

Agenda

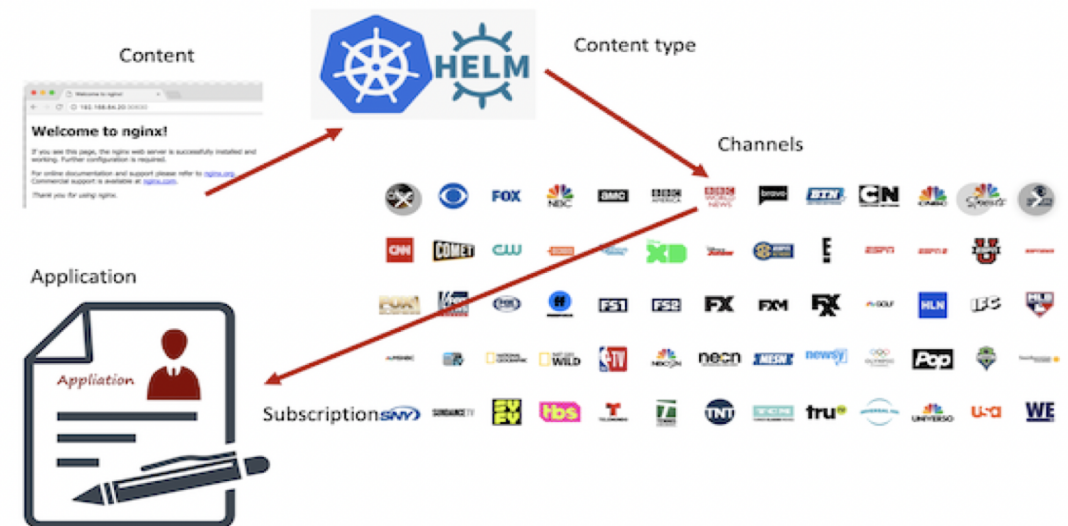
- How IBM Multicloud Manager is enhanced to help you continuous delivery
- Example project
- CP4MM & CP4A Use Cases
- How CP4MM adds value to CP4A

Enable and Deploy Applications with Policy Enforcement

- A repository can contain multiple subscriptions that can be deployed to one or more managed clusters by using a subscription custom resource definition.

In our example:

- All the applications which are packaged as helm charts will be hosted in one or more repositories.
- The repositories, which contain the application packages are defined as channels which broadcast across the clusters which are managed.
- If you want to deploy an application, you define a subscription to the channel with the name of the application (one or more) you want to deploy.
- This is similar to the subscription model of TV channels.



Application Model Overview

- **Subscriptions**

- Subscriptions identify deployables within channels by using annotations, labels and versions. Then, the subscription places the deployables (template or referenced Helm chart) on the subscribed target clusters.

- **Placement rules**

- Placement rules define the target clusters where subscriptions are delivered.

- **Deployables**

- Deployables are Kubernetes resources that contain templates that wrap other Kubernetes resources to be deployed. They are also used to represent Helm charts.

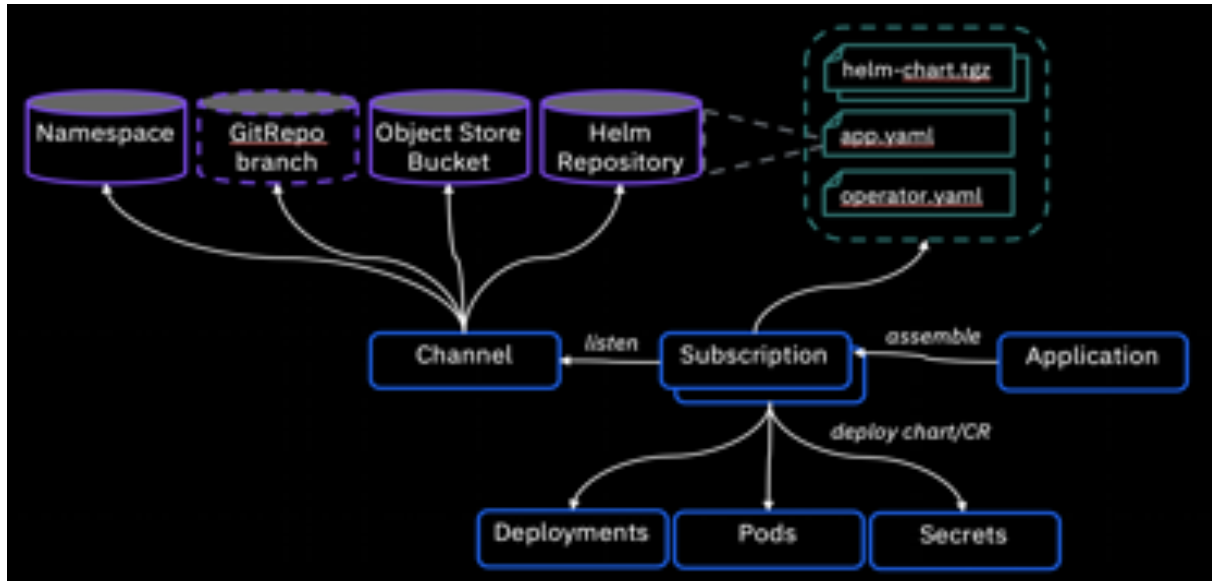
- **Channels**

- Channels point to repositories where Kubernetes resources are stored, such as a namespace, object store, or Helm repository. Channels use deployable resources to represent stored Kubernetes resources and Helm charts.

Application Model Overview

- Applications ([Application.app.k8s.io](https://application.app.k8s.io))
 - are used for grouping application components.
- Deployables (Deployable.app.ibm.com)
 - are Kubernetes resources that contain templates to wrap other Kubernetes resources or to represent Helm releases for deployment to clusters.
- Placement rules (PlacementRule.app.ibm.com)
 - define the target clusters where deployables can be deployed.
- Channels (Channel.app.ibm.com)
 - define a namespace within the hub cluster and point to a physical place where resources are stored for deployment; such as an object store, Kubernetes namespace, Helm repository, or GitHub repository.
- Subscriptions (Subscription.app.ibm.com)
 - are sets of definitions that identify deployables within channels by using annotations, labels, and versions. The subscription operator can monitor the channel for new or updated deployables, such as an updated Helm release or new Kubernetes deployable object.

Application Model Overview Diagram



Process:

- Define channels (namespace or helm repositories)
- Subscribe to the channels
- To deploy the applications across multiple clusters
- Depending on the subscription and placement policies.

Sample project orchestrating flow

In this example, we set up a channel:

- Point to the IBM Cloud Charts Helm Repository, which is a repository of IBM and third-party developer edition Helm charts.
- Sample project subscribes managed clusters to the Helm chart for IBM MQ Advanced for Developers.
- Subscription deploys the latest version of the chart
- Continues to update the deployed chart whenever a new version is published to the repository that is represented by the channel.
- You can use the same subscription to target multiple managed clusters so that you can update all clusters that need the same chart.

Creating the channels


- YAML files are used to create the channels and subscriptions for delivering and configuring resources.
 - Used to create a namespace type channel
 - and a Helm repository type channel
- The namespace type channel is used to hold secrets and configMaps
- The HelmRepo channel is used to identify the IBM Developer Edition Helm Chart repository on GitHub.

Creating the namespace type channels

```
[root@virtualserver01 multicloud-subscriptions-developer-editions-master]# more channels/1-secret-vault-channel.yaml
---
# Create a Namespace to hold the Channel "secrets"
apiVersion: v1
kind: Namespace
metadata:
  name: vault
---
# Create a Namespace to hold the Channel "secrets"
apiVersion: v1
kind: Namespace
metadata:
  name: developer-editions
---
# Create a Channel that declares resources that can be deployed
# Channel can be a Namespace, Object Store Bucket, or Helm Repo
apiVersion: app.ibm.com/v1alpha1
kind: Channel
metadata:
  name: secrets
  namespace: vault
spec:
  type: Namespace
  pathname: vault
[root@virtualserver01 multicloud-subscriptions-developer-editions-master]# kubectl apply -f ./channels/1-secret-vault-channel.yaml
namespace/vault created
namespace/developer-editions created
channel.app.ibm.com/secrets created
```

Channel

secrets

YAML 

0

Subscriptions in this channel

Channel (2) ^

Name ▼	Namespace	Cluster	Type	Pathname	Created	Labels
secrets	vault	local-cluster	Namespace	vault	16 minutes ago	

Creating the helm repository channel

```
[root@virtualserver01 multicloud-subscriptions-developer-editions-master]# kubectl apply -f ./channels/2-ibmcom-helm-channel.yaml
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
namespace/ibmcom configured
channel.app.ibm.com/ibm-developer-edition-charts created
configmap/insecure-skip-verify created
application.app.k8s.io/developer-editions created
placementrule.app.ibm.com/dev-placementrule created
placementrule.app.ibm.com/production-placementrule created
[root@virtualserver01 multicloud-subscriptions-developer-editions-master]#
```

Application name	Namespace	Managed clusters	Subscriptions (On managed clusters)	Policy violations	Created
developer-editions	developer-editions	0	0	0	a minute ago

Channel

ibm-developer-edition-charts

YAML

0

Subscriptions in this channel

0

Subscriptions in this channel

Applications / developer-editions / developer-editions

Overview

Resources

Incidents

Logs

application

Last update: 11:30:47

developer-editions

developer-editions

Search

```
1 apiVersion: app.k8s.io/v1beta1
2 kind: Application
3 metadata:
4   name: developer-editions
5   namespace: developer-editions
6   generation: 1
7 spec:
8   componentKinds:
9     - group: app.ibm.com/v1alpha1
10     kind: Subscription
11   descriptor: {}
12   selector:
13     matchExpressions:
14       - key: purpose
15         operator: In
16         values:
17           - developer-editions
```

Channels and subscriptions created

- Two channels and namespaces:
 - A secrets channel that uses the vault namespace. You can use this channel to store and apply secrets and configurations for the different developer edition charts.
 - An ibmcom channel that uses the ibmcom namespace. This channel represents the public IBM Helm repository.
- One developer-editions namespace:
 - This namespace is used to hold subscriptions, placement rules, and application resources
- Two placement rules that search for managed clusters:
 - The dev-placementrule rule targets managed clusters with the label
 - environment: Dev
 - The production-placementrule rule targets managed clusters with the label
 - environment: Production
- One application resource:
 - This resource associates subscriptions and placement rules by using a labelSelector on purpose: developer-editions.

Kubernetes resources created on the hub cluster

- Namespaces:
 - vault
 - developer-editions
 - ibmcom
- Channels:
 - A channel that is called secrets.
 - A channel that is called ibmcom.
- Application:
 - A developer-editions application to group all developer-editions subscriptions and placement rules together.
- Placement rules:
 - dev-placementrule
 - production-placementrule

Creating the Helm repository subscription

With the Helm chart channels set up:

- You can create a subscription to the IBM Cloud Charts Helm Repository to install the latest version of the IBM MQ Advanced for Developers chart.
- This YAML file is used to create the subscription resource for the Helm chart, an application resource, and a subscription resource to deliver the IBM MQ Administrator secret.
- The application resource is used to tie one or more subscriptions together for dependencies.
- This relationship allows the IBM Multicloud Manager management console to display the different resources that are associated with the deployed application:

Subscribe a managed cluster to an IBM Developer Edition Chart

- A subscription monitors the Helm repository for new versions of a chart.
- If no chart is deployed or if a new version of a deployed chart is found, the chart or the new version of the chart is deployed.
- The mq-adv-server-dev chart requires two resources to be deployed:
 - One is the resource that initiates the Helm install
 - the other is a secret that contains the password to be configured for the IBM MQ Administrators.

Apply subscriptions

```
[root@virtualserver01 multicloud-subscriptions-developer-editions-master]# kubectl apply -f ./subscriptions/3-mqadvanced-subscription.yaml
deployable.app.ibm.com/mq-secret-dev created
subscription.app.ibm.com/mq-advanced-server-secret-dev created
subscription.app.ibm.com/mq-advanced-server-dev created
[root@virtualserver01 multicloud-subscriptions-developer-editions-master]#
```

Applications / developer-editions / developer-editions

App monitoring | Grafana | Edit app

Overview Resources Incidents Logs

application cluster deployable rules subscription other

All channels
All subscriptions

ibm-developer-edition-charts
mq-advanced-server-dev

secrets
mq-advanced-server-secret-dev

Last update: 11:36:17

```
graph TD
    A[developer-editions] --> B[mq-advanced-server-dev]
    A --> C[mq-advanced-server-secret-dev]
    B --> D[op4mm-cluster]
    C --> D
    D --> E[dev-placementrule]
    E --> F[mq-secret-dev]
```

Search

```
1 apiVersion: app.k8s.io/v1beta1
2 kind: Application
3 metadata:
4   name: developer-editions
5   namespace: developer-editions
6   generation: 1
7 spec:
8   componentKinds:
9     - group: app.ibm.com/v1alpha1
10     kind: Subscription
11   descriptor: {}
12   selector:
13     matchExpressions:
14       - key: purpose
15         operator: In
16         values:
17           - developer-editions
18 ---
19 apiVersion: app.ibm.com/v1alpha1
20 kind: Subscription
21 metadata:
22   name: mq-advanced-server-dev
23   namespace: developer-editions
24   generation: 1
25   labels:
26     purpose: developer-editions
27 spec:
28   name: ibm-mqadvanced-server-dev
29   channel: ibmcom/ibm-developer-edition-charts
30   packageFilter:
31     version: 4.1.1
32   packageOverrides:
33     - packageName: ibm-mqadvanced-server-dev
34     packageOverrides:
35       - path: spec.values
36         value: >
37       # Additional values:
38       https://github.com/IBM/charts/tree/master/stable/ibm-mqadvanced-server-dev
39
40   persistence:
41     enabled: false
42     useDynamicProvisioning: false
43   license: accept
```

Channel	YAML
secrets	ibm-developer-edition-charts
0 Subscriptions in this channel	0 Subscriptions in this channel
1 Subscriptions in this channel	1 Subscriptions in this channel
Subscription mq-advanced-server-secret-dev Namespace: developer-editions Placement Rule: dev-placementrule	Subscription mq-advanced-server-dev Namespace: developer-editions Placement Rule: dev-placementrule
0% Resources completed	0% Resources in progress
0% Resources failed	0% Resources failed

Kubernetes resources created on the hub cluster

- Deployable:
 - An mq-secret-dev deployable is created in the vault/secret channel. This deployable contains a Kubernetes secret that has the IBM MQ Administrators password set in base64.
- Subscriptions:
 - An mq-advanced-server-secret-dev subscription that delivers the Kubernetes secret that is inside the mq-secret-dev deployable to the managed clusters that match the associated placement rule.
 - An mq-advanced-server-dev subscription that propagates to the managed clusters defined in the placement rule, and then deploys the mq-adv-server-dev Helm chart. This subscription also contains Helm values overrides for the chart.

View subscriptions

```
# kubectl -n developer-editions describe subscriptions
```

```
Status:
  Last Update Time: 2020-02-11T10:03:04Z
  Phase:            Propagated
  Statuses:
    Cp 4 Mm - Cluster:
    Tag - Cluster:
    Packages:
      Ibm - Developer - Edition - Charts - Ibm - Nginx - Dev - 1 . 0 . 1:
        Last Update Time: 2020-02-11T10:03:03Z
        Phase:            Subscribed
        Resource Status:
          Last Update: 2020-02-10T21:02:08Z
          Phase:         Success
Events:
  <none>
```

All applications (3)

Find resources

Application name	Namespace	Managed clusters	Subscriptions (On managed clusters)	Policy violations
developer-editions	developer-editions	1	1	0

Applications / developer-editions /

developer-editions

[App monitoring](#) | [Grafana](#) | [Edit app](#)

[Overview](#) [Resources](#) [Incidents](#) [Logs](#)

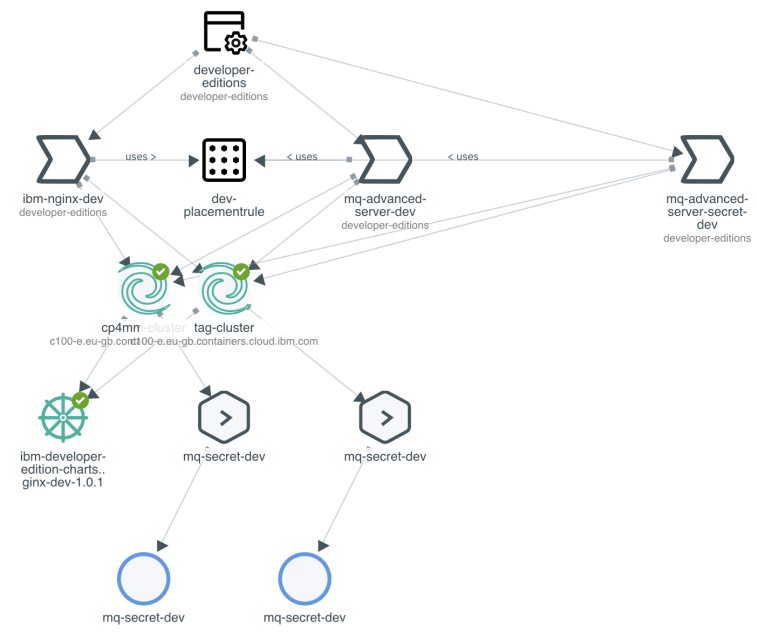
Resource highlights

<div>3</div> <div>SUBSCRIPTIONS</div> <div>On hub cluster</div> <div>0 failed</div>	<div>1</div> <div>MANAGED CLUSTER</div> <div>1 Total subscription</div> <div>0 failed</div>	<div>1</div> <div>POD</div> <div>0 running</div> <div>0 failed</div>	<div>0</div> <div>POLICY VIOLATIONS</div>	<div>0</div> <div>INCIDENTS</div>
---	---	--	---	-----------------------------------

> [View additional details](#)

Resource topology

- All channels
 - All subscriptions
- ibm-developer-edition-charts
 - ibm-nginx-dev
- ibm-developer-edition-charts
 - mq-advanced-server-dev
- secrets
 - mq-advanced-server-secret-dev



IBM Multicloud Manager subscriptions

- When the subscription is created on the hub cluster with a placement rule, the placement rule is evaluated.
- All managed clusters that match the placement rule get propagated a copy of the subscription.
- Each propagated subscription on the managed cluster completes a different action
 - The type: Namespace subscription subscribes the value namespace on the hub cluster and delivers the Kubernetes secret (IBM MQ Administrator password) by finding the deployable resource in that namespace and extracting the spec.template.
 - The type: HelmRepo subscription subscribes to the Helm repository path that was specified in the channel.
 - The subscription identifies the correct Helm release and version in the source repository, and then creates a HelmRelease resource on the managed cluster.
 - The HelmRelease resource then deploys the Helm release by communicating with Tiller.

CP4MM & CP4A Use Cases

With the introduction of the concepts of Channels and Subscription within the MCM application lifecycle context, the process of CI/CD has become far more streamlined.

Applications need to be deployed from MCM using the application specification yaml?

My question is how does this work from a CI/CD perspective?

How can I go from pipeline to deployment and get the benefits of the Application Lifecycle Management features in MCM?

- You have 2 channels pointing to a helm repo or a github repository, one for Dev and one for Production.
 - There are subscriptions associated with these channels to distribute the apps to the correct clusters using PlacementRules and other things (labels etc.)
 - Using your normal CI/CD process from CP4A, the developer will deliver the latest version of the chart or code to the HelmRepo or Github repository.
 - Now automatically the newest version of the chart/code is available in the channel.

CP4MM & CP4A Use Cases

- Additionally, you can setup gate conditions in the channel to route only the content that satisfies those conditions to flow into the channels.
 - For instance, the production channel can have a gate condition requiring the content to have a label as "production-ready".
 - Only those deployables that have this gate condition (i.e. label or annotation to indicate that they have been tested and ready to be promoted to prod) will be promoted to the Production channel.
 - Based on what the subscriptions are setup as (chart version or latest etc.), the subscription controller on the managed endpoints will automatically get this latest version and deploy it onto the clusters.
- Additionally, CP4MCM will provide Event management and Application performance monitoring
 - like event, incidents and alerts, including running automated runbooks, which would be key for an SRE after the apps have been deployed.
 - This is again particularly helpful when talking about (but not limited to) a multi-cluster deployment.
- The channel and subscription methodology works alongside your existing CI/CD pipeline to make things work especially for a multi-cluster application.

CP4MM & CP4A Value

- As a traditional application is modernized, you will need to have visibility into the traditional VM based app as well as containerized version
- As you create/build a greenfield application over time to strangle the monolith application, they need to be managed consistently during transition
- As you leverage CP4I to connect back end system to greenfield app and monolith application, you need those connections to be ensure no unauthorized access of the cluster running those APIs. Can create exposure to CP4Apps workloads without all connections governed
- Ensure authorized placement locations for all applications automatically with placement rules
- Create new containers instantly as more workloads are containerized (abstract away from the developer standing up new kubernetes)