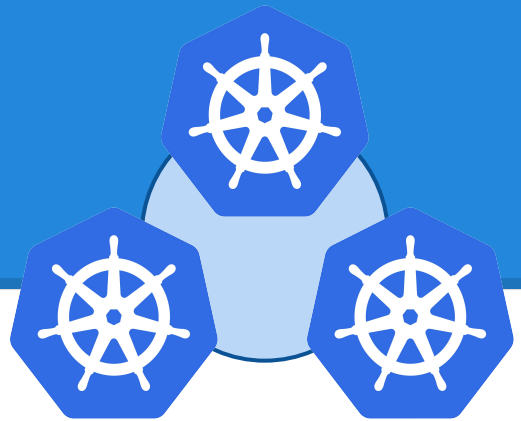
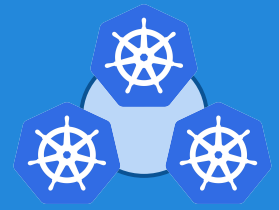
Federated Kubernetes

As a Platform for Distributed Scientific Computing



Kubernetes v1.9 - 03/2018

Who Am I?



Bob Killen

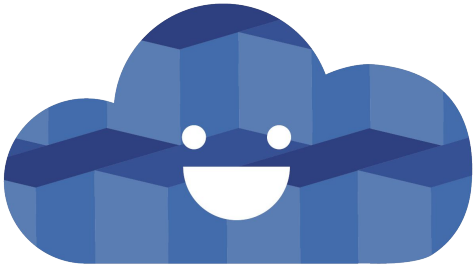
rkillen@umich.edu

Senior Research Cloud Administrator

CNCF Ambassador

Github: @mrbobbytables

Twitter: @mrbobbytables



CLOUD NATIVE
COMPUTING FOUNDATION

AMBASSADOR

SLATE

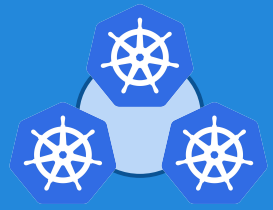
Service Layer at the Edge

slateci.io

Mission: Enable multi-institution computational research by providing a uniform, easy to consume method of access and distribution of research applications.

NSF Award Number: 1724821

The Problem

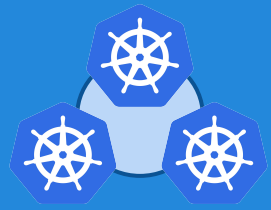


Well...it's hard..and...complicated

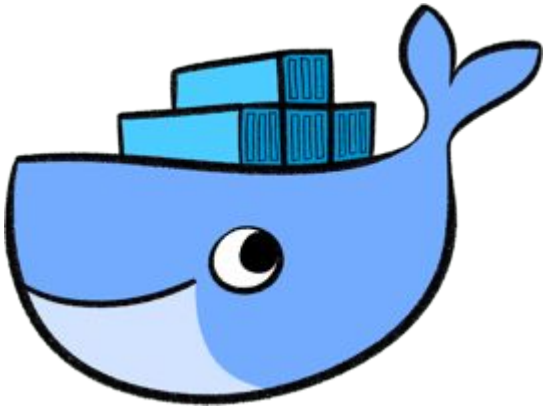
- Packaging
- Distribution
- Orchestration
- AuthN/AuthZ
- Storage
- Monitoring
- Security
- Resource Allocation
- multi-group administration

Across multiple disparate sites and infrastructures

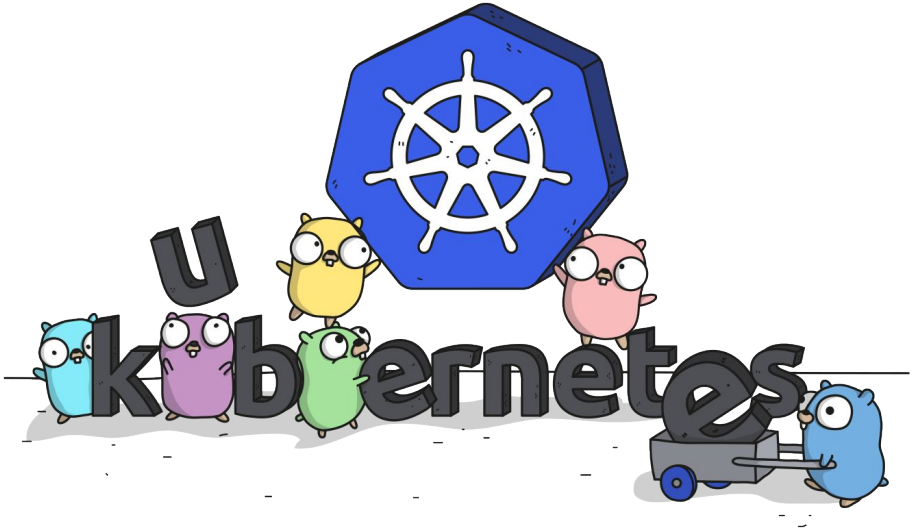
Solution Part 1



Docker and the large scale adoption of containers have (*mostly*) solved the technical aspects of the packaging and distribution problem.

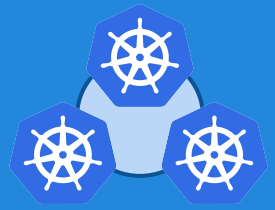


Solution Part 2



Kubernetes and its ecosystem provide the primitives to safely handle the rest.

Why Kubernetes

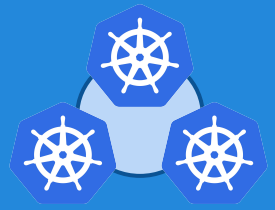


Kubernetes boasts one of the **largest community** backings with a relentlessly fast velocity.

It has been designed from the ground-up as a loosely coupled collection of components centered around deploying, maintaining and scaling a wide variety of workloads.

Importantly, it has the **primitives needed for “clusters of clusters” aka Federation**.

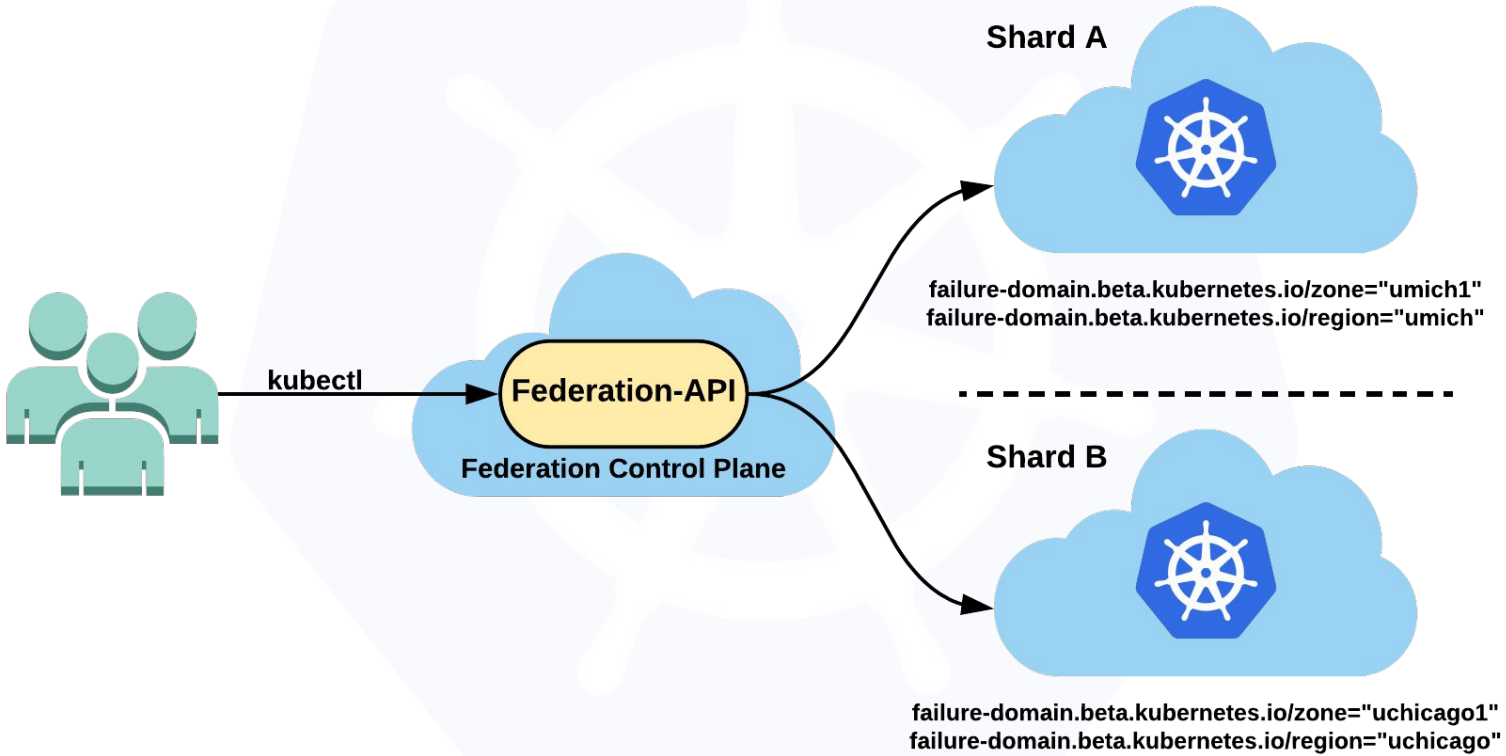
Federation aka “Ubernetes”



Federation extends Kubernetes by providing a **one-to-many** Kubernetes API endpoint.

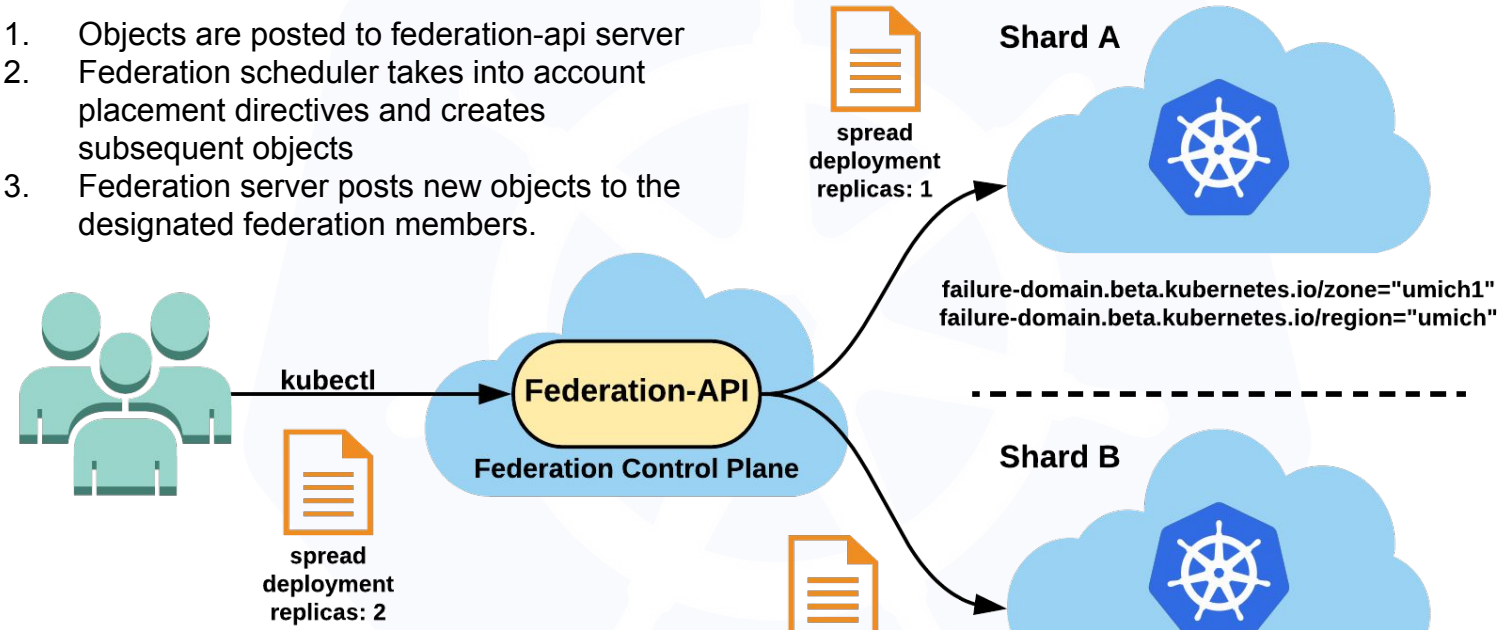
This endpoint is managed by the Federation Control Plane which handles the placement and propagation of the supported Kubernetes objects.

Federated Cluster



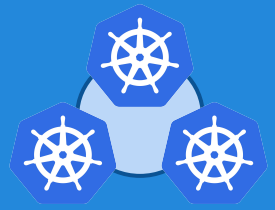
Federated Deployment

1. Objects are posted to federation-api server
2. Federation scheduler takes into account placement directives and creates subsequent objects
3. Federation server posts new objects to the designated federation members.




```
$ kubectl get deploy
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
spread    2         2         2            2           22m
```

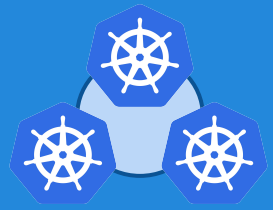
Federation API Server



Just because you can communicate with a Federated Cluster API Server as you would a normal Kubernetes cluster, does not mean you can treat it as a *“normal”* Kubernetes cluster.

The API is **100% Kubernetes compatible**, but does **not support all API types and actions**.

Supported API Types

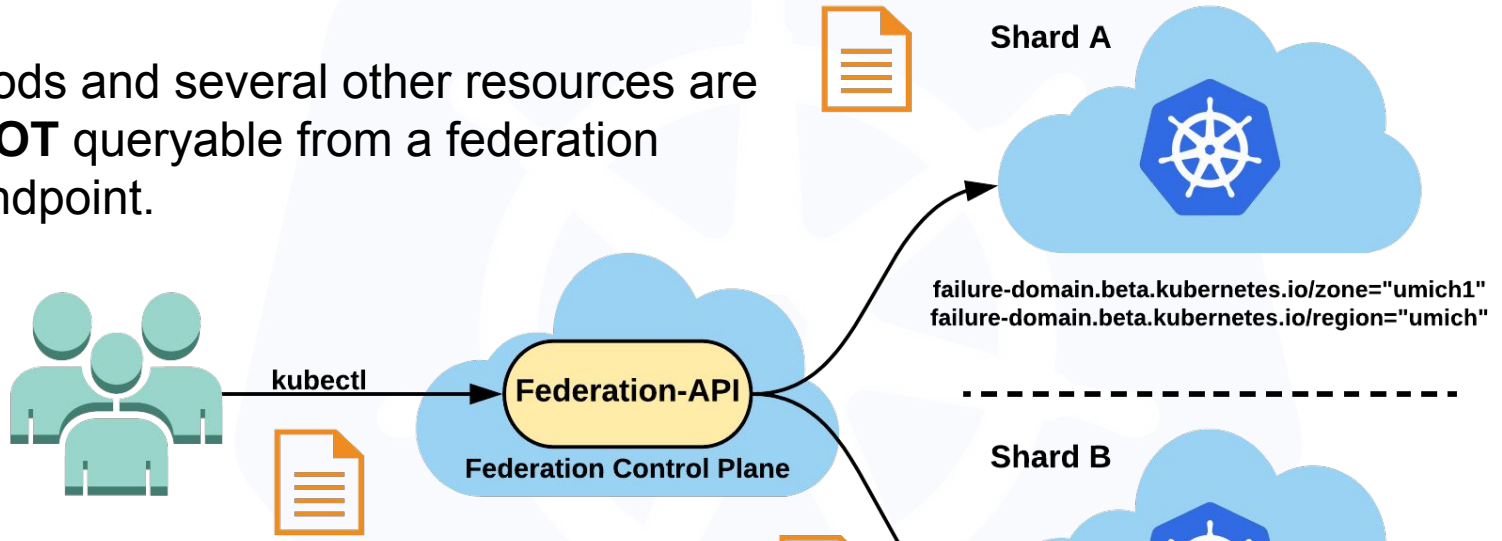


- Cluster
- ConfigMap
- DaemonSet
- Deployment
- Events
- HPA
- Ingress
- Job
- Namespace
- ReplicaSet
- Secret
- Service


WHERES THE POD KUBECTL?
memecrunch.com

Where’s the Pod?

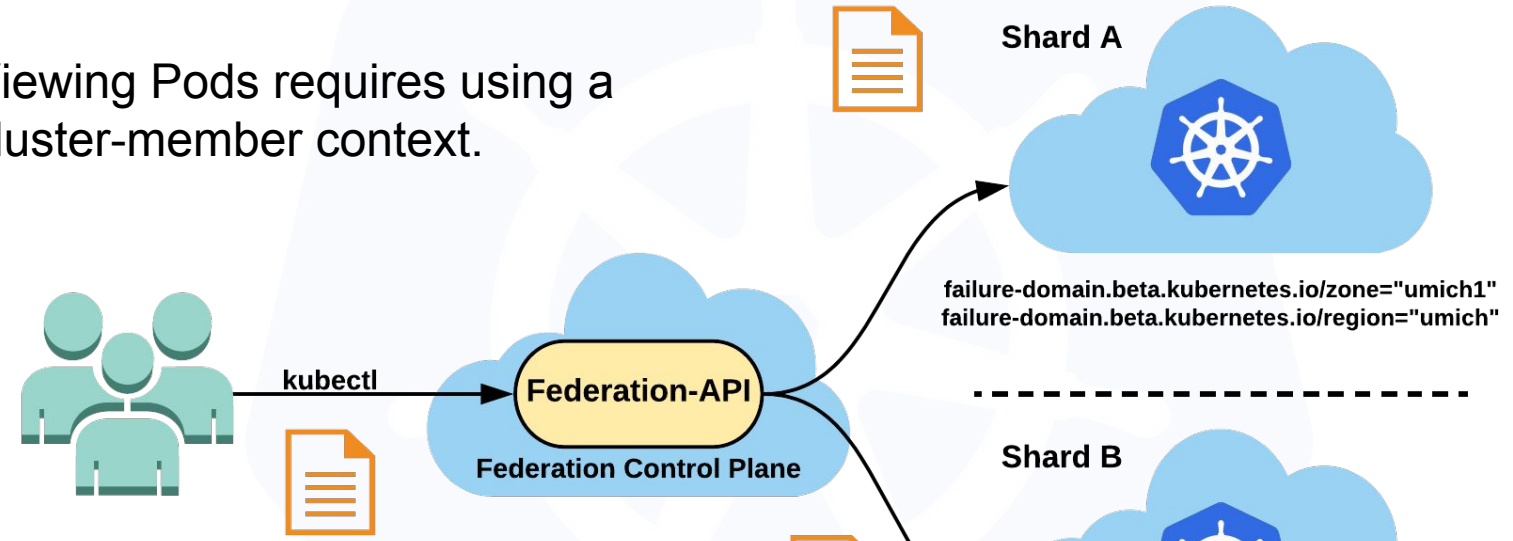
Pods and several other resources are **NOT** queryable from a federation endpoint.




```
$ kubectl get deploy
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
spread    2         2         2            2           40m
$ kubectl get pods
the server doesn't have a resource type "pods"
$ kubectl config use-context umich
Switched to context "umich".
$ kubectl get pods
NAME                      READY     STATUS    RESTARTS   AGE
spread-6bbc9898b9-btc8z   1/1       Running   0          40m
```

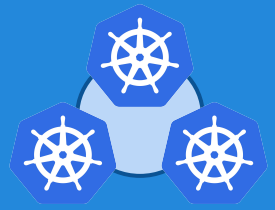
Where’s the Pod?

Viewing Pods requires using a cluster-member context.




```
$ kubectl get deploy
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
spread    2         2         2            2           40m
$ kubectl get pods
the server doesn't have a resource type "pods"
$ kubectl config use-context umich
Switched to context "umich".
$ kubectl get pods
NAME                      READY     STATUS    RESTARTS   AGE
spread-6bbc9898b9-btc8z   1/1       Running   0          40m
```

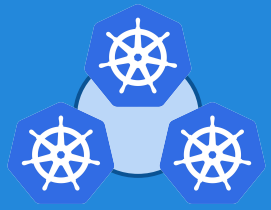
Working with the Federation API



The Federation API Server is better thought of as a deployment endpoint that will handle multi-cluster placement and availability.

Day-to-day operations will always be with the underlying Federation cluster members.

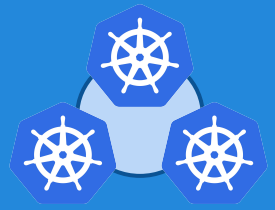
Is That a Show Stopper?



NO!

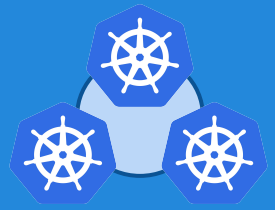
The greater challenge lies in coordinating the disparate sites in a way that their resources may easily be consumed.

Federation Requirements



- Federation Control Plane runs in its **own cluster**
- Kubernetes versions across sites must be **tightly managed**
- Must be backed by a DNS Zone managed in **AWS**, **GCP**, or **CoreDNS**
- Support service type `LoadBalancer`
- **Nodes labeled with**:
  - `failure-domain.beta.kubernetes.io/zone=<zone>`
  - `failure-domain.beta.kubernetes.io/region=<region>`

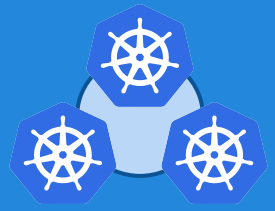
Federation Service Discovery



The DNS Zone and `LoadBalancer` Service Type when combined make the **fulcrum** for cross-cluster service discovery.

Providing the External IPs for those services when bare metal can be a challenge...

External IP Management

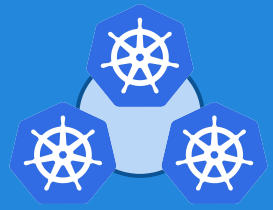


EVERY site must have a pool of available cross-cluster reachable IPs that can be managed by their associated Cluster.

Two tools for enabling this in an on-prem environment:

- **keepalived-cloud-provider**
  https://github.com/munnerz/keepalived-cloud-provider
- **metalLB**
  https://github.com/google/metallb

DNS



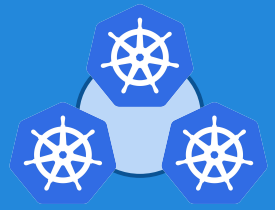
The services that are exposed on those external IPs are given similar names to their in-cluster DNS name.

With additional records being added for each region and zone.

```
<service name>.<namespace>.<federation>.svc.<domain>
<service name>.<namespace>.<federation>.svc.<region>.<domain>
<service name>.<namespace>.<federation>.svc.<zone>.<region>.<domain>

hello.default.myfed.svc.example.com
hello.default.myfed.svc.umich.example.com
hello.default.myfed.svc.umich1.umich.example.com
```

Importance of Consistency

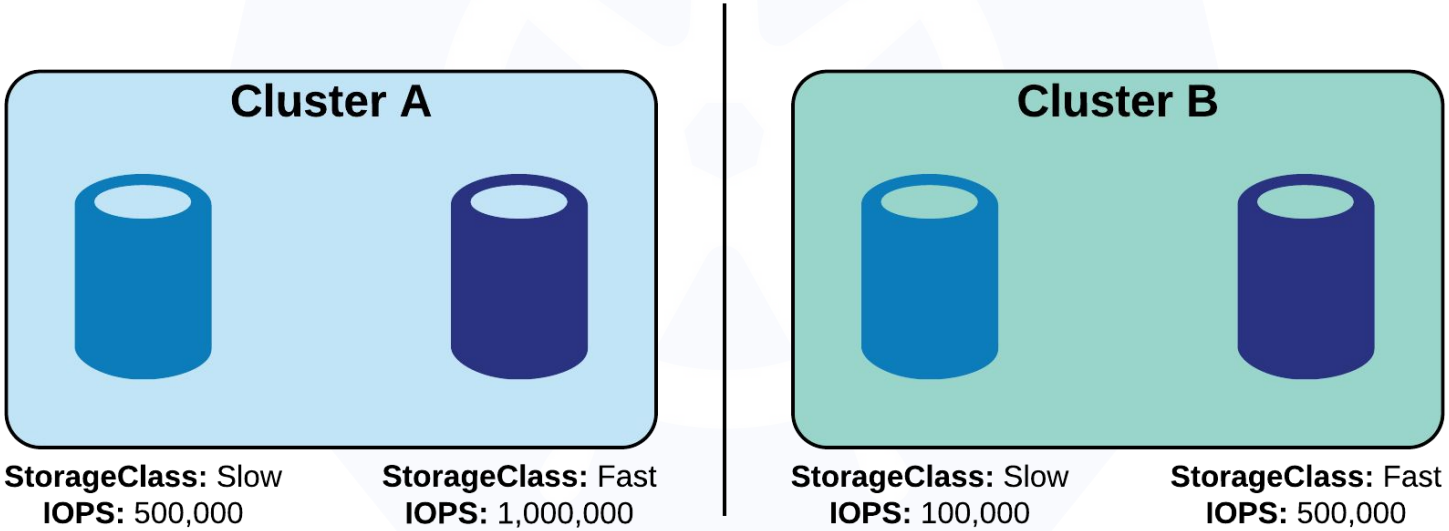


Consistent labeling and naming is **essential** in making federation successful.

Deployments must be able to make reference to resources by their attributes, and their attributes should equate to the same thing across all member clusters.

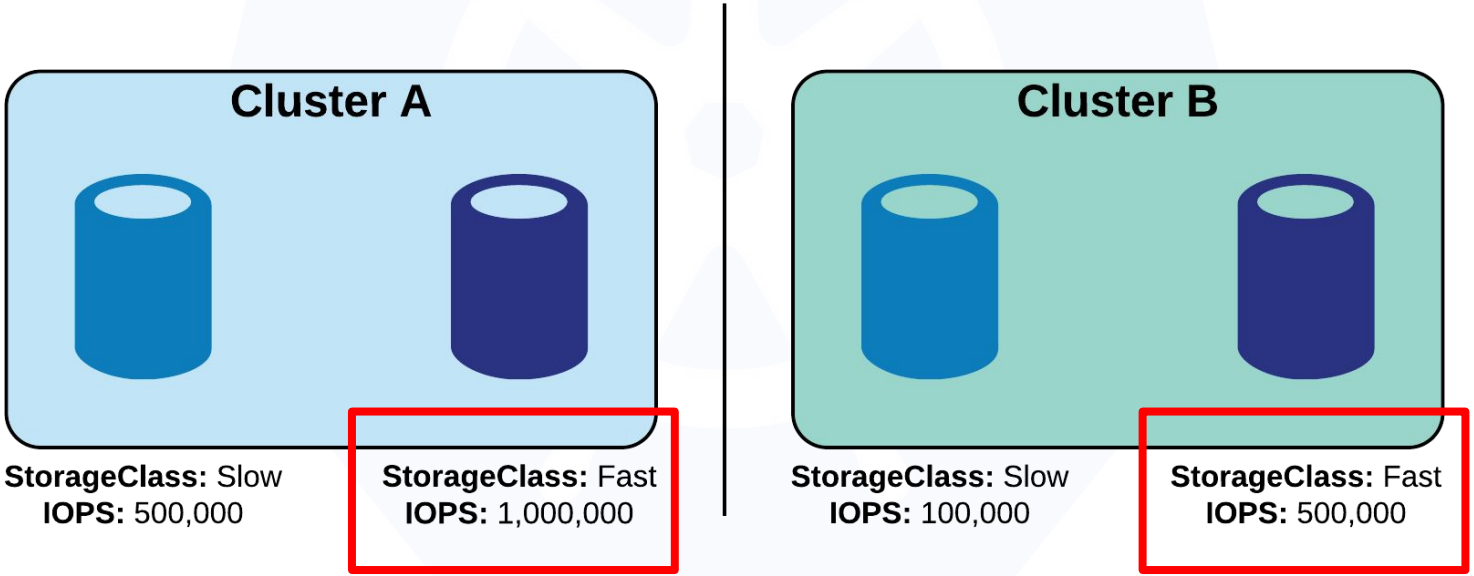
Importance of Consistency

A `PersistentVolumeClaim` targeting `StorageClass` "`Fast`" **may** have vastly different characteristics across the clusters.



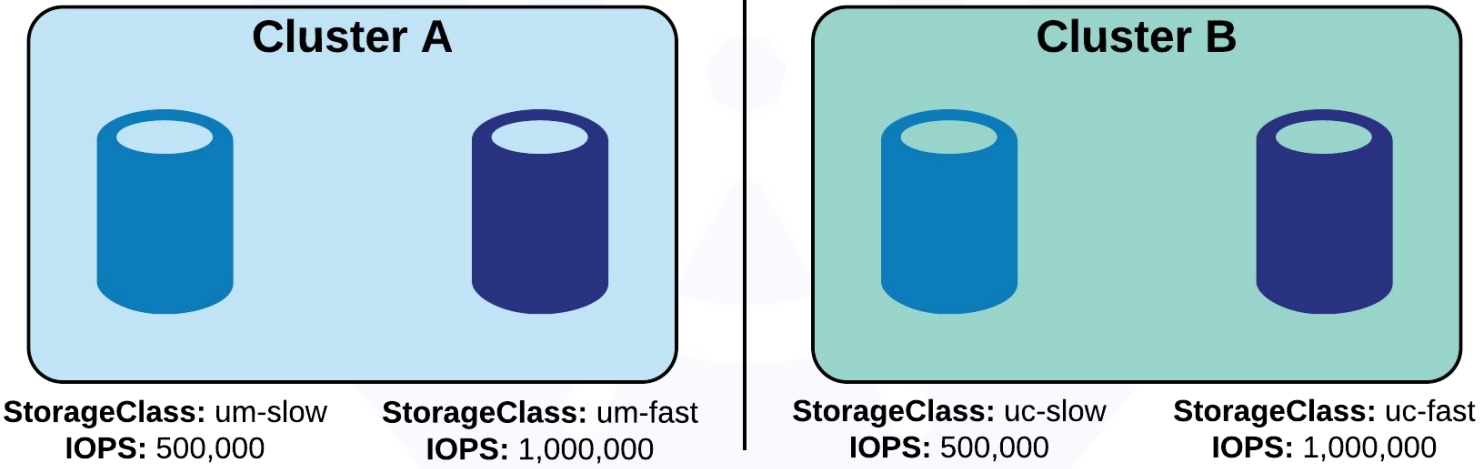
Importance of Consistency

A `PersistentVolumeClaim` targeting `StorageClass` "`Fast`" **may** have vastly different characteristics across the clusters.



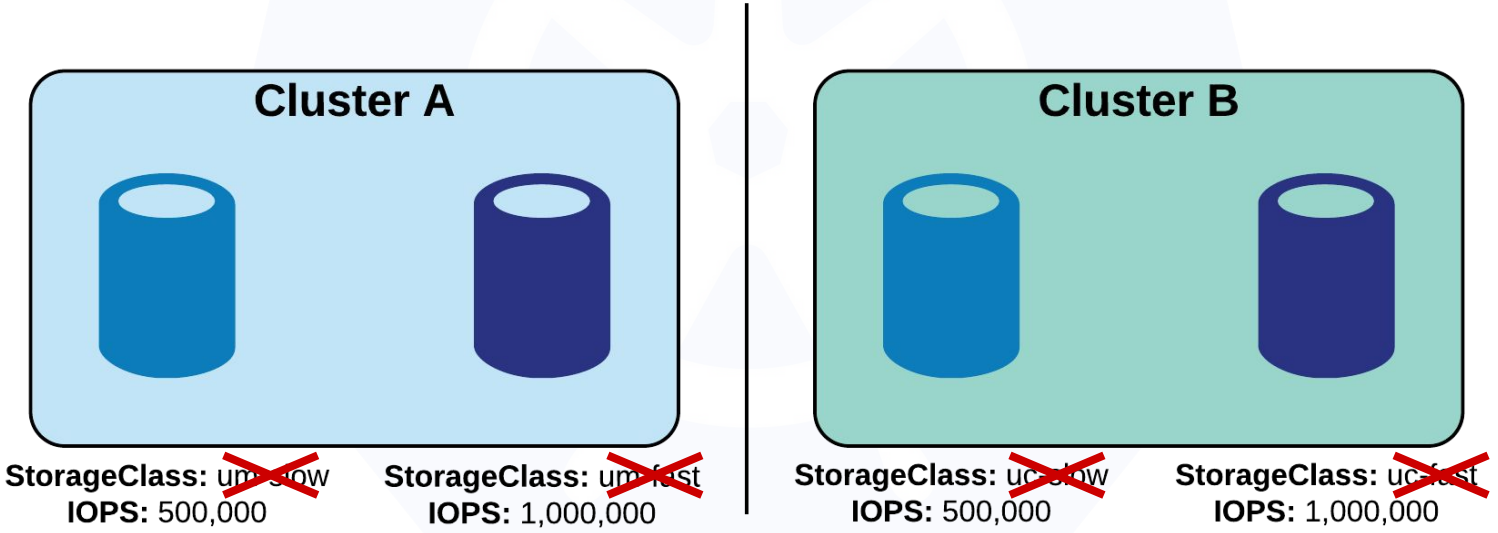
Consistent Labeling

Things will also quickly become unmanageable if site-specific data is forced into names.



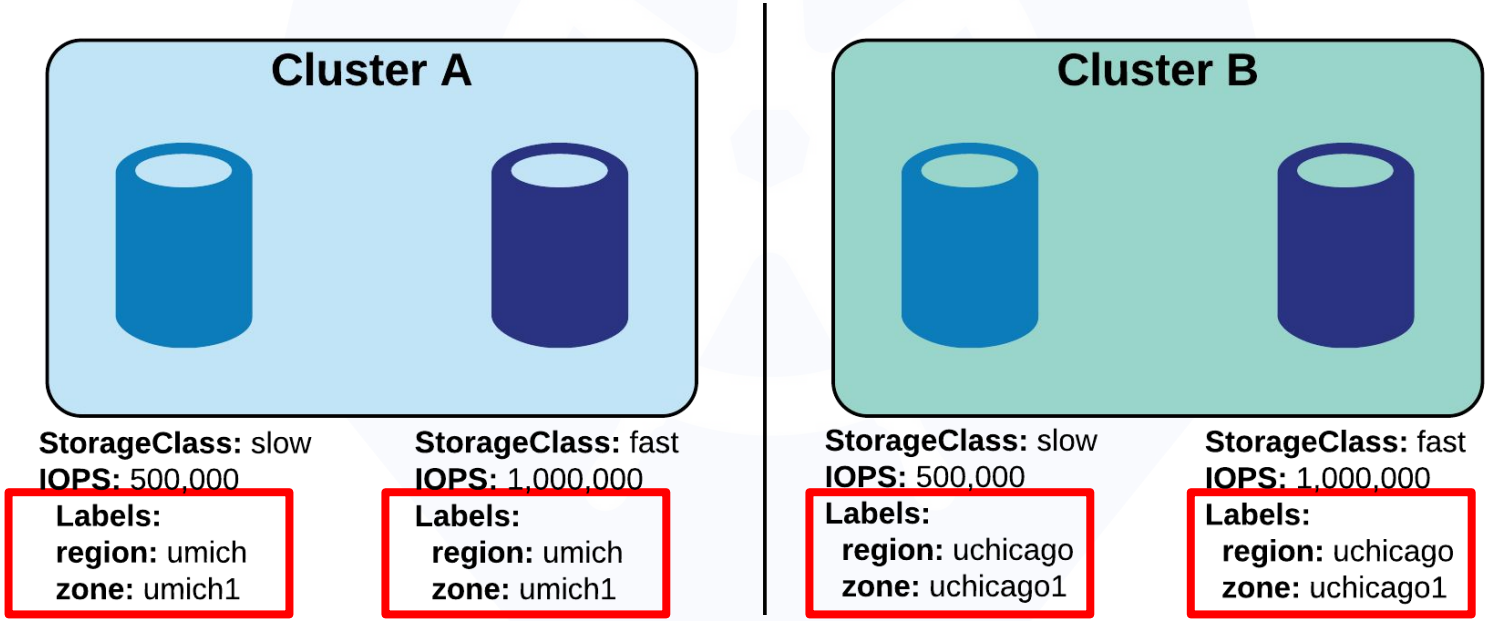
Consistent Labeling

Things will also quickly become unmanageable if site-specific data is forced into names.

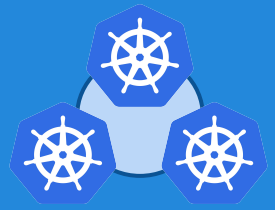


Consistent Labeling

The group managing the Federation should have a predetermined set of labels for **ALL** resources that may impact resource selection.



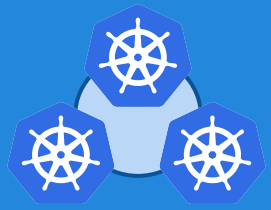
A Word on Placement



Workload locality **should still be considered** with preferences weighted to the most desirable location.

ALWAYS account for cross cluster services that may send traffic to another unencrypted.

End User (Researcher) Experience



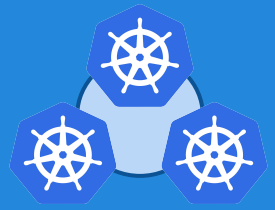
Research App User

- Does not want to use `kubectl`
- Does not want to have to think about placement.. resources.. security..etc
- Does want to get up and going quickly
- Does want it to "*just work*"

Research App Publisher

- Wants **CONSISTENCY** above all else
- Want to iterate application deployment design quickly
- An easy method to package entire app stacks

Solution: Helm

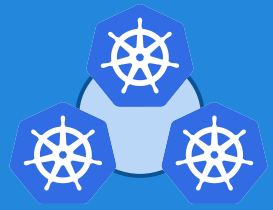


Helm is a Kubernetes Package Manager, that allows you to package up entire application stacks into “*Charts*”.

A Chart is a collection of templates and files that describe the stack you wish to deploy.

Helm also is one of the few tools that is Federation aware: https://github.com/kubernetes-helm/rudder-federation

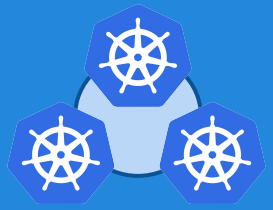
Logging and Monitoring



Logging and Monitoring are important to both the Research End User and the Administrator as a support tool.

Logs and metrics should be aggregated to a central location to give both a single pane-of-glass view of their resources.

Logging and Monitoring

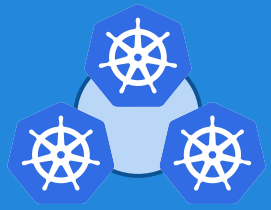


Popular Log Shipper with native Kubernetes integration and a massive plugin ecosystem.

Prometheus

Defacto standard for Kubernetes monitoring with full aggregation/federation support.

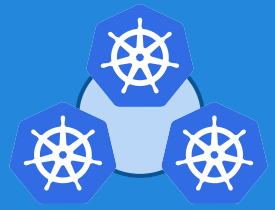
Administration



Being a part of a Federation implicitly means a high level of trust is needed.

A full federation administrator is essentially a cluster admin in **EVERY** cluster.

Future

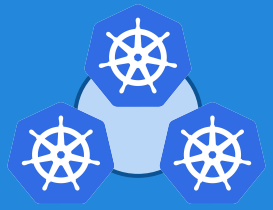


Federation v2 is well under development with the lessons learned from developing and managing v1.

Cluster discovery is being moved into the Kubernetes Cluster Registry.

Policy based placement becomes a first class citizen.

Useful Links



- **minifed - minikube + federation = minifed**
  https://github.com/slateci/minifed
- **Kubernetes Federation - The Hard Way**
  https://github.com/kelseyhightower/kubernetes-cluster-federation
- **Kubecon Europe 2017 Keynote: Federation**
  https://www.youtube.com/watch?v=kwOvOLnFYck
- **SIG Multicluster - Home of Federation**
  https://github.com/kubernetes/community/tree/master/sig-multicluster

Questions?