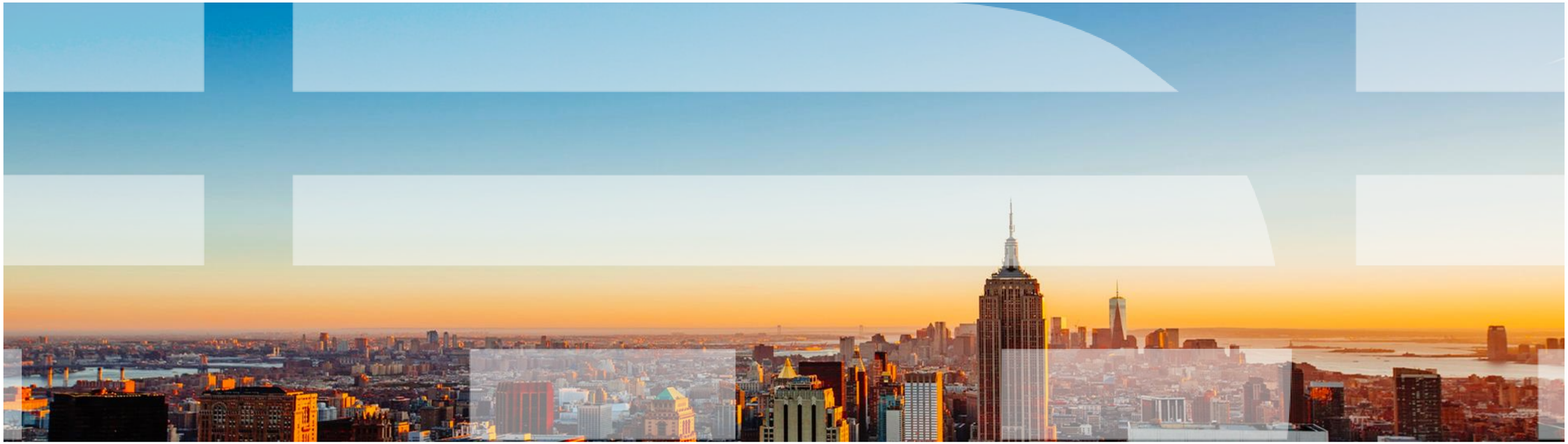
Federation of Kubernetes Clusters



About Me

Principal Architect @TenxCloud

https://github.com/JoshuaAndrew
https://blog.csdn.net/chweiweich

Stay in Touch!

@Joshuaweiwei

@CocoonJoshua

Agenda

1. Federation community current situation

   1.1> federation v1

   1.2> federation v2

2. Federation problems to be solved urgently

   2.1> Global Load balancing

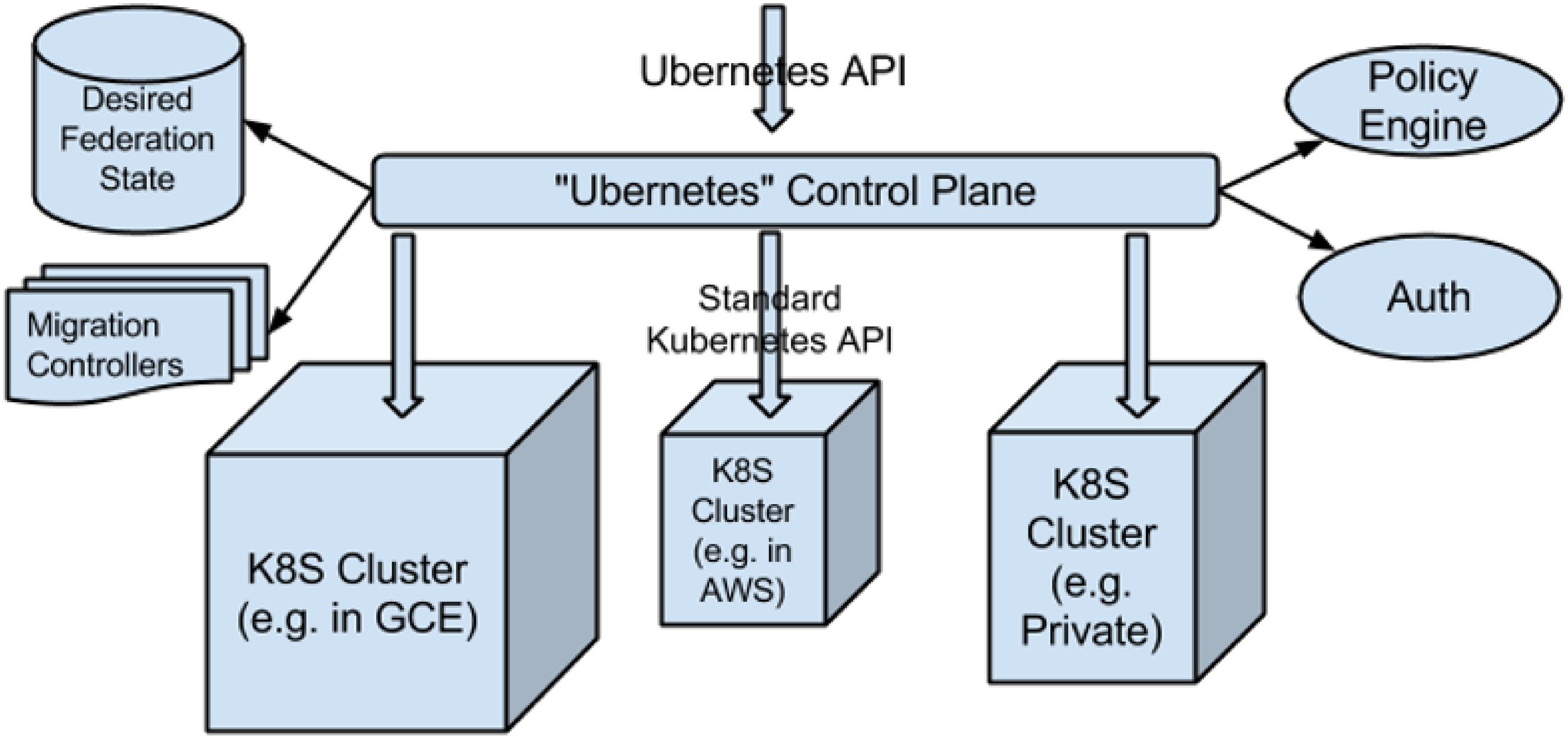
   2.2> Global Distributed Storage

Federation

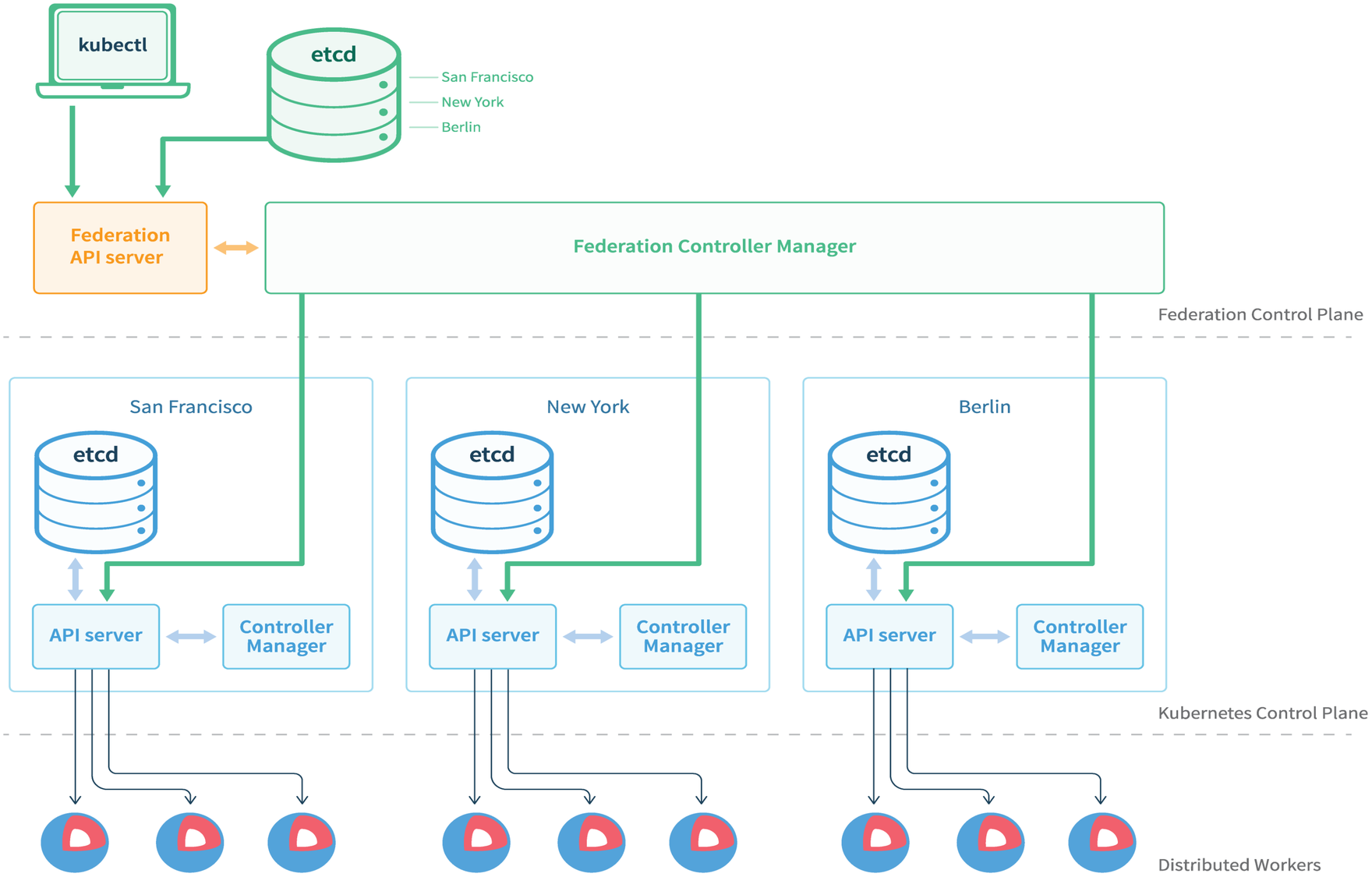
What use cases require Cluster Federation?

1. **Capacity Overflow**
   - location affinity
   - cross-cluster scheduling
   - cross-cluster service discovery
   - cross-cluster migration
   - cross-cluster load-balancing
   - cross-cluster Data synchronization
   - cross-cluster monitoring and auditing
2. **Sensitive Workloads**
   - What kinds of rules determine which workloads go where?
3. **Vendor lock-in avoidance**
   - *workloads to run across multiple cloud providers at all times*
4. **High Availability**
   - *spread my service across multiple such zones and have my service remain available even if one of the availability zones or cloud providers "goes down".*

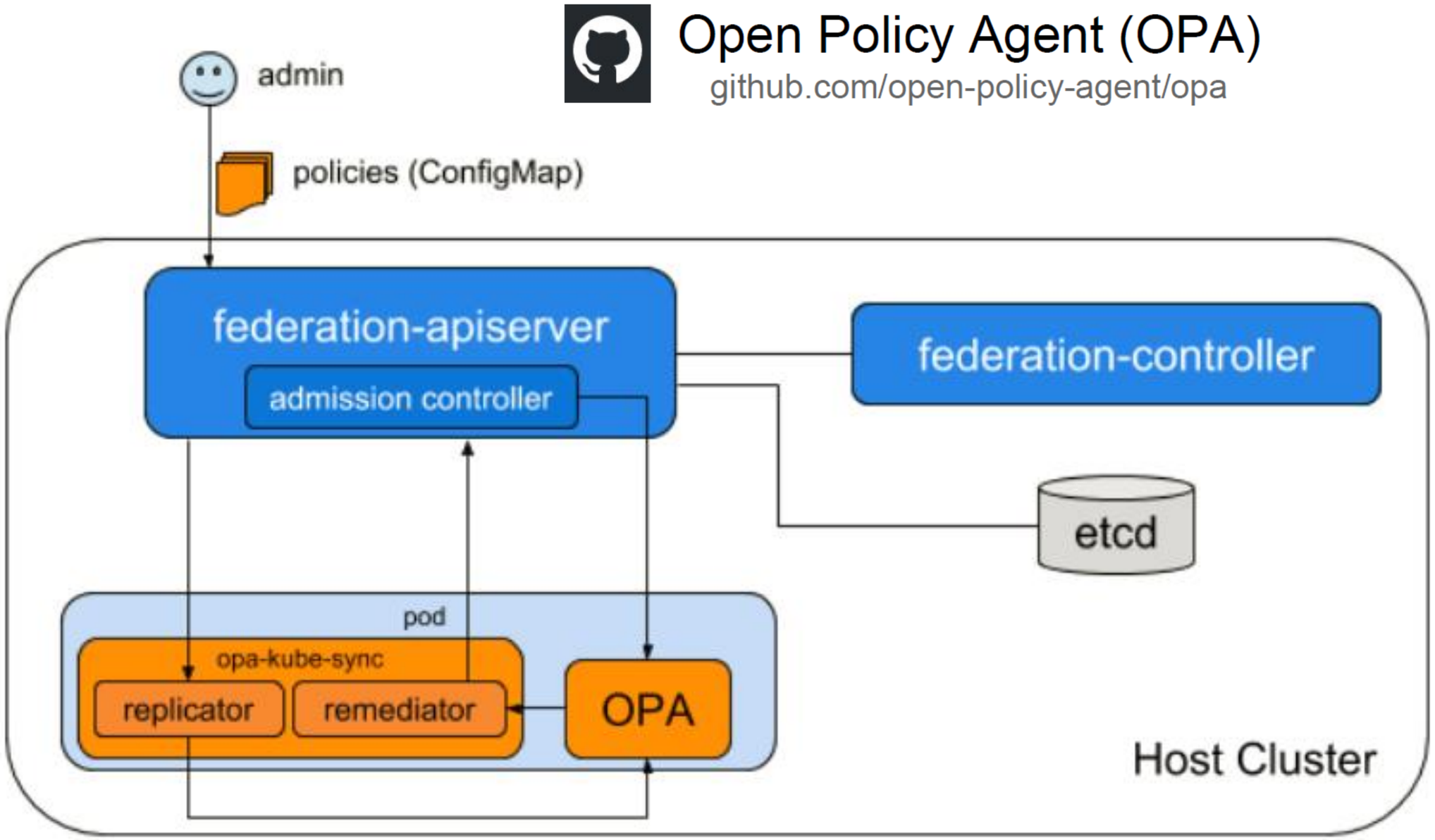
Federation v1



Federation v1



Federation v1



Federation v1 -- Resources

Top Level API Objects

- v1beta1.Deployment
- v1beta1.DeploymentList
- v1beta1.DeploymentRollback
- v1beta1.Scale
- v1beta1.DaemonSetList
- v1beta1.DaemonSet
- v1beta1.Ingress
- v1beta1.IngressList
- v1beta1.ReplicaSet
- v1beta1.ReplicaSetList

Top Level API Objects

- v1.Service
- v1.ServiceList
- v1.Event
- v1.EventList
- v1.Namespace
- v1.NamespaceList
- v1.Secret
- v1.SecretList
- v1.ConfigMap
- v1.ConfigMapList

Federation v2 -- Concept

The following terms are defined in the context of Federation:

- **Member Cluster:** Federated APIs manage resources in *member clusters*, which are Kubernetes clusters that the Federation API has been configured to access.

- **Template:** A *template* specifies a base Kubernetes resource that should be created in one or more member clusters.
- **Overrides:** *Overrides* enable fields of a template resource to vary by cluster.
- **Placement:** *Placement directives* identify which member clusters a resource or set of resources should exist in.

- **Propagation:** *Propagation* ensures that a resource definition (composed from the template, overrides and placement) exists in a given member cluster.
- **Status:** *Status* provides visibility into the underlying member cluster state that has resulted from propagation (e.g. how many pods exist for a replica set).

- **Scheduling:** *Scheduling* determines resource placement and overrides from both user intent (including per-resource scheduling configuration) and the live state (both spec and status) of member clusters.

Federation v2 -- Base Federated Resources

The pattern for constructing a federated API from an existing k8s API resource:

- New API group = federation. + existing API group
- ResourceNames = Federated + existing resource name
- FederatedX's spec looks like:
  - **template** : the 'template' for what the resource looks like, absent any cluster-specific differentiation
  - **overrides** : a list of partial overrides of the template for a specific cluster

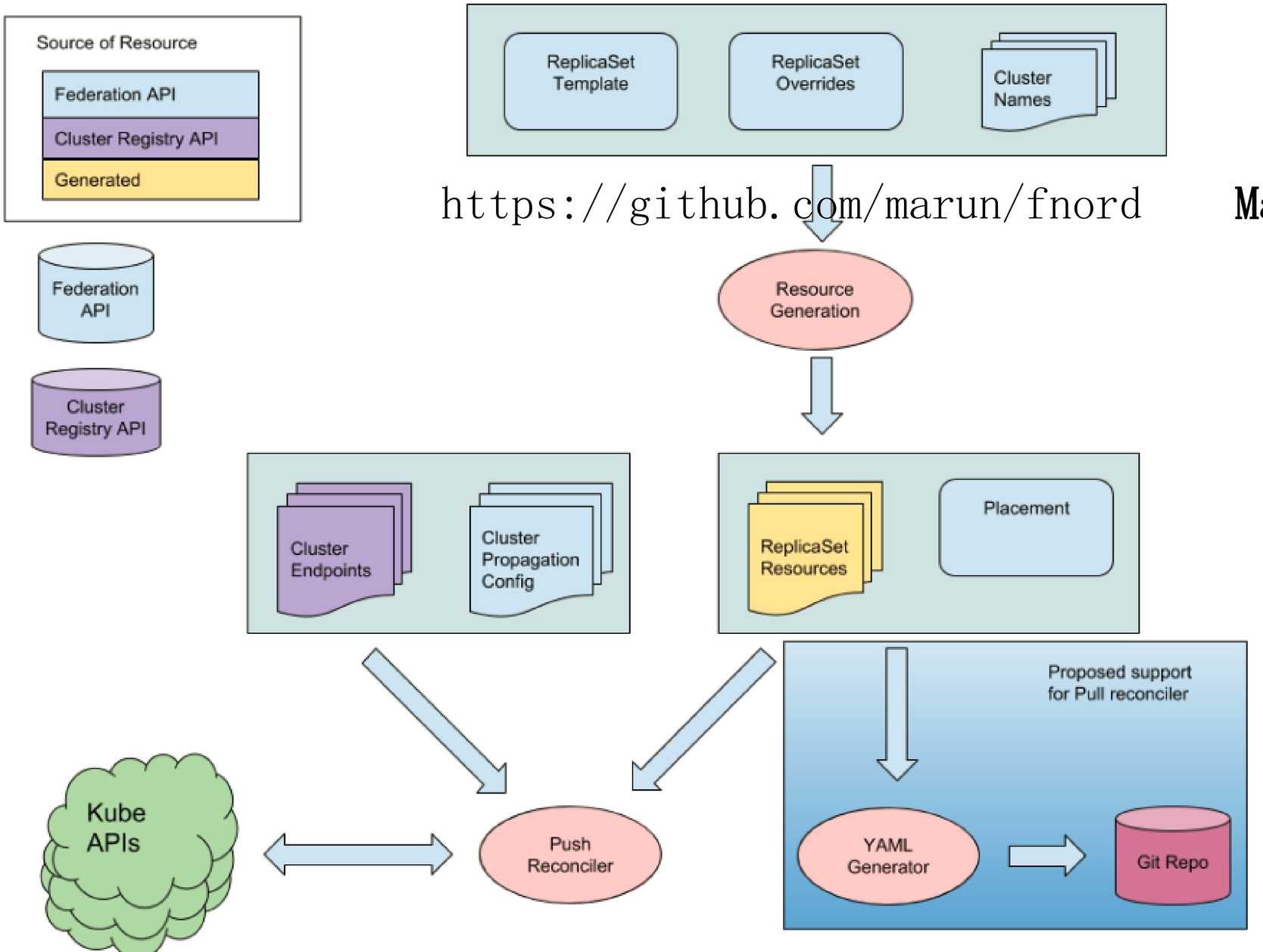
Federation v2 -- Auxiliary Resources

1. Cluster Placement Preferences - a set of hard or soft constraints influencing which clusters the resource should be federated to.
3. Template Substitution Preferences - a set of rules defining how and what values are substituted into templates on a per-cluster basis.
4. Template Substitution Outputs - These are actual kubernetes API resources, the results of applying cluster placement decisions and template substitution preferences to base federated resource templates.
5. Resource Propagation Configurations - these configure the mechanism by which template substitution decisions are propagated to specific clusters,
6. Propagation Status … TBD.

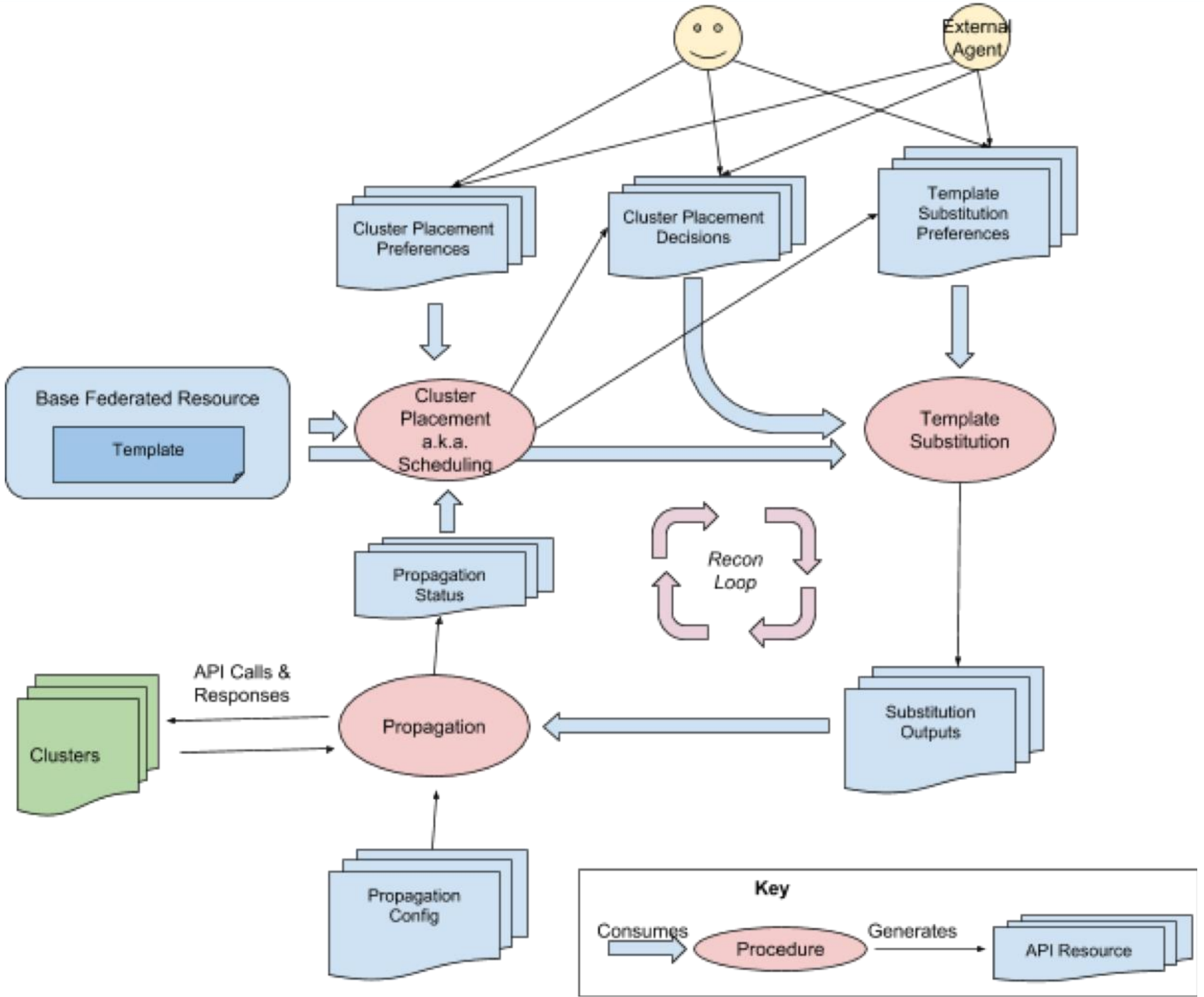
Federation v2 -- Work in progress

```yaml
apiVersion: federation.apps.k8s.io/v1alpha1
kind: FederatedReplicaSet
metadata:
  name: frontend
  namespace: my-ns
spec:
  template:
    apiVersion: apps/v1
    kind: ReplicaSet
    meta:
      name: frontend
      namespace: my-ns
      metadata:
        name: nginx
        labels:
          app: frontend
    spec:
      replicas: 3
      template:
        metadata:
          labels:
            app: guestbook
            tier: frontend
          spec:
          containers:
          - name: php-redis
            image: gcr.io/google_samples/gb-frontend:v3
          spec:
          containers:
          - name: php-redis
            image: gcr.io/google_samples/gb-frontend:v3
            resources:
              requests:
                cpu: 100m
                memory: 100Mi
            env:
            - name: GET_HOSTS_FROM
              value: dns
            ports:
            - containerPort: 80
  overrides:
  - clusterName: X
    override:
      replicas: 10
      spec:
        containers:
        - name: php-redis
          image: gcr.io/google_samples/gb-frontend:v4
```

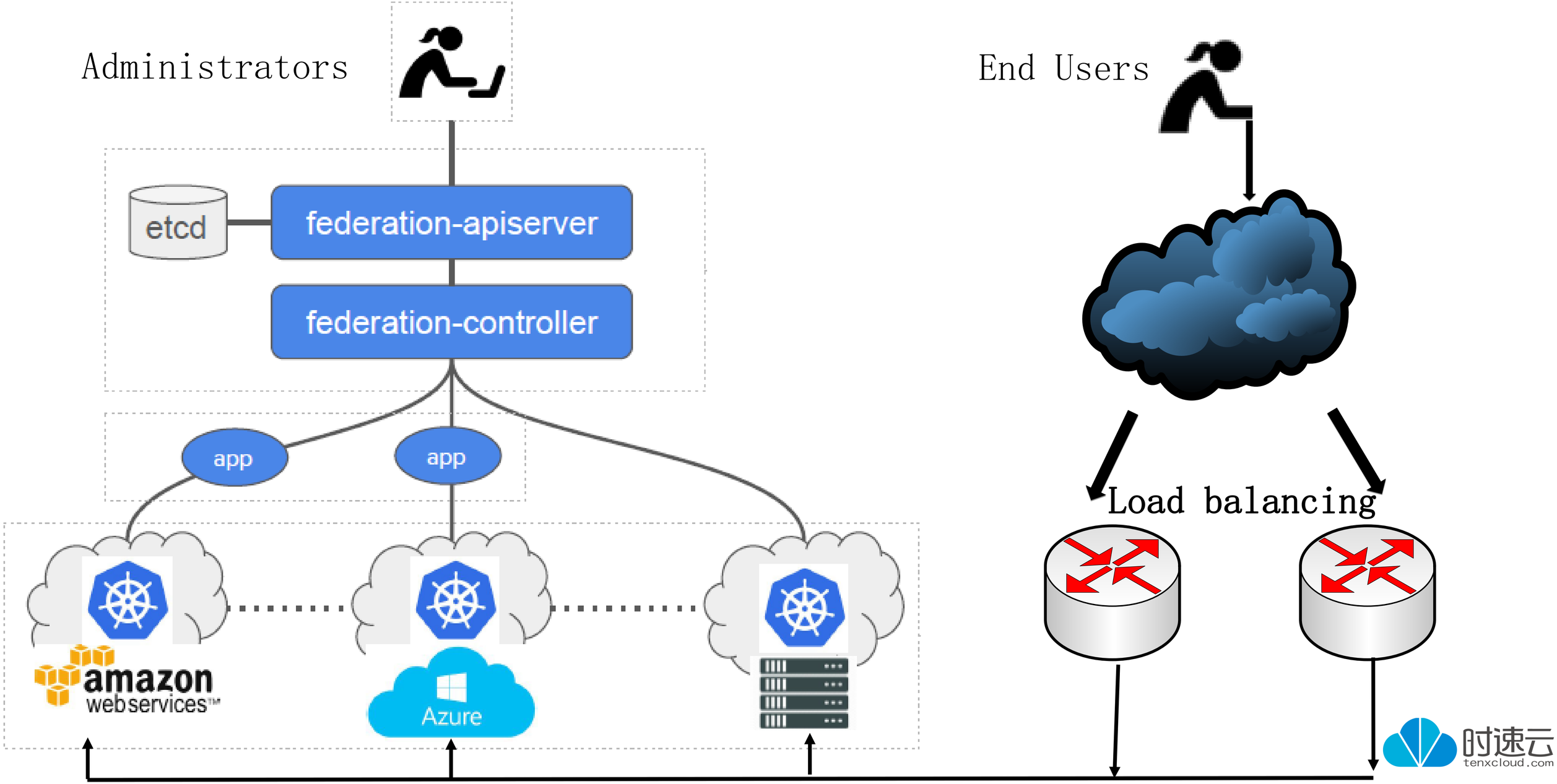
Federation v2 -- Prototype



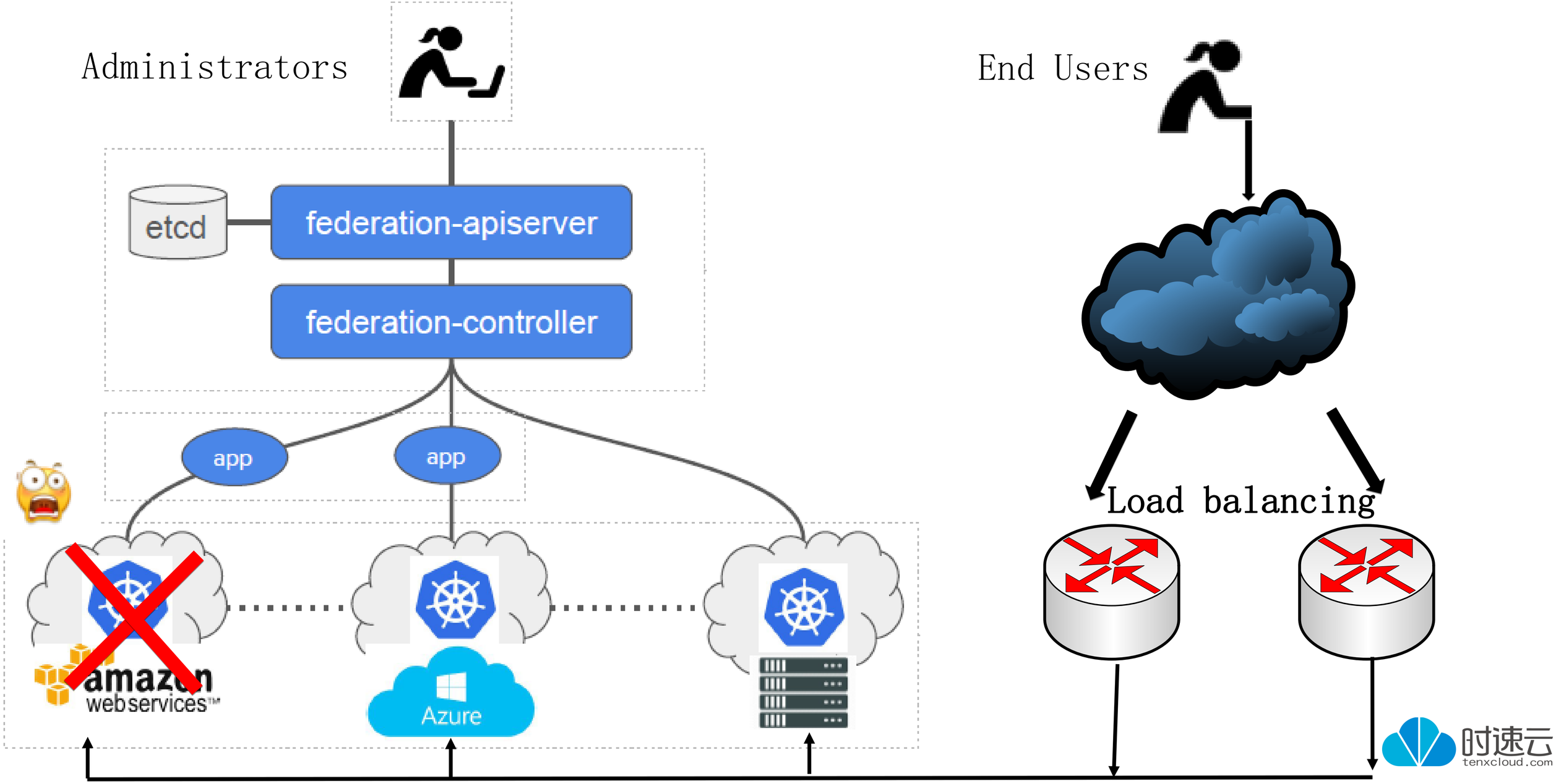
Federation v2 -- Work in progress



Global Load balancing



Global Load balancing



Global Load balancing -- Clarifying questions

How does global external service discovery work?

How does dynamic load balancing across clusters ?

How does this tie in with auto-scaling of services?

Global Load balancing

Options for global loadbalancing

Google Global Load Balancer

- Ideal solution with multiple clusters on Google Cloud
- No solution for Hybrid Cloud (yet)

Backplane.io

- Developer focused, very new...
- Working Kubernetes and Docker Swarm integration

DNS based (e.g. AWS Route53)

- Usual “DNS Problems”, e.g. misbehaving clients ignoring TTLs
- Failover times dependent on DNS TTLs and intermediate caching

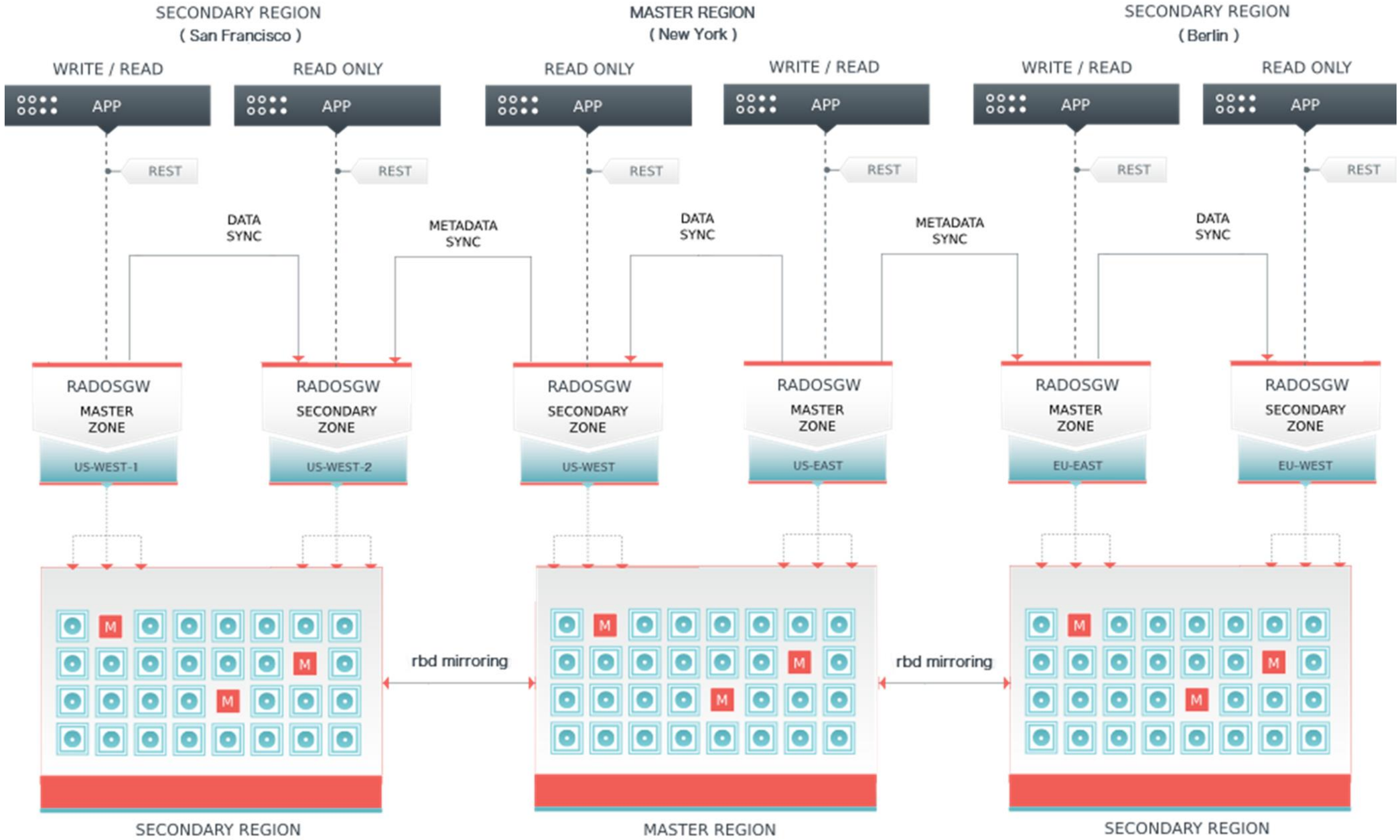
Content Delivery Networks

- Currently, manual configuration needed
- Best option for Hybrid installations now

Global Distributed Storage

1. Ceph
2. Glusterfs
3. Lustre
4. Moosefs
5. HDFS
6. ……

Global Distributed Storage



Global Distributed Storage

globally distributed databases
    like CockroachDB 、 Google Spanner etc.

non-globally distributed databases
    like zookeeper、etcd、memcache、redis etc.

Federation -- Caveats & Limitations

1. Still under heavy development
2. Not all functionality is currently available in Federation
3. Single federated control plane
4. Increased network bandwidth and cost
5. A bug in the federation control plane could impact all clusters

Federation – Github Repo

SIG-Federation

Federation v1(Kubernetes v1.9+)
https://github.com/kubernetes/federation

Federation v2
https://github.com/kubernetes-incubator/apiserver-builder
https://github.com/kubernetes/cluster-registry
https://github.com/kubernetes-sigs/federation-v2
https://github.com/GoogleCloudPlatform/k8s-multicluster-ingress
https://github.com/marun/fnord

Questions ?