CloudARK

KubePlus

Multi-tenant application stacks on Kubernetes

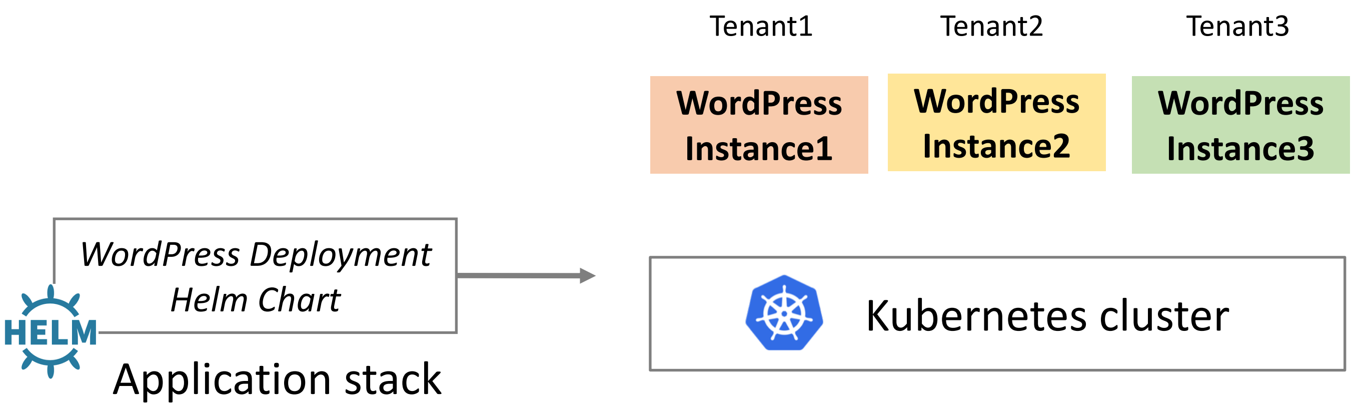
https://github.com/cloud-ark/kubeplus

Devdatta Kulkarni
Founder, CloudARK
https://www.linkedin.com/in/devdatta-kulkarni-192ab46/

cloudark.io

Application as-a-Service on K8s

Separate application instance per tenant



Tenant1

WordPress Instance1

Tenant2

WordPress Instance2

Tenant3

WordPress Instance3

WordPress Deployment Helm Chart

Application stack

Kubernetes cluster

Enterprise challenge

Multi-tenancy management

Tenant level policies

E.g., Separate node per tenant

Tenant level consumption metrics

E.g., CPU, Memory, Storage, Network consumption per tenant

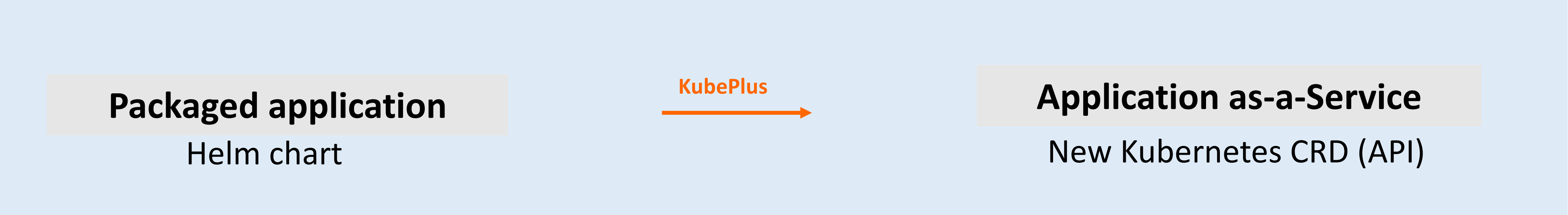
Tenant level resource topologies

Kubernetes resource relationship graph with relations between resources like Pod, Service, Custom Resource etc.

Labels are problematic: 1) Hard to check if convention is followed; 2) Custom resources complicate things

Our solution – KubePlus

Basic Idea: Wrap an API around Helm chart



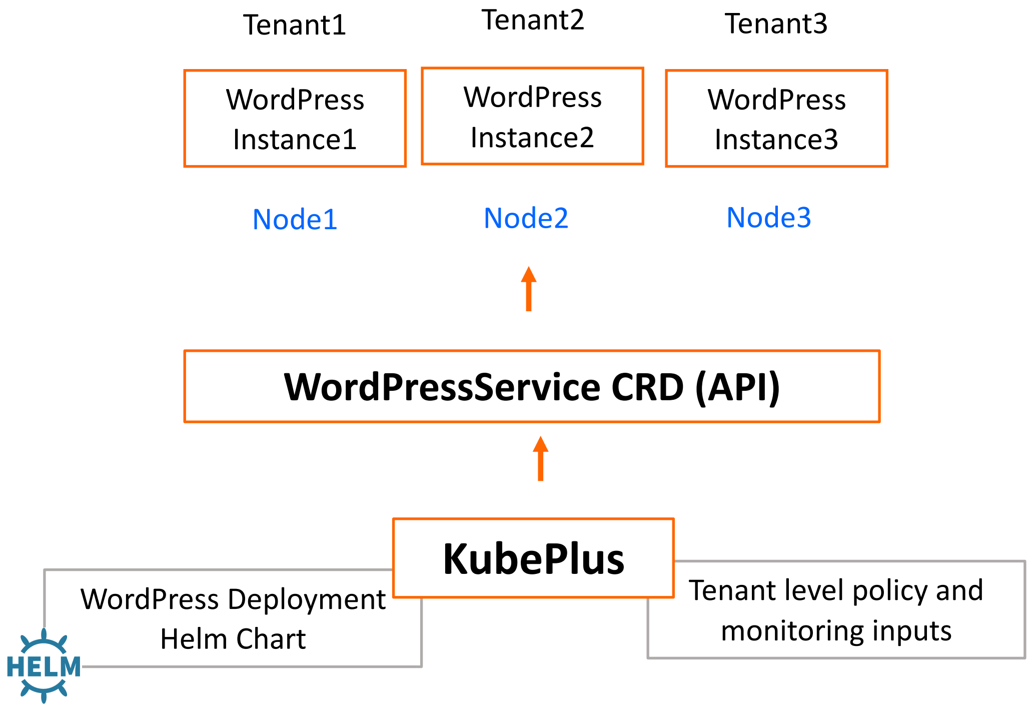
Packaged application

Helm chart

KubePlus

Application as-a-Service

New Kubernetes CRD (API)



Tenant1

WordPress Instance1

Node1

Tenant2

WordPress Instance2

Node2

Tenant3

WordPress Instance3

Node3

WordPressService CRD (API)

KubePlus

WordPress Deployment Helm Chart

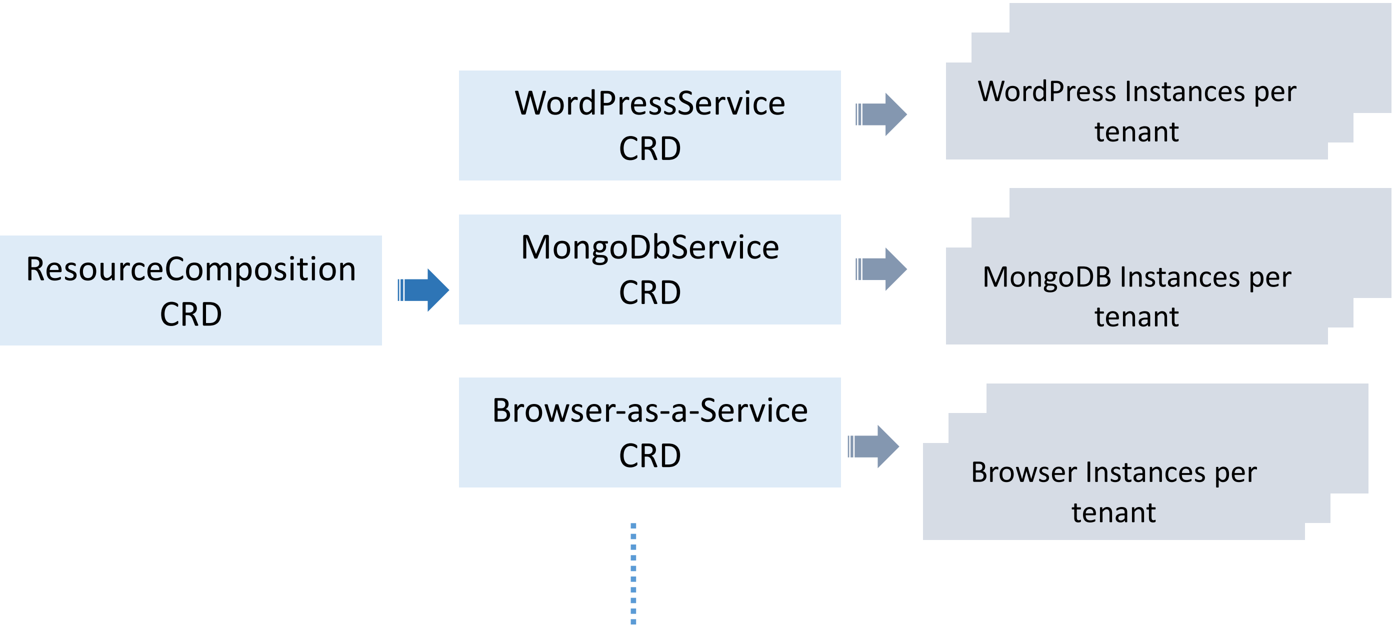
HELM

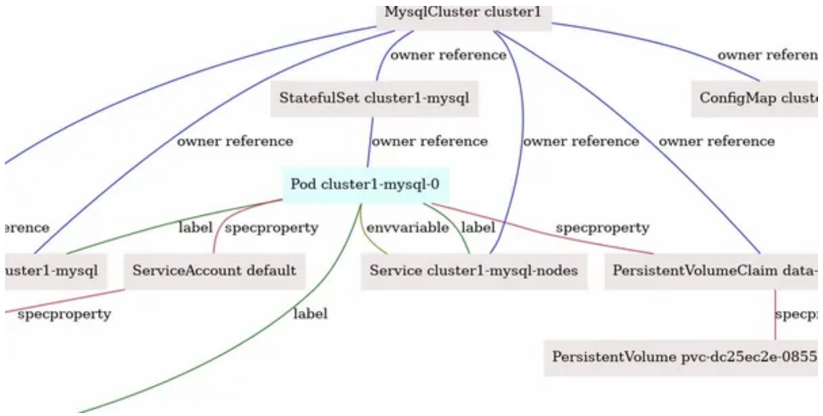
Tenant level policy and monitoring inputs

KubePlus

CloudARK

Open-source framework to design multi-tenant platform services declaratively


ResourceComposition CRD
WordPressService CRD
MongoDbService CRD
Browser-as-a-Service CRD
WordPress Instances per tenant
MongoDB Instances per tenant
Browser Instances per tenant



CRD for CRDs

Create API to manage your Helm charts

Kubectl plugins

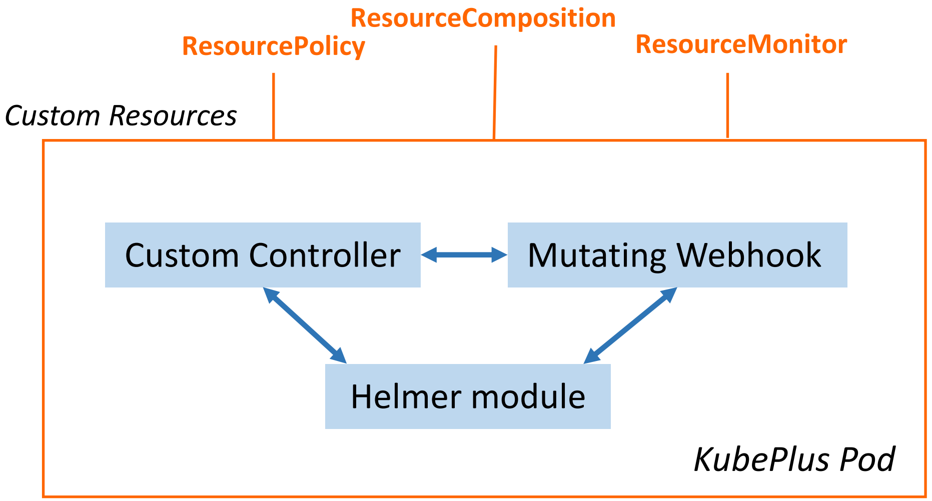
Visualize Kubernetes resource relationships

KubePlus - Platform-as-Code framework

Platform service in codified manner - as a declarative API

KubePlus CRD for CRDs components

Open-source framework to design multi-tenant platform services declaratively

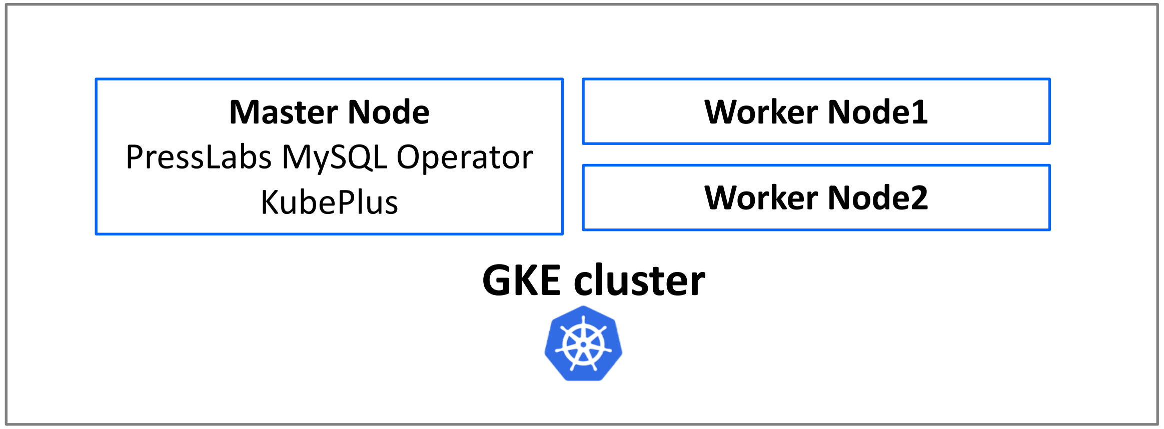


Demo scenario

Build a WordPress-as-a-service

HELM *WordPress Deployment Helm Chart*

- Pod
- MysqlCluster



Master Node
PressLabs MySQL Operator
KubePlus

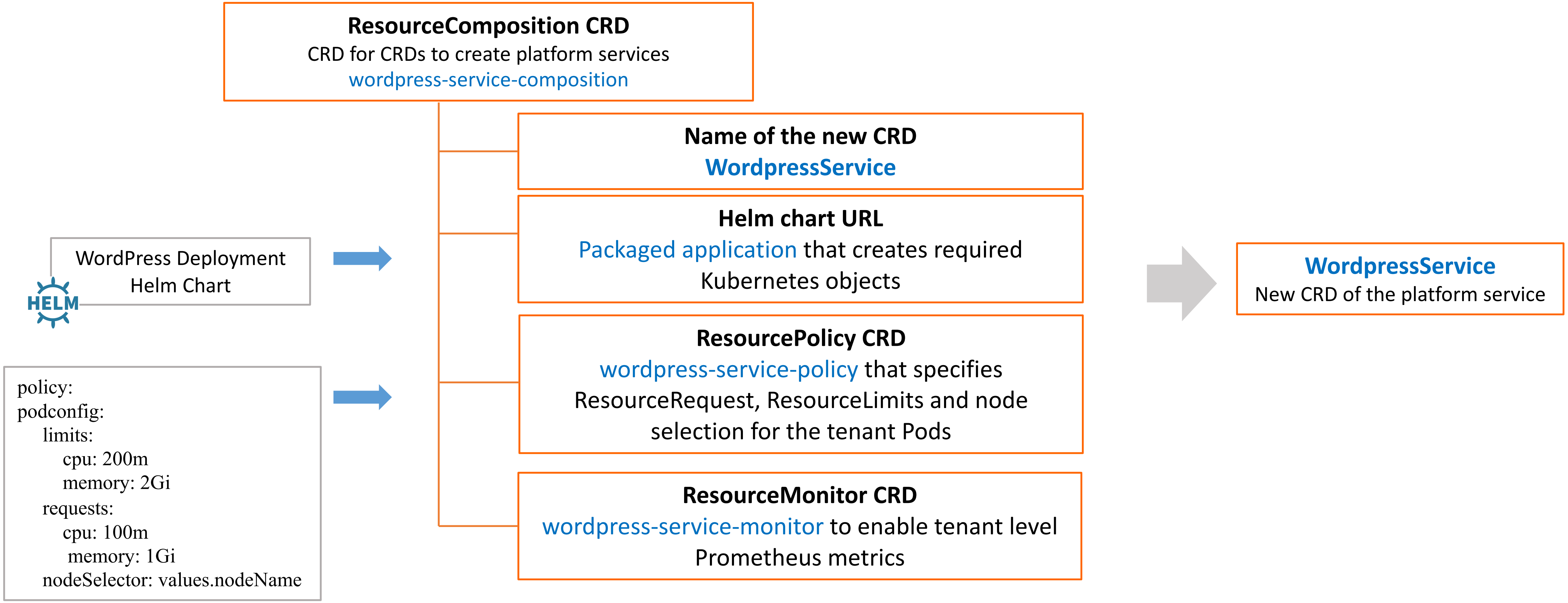
Worker Node1

Worker Node2

GKE cluster

CRD for CRD - Create WordpressService

Create a sample WordPress Service



ResourceComposition CRD
CRD for CRDs to create platform services
wordpress-service-composition

Name of the new CRD
WordpressService

Helm chart URL
Packaged application that creates required Kubernetes objects

ResourcePolicy CRD
wordpress-service-policy that specifies ResourceRequest, ResourceLimits and node selection for the tenant Pods

ResourceMonitor CRD
wordpress-service-monitor to enable tenant level Prometheus metrics

WordPress Deployment
Helm Chart

```
policy:
podconfig:
    limits:
        cpu: 200m
        memory: 2Gi
    requests:
        cpu: 100m
         memory: 1Gi
    nodeSelector: values.nodeName
```

WordpressService
New CRD of the platform service

Creating instance of WordPress Service

Create a WordPress Service instance – wp-service-tenant1

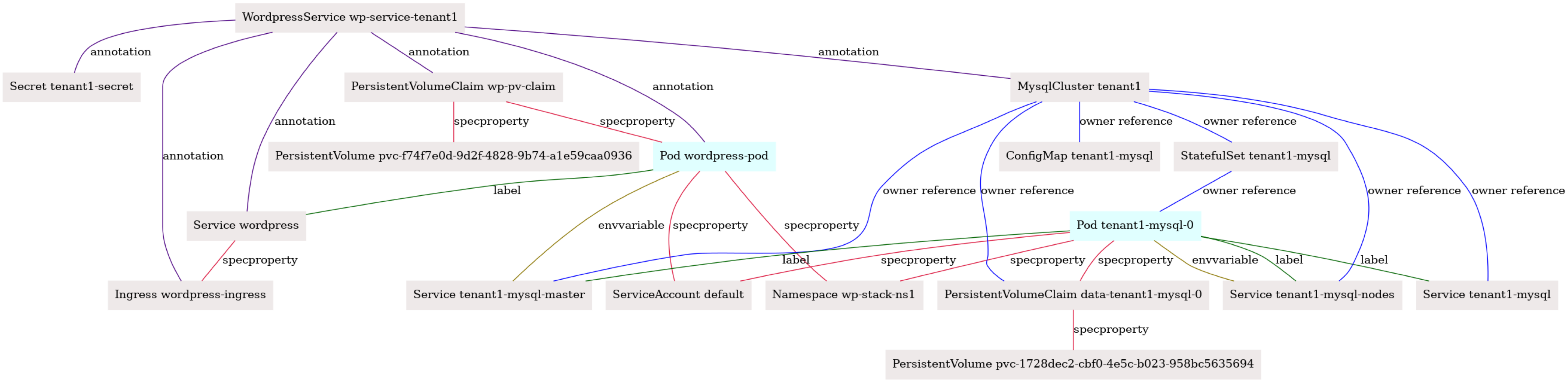
```
apiVersion: platformapi.kubeplus/v1alpha1
kind: WordpressService
metadata:
  name: wp-service-tenant1
  namespace: wp-stack-ns1
spec:
  namespace: wp-stack-ns1
  tenantName: tenant1
  nodeName: gke-cluster-4-default-pool-dacc3ab3-1x4v
```

HELM

Variables from values.yaml

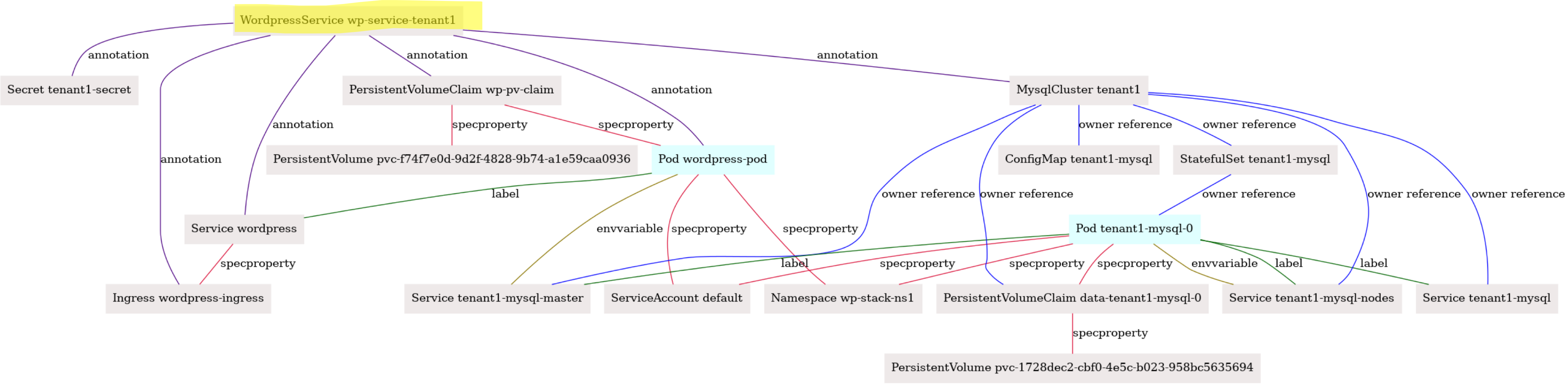
Tenant1 resource topology

Resource relationship graph



Tenant1 resource topology

Resource relationship graph



WordpressService wp-service-tenant1

annotation

annotation

annotation

annotation

annotation

annotation

Secret tenant1-secret

PersistentVolumeClaim wp-pv-claim

MysqlCluster tenant1

specproperty

specproperty

owner reference

owner reference

PersistentVolume pvc-f74f7e0d-9d2f-4828-9b74-a1e59caa0936

Pod wordpress-pod

ConfigMap tenant1-mysql

StatefulSet tenant1-mysql

label

owner reference

owner reference

owner reference

owner reference

owner reference

Service wordpress

envvariable

specproperty

specproperty

Pod tenant1-mysql-0

specproperty

label

specproperty

specproperty

specproperty

envvariable

label

label

Ingress wordpress-ingress

Service tenant1-mysql-master

ServiceAccount default

Namespace wp-stack-ns1

PersistentVolumeClaim data-tenant1-mysql-0

Service tenant1-mysql-nodes

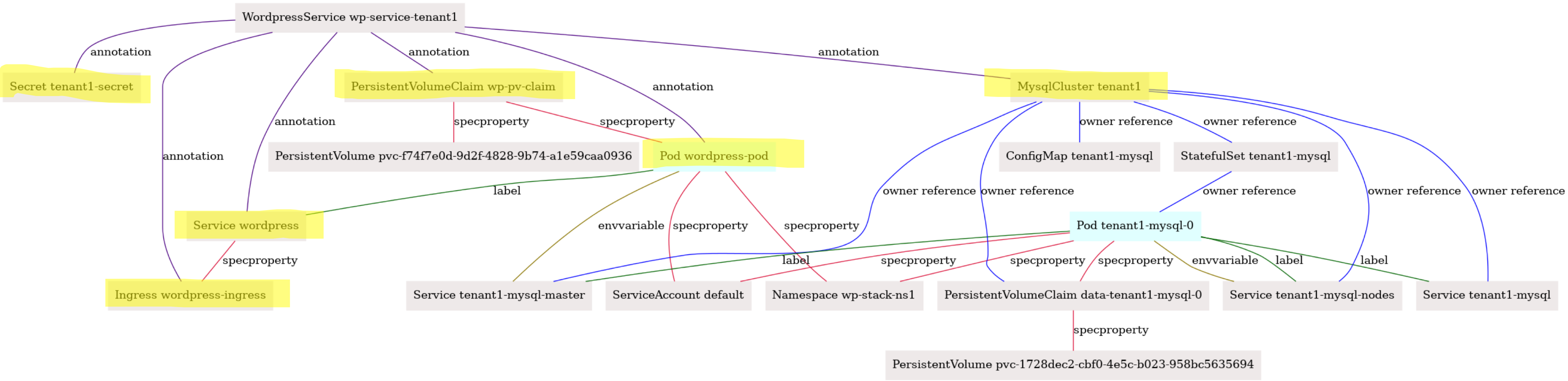
Service tenant1-mysql

specproperty

PersistentVolume pvc-1728dec2-cbf0-4e5c-b023-958bc5635694

Tenant1 resource topology

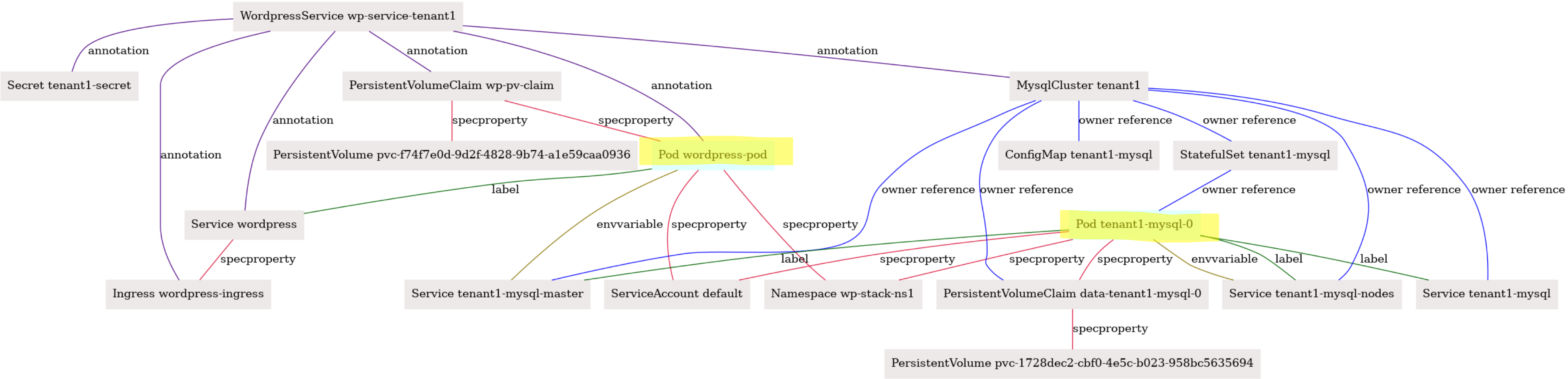
Resource relationship graph



Tenant1 resource topology

CloudARK

Resource relationship graph

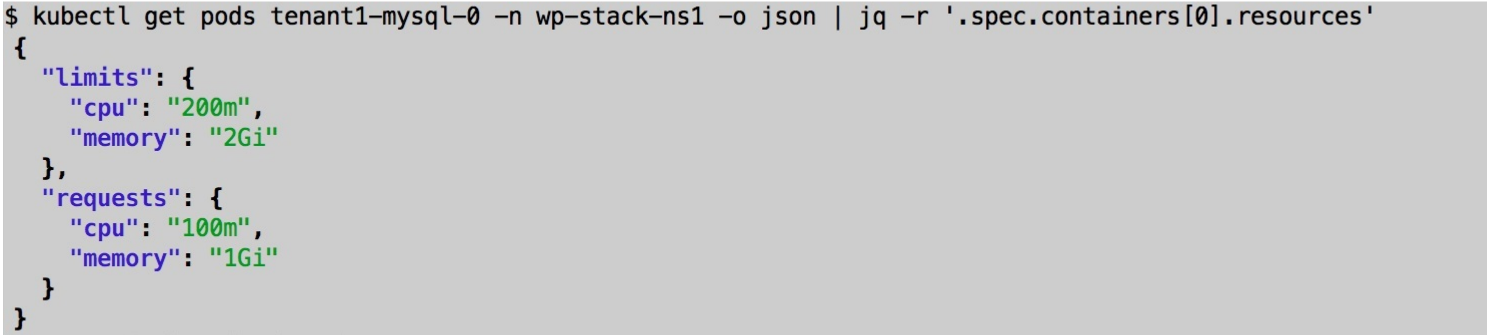


Verify policies

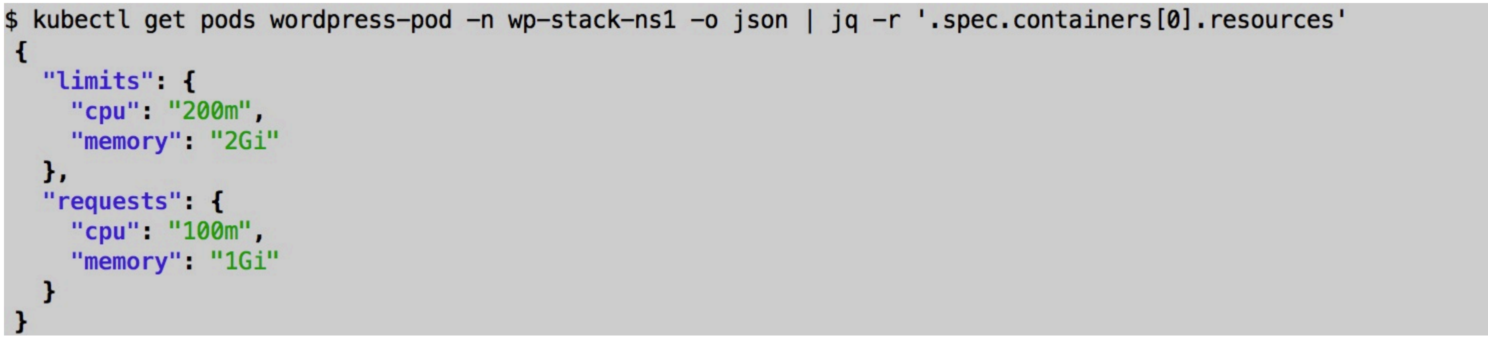
Policy Input

```
policy:
podconfig:
    limits:
        cpu: 200m
        memory: 2Gi
    requests:
        cpu: 100m
         memory: 1Gi
    nodeSelector: values.nodeName
```

Resource Requests and limits on two pods in the tenant1 stack

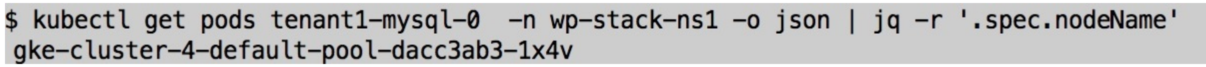


```
$ kubectl get pods tenant1-mysql-0 -n wp-stack-ns1 -o json | jq -r '.spec.containers[0].resources'
{
  "limits": {
    "cpu": "200m",
    "memory": "2Gi"
  },
  "requests": {
    "cpu": "100m",
    "memory": "1Gi"
  }
}
```

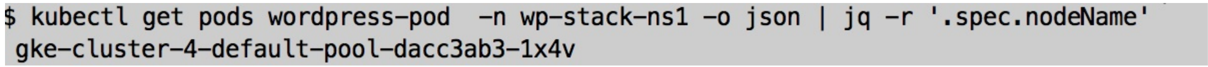


```
$ kubectl get pods wordpress-pod -n wp-stack-ns1 -o json | jq -r '.spec.containers[0].resources'
{
  "limits": {
    "cpu": "200m",
    "memory": "2Gi"
  },
  "requests": {
    "cpu": "100m",
    "memory": "1Gi"
  }
}
```

Pods running on the specified node



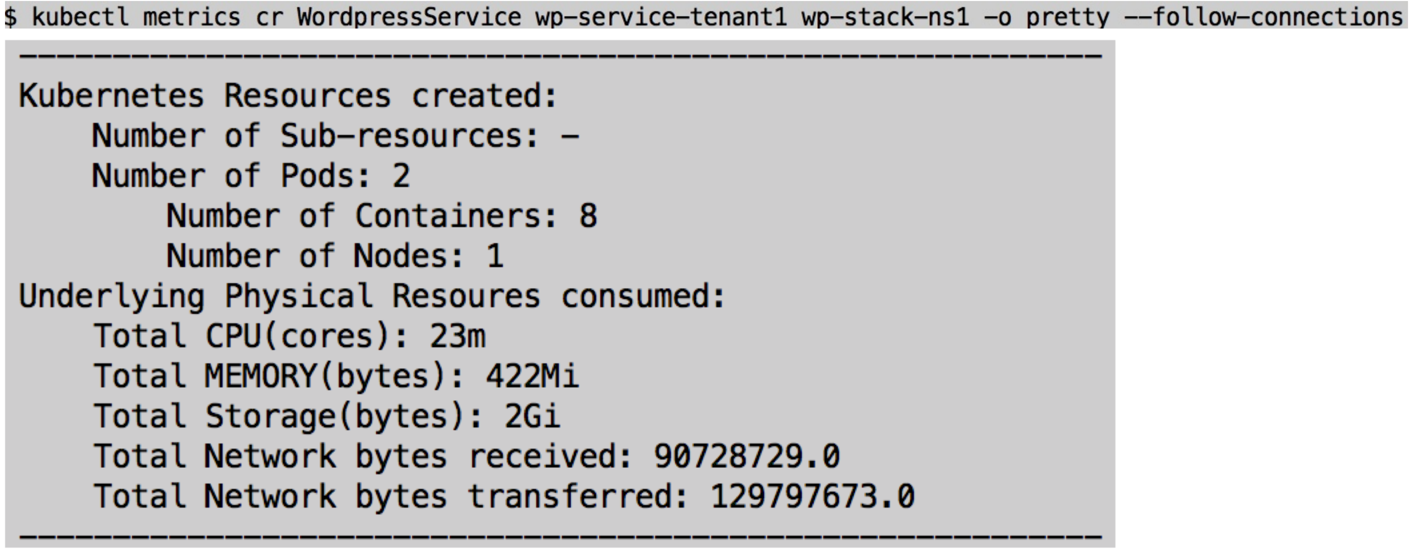
```
$ kubectl get pods tenant1-mysql-0  -n wp-stack-ns1 -o json | jq -r '.spec.nodeName'
 gke-cluster-4-default-pool-dacc3ab3-1x4v
```



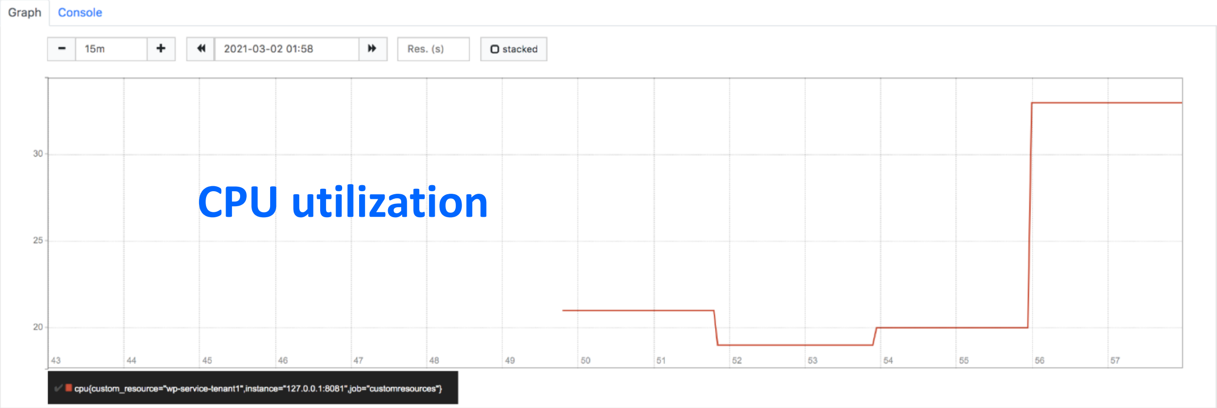
```
$ kubectl get pods wordpress-pod  -n wp-stack-ns1 -o json | jq -r '.spec.nodeName'
 gke-cluster-4-default-pool-dacc3ab3-1x4v
```

Tenant1 consumption metrics

Prometheus metrics for CPU utilization and network bytes received



```
$ kubectl metrics cr WordpressService wp-service-tenant1 wp-stack-ns1 -o pretty --follow-connections
---------------------------------------------------------------
Kubernetes Resources created:
    Number of Sub-resources: -
    Number of Pods: 2
        Number of Containers: 8
        Number of Nodes: 1
Underlying Physical Resoures consumed:
    Total CPU(cores): 23m
    Total MEMORY(bytes): 422Mi
    Total Storage(bytes): 2Gi
    Total Network bytes received: 90728729.0
    Total Network bytes transferred: 129797673.0
---------------------------------------------------------------
```

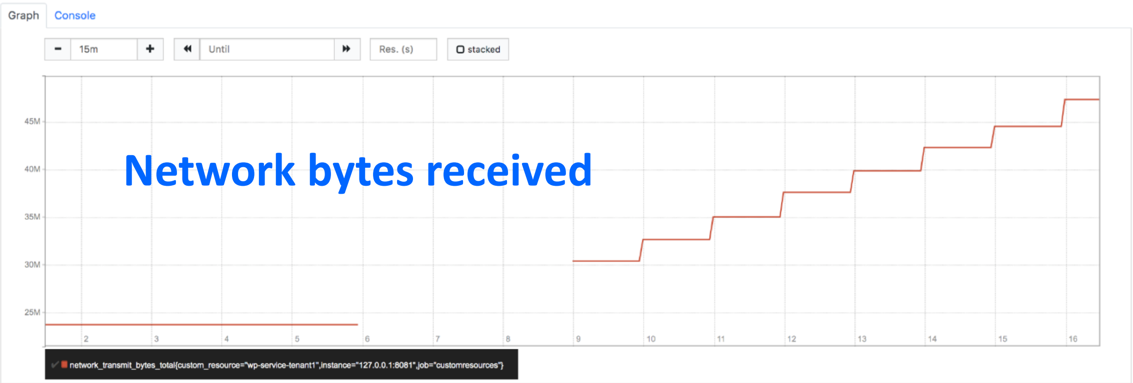


15m

2021-03-02 01:58

cpu{custom_resource="wp-service-tenant1",instance="127.0.0.1:8081",job="customresources"}

CPU utilization



15m

network_transmit_bytes_total{custom_resource="wp-service-tenant1",instance="127.0.0.1:8081",job="customresources"}

Network bytes received

Why CRD for CRD

Create Platform Services declaratively with policy and monitoring defined for the new API

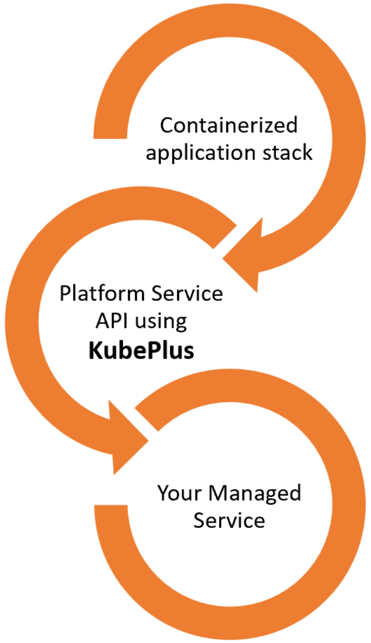
CRD for CRD vs SDK

Reduced footprint due to single Operator instance for multiple Platform Services

Create multiple services / CRDs without increasing the Operator footprint

Conclusion

KubePlus is a framework for platform engineering teams to build their multi-tenant platform services declaratively. It significantly accelerates the process of creating managed services on Kubernetes.


Containerized application stack
Platform Service API using **KubePlus**
Your Managed Service

Resources:

- KubePlus GitHub: https://github.com/cloud-ark/kubeplus
- KubePlus Documentation: https://cloud-ark.github.io/kubeplus/docs/html/html/index.html
- Platform-as-Code: https://cloudark.io/platform-as-code

Questions?