EMCO: Synchronizing Resources with Target Clusters via Git
GitOps Introduction

• GitOps is a set of practices to manage infrastructure and application configurations using Git, an open-source version control system.
• GitOps uses Git pull requests to automatically manage infrastructure provisioning and deployment.
• The Git repository contains the entire state of the system so that the trail of changes to the system state are visible and auditable.
• An agent is running inside environment continually, polls Git repo and/or container registry for changes.
• When it detects a mismatch between the defined state and the running state, the agent pulls the defined configuration into the environment.
• No inbound connections to the cluster.
Advantages

• Reduced security and compliance risk.
  – Because the CD agent is running inside of the cluster, there’s no need to store credentials externally.
  – Reduce or eliminate the holes in the firewall that allow inbound connections.

• Consistency
  – Agent polls Git repo and container registry for changes and compare the state of the cluster to the defined state in Git.
  – This can detect and remediate configuration drift if changes are made to the cluster manually or from other sources
GitOps Flow
GitOps Support

- GitOps pattern is being adopted in many public and private clouds.
- Flux CD and Argo CD, are two Kubernetes-native applications that facilitate and help enforce GitOps pattern.
- Azure supports GitOps on an Azure Arc-enabled Kubernetes cluster.
- GoogleCloud support GitOps with Anthos.
- EMCO plans to support various vendors and technologies with its extensible design.
Intro to Azure ARC

• Azure Arc Simplifies complex and distributed environments across on-premise’s, edge and multi-cloud

• It manages your entire environment, with a single pane of glass, by projecting your existing non-Azure, on-premises, or other-cloud resources into Azure Resource Manager

• It manages virtual machines, Kubernetes clusters and databases as if they are running in Azure
Azure Arc With GitOps

Azure Arc enabled Kubernetes GitOps Flow

1. Google Kubernetes Engine (GKE)
2. Arc Connected Kubernetes Cluster
3. GitOps Configurations
4. Flux Operator + Helm Operator
5. Application Deployment
6. Application V2 (New Desired State)
7. git merge
8. Application Changes

Git Repository

Any Kubernetes, any Infrastructure

Application Rolling Update

flux pickup changes
Fluxv2 Architecture
Flux v2 Flow

1a. Pushes change to Git app repo → Triggers

2. Docker Build → Pushes Docker image to Package Repository

1b. Pushes change to Git deploy repo → Updates

4. Flux controller watches Git deploy repo

5. Workload

0. Runs up Git infra repo

https://zwischenzugs.com/2021/07/31/a-hello-world-gitops-example-walkthrough/
EMCO Compliments GitOps

Additional requirements for Multi-cluster deployments not fulfilled by GitOps

- On-demand instantiation of applications on K8s clusters
- Intelligent selection of clusters to place the workloads
- On-demand scale-out (bursting) of the applications
- Customization of resources to the applications
- Automation of service mesh and other connectivity & security infrastructure
- Dependency and order of priority of application deployments between clusters
EMCO with GitOps
Rsync Plugin Framework

• Plugin selected based on the type of support in a cluster: direct access, Azure Arc cluster, FluxCD based, Google Anthos etc.

• The type of support available in cluster is provided at the time of cluster registration.

• Interfaces identified in Rsync for the Plugins
  – Resources Provider
  – Resources Reference Provider
  – Status Provider
Plugin Support

• Plans to support Plugins for
  – Azure Arc
  – Google Anthos
  – Fluxv2
  – ArgoCD

• Any others …
Rsync Plugin Framework

• Resources Provider interface - For actual Resources
  – *Methods* - Create, Apply, Delete, Get, Commit, IsReachable
  – *Examples*: K8s API, Git location

• Resources Reference interface - For configuration specific to cluster type
  – *Methods* – ApplyConfig and DeleteConfig
  – *Examples*: Azure Config, Flux CD CR’s (For ex: GitRepository, Kcustomize, etc.)

• Status Provider – For status handling
  – *Methods* – StartClusterWatcher, ApplyStatusCR, DeleteStatusCR
App Instantiation with EMCO

GitHub/Gitlab
Clusters/cluster1/context/100 – flux.yaml
Clusters/cluster1/context/100/app1 – service.yaml
Clusters/cluster1/context/100/app1 – pod.yaml

Azure Arc

1. Register clusters
Kubeconfig for cluster1
Git credentials for cluster2
Git & Azure credentials for cluster2

2. Instantiate app1 on cluster1, 2, 3

3. Apply Resources

4. Apply Azure config for cluster3
4.a Store Fluxv2 CR in git for cluster2 &
4.b Store Resources in git for cluster2 & cluster3

5. Pull resources
5a Configure Flux
5b Pull Resources

4. Apply Resources

Flux source controller
Flux Kustomize controller

Monitor

API Server

Rsync

CLM

Orchestrator

Mongo DB

API Server

Flux source controller
Flux Kustomize controller

Monitor

API Server

Flux source controller
Flux Kustomize controller

Monitor

API Server

Flux source controller
Flux Kustomize controller
App Monitoring with EMCO (WIP)

GitHub/Gitlab
Clusters/cluster2/status/100-ap1.yaml
Clusters/cluster2/status/100-app2.yaml
Clusters/cluster2/context/200-app1.yaml
Clusters/cluster2/context/200-app2.yaml

Control Git
clusters/cluster2/status/100-ap1.yaml
Clusters/cluster2/status/100-app2.yaml
Clusters/cluster2/context/200-app1.yaml
Clusters/cluster2/context/200-app2.yaml

Monitor
cluster1
Kubernetes Cluster
API Server
Status CR

Mongo DB
Rsync
CLM
Orchestrator

Monitor
cluster2
Kubernetes Cluster (GitOps Managed Cluster)
API Server
Status CR
Flux source controller
Flux kustomize controller

Push Status CR
Watch cluster
Pull Status CR

Notifications for CR change
Apply Status CR
Watch cluster
Store StatusCR for cluster2
Backup
Demo Azure GitOps Configuration

1) Example Git repo: https://github.com/Azure/arc-k8s-demo
2) The manifests in this repo provisions a few namespaces, deploy workloads and some team-specific configurations
3) K8s-configuration extension for Azure CLI is used.
4) az k8s-configuration create
   --name cluster-config
   --cluster-name AzureArcTest1
   --resource-group AzureArcTest
   --operator-instance-name cluster-config
   --operator-namespace cluster-config
   --repository-url https://github.com/Azure/arc-k8s-demo
   --scope cluster
   --cluster-type connectedClusters

https://docs.microsoft.com/en-us/azure/azure-arc/kubernetes/tutorial-use-gitops-connected-cluster#next-steps
Fluxv2 CR Example

- **GitRepository**: Provides details about the Git location to synchronize, branch and Repo

- **Kustomization**: Defines the source of Kubernetes manifests by referencing an object managed by source-controller, the path to the Kustomization file within that source, and the interval at which the kustomize build output is applied on the cluster

```yaml
apiVersion: source.toolkit.fluxcd.io/v1beta1
kind: GitRepository
metadata:
  name: app1
spec:
  interval: 30s
  ref:
    branch: main
    url: https://github.com/xxx/repo1

apiVersion: kustomize.toolkit.fluxcd.io/v1beta2
kind: Kustomization
metadata:
  name: kustapp1
  namespace: default
spec:
  interval: 5m0s
  path: ./clusters/10309/context/100
  prune: true
  sourceRef:
    kind: GitRepository
    name: app1
  targetNamespace: default
```
1) Config agent tracks new or updated configuration resources.
2) Deploys a Flux operator to watch the Git repo for each config resource.
3) Apply updates made to any configuration resource.