Red Hat Advanced Cluster Management for Kubernetes 2.0

管理应用程序

管理应用程序

Last Updated: 2020-09-17

Red Hat Advanced Cluster Management for Kubernetes 2.0 管理应用程序

管理应用程序

法律通告

摘要

管理 Red Hat Advanced Cluster Management for Kubernetes 中的应用程序

目录

第 1 章 管理应用程序

查看以下主题以了解更多有关创建、部署和管理应用程序的信息。本指南假定您对 Kubernetes 概念和术语有一定的了解。Kubernetes 关键术语和组件在此文档中并没有详细定义。有关 Kubernetes 概念的更多信息，请参阅 Kubernetes 文档。

应用程序管理功能为您提供了构建和部署应用程序及应用程序更新的统一和简化的选项。通过这些功能，开发人员和运维（DevOps）人员可通过基于频道和订阅的自动化功能在环境之间创建和管理应用程序。

请参见以下主题：

- 应用程序管理生命周期
- 应用程序模型和定义
- 应用程序资源
- 使用控制台管理应用程序
- 创建和管理频道
- 创建和管理订阅
- 创建和管理放置规则
- 创建和管理应用程序资源
- 使用应用程序资源进行部署
- 应用程序资源示例

1.1. 应用程序管理生命周期

应用程序模型基于订阅一个或多个 Kubernetes 资源仓库（repository）（*频道*资源），其中包含部署在受管集群上的资源。

*订阅（subscription）*组件使用*放置规则（placement rule）*资源来定义将要部署 Kubernetes 资源的受管集群。

应用程序模型中的最后一部分是*应用程序（application）*资源，它引用一个或多个仓库订阅。应用程序的目的是对部署的 Kubernetes 资源进行分组，并提供容易理解和管理的数据聚合。

从以下主题了解更多有关应用程序生命周期的信息：

- 应用程序模型和定义
- 使用控制台管理应用程序

1.1.1. 应用程序模型和定义

应用程序模型由以下 Kubernetes 自定义资源组成：频道、订阅、放置规则和应用程序。

- 频道 (**channel.apps.open-cluster-management.io**) 定义的是集群可通过订阅来订阅的源仓库，可以是以下类型：Git 仓库、Helm 发行 registry、项存储和 hub 集群上的资源模板（可部署）命名空间。

注： 最佳实践是在每个命名空间中创建一个频道。但是，Git 频道可以与其他类型的频道共享命名空间，包括 Git、Helm、Kubernetes 命名空间和对象存储。

- 订阅 (**subscription.apps.open-cluster-management.io**) 允许集群订阅源仓库（频道），可以是以下类型：Git 仓库、Helm 发行 registry、项存储或资源模板（可部署）命名空间。订阅可以应用到本地的枢纽（hub），也可以应用到受管集群。

- 放置规则 (**placementrule.apps.open-cluster-management.io**) 定义的是订阅部署和维护 Kubernetes 资源的目标集群。可以使用放置规则可帮助您促进多集群部署。可在订阅间共享放置规则。

- Red Hat Advanced Cluster Management for Kubernetes 中的应用程序 **(application.app.k8s.io)** 用于对组成应用程序的 Kubernetes 资源进行分组。

有关创建和管理应用程序资源的更多信息，请参阅：

- 创建和管理频道
- 创建和管理订阅
- 创建和管理放置规则
- 创建和管理应用程序资源

1.1.2. 使用控制台管理应用程序

控制台包括用于管理应用程序生命周期的仪表板。您可以使用控制台仪表板来创建和管理应用程序资源。您还可以查看应用程序的状态。仪表板包括增强的功能,开发人员和操作人员可使用该功能在集群中创建、部署、更新、管理和视觉化应用程序。

请参阅以下应用程序控制台功能：

- 使用拓扑视图可视化集群中部署的应用程序，包括任何关联的频道和订阅。
- 访问一个拓扑视图，其中包含新的应用程序资源定义，包括频道、订阅和放置规则。
- 查看应用程序上下文中的单个状态，包括部署、更新和订阅。
- 添加并编辑频道、订阅、放置规则及应用程序。

所有资源的示例位于应用程序资源示例文档中。

控制台包括不同的工具，它们各自提供不同的应用程序管理功能。通过这些功能，您可以轻松地创建、查找、更新和部署应用程序资源。

- 应用程序仪表板
- 搜索
- 资源拓扑

1.1.2.1. 应用程序仪表板

在主 *Applications* 仪表板中，您可以查看所有应用程序的信息，也可以选择并查看特定应用程序的信息。

在该仪表板上的 *Overview* 选项卡中，您可以为所有应用程序完成以下任务：

- 查看列出所有应用程序的表。
- 使用 *Find resources* 复选框过滤列出的应用程序。
- 查看应用程序名称和命名空间。
- 查看通过订阅部署应用程序的受管集群数量。
- 查看 *hub* 集群上用于部署应用程序的订阅数及相关状态。
- 查看应用程序的创建日期。
- 点击 **Options** 可以进行更多操作，如 **Delete application**。
- 点击表中的应用程序名称查看该特定资源的更多详情。
- 通过 *Resources* 选项卡访问资源管道。
- 要查看 *Resources*，点 **Subscription** 进入 *Search* 页面，然后查看 hub 集群订阅和部署到受管集群的所有订阅。
- 查看不同卡上所有应用程序的资源摘要。这些资源摘要卡显示以下有关应用程序和相关受管集群的信息：
  - hub 集群上的订阅数和无法部署到 hub 集群的订阅数。
  - 受管集群的总数。
  - 这些受管集群的活跃订阅数。
  - 所有应用程序的频道和放置规则的总数。

您可以点击每个摘要卡打开 *Search* 页面并显示有关所选资源的更多信息。

- 在本页中，您还可以点击多个 **Create** 选项来使用 YAML 编辑器创建更多资源。
  当您选择创建任何这些资源时，会打开 YAML 编辑器，供您用来定义资源。包含了默认示例 YAML 内容作为模板，用于指示所需字段以帮助您创建任何这些资源类型。

  查看所有应用程序的资源，其中包含一个显示应用程序和其他资源详情的表：
  - 每行提供单个应用程序的详情。所有应用程序会包括相应的行。如果需要，您可以使用搜索框过滤应用程序行来查找匹配的应用程序。
  - 展开每个应用程序的行可查看相关频道、订阅和放置规则的更多详情，查看相关事件，以及查看资源 (Pod) 部署的状态。
  - 选择频道和应用程序的关联订阅，查看有关该订阅的更多详情，包括订阅的可部署资源、关联的频道、命名空间和主机集群、任何相关资源 (Pod) 部署的状态，以及用于该订阅的放置设置。

1.1.2.1.1. 应用程序仪表板（单个应用程序）

在单个应用程序仪表板的 *Overview* 选项卡中，您可以完成以下任务：

- 查看不同卡上应用程序的主要资源摘要。这些资源摘要卡显示以下信息：
  - Hub 集群上用于应用程序的订阅数和无法部署到 hub 集群的订阅数。

  - 使用应用程序的受管集群的总数。此卡还显示这些受管集群上用于应用程序的订阅总数，以及无法部署到这些受管群集的订阅数。
  - 应用程序的 Pod 数量，包括正在运行和失败的数量。
  - 与应用程序相关的策略违反和事件数量。您可以点击每个摘要卡打开 *Search* 页面并显示有关所选资源的更多信息。

- 选择查看应用程序的附加详情，包括应用程序命名空间、版本、创建日期、关联标签和注解等等。
- 查看应用程序的资源拓扑，其中显示应用程序和相关组件，如频道、部署、相关服务、Pod 和放置规则。您可以选择编辑拓扑，并为应用程序和相关资源更新 YAML 定义，如添加、更改或移除资源。

在单个应用程序的 *Resources* 选项卡中，您可以完成以下任务：

- 查看应用程序的资源列表。此列表包含与应用程序关联的所有资源，包括 Pod、secret、服务等。此列表显示了每个资源的名称、命名空间、类型、API 组和状态。该列表还显示了部署了资源的集群、资源的创建日期以及资源的最后更新日期。
- 查看应用程序的资源管道，其中包含应用程序资源摘要、创建资源的选项以及一个提供了更多资源详情的表。单个应用程序的资源管道包括的信息与所有应用程序的管道类似。两个管道之间的区别在于此管道的范围仅限于所选应用程序。

1.1.2.2. 搜索

控制台 *Search* 页面支持按每个资源的组件 **kind** 搜索应用程序资源。要搜索资源，请使用以下值：

应用程序资源	类型（搜索参数）
Application	**Application**
频道	频**道**
可部署资源	**可部署**资**源**
Secret	**Secret**
放置规则	**PlacementRule**
Subscription	**Subscription**

您还可以按其他字段搜索，包括名称、命名空间、集群、标签等。

在搜索结果中，您可以查看每个资源的标识详情，包括名称、命名空间、集群、标签和创建日期。

如果需要，您也可以在某个资源的搜索结果中展开 *Options* 菜单，以选择删除该资源。

点击搜索结果中的资源名称后，会打开 YAML 编辑器并显示资源的 YAML 定义。您可以选择在编辑器中编辑定义。您保存的任何更改都会立即应用到相应资源。

有关使用搜索的更多信息，请参阅在控制台中进行搜索。

1.1.2.3. 资源拓扑

应用程序拓扑包括用于显示状态的应用程序卡的视觉化表示。每个应用程序的拓扑视图包括该应用程序的所有服务、部署、chart 和 Pod。

- 您可以在拓扑视图中选择任何组件来查看更多详情。
- 您可以将光标悬停在资源上，以查看组件类型、名称和命名空间，并通过链接查看资源或命名空间的搜索结果。
- 查看 pod 的详情。您可以选择查看该 pod 的日志。
- 查看集群 CPU 和内存。
  注： 显示的集群 CPU 和内存百分比是当前使用的百分比。这个值会被进行舍入处理，因此一个很小的值可能会显示为 **0**。

  = 应用程序资源

在 Red Hat Advanced Cluster Management for Kubernetes 中，应用程序由多个应用程序资源组成。Red Hat Advanced Cluster Management for Kubernetes 应用程序的基本资源是 **application** 资源和 **deployable** 资源。

另外，您可以使用频道、订阅和放置规则资源来帮助部署、更新和管理整个应用程序。

单集群和多集群应用程序使用相同的 Kubernetes 规格，但多集群应用程序涉及更多的部署和应用程序管理生命周期自动化。

Red Hat Advanced Cluster Management for Kubernetes 应用程序的所有应用程序组件资源都在 YAML 文件 spec 部分中定义。当需要创建或更新应用程序组件资源时，需要创建或编辑正确的 spec 部分，使其包含用于定义资源的标签。

如果您的应用程序需要 secret 资源才能运行，您可以使用订阅将 secret 部署到需要资源的受管集群。

如果您的应用程序需要的 Kubernetes 资源或 Helm chart 来自需要授权的频道，如授权 Git 仓库，您可以使用 secret 提供对这些频道的访问。您可以在订阅中包含对 secret 的引用，以便为您的订阅提供访问安全频道所需的凭证。通过此访问权限，您的订阅可以在保持数据安全的同时访问从这些频道部署的 Kubernetes 资源及 Helm chart。

查看以下应用程序资源部分：

1.1.3. Channels

频道 (**channel.apps.open-cluster-management.io**) 为您提供改进的持续集成以及持续交付功能，以便创建和管理 Red Hat Advanced Cluster Management for Kubernetes 应用程序。频道是自定义资源定义，可帮助您简化部署并将集群访问分开。

有关创建和管理频道的更多信息，请参阅创建和管理频道。请参阅应用程序资源示例获取示例 YAML 文件。

频道在 hub 集群中定义命名空间，并指向存储了用于部署的资源的物理位置，如项存储、Kubernetes 命名空间、Helm 仓库或 Git 仓库。集群可以订阅频道，以便标识要部署到每个集群的可部署资源。

- 频道中的可部署资源只能供订阅了该频道的集群访问。
  可部署资源的放置、覆盖和依赖项都在可部署资源的 spec 中定义。还可以在订阅中定义可部署资源的放置，例如在多集群部署中。

- secret 属于 Kubernetes 资源，可用于存储授权和其他敏感信息，如密码、OAuth 令牌和 SSH 密钥。
  通过将这些信息存储为 secret，您可以将信息与需要相应信息的应用程序组件分开以提高数据安全性。

请参阅管理 secret。请参阅 Secret 示例获取示例 YAML 文件。

对于 **Namespace** 和 **ObjectBucket** 频道类型，每个频道的 spec 可定义可部署资源要包含在相应频道中所必须满足的条件。这些条件定义为频道的 Kubernetes 标签，如源命名空间、软件包名称、标签和注解。可部署资源必须具有相同的标签才能包含在频道中。只有为可部署资源添加了与频道相同的标签时，可部署资源才可以包含在频道中。

1.1.4. 订阅（Subscription）

和频道一样，订阅 (**subscription.apps.open-cluster-management.io**) 为您提供改进的持续集成和持续交付功能以用于应用程序管理。有关创建和管理订阅的信息，请参阅创建和管理订阅。请参阅应用程序资源示例获取示例 YAML 文件。

订阅是通过使用注解、标签和版本在频道内标识 Helm chart、可部署资源和其他 Kubernetes 资源的定义集合。订阅可指向某个频道或存储位置，以便标识新的或已更新的可部署资源。然后，订阅 operator 可以在不先检查 Hub 集群的情况下，直接从存储位置下载订阅的 Helm chart、可部署资源或 secret 到目标受管集群。通过订阅，订阅 operator 可以监控该频道是否有新的或已更新的资源，而不是监控 Hub 集群。

1.1.5. 放置规则

放置规则 (**placementrule.apps.open-cluster-management.io**) 定义的是可以部署可部署资源的目标集群。使用放置规则帮助您促进可部署资源的多集群部署。有关创建和管理放置规则的更多信息，请参阅创建和管理放置规则。请参阅放置规则示例获取示例 YAML 文件。

放置规则的自定义资源定义 (CRD) 和控制器替代了用于之前 Red Hat Advanced Cluster Management for Kubernetes 版本中的应用程序的放置策略。放置策略仍用于监管和风险策略。

可以为订阅和可部署资源定义放置规则。在订阅级别为多集群部署定义放置规则。定义放置规则以用于单集群部署的特定可部署资源或用于覆盖放置设置。

1.1.6. Application

Red Hat Advanced Cluster Management for Kubernetes 中的应用程序 (**Application.app.k8s.io**) 用于查看应用程序组件。请参阅应用程序资源示例获取示例 YAML 文件。

1.1.6.1. 创建和管理频道

创建并使用频道来为 Red Hat Advanced Cluster Management for Kubernetes 应用程序的持续集成和交付功能提供应用程序资源。

频道是自定义资源，通过使用频道，可以把管理资源的操作与将这些资源部署到受管集群的操作独立开。所有资源（包括频道）的示例位于应用程序资源示例文档中。

频道（**channel.apps.open-cluster-management.io**）指向存储了用于部署的资源的物理位置。它们在其中也会存储与该频道相关的其他资源 hub 集群中的一个命名空间中创建。。

共有四种类型的频道。每种频道根据资源存储的源位置类型而有所不同：

- Kubernetes 命名空间（**Namespace**）：使用可部署资源来存储 Kubernetes 资源模板。此类型的频道的订阅将检索并部署模板。由该频道监控是否有新的或已更新的可部署资源的命名空间必须位于 hub 集群上。

- 对象存储（**ObjectBucket**）：存储 Kubernetes 资源 YAML 文件。每个 YAML 文件都包括一个资源的模板部分，而不是完整的可部署对象。对象存储可以使用位于 hub 集群上的可部署对象或直接从持续集成管道进行填充，也可以通过将所需的 YAML 文件包含到对象存储中进行填充。

- Helm 仓库（**HelmRepo**）：Stores Helm chart。有关如何构建 chart 的信息，请参阅 Helm 文档。

- Git 仓库（**Git**）：Stores Kubernetes 资源 YAML 文件和解包的 Helm chart。这些资源不需要嵌套或表示为可部署资源。频道控制器自动将资源作为可部署资源同步。
  请参阅列出的 **Git** 源：

  - GitHub
  - GitLab
  - Bitbucket
  - Gogs（webhook 不支持）

通过指向保存资源的位置在 hub 集群中创建频道。频道使用 **Channel** 自定义资源（CR）定义。

注： 最佳实践是在每个命名空间中创建一个频道。但是，Git 频道可以与其他类型的频道共享命名空间，包括 Git、Helm、Kubernetes 命名空间和对象存储。

当集群订阅该频道时，频道命名空间可供受管集群上的订阅 operator 使用。然后 operator 就可以获取 secret 以访问实际的频道。Operator 需要该访问权限来检查可部署资源是否满足频道要求。

如需更多信息应用程序资源，请参阅 应用程序资源。

了解有关频道的更多信息，然后查看以下任务：

- 创建频道
- 更新频道
- 删除频道
- 使用频道管理部署
- 频道状态和同步

1.1.6.1.1. 创建频道

1. 编写频道定义 YAML 内容。要创建或更新频道资源，您必须首先编写定义该资源的 YAML 文件。有关 YAML 结构的更多信息，包括所需字段，请参阅频道定义 YAML 结构。

2. 确保您在唯一命名空间中创建频道。所有频道需要单独的命名空间，但 Git 频道除外，它们可与另一个频道共享命名空间。

3. 可选。如果您要创建 **Namespace** 类型频道，请在您的 hub 集群上创建命名空间。
   可使用频道 spec 的 **spec.sourceNamespaces** 和 **spec.type** 字段来指定频道类型。当某个频道已创建并指向某个源时，频道控制器会将可部署资源的提升保持在该源。控制器还会同步源和频道命名空间。

您可以将命名空间定义为 YAML 定义的一部分，也可以使用 Kubernetes 命令行界面 (**kubectl**) 工具来创建命名空间。要使用 Kubernetes CLI 工具，请运行以下命令。将 **namespace name** 替换为新命名空间的名称。

```
kubectl create namespace <namespace name>`
```

4. 可选。如果您需要或希望使用 Kubernetes Secret 以验证频道对仓库或图表的访问权限，请创建 Kubernetes Secret 资源。您可以使用 Kubernetes 命令行界面 (**kubectl**) 工具，通过运行以下命令来创建 Kubernetes Secret：

```
kubectl create secret <secret name> generic --from-literal=user --from-literal=password=
```

您只能对 **HelmRepo**、**ObjectBucket** 和 **Git** 类型的频道使用 secret 进行身份验证。要将 secret 与频道关联，请在频道 YAML 定义中包括 **spec.secretRef.name** 设置。

5. 在 Red Hat Advanced Cluster Management for Kubernetes 中创建频道。您可以使用控制台、Kubernetes 命令行界面 (**kubectl**) 工具或 REST API：

- 使用控制台，

i. 在导航菜单中点击 **Manage applications**。所有应用程序的 **Overview** 选项卡都会打开。

ii. 点击 **Resources** 选项卡。

iii. 滚动到 *Resource pipeline* 部分。在按钮列表中找到并点击 **Channel**。此时会显示 *Create a Channel* 编辑器。

iv. 输入 YAML 内容以定义频道，或直接更新默认 YAML 模板以满足您的要求。

v. 完成后，点击 **Save** 创建频道。您的新频道会显示在相应应用程序的 **Resource pipeline** 中。

另外，您可以选择在使用特定应用程序时创建频道。

i. 在所有应用程序的 **Overview** 选项卡中点击 **All applications** 列表中的应用程序。该应用程序的 **Overview** 选项卡将打开。

ii. 点击该应用程序的 **Resources** 选项卡。

iii. 滚动到 *Resource pipeline* 部分。在资源摘要卡右侧的按钮列表中点击 **Channel**。此时会显示 *Create a Channel* 编辑器。

iv. 输入 YAML 内容以定义频道，或直接更新默认 YAML 模板以满足您的要求。

v. 完成后，点击 **Save** 创建频道。您的新频道会显示在应用程序的 **Resource pipeline** 中。

- 使用 Kubernetes CLI 工具，

vi. 使用您首选的编辑工具编写并保存您的频道 YAML 文件。

vii. 运行以下命令，将您的文件应用到 apiserver。将 **filename** 替换为您的文件名称：

```
kubectl apply -f filename.yaml
```

viii. 运行以下命令验证您的频道资源是否已创建：

> kubectl get Channel

确保您的新频道已在生成的输出中列出。** 要使用 REST API，请使用 APIs。

1.1.6.1.2. 更新频道

1. 编写您的频道的定义更新。有关 YAML 结构的更多信息，包括所需字段，请参阅频道定义 YAML 结构。
2. 更新定义。您可以使用控制台、Kubernetes 命令行界面 (**kubectl**) 工具或 REST API：
   - 使用控制台，
     i. 打开控制台。
     ii. 在导航菜单中点击 **Manage applications**。所有应用程序的 **Overview** 选项卡都会打开。
     iii. 点击 **Resources** 选项卡。
     iv. 将页面向下滚动到 **Resource pipeline** 部分。点击您要更新的频道的 **YAML** 编辑图标。这会打开 **Edit channel** 窗口。
     v. 编辑频道的 YAML。
     vi. 完成后，点击 **Save** 更新频道。

   您还可以使用控制台搜索来查找并编辑频道：

   i. 在导航菜单中点击 **Search**。
   ii. 在搜索框中，按 **kind:channel** 过滤来查看所有频道。
   iii. 在所有频道列表中点击您要更新的频道。此时会显示频道的 YAML。
   iv. 点击 **Edit** 来启用 YAML 内容编辑。
   v. 完成编辑后，点击 **Save**。您的更改会被自动保存并应用。
   - 要使用 Kubernetes CLI 工具，步骤与创建频道的步骤相同。

     > To use REST API, use the link:../apis#apis[APIs].

1.1.6.1.2.1. 删除频道

要删除频道，您可以使用控制台、Kubernetes 命令行界面 (**kubectl**) 工具或 REST API。

要使用控制台，请完成以下步骤：

1. 在导航栏中，点 **Manage application**。
2. 点击 **Resources** 选项卡。
3. 找到您要删除的频道资源卡。
4. 点 **Options** 可以访问更多操作。
5. 点 **Delete channel**

6. 验证频道列表被刷新，删除的频道将不再显示。

要使用 Kubernetes CLI 工具：

1. 运行以下命令以从目标命名空间中删除频道。
2. 将 **name** 和 **namespace** 替换为您的频道和目标命名空间的名称：

```
kubectl delete Channel <name> -n <namespace>
```

3. 运行以下命令验证您的频道是否已删除：

```
kubectl get Channel <name>
```

要使用 REST API，请使用 APIs。

1.1.6.1.3. 使用频道管理部署

您可以使用频道和订阅来管理向受管集群或其他命名空间持续交付可部署资源，如 Helm chart 和 Kubernetes 可部署对象。

当某个频道指向 Helm 仓库时，频道 operator 会创建一个可部署资源来表示仓库中找到的每个 Helm 发行版本。

对于 Kubernetes 可部署对象，要将对象添加到频道，您可以将对象嵌套为可部署资源。

在可部署定义中，您可以直接指定要提升可部署资源的频道。您也可以为可部署资源指定所需的 Kubernetes 标签，以满足让可部署资源自动添加到频道的最低频道要求。当频道控制器检测到可部署资源包含所需的 Kubernetes 标签时，控制器会将可部署资源提升到频道。

创建 **Namespace** 和 **ObjectBucket** 频道时，您可以在频道定义的 **spec.gates** 部分中设置频道最低要求。这些要求是可部署资源在可以提升到频道之前所必须包含的 Kubernetes 注解。例如，您可以指定注解以用于开发批准、测试和质量保证批准，并且用于指示可部署资源已准备好部署到生产环境集群中。这些最低要求可以是任何字段和值，如源命名空间、软件包名称、标签和注解。只有在可部署定义中包含匹配的字段和值时，才能在频道中提升可部署资源。当可部署资源满足定义的要求时，可部署资源会自动提升到频道并部署到订阅该频道的任何受管集群。最低要求不适用于 **HelmRepo** 和 **Git** 频道类型。

对于不包含最低要求的频道，频道控制器会将可部署资源的最新版本提升到频道。

集群可以订阅频道，以便标识要部署到每个集群的可部署资源。提升到某个频道的可部署资源只能供该频道的订阅访问。当存在频道和订阅时，频道和订阅可以一起从频道源中检索可部署资源，并将可部署资源放在目的地。目的地通常是抽象为命名空间的受管集群。受管集群或命名空间可订阅多个频道，以便标识要部署到集群的可部署资源。频道确保了有正确的可部署资源可用于检索。订阅确保了对可部署资源进行检索并将其放置到目标命名空间或集群上。要检索可部署资源，订阅 operator 会检查注解限制，并确定是否检索可部署资源并将其应用到受管集群。

1.1.6.1.4. 频道状态和同步

频道没有状态，而订阅某个频道的订阅却有状态。订阅的状态报告了订阅是否在 hub 集群上成功传播，以及订阅是否成功创建或应用了可部署模板，或创建了 helmRelease CR。使用频道和订阅部署的可部署资源状态与订阅分开报告。

由于频道没有状态，提升到频道命名空间的资源并非始终与实际频道存储同步。

当可部署资源包含到某个频道中或更新后，该频道的订阅 operator 会自动检测到新的或更新的可部署资源。您不需要从 hub 集群中向目标集群发出任何通知。

对于命名空间类型频道的订阅，只有在受管集群可以访问频道命名空间的情况下才会为该集群同步资源。由于频道源存在于 hub 集群上，因此只有存在对 hub 集群的访问权限时，订阅才能从源中拉取资源。

对于 Helm 仓库和对象存储类型频道的订阅，订阅会监控频道源仓库是否有新的或更新的 Helm chart 或可部署资源。订阅不需要与 hub 集群通信，除非源仓库位于 hub 集群上。如果将订阅设置为拉取 Helm 发行版本或可部署对象的最新版本，且仓库类型中包含新版本，订阅就可以检索这些版本。

1.1.6.1.5. 管理 secret

您可以创建 Kubernetes secret 资源来为您的订阅和应用程序组件存储授权凭证和其他敏感信息。

资源（包括 secret）的示例位于应用程序资源示例文档中。

1.1.6.1.5.1. Kubernetes secret

Secret (**Secret**) 属于 Kubernetes 资源，可用于存储授权和其他敏感信息，如密码、OAuth 令牌和 SSH 密钥。

使用 secret 时，您可以安全地隔离和存储信息，同时使用订阅在需要的地方发送信息。

例如，您可以使用镜像 pull secret，其中包括部署资源引用的私有 Docker 镜像 registry 的凭证。

如果将镜像 pull secret 包括在受管集群命名空间中，而后者包含了对 Helm chart 或部署的 Pod 的订阅，那么 Helm chart 或 Pod 可以在命名空间中使用 pull secret。

当您在一个频道中创建并包含用于部署到受管集群的 secret 时，该 secret 只能部署到目标受管集群上与频道命名空间相同的命名空间（订阅在 hub 集群和受管集群中使用相同的命名空间）。

所有 secret 数据都是端到端加密的。在 hub 集群和受管集群上，Kubernetes 会加密它的 secret（at rest）。在传输过程中，使用 TLS 加密。

1. 创建在 hub 集群中存储 *channel* 和 *secret* 资源的命名空间。
   需要的访问权限：集群管理员。您只需运行以下命令进行一次：

   ```
   oc new-project SECRET_NAMESPACE
   ```

   或者，您可以运行 **oc apply -f FILENAME.yaml** 来应用以下 YAML 示例。复制和粘贴：

   ```
   apiVersion: v1
   kind: Namespace
   metadata:
     name: SECRET_NAMESPACE
   ```

2. 创建频道资源secret 命名空间还需要包含频道资源。运行 **oc apply -f FILENAME.yaml** 命令以应用以下示例 YAML 文件：

   ```
   apiVersion: apps.open-cluster-management.io/v1
   kind: Channel
   metadata:
     name: CHANNEL_NAME
     namespace: SECRET_NAMESPACE
   spec:
     type: Namespace
     pathname: SECRET_NAMESPACE
   ```

以下 YAML 定义表包含需要提供的键值：

键	值
name	频道的唯一名称，受 dns 格式的限制。
namespace	可选。如果没有提供，则会在环境当前命名空间中创建频道。必须与您要订阅的 secret 相同。
pathname	应该与命名空间相同。

3. 通过运行 **oc apply -f FILENAME.yaml** 命令来创建 secret，以应用以下示例 YAML 文件。您可以根据需要创建多个 secret:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
     apps.open-cluster-management.io/deployables: "true"
  name: SECRET_NAME
  namespace: SECRET_NAMESPACE
data:
```

以下 YAML 定义表包含需要提供的键值：

键	值
name	唯一名称，受 dns 格式的限制
namespace	可选。如果没有提供，secret 会在环境当前命名空间中创建
data	这是您要安全存储的 YAML 或大型字符串块

4. 创建您的订阅。订阅用于订阅频道中的一个或多个 secret，需要在其自己的命名空间中创建。此命名空间传播至创建 secret 的受管集群。
以下示例定义了本地订阅放置：

```
placement:
  local: true
```

以下示例定义了受管集群（远程）订阅放置：

```
placement:
  placementRef:
    name: PLACEMENT_NAME
    kind: PlacementRule
```

以下示例定义了订阅：

-

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: SUBSCRIPTION_NAME
  namespace: SUBSCRIPTION_NAMESPACE
spec:
  channel: CHANNEL_NAMESPACE/CHANNEL_NAME
  placement:
   local: true
```

5. 要验证您是否应用了频道中的所有 secret，请运行以下命令：

```
oc -n SUBSCRIPTION_NAMESPACE describe appsub SUBSCRIPTION_NAME
```

1.1.6.2. 创建和管理订阅

您可以创建和管理订阅，以标识、检索新的和已更新的资源并部署到受管集群。通过使用订阅，您可以提高应用程序管理的持续交付功能。所有资源（包括订阅）的示例位于应用程序资源示例文档中。

了解更多有关订阅的信息，然后查看以下任务：

- 创建订阅
- 将订阅与应用程序匹配
- 更新订阅
- 调度部署
- 订阅 Git 资源
- 配置软件包覆盖
- 删除订阅

订阅 (**subscription.apps.open-cluster-management.io**) 属于 Kubernetes 资源，用作定义集合，通过使用注解、标签和版本来标识频道中的 Kubernetes 资源（在 Git 中、对象存储或 hub 集群可部署资源）Helm chart。

订阅资源可指向某个频道，以标识用于部署的新和已更新的 Helm chart 或 Kubernetes 资源。然后订阅 operator 会监视该频道是否有新的和已更新的 chart 和可部署资源。

当检测到新的或已更新的 Helm chart 或 Kubernetes 资源时，订阅 operator 会下载适用于指定 Helm chart 版本或指定 Kubernetes 资源的 Helm 发行版本。订阅 operator 可以在不先检查 hub 集群的情况下，直接下载这些对象，或作为可部署资源从存储位置下载到目标受管集群。

订阅可过滤提升到某个频道的可部署资源以选择特定的可部署资源。例如，订阅可以过滤可部署资源来选择一个特定的可部署版本。在本例中，订阅 operator 会检查 **version** 参数，以标识要选择的可部署版本。

1.1.6.2.1. 创建订阅

1. 为您的订阅编写定义 YAML 内容。有关 YAML 结构和示例的更多信息，请参阅应用程序资源示例文档。

2. 在 Red Hat Advanced Cluster Management for Kubernetes 中创建订阅。您可以使用控制台、Kubernetes CLI (**kubectl**) 工具或 REST API：

   - 使用控制台：

     i. 打开控制台。

     ii. 在导航菜单中点击 **Manage applications**。所有应用程序的 **Overview** 选项卡都会打开。

     iii. 点击 **Resources** 选项卡。

     iv. 滚动到 *Resource pipeline* 部分。在资源摘要卡右侧的按钮列表中点击 **Susbcription**。此时会显示 *Create a Susbcription* 编辑器。

     v. 输入 YAML 内容以定义订阅，或直接更新默认 YAML 模板以满足您的要求。

     vi. 完成后，点击 **Save** 创建订阅。您的新订阅会显示在相应应用程序和频道的 **Resource pipeline** 中。

     另外，您可以选择在使用特定应用程序时创建订阅。

     i. 在所有应用程序的 **Overview** 选项卡中点击 **All applications** 列表中的应用程序。该应用程序的 **Overview** 选项卡将打开。

     ii. 点击该应用程序的 **Resources** 选项卡。

     iii. 滚动到 *Resource pipeline* 部分。在资源摘要卡右侧的按钮列表中点击 **Susbcription**。此时会显示 *Create a Susbcription* 编辑器。

     iv. 输入 YAML 内容以定义订阅，或直接更新默认 YAML 模板以满足您的要求。

     v. 完成后，点击 **Save** 创建订阅。您的新订阅会显示在相应应用程序和频道的 **Resource pipeline** 中。

   注意：订阅最初可能不会显示在您的任何应用程序的资源管道中。要让订阅与某个应用程序关联，您需要在订阅定义中包含适当的值，以匹配应用程序定义中所定义的必需值。如需更多信息，请参阅将订阅与应用程序匹配。

   - 要使用 Kubernetes CLI 工具：

     i. 运行以下命令，将您的文件应用到 apiserver。将 **filename** 替换为您的文件名称：

     ```
     kubectl apply -f filename.yaml
     ```

     ii. 运行以下命令验证您的订阅资源是否已创建：

     ```
     kubectl get appsub
     ```

     确保您的新订阅已在生成的输出中列出。

   - 要使用 REST API，请使用 APIs。

创建订阅后，您的订阅可具有以下状态之一：

- **Subscribed**（仅限受管集群）
  订阅已成功订阅指定资源且这些资源已部署。

- **Propagated**（仅限 hub 集群）
  会根据指定的 **spec.placement** 设置生成后续的订阅，以便部署到适当的受管集群（若有）。但可能不会实际部署订阅。

- **Failed** 订阅无法订阅指定资源。

- **No status**
  订阅 operator 不在线，或者缺少订阅的 **spec.placement** 设置。

如果您是为多集群环境创建的订阅，则将在 Hub 集群上创建订阅。根据 **spec.placement** 设置，您的订阅在其他集群上的后续放置位置会有所不同。如果您在 **spec.placement** 部分包含多个放置设置，则订阅 operator 会使用以下优先级选择要使用的放置设置：

- **spec.placement.placementRef**
  订阅放置在由指定 **PlacementRule** 资源标识的集群上。

- **spec.placement.clusters**
  订阅放置在指定为该字段值的集群上。

- **spec.placement.clusterSelector**
  订阅放置在与指定标签选择器匹配的集群上，该选择器被定义为该字段的值。

如果在独立集群或想要直接管理的集群上创建的订阅，则只会在该集群上创建订阅。根据您在订阅定义中为 **spec.placement.local** 字段设置的值，订阅的后续行为会有所不同。

- **true**
  订阅与该集群本地的指定频道同步。

- **false**
  该订阅不会订阅指定频道中的任何资源。

1.1.6.2.2. 将订阅与应用程序匹配

要将订阅与应用程序关联，订阅和应用程序必须位于同一命名空间，以便订阅可从频道检索 Helm chart、可部署资源或其他资源。

在应用程序资源定义中，定义必须包含 **spec.componentKinds** 设置，以指示应用程序使用了订阅。定义还必须包含 **spec.selector** 设置，以定义用于将应用程序与订阅匹配的标签 (**matchLabels**) 或表达式 (**matchExpressions**)。

在订阅资源定义中，定义必须包含所需的值才能与应用程序定义的标签或表达式匹配。

当订阅与某个应用程序关联时，订阅将使用订阅或可部署资源的 **spec.placement** 设置来为应用程序部署任何订阅的 chart、可部署资源或其他 Kubernetes 资源。

有关应用程序的资源定义的更多信息，请参阅创建和管理应用程序资源。

1.1.6.2.3. 更新订阅

1. 为您的订阅编写定义 YAML 内容。

2. 在 Red Hat Advanced Cluster Management for Kubernetes 中创建订阅。您可以使用控制台、Kubernetes CLI (**kubectl**) 工具或 REST API：

   - 使用控制台：

     i. 打开控制台。

ii. 在导航菜单中点击 **Manage applications**。所有应用程序的 **Overview** 选项卡都会打开。

iii. 点击 **Resources** 选项卡。

iv. 将页面向下滚动到 **Resource pipeline** 部分。找到使用您要编辑的订阅的应用程序所在的行并展开。

v. 对于您要更新的订阅，请点击 YAML 的编辑图标。这会打开 **Edit subscription** 窗口。

vi. 编辑 YAML。

vii. 完成后，点击 **Save** 更新订阅。

另外，您可以选择在使用特定应用程序时更新订阅。

i. 在所有应用程序的 **Overview** 选项卡中点击 **All applications** 列表中的应用程序。该应用程序的 **Overview** 选项卡将打开。

ii. 点击该应用程序的 **Resources** 选项卡。

iii. 将页面向下滚动到 **Resource pipeline** 部分。

iv. 对于您要更新的订阅，请点击 YAML 的编辑图标。这会打开 **Edit subscription** 窗口。

v. 编辑 YAML。

vi. 完成后，点击 **Save** 更新订阅。

您还可以使用控制台搜索来查找并编辑订阅：

i. 点击标头中的**搜索**图标。

ii. 在搜索框中，按 **kind:subscription** 过滤来查看所有订阅。

iii. 在所有订阅列表中点击您要更新的订阅。此时会显示订阅的 YAML。

iv. 点击 **Edit** 来启用 YAML 内容编辑。

v. 完成编辑后，点击 **Save**。您的更改会被自动保存并应用。

- 要使用 Kubernetes CLI 工具，步骤同创建订阅。

> To use REST API, use the link:../apis#apis[APIs]

1.1.6.2.4. 调度部署

如果需要只在特定时间部署新的 Helm Chart 或其他资源，或更改 Helm Chart 或其他资源，您可以为这些资源定义订阅，以便只在特定时间才进行部署。另外，您可以限制在一个指定的时间窗期间开始部署，比如避免在商业高峰期内进行部署。

例如，您可以把每周五的晚上 10 点到 11 点期间定义为调度的维护窗口期，只在此期间对集群应用补丁或进行其他应用程序的更新。

另外，您可以限制或阻止在一个指定的时间窗期间开始部署，比如避免在商业高峰期内进行部署。例如，为了避免高峰期，您可以定义一个时间窗以避免在早上 8 点到下午 8 点间进行部署。

通过为订阅定义时间窗，您可以协调所有应用程序和集群的更新。例如，您可以定义订阅以只在 6:01 PM 和 11:59 PM 之间部署新的应用程序资源，并定义其他订阅仅在 12:00 AM 到 7:59 AM 之间部署这些资源的更新版本。

当为订阅定义一个时间窗时，则定义了订阅处于活跃变化的时间范围。作为定义时间窗的一部分，您可以指定在该窗口中订阅是 *active（活跃）* 或 *blocked（阻止）* 状态。只有在订阅处于活跃状态时，才会开始部署新的或更改的资源。无论订阅是活跃还是被阻止，订阅都会继续监控任何新的或被更改的资源。活跃和阻止的设置仅会影响到部署。

当检测到新的或更改的资源时，时间窗定义决定订阅的下一步操作。

- 对于 **HelmRepo**、**ObjectBucket** 和 **Git** 类型的频道的订阅：
- 如果在订阅激活的时间范围内检测到资源，则开始部署资源。
- 如果在订阅无法运行部署时检测到资源，部署资源的请求会被缓存。当订阅在下一个活跃时间时，缓存的请求会被应用并开始相关的部署。
- 对于 **Namespace** 类型的频道的订阅：
- 当订阅处于活跃状态时，订阅会与该频道同步，并开始部署需要部署的所有资源的最新版本。
- 当订阅被阻止时，订阅不会与频道同步来部署资源。

如果部署在指定的时间窗内开始，并在定义的时间窗结束时仍在运行，部署将继续运行以完成。

要为订阅定义时间窗，您需要在订阅资源定义 YAML 中添加所需的字段和值。

- 作为定义时间窗的一部分，您可以使用天和小时定义时间窗。
- 您还可以定义时间窗类型，这决定了部署可以开始的时间窗是在指定的时间段内还是在指定的时间段外。
- 如果时间窗类型是 **active**，则部署只能在指定的时间范围内开始。当希望部署只在特定的维护窗口中进行时，您可以使用此设置。
- 如果时间窗类型是 **block**，则部署不能在指定的时间范围内启动，但可以在任何其他时间开始。当您有需要的关键更新，同时需要避免在指定时间范围内进行部署，则可以使用这个设置。例如，您可以使用这个设置类型来定义一个时间窗，允许在 10 AM 到 2:00 PM 这个时间段外随时应用与安全相关的更新。
- 您可以为订阅定义多个时间窗，如在每周一和周三定义时间窗。

1.1.6.2.4.1. 订阅 Git 资源

您可以将 Git 资源订阅到多个命名空间。默认情况下，当您订阅到某个应用程序时，应用程序会部署到那个订阅命名空间中。要将这些资源应用到那个订阅命名空间以外的命名空间，请执行以下步骤：

需要的访问权限：集群管理员

1. 从控制台登录到 hub 集群。
2. 创建一个或多个用户。
   这些用户代表了 **app.open-cluster-management.io/subscription** 应用程序的管理员。在 OpenShift Container Platform 中，您可以将这些用户分为一个组来代表具有订阅管理权限的组。本文后面会介绍它。

3. 从终端再次登陆到 Red Hat Advanced Cluster Management hub 集群。

4. 使用以下命令在 **open-cluster-management:subscription-admin** ClusterRoleBinding 中添加以下 subjects 内容：

```
oc edit clusterrolebinding open-cluster-management:subscription-admin
```

请**注意： open-cluster-management:subscription-admin** ClusterRoleBinding 开始并没有 subject 的内容。

您的 subject 可能如下所示：

```
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: example-name
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: example-group-name
```

要进行验证，请参阅以下示例。在这个示例中，作为订阅管理员登录。在示例中，您可以从 Git 存储库中订阅示例资源 YAML 文件，按照前面的步骤进行，然后查看更新的 Configmap 设置。示例文件包含位于以下不同命名空间中的订阅：

- Configmap **test-configmap-1** 在 **multins** 命名空间中创建。
- ConfigMap **test-configmap-2** 在 **default** 命名空间中创建。
- ConfigMap **test-configmap-3** 可在 **subscription** 命名空间中创建。

```
---
apiVersion: v1
kind: Namespace
metadata:
  name: multins
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap-1
  namespace: multins
data:
  path: resource1
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap-2
  namespace: default
data:
  path: resource2
---
apiVersion: v1
kind: ConfigMap
metadata:
```

```
  name: test-configmap-3
data:
  path: resource3
```

1.1.6.2.5. 配置软件包覆盖

为订阅中所订阅的 Helm chart 或 Kubernetes 资源的订阅覆盖值配置软件包覆盖。

要配置软件包覆盖，请将 Kubernetes 资源 spec 中要覆盖的字段指定为 **path** 字段的值。将替换值指定为 **value** 字段的值。

例如，如果您需要在订阅的 Helm chart 的 Helm 发行版本的 spec 中覆盖 values 字段，则需要将订阅定义中的 **path** 字段的值设置为 **spec**。

```
packageOverrides:
- packageName: nginx-ingress
  packageOverrides:
  - path: spec
    value: my-override-values
```

value 字段的内容用于覆盖 **HelmRelease** spec 的 **spec** 字段中的值。

- 在 Helm 发行版本中，**spec** 字段的覆盖值合并到发行版本 **values.yaml** 文件中，以覆盖现有的值。此文件用于检索 Helm 发行版本的可配置变量。
- 如果您需要覆盖 Helm 发行版本的发行版本名称，请在定义中包含 **packageOverride** 部分。通过包含以下字段为 Helm 发行版本定义 **packageAlias**:
  - 用于标识 Helm chart 的 **packageName**。
  - 用于表示您将覆盖发行版本名称的 **packageAlias**。

  默认情况下，如果没有指定 Helm 发行版本名称，则使用 Helm chart 名称来标识该发行版本。在某些情况下，比如有多个发行版本订阅了同一 chart 时，可能会发生冲突。发行版本名称在命名空间中的不同订阅之间必须是唯一的。如果您创建的订阅的发行版本名称不是唯一的，则会出现错误。您必须通过定义 **packageOverride** 为您的订阅设置不同的发行版本名称。如果要在现有订阅中更改名称，必须首先删除该订阅，然后使用首选发行版本名称重新创建订阅。

  +

  ```
  packageOverrides:
  - packageName: nginx-ingress
    packageAlias: my-helm-release-name
  ```

1.1.6.2.6. 删除订阅

要删除订阅，您可以使用控制台、Kubernetes 命令行界面 (**kubectl**) 工具或 REST API。

要使用控制台，请完成以下步骤：

1. 在导航栏中，点 **Manage application**。
2. 点击 **Resources** 选项卡。
3. 找到您要删除的订阅资源卡。

4. 点 **Options** 可以访问更多操作。
5. 点 **Delete subscription**
6. 验证在刷新所有订阅列表时，确认您删除的订阅将不再显示。

要使用 Kubernetes CLI 工具：

1. 运行以下命令以从目标命名空间中删除订阅。
2. 将 **name** 和 **namespace** 替换为您的订阅和目标命名空间的名称：

```
kubectl delete appsub <name> -n <namespace>
```

3. 运行以下命令验证您的订阅是否已删除：

```
kubectl get appsub <name>
```

要使用 REST API，请使用 APIs。

1.1.6.3. 创建和管理放置规则

您可以创建和管理放置规则，以定义 Helm chart 和可部署资源的部署位置和方式。放置规则可帮助您促进可部署资源的多集群部署。所有资源（包括放置规则）的示例位于应用程序资源示例文档中。

- 创建放置规则
- 分配放置规则
- 查看放置状态
- 更新放置规则
- 删除放置规则

放置规则的自定义资源定义 (CRD) 和控制器替代了用于之前 Red Hat Advanced Cluster Management for Kubernetes 版本中的应用程序的放置策略。放置策略仍用于监管和风险策略。

可以为订阅和可部署资源定义放置规则。在订阅级别为多集群部署定义放置规则。定义放置规则以用于单集群部署的特定可部署资源或用于覆盖放置设置。

1.1.6.3.1. 创建放置规则

可以为订阅和可部署资源定义放置规则。在订阅级别为多集群部署定义放置规则。定义放置规则以用于单集群部署的特定可部署资源或用于覆盖放置设置。

先决条件： 确保 **klusterlet-addon-appmgr** pod 正在运行。您可以运行 **oc get pods -n open-cluster-management-agent-addon** 来验证 pod。

1. 编写放置规则的定义 YAML 内容。所有资源（包括放置规则）的示例位于应用程序资源示例文档中。
2. 在 Red Hat Advanced Cluster Management for Kubernetes 中创建放置规则。您可以将放置规则定义为单独的资源，或者在可部署资源或订阅的定义中定义规则。

作为最佳实践，当规则可能需要被多个资源引用时，将放置规则定义为单独的资源。

+ 要将放置规则作为单独的资源来创建，您可以使用控制台、Kubernetes CLI (**kubectl**) 工具或 REST API：

- 使用控制台：
  i. 打开控制台。
  ii. 在导航菜单中点击 **Manage applications**。所有应用程序的 **Overview** 选项卡都会打开。
  iii. 点击 **Resources** 选项卡。
  iv. 滚动到 *Resource pipeline* 部分。在按钮列表中找到并点击 **Placement Rule**。此时会显示 *Create a placement rule* 编辑器。
  v. 输入 YAML 内容以定义放置规则，或直接更新默认 YAML 模板以满足您的要求。
  vi. 添加或编辑完 YAML 后，点击 **Save** 创建放置规则。
  vii. 在资源概述卡列表中点击 *PLACEMENT RULES* 卡。此时会显示 *Search* 仪表板，并列出 *Applications* 仪表板上选择的应用程序使用的所有放置规则。在规则被应用程序使用的订阅引用前，您的新放置规则不会显示在此列表中。要验证您的新放置规则是否已创建，请在搜索框中输入 **kind:placementrule** 并运行搜索。确保您的新规则已显示在搜索结果列表中。
- 要使用 Kubernetes CLI 工具：
  i. 运行以下命令，将您的文件应用到 apiserver。将 **filename** 替换为您的文件名称：

  ```
  kubectl apply -f filename.yaml
  ```

  ii. 运行以下命令验证您的放置规则是否已创建：

  ```
  kubectl get PlacementRule
  ```

  确保您的新放置规则已在生成的输出中列出。
- 要使用 REST API，请使用 APIs。

1.1.6.3.2. 分配放置规则

您可以为可部署资源或订阅分配放置规则。要分配放置规则，您需要更新可部署资源或订阅的 spec 来引用放置规则。

在您的可部署资源或订阅 spec 中包含以下 **placement** 字段以及相应的值以引用放置规则：

```
placement:
  placementRef:
    name:
    kind: PlacementRule
```

包含放置规则的名称作为 **name** 字段的值。

当为订阅分配了放置规则时，您可以使用以下方法在控制台中的 Applications 仪表板上查看分配情况。

1. 打开控制台。
2. 在导航菜单中点击 **Manage applications**。所有应用程序的 **Overview** 选项卡都会打开。

3. 点击 **Resources** 选项卡。

4. 滚动到 **Resource pipeline** 部分。在列出应用程序的表中，展开应用程序所在的行，其中包括被分配了放置规则的订阅。

5. 在展开的应用程序视图中，您可以看到每个频道的可用订阅。每个订阅的详情包括任何已分配的放置规则。如果需要，您可以选择从该资源管道表中查看或编辑放置规则、订阅和频道的 YAML。

1.1.6.3.3. 查看放置规则状态

当放置规则已创建且正在使用时，您可以查看该规则的状态详情。此状态附加到放置规则的 YAML 定义中，并指示使用该规则来放置可部署资源的目标集群。

要查看放置规则的状态字段，您可以使用控制台、Kubernetes 命令行界面 (**kubectl**) 工具或 REST API。

- 使用控制台，

  a. 打开控制台。

  b. 点击标头中的**搜索**图标。

  c. 在搜索框中，按 **kind:placementrule** 过滤来查看所有放置规则。

  d. 在所有放置规则列表中，点击您要查看的放置规则。此时会显示该规则的 YAML。

  e. 查看 YAML 内容的 **status** 部分中的字段和值。

- 要使用 Kubernetes CLI 工具，请运行以下命令。将 **name** 和 **namespace** 替换为放置规则和目标命名空间的名称：

  a. 运行以下命令

  ```
  kubectl get PlacementRule <name> -n <namespace>
  ```

  b. 查看 YAML 内容的 **status** 部分中的字段和值。

要使用 REST API，请使用 APIs

1.1.6.3.4. 更新放置规则

要更新作为单独资源的放置规则，您可以使用控制台、Kubernetes 命令行界面 (**kubectl**) 工具或 REST API。

- 要使用控制台编辑放置规则，请完成以下步骤：

  a. 打开控制台。

  b. 点击标头中的**搜索**图标。

  c. 在搜索框中，按 **kind:placementrule** 过滤来查看所有放置规则。

  d. 在所有放置规则列表中，点击您要更新的放置规则。此时会显示该规则的 YAML。

  e. 点击 **Edit** 来启用 YAML 内容编辑。

  f. 完成编辑后，点击 **Save**。您的更改会被自动保存并应用。

或者，您可以选择从 Applications 仪表板资源管道表中编辑 YAML。

a. 在导航菜单中点击 **Manage applications**。所有应用程序的 **Overview** 选项卡都会打开。

b. 点击 **Resources** 选项卡。

c. 滚动到 **Resource pipeline** 部分。在列出应用程序的表中，展开应用程序所在的行，其中包括被分配了放置规则的订阅。

d. 在展开的应用程序视图中，您可以看到每个频道的可用订阅。每个订阅的详情包括任何已分配的放置规则。点击放置规则的链接打开 *Edit placement rule* 编辑器。此时会显示该规则的 YAML。

e. 完成编辑后，点击 **Save**。您的更改会被自动保存并应用。

- 要使用 Kubernetes CLI 工具，步骤与创建放置规则的步骤相同。

要使用 REST API，请使用 APIs。

要更新在可部署资源或订阅的定义中定义的放置规则，步骤与更新相应资源的步骤相同。

1.1.6.3.5. 删除放置规则

要删除作为单独资源的放置规则，您可以使用控制台、Kubernetes 命令行界面 (**kubectl**) 工具或 REST API。

要使用控制台，请完成以下步骤：

1. 在导航栏中，点 **Manage application**。
2. 点击 **Resources** 选项卡。
3. 查找您要删除的放置规则的订阅资源卡。
4. 点 **Options** 可以访问更多操作。
5. 点 **Delete placement rule**
6. 验证在刷新所有订阅列表时，确认您删除的放置规则不再显示。

要使用 Kubernetes CLI 工具，请完成以下步骤：

1. 运行以下命令以从目标命名空间中删除放置规则。
2. 将 **name** 和 **namespace** 替换为您的放置规则和目标命名空间的名称：

```
kubectl delete PlacementRule <name> -n <namespace>
```

3. 运行以下命令验证您的放置规则资源是否已删除：

```
kubectl get PlacementRule <name>
```

要使用 REST API，请使用 APIs。

1.1.6.4. 创建和管理应用程序资源

创建应用程序资源来查看组成您的整个 Red Hat Advanced Cluster Management for Kubernetes 多集群应用程序的应用程序组件并进行分组。所有资源的示例位于应用程序资源示例文档中。

- 创建应用程序
- 将订阅与应用程序匹配
- 更新应用程序
- 删除应用程序

1.1.6.4.1. 创建应用程序

1. 编写应用程序定义 YAML 内容。要创建或更新应用程序资源，您必须首先编写定义该资源的 YAML 文件。
2. 在 Red Hat Advanced Cluster Management for Kubernetes 中创建应用程序。您可以使用控制台、Kubernetes 命令行界面 (**kubectl**) 工具或 REST API：
   - 使用控制台，
     i. 打开控制台。
     ii. 在导航菜单中点击 **Manage applications**。所有应用程序的 **Overview** 选项卡都会打开。
     iii. 点击 **New application**。这会打开 *Create application* 窗口。
     iv. 更新或替换编辑器中的默认 YAML 内容，以定义您的应用程序。
     v. 添加和编辑完 YAML 后，点击 **Save** 创建应用程序。您的新应用程序会显示在 **Overview** 选项卡上的应用程序列表中。
   - 使用 Kubernetes CLI 工具，
     i. 使用您首选的编辑工具编写并保存您的应用程序 YAML 文件。
     ii. 运行以下命令，将您的文件应用到 apiserver。将 **filename** 替换为您的文件名称：

     ```
     kubectl apply -f filename.yaml
     ```

     iii. 运行以下命令验证您的应用程序资源是否已创建：

     ```
     kubectl get Application
     ```

     确保您的新应用程序已在生成的输出中列出。
   - 要使用 REST API，请使用 APIs。

1.1.6.4.2. 更新应用程序

1. 编写应用程序定义更新。所有资源的示例位于应用程序资源示例文档中。
2. 更新应用程序定义。您可以使用控制台、Kubernetes 命令行界面 (**kubectl**) 工具或 REST API：
   - 要使用控制台，请完成以下步骤：
     i. 打开控制台。

ii. 在导航菜单中点击 **Manage applications**。所有应用程序的 **Overview** 选项卡都会打开。

iii. 在所有应用程序列表中，找到您要更新的应用程序所在的行。对于该行，展开 *Options* 菜单并点击 **Edit application**。这会打开 *Edit application* 窗口。

iv. 更新应用程序的 YAML 内容。

v. 完成编辑后，点击 **Close editor**。您的更改会被自动保存并应用。

您还可以使用控制台搜索来查找并编辑应用程序：

i. 点击标头中的*搜索*图标打开*搜索*页面。

ii. 在搜索框中，按 **kind:application** 过滤来查看所有应用程序。

iii. 在所有应用程序列表中点击您要更新的应用程序。该应用程序的 *Overview* 页面将打开。

iv. 点击 **Edit app**。这会打开 *Edit application* 窗口。

v. 更新应用程序的 YAML 内容。

vi. 完成编辑后，点击 **Close editor**。您的更改会被自动保存并应用。

- 要使用 Kubernetes CLI 工具，步骤与创建应用程序的步骤相同。

要使用 REST API，请使用 APIs。

1.1.6.4.3. 删除应用程序

要删除应用程序，您可以使用控制台、Kubernetes 命令行界面 (**kubectl**) 工具或 REST API：

要使用控制台，请完成以下步骤：

1. 在导航菜单中点击 **Manage applications**。所有应用程序的 **Overview** 选项卡都会打开。
2. 在所有应用程序列表中，找到您要删除的应用程序所在的行。
3. 在该行中展开 *Options* 菜单并点击 **Delete application**。这会打开确认窗口。
4. 确认移除以删除应用程序。
5. 当刷新所有应用程序列表时，将不再包含该应用程序。

要使用 Kubernetes CLI 工具，请完成以下步骤：

1. 运行以下命令以从目标命名空间中删除应用程序。
2. 将 **name** 和 **namespace** 替换为您的应用程序和目标命名空间的名称：

```
kubectl delete Application <name> -n <namespace>
```

3. 运行以下命令验证您的应用程序资源是否已删除：

```
kubectl get Application <name>
```

要使用 REST API，请使用 APIs。

1.1.6.4.4. 使用应用程序资源进行部署

要为部署设置和使用频道、订阅和放置规则，请完成以下步骤：

1. 如果包括对象存储、Kubernetes 命名空间或 Helm 仓库的频道不存在，则创建一个频道。在将可部署资源提升到频道前，确保定义了可部署资源需要的任何标签或注解。如需更多信息，请参阅创建和管理频道。

2. 如果您的一个或多个目标集群没有订阅该频道，请创建订阅。如需更多信息，请参阅创建和管理订阅。

3. 定义要从频道部署的可部署资源的放置规则。如需更多信息，请参阅创建和管理放置规则。

可为订阅和可部署资源定义放置规则。**要将**规则应**用于多个可部署**资**源和集群，**请为订阅**定**义**放置**规则**。**要防止规则应用到订阅的所有可部署资源，或者要覆盖订阅级别的放置规则，请为可部署资源定义放置规则。** 如果要引用放置规则，您还可以为可部署资源包含覆盖设置，以使用特定于可部署资源的值覆盖某些放置规则值。可选。如果以独立资源的形式创建放置规则，请编辑订阅或可部署资源的定义来引用您的放置规则。编辑您的可部署资源和频道的定义，以确保将可部署资源提升到该频道。如需更多信息，请参阅*将可部署资源提升到频道*部分。使用控制台监控频道部署到一个或多个目标集群的状态。

1.1.6.4.5. 将可部署资源提升到频道

要将可部署资源（一个*资源模板*）提升到某个频道，您可以使用以下任一方法：

- 通过为可部署资源定义中的 **spec.channels** 字段配置正确的注解来标识频道，将可部署资源指向特定频道。
  spec.channels 参数可用于可部署 (**deployable.apps.open-cluster-management.io**) 资源，以便标识要提升可部署资源的频道。以下示例显示了一个可部署资源，它定义了要包含该可部署资源的频道：

```
apiVersion: apps.open-cluster-management.io/v1
kind: Deployable
metadata:
  name: deployable1
  namespace: default
  spec:
    template:
    placement:
    overrides:
    dependencies:
    channels:
    - mychannel
    - otherchannel
```

- 更新频道定义，以标识要包含在频道中的可部署资源。使用 **spec.package** 和 **spec.packageFilter** 字段来指定可部署资源。
  例如，当向频道的源 Helm 仓库发布新 chart 时，以下频道会自动拉取最近或最新的 **nginx** chart。Chart 可部署资源必须具有一个匹配的版本 **1.x** 才能被提升到频道。

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: predev-ch
  namespace: ns-ch
  labels:
```

```
    app: nginx-app-details
spec:
  type: HelmRepo
  pathname: https://kubernetes-charts.storage.googleapis.com/
---
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: nginx
  namespace: ns-sub-1
  labels:
    app: nginx-app-details
spec:
  channel: ns-ch/predev-ch
  name: nginx-ingress
  packageFilter:
    version: "1.36.x"
  placement:
    placementRef:
      kind: PlacementRule
      name: towhichcluster
```

- 更新订阅定义，以标识可部署资源。在订阅定义中还可指定将可部署资源提升到频道的配置。

以下订阅示例表示最近的 **nginx** 版本 **1.x** chart 将通过频道来提升以使用该订阅进行部署。

```
```yaml
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
   name: mydevsub
   namespace: myspace
spec:
 source: https://kubernetes-charts.storage.googleapis.com/
 package: nginx
 packageFilter:
   version: 1.x
 placement:
   clusters:
   - name: mydevcluster1
```
```

- 更新频道定义，以指定频道最低要求，并更新可部署资源的定义，使其包含与最低要求匹配的字段和值。
  在频道定义的 **spec.gate** 部分中定义频道最低要求。如果可部署资源的字段与频道的 **spec.gate** 值匹配，则会将可部署资源提升到该频道。在这种情况下，可部署文件不需要通过 **spec.channels** 字段指向特定频道。

  在上例中，**packageFilter.version: "1.36.x"** 表示特定的 **nginx** 版本 **1.36.x** chart 将通过频道来提升以使用该订阅进行部署。

1.1.6.4.5.1. 以百分比推出部署的方式进行部署

如果要将部署推出到您自己的目标受管集群，而不是部署到所有目标集群，可以配置为一次仅将可部署资源或 chart 部署到某个百分比的受管集群。

例如，您可能想在需要部署更新时推出部署，但您不想一次影响所有集群。当在一个集群上部署成功时，会向另一个集群推出部署。

1.1.6.5. 应用程序资源示例

在 Red Hat Advanced Cluster Management for Kubernetes 中，应用程序由多个应用程序资源组成。Red Hat Advanced Cluster Management for Kubernetes 应用程序的基本资源是 **application** 资源和 **deployable** 资源。

另外，您可以使用频道、订阅和放置规则资源来帮助部署、更新和管理整个应用程序。

单集群和多集群应用程序使用相同的 Kubernetes 规格，但多集群应用程序涉及更多的部署和应用程序管理生命周期自动化。

Red Hat Advanced Cluster Management for Kubernetes 应用程序的所有应用程序组件资源都在 YAML 文件 spec 部分中定义。当需要创建或更新应用程序组件资源时，需要创建或编辑正确的 spec 部分，使其包含用于定义资源的标签。

查看以下应用程序资源示例：

- 频道示例
- 订阅示例
- 放置规则示例
- 应用程序示例

1.1.6.5.1. 频道示例

查看可以用来构建文件的示例和 YAML 定义。频道 (**channel.apps.open-cluster-management.io**) 为您提供改进的持续集成以及持续交付功能，以便创建和管理 Red Hat Advanced Cluster Management for Kubernetes 应用程序。要了解更多信息，请参阅创建和管理频道。

1.1.6.5.1.1. 频道 YAML 结构

以下 YAML 结构显示频道的必需字段，以及一些常见可选字段。您的 YAML 结构需要包含一些必需字段和值。根据应用程序管理要求，您可能需要包含其他可选字段和值。您可以使用任何工具编写您自己的 YAML 内容。

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name:
  namespace: # Each channel needs a unique namespace, except Git channel.
spec:
  sourceNamespaces:
  type:
  pathname:
  secretRef:
    name:
  gates:
    annotations:
  labels:
```

1.1.6.5.1.2. 频道 YAML 表

字段	描述
apiVersion	必需。将值设为 **apps.open-cluster-management.io/v1**。
kind	必需。将值设为 **Channel** 以表示资源是频道。
metadata.name	必需。频道的名称。
metadata.namespace	必需。频道的命名空间；除 Git 频道外，每个频道都需要一个唯一的命名空间。
spec.sourceNamespaces	可选。标识频道控制器监控的命名空间，看是否有新的或已更新的可部署资源以检索并提升到频道。对于 **Namespace** 频道，命名空间必须位于 hub 集群上。
spec.type	必需。频道类型。支持的类型有：**Namespace**、**HelmRepo**、**Git** 和 **ObjectBucket**
spec.pathname	对于 **HelmRepo**、**Git**、**ObjectBucket** 和 **Namespace** 频道是必需的。对于 **HelmRepo** 频道，将值设置为 Helm 仓库的 URL。对于 **ObjectBucket** 频道，将值设置为对象存储的 URL。对于 **Git** 频道，将值设置为 Git 仓库的 HTTPS URL。对于 **Namespace** 频道，将该值设置为包含该频道的命名空间。
spec.secretRef.name	可选。标识用于身份验证的 Kubernetes Secret 资源，如用于访问仓库或 chart。您只能对 **HelmRepo**、**ObjectBucket** 和 **Git** 类型的频道使用 secret 进行身份验证。
spec.gates	可选。定义在频道中提升可部署资源的要求。如果没有设置任何要求，则会将添加到频道命名空间或源的任何可部署资源提升到频道。**gates** 不适用于 **HelmRepo** 和 **Git** 频道类型，仅适用于 **Namespace** 和 **ObjectBucket** 频道类型。
spec.gates.annotations	可选。频道注解。可部署资源必须具有匹配的注解才能包含在频道中。
spec.labels	可选。频道的标签。

频道的定义结构类似以下 YAML 内容：

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
```

```
metadata:
  name: predev-ch
  namespace: ns-ch
  labels:
    app: nginx-app-details
spec:
  type: HelmRepo
  pathname: https://kubernetes-charts.storage.googleapis.com/
```

1.1.6.5.1.3. Kubernetes 命名空间 (Namespace) 频道

您需要在频道命名空间中手动创建可部署的资源。

要正确创建可部署资源，将可部署资源所需的以下两个标签添加到订阅控制器中用于标识需要添加哪些可部署资源：

```
labels:
    apps.open-cluster-management.io/channel: <channel name>
    apps.open-cluster-management.io/channel-type: Namespace
```

不要在每个可部署 **spec.template.metadata.namespace** 中指定模板命名空间。

对于命名空间类型频道和订阅，所有可部署的模板都部署到受管集群的订阅命名空间中。因此，会跳过在订阅命名空间之外定义的可部署模板。

以下示例频道定义将命名空间抽象为包含可部署资源的频道。当应用此 YAML 时，会为名为 **qa** 的频道创建一个命名空间 **ch-qa**。创建后，此频道会指向源默认命名空间来标识可部署资源。频道控制器在实际命名空间位置维护资源，并确保资源保持最新状态。

```
  apiVersion: apps.open-cluster-management.io/v1
  kind: Channel
  metadata:
    name: qa
    namespace: ch-qa
  spec:
    sourceNamespaces:
    - default
    type: Namespace
    pathname: ch-qa
    gates:
      annotations:
        dev-ready: approved
```

```
apiVersion: apps.open-cluster-management.io/v1
kind: Deployable
metadata:
  labels:
    app: gbchn
    apps.open-cluster-management.io/channel: gbchn
    apps.open-cluster-management.io/channel-type: Namespace
    release: gbchn
  name: gbchn-service
  namespace: gbchn
spec:
  template:
```

```
    apiVersion: v1
    kind: Service
    metadata:
      labels:
        app: gbchn
        release: gbchn
      name: gbchn
    spec:
      ports:
      - port: 80
      selector:
        app: gbchn
```

1.1.6.5.1.4. 对象存储存储桶 (ObjectBucket) 频道

以下示例频道定义将对象存储存储桶抽象为频道：

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
 name: dev
 namespace: ch-obj
spec:
 type: ObjectBucket
 pathname: [http://9.28.236.243:31311/dev] # URL is appended with the valid bucket name, which
matches the channel name.
 secretRef:
   name: miniosecret
 gates:
   annotations:
     dev-ready: true
```

1.1.6.5.1.5. Helm 仓库 (HelmRepo) 频道

以下示例频道定义将 Helm 仓库抽象为频道：

```
apiVersion: v1
kind: Namespace
metadata:
  name: hub-repo
---
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: helm
  namespace: hub-repo
spec:
    pathname: [https://9.21.107.150:8443/helm-repo/charts] # URL points to a valid chart URL.
    configRef:
      name: insecure-skip-verify
    type: HelmRepo
---
apiVersion: v1
data:
```

```
  insecureSkipVerify: "true"
kind: ConfigMap
metadata:
  name: insecure-skip-verify
  namespace: hub-repo
```

以下频道定义显示了 Helm 仓库频道的另一个示例：

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: predev-ch
  namespace: ns-ch
  labels:
    app: nginx-app-details
spec:
  type: HelmRepo
  pathname: https://kubernetes-charts.storage.googleapis.com/
```

1.1.6.5.1.6. Git（Git）仓库频道

以下示例频道定义显示了 Git 仓库的频道示例。在以下示例中，**secretRef** 是指用于访问 **pathname** 中指定的 Git 仓库的用户身份。如果您有一个公共仓库库，则不需要 **secretRef**：

```
apiVersion: apps.open-cluster-management.io/v1
kind: Channel
metadata:
  name: hive-cluster-gitrepo
  namespace: gitops-cluster-lifecycle
spec:
  type: Git
  pathname: https://github.com/open-cluster-management/gitops-clusters.git
  secretRef:
    name: github-gitops-clusters
---
apiVersion: v1
kind: Secret
metadata:
  name: github-gitops-clusters
  namespace: gitops-cluster-lifecycle
data:
  user: dXNlcgo=            # Value of user and accessToken is Base 64 coded.
  accessToken: cGFzc3dvcmQ
```

1.1.6.5.1.7. Secret 示例

Secret (**Secret**) 属于 Kubernetes 资源，可用于存储授权和其他敏感信息，如密码、OAuth 令牌和 SSH 密钥。通过将这些信息存储为 secret，您可以将信息与需要相应信息的应用程序组件分开以提高数据安全性。有关 secret 的更多信息，请参阅管理 secret。

Secret 的定义结构类似以下 YAML 内容：

1.1.6.5.1.7.1. Secret YAML 结构

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
     apps.open-cluster-management.io/deployables: "true"
  name: [secret-name]
  namespace: [channel-namespace]
data:
  AccessKeyID: [ABCdeF1=] #Base64 encoded
  SecretAccessKey: [gHIjk2lmnoPQRST3uvw==] #Base64 encoded
```

1.1.6.5.2. 订阅**示例**

查看可以用来构建文件的示例和 YAML 定义。和频道一样，订阅 (**subscription.apps.open-cluster-management.io**) 为您提供改进的持续集成和持续交付功能以用于应用程序管理。要了解更多信息，请参阅创建和管理订阅。

1.1.6.5.2.1. 订阅 YAML 结构

以下 YAML 结构显示订阅的必需字段，以及一些常见可选字段。您的 YAML 结构需要包含某些必需字段和值。根据应用程序管理要求，您可能需要包含其他可选字段和值。您可以使用任何工具编写您自己的 YAML 内容：

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name:
  namespace:
  labels:
spec:
  sourceNamespace:
  source:
  channel:
  name:
  packageFilter:
    version:
    labelSelector:
      matchLabels:
        package:
        component:
    annotations:
  packageOverrides:
  - packageName:
    packageAlias:
    - path:
      value:
  placement:
    local:
    clusters:
      name:
    clusterSelector:
    placementRef:
      name:
      kind: PlacementRule
  overrides:
```

```
clusterName:
clusterOverrides:
  path:
  value:
```

1.1.6.5.2.2. 订阅 YAML 表

字段	描述	
apiVersion	必需。将值设为 **apps.open-cluster-management.io/v1**。	
kind	必需。将值设为 **Subscription** 以表示资源是订阅。	
metadata.name	必需。用于标识订阅的名称。	
metadata.namespace	必需。用于订阅的命名空间资源。	
metadata.labels	可选。订阅的标签。	
spec.channel	可选。为订阅定义频道的命名空间名称 ("Namespace/Name")。定义 **channel**、**source** 或 **sourceNamespace** 字段。通常，使用 **channel** 字段来指向频道，而不要使用 **source** 或 **sourceNamespace** 字段。如果定义了多个字段，则会使用定义的第一个字段。	
spec.sourceNamespace	可选。将可部署资源存储在 Hub 集群上的源命名空间。仅将该字段用于命名空间频道。定义 **channel**、**source** 或 **sourceNamespace** 字段。通常，使用 **channel** 字段来指向频道，而不要使用 **source** 或 **sourceNamespace** 字段。	
spec.source	可选。存储可部署资源的 Helm 仓库的路径名称 ("URL")。仅将该字段用于 Helm 仓库频道。定义 **channel**、**source** 或 **sourceNamespace** 字段。通常，使用 **channel** 字段来指向频道，而不要使用 **source** 或 **sourceNamespace** 字段。	

字段	描述	
spec.name	对于 **HelmRepo** 类型的频道是必需的，但对于 **Namespace** 和 **ObjectBucket** 类型的频道是可选的。目标 Helm chart 或可部署资源在频道中的具体名称。对于该字段为可选的频道类型，如果没有定义 **name** 或 **packageFilter**，则会找到所有可部署资源，并检索每个可部署资源的最新版本。	
spec.packageFilter	可选。定义用来查找目标可部署资源或一个可部署资源子集的参数。如果定义了多个过滤器条件，则可部署资源必须满足所有过滤器条件。	
spec.packageFilter.version	可选。可部署资源的一个或多个版本。您可以使用 **>1.0** 或 **<3.0** 的形式来指定版本范围。默认情况下会使用带有最新 "creationTimestamp" 值的版本。	
spec.packageFilter.annotations	可选。可部署资源的注解。	
spec.packageOverrides	可选。用于为订阅中所订阅的 Kubernetes 资源（如 Helm chart、可部署资源或频道中的其他 Kubernetes 资源）定义覆盖的部分。	
spec.packageOverrides.packageName	可选，但在设置覆盖时是必需的。标识将被覆盖的 Kubernetes 资源。	
spec.packageOverrides.packageAlias	可选。为将被覆盖的 Kubernetes 资源指定别名。	
spec.packageOverrides.packageOverrides	可选。用于覆盖 Kubernetes 资源的参数和替换值配置。如需更多信息，请参阅配置软件包覆盖。	
spec.placement	必需。标识需要放置可部署资源的订阅集群，或标识定义集群的放置规则。使用放置配置为多集群部署定义值。	

字段	描述	
spec.local	可选，但对于独立集群或者您想要直接管理的集群是必需的。定义是否必须在本地部署订阅。将值设为 **true** 可将订阅与指定频道进行同步。将值设为 **false** 可防止在订阅中订阅指定频道中的任何资源。当集群属于独立集群或您将要直接管理此集群时，请使用此字段。如果您的集群是多集群的一部分，并且不想直接管理集群，则只使用 **clusters**、**clusterSelector** 或 **placementRef** 中的一个来定义您的订阅要放置的位置。如果您的集群是多集群的 Hub，而您想要直接管理集群，则在订阅 operator 可以在本地订阅资源前，必需将 Hub 注册为受管集群。	
spec.placement.clusters	可选。定义要放置订阅的集群。仅使用 **clusters**、**clusterSelector** 或 **placementRef** 中的一个来为多集群定义要在哪里放置订阅。如果集群是一个不属于 Hub 集群的独立集群，则也可以使用 **local**。	
spec.placement.clusters.name	可选，但在定义订阅集群时是必需的。订阅集群的一个或多个名称。	
spec.placement.clusterSelector	可选。定义标签选择器，用于标识要放置订阅的集群。仅使用 **clusters**、**clusterSelector** 或 **placementRef** 中的一个来为多集群定义要在哪里放置订阅。如果集群是一个不属于 Hub 集群的独立集群，则也可以使用 **local**。	
spec.placement.placementRef	可选。定义用于订阅的放置规则。仅使用 **clusters**、**clusterSelector** 或 **placementRef** 中的一个来为多集群定义要在哪里放置订阅。如果集群是一个不属于 Hub 集群的独立集群，则也可以使用 **local**。	
spec.placement.placementRef.name	可选，但在使用放置规则时是必需的。订阅的放置规则名称。	

字段	描述	
spec.placement.placementRef.kind	可选，但在使用放置规则时是必需的。将值设为 **PlacementRule** 以表示将使用放置规则以通过订阅进行部署。	
spec.overrides	可选。需要覆盖的任何参数和值，如特定于集群的设置。	
spec.overrides.clusterName	可选。参数和值将被覆盖的一个或多个集群的名称。	
spec.overrides.clusterOverrides	可选。要覆盖的参数和值的配置。	
spec.timeWindow	可选。定义用于在订阅处于活跃状态或受阻时配置时间窗的设置。	
spec.timeWindow.type	可选，但在配置时间窗时是必需的。表示订阅在配置的时间窗内是处于活跃状态还是受阻。只有在订阅处于活跃状态时才会进行订阅的部署。	
spec.timeWindow.location	可选，但在配置时间窗时是必需的。为时间窗配置的时间范围的时区。所有时区都必须使用 Time Zone (tz) 数据库名称格式。如需更多信息，请参阅Time Zone 数据库。	
spec.timeWindow.daysofweek	可选，但在配置时间窗时是必需的。表示当使用了时间范围来创建时间窗时的每周天数。每周天数列表必须定义为数组，如 **daysofweek: ["Monday", "Wednesday", "Friday"]**。	
spec.timeWindow.hours	可选，但在配置时间窗时是必需的。定义时间窗的时间范围。必须为每个时间窗定义小时范围的开始时间和结束时间。您可以为订阅定义多个时间窗范围。	
spec.timeWindow.hours.start	可选，但在配置时间窗时是必需的。定义时间窗开始的时间戳。时间戳必须使用 Go 编程语言 Kitchen 格式 **"hh:mmpm"**。如需更多信息，请参阅 Constants。	

字段	描述	
spec.timeWindow.hours.end	可选，但在配置时间窗时是必需的。定义时间窗结束的时间戳。时间戳必须使用 Go 编程语言 Kitchen 格式 **"hh:mmpm"**。如需更多信息，请参阅 Constants。	-->

备**注：**

- 在定义 YAML 时，订阅可以使用 **packageFilters** 来指向多个 Helm chart、可部署资源或其他 Kubernetes 资源。但是，订阅只会部署一个 chart、可部署资源或其他资源的最新版本。
- 对于 **Namespace** 类型的频道，订阅 operator 使用注解来搜索可部署资源的版本。订阅 operator 通过搜索版本以查找要检索的适当可部署资源版本。如果您的频道属于 **Namespace** 频道，请包含用于标识可部署资源版本的注解。
- 对于时间窗，当您定义一个时间窗的时间范围时，必须将开始时间设置在结束时间之前。如果您要为订阅定义多个时间窗，时间窗的时间范围不能重叠。实际时间范围基于 **subscription-controller** 容器时间，可以设置为与您所在的时间和位置不同的时间和位置。
- 在您的订阅 spec 中，还可以定义 Helm 发行版本或可部署资源的放置位置，作为订阅定义的一部分。与可部署资源的定义类似，每个订阅都可在订阅定义中直接引用现有放置规则，或者直接定义放置规则。
- 当您要在 **spec.placement** 部分中定义订阅的放置位置时，对于多集群环境仅使用 **cclusters**、**clusterSelector** 或 **placementRef** 中的一个。如果您包含 **clusters**、**clusterSelector** 或 **placementRef** 中的多个，则会使用以下优先级确定订阅 operator 使用哪个设置：
  a. **placementRef**
  b. **clusters**
  c. **clusterSelector**

您的订阅类似以下 YAML 内容：

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: nginx
  namespace: ns-sub-1
  labels:
    app: nginx-app-details
spec:
  channel: ns-ch/predev-ch
  name: nginx-ingress
  packageFilter:
    version: "1.36.x"
  placement: # Placement rules help you facilitate multi-cluster deployments, see placement rules documentation.
    placementRef:
      kind: PlacementRule
      name: towhichcluster
```

```
  overrides: # See Deployable documentation for more about overrides. Include overrides for any
single cluster than requires some different settings
  - clusterName: "/"
    clusterOverrides:
    - path: "metadata.namespace"
      value: default
```

1.1.6.5.2.3. 订阅**文件示例**

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: nginx
  namespace: ns-sub-1
  labels:
    app: nginx-app-details
spec:
  channel: ns-ch/predev-ch
  name: nginx-ingress
```

1.1.6.5.2.3.1. 订阅时间**窗示例**

以下示例订阅包含多个配置的时间窗。一个时间窗在每个周一、周三和周五的 10:20 AM 和 10:30 AM 之间发生。一个时间窗也在每周、周三和周五的 12:40 PM 到 1:40 PM 之间发生。订阅只在这 6 个每周时间窗内开始部署。

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: nginx
  namespace: ns-sub-1
  labels:
    app: nginx-app-details
spec:
  channel: ns-ch/predev-ch
  name: nginx-ingress
  packageFilter:
    version: "1.36.x"
  placement:
    placementRef:
      kind: PlacementRule
      name: towhichcluster
  timewindow:
    windowtype: "active" #Enter active or blocked depending on the purpose of the type.
    location: "America/Los_Angeles"
    daysofweek: ["Monday", "Wednesday", "Friday"]
    hours:
      - start: "10:20AM"
        end: "10:30AM"
      - start: "12:40PM"
        end: "1:40PM"
```

1.1.6.5.2.3.2. 带**有覆盖的**订阅**示例**

以下示例包含软件包覆盖，以定义 Helm chart 的 Helm 发行版本的不同版本名称。软件包覆盖设置用于将名称 **my-nginx-ingress-releaseName** 设置为 **nginx-ingress** Helm 发行版本的不同发行版本名称。

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: simple
  namespace: default
spec:
  channel: ns-ch/predev-ch
  name: nginx-ingress
  packageOverrides:
  - packageName: nginx-ingress
    packageAlias: my-nginx-ingress-releaseName
    packageOverrides:
    - path: spec
      value:
        defaultBackend:
          replicaCount: 3
  placement:
    local: false
```

1.1.6.5.2.3.3. Helm 仓库订阅**示例**

以下订阅会自动拉取版本 **1.36.x** 的最新 **nginx** Helm 发行版本。当源 Helm 仓库中有新版本可用时，Helm 发行版本可部署资源会放置在 **my-development-cluster-1** 集群上。

spec.packageOverrides 部分显示了用于覆盖 Helm 发行版本值的可选参数。覆盖值合并到 Helm 发行版本 **values.yaml** 文件中，用于检索 Helm 发行版本的可配置变量。

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: nginx
  namespace: ns-sub-1
  labels:
    app: nginx-app-details
spec:
  channel: ns-ch/predev-ch
  name: nginx-ingress
  packageFilter:
    version: "1.36.x"
  placement:
    clusters:
    - name: my-development-cluster-1
  packageOverrides:
  - packageName: my-server-integration-prod
    packageOverrides:
    - path: spec
      value:
        persistence:
          enabled: false
          useDynamicProvisioning: false
        license: accept
        tls:
```

```
    hostname: my-mcm-cluster.icp
  sso:
    registrationImage:
      pullSecret: hub-repo-docker-secret
```

1.1.6.5.2.3.4. Git 仓库订阅**示例**

1.1.6.5.2.3.4.1. 订阅 **Git** 仓库**的特定分支和目**录

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
  name: sample-subscription
  namespace: default
  annotations:
    apps.open-cluster-management.io/git-path: sample_app_1/dir1
    apps.open-cluster-management.io/git-branch: branch1
spec:
  channel: default/sample-channel
  placement:
    placementRef:
      kind: PlacementRule
      name: dev-clusters
```

在本示例订阅中，注解 **apps.open-cluster-management.io/git-path** 表示订阅中订阅了频道中指定的 Git 仓库 **sample_app_1/dir1** 目录中的所有 Helm chart 和 Kubernetes 资源。默认情况下会在订阅中订阅 **master** 分支（branch）。在本示例订阅中，指定了注解 **apps.open-cluster-management.io/git-branch: branch1** 来订阅仓库的 **branch1** 分支。

1.1.6.5.2.3.4.2. 添加 **.kubernetesignore** 文件

您可以在 Git 仓库根目录中，或者在订阅注解中指定的 **apps.open-cluster-management.io/git-path** 目录中包含 **.kubernetesignore** 文件。

您可以使用此 **.kubernetesignore** 文件指定文件和/或子目录模式，以在订阅部署仓库中的 Kubernetes 资源或 Helm chart 时忽略它们。

您还可以使用 **.kubernetesignore** 文件进行精细过滤，以选择性地应用 Kubernetes 资源。**.kubernetesignore** 文件的特征格式与 **.gitignore** 文件相同。

如果没有定义 **apps.open-cluster-management.io/git-path** 注解，订阅会在仓库根目录中查找 **.kubernetesignore** 文件。如果定义了 **apps.open-cluster-management.io/git-path** 字段，订阅会在 **apps.open-cluster-management.io/git-path** 目录中查找 **.kubernetesignore** 文件。订阅不会在任何其他目录中搜索 **.kubernetesignore** 文件。

1.1.6.5.2.3.4.3. 应**用 Kustomize**

如果订阅的 Git 文件夹中有 **kustomization.yaml** 或 **kustomization.yml** 文件，则会应用 kustomize。

您可以使用 **spec.packageOverrides** 在订阅部署时覆盖 **kustomization**。

```
apiVersion: apps.open-cluster-management.io/v1
kind: Subscription
metadata:
```

```
  name: example-subscription
  namespace: default
spec:
  channel: some/channel
  packageOverrides:
  - packageName: kustomization
    packageOverrides:
    - value: |
patchesStrategicMerge:
- patch.yaml
```

为了覆盖 **kustomization.yaml** 文件，需要在 **packageOverrides** 中使用 **packageName: kustomization**。覆盖操作会添加新条目或更新现有条目。它不会移除现有的条目。

1.1.6.5.2.3.4.4. 启用 Git Webhook

默认情况下，Git 频道订阅每分钟克隆频道中指定的 Git 仓库，并在提交 ID 发生更改时应用更改。另外，您还可以将订阅配置为仅在 Git 仓库发送存储库 PUSH 和 PULL webhook 事件通知时应用更改。

要在 Git 仓库中配置 webhook，需要一个目标 webhook 有效负载 URL 和可选的 secret。

1.1.6.5.2.3.4.4.1. 有效负载（Payload） URL

在 hub 集群中创建路由（ingress），以公开订阅 operator 的 webhook 事件监听程序服务。

```
  oc create route passthrough --service=multicluster-operators-subscription -n open-cluster-management
```

然后，使用 **oc get route multicluster-operators-subscription -n open-cluster-management** 命令查找外部可访问的主机名。Webhook 有效负载 URL:

```
https://<externally-reachable hostname>/webhook`
```

1.1.6.5.2.3.4.4.2. WebHook secret

WebHook secret 是可选的。在频道命名空间中创建 Kubernetes secret。secret 必须包含 **data.secret**。请参见以下示例：

```
apiVersion: v1
kind: Secret
metadata:
  name: my-github-webhook-secret
data:
  secret: BASE64_ENCODED_SECRET
```

data.secret 的值是您要使用的 base-64 编码 WebHook secret。

最佳实践：为每个 Git 仓库使用唯一的 secret。

1.1.6.5.2.3.4.4.3. 在 Git 仓库中配置 WebHook

使用有效负载 URL 和 webhook secret 在 Git 仓库中配置 WebHook。

1.1.6.5.2.3.4.4.4. 在频道中启用 WebHook 事件通知

为订阅频道添加注解。请参见以下示例：

```
oc annotate channel.apps.open-cluster-management.io <channel name> apps.open-cluster-management.io/webhook-enabled="true"
```

如果您使用了一个 secret 来配置 WebHook，也需要为频道添加该注解，其中 **<the_secret_name>** 是包含 webhook secret 的 kubernetes secret 名称。

```
oc annotate channel.apps.open-cluster-management.io <channel name> apps.open-cluster-management.io/webhook-secret="<the_secret_name>"
```

1.1.6.5.2.3.4.4.5. 启用了 webhook 的频道的订阅

订阅中不需要特定于 webhook 的配置。

1.1.6.5.3. 放置规则示例

放置规则 (**placementrule.apps.open-cluster-management.io**) 定义的是可以部署可部署资源的目标集群。使用放置规则帮助您促进可部署资源的多集群部署。

1.1.6.5.3.1. 放置规则 YAML 结构

以下 YAML 结构显示放置规则的必需字段，以及一些常见可选字段。您的 YAML 结构需要包含一些必需字段和值。根据应用程序管理要求，您可能需要包含其他可选字段和值。您可以使用任何工具编写您自己的 YAML 内容。

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
  name:
  namespace:
  resourceVersion:
  labels:
    app:
    chart:
    release:
    heritage:
  selfLink:
  uid:
spec:
  clusterSelector:
    matchLabels:
      datacenter:
      environment:
  clusterReplicas:
  clusterConditions:
  ResourceHint:
    type:
    order:
  Policies:
```

1.1.6.5.3.2. 放置规则 YAML 值表

字段	描述
apiVersion	必需。将值设为 **apps.open-cluster-management.io/v1**。
kind	必需。将值设为 **PlacementRule** 以表示资源是放置规则。
metadata.name	必需。用于标识放置规则的名称。
metadata.namespace	必需。用于放置规则的命名空间资源。
metadata.resourceVersion	可选。放置规则资源的版本。
metadata.labels	可选。放置规则的标签。
spec.clusterSelector	可选。用于标识目标集群的标签
spec.clusterSelector.matchLabels	可选。目标集群必须存在的标签。
status.decisions	可选。定义放置可部署资源的目标集群。
status.decisions.clusterName	可选。目标集群的名称
status.decisions.clusterNamespace	可选。目标集群的命名空间。
spec.clusterReplicas	可选。要创建的副本数量。
spec.clusterConditions	可选。为集群定义任何条件。
spec.ResourceHint	可选。如果多个集群与您在前面字段中提供的标签和值匹配，您可以指定特定于资源的条件来选择集群。例如，您可以选择包含最多可用 CPU 内核的集群。
spec.ResourceHint.type	可选。将值设为 **cpu** 以根据可用 CPU 内核选择集群，或者设为 **memory** 以根据可用内存资源选择集群。
spec.ResourceHint.order	可选。将值设为 **asc** 表示升序，或者设为 **desc** 表示降序。
spec.Policies	可选。放置规则的策略过滤器。

1.1.6.5.3.3. 放置规则示例文件

现有放置规则可以包括以下字段来指示放置规则的状态。此状态部分附加在规则的 YAML 结构中的 **spec** 部分后面。

```
status:
  decisions:
    clusterName:
    clusterNamespace:
```

字段	描述
status	放置规则的状态信息。
status.decisions	定义放置可部署资源的目标集群。
status.decisions.clusterName	目标集群的名称
status.decisions.clusterNamespace	目标集群的命名空间。

- 示例 1

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: gbapp-gbapp
  namespace: development
  labels:
    app: gbapp
spec:
  clusterSelector:
    matchLabels:
      environment: Dev
  clusterReplicas: 1
status:
  decisions:
    - clusterName: local-cluster
      clusterNamespace: local-cluster
```

- 示例 2

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: towhichcluster
  namespace: ns-sub-1
  labels:
    app: nginx-app-details
spec:
  clusterReplicas: 1
  clusterConditions:
    - type: ManagedClusterConditionAvailable
      status: "True"
  clusterSelector:
    matchExpressions:
    - key: environment
```

```
      operator: In
      values:
      - dev
```

1.1.6.5.4. 应用程序示例

查看可以用来构建文件的示例和 YAML 定义。Red Hat Advanced Cluster Management for Kubernetes 中的应用程序 (**Application.app.k8s.io**) 用于查看应用程序组件。

有关创建和管理应用程序的更多信息，请参阅 创建和管理应用程序资源。

1.1.6.5.4.1. 应用程序 YAML 结构

要编写用于创建或更新应用程序资源的应用程序定义 YAML 内容，YAML 结构需要包含一些必需字段和值。根据应用程序要求或应用程序管理要求，您可能需要包含其他可选字段和值。

以下 YAML 结构显示应用程序的必需字段，以及一些常见可选字段。

```
apiVersion: app.k8s.io/v1beta1
kind: Application
metadata:
  name:
  namespace:
  resourceVersion:
  annotations:
  labels:
    app:
    chart:
    heritage:
    name:
    release:
spec:
  componentKinds:
  - group:
    kind:
  descriptor:
  selector:
    matchExpressions:
    - key:
      operator:
      values:
```

1.1.6.5.4.2. 应用程序 YAML 表

字段	描述
apiVersion	必需。将值设为 **app.k8s.io/v1beta1**。
kind	必需。将值设为 **Application** 以表示资源是应用程序资源。
metadata.name	必需。用于标识应用程序资源的名称。

字段	描述
metadata.namespace	用于应用程序的命名空间资源。
metadata.resourceVersion	应用程序资源的版本。
metadata.annotations	可选。应用程序的注解。
metadata.labels	可选。可部署资源的标签。
spec.componentKinds	可选。要与应用程序关联的资源类型列表。
spec.selector.matchExpressions	可选。用于将其他 Kubernetes 资源与应用程序关联的标签选择器。
spec.selector.matchExpressions.key	在定义标签选择器时是必需的。资源需要匹配的 Kubernetes 标签键。
spec.selector.matchExpressions.values	在定义标签选择器时是必需的。资源需要匹配的 Kubernetes 标签值。

定义这些应用程序的 spec 是基于 Kubernetes 特别兴趣小组 (SIG) 提供的应用程序元数据描述符自定义资源定义。您可以使用此定义来帮助编写自己的应用程序 YAML 内容。有关此定义的更多信息，请参阅 Kubernetes SIG 应用程序 CRD 社区规格。

1.1.6.5.4.3. 应用程序文件示例

应用程序的定义结构类似以下示例 YAML 内容：

```
apiVersion: app.k8s.io/v1beta1
kind: Application
metadata:
  labels:
    app: nginx-app-details
  name: nginx-app-3
  namespace: ns-sub-1
spec:
  componentKinds:
  - group: apps.open-cluster-management.io
    kind: Subscription
  selector:
    matchLabels:
      app: nginx-app-details
status: {}
```