**Continuously Deliver Kubernetes Applications with Flux**

Flux is an Open and extensible continuous delivery solution for Kubernetes. Flux is a GitOps tool for Kubernetes that synchronizes the state of manifests in a Git repository to what is running in a cluster.

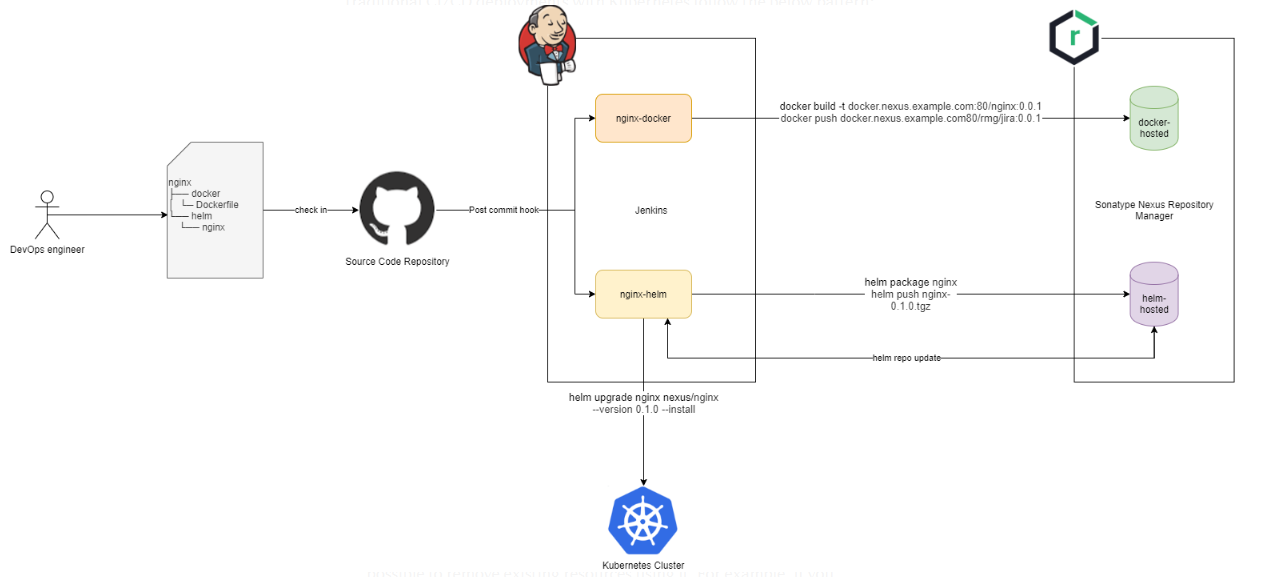
GitOps provides a way for developers to manage operational workflow for using Kubernetes using Git. It is all about using a version-controlled system for the deployment of applications in Kubernetes.

## Features of Flux:

1. Automated synchronization between a version control repository and a cluster.
2. Any changes made to the repository are instantly reflected in the cluster.
3. Developers can directly push the code into production from the repositories.
4. All the configuration is stored in the version control system and is up to date.
5. Built-in support for kustomize and helm.
6. It can also be integrated with [**flagger**](https://flagger.app/).
7. In case of a disaster, the new cluster can be brought up with the same configuration.

# Why Flux CD?

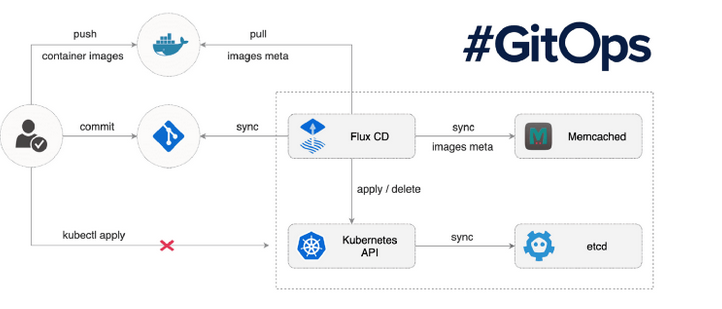
Traditional CI/CD deployments with Kubernetes follow the below pattern:



# How Flux CD Works

Flux CD allows teams to specify all required Kubernetes configurations using YAML manifests declaratively.

* Teams write the Kubernetes manifests and push them to a source code repository.
* The memcached pod stores the current configuration.
* Flux periodically (default five minutes) polls the repository using a Kubernetes operator for changes. The Flux container compares that with the existing configuration in memcached.
* If it detects changes, it syncs the configuration with the cluster by running a series of kubectl apply/delete commands. It then stores the latest metadata on the memcached store again.



Also, Flux CD allows you to poll container registries and update the Kubernetes manifests on your Git repository with the latest images if you want to automate upgrades to your workloads.

As Flux CD runs as a Kubernetes operator, it is simple to set up and great to start.

Let’s look into a hands-on demonstration so that we understand it better.

# **Prerequisites for Flux**

Ensure that you have a running Kubernetes cluster and the cluster-admin role to deploy Flux CD

# **Install fluxctl**

wget <https://github.com/fluxcd/flux/releases/download/1.19.0/fluxctl_linux_amd64>  
mv fluxctl\_linux\_amd64 /usr/bin/fluxctl  
sudo chmod +x /usr/bin/fluxctl

let’s use GitHub as the source code repository.

Fork the khaleel-cloud786/flux-k8s-demo repository within your GitHub account.

The repository contains nginx-deployment and nginx-service manifests within the workloads directory and the web namespace definition within the namespaces directory.

Provide the name of your GitHub user within the GHUSER environment variable and the GitHub repo within the GHREPO environment variable, as shown below. Create a new namespace called flux and install the Flux CD operator in the Kubernetes cluster.

The fluxctl install command generates the required Kubernetes manifests according to the following options:

* git-user — The Git user. In this case, the GitHub username
* git-email — The Git user email. In this case, the default GitHub email
* git-url — The URL of the Git repository
* git-path — The directories within the Git repository to sync changes from
* namespace — The namespace to deploy the flux operator

export GHUSER="khaleel-cloud786"

export GHREPO="flux-k8s-demo"

export GHEMAIL="khaleel.cloud786@gmail.com"

fluxctl install \

--git-user=${GHUSER} \

--git-email=${GHEMAIL} \

--git-url=git@github.com:${GHUSER}/${GHREPO} \

--git-path=namespaces,workloads \

--manifest-generation=true \

--git-branch=main \

--namespace=flux | kubectl apply -f –

service/memcached created  
serviceaccount/flux created  
clusterrole.rbac.authorization.k8s.io/flux created  
clusterrolebinding.rbac.authorization.k8s.io/flux created  
deployment.apps/flux created  
secret/flux-git-deploy created  
deployment.apps/memcached created

**Check if Flux deployment is successful.**

kubectl -n flux rollout status deployment/flux  
deployment "flux" successfully rolled out

**Let’s get all resources within the flux namespace to see the current state of the objects.**

kubectl get all -n flux

# Authorise Flux CD to Connect to Your Git Repository

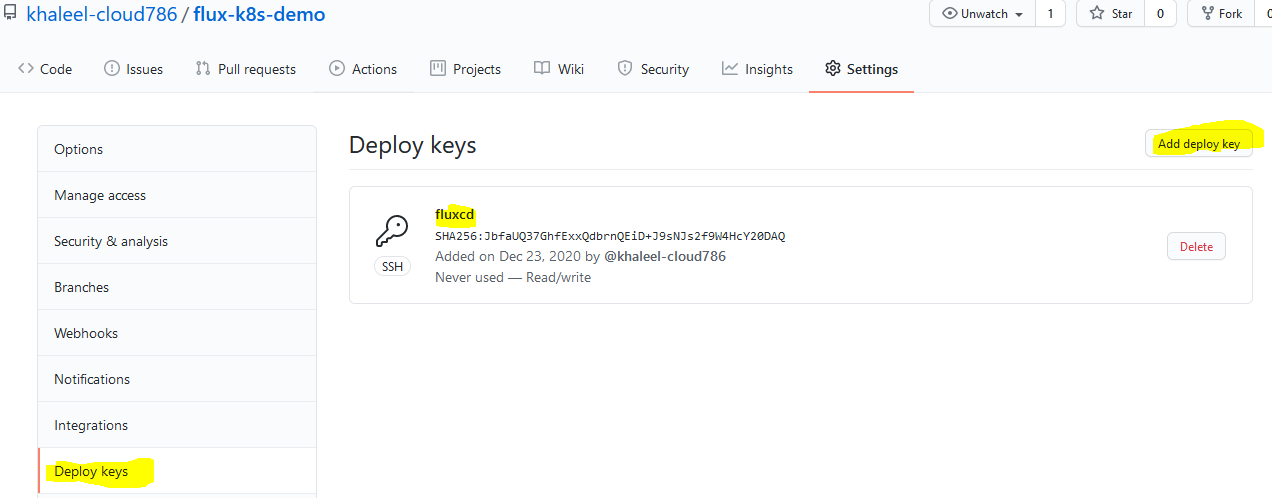
We now need to allow the Flux CD operator to interact with the Git repository, and therefore, we need to add its public SSH key to the repo.

Get the public SSH key using fluxctl.

$ fluxctl identity --k8s-fwd-ns flux

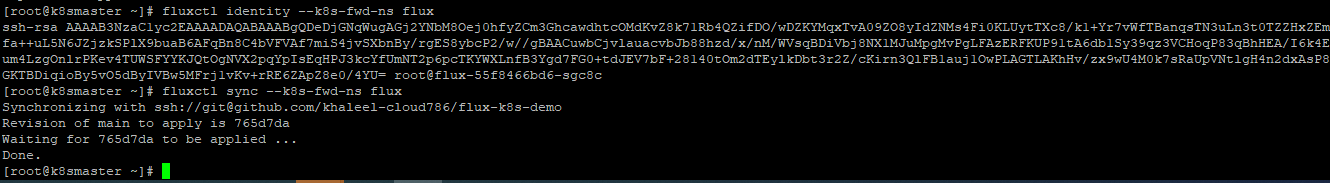
Add the SSH key to your repository so that Flux CD can access it.

* Go to https://github.com/khaleel-cloud786/flux-k8s-demo/settings/keys
* Add a name to the Key in the Title section.
* Paste the SSH Key in the Key section.
* Check “Allow write access.”



Flux CD synchronises automatically with the configured Git repository every five minutes. However, if you want to synchronise Flux with the Git repo immediately, you can use fluxctl sync, as below.

fluxctl sync --k8s-fwd-ns flux



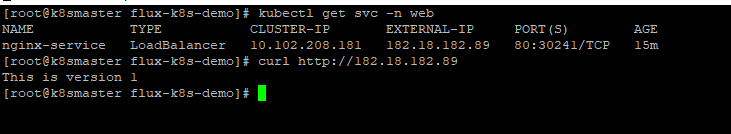
Now let’s get the pods to see if we have two replicas of nginx.

kubectl get pod -n web

kubectl get svc -n web

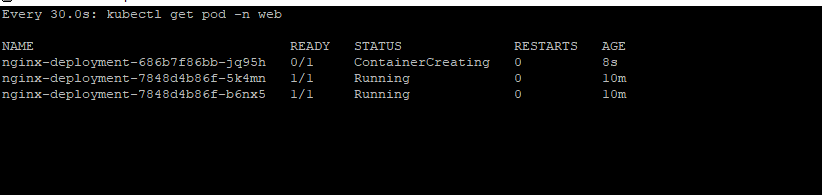
Test the service using the external IP. If your cluster does not allow you to spin load balancers, you can use the NodeIP:NodePort combination.

curl <http://182.18.182.89>



Update the image to nginx:v2 on workloads/nginx-deployment.yaml

sed -i "s/nginx:v1/nginx:v2/g" workloads/nginx-deployment.yaml  
git add --all  
git commit -m 'Updated version to v2'  
git push origin master





Congratulations! You have successfully set up Flux CD on your Kubernetes cluster.

To Uninstall

fluxctl install \

--git-user=${GHUSER} \

--git-email=${GHEMAIL} \

--git-url=git@github.com:${GHUSER}/${GHREPO} \

--git-path=namespaces,workloads \

--manifest-generation=true \

--git-branch=main \

--namespace=flux | kubectl delete -f –

kubectl delete ns flux

kubectl delete deployment nginx-deployment -n web

kubectl delete svc nginx-service -n web

kubectl delete ns web