Open Cluster Management

Workload distribution with Placement API

KubeCon 2022

Jian Qiu (@qiujian16)
Le Yang (@elgnay)
Qing Hao (@haoqing0110)

A Summary of Open Cluster Management

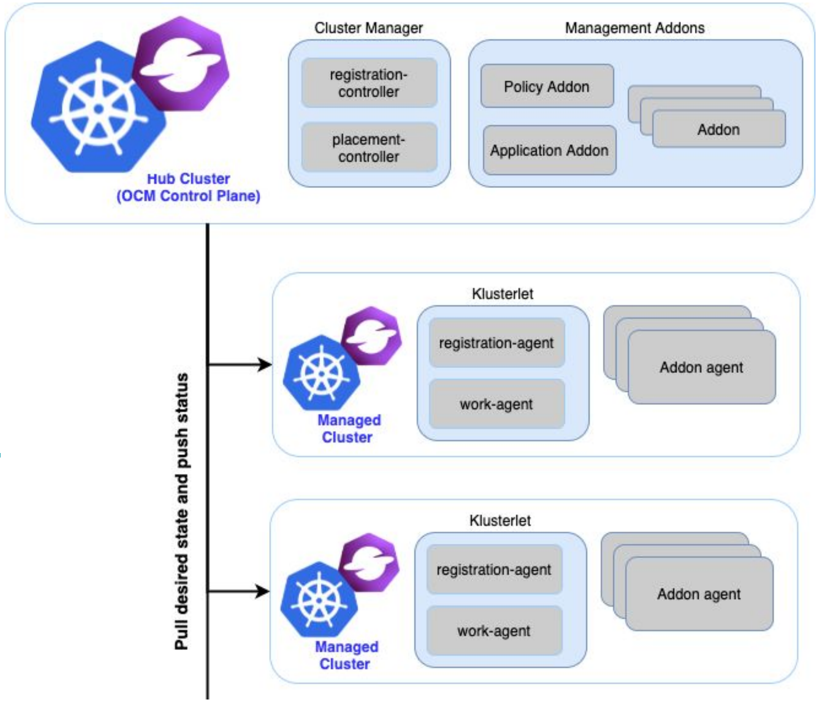
Simplify fleet management across the open hybrid cloud at scale.

- An open-source CNCF Sandbox project
- Simplifies the management of Kubernetes clusters
- Hub and spoke architecture
- Allows targeted distribution of Kubernetes manifests from the Hub
- Integration point for making Kubernetes capabilities multicluster aware

https://open-cluster-management.io/

CLOUD NATIVE
COMPUTING FOUNDATION

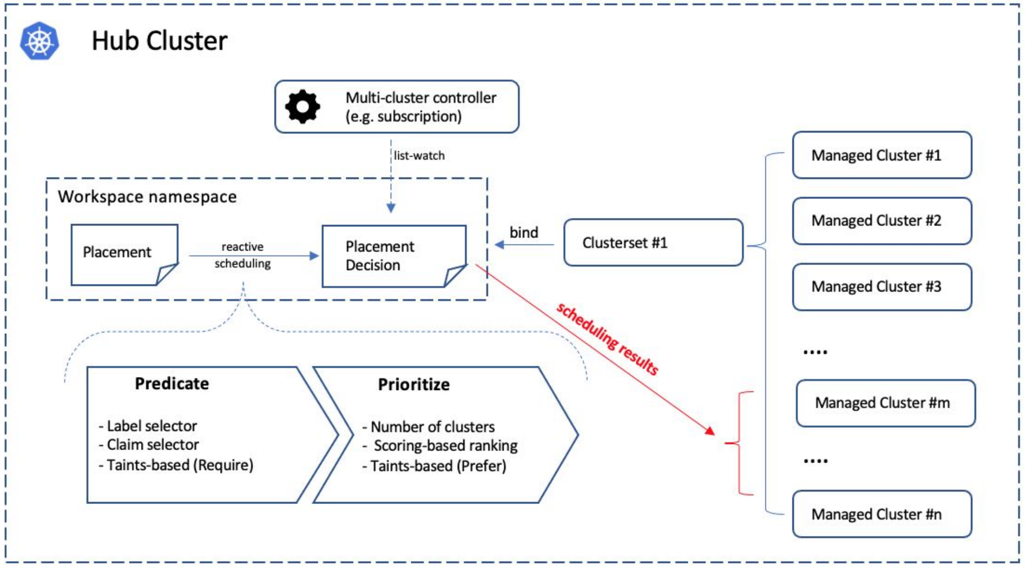
CLOUD NATIVE
SANDBOX



What is Placement?

Placement concept is used to dynamically select a set of managed clusters in one or multiple ManagedClusterSet so that higher level users can either replicate Kubernetes resources to the member clusters or run their advanced workload i.e. multi-cluster scheduling.

The “input” and “output” of the scheduling process are decoupled into two separated Kubernetes API **Placement** and **PlacementDecision**.



Placement API

- The Placement API is namespaced resource.
    - ManagedClusters(M): ManagedClusterSets(N)
    - ManagedClusterSets(M): Namespaces(N).
    - Placement can only select ManagedClusters that are bound to its namespace.
- The Placement API parameters and scheduling logic is divided into two phases.
    - Predicate: Hard requirements for the selected clusters.
        - ManagedClusterSets
        - Label/Claim selector
        - Taints/Tolerations
    - Prioritize: Rank the clusters by the soft requirements and select a subset among them.
        - Builtin & AddOn prioritizers
        - Support extensible scheduling

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
 name: placement1
 namespace: default
spec:
 numberOfClusters: 4
 clusterSets:
  - clusterset1
  - clusterset2
 predicates:
  - requiredClusterSelector:
     labelSelector:
      matchLabels:
       vendor: OpenShift
 tolerations:
  - key: "cluster.open-cluster-management.io/unreachable"
    operator: Exists
 prioritizerPolicy:
  mode: Exact
  configurations:
   - scoreCoordinate:
      builtIn: ResourceAllocatableMemory
   - scoreCoordinate:
       builtIn: Steady
      weight: 3
   - scoreCoordinate:
      type: AddOn
      addOn:
       resourceName: default
       scoreName: cpuratio
```

PlacementDecision API

- The PlacementDecision will be created by placement controller in the same namespace, each with a label of `cluster.open-cluster-management.io/placement={placement name}`.
- The PlacementDecision API contains the scheduling result.
  - The status.decisions list the top N clusters with highest score and ordered by names.
  - The status.decisions changes over time, the scheduling result update based on what endpoints exist.
- The PlacementDecision is paginated.
  - It is designed to be paginated with its page index as the name’s suffix, eg, placement1-decision-1, placement1-decision-2, placement1-decision-N.
  - Avoid “too large object” issue from the underlying Kubernetes API framework.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: PlacementDecision
metadata:
  labels:
    cluster.open-cluster-management.io/placement: placement1
  name: placement1-decision-1
  namespace: default
status:
  decisions:
    - clusterName: cluster1
    - clusterName: cluster2
    - clusterName: cluster3
```

PlacementDecision API

- The PlacementDecision can be parsed with a script and then operate on the target clusters. Or integrated with a high-level workload orchestrator to leverage its scheduling capabilities.
    - For example, Argo has an integration with Placement.

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: book-import
spec:
  generators:
   - clusterDecisionResource:
      configMapRef: ocm-placement
      labelSelector:
      matchLabels:
        cluster.open-cluster-management.io/placement: cluster1
      requeueAfterSeconds: 30
  Template:
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: ocm-placement
data:
  apiVersion: cluster.open-cluster-management.io/v1alpha1
  kind: placementdecisions
  statusListKey: decisions
  matchKey: clusterName
```

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: PlacementDecision
metadata:
  labels:
    cluster.open-cluster-management.io/placement: placement1
  name: placement1-decision-1
  namespace: default
status:
  decisions:
    - clusterName: cluster1
    - clusterName: cluster2
    - clusterName: cluster3
```

Demo: distribute workload with placement selected managed clusters

Predicates - ManagedClusterSets

- The **spec.clusterSets** section represents the ManagedClusterSets from which the ManagedClusters are selected.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: default
spec:
  numberOfClusters: 3
  clusterSets:
    - prod
```

Predicates - Label/Claim selector

- In the **spec.predicates** section, you can select clusters by labels or ClusterClaims.
- For instance, you can select 3 clusters with labels purpose=test and ClusterClaim platform.open-cluster-management.io=aws as seen in the example.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: default
spec:
  numberOfClusters: 3
  clusterSets:
    - prod
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchLabels:
            purpose: test
        claimSelector:
          matchExpressions:
            - key: platform.open-cluster-management.io
              operator: In
              values:
                - aws
```

Predicates - Taints/Tolerations

- **Taints** are properties of managed clusters, they allow a placement to repel a set of managed clusters.
  - The field **key, value** and **effect** working similar kubernetes node.spec.taints.
  - The **timeAdded** is the time at which the taint was added.
- **Tolerations** are applied to placements, and allow the managed clusters with matching taints to be scheduled onto placements.
  - The field **key, value, operator** and **effect** working similar to kubernetes pod.spec.tolerations.
  - The **tolerationSeconds** represents the period of time the toleration tolerates the taint. In this example, the toleration expires at "2022-03-07T02:**31**:19Z"

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
...
spec:
 taints:
 - effect: NoSelect
   key: gpu
   value: true
   timeAdded: "2022-03-07T02:01:19Z"
```

```
# Tolerate clusters with taint
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
...
spec:
 tolerations:
  - key: gpu
    operator: Equal
    value: true
    effect: NoSelect
    tolerationSeconds: 30
```

Demo: cluster maintenance with placement taints/tolerations

Prioritizers

Prioritizers is used to rank the clusters filtered from the hard requirements, and choose the clusters with higher score. Available builtin prioritizers are:

- **Balance:** Balance the number of decisions among the clusters. The cluster with more decision is given a lower score.
- **Steady:** Keeps the decision result steady. The clusters that existing decisions choose are given the higher score.
- **ResourceAllocatableCPU & ResourceAllocatableMemory:** Prefer to Select clusters with more allocatable resource.

Besides the builtin prioritizers, placement support ranking clusters by customized score, this is also called extensible scheduling. The customized score can be defined in **addOn** section.

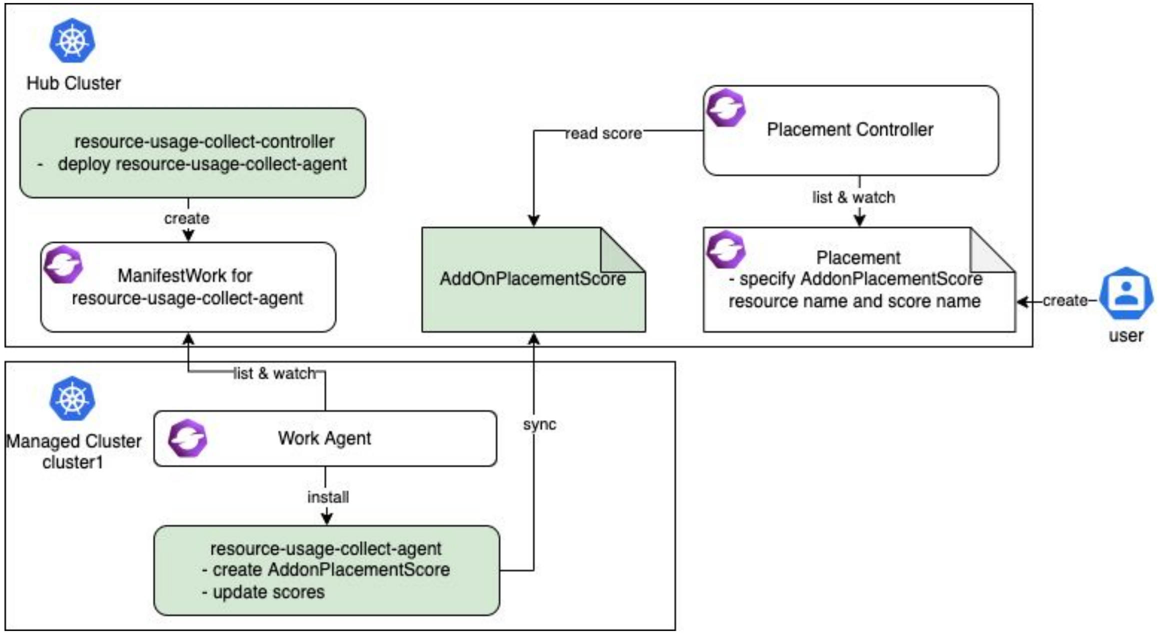
The prioritizers score range is [-100, 100], priorizer weight range is [-10, 10]. Cluster score is the sum of score*weight.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
 name: placement1
 namespace: default
spec:
 numberOfClusters: 2
 prioritizerPolicy:
  mode: Exact
  configurations:
   - scoreCoordinate:
      builtIn: ResourceAllocatableMemory
   - scoreCoordinate:
      builtIn: Steady
     weight: 3
   - scoreCoordinate:
      type: AddOn
      addOn:
       resourceName: default
       scoreName: cpuratio
```

Cluster score = Steady_Score * 3 + ResourceAllocatableMemory_Score + AddOn_Score

Prioritizers - Extensible scheduling

- Extend the multicluster scheduling capabilities with placement.
  https://open-cluster-management.io/scenarios/extend-multicluster-scheduling-capabilities/



Hub Cluster
resource-usage-collect-controller
- deploy resource-usage-collect-agent
create
ManifestWork for resource-usage-collect-agent
read score
Placement Controller
list & watch
AddOnPlacementScore
Placement
- specify AddonPlacementScore resource name and score name
create
user
list & watch
Managed Cluster cluster1
Work Agent
sync
install
resource-usage-collect-agent
- create AddonPlacementScore
- update scores

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: AddOnPlacementScore
metadata:
 name: default
 namespace: {managed cluster namespace}
status:
 conditions:
...
 validUntil: "2021-10-29T18:31:39Z"
 scores:
 - name: "cpuratio"
  value: 88
 - name:: "memratio"
  value: 77
```

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: Placement
metadata:
 name: placement
 namespace: demo
spec:
 numberOfClusters: 1
 prioritizerPolicy:
  mode: Exact
  configurations:
   - scoreCoordinate:
     type: AddOn
     addOn:
      resourceName: default
      scoreName: cpuratio
    weight: 1
```

Demo: extend the multicluster scheduling capabilities with placement

Future

- Spread Policy across Failure-domains in Placement APIs.
  https://github.com/open-cluster-management-io/enhancements/pull/70
- More user scenarios:
  - How to use placement and other open source tools to perform workload or storage disaster recovery.
    https://github.com/open-cluster-management-io/OCM/issues/60
  - How to do cluster maintenance and its implementation on placement.
    https://github.com/open-cluster-management-io/OCM/issues/61

Get Involved

- GitHub: https://github.com/open-cluster-management-io/OCM
- Website: https://open-cluster-management.io/
- Docs: https://open-cluster-management.io/concepts/
- Slack: https://kubernetes.slack.com/channels/open-cluster-mgmt
- YouTube: https://www.youtube.com/c/OpenClusterManagement
- Mailing Group: https://groups.google.com/g/open-cluster-management
- Community Meetings: https://calendar.google.com/calendar/u/0/embed?src=openclustermanagement@gmail.com

Open Cluster Management
https://open-cluster-management.io/

Open Cluster Management

https://open-cluster-management.io/

Thank you for joining.

Questions?

Troubleshooting

```
$ kubectl describe placement <placement-name>
...
Status:
  Conditions:
    Last Transition Time:    2022-09-30T07:39:45Z
    Message:                 Placement configurations check pass
    Reason:                  Succeedconfigured
    Status:                  False
    Type:                    PlacementMisconfigured
    Last Transition Time:    2022-09-30T07:39:45Z
    Message:                 No valid ManagedClusterSetBindings found in placement namespace
    Reason:                  NoManagedClusterSetBindings
    Status:                  False
    Type:                    PlacementSatisfied
  Number Of Selected Clusters:  0
```

```
$ kubectl describe placement <placement-name>
...
Events:
  Type    Reason          Age    From                 Message
  ----    ------          ----   ----                 -------
  Normal  DecisionCreate  2m10s  placementController  Decision demo-decision-1 is created with placement demo in namespace ns1
  Normal  DecisionUpdate  2m10s  placementController  Decision demo-decision-1 is updated with placement demo in namespace ns1
  Normal  ScoreUpdate     2m10s  placementController  cluster1:0 cluster2:100 cluster3:200
  Normal  DecisionUpdate  3s     placementController  Decision demo-decision-1 is updated with placement demo in namespace ns1
  Normal  ScoreUpdate     3s     placementController  cluster1:200 cluster2:145 cluster3:189 cluster4:200
```

```
# oc describe placementdecision <placement-name>-decision-1
...
Status:
  Decisions:
    Cluster Name:  cluster3
    Reason:
    Cluster Name:  cluster4
    Reason:
```