Model Deployment to Edge Clusters

Status: In Progress

Sponsor User: *IBM*

Date of Submission: 19 Aug 2021

Submitted by: David Booz

Affiliation(s): IBM

<Please fill out the above fields, and the Overview, Design and User Experience sections below for an initial review of the proposed feature.>

Scope and Signoff: *(to be filled out by Chair)*

Overview

<Briefly describe the problem being solved, not how the problem is solved, just focus on the problem. Think about why the feature is needed, and what is the relevant context to understand the problem.>

Support for independent and autonomous AI and ML model deployment was added to OpenHorizon a few years ago. Since that time, support for edge nodes which are manifest as Kubernetes clusters has also been added. However, when the edge cluster support was added, there was not sufficient time and resource to support model deployment to edge clusters. This design proposes to address that problem by introducing model deployment to edge clusters.

Model deployment in OpenHorizon touches on several aspects of the system and involves several roles that interact with the models at various stages of the model deployment. The goal of this feature is to enable model deployment to edge clusters without altering the way in which the system handles models nor the way in which the relevant roles interact with the system. Concretely, this design will:

- Enable policy based deployment of models to edge clusters, with no changes to the existing model policy schema.
- Enable deployed applications to receive models and model updates using the same APIs used by applications which run on an edge device.

Design

<Describe how the problem is fixed. Include all affected components. Include diagrams for clarity. This should be the longest section in the document. Use the sections below to call out specifics related to each aspect of the overall system, and refer back to this section for context. Provide links to any relevant external information.>

The main problem to be solved in this design is to ensure that the existing programming model (i.e. the ESS API) for receiving models (and updates) can be used when a service is running on an edge cluster. This problem is decomposed into 2 smaller problems:

- How to enable network access from a service in the edge cluster to the agent's ESS API?
- How to communicate the protocol, host, port, SSL cert and login credentials necessary to access the ESS API?

These two problems are discussed in the following sections.

Receiving models

The ESS API is the means by which the service can poll for new and updated models. On edge devices, when a service is started by the agent, it is provided with a URL, login credentials, and an SSL certificate for accessing the ESS API. The URL and SSL certificate are the same for every service that is started by a given agent. The login credentials are unique to each service instance, and are the means of identifying which models the service is able to receive. The URL is provided through OpenHorizon platform environment variables (HZN_ESS_API_PROTOCOL, HZN_ESS_API_ADDRESS, HZN_ESS_API_PORT), the login credentials and SLL cert are mounted to the service container at a location indicated by two other environment variables; HZN_ESS_AUTH and HZN_ESS_CERT. Please note that the SSL cert does not contain a private key, it is a client side cert. The only truly sensitive information is the login credentials.

On edge clusters, the service that is deployed is actually a k8s operator (built by the service developer). The operator is responsible for starting the real application containers. Because OpenHorizon has no visibility to the application containers, it is the responsibility of the OH deployed operator to forward the HZN_ESS environment variables, login credentials and SSL cert to the relevant application containers. An operator deployed as an OH service does not need to perform this forwarding if model deployment is not a feature required by the application.

There is a subtle but important difference in how the operator will interact with the HZN_ESS_AUTH and HZN_ESS_CERT environment variables. On edge clusters, these env vars will contain the name of a k8s secret containing the respective information; one for the login credentials and one for the SSL certificate. This is different from edge devices, where that env var contains the name of the folder where the credentials are mounted. This difference will enable the operator to simply attach the secrets to any application containers that need them, in a way that is natural for k8s application developers. The OH agent will create these two secrets as part of deploying the operator.

Enabling the ESS API

On edge otdevices, the ESS API endpoint is provided by the agent itself. The device agent uses a Unix Domain Socket as the network address of the API. Clearly this will not work for the edge cluster agent, for many reasons. Therefore, the agent's ESS API needs to be accessible over the cluster's internal network. This is accomplished in k8s by attaching a k8s service (the term service is now overloaded) to the agent's deployment. It is the responsibility of the agent install script to establish a k8s service for the agent's ESS API. The k8s service looks something like this:

```
apiVersion: v1
kind: Service
metadata:
  name: agent-service
  namespace: openhorizon-agent
spec:
  selector:
    app: agent
  ports:
    - protocol: TCP
      port: 8443
```

This k8s service definition includes a few notable aspects:

1. The API host name provided to the application is the metadata.name field; HZN_ESS_API_ADDRESS = "agent-service"
2. The API protocol is https; HZN_ESS_API_PROTOCOL="https"
3. The API port is 8443; HZN_ESS_API_PORT=8443
4. This k8s service is attached to an app called "agent" which is the app name given to the agent when it is installed.
5. This service is defined in the "openhorizon-agent" k8s namespace.

In order for the above settings to be variable, i.e. set by the agent installer, the k8s service definition above needs to be conditioned to reflect those differences before it is installed to the edge cluster.

This k8s service allows the edge cluster agent to continue to be the ESS API provider, and enables application containers within the cluster to access the API, even if the agent is moved from one pod to another.

Model Deployment

There are numerous "node type" checks throughout the anax code for the agent and the agbot, some of which disable the deployment of models to edge clusters. These checks should be removed were appropriate to the re-enable model deployment. Removing these checks will allow the ESS to be started in edge cluster agent and allow the agbot to route models to edge cluster nodes. Aside from these minor code updates, model deployment should work exactly as it does for device nodes. When an agreement is formed, the agbot instructs the MMS to deploy models to the node. Since the ESS will be enabled in the edge cluster agent, the agbot's routing instructions will be performed by the MMS exactly as is done for agreements with device nodes.

Model Storage in the Agent

On edge devices, models deployed to an edge node are stored in root protected storage on the host. For edge clusters, the models are stored in a k8s persistent volume that is available when the agent is installed. A persistent volume is required in case the agent is moved from one pod to another. The persistent volume must be large enough to accommodate the expected number and size of models that will be needed by the edge cluster. Given the potential variability of storage requirements, the node owner must be able to provide this persistent volume to the agent install script. A default persistent volume can be created by the agent install script if one is not provided by the node owner, but that default is unlikely to meet the requirements of all use cases.

User Experience

<Describe which user roles are related to the problem AND the solution, e.g. admin, deployer, node owner, etc. If you need to define a new role in your design, make that very clear. Remember this is about what a user is thinking when interacting with the system before and after this design change. This section is not about a UI, it's more abstract than that. This section should explain all the aspects of the proposed feature that will surface to users.>

As an org admin, I want to write a model deployment policy that targets a service that is deployed to an edge cluster.

As a service developer, I want to receive deployed models by using the agent's ESS API when my application is running on an edge cluster.

As a node owner, I want the agent installed and configured on an edge cluster to automatically support deployment of models to services that run on my edge cluster.

Command Line Interface

<Describe any changes to the hzn CLI, including before and after command examples for clarity. Include which users will use the changed CLI. This section should flow very naturally from the User Experience section.>

None

External Components

<Describe any new or changed interactions with components that are not the agent or the management hub.>

None

Affected Components

<List all of the internal components (agent, MMS, Exchange, etc) which need to be updated to support the proposed feature. Include a link to the github epic for this feature (and the epic should contain the github issues for each component).>

Agent (the k8s agent container)

Agent install

Security

<Describe any related security aspects of the solution. Think about security of components interacting with each other, users interacting with the system, components interacting with external systems, permissions of users or components>

APIs

<Describe and new/changed/deprecated APIs, including before and after snippets for clarity. Include which components or users will use the APIs.>

None

Build, Install, Packaging

<Describe any changes to the way any component of the system is built (e.g. agent packages, containers, etc), installed (operators, manual install, batch install, SDO), configured, and deployed (consider the hub and edge nodes).>

Documentation Notes

<Describe the aspects of documentation that will be new/changed/updated. Be sure to indicate if this is new or changed doc, the impacted artifacts (e.g. technical doc, website, etc) and links to the related doc issue(s) in github.>

Edge cluster agent install

Test

<Summarize new automated tests that need to be added in support of this feature, and describe any special test requirements that you can foresee.>

1. Ensure that the ESS starts and stops (use unregister command) on the edge cluster agent.
2. Ensure that models are removed from the edge cluster agent (including the storage) when deleted from the CSS or undeployed (e.g. model policy changed) from the node.