EMCO: Logical Clouds

Distributed Cloud Manager

Igor D.C. @igordcard @igordc
Ritu Sood
Cluster Registration Controller registers clusters by cluster owners
Distributed Application Scheduler provides simplified, and extensible placement; tenant mgmt; LCM implementation
Hardware Platform Aware Controller enables scheduling with auto-discovery of platform features/capabilities; Others: Cost, Power Savings, Latency aware... (WIP)
Distributed Cloud Manager presents a single logical cloud from multiple edges
Traffic Connectivity controller auto-configure service mesh (ISTIO) and security policy (NAT, firewall), DNS and SLB entities of edges - WIP
Day2 generic configuration configures Day2 configuration of any app/network function via templates & configs - WIP
Resource Synchronizer & Monitoring synchronizes resources across multiple edge/cloud platforms and then monitors the status of deployed resources
From the EMCO overview earlier

**Distributed Cloud Manager** presents a single logical cloud from multiple edges.
From the EMCO overview earlier.
From the EMCO overview earlier
Distributed Cloud Manager (DCM)

• DCM provides:
  
  – The ability to instantiate Logical Clouds, *i.e.* clouds of clouds.
    (clouds spanning multiple clusters)
Logical Clouds
Distributed Cloud Manager (DCM)

- DCM is one of the key microservices in EMCO.
- Logical Clouds - collections of clusters.
  - Geographically disperse
- Multitenancy support.
- Abstraction layer for different Cluster APIs.
Main DCM API paths

- /projects/PROJECT/logical-clouds
- /projects/PROJECT/logical-clouds/LC/cluster-references
- /projects/PROJECT/logical-clouds/LC/cluster-references/CR/kubeconfig
- /projects/PROJECT/logical-clouds/LC/cluster-quotas
- /projects/PROJECT/logical-clouds/LC/user-permissions
- /projects/PROJECT/logical-clouds/LC/apply
- /projects/PROJECT/logical-clouds/LC/terminate
- /projects/PROJECT/logical-clouds/LC/status
- /projects/PROJECT/logical-clouds/LC/stop
Types of Logical Clouds

- Administrative
- Standard
- Privileged
Administrative Logical Cloud

- Directly connect the K8s* clusters using credentials provided by the cluster manager microservice.

- Essentially using the default namespace to deploy any resource/application (including additional namespaces).

* assumes K8s clusters
Standard Logical Cloud

• Resources are installed in the K8s* clusters, starting with Namespace, to create a “partition” of the cluster (and between the clusters) to be used by EMCO.

• Limited applications can be deployed due to constrained access privileges.

* assumes K8s clusters
Privileged Logical Cloud

- Structure-wise, same as a Standard Logical Cloud.

- However, access privileges are associated to the Logical Cloud (at the namespace and cluster levels), as well as towards other namespaces.

- This allows for a significantly wider range of applications that can be deployed.

* assumes K8s clusters
Example of a Logical Cloud

metadata:
  name: permission1
spec:
  namespace: mynamespace
  apiGroups:
    - ""
    - "apps"
    - "k8splugin.io"
  resources:
    - secrets
    - pods
    - configmaps
    - services
    - deployments
    - resourcebundlestates
  verbs:
    - get
    - watch
    - list
    - create
    - delete

User Permission API resource

(defined this resource automatically promotes the Logical Cloud from Standard to Privileged)
Example of a Logical Cloud

metadata:
  name: myquota
spec:
  limits.cpu: '400'
  limits.memory: 1000Gi
  requests.cpu: '300'
  requests.memory: 900Gi
  requests.storage: 500Gi
  requests.ephemeral-storage: '500'
  limits.ephemeral-storage: '500'
  persistentvolumeclaims: '500'
  pods: '500'
  configmaps: '1000'
  replicationcontrollers: '500'
  resourcequotas: '500'
  services: '500'
  services.loadbalancers: '500'
  services.nodeports: '500'
  secrets: '500'
  count/replicationcontrollers: '500'

Cluster Quota API resource
Example of a Logical Cloud

```
metadata:
  name: lccluster1
spec:
  clusterProvider: cp1
  cluster: cp1-1

metadata:
  name: lccluster2
spec:
  clusterProvider: cp1
  cluster: cp1-2
```

Cluster Reference API resource (two of them)
Example of a Logical Cloud

```yaml
metadata:
    name: myprivilegedcloud

spec:
    namespace: mynamespace
    labels:
        x: y
        team: dev
    user:
        userName: user-1
        type: certificate

Logical Cloud API resource
```
Demo environment

(1st destination cluster)  
“cluster01”

(2nd destination cluster)  
“cluster02”

(EMCO services running on tmux)  
“frostcanyon”

My machine
<see recording>
CSR approval workflow
Future Work

(slide added after the session)

• Supporting more cluster backends (GitOps, etc.)

• Updating logical cloud details: add/remove cluster, permissions, quotas, etc.

• Up to full feature parity with using a K8s cluster directly (such as via kubectl).
Questions

<see recording>
Thank you!

- The EMCO development team at Intel 😊

https://project-emco.io/
https://gitlab.com/project-emco/core/emco-base
https://wiki.lfnetworking.org/display/EMCO/Welcome+to+the+EMCO+Wiki