

释放 CEL 在高级多集群调度中的潜力

Unlocking the Power of CEL for Advanced Multi-Cluster Scheduling

Qing Hao (郝青)

Senior Software Engineer @ Red Hat | Maintainer @ Open Cluster Management

CNCF Ambassador

Content 目录

01 多集群调度遇到的挑战

02 通用表达式语言(CEL)

03 CEL在多集群调度的应用

AI

Part 01

多集群调度遇到的挑战

The challenges we meet in advanced multi-cluster scheduling

AI



Open Cluster Management

Simplify fleet management across the open hybrid cloud at scale.

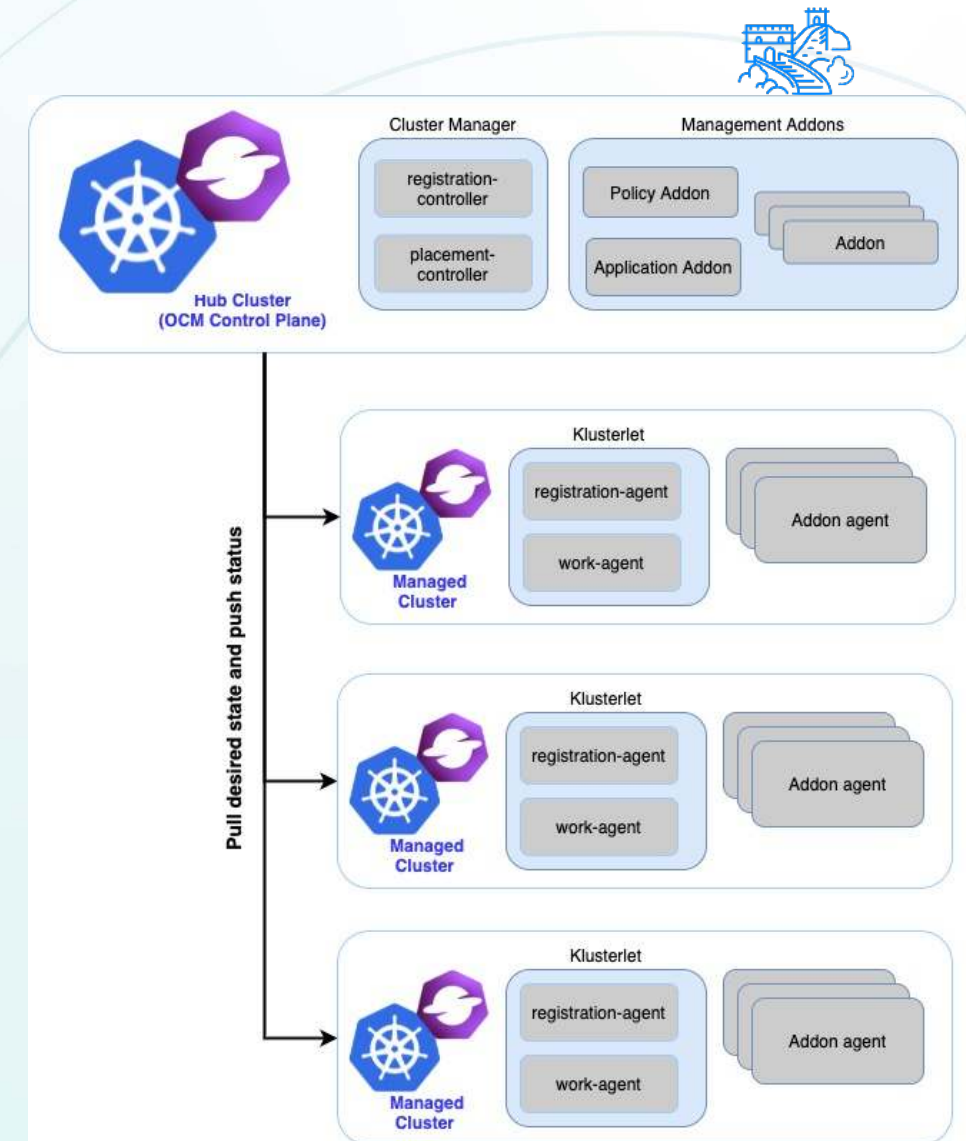
- **Hub and spoke architecture**
- **Cluster Inventory**
Registration of multiple clusters to a hub cluster to place them for management.
- **Work Distribution**
The work API enables resources to be applied to managed clusters from hub.
- **Content Placement**
Dynamic placement of content and behavior across multiple clusters.
- **Add-On Extensibility**
Integration point for making Kubernetes capabilities multicluster aware.
- **Vendor Neutral APIs**
Avoid vendor lock-in by using APIs that are not tied to any cloud providers or proprietary platforms.



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



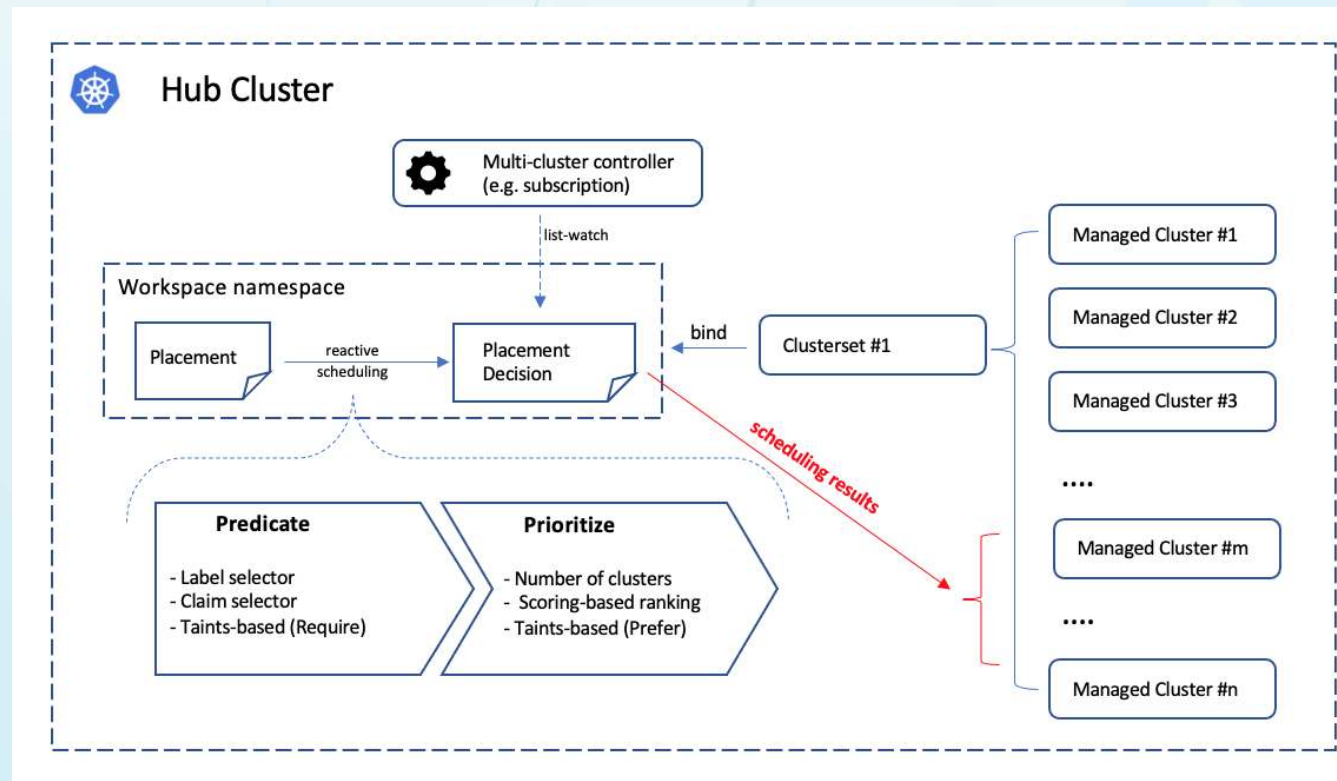
CLOUD NATIVE
COMPUTING FOUNDATION



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 - Predicates

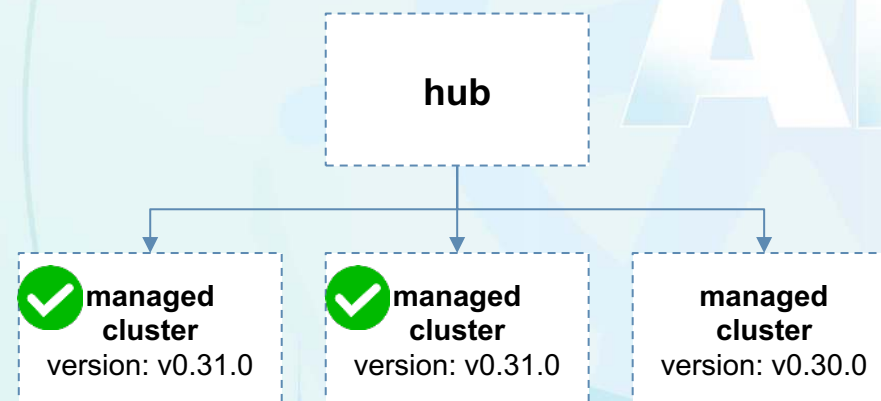
- **ManagedClusterSets:** The spec.clusterSets section represents the ManagedClusterSets from which the ManagedClusters are selected.
- **Label/Claim selector:** In the spec.predicates section, you can select clusters by labels or ClusterClaims.
- **Taints/Tolerations:**
 - Taints are properties of managed clusters, they allow a placement to repel a set of managed clusters.
 - Tolerations are applied to placements, and allow the managed clusters with matching taints to be scheduled onto placements.

```
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
```

Placement - Predicates

I **can** select with labels version=v1.31.0.

```
apiVersion: cluster.open-cluster-  
management.io/v1beta1  
kind: Placement  
metadata:  
  name: placement1  
  namespace: default  
spec:  
  predicates:  
    - requiredClusterSelector:  
      labelSelector:  
        matchLabels:  
          version: v1.31.0
```



Placement - Predicates challenges

I **want to** select clusters with version < v1.31.0.

```
apiVersion: cluster.open-cluster-  
management.io/v1beta1
```

```
kind: Placement
```

```
metadata:
```

```
  name: placement1
```

```
  namespace: default
```

```
spec:
```

```
  predicates:
```

```
    - requiredClusterSelector:
```

```
      labelSelector:
```

```
        matchExpressions:
```

```
          - key: version
```

```
            operator: In
```

```
            values:
```

```
              - v1.31.0
```

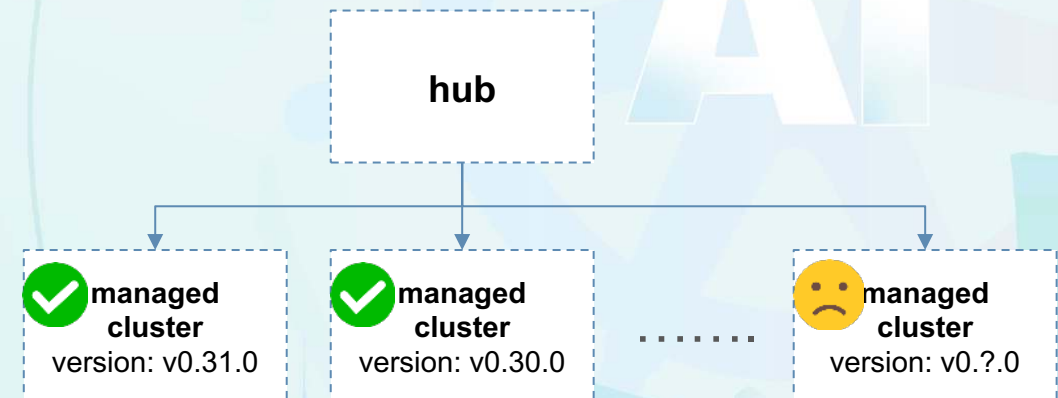
```
              - v1.30.0
```

```
              - v1.29.0
```

```
              - .....
```



- List versions one by one is not smart.
- The version info is even stored in status rather than labels or claims.



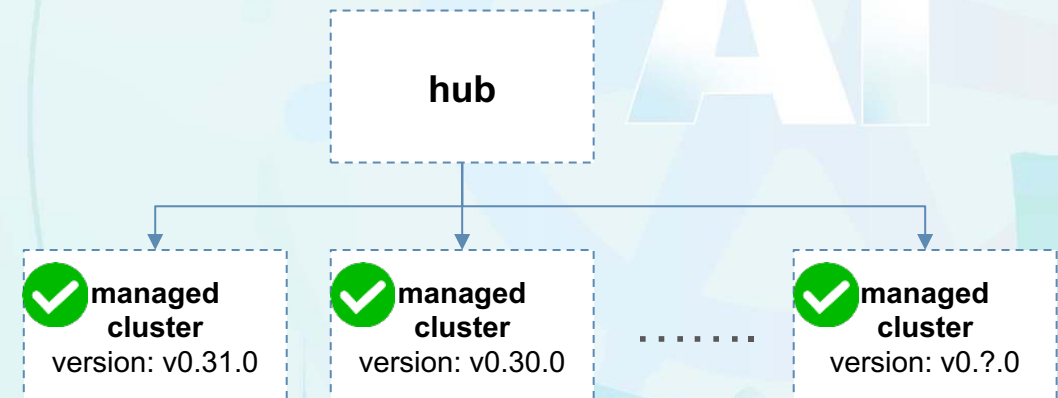
Placement - Predicates challenges

I **want to** select clusters with version < v1.31.0.

```
apiVersion: cluster.open-cluster-  
management.io/v1beta1  
kind: Placement  
metadata:  
  name: placement1  
  namespace: default  
spec:  
  predicates:  
    - requiredClusterSelector:  
      labelSelector:  
        matchExpressions:  
          - key: version  
            operator: lessThan  
            values:  
              - v1.31.0
```



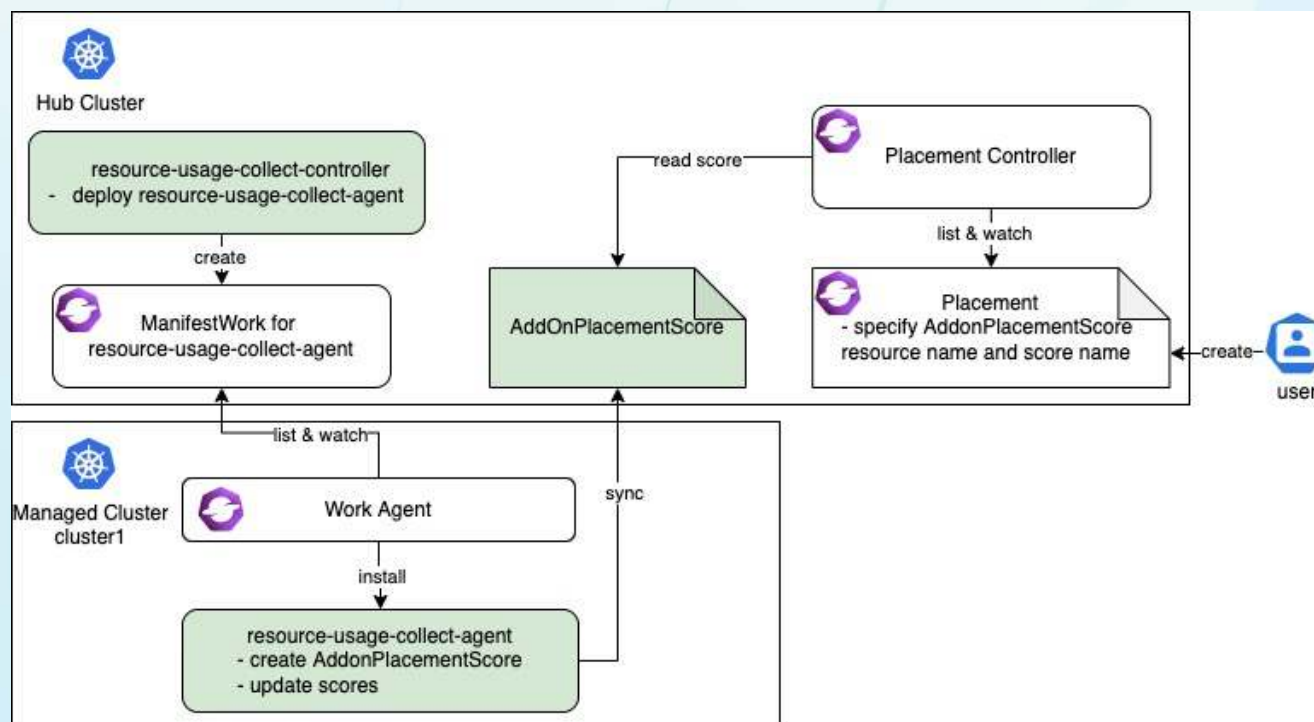
- Add a new operand is good, could there be a more flexible approach?
- What if the properties for scheduling is not labels or claims?



Placement - Prioritizers

Prioritizers is used to rank the clusters filtered from the hard requirements, and choose the clusters with higher score.

- **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.
- **Customized score:** This is also called extensible scheduling. The customized score can be defined in addOn section.

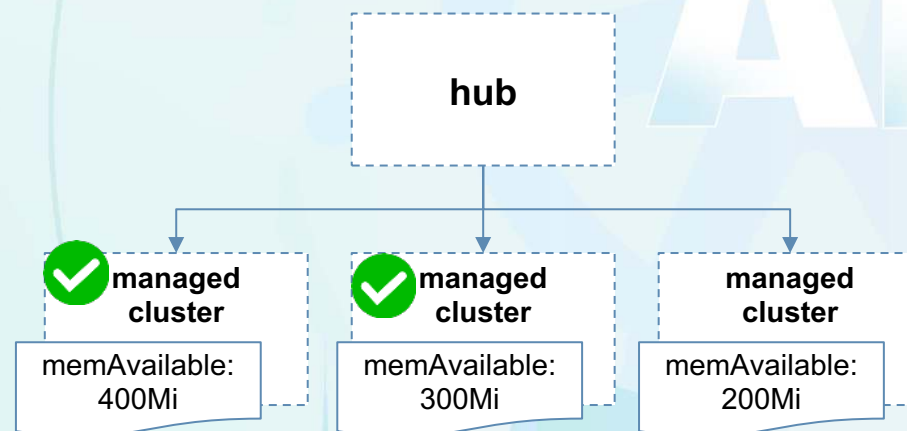


Placement - Prioritizers

I **can** sort the clusters by customized score memAvailable and select the top 2 clusters.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: default
spec:
  numberOfClusters: 2
  prioritizerPolicy:
    configurations:
      - scoreCoordinate:
          type: AddOn
          addOn:
            resourceName: default
            scoreName: memAvailable
```

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: AddOnPlacementScore
metadata:
  name: default
  namespace: {managed cluster namespace}
status:
  ...
  scores:
    - name: "memAvailable"
      value: 70
      quantity: 200Mi
    - name: "cpuAvailable"
      value: 80
      quantity: 1
```



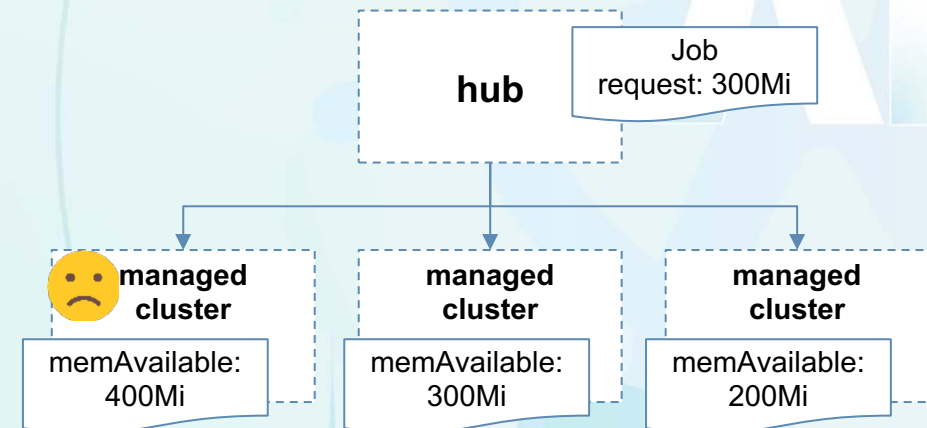
Placement - Prioritizers challenges

I **want to** maximize resource utilization (binpacking issue).

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: default
spec:
  numberOfClusters: 1
  prioritizerPolicy:
    configurations:
      - scoreCoordinate:
          type: AddOn
          addOn:
            resourceName: default
            scoreName: memAvailable
```

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: AddOnPlacementScore
metadata:
  name: default
  namespace: {managed cluster namespace}
status:
  ...
  scores:
    - name: "memAvailable"
      value: 70
      quantity: 200Mi
    - name: "cpuAvailable"
      value: 80
      quantity: 1
```

- My workload requires 300Mi memory and I want to maximize resource utilization on the managed clusters.
- Select the top 1 cluster is not the best choice.



Placement - Prioritizers challenges

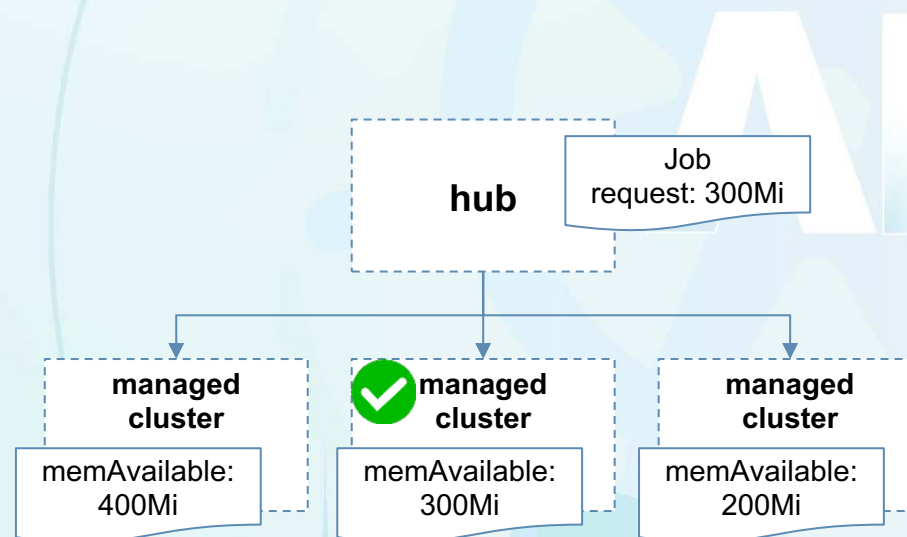
I **want to** maximize resource utilization (binpacking issue).

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: default
spec:
  numberOfClusters: 1
  prioritizerPolicy:
    configurations:
      - scoreCoordinate:
          type: AddOn
          addOn:
            resourceName: default
            scoreName: memAvailable
```



Can I filter the clusters based on the external resource before prioritizing?

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: AddOnPlacementScore
metadata:
  name: default
  namespace: {managed cluster namespace}
status:
  ...
  scores:
    - name: "memAvailable"
      value: 70
      quantity: 200Mi
    - name: "cpuAvailable"
      value: 80
      quantity: 1
```



The advanced multi-cluster scheduling challenges



Need a more flexible way to meet the scheduling challenges.

- **More flexible** operand.
- Schedule based on **more properties**, not only labels or claims.
- Schedule based on **external resource**.
- DO NOT change code every time when there's a new requirement.

AI

Part 02

通用表达式语言(CEL)

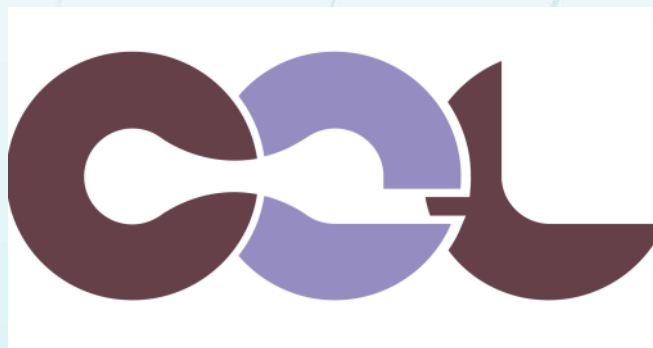
The common expression language (CEL)

AI



The common expression language (CEL)

Common Expression Language (CEL) is an expression language that's fast, portable, and safe to execute in performance-critical applications. CEL is designed to be embedded in an application, with application-specific extensions, and is ideal for extending declarative configurations that your applications might already use.



Fast

Accelerated expression evaluation in performance-critical paths from nanoseconds to microseconds.



Portable

Developer friendly, light weight with common syntax across multiple Google and external systems.



Extensible

Supports subsetting and extension, easy to embed and tailor to configuration and policy requirements.



Safe

Non-Turing complete, and only accesses data provided by the host application.

The common expression language (CEL)

CEL is used in

- Kubernetes: CRD validation rules, Validating Admission Policy, Mutating Admission Policy.
- Istio: MetricsOverrides, access log filter.
- Envoy Gateway: access log filter.
- Kyverno: CEL Preconditions and CEL Variables in Kyverno Policies.
- ...



The common expression language (CEL)

CEL expressions examples

Rule	Purpose
<code>self.minReplicas <= self.replicas && self.replicas <= self.maxReplicas</code>	Validate that the three fields defining replicas are ordered appropriately
<code>'Available' in self.stateCounts</code>	Validate that an entry with the 'Available' key exists in a map
<code>(self.list1.size() == 0) != (self.list2.size() == 0)</code>	Validate that one of two lists is non-empty, but not both
<code>self.envvars.filter(e, e.name == 'MY_ENV').all(e, e.value.matches('^[a-zA-Z]*\$'))</code>	Validate the 'value' field of a listMap entry where key field 'name' is 'MY_ENV'
<code>has(self.expired) && self.created + self.ttl < self.expired</code>	Validate that 'expired' date is after a 'create' date plus a 'ttl' duration
<code>self.health.startsWith('ok')</code>	Validate a 'health' string field has the prefix 'ok'
<code>self.widgets.exists(w, w.key == 'x' && w.foo < 10)</code>	Validate that the 'foo' property of a listMap item with a key 'x' is less than 10
<code>type(self) == string ? self == '99%' : self == 42</code>	Validate an int-or-string field for both the int and string cases
<code>self.metadata.name == 'singleton'</code>	Validate that an object's name matches a specific value (making it a singleton)
<code>self.set1.all(e, !(e in self.set2))</code>	Validate that two listSets are disjoint
<code>self.names.size() == self.details.size() && self.names.all(n, n in self.details)</code>	Validate the 'details' map is keyed by the items in the 'names' listSet

The common expression language (CEL)

CEL options, language features, and libraries.

- <https://github.com/google/cel-spec/blob/master/doc/langdef.md>

CEL option, library or language feature	Included
Standard macros	has, all, exists, exists_one, map, filter
Standard functions	Equality: ==, != Numbers: int, uint, and double Lists and Maps: a == b Regular Expressions ...
Extended strings library	charAt, indexOf, lastIndexOf, lowerAscii, upperAscii, replace, split, join, substring, trim

The common expression language (CEL)

CEL options, language features, and libraries.

- <https://kubernetes.io/docs/reference/using-api/cel/#kubernetes-cel-libraries>

Kubernetes CEL libraries

Kubernetes list library

Kubernetes regex library

Kubernetes URL library

Kubernetes authorizer library

Kubernetes quantity library

CEL Expression

Purpose

`"abc 123".find('[0-9]+')`

Find the first number in a string

`"1, 2, 3, 4".findAll('[0-9]+').map(x, int(x)).sum() < 100`

Verify that the numbers in a string sum to less than 100

CEL Expression

Purpose

`url('https://example.com:80/').getHost()`

Gets the 'example.com:80' host part of the URL

`url('https://example.com/path with spaces/').getEscapedPath()`

Returns '/path%20with%20spaces/'

`quantity("150Mi").isGreaterThan(quantity("100Mi"))`

Test if a quantity is greater than the receiver

`quantity("50M").isLessThan(quantity("100M"))`

Test if a quantity is less than the receiver

The common expression language (CEL)

Example: Kubernetes CRD validation rules

```
...
openAPIV3Schema:
  type: object
  properties:
    spec:
      type: object
      x-kubernetes-validations:
        - rule: "self.minReplicas <= self.replicas"
          message: "replicas should be greater than or equal to
minReplicas."
        - rule: "self.replicas <= self.maxReplicas"
          message: "replicas should be smaller than or equal to
maxReplicas."
      properties:
        ...
        minReplicas:
          type: integer
        replicas:
          type: integer
        maxReplicas:
          type: integer
      required:
        - minReplicas
        - replicas
        - maxReplicas
```

Validation rules use the Common Expression Language (CEL) to validate custom resource values. Validation rules are included in CustomResourceDefinition schemas using the x-kubernetes-validations extension.

x-kubernetes-validations describes a list of validation rules written in the CEL expression language.



The common expression language (CEL)

Example: Kubernetes Mutating Admission Policy (alpha)

```
apiVersion: admissionregistration.k8s.io/v1alpha1
kind: MutatingAdmissionPolicy
metadata:
  name: "sidecar-policy.example.com"
spec:
  paramKind:
    kind: Sidecar
    apiVersion: mutations.example.com/v1
  matchConstraints:
    resourceRules:
      - apiGroups: [""]
        apiVersions: ["v1"]
        operations: ["CREATE"]
        resources: ["pods"]
  matchConditions:
    - name: does-not-already-have-sidecar
      expression: "!object.spec.initContainers.exists(ic, ic.name == \"mesh-proxy\")"
  failurePolicy: Fail
  reinvokePolicy: IfNeeded
  mutations:
    - patchType: "JSONPatch"
      jsonPatch:
        expression: >
          [
            JSONPatch{
              op: "add", path: "/spec/initContainers/-",
              value: Object.spec.initContainers{
                name: "mesh-proxy",
                image: "mesh-proxy/v1.0.0",
                restartPolicy: "Always"
              }
            }
          ]
```

Kubernetes v1.32: Penelope highlights of new features in Alpha.

The Kubernetes API server now supports Common Expression Language (CEL)-based Mutating Admission Policies, providing a lightweight, efficient alternative to mutating admission webhooks.



Match conditions are **CEL** expressions. All match conditions must evaluate to true for the resource to be evaluated.

jsonPatch defines a JSON patch to perform a mutation to the object. A **CEL** expression is used to create the JSON patch.



Part 03

CEL在多集群调度的应用

Use CEL in multi-cluster scheduling

AI



Use CEL in multi-cluster scheduling

```
// ManagedClusterLib defines the CEL library for ManagedCluster evaluation.  
// It provides functions and variables for evaluating ManagedCluster properties  
// and their associated resources.  
//
```

```
// Variables:  
//  
// managedCluster  
//  
// Provides access to ManagedCluster properties.  
//
```

```
// Functions:  
//  
// scores  
//  
// Returns a list of AddOnPlacementScoreItem for a given cluster and AddOnPlacementScore resource name.  
//  
// scores(<ManagedCluster>, <string>) <list>  
//  
// The returned list contains maps with the following structure:  
// - name: string - The name of the score  
// - value: int - The numeric score value  
// - quantity: number|string - The quantity value, represented as:  
// - number: for pure decimal values (e.g., 3)  
// - string: for values with units or decimal places (e.g., "300Mi", "1.5Gi")  
//  
// Examples:  
//  
// managedCluster.scores("cpu-memory") // returns [{name: "cpu", value: 3, quantity: 3}, {name: "memory", value: 4, quantity: "300Mi"}]
```

```
// Semver Library:  
//  
// Semver provides a CEL function library extension for [semver.Version].  
//  
// Examples:  
//  
// semver("1.2.3").isLessThan(semver("1.2.4")) // returns true  
// semver("1.2.3").isGreaterThan(semver("1.2.4")) // returns false  
//
```

```
// Quantity Library:  
//  
// Quantity provides a CEL function library extension of Kubernetes resource.  
//  
// Examples:  
//  
// quantity("100Mi").isLessThan(quantity("200Mi")) // returns true  
// quantity("100Mi").isGreaterThan(quantity("200Mi")) // returns false
```



ManagedCluster Library

Variable managedCluster

Function scores()

Kubernetes semver library

Kubernetes quantity library

NOW I CAN

Select clusters with version < v1.31.0.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
```

```
kind: Placement
```

```
...
```

```
spec:
```

```
  predicates:
```

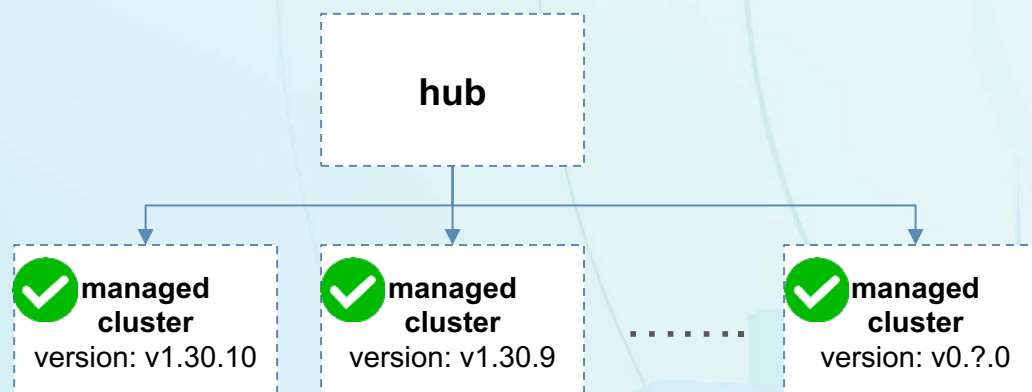
```
    - requiredClusterSelector:
```

```
      celSelector:
```

```
        celExpressions:
```

```
          - semver(managedCluster.metadata.labels["version"], true).isLessThan(semver("v1.31.0", true))
```

Upstream enhancement: semver support v is a prefix
<https://github.com/kubernetes/kubernetes/pull/130648>



NOW I CAN

Maximize resource utilization (resolve binpacking issue).

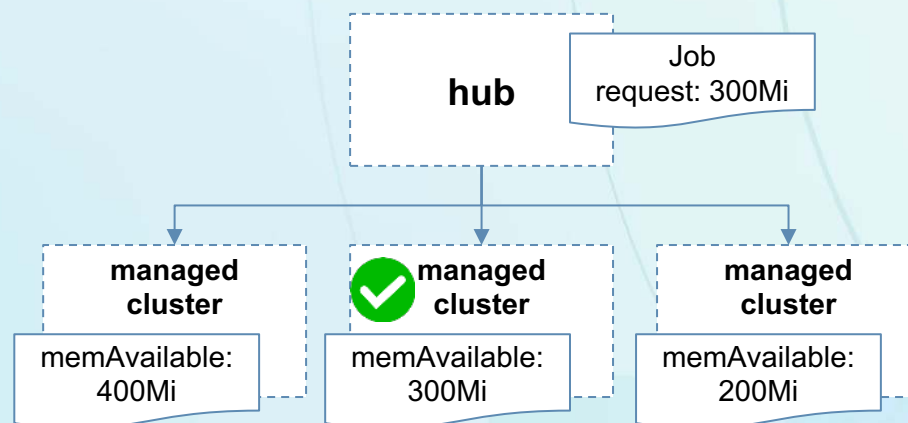
```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
```

```
...
spec:
  numberOfClusters: 1
  predicates:
    - requiredClusterSelector:
        celSelector:
          celExpressions:
            - managedCluster.scores("default").filter(s, s.name == 'memAvailable').all(e, quantity(e.quantity).isGreaterThan(quantity("300Mi")))
  prioritizerPolicy:
    configurations:
      - scoreCoordinate:
          type: AddOn
          addOn:
            resourceName: default
            scoreName: memAvailable
            weight: -1
```

3. select the top 1 cluster

1. Filter clusters memAvailable > 300Mi.
Select clusters based on scores before sorting.

2. sort cluster in reverse by memAvailable



NOW I CAN



Select clusters with other properties.

ClusterProfile API is a new API over at k8s-sigs, it is more useful with properties that can help workload orchestration decisions.

- **Core properties** are the properties that each implementation MUST provide.
 - cluster-endpoints.k8s.io
- **Standard properties** are the properties that each implementation MUST compliant to if it wish to implement.
 - location.topology.k8s.io
 - count.node.k8s.io
 - sku.node.k8s.io
 - metrics-endpoints.k8s.io
 - custom.resources.k8s.io
- **Extension properties** are defined by any implementation and not required to a corresponding about api.


```
apiVersion: multicluster.x-k8s.io/v1alpha1
kind: ClusterProfile
metadata:
  name: bravelion
  namespace: ...
...
properties:
  - name: cluster-endpoints.k8s.io
    value: 100.3.3.4:5683,qs-oar7gr9p.azmk8s.io:443
...
```

NOW I CAN

Select clusters with other properties.


```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
...
spec:
  predicates:
    - requiredClusterSelector:
        celSelector:
          celExpressions:
            - managedCluster.status.properties.exists(c, c.name == "sku.node.k8s.io" && c.value.split(",").exists(e, e == "g6.xlarge"))
```

Select clusters whose properties sku.node.k8s.io contains value g6.xlarge.



```
apiVersion: multicluster.x-k8s.io/v1alpha1
kind: ClusterProfile
metadata:
  name: bravelion
  namespace: ...
...
properties:
  - name: sku.node.k8s.io
    value: g6.xlarge,Standard_NC48ads_H100,m3-ultramem-32
...
```

Standard properties "sku.node.k8s.io", value is a comma separated string.



NOW I CAN

Select clusters with other properties.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
...
spec:
  predicates:
    - requiredClusterSelector:
        celSelector:
          celExpressions:
            - managedCluster.status.properties.
              exists(c, c.name == "sku.gpu.kubernetes-fleet.io"
                && c.value.parseJSON().H100.exists(e,
                  e.Standard_NC96ads_H100_v4 == 2)
```

```
apiVersion: multicluster.x-k8s.io/v1alpha1
kind: ClusterProfile
metadata:
  name: bravelion
  namespace: ...
...
properties:
  - name: sku.gpu.kubernetes-fleet.io
    value: |
      {
        "H100": [
          {
            "Standard_NC48ads_H100_v4": 10
          }
          {
            "Standard_NC96ads_H100_v4": 2
          }
        ]
        "A100": [
          {
            "Standard_NC48ads_A100_v4": 50
          }
          {
            "Standard_NC96ads_A100_v4": 20
          }
        ]
      }
    ...
```

Properties "sku.gpu.kubernetes-fleet.io" contains a list of gpuSku:count in a json format



NEXT

- Join the discussion
 - <https://github.com/open-cluster-management-io/enhancements/pull/137>
- Future work:
 - Runtime visibility: add metrics for time spent evaluating CEL at runtime.
 - A budget of how much time we'll spend on CEL at runtime.
 - Adopt the new k8s-sigs about-api project's ClusterProperty API. <https://github.com/open-cluster-management-io/ocm/issues/838>

Get Involved



- GitHub: <https://github.com/open-cluster-management-io/OCM>
- Website: <https://open-cluster-management.io/>
- 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/>

Thanks.

AI

