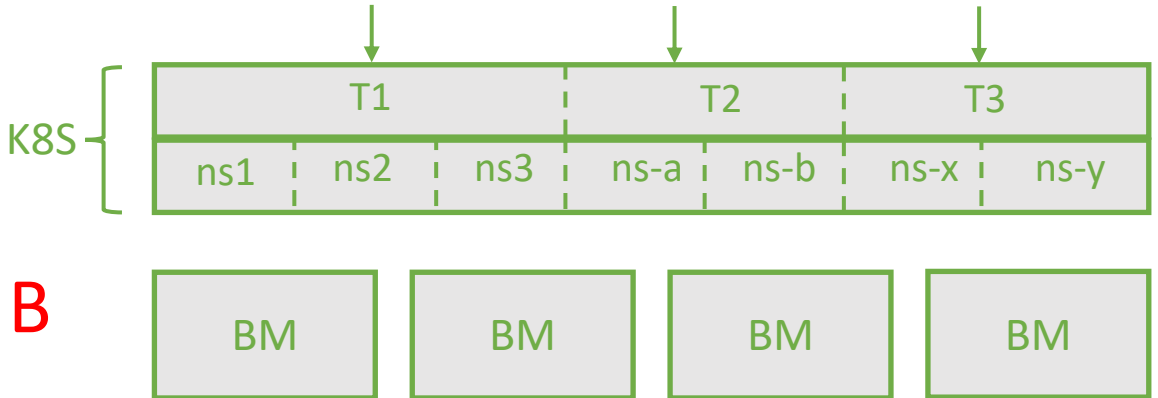
Kubernetes Multi-tenancy: What is it ?

- Some relevant side questions:
  - Is a Tenant a set of “resources” ? Or a set of “users” ? Or other ?
  - Scope of a Tenant: Single cluster ? Or Across Multiple clusters ?
  - Do we need to support multiple different degrees of “isolation” between tenants ?
  - What aspects of multi-tenancy should be standardized vs left to vendor specific customization ?
- Working assumptions (for now):
  - A Tenant is a subset of resources (not users … and btw k8s wants to keep users out anyway) and its scope is a single cluster for now (multi-cluster in future)
  - Multiple potential degrees of isolation with 2 core use cases:
    - Soft Multitenancy (Typically an Enterprise use case)
      - Use case: Multiple teams within an enterprise that still require secure isolation
    - Hard Multitenancy (Typically a “K8S Provider” use case)
      - Use case: Service provider class multitenant cluster with “hard isolation” between completely untrusted and independent tenants
- K8S Multitenancy Working group working on standardization

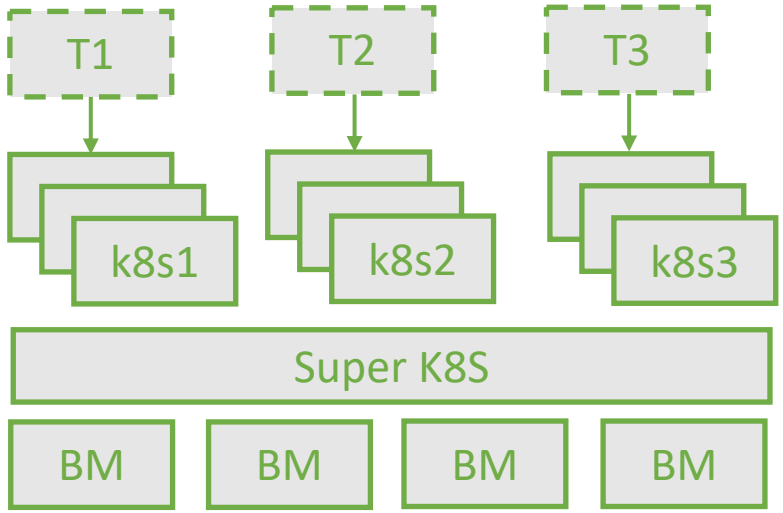
Multi-tenancy Architecture Options



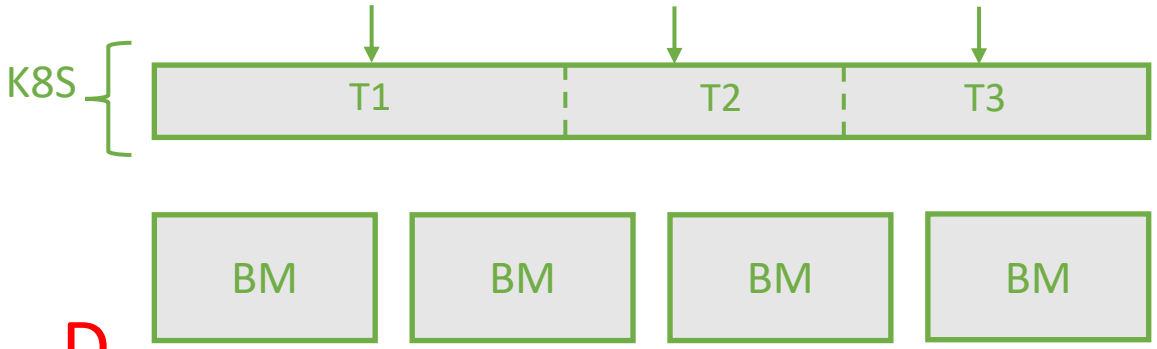
A



B



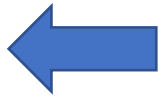
C



D

Architecture Options

Multitenancy Architecture Option	Resource efficiency	Level of Tenant isolation	Tenant/ application Config restrictions	All "Cloud Native" architecture	Architecture maturity & production readiness
A: Multiple K8S clusters on top of a Virtualization IaaS	Low-medium	High	No	No (multiple separate platforms, orch.)	Medium-High
B: Namespace grouping with Tenant resources	High	Medium-High	Minor restrictions	Yes	Medium
C: Virtual Kubernetes Clusters	High	High	No (?)	Yes	Low (very early)
D: Core Kubernetes change (Tenant as 1st class resource)	High	High	No (?)	Yes (in theory)	Very low (design does not exist)



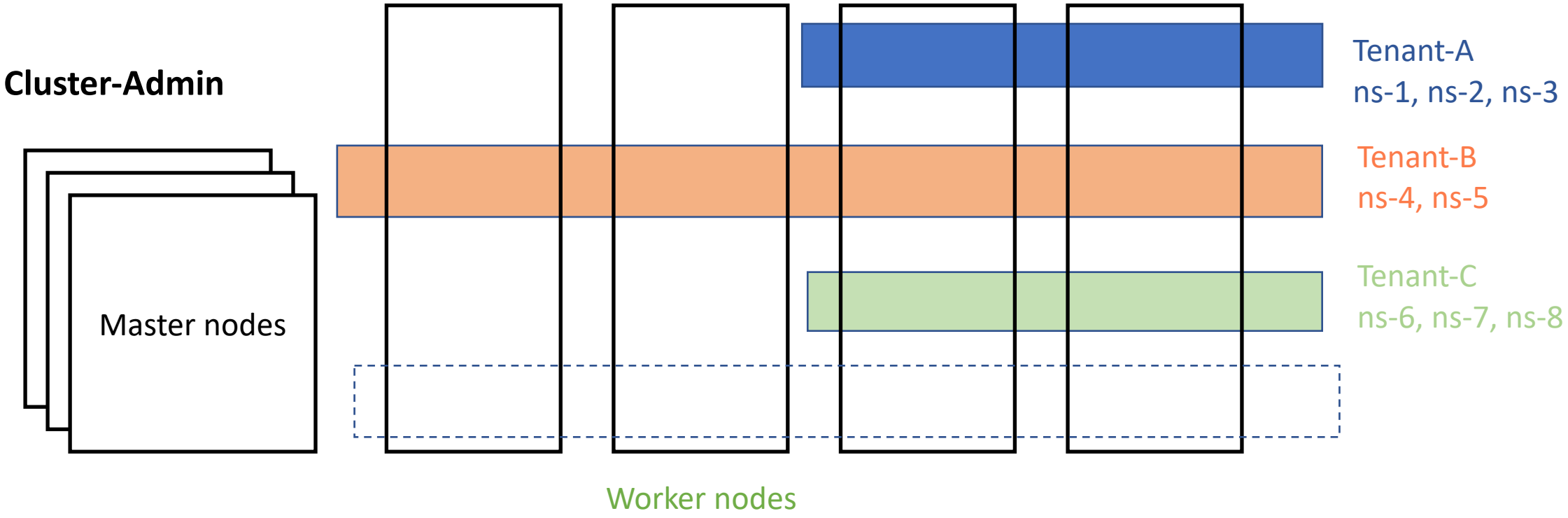
Initial focus (+ continue Investigation of other options in parallel)

Sample conservative reference solution for today

- Hybrid solution (Option A + Option B)
  - Dedicated clusters (Option A) for tenants that need hard isolation, privileges
  - Shared clusters (Option B) for non-privileged tenants/ applications

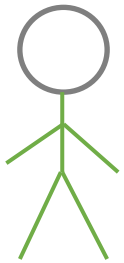
- Potential reference behavioral model for option B clusters today
  - Tenant containers have no kernel or host level privileges at all (pure user space cloud native applications)
  - Tenants can not provision any cluster scoped k8s resources (e.g. pod security policies) and may even be restricted from some Namespace scoped k8s resources (e.g. CNI NetworkPolicies not available to tenants, only cluster admins)
  - K8S pods/ resources in one tenant can not communicate with pods in other tenants
  - Cluster admin enforces tight control via RBAC, Pod Security policies, CNI network policies
  - Move cluster security profile from “tribal knowledge” to curated (and standardized) profiles

Option B: Namespace grouped Tenants

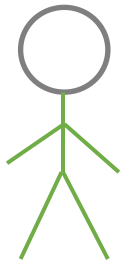


Tenant holds 1 or more fully self-contained applications/ services
No direct east-west communication option between tenants except via N-S apis
Strictly non-privileged pods within such tenants
Resource quotas, chargeback and billing at tenant level

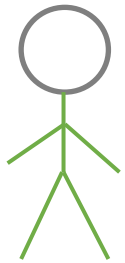
Option B: Personas and workflows



Cluster-admin



Tenant-admin



Tenant-user

Cluster-admin provisions K8S cluster with 1 (of N) recommended security profiles → **Cluster-admin** provisions Tenant template and Namespace template objects

Tenant-admin provisions a new tenant referring to these templates → **Tenant-admin** provisions access controls for the new tenant including other admins & non-admin user RBAC

Tenant-admin performs CRUD operations and tenant life cycle mgmt. on the tenant resource itself

Tenant-user provisions namespace scoped k8s resources within tenant

Option B: Secure Multitenancy Profile & Enablers

Profile 1: Bare minimum to get started

- Secure by default Kubernetes configuration
  - Disable anonymous authentication
  - Disable ABAC, disable local authorization,
  - K8S secrets encryption enabled
  - CIS Kubernetes benchmarks Level 2 requirements
- Enable RBAC
- Recommended default set of admission controllers (NodeRestriction, AlwaysPullImages, PodSecurityPolicy etc)
- Pod Admission controller (PodSecurityPolicy)
- CNI Container Network Policy enabled including ingress and egress policies
- Docker run-time with Seccomp, AppArmor/ SELinux default profiles
- Best effort multi-tenancy for services (monitoring, logging etc)

Profile 2: Recommended Phase 2 (WIP @ WG)

- Profile 1 + additional required enhancements including:
- Dynamic policy engine (e.g. OPA) based enhancement for
  - Access control/ RBAC
  - Admission control (beyond Pod Security policies)
  - Advanced policy controls (e.g. ingress route policies)
- Newer container runtimes & runtime sandboxing options (CRI-O, containerD w/ Kata runtime, Firecracker/ gVisor)
- Complete solution for multi-tenancy across monitoring, logging, storage, service mesh ..
- Tenancy across Multi-cluster, multi-cloud

Tenant & NamespaceTemplate CR & Workflow

Tenant CRs created by any user, Admins named explicitly for subsequent CRUD operations

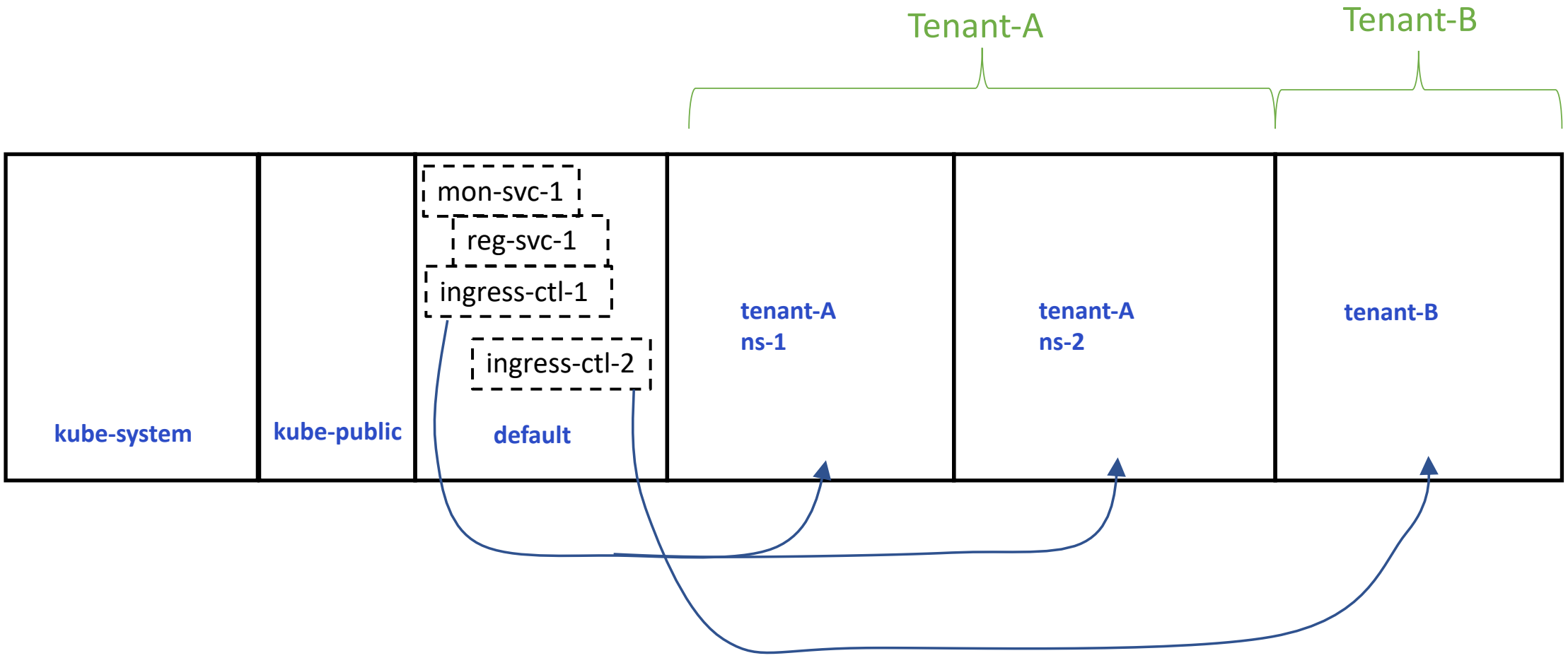
```
Kind: Tenant
Metadata:
  Name:  Tenant-A
Spec:
  Namespaces:
      ns-1
          template   t1
      ns-2
          template   t2
  Admins:
      user1
      group2
      rbacsubjectfoo
```

```
Kind: NamespaceTemplate
Metadata:
  Name:  t1
Spec:
    PodSecurityPolicy  P1
    NetworkPolicy      N1
```

```
Kind: NamespaceTemplate
Metadata:
  Name:  t2
Spec:
    PodSecurityPolicy  P1
    NetworkPolicy      N2
```

NamespaceTemplates Pre-created by ClusterAdmin

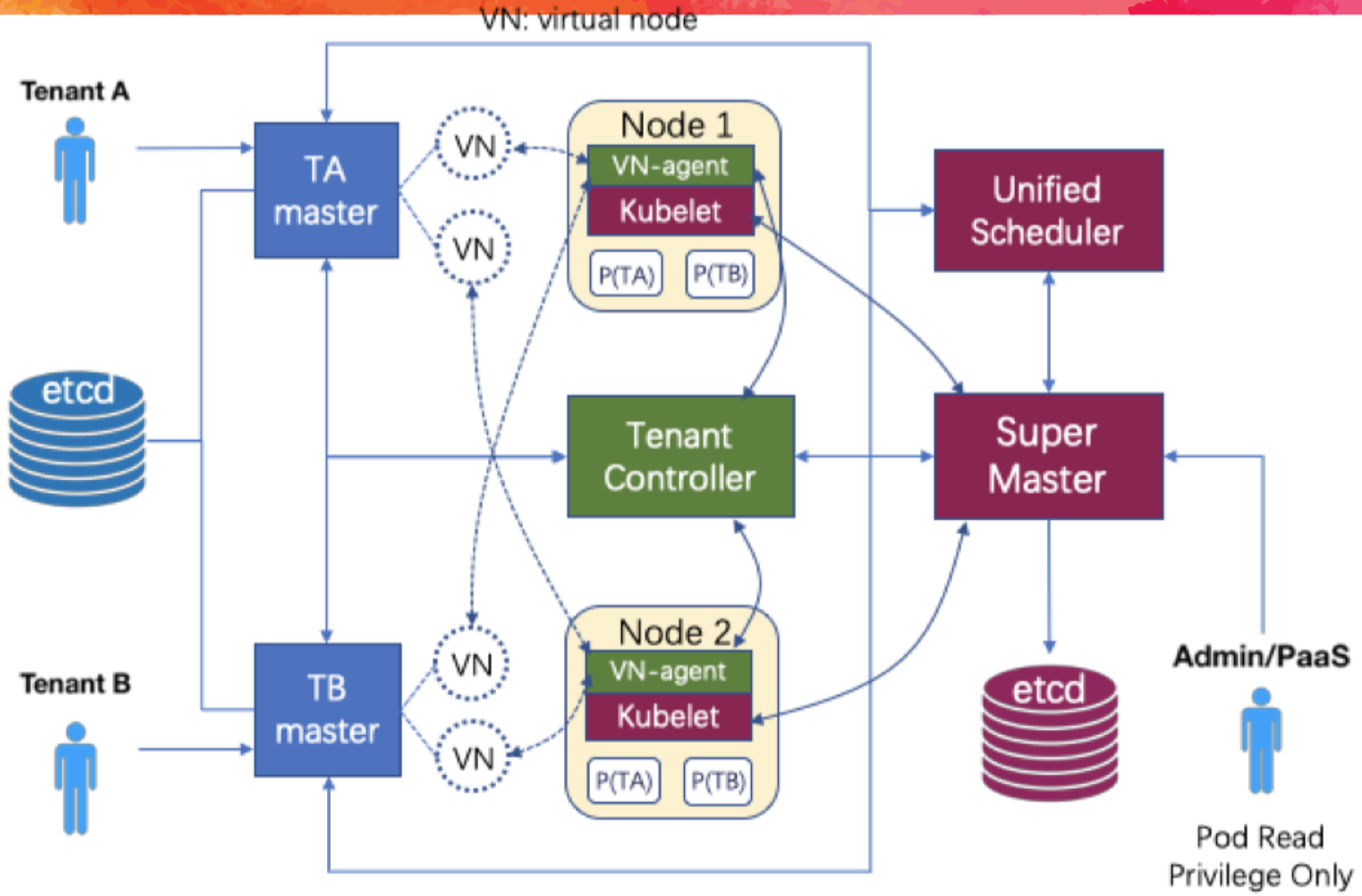
Tenant namespaces layout incl shared services



Option B: Some ongoing works in progress

- Complete multitenancy profile reference models
- Namespace naming conflicts across tenants:
  - Tenant controller prototype prefixes tenant name to namespace name
- Resource quota management for tenants across namespaces
- Should tenant users be allowed to create CNI network policies within their namespaces ?
- Should tenant users be allowed to create a subset of cluster scoped resources or bindings ?
- Multitenancy for shared services, Prometheus, logging etc

Option C (brief early look): Virtual Kubernetes Clusters



Virtual Cluster Architecture Proposal; F Guo et al; Alibaba Cloud

Demo – Tenant Controller & Workflow PoC

Thanks Yisui Hu!

Sample manifests used in demo

Sample namespace template definition:

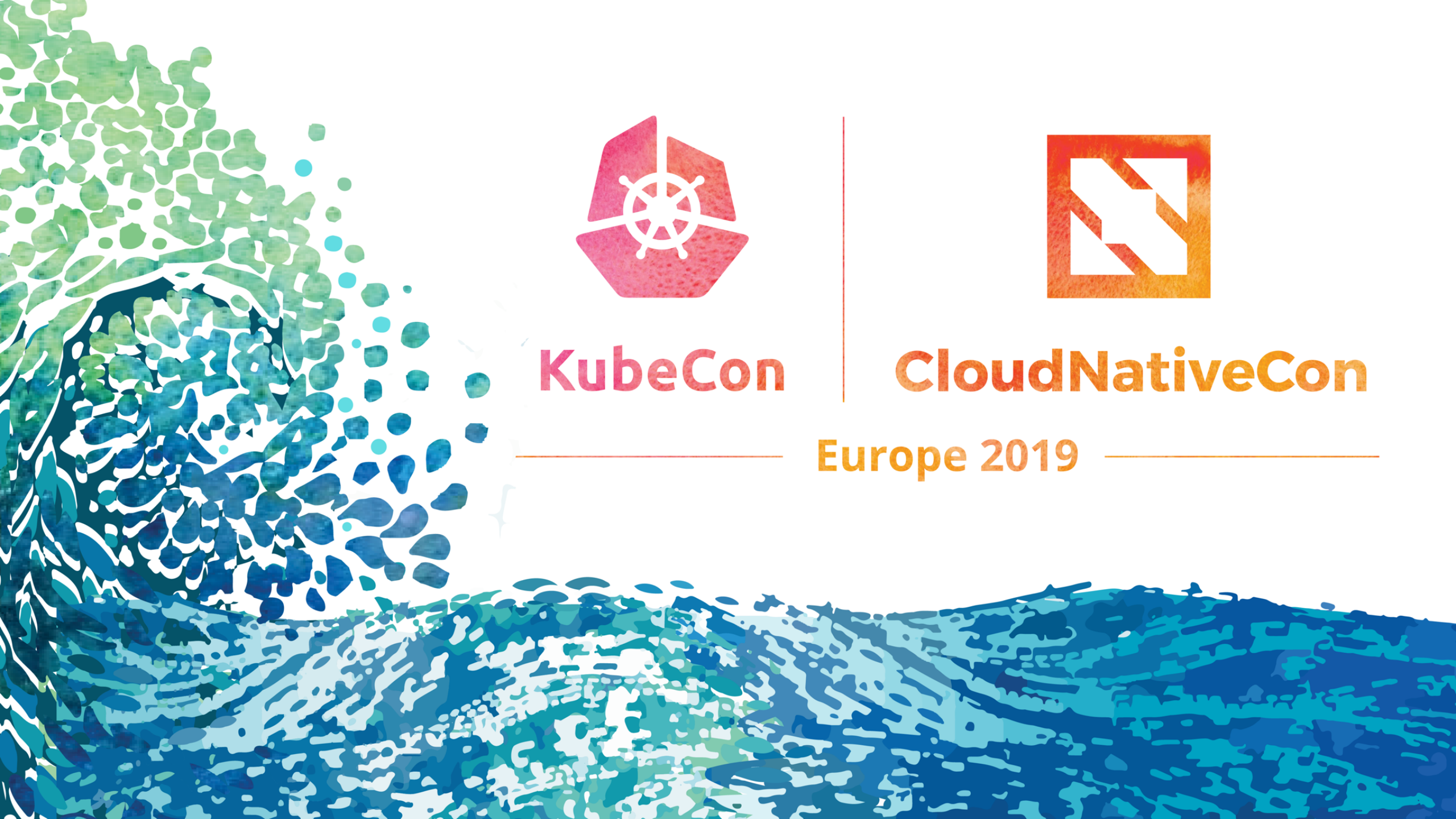
```
apiVersion: tenants.k8s.io/v1alpha1
kind: NamespaceTemplate
metadata:
  name: restricted
spec:
  templates:
  - apiVersion: rbac.authorization.k8s.io/v1
    kind: RoleBinding
    metadata:
      name: restricted-tenant-psp
    roleRef:
      apiGroup: rbac.authorization.k8s.io
      kind: ClusterRole
      name: 00-restricted-psp
    subjects:
    - kind: Group
      apiGroup: rbac.authorization.k8s.io
      name: system:serviceaccounts
```

Sample tenant instance definition

```
apiVersion: tenants.k8s.io/v1alpha1
kind: Tenant
metadata:
  name: tenant-a
spec:
  namespaces:
    - name: ns-1
      template: restricted
    - name: ns-2
      template: restricted
  admins:
    - apiGroup: rbac.authorization.k8s.io
      kind: Group
      name: tenant-a-admins
```

Takeaways

- Multitenancy standardization still very early although lots of useful custom solutions exist and should be evaluated
- Contribute your comments/ requirements to Multitenancy WG
- Evaluate initial set of CRD prototypes
- Contribute to development of multitenancy models, profiles, test criteria, custom controllers, security threat modeling etc
- Kubernetes Multitenancy working group
  - Join us on slack/ email/ bi-weekly meetings
  - https://github.com/kubernetes/community/tree/master/wg-multitenancy
- Questions/ comments ?


KubeCon
CloudNativeCon
Europe 2019

Contact information


Working Group home page
https://github.com/kubernetes/community/tree/master/wg-multitenancy

Sanjeev Rampal
srampal@cisco.com
Github: srampal

Ryan Bezdicek
ryan.j.bezdicek@gmail.com
Github: rjbez17