GitOps Flux policy

Last updated by | Ross Wilson | Nov 24, 2022 at 3:55 PM GMT+1

Note: A new policy or initiative assignment takes about 30 minutes to be applied. New or updated resources within scope of an existing assignment become available in about 15 minutes. A standard compliance scan occurs every 24 hours.

Concept

- User creates an AKS cluster
- Azure Policy installs Flux GitOps toolkit
- Azure Policy then creates a Flux configuration
- Flux pulls Helm/Kustomize code from git and applies it
- Helm/Kustiomize code contains governance components

Implementation

Azure Policy

We use Azure Policy to implement a policy with a `denyIfNotExist` effect, which targets all `Microsoft.ContainerService/managedClusters` resources i.e. AKS clusters. The policy must deploy two resources:

1. a `Microsoft.KubernetesConfiguration/extensions` resource; this will deploy the `Microsoft.Flux` extension i.e. GitOps Flux, onto the AKS cluster

```
{
        "type": "Microsoft.KubernetesConfiguration/extensions",
        "apiVersion": "2021-09-01",
        "name": "fluxExtension",
        "location": "[parameters('clusterRegion')]",
        "identity": {
                "type": "SystemAssigned"
        },
        "properties": {
                "extensionType": "Microsoft.Flux",
                "autoUpgradeMinorVersion": true
        },
        "scope": "[concat('Microsoft.ContainerService/managedClusters/', split(parameters('clusterResourceI
},
```

2. a `Microsoft.KubernetesConfiguration/fluxConfigurations` resource; this will deploy the config which causes the AKS cluster to deploy the mandatory resources which we define in the repo this config points to
   - notice the `httpsKey` value which must be a base64-encoded string of the PAT to be used for pulling from Azure DevOps git over HTTPS

```
{
        "type": "Microsoft.KubernetesConfiguration/fluxConfigurations",
        "apiVersion": "2022-03-01",
        "name": "baseline-configuration",
        "scope": "[concat('Microsoft.ContainerService/managedClusters/', split(parameters('clusterResourceI
        "properties": {
                "configurationProtectedSettings": {
                        "httpsKey": "base64-encoded string"
                },
                "gitRepository": {
                        "httpsUser": "git",
                        "repositoryRef": {
                                "branch": "main"
                        },
                        "syncIntervalInSeconds": 600,
                        "timeoutInSeconds": 600,
                        "url": "https://mcp-abnamro-stratus@dev.azure.com/mcp-abnamro-stratus/Managed%20Cor
                },
                "kustomizations": {
                        "baseline": {
                                "path": "flux/aks-baseline",
                                "dependsOn": [],
                                "timeoutInSeconds": 600,
                                "syncIntervalInSeconds": 600,
                                "validation": "none",
                                "prune": true
                        }
                },
                "scope": "cluster",
                "sourceKind": "GitRepository"
        },
        "dependsOn": [
                "[extensionResourceId(concat('Microsoft.ContainerService/managedClusters/', split(parameter
        ]
}
```

Tips

- pay attention to the top-level `scope` and `dependsOn` values for both resources
  - the `Microsoft.KubernetesConfiguration/extensions` is scoped to the cluster and has no dependencies within our `depolyIfNotExist` deployment
  - the `Microsoft.KubernetesConfiguration/fluxConfigurations` is *also* scoped to the cluster, but it is dependent on the previously defined `Microsoft.KubernetesConfiguration/extensions` (you can't configure Flux if it's not installed)
- the `httpsKey` value must be an Azure DevOps PAT, base64 encoded ; when using the `base64` command you must remember to remove the trailing newline i.e. use `base64 -n`
- it `prune` is not set to `true` , any resources we remove from the governance repo will **not** be removed from the AKS clusters!

Custom RBAC Role

The Managed Identity of the Azure Policy Assignment is the identity used to depoly the DINE resources. By default it has little to no permissions, so we need to grant it the permissions it needs.

We can grant these with two roles:

1. a built-in role called `Kubernetes Extension Contributor` , which allows installation of the GitOps Flux extension
2. a custom role, which we can call `Flux Extension Contributor` , which allows creation of the Flux configurations

  - notice the fixed value for `assignableScopes` which needs to be adjusted for actual deployment

JSON Role Definition

```
{
        "id": "/subscriptions/0a1e54a3-58b3-40f8-82bd-efd7844068a5/providers/Microsoft.Authorization/roleDe
        "properties": {
                "roleName": "Flux Configuration Contributor",
                "description": "",
                "assignableScopes": [
                        "/subscriptions/0a1e54a3-58b3-40f8-82bd-efd7844068a5/resourceGroups/mcp-e-rg"
                ],
                "permissions": [
                        {
                        "actions": [
                                "Microsoft.KubernetesConfiguration/fluxConfigurations/write",
                                "Microsoft.KubernetesConfiguration/fluxConfigurations/delete",
                                "Microsoft.KubernetesConfiguration/fluxConfigurations/operations/read",
                                "Microsoft.KubernetesConfiguration/fluxConfigurations/read"
                        ],
                        "notActions": [],
                        "dataActions": [],
                        "notDataActions": []
                        }
                ]
        }
}
```

Setup

1. deploy the Azure Policy
2. deploy the custom Azure RBAC Role
3. grant the two required roles
   - If using a User-assigned Managed Identity, this need only be assigned once; otherwise this needs to be assigned everytime the Azure Policy or Policy Assignment is updated!