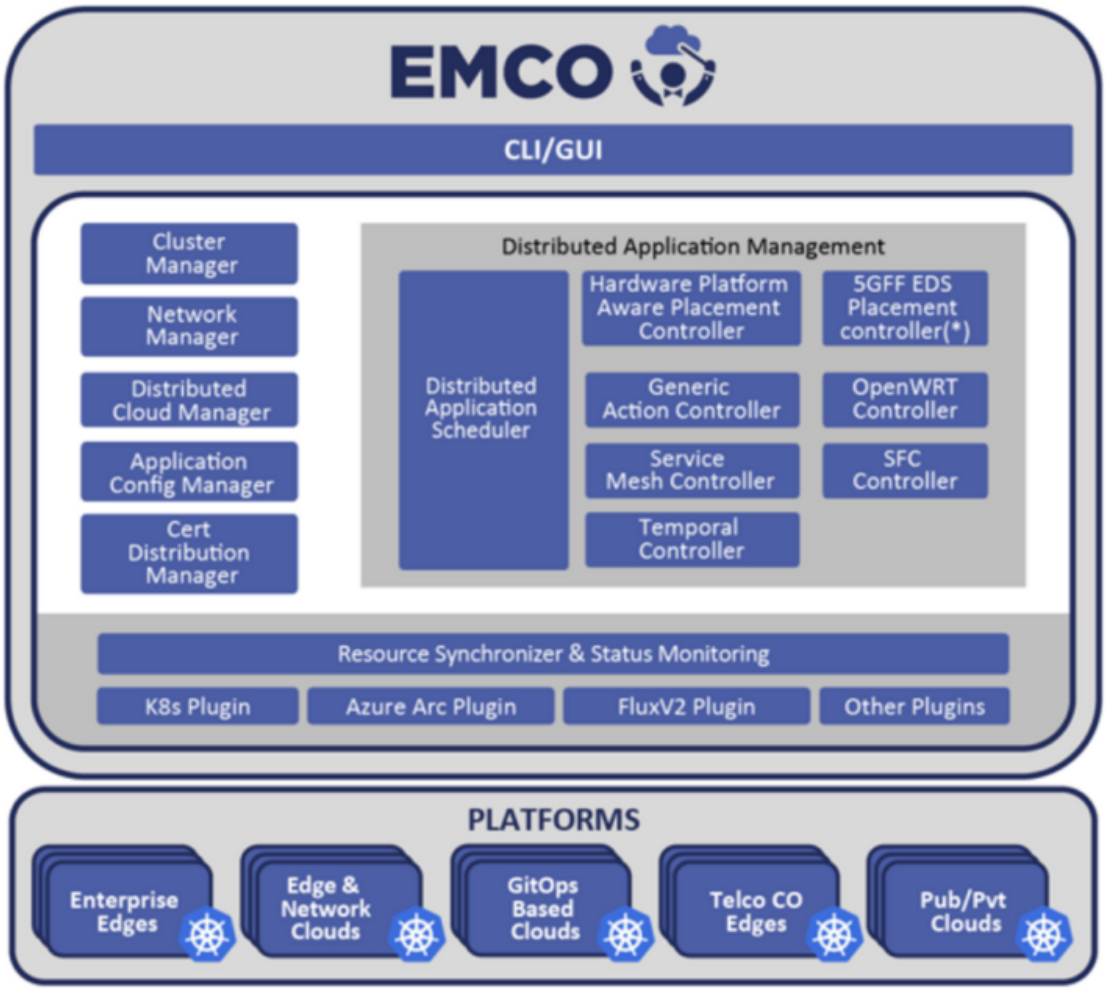
Simplifying Edge deployments using EMCO and GitOps

Edge computing is becoming increasingly important. In this blog, we would like to explain how Edge deployments can be simplified using EMCO and GitOps.



What is EMCO and its significance in the Edge cloud sector?

The Edge Multi-Cluster Orchestrator (EMCO) is an open-source project, part of the Linux Foundation Networking umbrella backed by companies including Intel, Equinix, Nokia, Aarna Networks, and others. It is a software framework for the intent-based deployment of cloud-native applications to a set of Kubernetes clusters, spanning enterprise data centers, multiple cloud service providers, and numerous edge locations. It can be leveraged for Private 5G, O-RAN (Open Radio Access Networks), and multi-access edge computing (MEC) applications. It has many unique features like extensibility via an intelligent selection of clusters to place the workloads, tenant Isolation using logical clouds, automation of service mesh and other connectivity & security infrastructure, updates, and rollbacks.

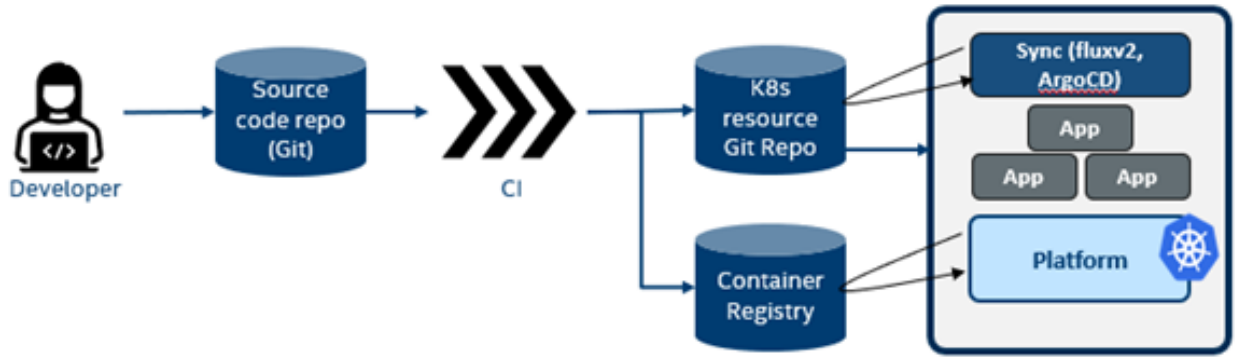
In brief, we can say it's a one-stop solution for edge deployments using Kubernetes, and what makes it better is being open-sourced, it is completely free.

What is GitOps and what are its advantages?

In simple words, GitOps is a set of practices to manage infrastructure and application configurations using the Git version control system. It seeks to bring the advantages of git together with what are now becoming more standard tools and concepts to the DevOps and Automation world like Infrastructure as Code, Merge Requests/Pull Requests, and Continuous Integration and Delivery/Deployment.

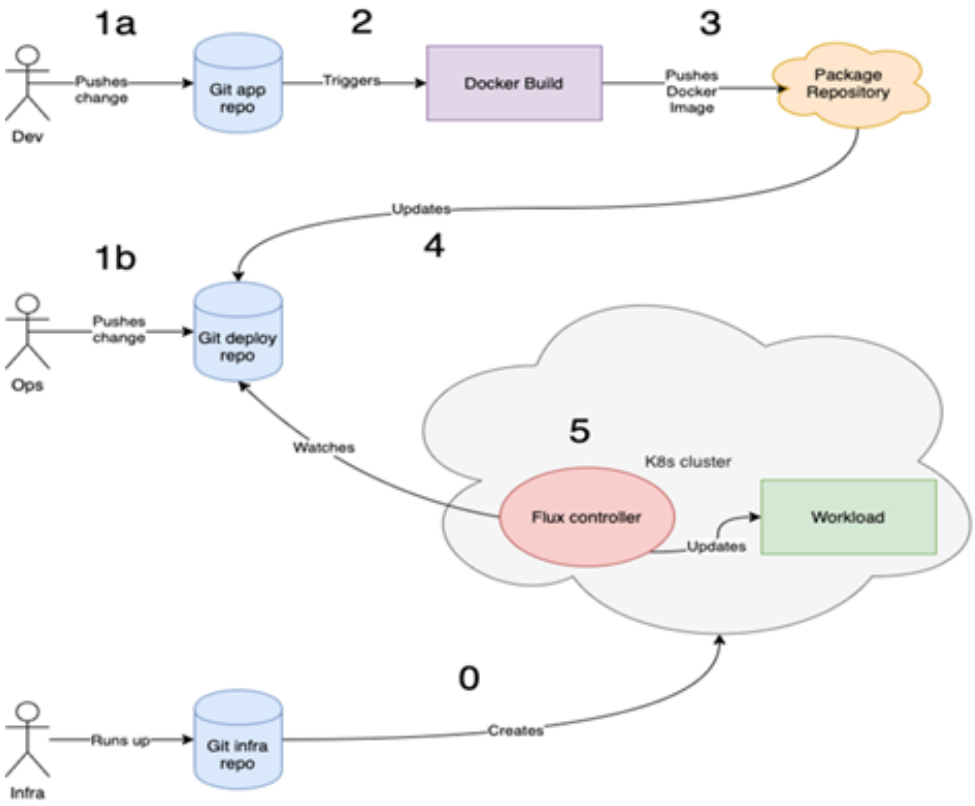
To elaborate, in a basic GitOps flow the user pushes all the resources to the git repo considered to contain the entire state of the system. And usually, an agent like FluxCD or ArgoCD running inside the environment continuously polls this Git repo and/or container registry for changes. As soon as it detects a mismatch between the defined state and the running state, the agent pulls the defined configuration into the environment. A few major advantages of using this mode of deployment are

- Reduced security and compliance risk: Because the agent is running inside the cluster, there is no need to store credentials externally.
- Consistency: Due to the poll and compare mechanism followed by the agents it can detect and remediate configuration drift if changes are made to the cluster manually.



GitOps flow

The following figure shows a typical GitOps flow.



Let us try to understand this GitOps flow, which has three actors: a developer (dev), Operations Engineer (Ops), and an Infrastructure Engineer (Infra). Let us cycle through steps 0-5 and understand what actions are being taken by these three actors.

- In step 0, the infrastructure engineer creates the Kubernetes cluster with a GitOps controller like FluxCD/ArgoCD. This GitOps controller is watching the Git Deploy Repository for changes to synchronize.

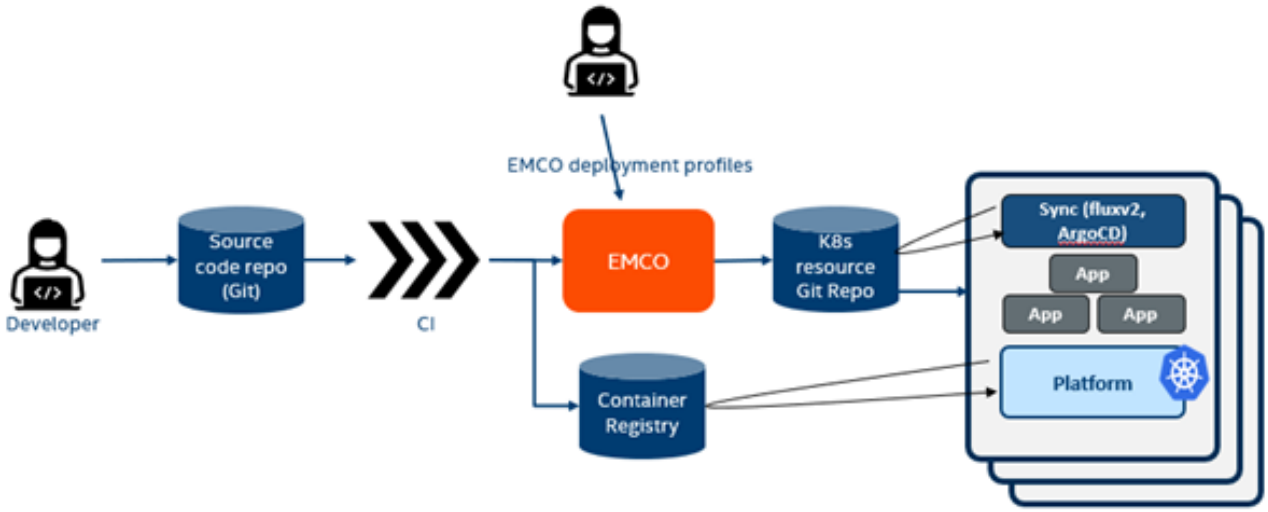
- In step 1a, the developer pushes the application changes to the Git App Repository, which triggers step 2 docker build. Once the build gets completed, step 3 kicks in, and the images get pushed to a package repository.
- After this, step 4 gets triggered and updates the Git Deploy repository with these new images.
- As mentioned earlier, the GitOps controller deployed in the Kubernetes continuously watches the Git Deploy repository for changes. As soon as the new images get updated, the GitOps controller picks up the changes as shown in step 5 and updates the workload deployed in the Kubernetes cluster.

Now each time the developer pushes in changes these steps are repeated, thus ensuring that the deployments in the cluster always remain in sync with the latest updates published in the Git repository.

How well does EMCO complement GitOps?

In the context of a multi-cluster, multi-app edge/cloud deployment, such deployments, which involve deploying multiple applications on different cluster types located in different environments, add complexity to management, security, and consistency. GitOps simplifies the process but lacks the intelligence to solve these complexities. So, EMCO can become the intelligent agent that writes resources to the git location to solve these complexities and accomplish all the below-mentioned points.

- On-demand instantiation of applications on K8s clusters
- Workload placement using intelligent cluster selection.
- 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



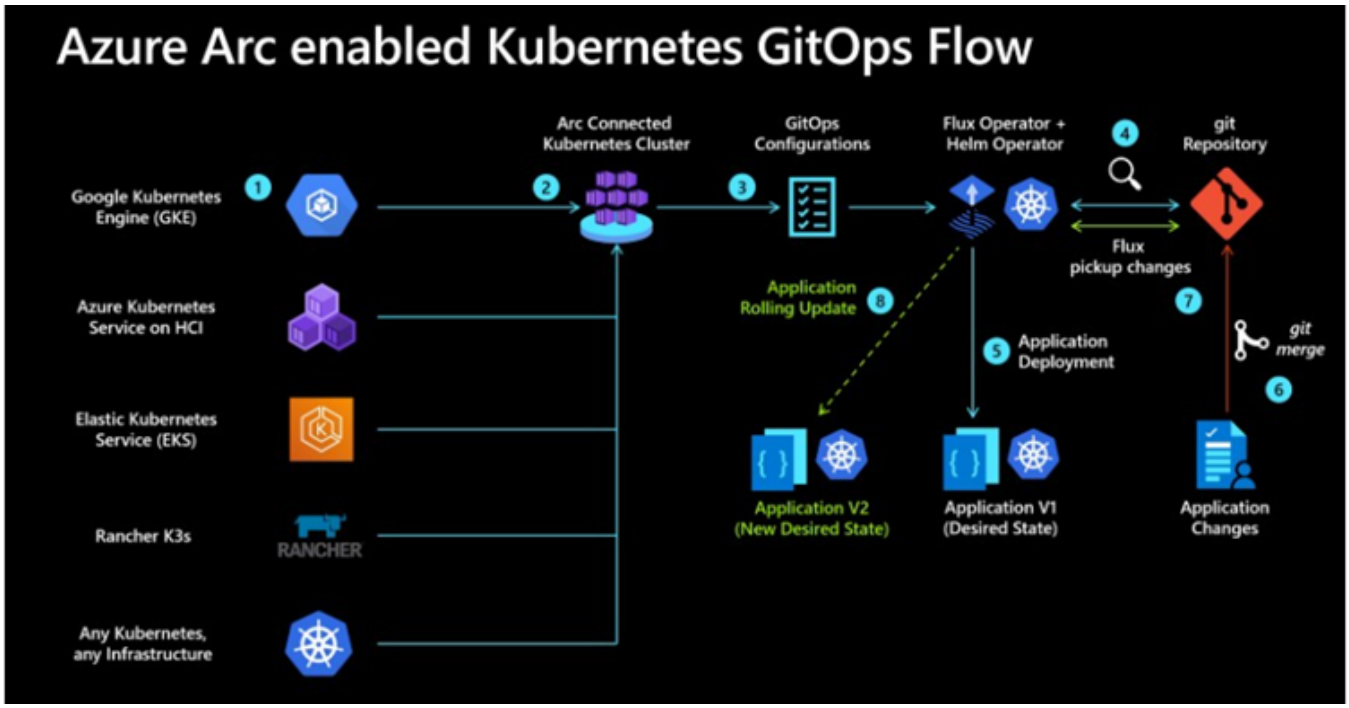
What GitOps Plugins are available in EMCO?

GitOps pattern is being adopted in many public and private clouds. FluxCD and ArgoCD are two Kubernetes-native applications that facilitate and help enforce the GitOps pattern. Azure supports GitOps on an Azure Arc-enabled Kubernetes cluster, and GCP (Google Cloud Platform) supports GitOps with Anthos.

EMCO provides GitOps plugins for

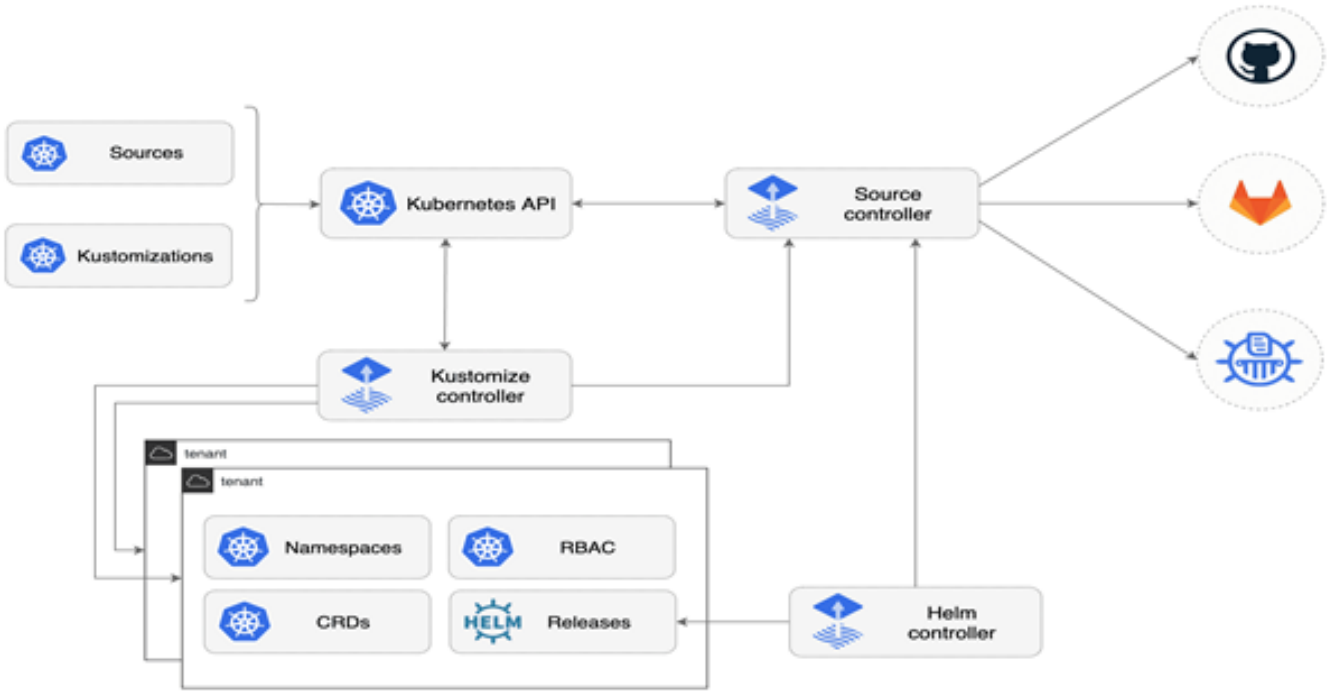
- Azure Arc-enabled Kubernetes clusters

  Azure Arc-enabled Kubernetes allows you to attach and configure Kubernetes clusters running anywhere.



- FluxCD

  FluxCD is a set of continuous and progressive delivery solutions for Kubernetes that are open and extensible.



- Google Anthos, and plans to support various other vendors and technologies with its extensible design.

Please feel free to read more about Azure-Arc enabled Kubernetes Clusters and FluxCD

Demo examples

We have a few demo examples to demonstrate how to use GitOps with Azure Arc-enabled clusters, clusters with standalone FluxCD controllers, and Google Anthos using EMCO as the intelligent agent. Please feel free to try it.

- Azure-Arc GitOps Demo using EMCO
- FluxCD GitOps Demo using EMCO
- Google Anthos GitOps Demo using EMCO