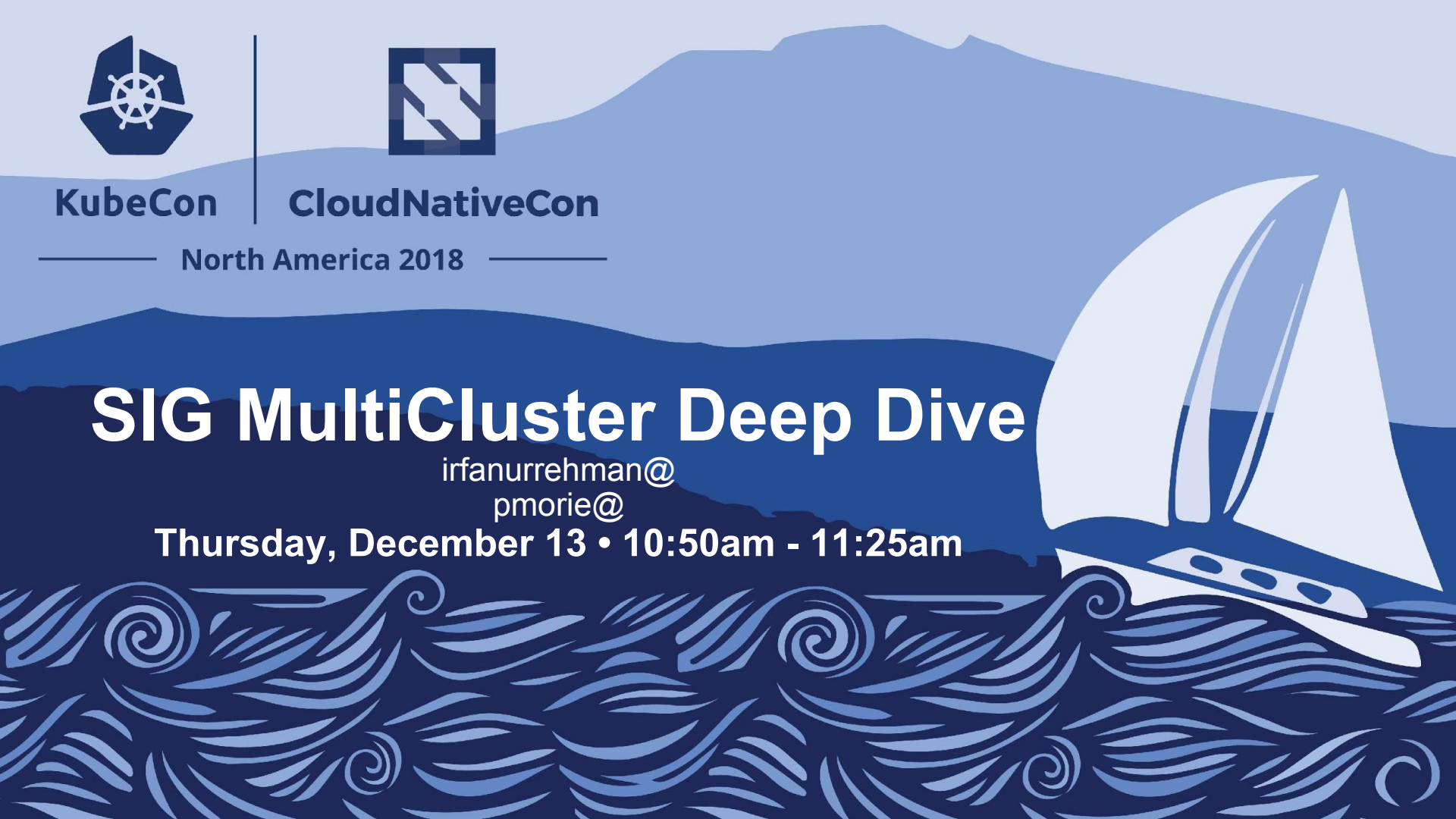


SIG MultiCluster Deep Dive

irfanurrehman@
pmorie@

Thursday, December 13 • 10:50am - 11:25am

Agenda

- Mission
- Sub-projects: Current status
- Federation v2
- Deep dive into federation v2 API
- Deep dive into federation v2 concepts and architecture
- Q&A

SIG-Multicluster Mission

- Solving common challenges related to the **management of multiple Kubernetes clusters, and applications that exist therein**
- Designing, discussing, implementing and maintaining
  - **API's, tools and documentation**
  - related to multi-cluster administration and application management
- Includes **not only active automated approaches** such as Cluster Federation
  - also those that employ batch workflow-style continuous deployment systems
- Includes:
  - standalone building blocks (for example a cluster registry), and
  - proposed changes to kubernetes core where appropriate
- See more at https://github.com/kubernetes/community/blob/master/sig-multicluster/README.md

SIG-Multicluster Sub-projects

- **Federation v2**
  - Control Plane for Multicluster-specific APIs.
    - Currently supports both:
      - Propagation of Kubernetes objects to multiple clusters
      - Higher Level Features (e.g. cross cluster replica distribution for deployments, service discovery, load balancing etc
    - Plan to Beta in Q1 2019.
- **Cluster Registry**
  - Common abstraction for a Registry of Clusters that can store per-Cluster metadata.
    - Deployed to an API server as a CRD.
- **Kubemci (Kube Multi-cluster Ingress)**
  - Standalone tool to create ingress with load balancing across multiple clusters
    - Similar functionality to Federation v1 Federated Ingress
    - Currently only supports Google Cloud, but can be expanded to others.

Sub-project: Federation original goals

- **Capacity Overflow**
  - What happens if I run out of capacity in my cluster.
- **Sensitive Workloads**
  - I have multiple clusters but want to run sensitive workloads only in specific clusters.
- **Vendor lock-in avoidance**
  - Run workloads in multiple service providers clusters.
- **HA**
  - Single region outage does not impact the availability of workloads.

Sub-project: Federation v1 to v2

- Today's world is different
- CRDs change equation significantly

Sub-project: Federation v2 now

- Building on the definition of federation as a common API surface to multiple kubernetes clusters.
- CRD based API implementation of federation features
  - https://github.com/kubernetes-sigs/federation-v2

- Allows simple federation of any k8s type, via configuration, including CRDs:
  - We achieve this using **kubefed2 federate** (details later).
- Uses the Cluster Registry as a source of Kubernetes Cluster Endpoints.

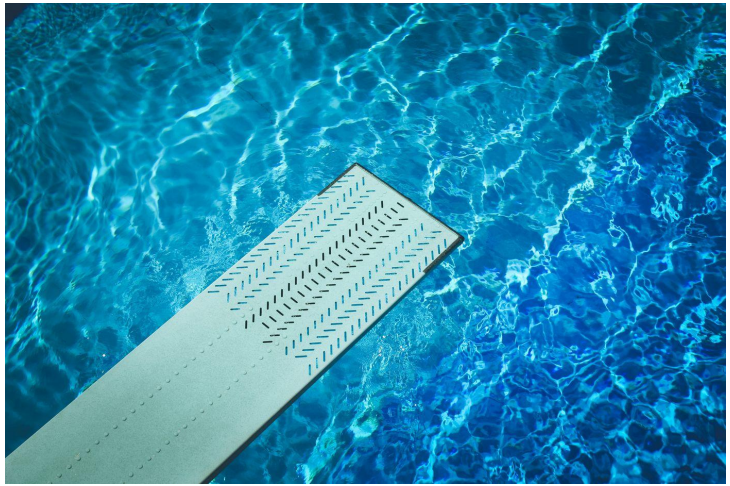
Use Cases Supported Today

- Federate any k8s API resource without writing code
- Unified workload deployment across multiple clusters with active reconciliation and cluster specific customizations
- Customise (override fields) resources per cluster

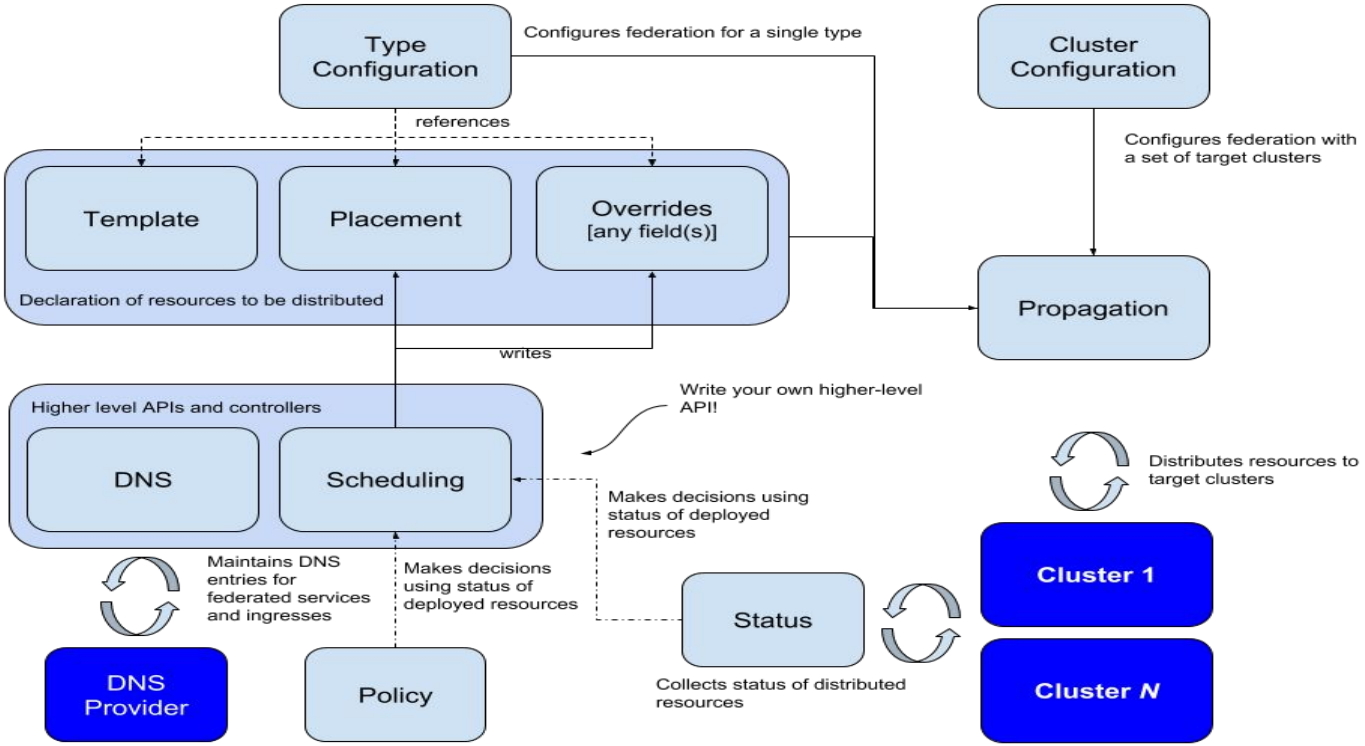
- Cross cluster service discovery, service failover across clusters
- Distribution and dynamic rebalancing of replica workloads across clusters

- Namespaced federation:
  - Allow multiple users to federate same clusters
  - Deploy multiple control planes in same cluster

Federation v2 deep dive

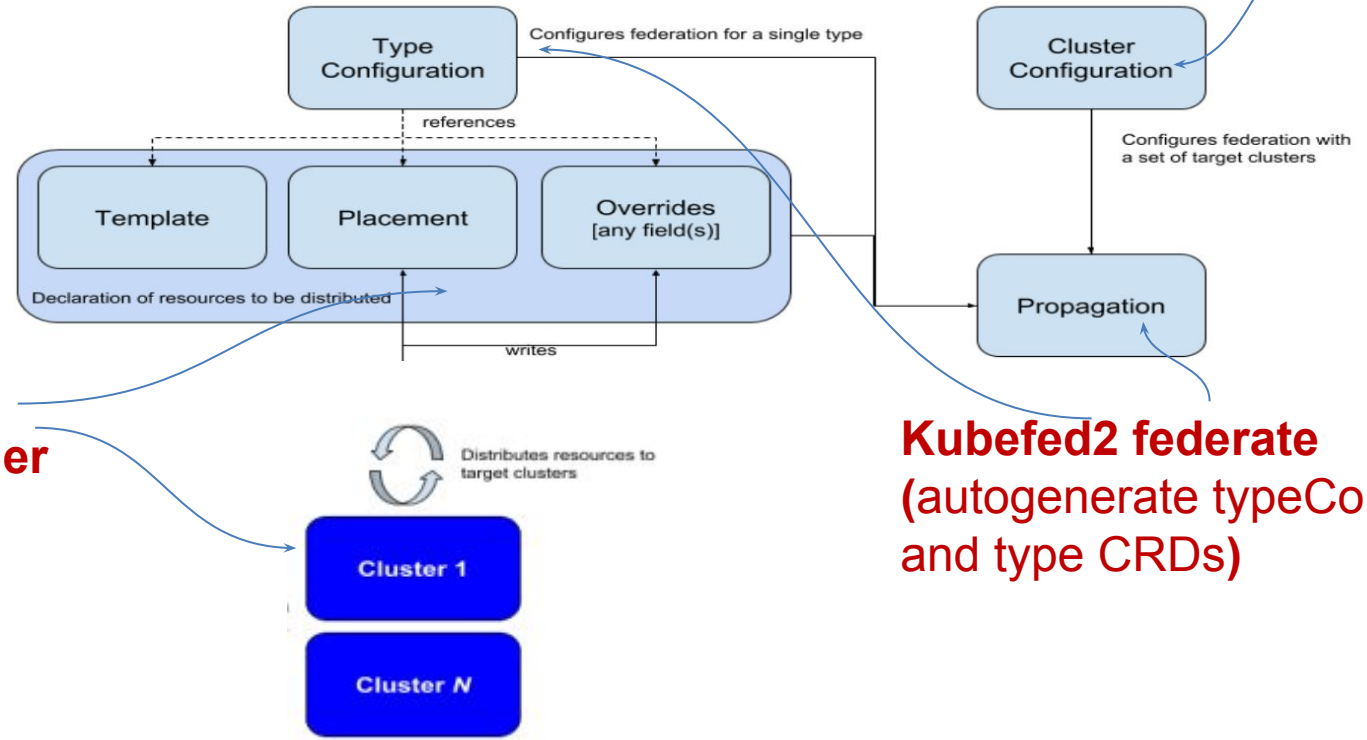


Federation v2 concepts



Federation v2 concepts

kubefed2 join/unjoin



sync controller

Kubefed2 federate (autogenerate typeConfig and type CRDs)

FederatedTypeConfig

```
apiVersion: core.federation.k8s.io/v1alpha1
kind: FederatedTypeConfig
metadata:
  name: deployments.apps
  namespace: federation-system
spec:
  namespaced: true
  target:
    kind: Deployment
    version: v1
  template:
    group: core.federation.k8s.io
    kind: FederatedDeployment
  override:
    group: core.federation.k8s.io
    kind: FederatedDeploymentOverride
  placement:
    group: core.federation.k8s.io
    kind: FederatedDeploymentPlacement
  propagationEnabled: true
```

FederatedDeployment (**template**) type CRD

```
apiVersion:
apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name:
federateddeployments.primitives.federation.k8s.io
spec:
  group: primitives.federation.k8s.io
  names:
    kind: FederatedDeployment
    plural: federateddeployments
  scope: Namespaced
  version: v1alpha1
  validation:
    …….
```

Federated**DeploymentPlacement** type CRD

Federated**DeploymentOverride** type CRD

FederatedDeployment
(**template object**)

```
apiVersion:
core.federation.k8s.io/v1alpha1
kind: FederatedDeployment
metadata:
  name: test-deployment
  namespace: test-namespace
spec:
  template:
    spec:
      replicas: 3
      template:
        spec:
          containers:
          …….
          …….
```

FederatedDeploymentPlacement (**placement object**)

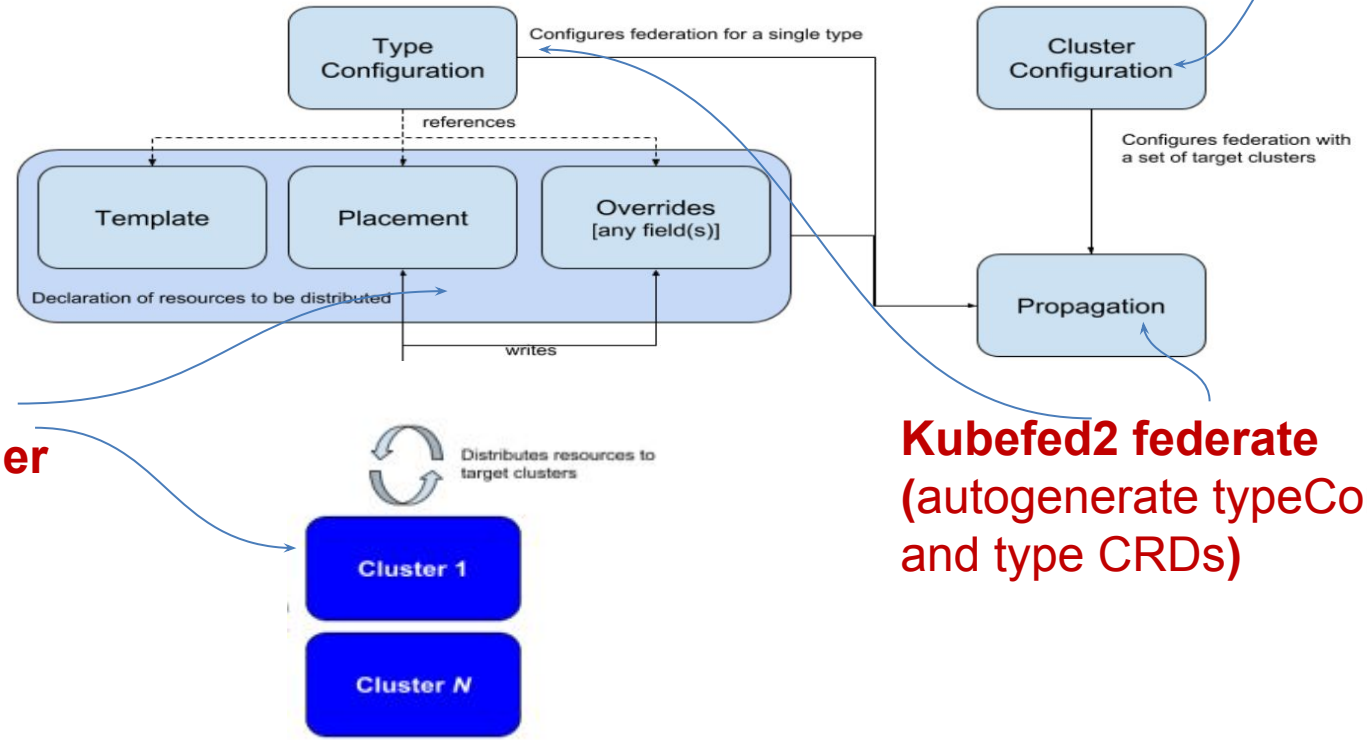
```
apiVersion:
core.federation.k8s.io/v1alpha1
kind: FederatedDeploymentPlacement
metadata:
  name: test-deployment
  namespace: test-namespace
spec:
  clusterNames:
  - cluster2
  - cluster1
```

FederatedDeploymentOverride (**override object**)

```
apiVersion:
core.federation.k8s.io/v1alpha1
kind: FederatedDeploymentOverride
metadata:
  name: test-deployment
  namespace: test-namespace
spec:
- clusterOverrides:
    - clusterName: cluster2
      path: spec.replicas
      value: 2
```

Federation v2 concepts

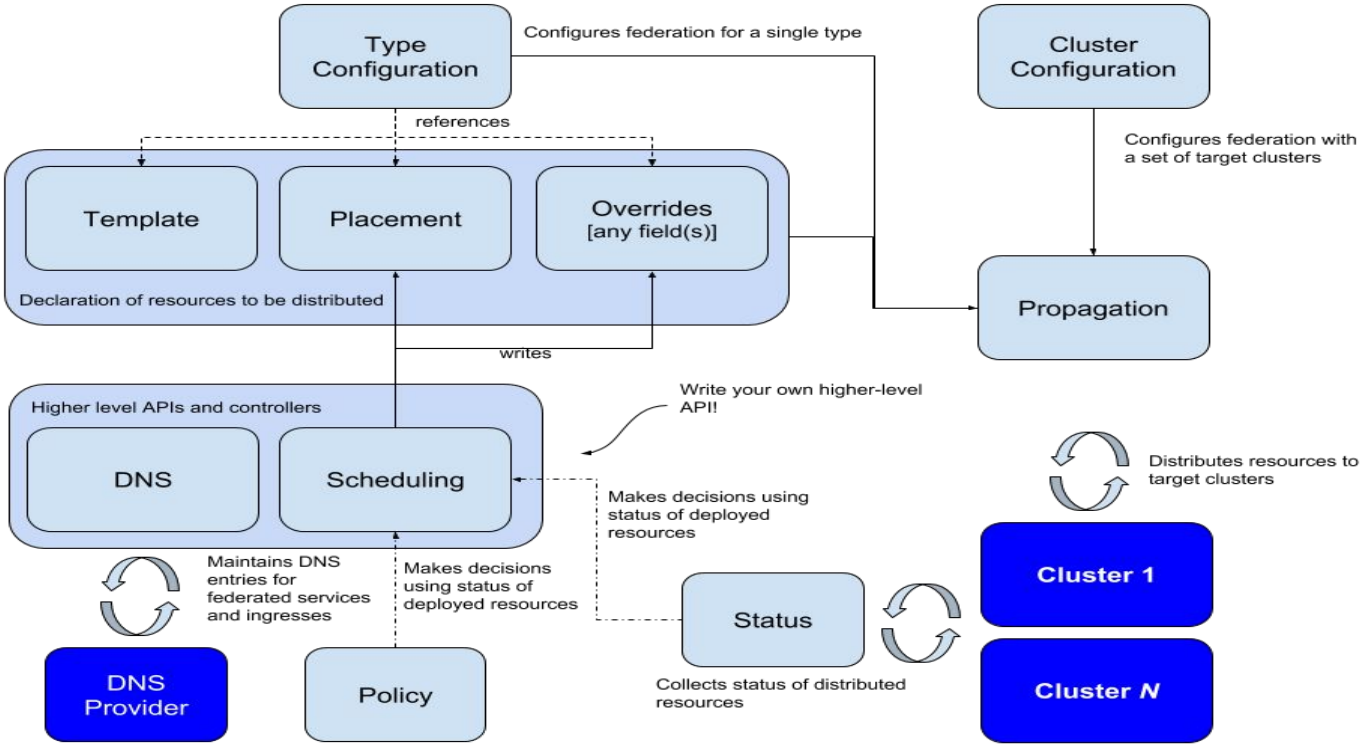
kubefed2 join/unjoin



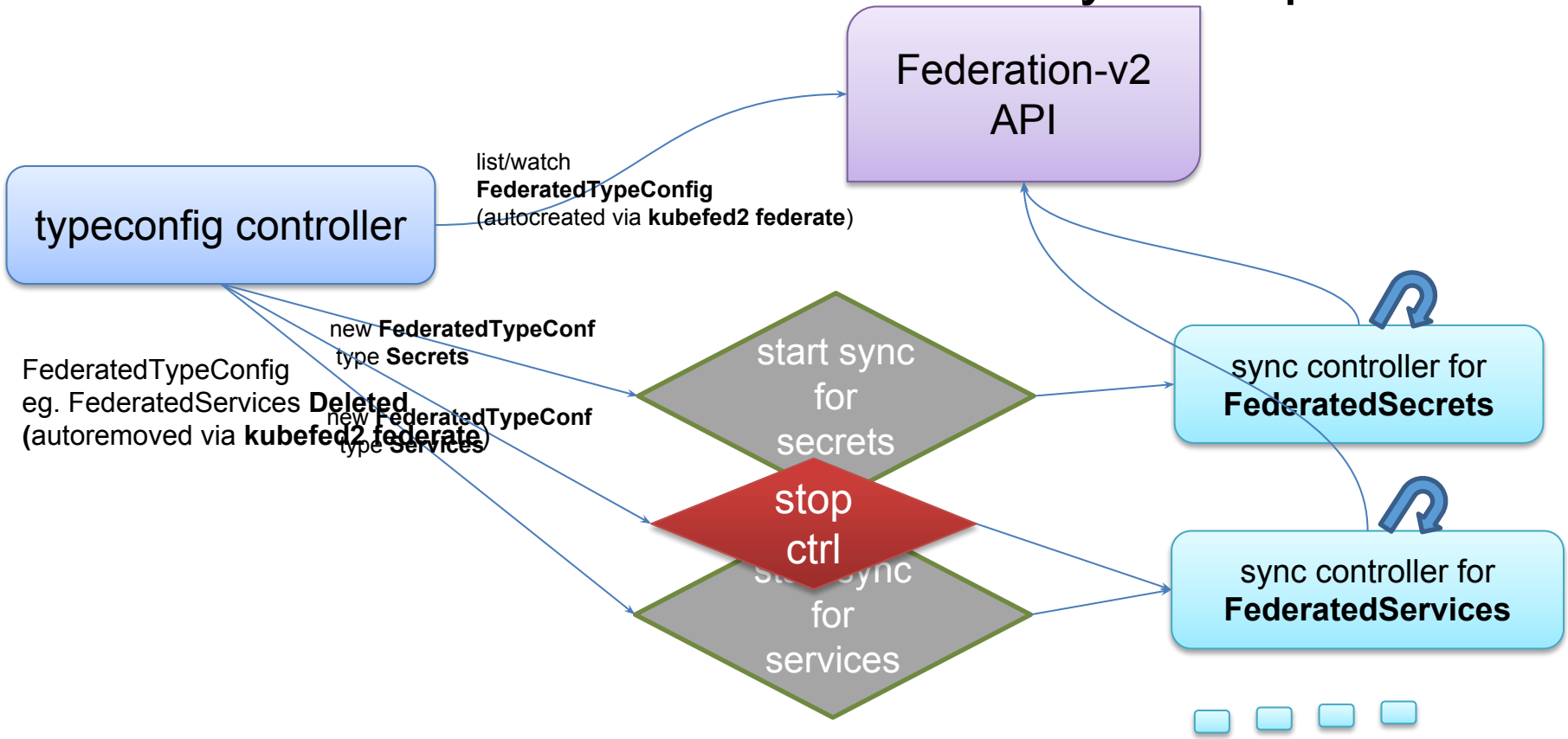
sync controller

Kubefed2 federate (autogenerate typeConfig and type CRDs)

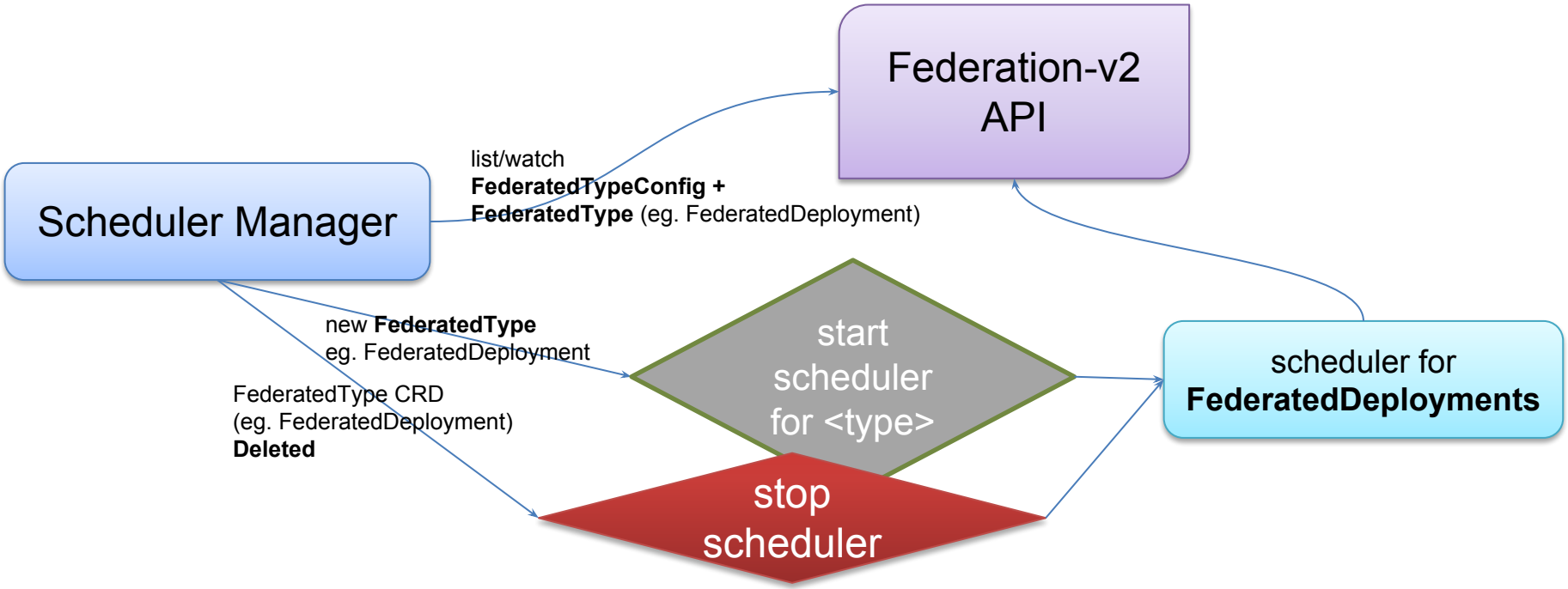
Federation v2 concepts



Primitives controllers nuts and bolts: sync loop



Schedulers: nuts and bolts



Federation V2: API grouping

- **core**
  - FederatedCluster
  - FederatedTypeConfig
  - PropagatedVersion
- **primitives (autogenerated)**
  - FederatedXXXX (template)
  - FederatedXXXXPlacement
  - FederatedXXXXOverrides
  - ….
- **scheduling**
  - ReplicaSchedulingPreference
  - JobSchedulingPreferences
- **multiclusterdns**
  - DNSEndpoint
  - IngressDNSRecord
  - ServiceDNSRecord

Federation V2: Resource naming scheme

- **primitives (example deployment)**
  - FederatedDeployment - **myns/my-dep**
  - FederatedDeploymentPlacement - **myns/my-dep**
  - FederatedDeploymentOverrides - **myns/my-dep**
- **scheduling**
  - ReplicaSchedulingPreference - **myns/my-dep**
- **multiclusterdns**
  - DNSEndpoint - **myns/service-my-svc || myns/service-my-svc**
  - IngressDNSRecord - **myns/my-svc**
  - ServiceDNSRecord - **myns/my-svc**

Federation V2: Federate without code

kubefed2 federate/delete type

Example:

- **`kubefed2 federate type deployments`**
  - create **Typeconfig** for deployments.extensions and enables sync.
  - create **FederatedDeployments** CRD resource.
  - create **FederatedDeploymentPlacement** CRD resource.

- **`kubefed2 disable type deployments`**
- **`kubefed2 delete type deployments`**

Federation V2: Federate without code..

Next step *(in pipeline)*

kubefed2 federate resource <typeName> <resourceName>

- eg.
  - **`kubefed2 federate resource deployment my-deployment`**
    - creates **FederatedDeployment** with .template = my-deployment
    - create **FederatedDeploymentPlacement** with cluster-list = <all clusters>
  - **`kubefed2 federate resource <type> <name> -o yaml`**
    - would also generate federated yamls for existing k8s resource.
  - **`kubefed2 federate resources -i yaml -o yaml`**
    - convert existing k8s manifests to default federated manifests.

Federation V2: In pipeline

- **Usability**
  - Tooling to ease translating a k8s resource into federated types
  - Merge multiple API resources?
  - Higher level user facing API
  - Status aggregation (simple version already available)
    - Individual per cluster
    - Consolidated cross-cluster
- More high level scheduling behaviours
- Pull reconciliation

Q&A