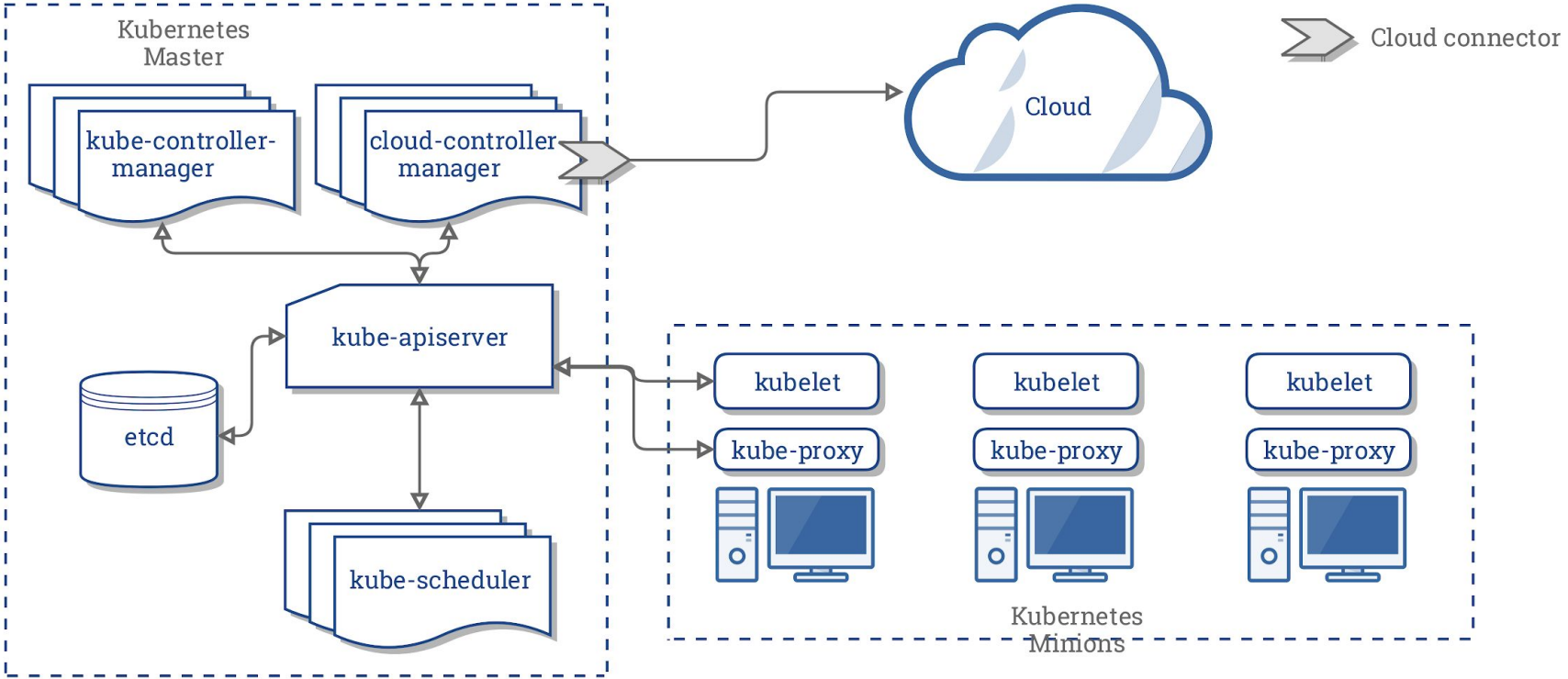
Deploying and Operating Kubernetes

Ricardo Rocha <ricardo.rocha@cern.ch>

CERN IT-CM-RPS


Kubernetes
Master
kube-controller-
manager
cloud-controller
manager
Cloud
Cloud connector
kube-apiserver
etcd
kube-scheduler
kubelet
kube-proxy
kubelet
kube-proxy
kubelet
kube-proxy
Kubernetes
Minions

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
  labels:
    app: myapp
spec:
  replicas: 10
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp
        image: nginx
        resources:
          requests:
            cpu: 1000m
            memory: 256Mi
          limits:
            cpu: 1000m
            memory: 256Mi
```

Uniform API and Resource Definition

Deployment, Service, Ingress, Volume, StorageClass, Role, ClusterRole, Metric, Job, CustomResourceDefinition

...

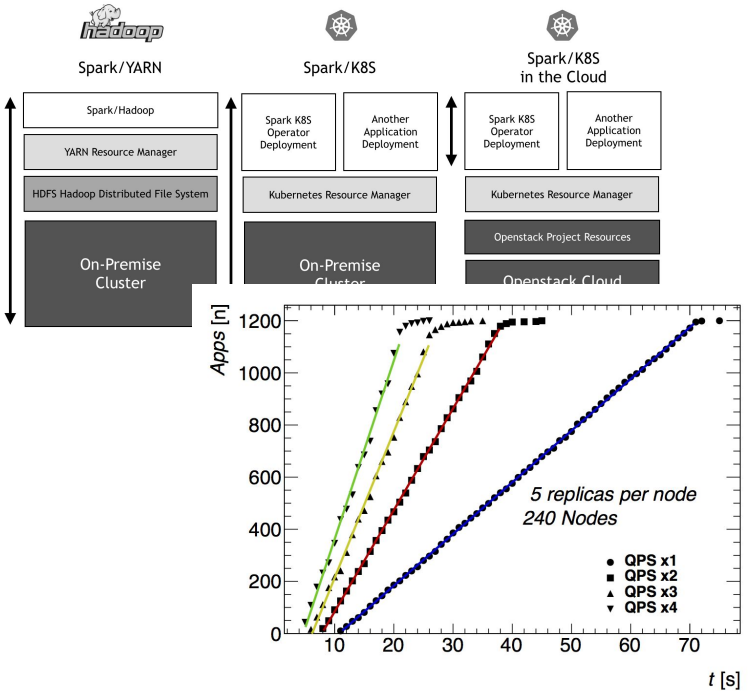
Support in all major cloud vendors

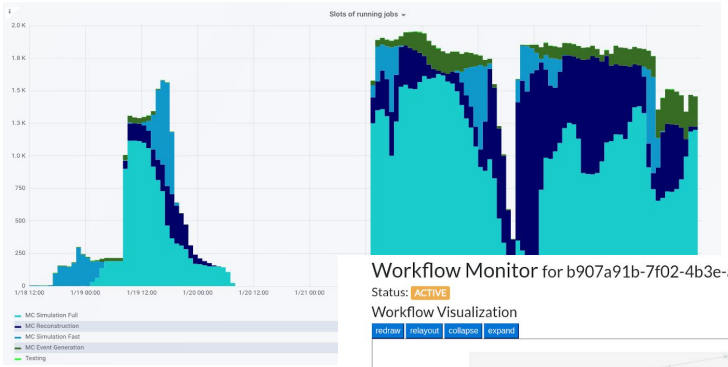
Simplified Infrastructure

Monitoring, Lifecycle, Alarms

Simplified Deployment

Uniform API, Replication, Load Balancing





Spark, Kubeflow, JupyterHub, Binder, REANA/RECAST…

Deployment

Easiest with a cloud provider

```
$ openstack coe cluster create --cluster-template kubernetes-1.13.3-1
        --node-count 100 mycluster-001

$ gcloud container clusters create --region europe-west1
         --num-nodes 100 mycluster-001

$ az aks create --node-count 100 mycluster-001

$ aws eks --region eu-west-1 create-cluster --name mycluster-001
```

Deployment

kubeadm: simple, single master cluster deployments

Supported by the Kubernetes community

Integrates well with Ansible, Terraform and other provisioning systems

Master

```
$ kubeadm init --pod-network-cidr=192.168.0.0/16
```

Minions

```
$ kubeadm join <master-ip:master-port> --token <token> ...
```

Deployment

kops: opinionated provisioning system for multi cluster deployment

Builds on kubeadm, support for self-healing, HA

Support for AWS, Integration with Terraform

```
$ kops create cluster --zones=us-east-1c useast1.mycluster.com

$ kops update cluster useast1.mycluster.com --yes
```

Cluster Add-ons

Base cluster deployment includes networking, DNS

Add-ons: Cluster and Service Monitoring, Log Collection, Storage Integration, …

Helm: The package manager for Kubernetes

```
$ helm install --namespace kube-system --name addon-name ./chartname ...
```

https://github.com/helm/charts

But other options exist, these are all run as kubernetes resources

Add-on: Prometheus

Most popular system and service monitoring for Kubernetes

Easy deployment using the Prometheus operator and helm chart

```
$ helm install stable/prometheus-operator
```

Built-in basic metric collection (cpu, memory, …) for Pods and Nodes

Custom metric definition and polling

“ Auto scale my application based on job queue size “

K8s / Compute Resources / Cluster

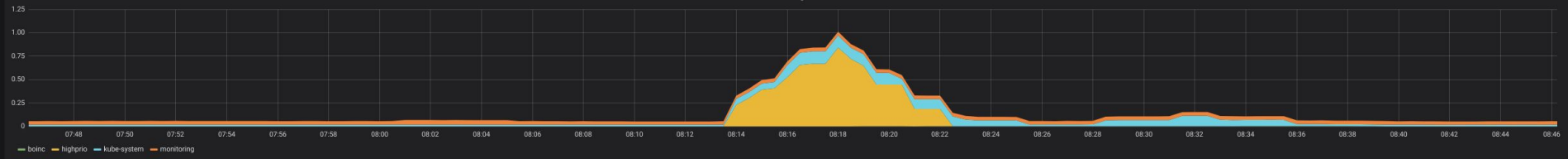
Last 1 hour Refresh every 10s

datasource Prometheus

Headlines

CPU Utilisation	CPU Requests Commitment	CPU Limits Commitment	Memory Utilisation	Memory Requests Commitment	Memory Limits Commitment
N/A	0.667%	0.333%	29.4%	2.59%	0.648%

CPU

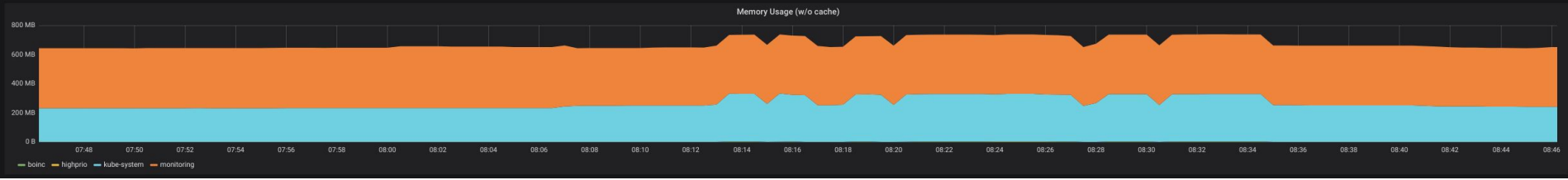


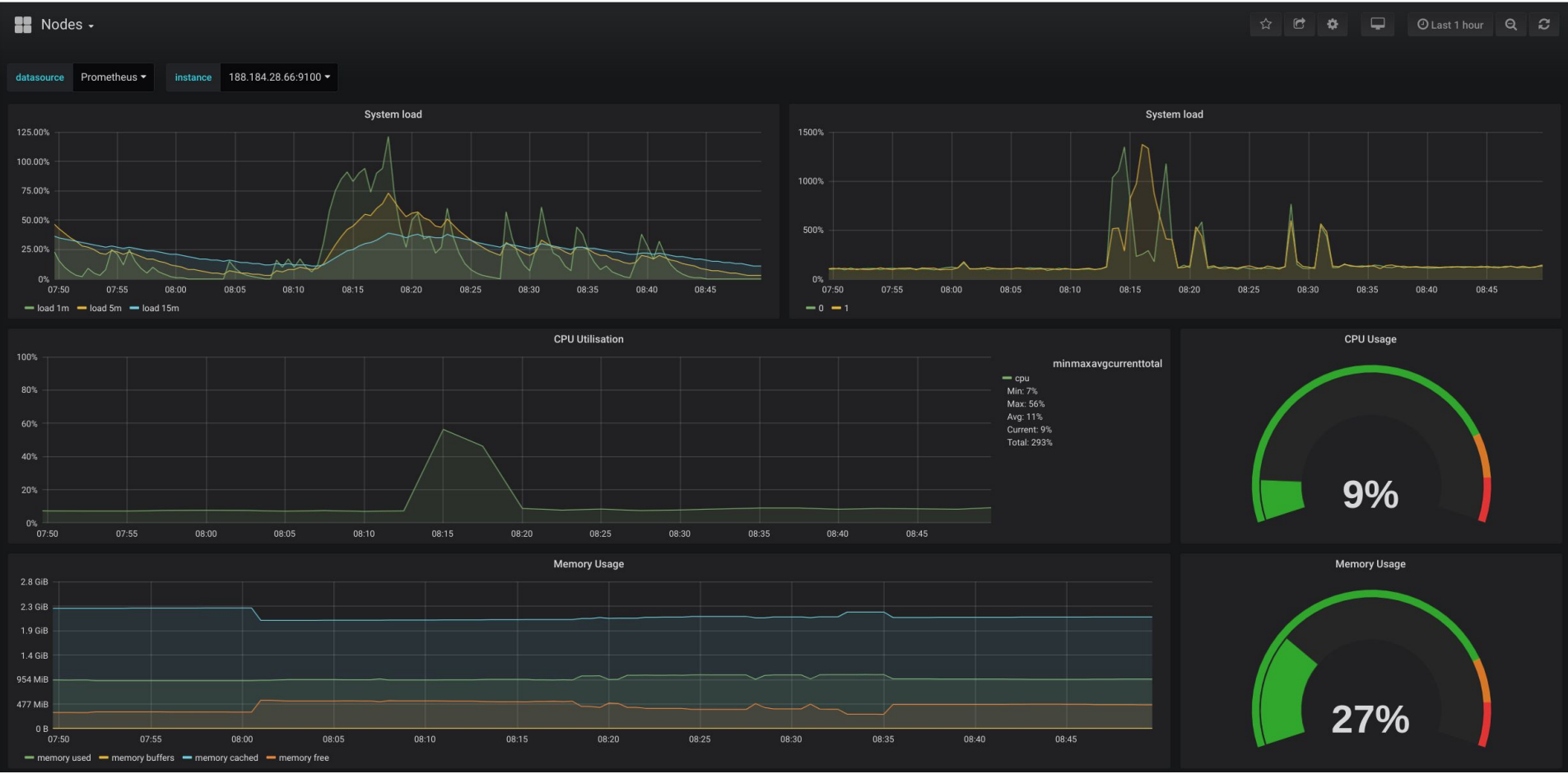
CPU Quota

CPU Quota

Namespace	CPU Usage	CPU Requests	CPU Requests %	CPU Limits	CPU Limits %
monitoring	0.04	0.02	196.70%	0.02	196.70%
kube-system	0.02	0.02	80.30%	-	-

Memory




Nodes
Last 1 hour
datasource
Prometheus
instance
188.184.28.66:9100
System load
125.00%
100.00%
75.00%
50.00%
25.00%
0%
07:50
07:55
08:00
08:05
08:10
08:15
08:20
08:25
08:30
08:35
08:40
08:45
load 1m
load 5m
load 15m
System load
1500%
1000%
500%
0%
0
1
CPU Utilisation
100%
80%
60%
40%
20%
0%
minmaxavgcurrenttotal
cpu
Min: 7%
Max: 56%
Avg: 11%
Current: 9%
Total: 293%
CPU Usage
9%
Memory Usage
2.8 GiB
2.3 GiB
1.9 GiB
1.4 GiB
954 MiB
477 MiB
0 B
memory used
memory buffers
memory cached
memory free
Memory Usage
27%

Add-on: FluentD

Open Source (log) data collector

Unified logging layer with multiple sources and destinations

Source plugins for all popular applications

nginx, apache, mysql, syslog, …

Destination plugins for a variety of popular log aggregation systems

ElasticSearch, Splunk, HDFS, …

https://www.fluentd.org/plugins/all

Add-on: Cluster Auto Scaling

In addition to the Pod auto scaler (application), auto scale the cluster nodes

Support for all popular cloud providers

AWS, Azure, GCE/GKE, AliCloud, OpenStack

Scale based on **cpu**, **memory** or any **other metric**

Expand cluster up to **max nodes** when needed

Shrink the cluster up to a **min nodes** when no longer needed

https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler

Add-on: Node Problem Detector

Check and report node issues to the Kubernetes control plane

ntp down, bad cpu, bad memory, storage issues, …

KernelMonitor, AbrtAdaptor, **CustomPluginMonitor**

Drain / replace nodes automatically when issues are triggered (NotReady state)

```
$ helm install stable/node-problem-detector
```

https://github.com/kubernetes/node-problem-detector

RBAC

Role Based Authorization Control

Fine grained access control to Kubernetes resources

```yaml
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

```yaml
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

Pod Preemption and Priority

Available since Kubernetes 1.8

Enabled by default in recent versions (>=1.11)

https://kubernetes.io/docs/concepts/configuration/pod-priority-preemption/

Possible to implement **fair sharing** in a single cluster

Good solution to deploy backfill workloads

```
apiVersion: scheduling.k8s.io/v1beta1
kind: PriorityClass
metadata:
  name: default
value: 10000
globalDefault: true
---
apiVersion: scheduling.k8s.io/v1beta1
kind: PriorityClass
metadata:
  name: backfill
value: -1000
```

Storage

Container Storage Interface (CSI) is 1.0

CSI Plugin for CVMFS available

https://github.com/cernops/cvmfs-csi

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-cvmfs-cms
provisioner: csi-cvmfsplugin
parameters:
  repository: cms.cern.ch
```

Physics data access using remote I/O protocols works out of the box

Federation

Google

Site X

kubefed init fed --host-cluster-context=master-cluster ...

*kubefed join --context fed google *
--host-cluster-context master-cluster --cluster-context google

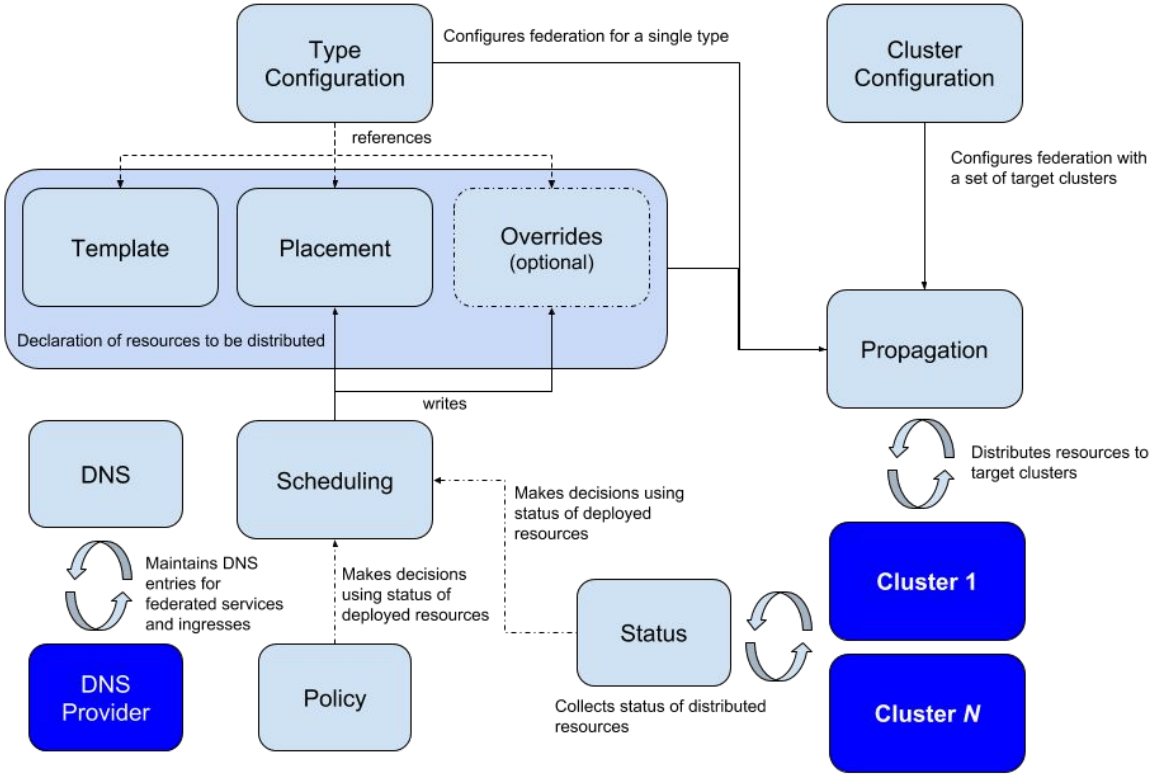
Federation Master

Federation v1 allowed simple aggregation of distinct Kubernetes clusters

Federation v2 is more complex, but adds interesting bits

Cluster Weights, Placement Policies, Replica Scheduling

Federation


Type Configuration
Configures federation for a single type
Cluster Configuration
references
Configures federation with a set of target clusters
Template
Placement
Overrides (optional)
Declaration of resources to be distributed
Propagation
writes
DNS
Scheduling
Distributes resources to target clusters
Makes decisions using status of deployed resources
Maintains DNS entries for federated services and ingresses
Makes decisions using status of deployed resources
Cluster 1
Status
DNS Provider
Policy
Collects status of distributed resources
Cluster N

Federation

```yaml
apiVersion: scheduling.federation.k8s.io/v1alpha1
kind: ReplicaSchedulingPreference
metadata:
  name: jupyterhub
  namespace: hub
spec:
  targetKind: FederatedDeployment
  totalReplicas: 9
  clusters:
    CERN:
      minReplicas: 2
      maxReplicas: 6
      weight: 100
    Azure:
      minReplicas: 0
      maxReplicas: 3
      weight: 20
```

“ Fill up CERN before using Cloud X “

“ Prefer Cloud X for GPU workloads “

“ Send TPU workloads to Google Cloud “

Questions?