Using an unstructured repo

Creating an *unstructured repo* allows you to organize your repo arbitrarily, without following the default repo directory structure (/anthos-config-management/docs/concepts/repo#structure_of_the_repo). Anthos Config Management syncs the same set of Kubernetes objects to your cluster as in the default, hierarchical, repo.

This allows you to sync your existing Kubernetes configuration to your Anthos Config Management repo. For example, if you want to sync a Helm (https://helm.sh/) chart to your cluster, you can run `helm template` (https://helm.sh/docs/helm/helm_template/) and commit the rendered manifest to your repo.

Limitations

Unstructured repos do not support all Anthos Config Management features. This section describes the primary differences between hierarchical and unstructured repos.

Unsupported objects

You cannot use the `Repo` and `HierarchyConfig` Kubernetes Objects in an unstructured repo.

Namespace-scoped objects

Abstract namespaces (/anthos-config-management/docs/concepts/namespace-inheritance) are not supported in unstructured repos.

In an unstructured repo, all namespace-scoped objects (/anthos-config-management/docs/how-to/namespace-scoped-objects) should declare either the `metadata.namespace` or the `NamespaceSelector` annotation. Declaring both annotations is invalid. If namespace-scoped resources don't declare `metadata.namespace` or the `NamespaceSelector` annotation, then Anthos Config Management uses the cluster's "default" namespace. See Selectors (#selectors) below for NamespaceSelector semantics.

Selectors

You can declare ClusterSelectors and NamespaceSelectors in an unstructured repo.

In an unstructured repo, ClusterSelectors function normally.

In an unstructured repo, Objects that declare the NamespaceSelector annotation are applied to all namespaces that satisfy the NamespaceSelector's conditions. Before you create an unstructured repo with Objects that were previously used in a hierarchical repo, verify that your NamespaceSelectors do not apply to additional resources.

Configuring an unstructured repo

To configure an unstructured repo, set the value of `spec.sourceFormat` to `unstructured` in `config-management.yaml`.

For example, the `config-management.yaml` below sets up the example Continuous Integration pipeline.

```
fig-management.yaml

rsion: configmanagement.gke.io/v1
 ConfigManagement
ata:
e: config-management

lusterName is required and must be unique among all managed clusters
sterName: my-cluster
:
yncRepo: https://github.com/GoogleCloudPlatform/csp-config-management/
yncBranch: 1.0.0
ecretType: ssh
olicyDir: ci-pipeline-unstructured
rceFormat: unstructured
```

Unstructured mode does not support a repo in the root directory of your git repository. You must declare **`yDir`** to be a directory within your repository. For example, if your git repository on your local machine is at **`fo`**
, you may place your configs in **`foo-corp/policies`** and declare **`policyDir: policies`**.

What's next?

- Create cluster-scoped objects (/anthos-config-management/docs/how-to/cluster-scoped-objects) .
- Create namespace-scoped objects (/anthos-config-management/docs/how-to/namespace-scoped-objects).

See the example unstructured repo in the Continuous Integration pipeline (https://github.com/GoogleCloudPlatform/csp-config-management//tree/1.0.0/ci-pipeline-unstructured).

Last updated 2020-06-22 UTC.