Edge Cluster Agent with Namespace Permission

Status: In Progress

Sponsor User: *IBM*

Date of Submission: 12 Sep 2022

Submitted by: David Booz (booz@us.ibm.com)

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.>

Currently, the edge cluster agent is installed with cluster permissions which must be granted by the Kubernetes cluster admin. This means that a DevOps team wishing to make use of Open Horizon is required to engage the Ops team responsible for providing Kubernetes services. In order to enable DevOps to be more self sufficient, this barrier needs to be removed. In effect, a DevOps team needs to be able to install an edge cluster agent with permission to a specific namespace so that the agent can manage service deployments in that namespace, and only in that namespace. As a result, a namespace scoped edge cluster agent is no longer able to deploy services into any namespace (as it does currently).

Once this barrier is removed, another set of seemingly disjoint use cases is also solved. When multiple DevOps teams are utilizing an edge cluster in this way, they are effectively using it in a pseudo multi-tenant fashion. That is, each DevOps team would expect to be able to manage their own agents and services deployed by those agents without interference from agents in other namespaces within the same cluster. To the extent that Kubernetes administration enables multi-tenancy within a cluster, a namespace scoped agent supports those goals. Thus, a provider of kubernetes services could enable each of their customers to independently exploit OH in their own namespace.

The use cases for a single cluster scoped agent with cluster wide permissions are still valid and are not altered by this design. Further, it is desirable that OH can support a single edge cluster containing both a cluster scoped agent and one or more namespace scoped agents.

It is not a goal of this design to provide an edge cluster agent that supports more than 1 namespace but less than the entire cluster.

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.>

Assumptions:

This design assumes that when edge cluster deployers are deploying a given service, they will be dealing primarily with namespace scoped nodes or cluster scoped nodes, but not a mix. Therefore the design should enable a simple experience for these two cases. Further the design assumes that when edge cluster deployers are deploying a given service, it MUST be possible for them to work with a mix of namespace and cluster scoped nodes, but that these situations are more complex and therefore require more cognitive energy to understand.

Prior to this design, the OH cluster agent allows an edge cluster service definition to contain a kubernetes namespace definition (yaml) embedded within the operator definition. This was a tactical step taken to enable service deployment into a user specified namespace. This feature is inconsistent with the proper separation of concerns between implementation and deployment, and therefore it's continued use will be discouraged (but not yet deprecated). This design accommodates edge cluster services that are already built this way, but does not encourage continued usage.

Agent Install:

The agent install script is updated to include a namespace flag indicating the target namespace of the agent:

```
./agent_install.sh --namespace MyProjectNamespace ...
```

The user invoking the install script MUST have permission to the MyProjectNamespace, otherwise the install will fail. When installing a namespace scoped agent, the kubernetes role bindings for the agent will be limited to namespace scoped permissions. This ensures that the agent runs with limited capability inside the kubernetes cluster.

The absence of the --namespace flag indicates a desire to install the agent as it is done prior to this design. The agent will have cluster wide permissions, and it will be installed into the **openhorizon-agent** namespace.

Note: The use of SDO/FDO to install agents in only supported for devices, therefore SDO/FDO install is out of scope for this design.

Node Properties:

A new built-in node property called **openhorizon.kubernetesNamespace** is introduced, the value reflects the namespace in which the agent is installed. This property is read-only, it is always set by the OH runtime and is not settable by any user role. This property MAY be used in a deployment or policy constraint expression.

Deployment:

When an edge cluster service is deployed, by default, it is deployed into the same namespace as the agent/node.

When deploying an edge cluster service, the service deployer MAY write a constraint expression referencing the built-in **openhorizon.kubernetesNamespace** property in order to limit the placement of the edge service to nodes in a specific namespace or set of namespaces. This is the recommended way to target edge cluster services to nodes in a specific namespace (or namespaces).

However, OH currently allows an edge cluster service definition to contain a kubernetes namespace definition (yaml) embedded within the operator definition. The namespace definition indicates the target namespace into which the service should be deployed. There are two problems with this feature. First, it is the wrong placement of function because the namespace in which a service runs is a deployment concern, not an implementation concern. Second, it creates a semantic conflict when the deployer tries to deploy to a namespace scoped node in a different namespace. The use of this feature is not recommended and MAY be deprecated in a future version of OH.

To address both concerns, the deployer needs a way to explicitly indicate the target namespace of a deployment.

A new field is added to the service section of a deployment policy, indicating the target namespace for the deployment. This namespace overrides a namespace definition in the operator definition of an edge cluster service.

```
"service": { ...
   "clusterNamespace": <string>
}
```

This field is optional and ignored for services deployed to a device.

For cluster scoped nodes, this field defines the namespace where the deployment occurs, effectively overriding an embedded namespace definition in the service.

For namespace scoped nodes, this field acts as a built-in constraint that causes namespace scoped nodes in namespaces other than the one specified by this field to be eliminated as deployment targets. Effectively, this field acts as a built-in constraint ANDed to the user specified constraint expression. The deploycheck CLI MUST detect this case.

In the absence of the `"clusterNamespace"` field, a namespace definition embedded in the operator definition acts as a built-in constraint that causes namespace scoped nodes in namespaces other than embedded namespace definition to be eliminated as deployment targets. Effectively, the embedded namespace definition acts as a built-in constraint ANDed to the user specified constraint expression. The deploycheck CLI MUST detect this case.

There could be a semantic conflict if the deployer specifies a `"clusterNamespace"` and uses **openhorizon.kubernetesNamespace** in a constraint expression. It is likely that such a policy will result in no deployments, which would be the technically correct behavior but it might also be surprising to the user. This is why the hzn deploycheck command exists, to help the user understand semantic mismatches in deployment policy. The deploycheck command MUST detect whether or not there is a real semantic conflict in this case.

Deployment Examples:

Here are some examples that illustrate deployment outcomes resulting from the behavior described above. The examples show that the design is compatible with existing OH behavior and also enables control of the target namespace for the deployer.

1. An edge cluster node with cluster privileges is chosen as a deployment target by the policy's constraint expression. The deployment policy has no `"clusterNamespace"` defined and no embedded namespace definition. The edge service is deployed in the openhorizon-agent namespace.
2. An edge cluster node with cluster privileges is chosen as a deployment target by the policy's constraint expression. The deployment policy has `"clusterNamespace":"ABC"`. The edge service is deployed in the ABC namespace.
3. An edge cluster node with cluster privileges is chosen as a deployment target. The deployment policy has no `"clusterNamespace"` defined. The edge service has an embedded namespace definition for XYZ. The edge service is deployed in the XYZ namespace.
4. An edge cluster node with cluster privileges is chosen as a deployment target. The deployment policy has `"clusterNamespace":"ABC"` defined. The edge service has an embedded namespace definition for XYZ. The edge service is deployed in the ABC namespace.

And finally, some additional examples for namespace scoped nodes.

1. A node in namespace ABC. The deployment policy has `"clusterNamespace":"ABC"`. The edge service is deployed to the node.
2. A node in namespace ABC. The deployment policy has no `"clusterNamespace"`. The edge service is deployed to the node.
3. A node in namespace ABC. The deployment policy has no `"clusterNamespace"`. The deployment policy has a constraint expression **openhorizon.kubernetesNamespace=XYZ**. The edge service is NOT deployed to the node.
4. A node in namespace ABC. The deployment policy has `"clusterNamespace":"ABC"`. The deployment policy has a constraint expression **openhorizon.kubernetesNamespace=XYZ**. The edge service is NOT deployed to the node. The user received a warning when publishing the policy that the policy might result in no service deployments.

Pattern:

A new field is added to the schema of a pattern (as a top level field in the schema), indicating the target namespace for the pattern's deployment.

```
 "clusterNamespace": <string>
```

The "clusterNamespace" field is optional and ignored for patterns deployed to a device. Namespace scoped nodes not in the specified namespace are not eligible to deploy the pattern. Cluster scoped nodes only deploy patterns that have an empty "clusterNamespace" field. A pattern with an empty "clusterNamespace" MUST NOT be deployed to a namespace scoped node.

A namespace specified in the pattern overrides any namespace defined in the operator definition of all services in the pattern.

A pattern is in error if it attempts to deploy services to a namespace scoped node where the collection of services in the pattern are NOT deployable to the same namespace. Clearly this can only happen when the "clusterNamespace" is specified in the pattern definition and one of the services contains an embedded namespace definition.

Node:

The node resource is also updated to contain the cluster namespace so that it is displayed in the node related CLI commands.

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.>

Terminology:

Cluster scoped agent - An OH agent installed in an edge cluster node where the agent has permission to deploy services into any namespace.

Namespace scoped agent - An OH agent installed in an edge cluster where the agent has permission to deploy services into ONLY the namespace where it is installed.

DevOps user - a conflation of roles found in the practice of DevOps; e.g. service developer, or service deployer.

Usage scenarios:

As a DevOps user, I want to install the OH agent into one or more namespaces that I have permission to use for my project.

As a service deployer, I want to select the namespace into which a service is deployed, for both cluster scoped and namespace scoped agents.

As a service deployer, I want to use a node's namespace as (one of) the criteria for selecting deployment targets.

As a service deployer, I want to know if my deployment policy will deploy to a namespace or cluster scoped node.

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.>

There are no new commands or verbs introduced by this design, but the behavior of some existing CLI commands is changed to accommodate this feature.

hzn exchange deployment addPolicy -f <policy_definition>

If the policy specifies `"clusterNamespace"` and the property **openhorizon.kubernetesNamespace** is present in the constraint expression, a warning is provided telling the user that this deployment policy may result in no service deployment. The user will be directed to use the hzn deploycheck command to verify whether or not a deployment will result.

hzn deploycheck

There are several situations outlined in the design above where deployment may or may not occur as expected, depending on the depth of knowledge of the user. The hzn deploycheck command is enhanced to identify:

- a semantic conflict if the deployer specifies a `"clusterNamespace"` and uses **openhorizon.kubernetesNamespace** in a constraint expression
- deployment target selection (or not) in the absence of the `"clusterNamespace"` field, and a namespace definition embedded in the operator definition

- deployment target selection (or not) in the presence of the `"clusterNamespace"` field, and a namespace definition embedded in the operator definition
- correct deployment target selection as defined in the Deployment section above

hzn exchange service publish

If the service definition contains an embedded namespace definition, provide a warning to the user that due to the use of an embedded namespace definition they are not following best practices and therefore the service might not be deployable. Use the hzn deploycheck command before this service is deployed to ensure that the policy will select nodes as expected.

hzn exchange pattern

External Components

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

In some sense, the agent being installed into a specific kubernetes namespace is a change to the integration between the agent and kubernetes. However, the design does not require the use of new kubernetes APIs (that aren't already being used) or concepts.

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).>

The agent and Exchange are the only components affected by this design.

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>

When a namespace scoped agent is installed, it is given kubernetes privileges within its specific namespace but not beyond that scope.

APIs

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

The Exchange API is updated to support the new **clusterNamespace** field in a deployment policy resource.

The Exchange is unaware of node, service or deployment properties within its schema definitions.

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).>

This feature extends the agent install capabilities to include installation of edge cluster agents into a specific kubernetes namespace so that the agent can manage service deployment within that namespace.

There are no changes to how agents are built, packaged or distributed.

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.

Need doc for:

- Authoring edge cluster services: Add a note that packaging a namespace definition inside an operator definition is not considered a best practice. Service developers should allow deployers to choose the target namespace in the deployment policy.
- Deploying edge cluster services: Document the new **clusterNamespace** field in the deployment policy, node and pattern. Describe how it's used and how it plays into the algorithm used by the Agbot to determine where edge cluster services are placed.
- Policy: Document the new built-in property **openhorizon.kubernetesNamespace** for edge node policies.
- Installing edge cluster agent: Document the new **--namespace** flag in the agent-install script.

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.>

A comprehensive test would cover:

- Agent installation on microk8s and k3s
- All changes to the CLI behaviors
- Regression test of previous behavior with the cluster scoped agent
- See the deployment examples as the basis for a rigorous test of new deployment behavior