Continuously Available Node Groups

Status: Complete

Sponsor User: *IBM*

Date of Submission: 12 May 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.>

Open Horizon generally treats nodes as entities with an independent lifecycle, apart from all other nodes. But there are use cases, such as using sensors for monitoring critical systems, where it is important to have redundant monitoring in place so that there is always at least one monitoring agent operating. This is of course similar to the principles of high availability and continuous availability that are commonly found within IT systems. Open Horizon already contains a little known feature, called HA Groups, that enables nodes to be associated as a group such that at least 1 copy of a service deployed to the group is always running. Further, Open Horizon also ensures that when services are upgraded, the upgrade will be rolled across all the members of a node group such that at least 1 copy of the service is always running. One of the problems with the existing HA Group support is that it is not dynamic. For example, nodes must be added to a group as part of registering them with the management hub. Nodes cannot be removed from a group. Nodes cannot be added to an existing group without unregistering the entire group and registering each node again with the new group members. Node registration is something that happens once for the lifetime of the node. A node should never need to be unregistered unless it is being decommissioned.

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 design proposes to enhance the current concept of HA node groups by enabling organization administrators and node owners to create HA node groups at any time in the lifecycle of a node. Further, the design proposes to loosen the current restriction that all services deployed to a node in an HA group are deployed on all nodes in the group, enabling the use of heterogeneous node equipment within a group. Note that for the purposes of this design, the HA node group concept is intended to provide both HA and CA for the services running on those nodes. Following are the key design constraints which define a new HA node group concept:

1. Nodes in an HA group MAY have different node policies.
2. Adding a node to an HA Group MUST NOT terminate/restart running services.
3. Nodes MAY be placed into an HA Group after node registration.
4. A node MUST be in 0 or 1 HA Groups. A node MUST NOT be in more than 1 HA Group.
5. The nodes in an HA Group are identified by node Id.
6. When forming an HA Group, the system MUST ensure that group membership is recorded atomically in all relevant API resources.
7. A user MUST have permission to modify all the node's (resources) in an HA Group in order to form the group.
8. A service that is deployed to nodes in an HA Group MUST be upgraded in a rolling restart in order to avoid a complete outage of the service.
9. The node agent on nodes in an HA Group MUST be upgraded in a rolling restart in order to avoid a complete outage of the agent itself.

Agbot

The Agbot is responsible for ensuring that all deployment policies have been checked for compatibility against all nodes in the system, and making adjustments to the deployed state of services as node policy, deployment policy and service policy change over time. The Agbot already has support for ensuring that service upgrades are performed in a rolling fashion across an HA Group. The existing support will have to be augmented in the following ways:

1. HA Group membership is obtained from the new /hagroup resource in the Exchange. The existing HA support obtains this info from an internal representation of the node (in the code it's called the producer policy, and the info is also saved in the Agbot's agreement object in the DB). The HA Group membership should be removed from this internal representation and obtained from the /hagroup resource. The /hagroup resources will also need to be added to the resource cache in the Agbot.
2. The existing HA support assumes that ALL services running on a node in an HA Group are supposed to be running on ALL nodes in the group. This assumption is no longer true with this design. The Agbot needs to perform some additional checking (while managing a rolling upgrade) to understand which nodes in the group the service should be running on, and ensuring that it is always running on at least one of them. The Agbot MUST NOT assume that the service being upgraded is intended to be running on all nodes in the group. A service is intended to be running on a node if the node policy is compatible with the service's policy and all deployment policies that reference the service.

Exchange Changes API

When new resources are added to the system, the scope of change notification of those resources needs to be defined. Both the Agbot and the agent need to be aware of /hagroup resource creation/update/deletion.

Agent Upgrades

In addition to rolling upgrade support for services, agent upgrades also need to be performed in a rolling fashion across all the nodes in an HA node group. Agents are responsible for autonomously upgrading based on node management policy (NMP) as defined by the administrator, therefore there is no central entity that is able to coordinate across agents within a group. The only entity in the system capable of assisting with the coordination is the Agbot.

When a node in an HA group matches with an NMP it will start the process of upgrading by first downloading the required packages. The agent calls an Agbot API to assist with coordination across the group. The call to the Agbot occurs after downloading the upgrade packages because the node will not know until it downloads the manifest whether or not the NMP will require it to take any action. Once the download is complete, the agent's NMP subworker (that is also tasked with initiating download) will call the Agbot API with the name of the NMP it is ready to execute. When the Agbot receives the API request, it will lock the table in its database that contains currently upgrading nodes. Within this transaction, the Agbot will check if any of the node's peers are in that table. If it finds none, it will add the node to the table before releasing the lock. The lock will prevent peer Agbots from being able to change the table state in between the read and write which could result in multiple nodes in one HA group upgrading simultaneously. The thread carrying out the atomic database read/write call will be an Agbot API worker thread, so waiting for a lock is not a problem for performance. Finally, the Agbot will monitor the currently upgrading HA node table by using a subworker that periodically checks the status of the NMP for each node in the table. The subworker can query through the exchange cache and listen for node change notifications and the changes worker will handle dropping stale caches as it does for other resources currently. When a node receives a 'no' from the Agbot, the NMP monitoring subworker that made the call will end its process and try again when it runs next.

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 place two or more nodes into an HA Group with rolling service upgrades so that my services will be continuously available.
- As an org admin, I want to add a node to an HA Group without affecting currently deployed services.
- As an org user, I want to place two or more nodes into an HA Group with rolling service upgrades so that my services will be continuously available.
- As an org user, I want to add a node to an HA Group without affecting currently deployed services.
- As a device owner, I want to place two or more nodes into an HA Group with rolling service upgrades so that my services will be continuously available.
- As a service deployer, I want to deploy non-HA services to a subset of members in an HA Group to avoid compute resource consumption for services that don't need to be continuously available.
- As a service deployer, I want to deploy a service ONLY to nodes in an HA Group.
- As a device owner, I want rolling agent upgrades within my HA Group, so that my services (and nodes) will be continuously available.
- As an org administrator, I want rolling agent upgrades within my HA Group, so that my services (and nodes) will be continuously available.

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.>

To create, modify and delete HA Groups, use the following commands:

hzn exchange hagroup create <name> --nodeId node1 --nodeId node2 [--nodeId node3]

hzn exchange hagroup update <name> --nodeId node1 --nodeId node2 [--nodeId node3]

hzn exchange hagroup delete <name>

hzn exchange hagroup list [<name>]

To check if a service will be deployed to all nodes in an HA Group, we need a new flag on the command. This flag will cause the command to retrieve the list of nodes in the HA group and compare them all against the policy inputs:

hzn deploycheck --checkHA

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 - Use of the Agbot API to direct agent upgrades.

Agbot - Awareness of a node's HA Group membership for making agreements, and an API for tracking rolling agent upgrades.

CLI - To list, add, remove nodes from an HA Group.

Exchange - To hold the new HA Group membership API.

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>

None

APIs

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

Exchange APIs

The following new APIs are introduced in this design. Any user in an org can use these APIs (or corresponding CLI). Org users can only create/modify /delete HA groups containing nodes that the user has permission to modify.

The HA Group object schema:

```
{
    "name": "hagroup name",
    "members": [ "node1234", ... ],
    "updated": <update time stamp>
}
```

Create a new node group. The caller must have permission to modify all the nodes listed in the body (shown above). The Exchange will set a reference to this object onto all the node resources listed in the body. The Exchange will return an error (409) if one of the nodes is already in an hagroup. To remove a node from a group, use the PUT API to provide the list of nodes that should be in the group.

POST /org/<org>/hagroups

Modify the group membership of an existing group. All the desired members of the group MUST be listed in the body. This API behaves like a full replace.

PUT /org/<org>/hagroups

List all the hagroups.

GET /org/<org>/hagroups

List all members of an hagroup.

GET /org/<org>/hagroups/<name>

Delete an hagroup.

DELETE /org/<org>/hagroups/<name>

Node Resource:

In addition to the new hagroup resource, the node resource is also extended with a new field called **"ha_group"** that contains a reference to the HA group in which this node is a member. This field is updated for all nodes in an HA group when a new HA group is created, updated or deleted. Updates to this field are atomic with updates to the hagroup resource.

Agbot APIs

To tell the Agbot that the node is ready to upgrade the agent.

POST /org/<org>/node/<node-id>/upgrade

200 - yes, go ahead with the upgrade

409 - no, another node is upgrading

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).>

None

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.>

1. Need an Overview of how HA Groups work on the OH doc site. Hopefully the material from this doc can be used for that content.
2. Need a new article describing how to use HA Groups, this would be focused toward the administrator and node owners. It could be the same article for both roles (but we might change our mind on this AFTER we have tried to write it). This article would show the use of the CLI and probably need to include or refer to a CLI reference section.
3. Update/remove HA doc in the anax repo for the HA /attribute API. Support for this capability is being removed.

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. Edge Clusters are out of scope for this support. Edge clusters already natively support HA and CA, and therefore don't need any special assistance from OpenHorizon.
2. Test service lifecycles with services deployed to HA groups that are homogeneous (all members have the same services) and heterogeneous (there is at least 1 service common to all members, but some services are only running on a subset of members).
3. Performance test of service upgrades with and without HA groups in the system.