
KubeCon
CloudNativeCon
North America 2019

Growth and Design Patterns in the Extensions Ecosystem

Eric Tune
Google, Inc.

API Extensions

Kubernetes APIs are mostly about Containers: *Pod*, *Service*

The **Kubernetes Resource Model (KRM)** is:

- a pattern for building declarative APIs
- is not specific to Containers.
- easily accessible

What KRM Isn't Quite

- It isn't quite API management.
- It isn't quite a Restful Web API Framework
- It isn't just JSONSchema or OpenAPI.
- It isn't Terraform

What it is

- Not just for containers anymore.
- A KRM API is both a config format and an API.
- It is a system for building consistent declarative APIs.
- KRM APIs share:
  - a CLI
  - Metadata, Labels, Annotations
  - State storage
  - Authentication/Authorization/Auditing
  - Consistency model
  - Language Clients and Wire Protocols
  - Schema Reflection
  - Dry-run and Apply
  - Client Side Configuration and Packaging Tools

Examples

ML

- Kubeflow
- Seldon.io
- PipelineAI

CI & CD

- Tekton
- Jenkins-X
- Argo-CD

Serverless

- Knative
- Kyma

Storage

- Rook
- OpenEBS.io

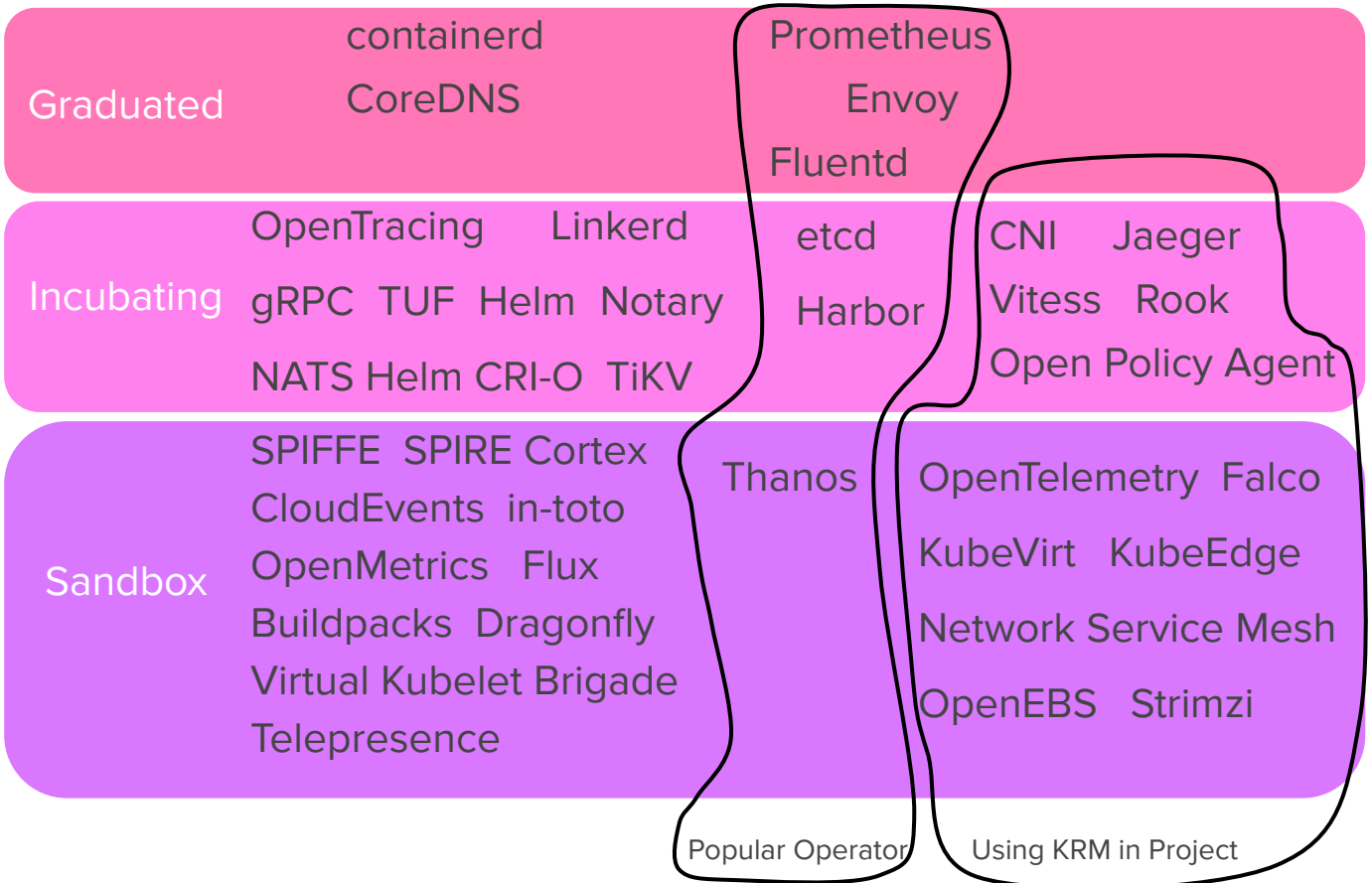
Mesh / Proxy

- Istio
- linkerd
- Kong Kuma
- Traefik

Database Operators

- Kafka (Strimzi)
- PostgreSQL (CrunchyData, Zalando)

CNCF uses KRM



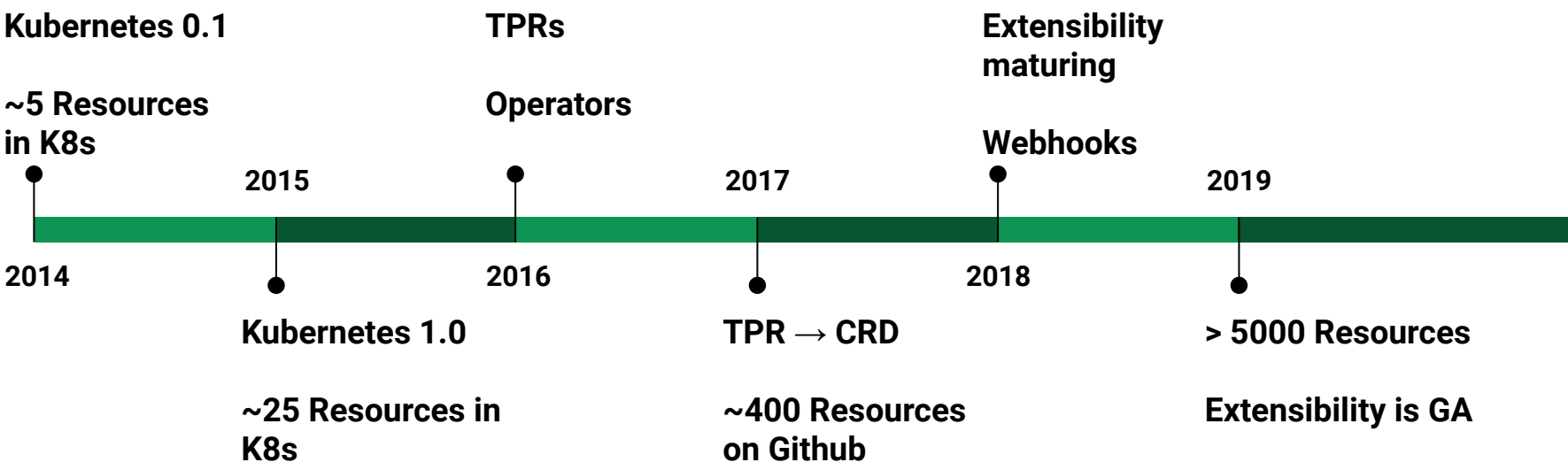
18 Projects

136 Types

CNCF uses KRM

KRM APIs defined in 17 CNCF projects

Another 6 CNCF projects with 3rd party KRM API


Kubernetes 0.1
~5 Resources in K8s
2014
2015
Kubernetes 1.0
~25 Resources in K8s
TPRs
Operators
2016
2017
TPR → CRD
~400 Resources on Github
Extensibility maturing
Webhooks
2018
2019
> 5000 Resources
Extensibility is GA

How to collect Seashells

There is no zoo or museum with all the shells.

Walk on the beach. Pick up everything that looks like a shell.

Sit down. Throw away the stones.

Sort them. Set aside the duplicates.

Study them. Look for common patterns and differences.

How to collect KRM APIs

There is no list with all the APIs.

Search Github for everything that looks related to a KRM API.

Try to parse the files. Throw away ones that don't parse.

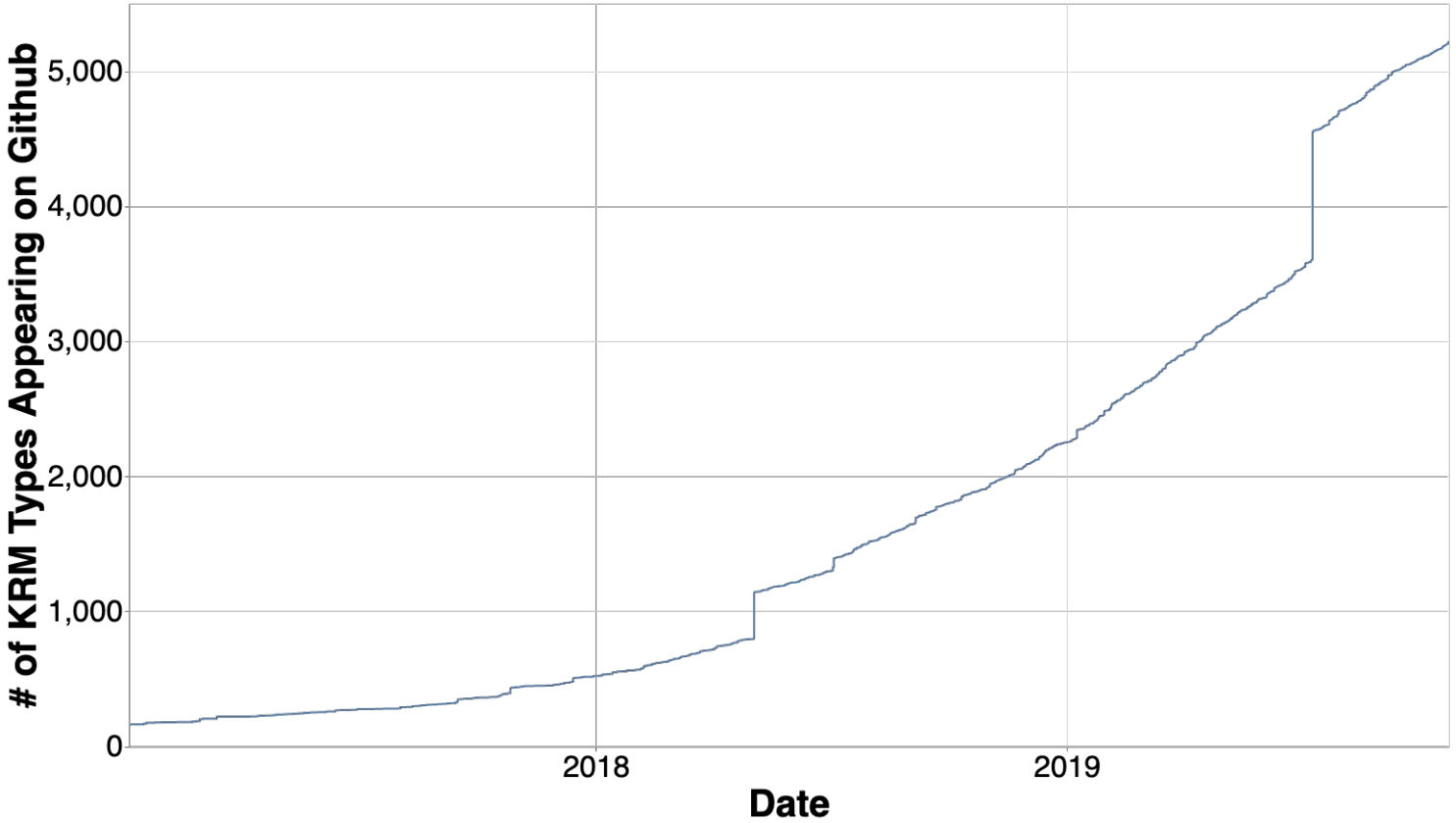
Sort them by API Group and Kind. Set aside the revisions.

Search for patterns in the schemas. Identify patterns.

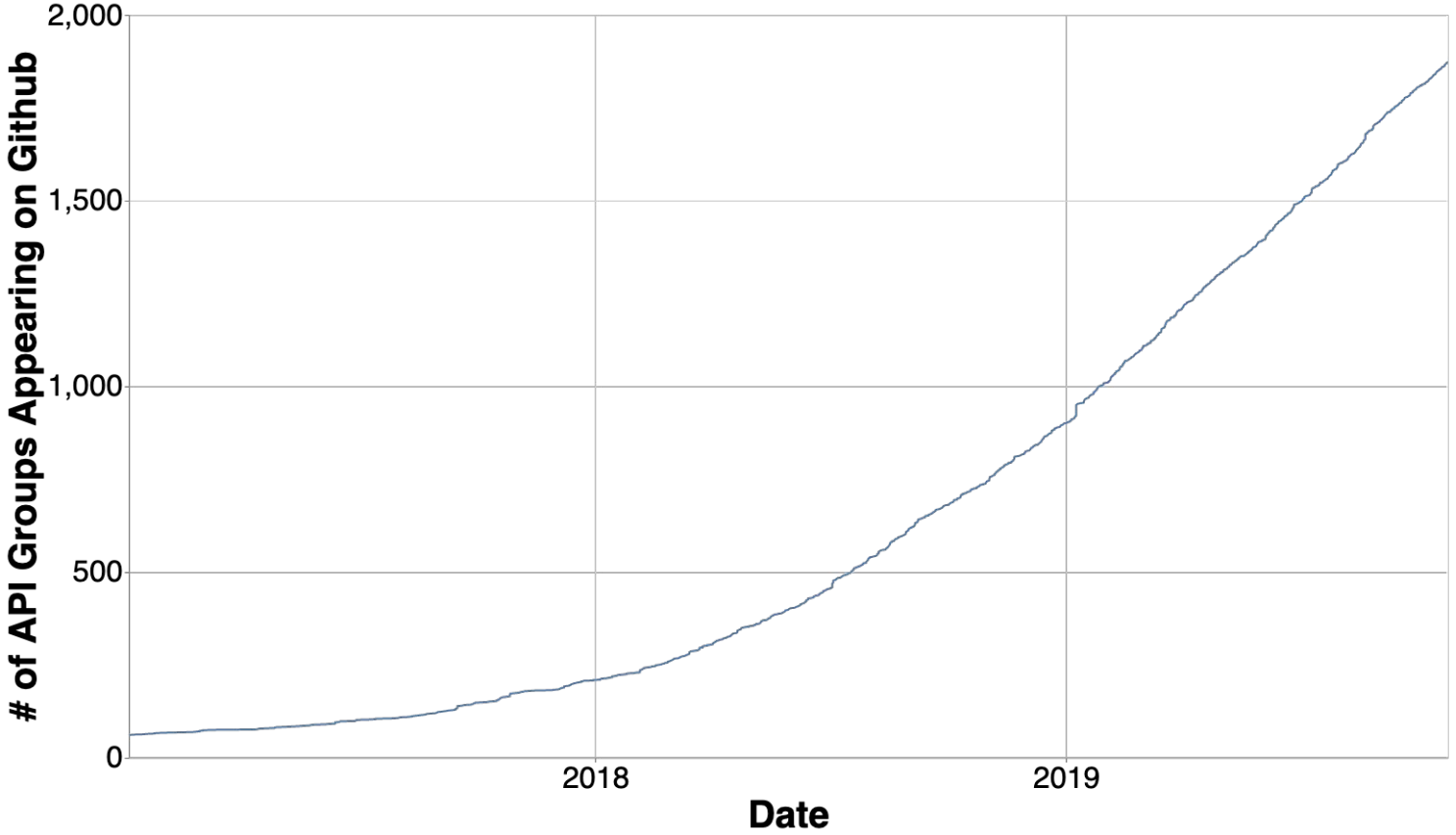
What I found

- 27387 YAML files containing "CustomResourceDefinition"
- Spanning 5690 GitHub Repos
- 126376 KRM objects
- 67672 CRD objects
- 7967 unique CRD objects
- 5605 different (Group, Kind) tuples

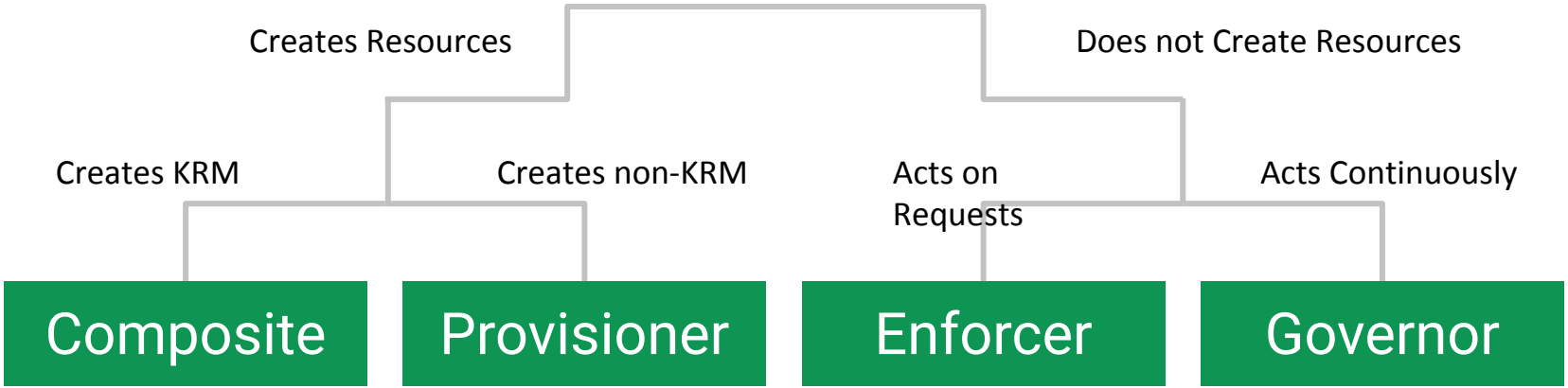
New Types


# of KRM Types Appearing on Github
5,000
4,000
3,000
2,000
1,000
0
2018
2019
Date

New API Groups

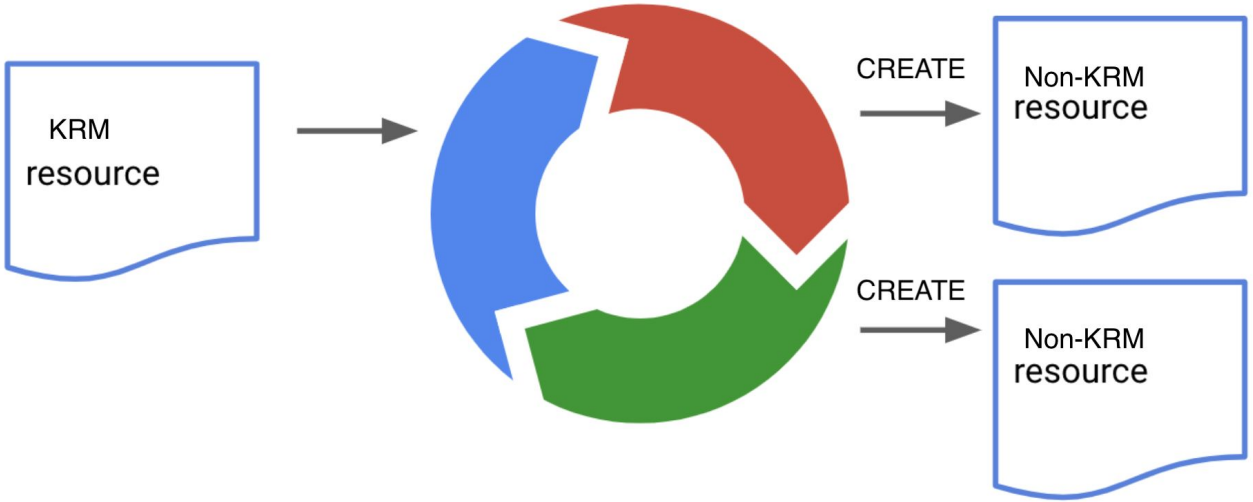


Controller Patterns

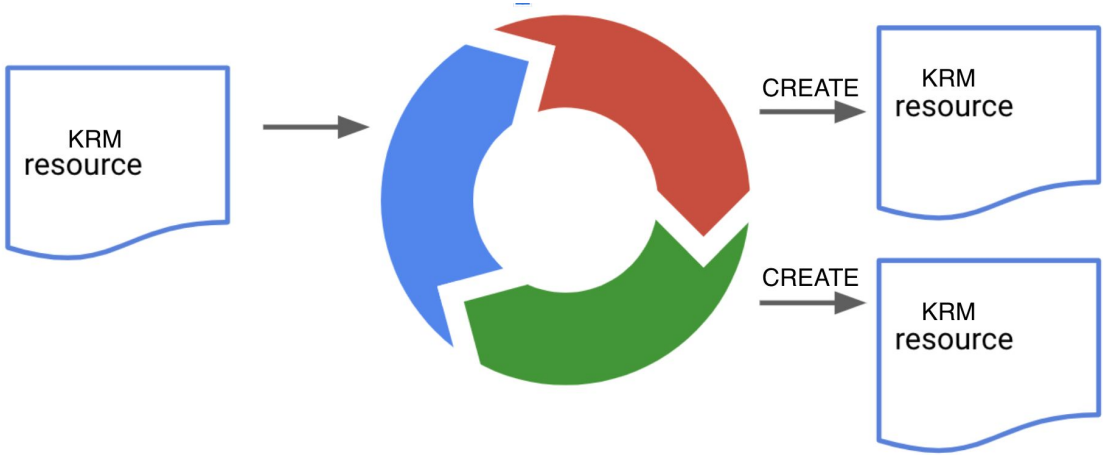


Provisioner Pattern

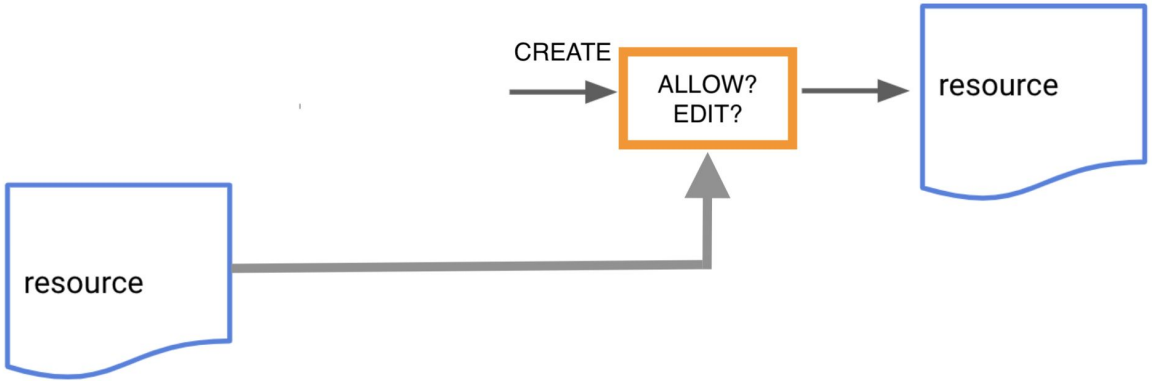
•



Composite Pattern



Enforcer Pattern



Governor Pattern

Pair Patterns

- X & X-Class:
- X & X-Claim
- X & Cluster-X

X / ClusterX

What:

- Two types with the same schema
- X is Namespaced
- ClusterX is not (cluster-scoped)

Why:

- Cluster resources are referenceable by all users of the cluster. Created by Admins
- Namespaced version of the resource is referenceable only within a namespace. Typically can be created by ordinary user.

X / ClusterX

Examples

- Kubernetes:
  - Role / ClusterRole
  - RoleBinding / ClusterRoleBinding.
- Cert-Manager:
  - Issuer / ClusterIssuer
- tekton.dev : Task / ClusterTask
- + 40 more

X / XClaim

What:

- XClaim is created by a user, representing a request.
- XClaim is fulfilled with an X resource.

Why:

- X and XClaim have different lifetimes
- X can be recycled.
- Separate infrastructure provisioning API from implementation.
- Different permissions for X and XClaim.

X / XClaim

What:

When to use it

When not to use it.

Pros/Cons.

Examples of it.

```yaml
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: my-bucket-claim
spec:
  generateBucketName: "my-bucket-"
  storageClassName: noobaa-default-class
   SSL: false
```

X / XClaim

Examples

- Kubernetes:
  - PersistentVolumeClaim → PersistentVolume
- openebs.io
  - BlockDeviceClaim
  - StoragePoolClaim
- Rook
  - ObjectBucketClaim
- 8 other APIs

X / XClass

What?

- XClass holds defaults or preferences for type X.
- X copies the values from XClass when it is created.
-

How?

- Cluster administrator creates an XClass
- Less-privileged users reference XClass by name from X's.
- Either:
  - XClass values copied into X when it is created (Enforcer)
    - e.g. PriorityClass
  - OR XClass affects created resources (Provisioner/Composite)
    - e.g. StorageClass

X / XClass

Type X → refers to an→ XClass

Examples:

- Kubernetes APIs
  - PVC → StorageClass
  - Pod → Runtime Class
  - VolumeSnapshot → VolumeSnapshotClass
- Other APIs using the pattern:
  - Crossplane.io :
    - 17 XClass types
  - Cluster API:
    - Machine → MachineClass
  - SAPCloud.io (Gardner):
    - 5 XClass types
    -

Adoption Levels Vary

• CRDs	5605	100x!
• Extension API Servers	50	
• Validation	3356 of 5605	
• AdditionalPrinterColumns	1676 "	
• Status Subresource	2673 "	
• Status.Conditions	287 "	
• Scale Subresource	57 "	
• ClusterX Pattern	43 "	
• XClass Pattern	28 "	
• XClaim Pattern	8 "	

Summary

- KRM APIs are **not just about containers** and Kubernetes
- Large and Rapidly Growing KRM Ecosystem
  - 2000+ APIs you can easily install and use.
  - Broad range of uses
  - 1000+ expected to be added in next year.
- Consistency across all those APIs
  - Reuse of tooling. Reuse understanding.
- Framework adoption much faster than Pattern adoption.
- If you are building an API, try **http://kubebuilder.io/**
- If you want to hear more: follow me: **http://twitter.com/erictune4**