LF NETWORKING

LFN Developer & Testing Forum

LF NETWORKING

LFN Developer & Testing Forum

EMCO: Synchronizing Resources with Target Clusters via Git

EMCO & GitOps

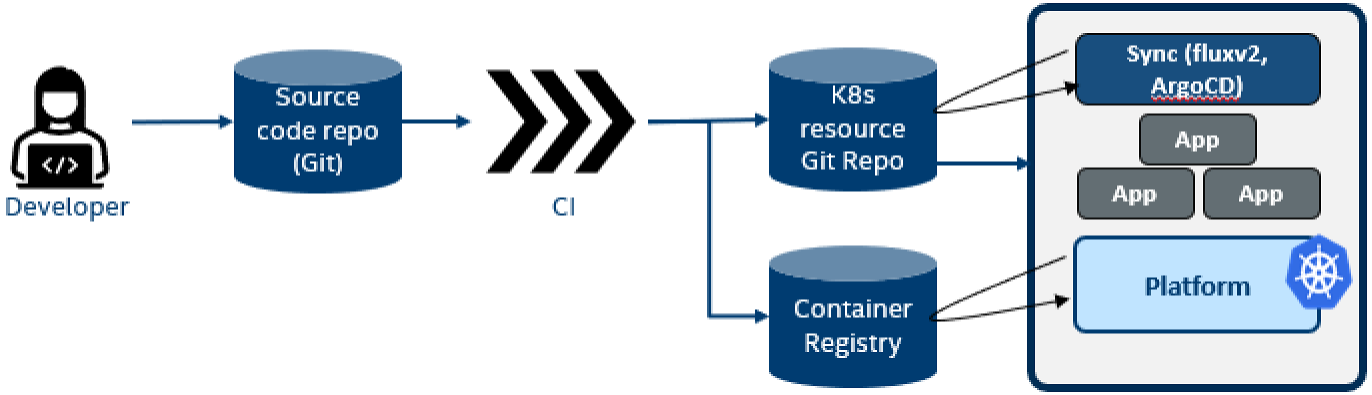
GitOps Introduction

- 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


Developer
Source code repo (Git)
CI
K8s resource Git Repo
Container Registry
Sync (fluxv2, ArgoCD)
App
App
App
Platform

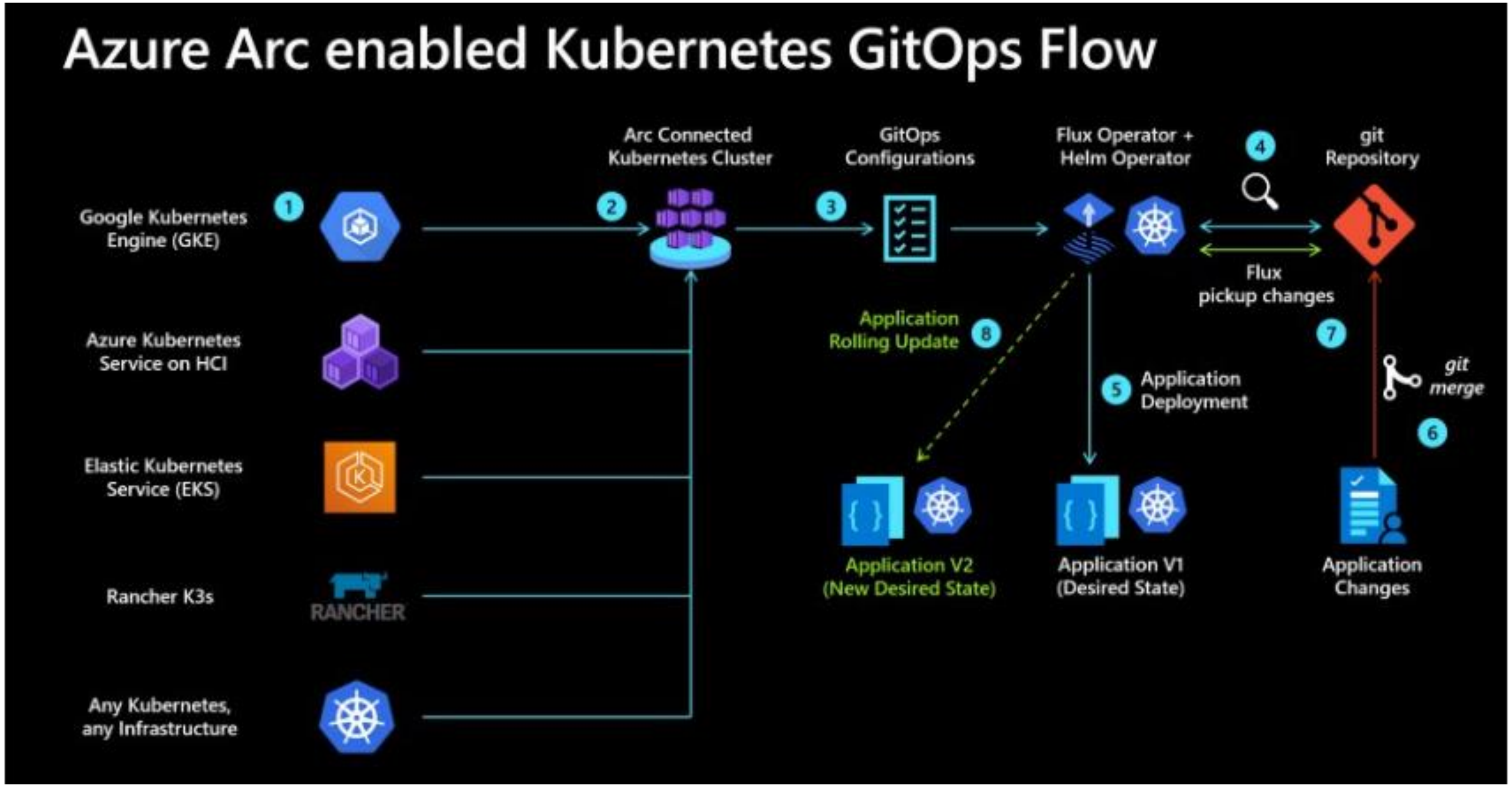
GitOps Support

- 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.

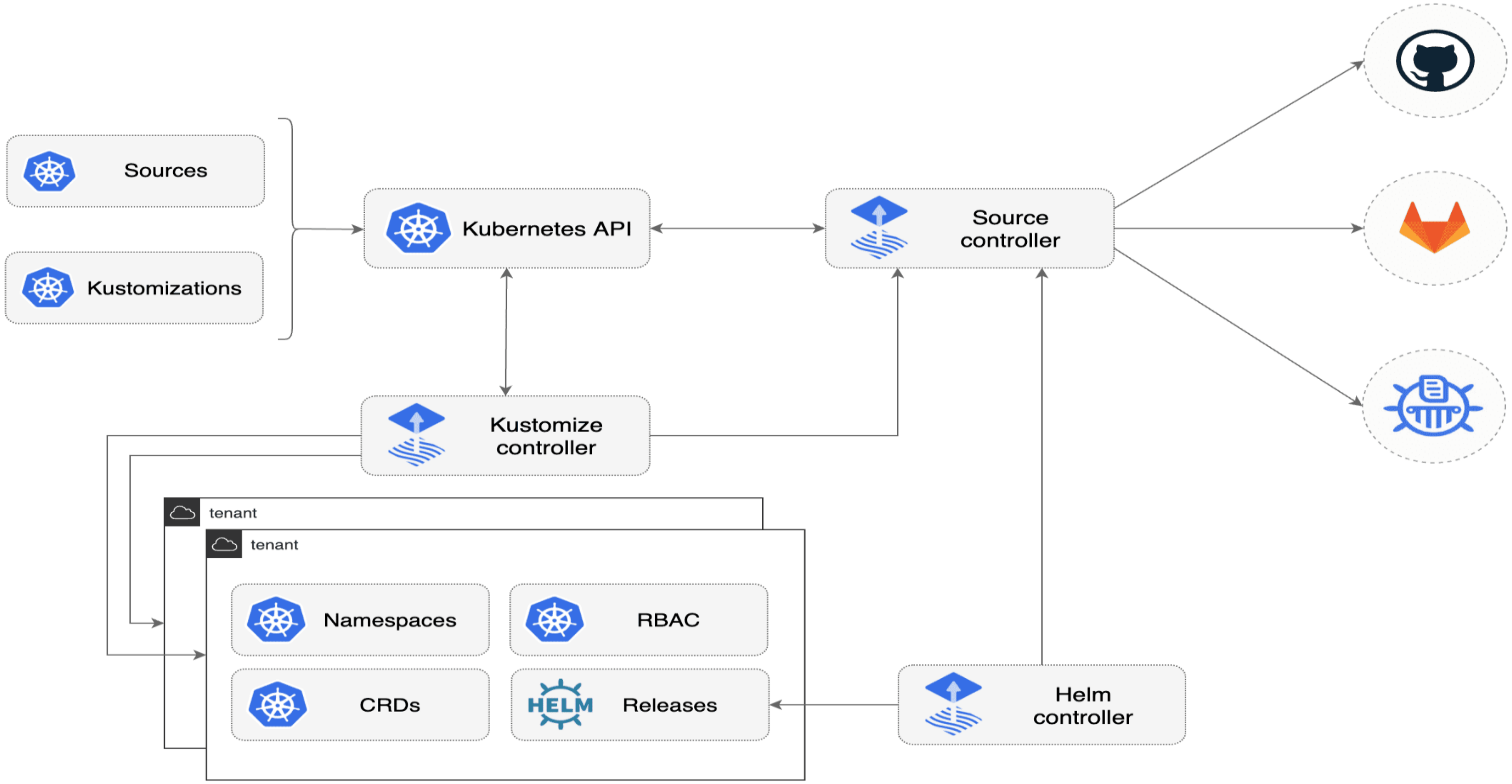
Intro to Azure ARC

- 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

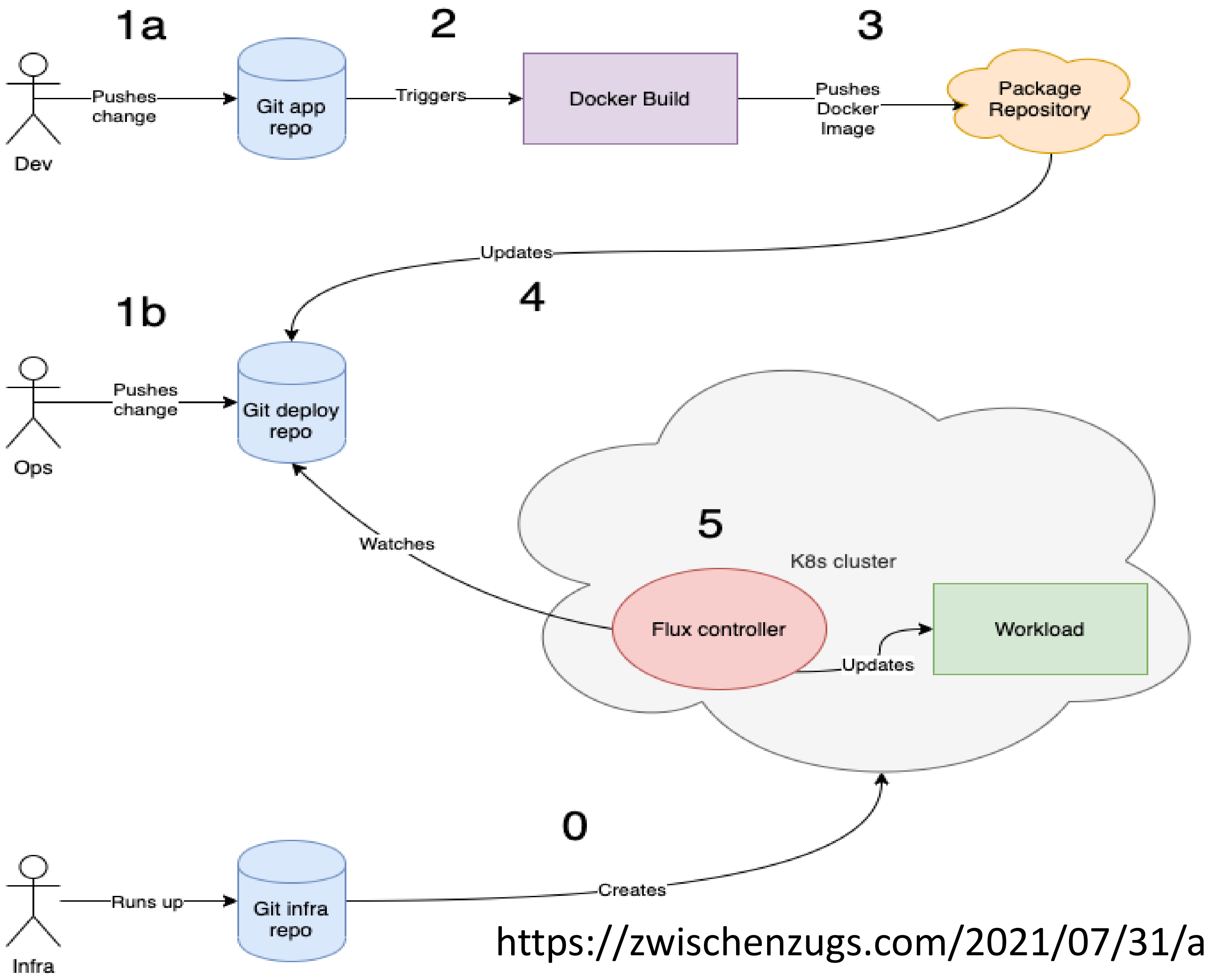
Azure Arc With GitOps


Azure Arc enabled Kubernetes GitOps Flow
Google Kubernetes Engine (GKE)
Azure Kubernetes Service on HCI
Elastic Kubernetes Service (EKS)
Rancher K3s
RANCHER
Any Kubernetes, any Infrastructure
1
2
Arc Connected Kubernetes Cluster
3
GitOps Configurations
Flux Operator + Helm Operator
4
git Repository
Flux pickup changes
7
Application Rolling Update
8
5
Application Deployment
git merge
6
Application V2 (New Desired State)
Application V1 (Desired State)
Application Changes

Fluxv2 Architecture



Flux v2 Flow



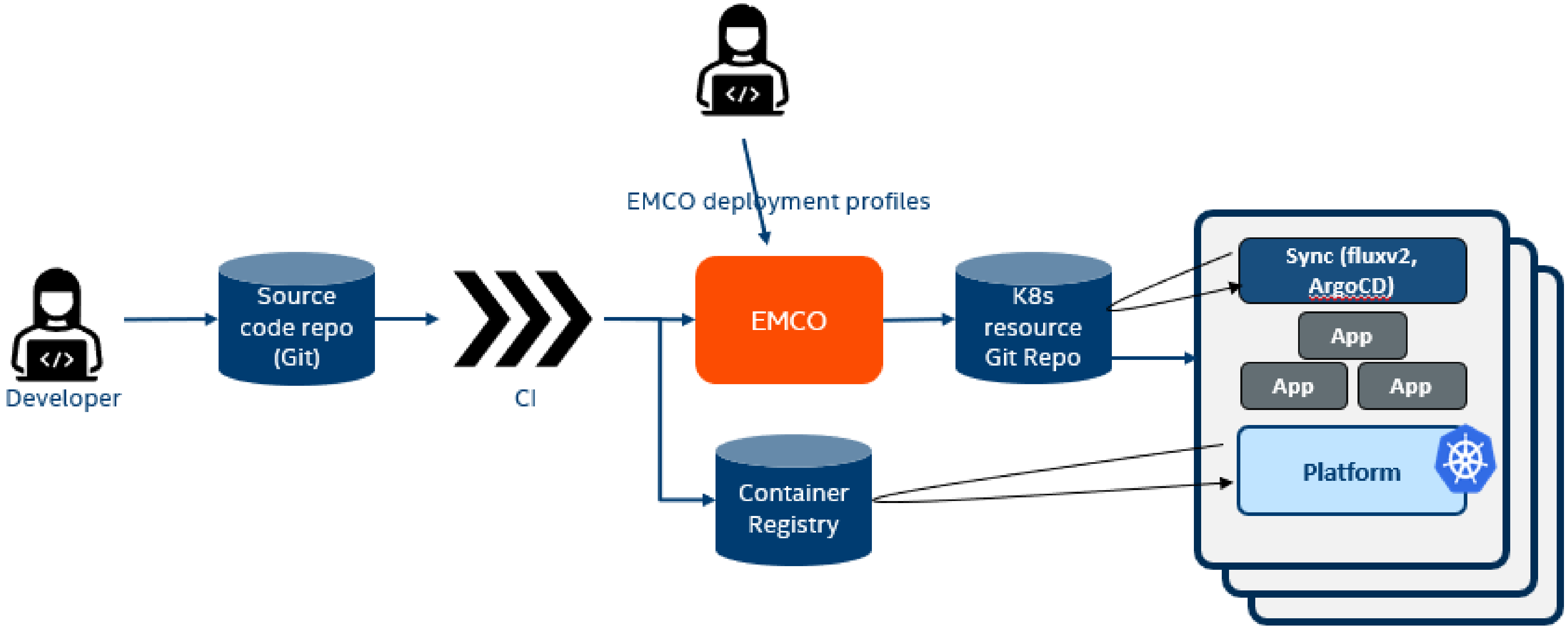
https://zwischenzugs.com/2021/07/31/a-hello-world-gitops-example-walkthrough/

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 …

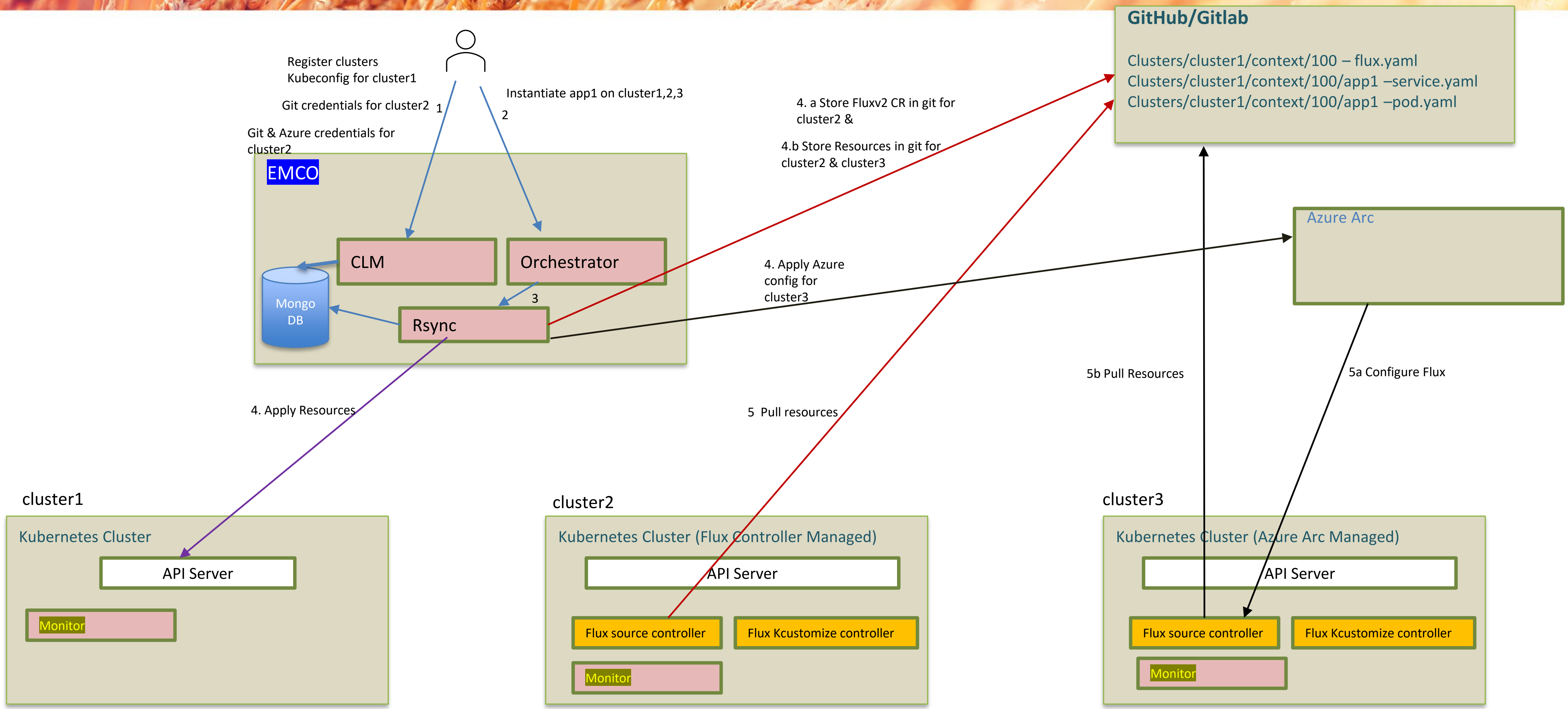
Rsync Plugin Framework

- 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

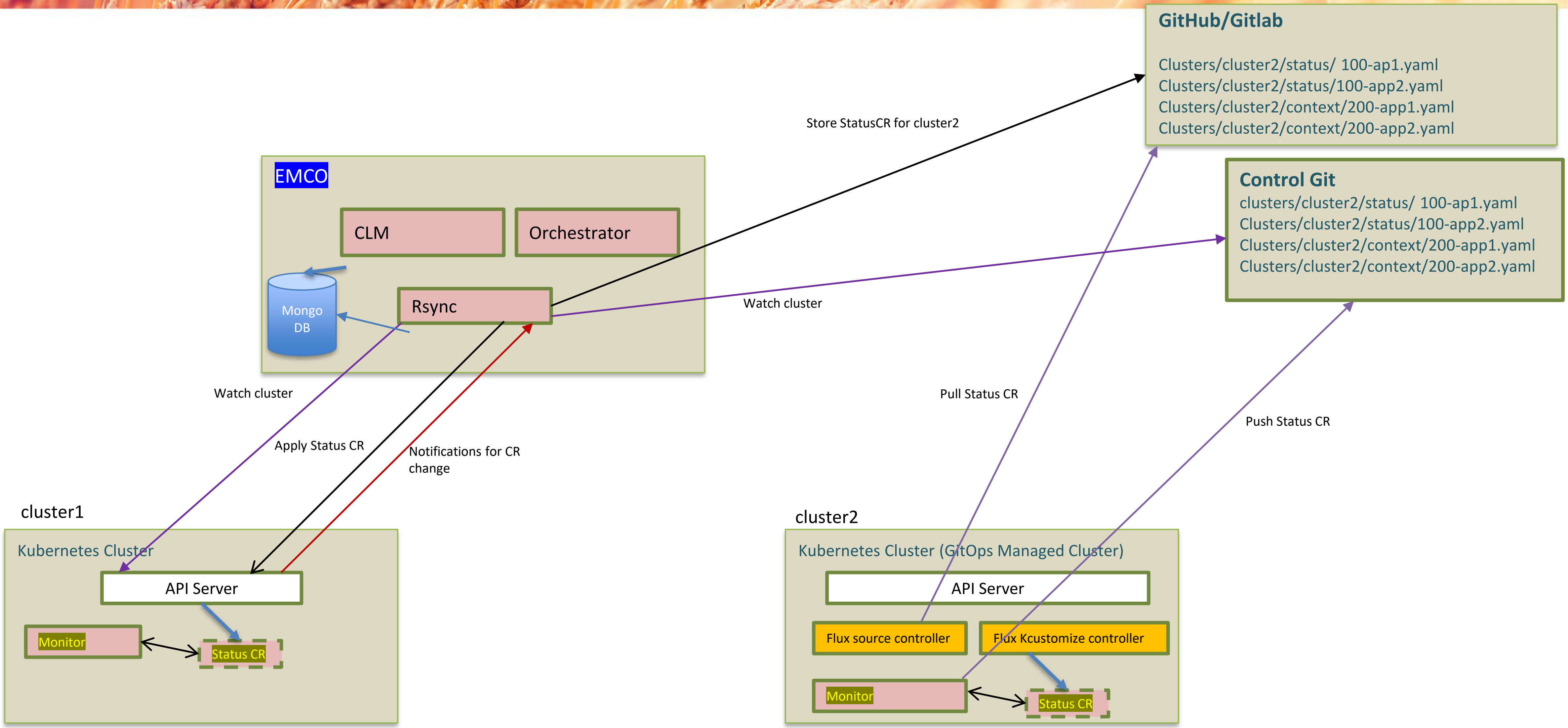
App Instantiation with EMCO


GitHub/Gitlab
Clusters/cluster1/context/100 – flux.yaml
Clusters/cluster1/context/100/app1 –service.yaml
Clusters/cluster1/context/100/app1 –pod.yaml
Register clusters
Kubeconfig for cluster1
Git credentials for cluster2
1
Git & Azure credentials for cluster2
Instantiate app1 on cluster1,2,3
2
4. a Store Fluxv2 CR in git for cluster2 &
4.b Store Resources in git for cluster2 & cluster3
EMCO
CLM
Orchestrator
Mongo DB
3
Rsync
Azure Arc
4. Apply Azure config for cluster3
5b Pull Resources
5a Configure Flux
4. Apply Resources
5 Pull resources
cluster1
Kubernetes Cluster
API Server
Monitor
cluster2
Kubernetes Cluster (Flux Controller Managed)
API Server
Flux source controller
Flux Kcustomize controller
Monitor
cluster3
Kubernetes Cluster (Azure Arc Managed)
API Server
Flux source controller
Flux Kcustomize controller
Monitor

App Monitoring with EMCO (WIP)

LF NETWORKING

LFN Developer & Testing Forum



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
Store StatusCR for cluster2
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
EMCO
CLM
Orchestrator
Mongo DB
Rsync
Watch cluster
Watch cluster
Apply Status CR
Notifications for CR change
Pull Status CR
Push Status CR
cluster1
Kubernetes Cluster
API Server
Monitor
Status CR
cluster2
Kubernetes Cluster (GitOps Managed Cluster)
API Server
Flux source controller
Flux Kcustomize controller
Monitor
Status CR

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

LF NETWORKING

LFN Developer & Testing Forum

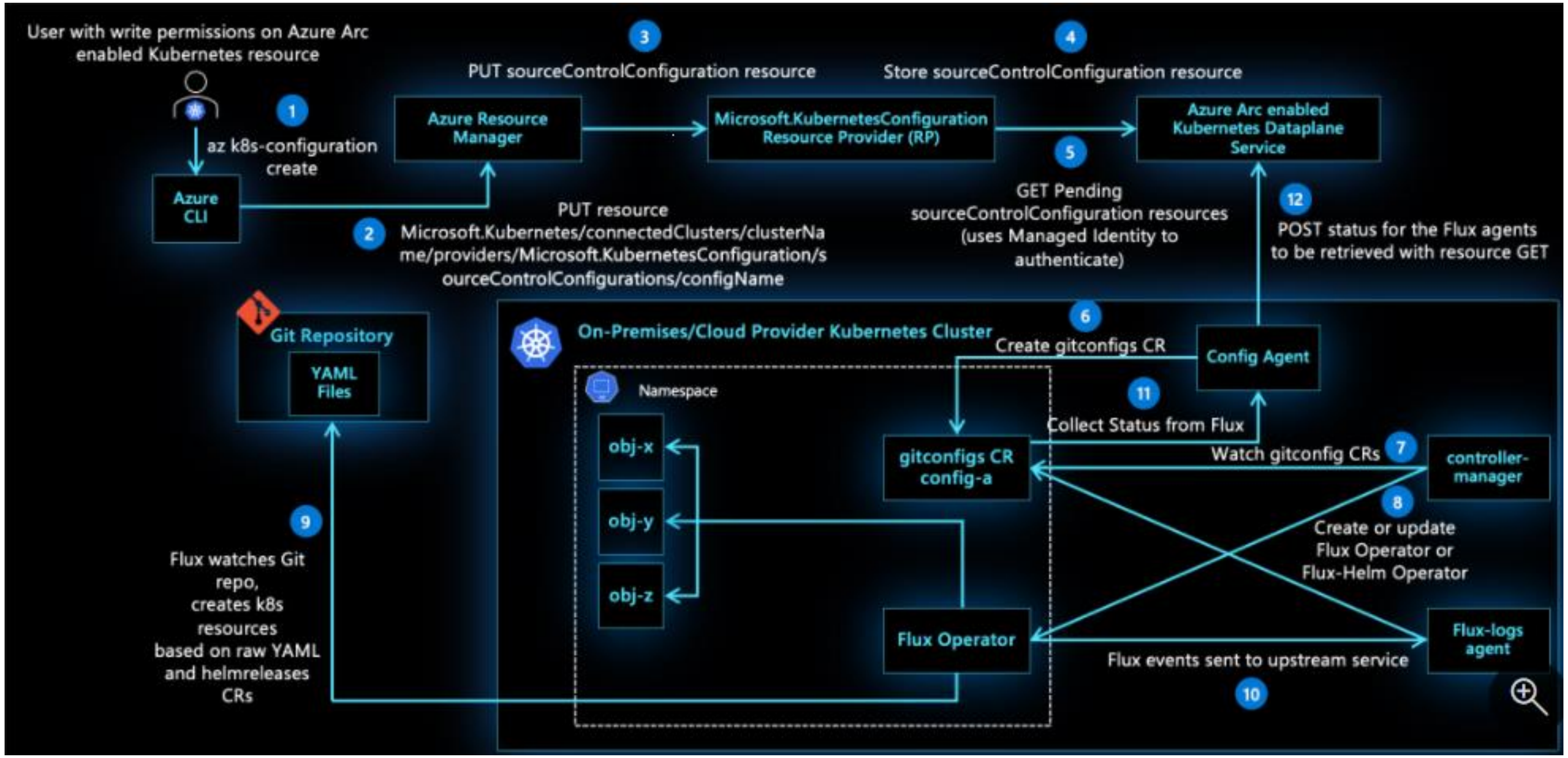
- **GitRepository**: Provides details about the Git location to synchronize, branch and Repo

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

- **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

```
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
```

Configurations and GitOps with ARC


User with write permissions on Azure Arc enabled Kubernetes resource
1
az k8s-configuration create
Azure CLI
2
PUT resource Microsoft.Kubernetes/connectedClusters/clusterName/providers/Microsoft.KubernetesConfiguration/sourceControlConfigurations/configName
Azure Resource Manager
3
PUT sourceControlConfiguration resource
Microsoft.KubernetesConfiguration Resource Provider (RP)
4
Store sourceControlConfiguration resource
Azure Arc enabled Kubernetes Dataplane Service
5
GET Pending sourceControlConfiguration resources (uses Managed Identity to authenticate)
12
POST status for the Flux agents to be retrieved with resource GET
Git Repository
YAML Files
On-Premises/Cloud Provider Kubernetes Cluster
6
Create gitconfigs CR
Config Agent
Namespace
obj-x
obj-y
obj-z
11
Collect Status from Flux
gitconfigs CR config-a
7
Watch gitconfig CRs
controller-manager
8
Create or update Flux Operator or Flux-Helm Operator
Flux Operator
Flux-logs agent
Flux events sent to upstream service
10
9
Flux watches Git repo, creates k8s resources based on raw YAML and helmreleases CRs

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.