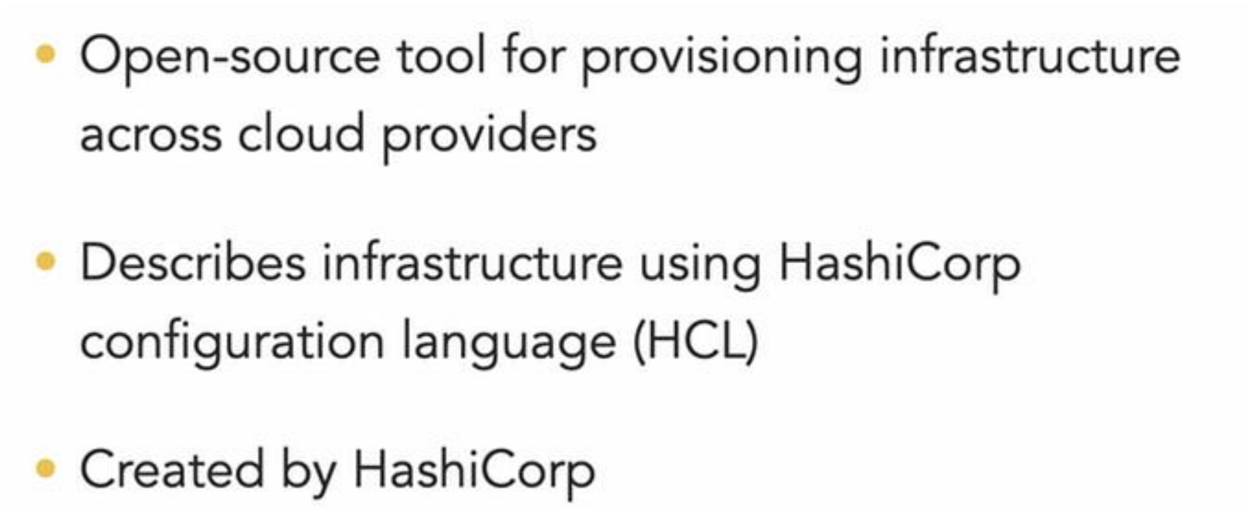
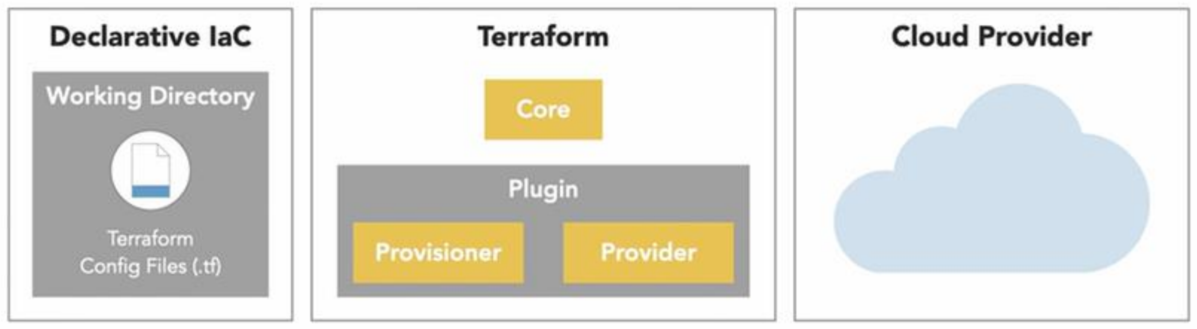
Terraform through the GitOps Approach



- Open-source tool for provisioning infrastructure across cloud providers
- Describes infrastructure using HashiCorp configuration language (HCL)
- Created by HashiCorp

Remember how to terraform Cloud works.



We have all the configuration files stored in the workspace of terraform.

We usually use the terraform cloud to start a new run.

GitOps

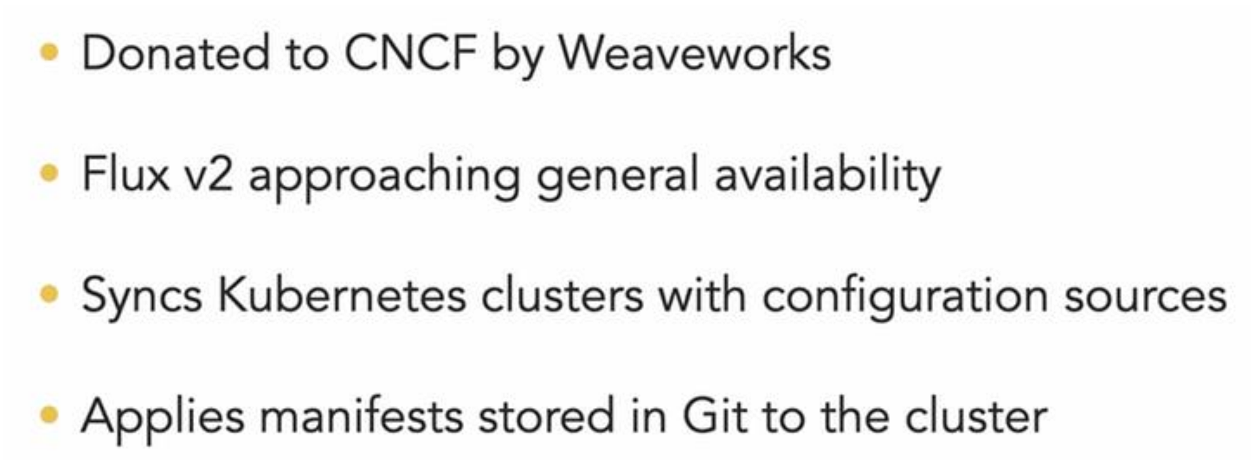
Instead of storing these files in the workspace of the terraform cloud, they should instead be stored inside Got because they represent the desired state of the infrastructure.

GitOps requires us to run the terraform life cycle through Git only. For this to happen, we must use the CLI-Driven workflow and not the version control-driven workflow.

Flux – Declarative Deployment Tool for the Kubernetes.

Flux is an operator that focuses on continuous deployment in the Kubernetes cluster.

Flux focuses on continuous deployment on the Kubernetes cluster.



- Donated to CNCF by Weaveworks
- Flux v2 approaching general availability
- Syncs Kubernetes clusters with configuration sources
- Applies manifests stored in Git to the cluster

It is important to remember that the Flux operator also has a Helm controller that synchronizes the Cluster with the Chart.

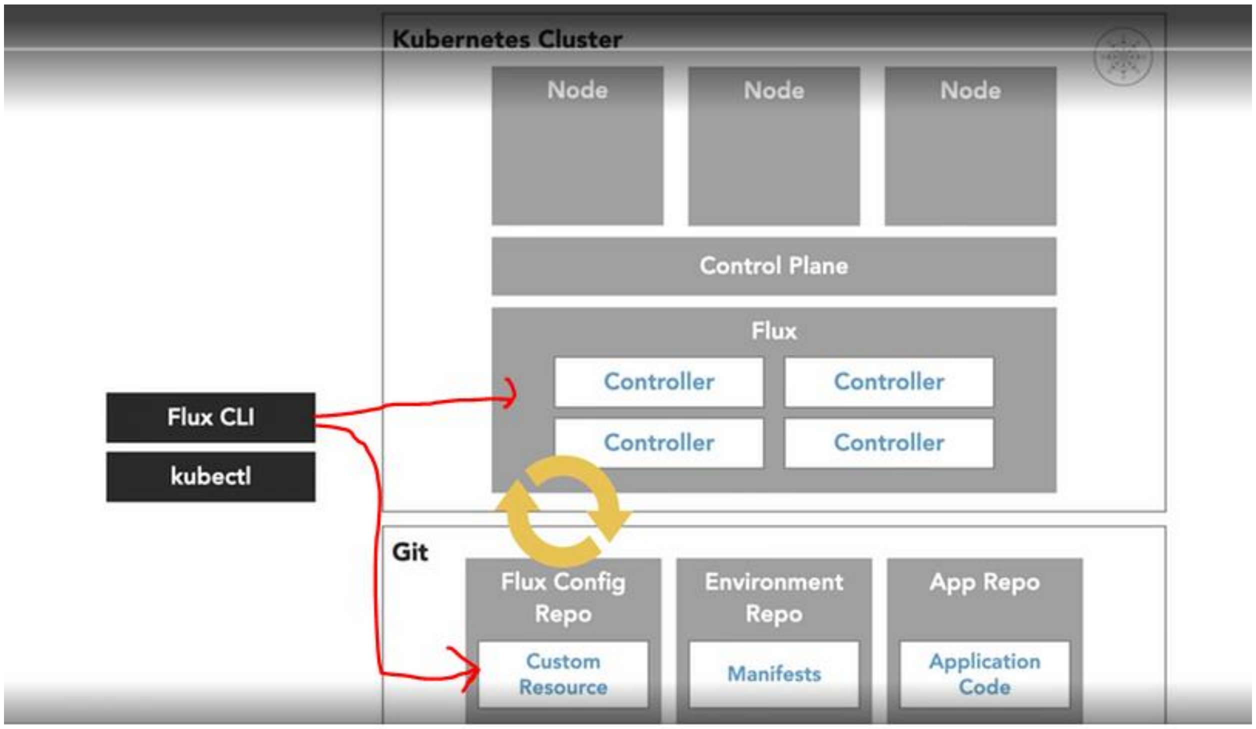
Below is how Flux Bootstrapping works

You must have Kubectl installed in your Cluster to run the Flux command.

Flux command-line interface exposes a bootstrap command that does two major things.

First, it deploys Flux to the Cluster and several controllers that Flux uses to manage the Cluster.

Second, it creates a repository in GitHub that it will use to manage itself.



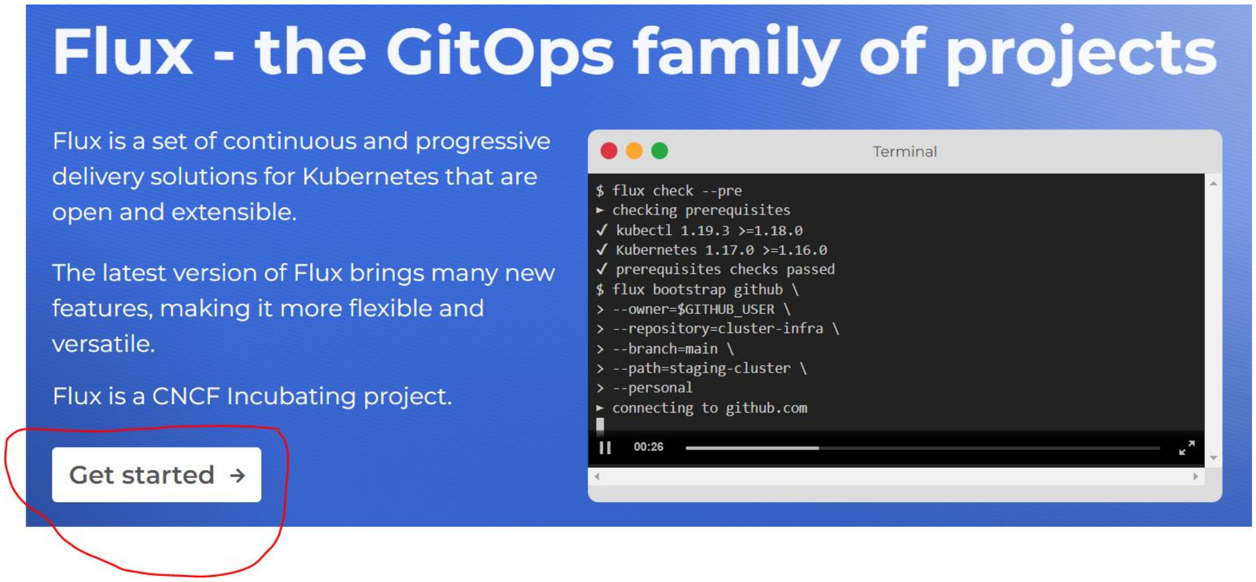
Within the Github repository, we will place definitions for custom resources used by Flux controllers to oversee operations on the Cluster. Flux monitors this repository to determine which custom resources should be applied.

Flux uses GitOps to manage itself and create a flux workflow used to deploy our application.

Let us Bootstrap the Kubernetes Cluster

https://fluxcd.io/

Click on



Flux - the GitOps family of projects

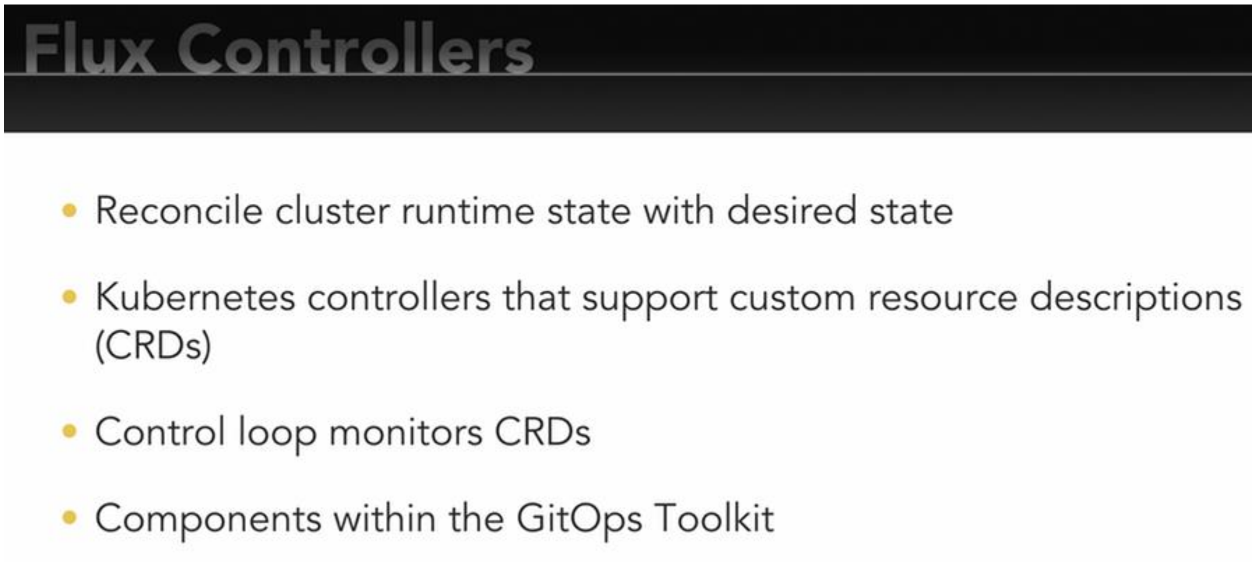
Flux is a set of continuous and progressive delivery solutions for Kubernetes that are open and extensible.

The latest version of Flux brings many new features, making it more flexible and versatile.

Flux is a CNCF Incubating project.

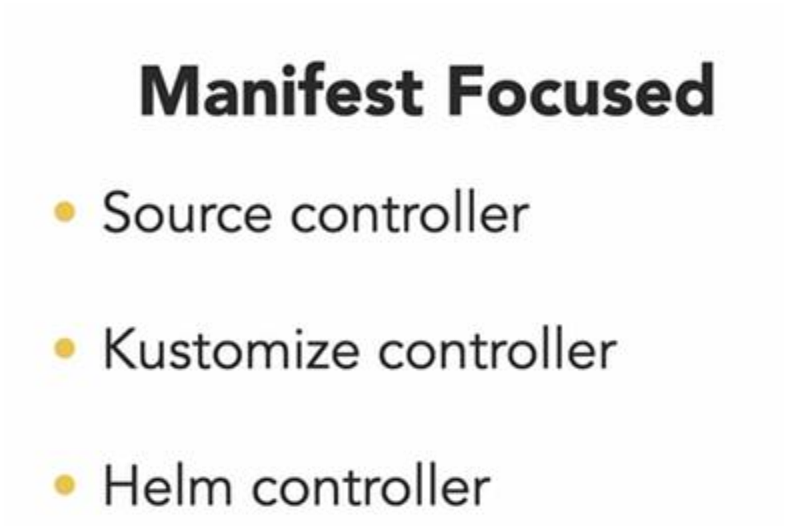
```
$ flux check --pre
► checking prerequisites
✓ kubectl 1.19.3 >=1.18.0
✓ Kubernetes 1.17.0 >=1.16.0
✓ prerequisites checks passed
$ flux bootstrap github \
> --owner=$GITHUB_USER \
> --repository=cluster-infra \
> --branch=main \
> --path=staging-cluster \
> --personal
► connecting to github.com
```

Flux Controllers



Flux Controllers

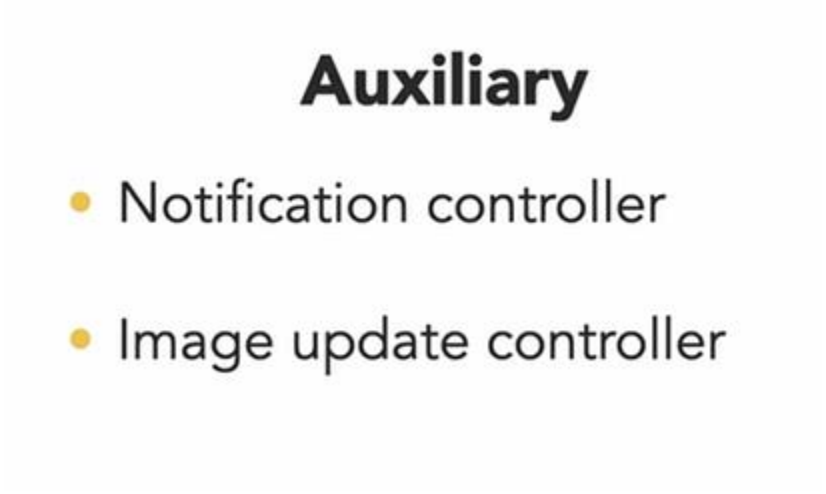
- Reconcile cluster runtime state with desired state
- Kubernetes controllers that support custom resource descriptions (CRDs)
- Control loop monitors CRDs
- Components within the GitOps Toolkit



Manifest Focused

- Source controller
- Kustomize controller
- Helm controller

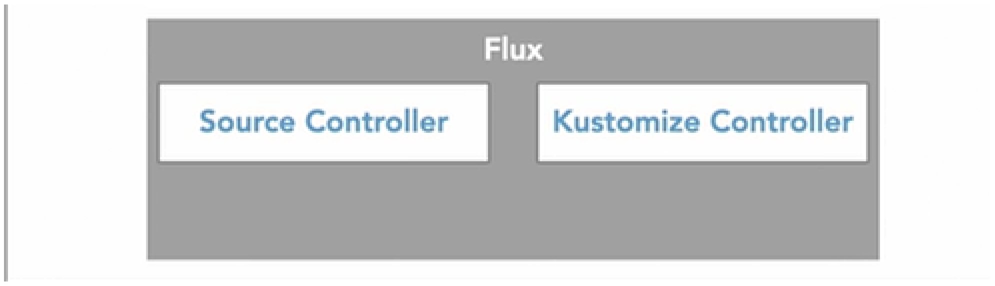
Three major controllers are used to reconcile declarative manifests inside the Cluster.



Auxiliary

- Notification controller
- Image update controller

These two controllers are used to monitor changes in Flux and update the manifest based on the changes or newly built image.



These two controllers are used to synchronize resources on the Cluster with the manifest defining our desired state and stored in the Git repository.

To do so, we start by creating a source for a source Customer Resource which we commit into the Flux configuration repository.

From here, Flux will sync with its configuration repository and apply the new resources into its namespace within the Cluster.

The source will define the source repository where the source controller can find the source manifest with all the manifestations containing the desired state we want to run in the Cluster.

At this point, the source controller will pull the repository defined in the source and start to monitor it for any changes.

Now it is time for another controller called kustomization to enter the picture. This controller is critical and is used to reconcile the source with the running state in the Cluster.

Also, this one is committed to the Flux configuration repository. Flux will then synchronize with its Flux configuration repository, and new customization will be applied within the Cluster.

Suppose the kustomixation controller notices any changes between the runtime state and the Cluster's desired state. In that case, it starts the reconciliation process carried through the Kubernetes management API, which enables the Control Plane to pull and deploy the correct desired state as expressed in the Flux Configuration Repository.

Continuous Delivery with Flux

1. We need Custom resource Definition (CRDs) - This points to the location of our system's desired state. In our case, this is the repository that stores our infrastructure code.

Let us create a custom resource definition.

We must be in the directory where the Flux configuration repository is located.

Run the PWD command.

Custom resource definition does not create resources in our Cluster. It only put the source definitions into a YAML file.

2. We are going to create Kustomization. You can think of Kustomization as a resource that defines what needs to be synced into our Cluster. Similar to the

Custom Resource definition or the source, we can use the Flux CLI to create the kustomization.
This kustomization usually refers to the new Gitops source created and committed to the Flux configuration repository.
Pay attention to prune, which allows the removal of anything running in the Cluster that does not sync with the desired state described in the source infrastructure code.
The Kustomization definition is also stored in the YAML file.

3. Commit all the YAML definitions so that Flux can take action to deploy automatically into the Cluster.
4. Run # watch flux get kustomizations
5. You can observe the kustomization controller reconciling the new changes from the desired state to equal the run state in the Cluster.