Platform Engineering UNIVERSITY

GITOPS FOR PLATFORM ENGINEERING

GitOps Tooling 101 – Argo CD, Flux CD and Sveltos Addon Controller

MODULE 04

Helm vs Kustomize

Helm vs Kustomize (I)

Helm (Template-driven): Think of it as a Programming Language for YAML. You define placeholders and fill them with data. It's a full package manager (like `apt` or `npm`).

```yaml
#deployment.yaml
kind: Deployment
metadata:
  name: {{ .Values.appName }}
spec:
  replicas: {{ .Values.replicaCount }}

#values.yaml
appName: my-app
replicaCount: 3
```

Outcome: Helm renders the final YAML by injecting values into the placeholders.

Kustomize (Overlay-driven): Think of it as Transparent Layers. You take a plain YAML file and "patch" it. There are no templates, just original manifests plus your specific changes.

```yaml
# base/deployment.yaml
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 1

# overlays/production/kustomization.yaml
resources:
  - ../../base
patches:
  - target:
      kind: Deployment
      name: my-app
    patch: |-
      - op: replace
        path: /spec/replicas
        value: 3
```

Outcome: Kustomize takes the base and overwrites the replicas to 3 for prod.

Use Helm if you want to **package** your software as a **reusable product for others**; use **Kustomize** if you have a **base configuration** and want to **tweak** it for **different environments.**

Helm vs Kustomize (II)

Feature	Helm (since 2015, v4, ⭐~29,2k)	Kustomize (since 2017, v5, ⭐~11,9k)
Philosophy	**Abstraction:** Hide complexity.	Transparency: Show everything.
Logic	Loops, if/else, functions.	No logic (static patching).
Packaging	Versioned "Charts" (.tgz).	Just a folder with YAML files.
Integration	Requires Helm CLI.	Built into kubectl (-k)
Best For	Third-party apps (Postgres, Redis).	Internal microservices.

Helm Charts – The "Wrapper Chart" Pattern

A standard Helm Chart represents a **single application** or service.

An **Umbrella Chart** is a Chart that doesn't contain its own application logic. Instead, it lists multiple other charts as **dependencies**.

A Wrapper Chart (A specialized Umbrella Chart) is used to bundle a **Third-Party Chart** (as a dependency) together with your own **Custom Resources** (in the `/templates` folder) to create a "production-ready" package.

Why use this?

- **Completeness:** The official Cert-Manager chart only installs the controller. Your Wrapper Chart adds the missing `ClusterIssuer`.
- **Standardization:** You ensure that every time Cert-Manager is installed, your company-specific security policies and namespaces are included.
- **GitOps-Friendly:** Argo CD only has to manage one "App," which includes both the tool and its configuration.

Helm Charts – The "Wrapper Chart" Pattern II

```yaml
# Chart.yaml (The Wrapper or Umbrella Chart)
apiVersion: v2
name: my-cert-manager-package
dependencies:
  - name: cert-manager
    version: v1.13.0
    repository: https://charts.jetstack.io
```

```yaml
#templates/cluster-issuer.yaml (Your Custom Add-on):
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-prod
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    # ... more config
```

Use **Wrapper Charts** to bridge the gap: Depend on official Third-Party Charts for the engine, but use your own templates to add the 'missing pieces' like *ClusterIssuer* or *Namespace*.

Helm Chart x Kustomize

The Problem: Sometimes a Third-Party Helm Chart is "inflexible." If a specific setting (like a `SecurityContext` or a custom `Label`) is **hardcoded** in the Chart or simply **not exposed** as a variable in *values.yaml*, you are stuck. You can't change the Chart without forking it.

The Solution: Use **Kustomize as a Post-Renderer**.

1. **Helm** "inflates" the Chart (generates the YAML based on your values).
2. **Kustomize** takes that generated YAML and applies **patches** to fix or add the missing pieces.

Why this makes sense (Key Benefits):

- **No Forking:** You don't have to copy and modify the original Helm Chart. You stay "upstream-compatible" and can still update the Chart easily.
- **Fix Hardcoded Values:** If a developer forgot to make a field configurable, Kustomize can "force" the change via a JSON patch.

Building a GitOps Service Catalog with Helm

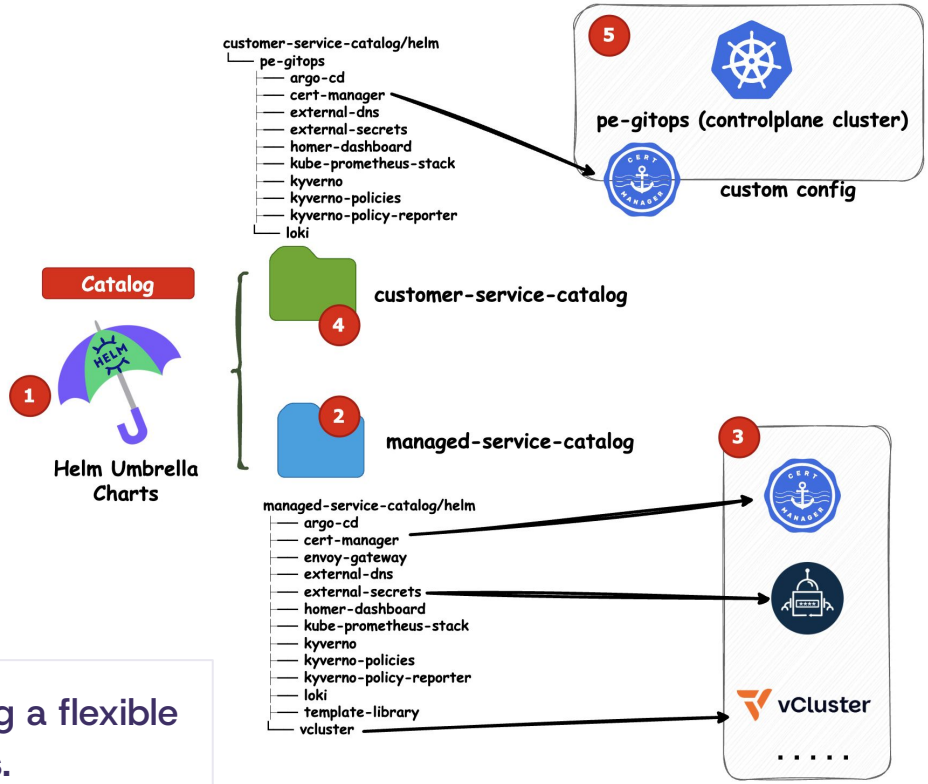
Let's build a Service Catalog for your Platform

What is a Service Catalog?

A service catalog based in our case on **umbrella helm chart** is a **collection of third-party** tools like **external-secrets** for secrets management, cert-manager for cert management or **vCluster** to create multi-tenant in one central place.

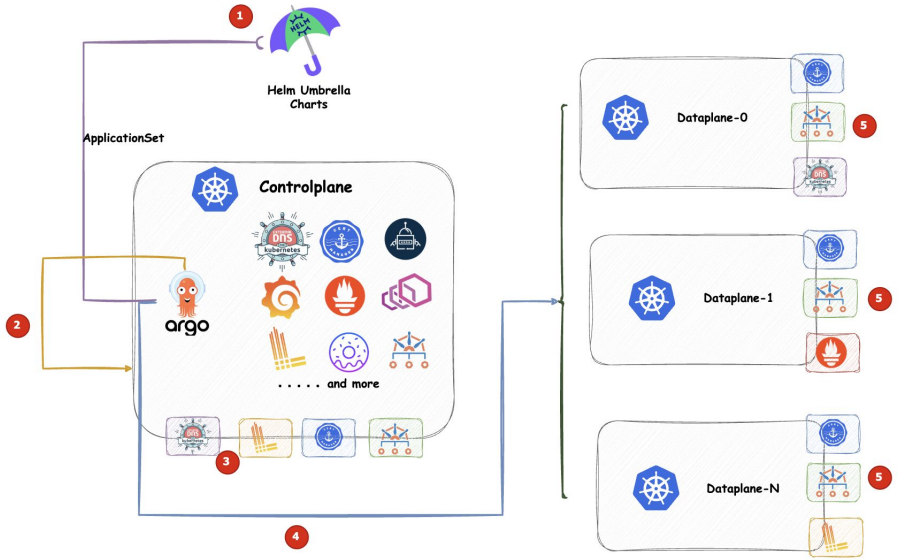
Why do we need it?

If you want to manage **multiple cluster**, which **required same or similar tool stack** it makes easier having them **on one place configured** with best practices, but still flexible and allows you to set **custom values** for every tool for every cluster.



This setup allows **GitOps at scale**, managing a flexible cluster fleet and covering **day-2 operations.**

Let's build a Service Catalog for your Platform



You can use services from **the service catalog** (1) and manage them with *ApplicationSets*.
Based on labels (3,5), you can manage the control plane cluster itself (2) and **also a fleet** of data plane or workload clusters (4) — **simply by setting labels on them.**

You can even use labels like **T-shirt sizes**. For example: every cluster with **size S** gets the **common tool stack**, but without observability.

And then you can easily extend clusters with **additional services based on their labels.**

You can manage **1,000+ clusters** with one tool and one interface — over UI or API.
But when that **tool or its catalog** backend becomes a bottleneck, the **state store matters at real scale.**

Overview: Argo CD, Flux CD and Sveltos Addon Controller

Argo CD vs Flux CD vs Sveltos

Argo CD

First Release: 2019 (v1.0) – *Started as an internal project at **Intuit** in late 2018.*

Core Functionality: An application-centric GitOps controller that provides a powerful **Single Pane of Glass.** It visualizes the entire resource hierarchy of an application and allows for manual or automated synchronization.

Best for developer self-service and teams that prioritize visibility, observability, and rich **UI dashboards**.

Flux CD

First Release: 2016 (by Weaveworks) – *The pioneer of the term "GitOps".*

Core Functionality: A lightweight, Kubernetes-native toolkit consisting of specialized controllers (Source, Helm, Kustomize). It follows the "**Unix philosophy**", focusing on automation, security, and low-footprint background synchronization.

Best for platform engineers who want a modular, **invisible "GitOps engine" that integrates deeply with native K8s RBAC and OCI registries.**

Sveltos

NEW!

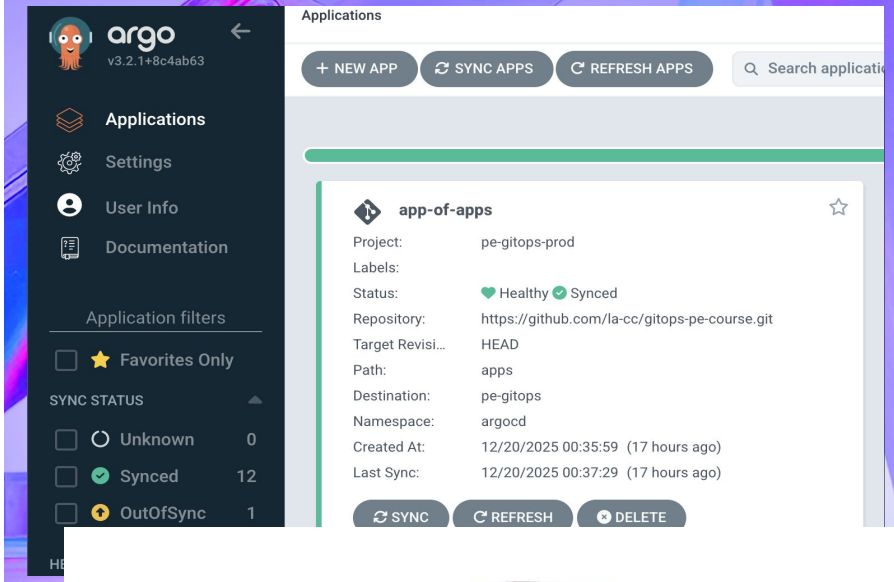
First Release: 2022 (by the Open Source Community)

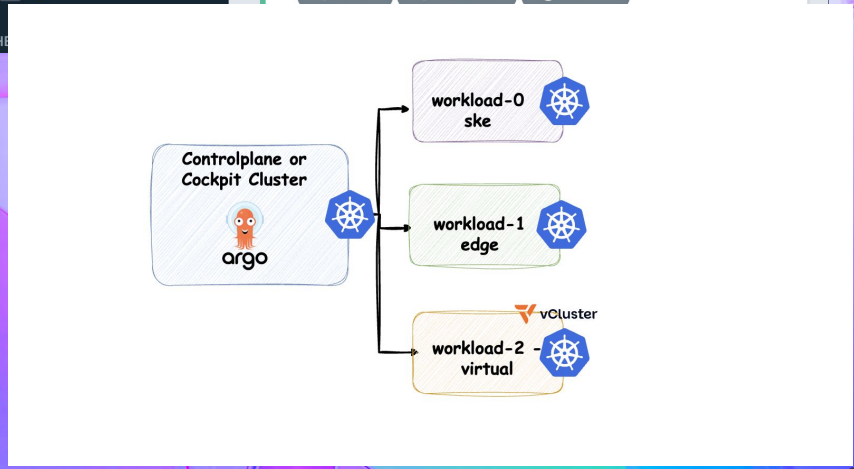
Core Functionality: A Multi-Cluster Add-on Controller designed for massive scale. It uses label-based "*ClusterProfiles*" and event-driven triggers to distribute **apps** and **infrastructure components across a fleet.**

Best for **managing large-scale cluster** fleets with an **agent-based** architecture and native multi-tenancy support.

Argo CD

Year: 2019 v1.0 (not 2017, Argo Workflows)
Stars: 21.5k ⭐ by GitHub
Adopters: 430+ companies like Red Hat, Adobe, Allianz, Intuit, Akuity, Codefresh and also iits-consulting)
CNCF Status: Graduated since 2022
UI: yes, built-in since beginning
CLI: yes, argocd
Topologies: Hub and Spoke, Dedicated instance per clusters, Agent based (Experience status)
Contributors: 1798
Bootstrapping: Helm or YAML Manifests
OCI Support: Yes, since v3.1, 2025 NEW!
Event Driven: yes, but just with Argo Events
Multi-Tenancy: centralized RBAC, SSO/OIDC, Projects
Integration: EKS Capability for Argo CD, 2025 NEW!





Argo CD – Bootstrapping – YAML

Basic Installation or Quick Start

```
kubectl create namespace argocd
```

```
kubectl apply -n argocd -f
https://raw.githubusercontent.com/argoproj/ar
go-cd/stable/manifests/install.yaml
```

This will create a new namespace, argocd, where Argo CD services and application resources will live.

Then you need to port-forward to connect to the API, when an Ingress does not exist.

```
kubectl port-forward svc/argocd-server -n
argocd 8080:443
```

Now you need to get the password, user is admin.

```
kubectl -n argocd get secret
argocd-initial-admin-secret \ -o
jsonpath="{.data.password}" | base64 -d;
echo
```

Now you need to get the password, user is admin. Open the UI in your browser and Login:

```
http://localhost:8080/argocd/
```

Then you can create your Application in the UI and create an App of Apps. Declaratively specify one Argo CD app that consists only of other apps.

Argo CD – Bootstrapping – Helm

Basic Installation or Quick Start

```
helm repo add argo
https://argoproj.github.io/argo-helm

helm repo update

helm install argocd argo/argo-cd \
  --namespace argocd \
  --create-namespace
```

This will create a new namespace, argocd, where Argo CD services and application resources will live.

Then you need to port-forward to connect to the API, when an Ingress does not exist.

```
kubectl port-forward svc/argocd-server -n
argocd 8080:443
```

Now you need to get the password, user is admin.

```
kubectl -n argocd get secret
argocd-initial-admin-secret \ -o
jsonpath="{.data.password}" | base64 -d;
echo
```

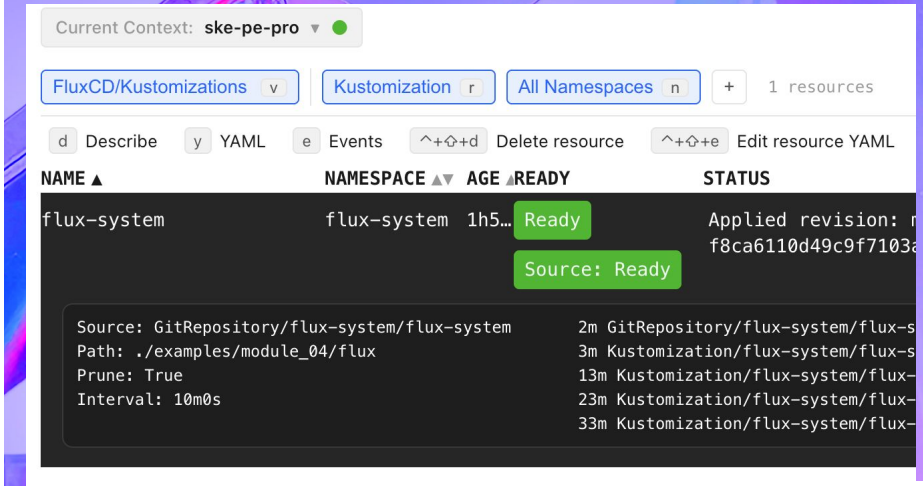
Now you need to get the password, user is admin. Open the UI in your browser and Login:

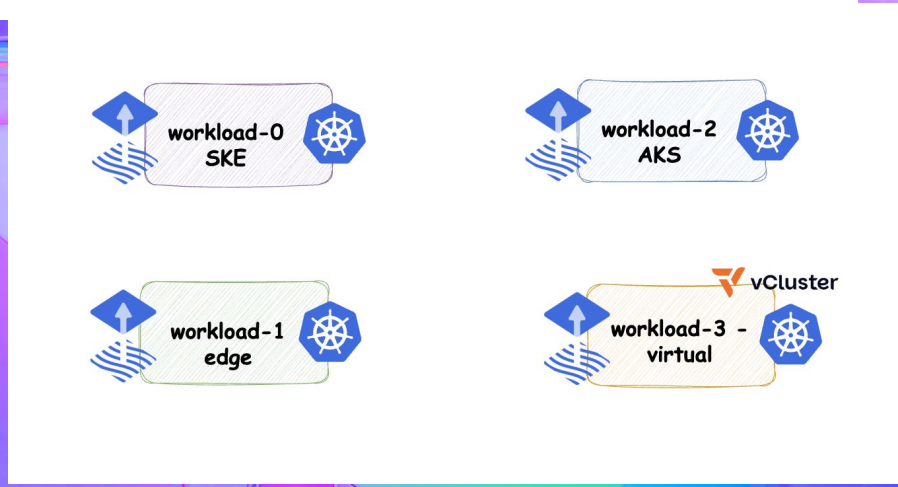
```
http://localhost:8080/argocd/
```

Then you can create your Application in the UI and create an **App of Apps**. Declaratively, specify one Argo CD app that consists only of other apps.

Flux CD

Year: 2016 v1 (Monolith) and 2020 v2 (Controllers)
Stars: 7.3k ⭐ by GitHub
Adopters: 164 (112+ v2 and 52+ v1 (Legacy)), SAP, Grafana, Cisco and again iits-consulting
CNCF Status: Graduated since 2022
UI: not direct, just third party like Capacitor
CLI: yes, flux
Topologies: Sharding, Dedicated instance per clusters
Contributors: 177
Bootstrapping: CLI, Operator or Community Helm Chart
OCI Support: Yes, since v0.31 2022
Event Driven: yes, but minimal for trigger, notifications
Multi-Tenancy: native Kubernetes RBAC, Namespaces
Integration: Azure AKS Extension, since 2022





Flux CD – Bootstrapping CLI

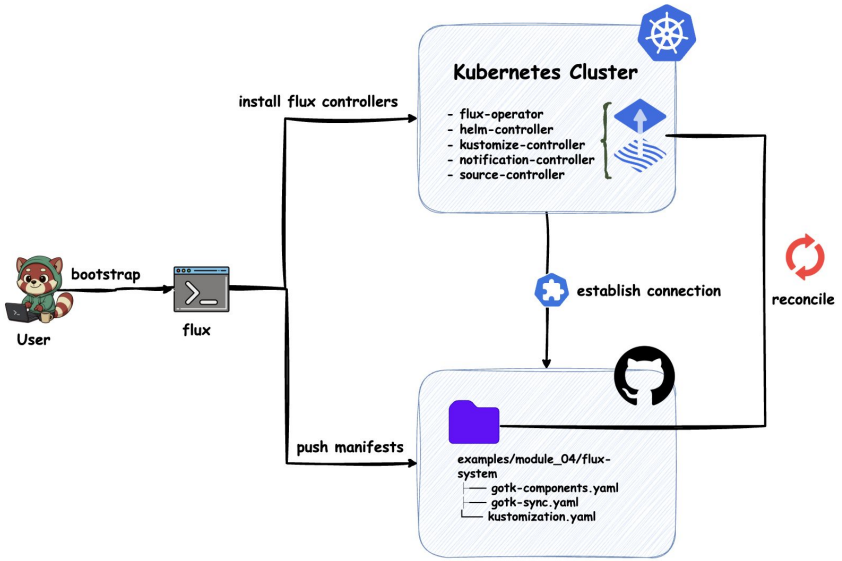
Install the Flux CLI (MacOS)

```
brew install fluxcd/tap/flux
```

Bootstrap with Flux CLI

The `flux bootstrap` command deploys the Flux controllers on Kubernetes cluster(s) and configures the controllers to sync the cluster(s) state from a Git repository

```
flux bootstrap github \
  --token-auth \
  --owner=la-cc \
  --repository=gitops-pe-course \
  --branch=main \
  --path=examples/module_04/  \
  --personal
```



Flux CD – Bootstrapping – Flux Operator

Example of installing the Flux Operator using Helm:

```
helm install flux-operator
oci://ghcr.io/controlplaneio-fluxcd/charts/flux-
operator \
  --namespace flux-system \
    --create-namespace
```

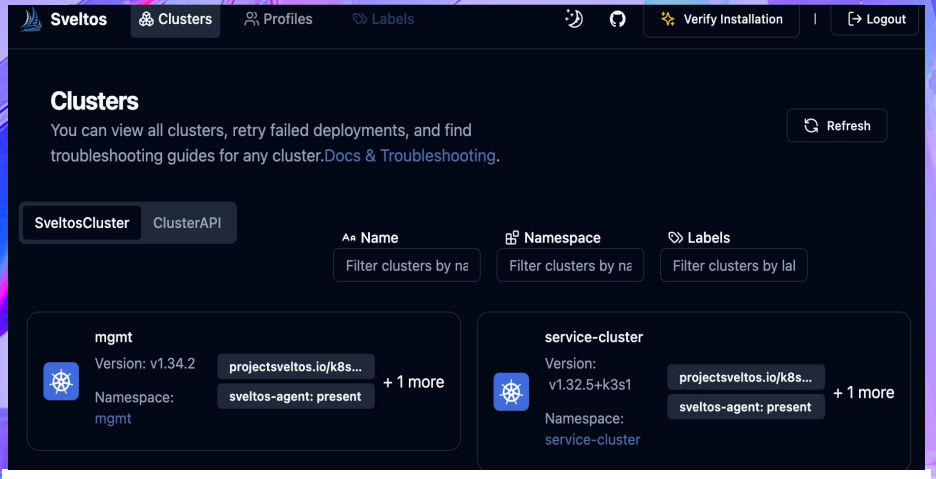
Example of configuring the Flux instance (Custom Resources)

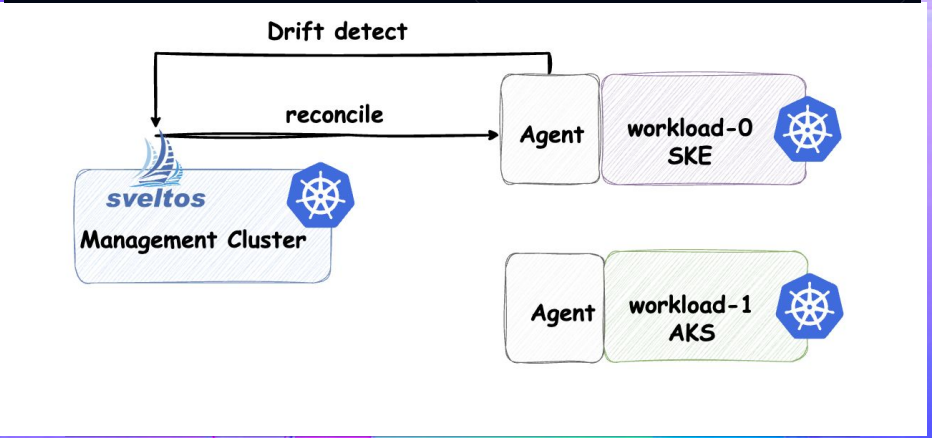
```
apiVersion: fluxcd.controlplane.io/v1
kind: FluxInstance
metadata:
  name: flux
  namespace: flux-system
  annotations:
    fluxcd.controlplane.io/reconcileEvery: "1h"
    fluxcd.controlplane.io/reconcileTimeout: "5m"
spec:
  distribution:
    version: "2.7.x"
    registry: "ghcr.io/fluxcd"
    artifact:
"oci://ghcr.io/controlplaneio-fluxcd/flux-operator-m
anifests"
  components:
    - source-controller
    - kustomize-controller
    - helm-controller
    - notification-controller
…
```

Sveltos

Year: 2022
Stars: 471 ⭐ by GitHub
Adopters: 10+ companies like Mirantis, CLASTIX, Platform9
CNCF Status: Not a CNCF Project, but OSS (Apache 2.0)
UI: yes, read only 

CLI: yes, sveltosctl
Topologies: Hub and Spoke with Agent based (Push and Pull)
Contributors: 19
Bootstrapping: Helm, Kustomize and YAML Manifests
OCI Support: Yes, since 2023, v0.12.0
Event Driven: yes, in built
Multi-Tenancy: delegated, RoleRequest, Tenants
Integration: k0RDENT policy Driven Cluster Management





Sveltos – Bootstrapping YAML

Sveltos supports two modes: **Mode 1** and **Mode 2**.

- **Mode 1:** Will deploy up to two agents, *sveltos-agent* and *drift-detection-manager*, in each **managed cluster**.
- **Mode 2:** Sveltos agents will be created, per managed cluster, in the management cluster. The agents, while centrally located, will still monitor their designated managed cluster's API server. Sveltos leaves no footprint on managed clusters in this mode.

Mode 1: Local Agent Mode

```
kubectl apply -f
https://raw.githubusercontent.com/projectsvelt
os/sveltos/v0.57.2/manifest/manifest.yaml
```

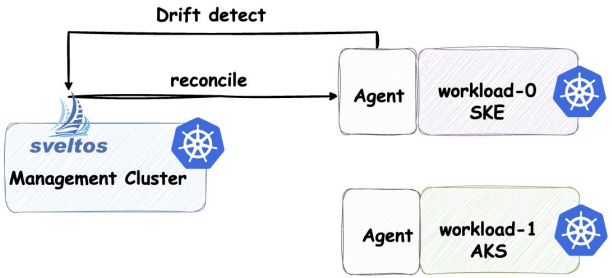
```
kubectl apply -f
https://raw.githubusercontent.com/projectsveltos
/sveltos/v0.57.2/manifest/default-instances.yaml
```

Sveltos – Bootstrapping Helm

Mode 1: Local Agent Mode

```
helm install projectsveltos
projectsveltos/projectsveltos -n
projectsveltos --create-namespace
```

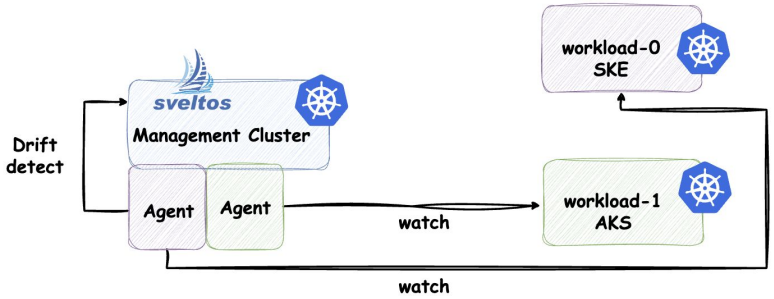
```
helm list -n projectsveltos
```



Mode 2: Centralised Agent Mode

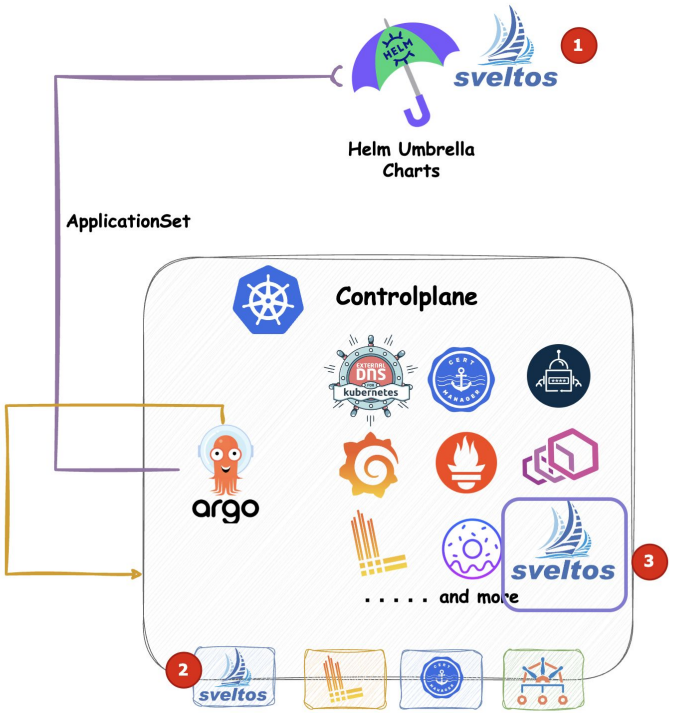
```
helm install projectsveltos
projectsveltos/projectsveltos -n
projectsveltos --create-namespace --set
agent.managementCluster=true
```

```
helm list -n projectsveltos
```



Sveltos – Bootstrapping: The Kubara Way

1. Create an Umbrella Helm Chart and add it to the catalog
2. Put a label on the Controlplane
3. Lets Argo CD doing the magic, deploying and manage Sveltos



General Info and Community Adoption

Feature	Argo CD	Flux CD	Sveltos
First Release	2019 (v1.0)	2016 (v1 Monolith)	2022
GitHub Stars	~21.5k ⭐	~7.3k ⭐	~471 ⭐
Contributors	~1798	~177	~19
Adopters	~430+ (Adobe, Tesla, iits-consulting)	~164 (SAP, Cisco, iits-consulting)	~10+ (Mirantis, Platform9)
CNCF Status	Graduated (since 2022)	Graduated (since 2022)	Open Source (Apache 2.0)
Web UI	Yes (built-in since start)	No (only via external tools)	Yes (read-only GUI)
CLI	argocd	flux	sveltosctl
Best For...	Dev Self-Service & Visibility	Platform Engineers ("Invisible Engine")	Massive Fleet & Add-on Management

Technical Architecture & Modern Features

Feature	Argo CD	Flux CD	Sveltos
Topologies	Hub & Spoke, Dedicated, Agent (Exp.)	Hub & Spoke, Sharding, Dedicated per cluster	Hub & Spoke with Agent (Push & Pull)
Multi-Tenancy	Centralized RBAC, SSO/OIDC, Projects	Native K8s RBAC & Namespaces	Delegated, RoleRequest, Tenants
Bootstrapping	Helm or YAML based manifests	CLI, Operator, or Community Helm	Helm or YAML based manifests
OCI Support	Yes (since v3.1, 2025)	Yes (since v0.31, 2022)	Yes (since v0.12.0, 2023)
Event Driven	Yes (via Argo Events)	Yes (Notification Controller)	Yes (native / built-in)
Integrations	EKS Capability (2025)	Azure AKS Extension (2022)	kORDENT, Cluster-API, Kamaji
Progressive Del.	Using Argo Rollouts	Using Flagger	–

Which Tool to choose...

If you need a Rich Ecosystem: Choose **Argo CD** (includes Workflows, Events, Rollouts, and Image Updater).
If you need Progressive Delivery: Use **Argo CD** (with Rollouts) or **Flux CD** (with Flagger).
If you need Multi-Tenancy: **Sveltos** and **Flux CD** are the strongest candidates.
If you need secure Hub&Spoke : **Sveltos** with agent based pull approach.
If you manage a Fleet of Clusters: **Argo CD** and **Sveltos** are built for this scale.

If you want Kubernetes-Native feel: **Flux CD** and **Sveltos** integrate most naturally.
If you build Event-Driven Services: **Sveltos** is the specialized choice for cloud-native triggers.
If you need an Easy Entry Point: **Argo CD** wins (no direct K8s access required for users).
If you want the Highest Adoption: **Argo CD** is the industry standard for everyone from Juniors to Experts.
If you run on Edge / Limited Hardware: **Flux CD** or **Sveltos** are ideal due to their low resource footprint.
If you run just 1 or 2 Cluster: **Flux CD** or **Argo CD**

> These are only rough indications. As always, it's a depends-on answer!

Who says you need...

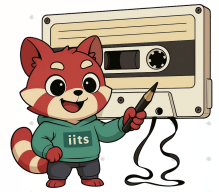
You can combine the best of different tools!

Use **Flux CD as an extension** on Azure to manage infrastructure, and **Argo CD for application management.**

Use **Argo CD to manage Sveltos** Custom Resources, and **Sveltos to manage event-driven fleets of clusters.**

Use **Argo CD to roll out Flux CD on dedicated edge clusters**, and **Flux to manage these resources efficiently.**

Recap: GitOps Tooling 101...



- **Use Helm** if you want to **package** your software as a **reusable product for others;** use **Kustomize** if you have a **base configuration** and want to **tweak** it for **different environments.**
- **Build a catalog** or central Way to manage your config – Don't spread it over a lot of repositories!
- Be aware of which GitOps Tools exists and about their capabilities – **no need to focus on one tool,** you can combine strengths of the tools

Demo