THIS DOC IS PUBLIC

Implicit Namespace Adoption Proposal

Status: Obsolete
Authors: Sam Dowell
Last Updated: Oct 7, 2022
Approvals:

- [ ] **Janet Kuo**
- [x] ~~**Karl Isenberg**~~
- [ ] **Nan Yu**

Reviews:

- [ ] **tomasl@spotify.com**
- [ ] **Andy Lu**
- [ ] **(Suggest your own name here)**

Background

This PR sparked some discussion around how Config Sync handles “implicit” namespaces. An implicit namespace is defined as a namespace that is created by a RootSync that manages resources in a namespace but not the namespace itself. If the namespace does not exist when such a RootSync is created, then the namespace gets created “implicitly” so that the remaining resources can be reconciled. An implicit namespace is given the normal managed resource annotations in addition to a `PreventDeletion` annotation. The `PreventDeletion` annotation causes the namespace to be orphaned when all resources in the implicit namespace are deleted from the RootSync repository.. This is because we can't infer whether it is safe to delete a namespace that was created implicitly, so we always leave them on the cluster. Implicit namespaces are only created for unstructured repos.

The PR above defines the following generic configuration:

- A `RootSync` named `sync-a` declares `namespace-a` explicitly
- A `RootSync` named `sync-x` declares resources in `namespace-a`, but not `namespace-a` itself
- A `RootSync` named `sync-y` also declares resources in `namespace-a`, but not `namespace-a` itself
- A `RootSync` named `root` declares all the above RootSyncs

Implicit namespaces can lead to undesirable scenarios in configurations such as this one where one RootSync manages the namespace itself and other RootSync(s) manage resources in the namespace. There exists a race condition where if `sync-x` or `sync-y` create the namespace implicitly, then `sync-a` will fail to sync since the namespace already exists and is managed by another `RootSync`. The above configuration will only succeed if the reconciler of `sync-a` creates `namespace-a` before `sync-x` and `sync-y`,or the explicit `namespace-a` needs to be removed from `sync-a`.

Objective

A solution to this problem should ideally have the following properties:

- `sync-x` and `sync-y` should be able to sync even if `sync-a` has not been synced yet (i.e. they should be able to create the namespace implicitly)
- `sync-a` should be able to sync and claim ownership of the namespace, even if it was previously created implicitly
- `sync-a` should be able delete the (explicit) namespace, after having claimed ownership
- `sync-x` and `sync-y` should not be able to delete the (implicit) namespace

Design Proposal

The proposal is to introduce a softer idea of ownership than the current managed/unmanaged resource annotation. A namespace that is created implicitly will be given an “implicitly managed” annotation, which will behave similarly to the existing managed annotation with some exceptions.

Some key differences in how an implicitly managed namespace is treated:

- If a RootSync manages resources in a namespace but not the namespace itself, it will create the namespace if it does not exist and give it the “implicitly managed” annotation.
- If a RootSync wants to explicitly manage a namespace that currently has the implicitly managed annotation, it can “adopt” the namespace by removing the implicit annotation and managing the namespace itself.
- Namespaces with the “implicitly managed” annotation will be protected from deletes but not updates.
    - This will require changes to the webhook and remediator.
    - But as long as we don't pass the namespace to the applier if it already exists, then we probably don't need to update the applier itself.

Scenarios

Create implicit RootSync before explicit RootSync

- `sync-x` is created which creates an implicit namespace
- `sync-y` is created which syncs without updating the namespace

Create implicit RootSync before explicit RootSync

- `sync-x` is created which creates an implicit namespace
- `sync-a` is created which then takes ownership of the implicit namespace

Create explicit RootSync before implicit RootSync

- `sync-a` is created which creates a managed namespace
- `sync-x` is created which syncs without updating the namespace

Scenario	Final annotation	Namespace deleted?
- `sync-x` creates implicit namespace - `sync-a` takes ownership of namespace - `sync-x` resources are pruned	Managed	No
- `sync-x` creates implicit namespace - `sync-y` uses implicit namespace - `sync-x` resources are pruned	Implicit	No
- `sync-x` creates implicit namespace - `sync-a` takes ownership of namespace - `sync-a` resources are pruned	Implicit*	No*

Prune implicit RootSync with managed namespace

- `sync-x` is created which creates an implicit namespace
- `sync-a` is created which takes ownership of the implicit namespace
- `sync-x` resources are pruned

In this scenario the namespace is managed by `sync-a`, so `sync-x` resources are pruned without deleting the namespace.

Prune implicit RootSync with implicit namespace

- `sync-x` is created which creates an implicit namespace

- `sync-y` is created which uses the existing implicit namespace
- `sync-x` resources are pruned

In this scenario no RootSync has taken ownership of the namespace, so we don't know if there may be some other RootSync which depends on the implicit namespace unless we tracked all namespaces which have an implicit dependency on the namespace. The namespace is not deleted and the implicit annotation remains on the namespace. Another RootSync can still take ownership of this namespace in the future.

Prune explicit RootSync with managed namespace

- `sync-x` is created which creates an implicit namespace
- `sync-a` is created which takes ownership of the implicit namespace
- `sync-a` resources are pruned

In this scenario the namespace is explicitly managed by `sync-a`, but there are resources in the namespace that are managed by `sync-x`.

If the `PreventDeletion` annotation is on `namespace-a` (applied by user):

- `namespace-a` is not pruned
- `sync-x` resources are left in-place

If the admission webhook is disabled:

- The namespace and all its resources will be pruned and then recreated implicitly by `sync-x`.

If the admission webhook is enabled:

- The namespace will get the deletionTimestamp and finalizer but the garbage collection controller will error loop with an error from the webhook denying it from deleting some of the contents of the namespace. The namespace will become stuck until the contents are deleted, either by sync-x deleting them or the webhook being disabled and the GC controller deleting them.
- The user can proceed with namespace deletion by pruning resources from `sync-x` (and any other RootSyncs which manage resources in the same namespace).

Open Questions