TECH NOTE

Control-M and Kubernetes: Introduction

Orchestrate business workflows running in a Kubernetes environment

INTRODUCTION

Kubernetes (sometimes referred to as K8S for short) is a leading distributed computing platform for both traditional and cloud-native workloads. This document provides an overview of various implementation options for Control-M in this environment.

Audience
This document is intended for anyone familiar with the use of Control-M who has general knowledge of Kubernetes.

Additional information
For more about the subject presented here visit the links below:

Kubernetes

General Control-M Documentation

Control-M Automation API

Jobs-as-Code

Deployment patterns

There are numerous permutations for managing application workload and deploying Control-M components in a K8S environment. A common requirement that has been observed is to start, monitor and collect the completion status and output of an application running to completion in a pod.

To manage that pod, the two common approaches for deploying Control-M components are:

1. a traditional Control-M agent running outside of the Kubernetes cluster; or
2. a Control-M agent placed inside the K8S cluster.

In option 1, Kubernetes can be viewed as an application that exposes an automation interface via the kubectl cli. Please see here one implementation for integration with Control-M using Application Integrator.

Both Kubernetes and Control-M offer myriad options with almost unlimited flexibility for variety.

The rest of this document discusses some general principles to consider when embedding Control-M components within K8S clusters to orchestrate business application workflows running in Kubernetes. Additional documents are available containing detailed descriptions and examples of specific implementations.

If there are specific requirements in your environment that you don't find addressed in this document, please engage with your BMC Software contact. It is almost certain that additional options are available to offer a solution that will meet your needs.

RUNNING WORKLOAD

The fundamental requirement of application workflow orchestration in K8S is to manage applications contained in pods. This includes:

- start them
- monitor their execution
- determine their completion status
- collect their output
- perform operations like kill or rerun (this is a logical action at the Control-M job level which will almost always result in creation of a new Pod).

Furthermore, initiating the creation of the Pod should be determined by Control-M so that you get the full benefit of Control-M's workflow orchestration capabilities which include:

- Visualization of the Pod workload within the context of an end-to-end business service
- Intelligent SLA management
- Operational best practices
- User and authorization management
- Logging, reporting and auditing

The goal of combining these two technologies is to utilize Control-M to power business application workflows while harnessing the awesome capabilities of Kubernetes to manage compute resources, storage and networking.

Managing applications

Control-M starts application processes by:

1. Issuing requests to the K8S API Server. Examples are using the Kubernetes Python client, cURL or kubectl.
2. Running local jobs within the same Pod as the Control-M Agent (see next section).
3. Invoking an API exposed by an application. The application would likely be running in the K8S cluster but that is not mandatory. If the API endpoint/application server is reachable, the application can be running anywhere.

For option 1 the request creates a K8S object like a Pod, Deployment or Job. Either a YAML manifest is provided or all the attributes required for the object are passed to Control-M and submitted to the K8S API Server as part of the request.

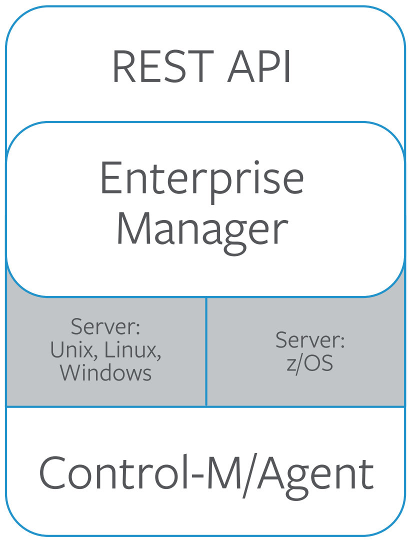
For option 2, a script or executable is available in the file system of the Pod in which a Control-M agent is running. There are several ways this can be configured including a Persistent Volume Claim or a separate container within the same Pod as the agent (sidecar container).

Option 3 is for interacting with an application that runs as a service and exposes APIs unique to its functionality. This use case is no different than any integration accomplished by Control-M either via a Control Module, Application Integrator, scripts or cli clients provided by the target application.

DEPLOYING CONTROL-M COMPONENTS

Control-M Architecture

This is a simplified view of the Control-M architecture.



The **Enterprise Manager** is responsible for visualization and presentation services.

The **Server** makes all decisions about tasks being run including which ones to launch next, analyze completion status and perform post-execution actions.

The **Control-M/Agent** runs and monitors tasks based on instructions received from Server.

The **REST API** exposes administrative and workflow management functions as RESTful endpoints.

Just as with Kubernetes itself there is the task of resource management and optimization and the "control plane" that helps to achieve that task, so too with Control-M the primary goal is running the application workflow described in the previous section and the components that make up the Control-M "control plane" consisting of the architecture described above.

Control-M manages workflows in widely varied infrastructure and application environments including on-premises, public and private cloud, ERPs, file transfers, and more. And similarly, the configuration options for how and where components are deployed are equally varied.

Control-M Agent Placement

Control-M/Agent can be deployed inside or outside the Kubernetes cluster, as mentioned earlier. The decision depends on:

1. If the application being managed is accessible remotely or not
2. Organizational preferences and ownership
3. Familiarity and level of comfort with technology

For example, a Kubernetes JOB object is started by invoking a RESTful Web Service, which can be performed from anywhere. Whereas running a specific script within a container may require "localhost" access and thus placement in the same pod. In the first case, the Control-M agent may be placed inside or outside the K8S cluster. In the second case, the agent must be in the same Pod as the script.

This document focuses on deploying a Control-M agent inside the Kubernetes cluster.

Agent Pod Structure

- Agent only - the pod consists only of a Control-M/Agent container
- Application and Agent (sidecar) - the pod consists of one or more application containers and a Control-M/Agent container. All containers in the Pod access each other as "localhost".
- Remote Host - a lightweight container with an SSH server to enable Control-M Remote Host functionality

Deploying an Agent Pod in Kubernetes

The agent can be run in any fashion from a singleton pod to any type of K8S Controller as long as the following requirements are met:

- The Control-M/Server host is reachable from the agent pod via hostname or IP Address
- The agent-to-server and server-to-agent ports are available

Considerations

Managing business workflows is frequently a mission-critical activity. You want to ensure you manage resources wisely, account for spikes in demand and provide resilience in case of failures. This section discusses some considerations for achieving these goals.

Hostgroups

Avoid binding jobs to specific pods by using hostgroups in job definitions.

Hostgroups provide load balancing and dynamic scaling during usage spikes.

Networking

Kubernetes offers a variety of ways to manage communications between pods and applications running outside the cluster. This relates to the communication between Control-M agent and Control-M Server. Since the Control-M agent currently is stateful the path between server and agent must remain static for the life of that specific session. That means controllers like Services whose function is to manage external-to-internal (to the K8S cluster) communications, typically will run only a single pod. Such an implementation may require multiple services to accomplish load balancing and high availability.

Node Placement

Placement of pods on nodes can be controlled by nodeSelector labels and pod Affinity/Anti-Affinity specifications. These are standard Kubernetes features and operate completely independent of Control-M.

Replication

Control-M agent is currently a stateful application that creates a unique connection with a Control-M Server. The workload (jobs) managed by one agent is not known to other agents. Therefore, patterns in which one external identity is mapped to several pods will not operate correctly. That functionality is provided by Control-M hostgroups. This was also mentioned in the "Networking" discussion and this means K8S objects like Deployments, ReplicaSets and Services should either have a one-to-one mapping between external identities and pods or not rely on the external-facing identity and allow pods to communicate directly with Control-M server (via hostNetwork or similar facilities).

DaemonSet is an object that already ensures the one-to-one mapping for each node in the K8S cluster.

Additional Reading

This document provides a high-level overview of the topics you may wish to consider when deploying Control-M agent to a K8S cluster. Detailed documents including specific best practices are under construction. This section will be updated with links as those documents become available.

CONCLUSION

Control-M is easily deployed in Kubernetes-based environments, allowing you to take full advantage of its enterprise grade workflow orchestration. Control-M meets operational production standards while providing advanced capabilities easily consumed by Dev, Ops and lines-of-business alike, including:

- End-to-end workflow connectivity – any application, any data source, and all your critical systems of record, mainframe to cloud
- SLA management with intelligent predictive analytics
- Auditing for compliance and governance
- Logs and output capture and management
- Proven stability with thousands of companies scaling from 10s to millions of jobs with zero downtime

FOR ADDITIONAL INFORMATION ON CONTROL-M

Visit our web page

521302