Abstract, 15 min talk

At Lunar bank we had a good problem, our customers rely on us to move quickly and provide new features and to do so in a highly reliable manner. To meet their needs we set out on a journey to move from canary deployments, where we could test new features in a safe fashion, to canary clusters. We envisioned a world where our production clusters were truly disposable and after 3 years we finally achieved that goal. In this session we will share how we did it, and how you can too.

Today any engineer at Lunar bank can fail over the entire platform in 40 minutes. By deeply integrating with our infrastructure provider, writing some new custom operators, and moving most state out of the cluster Lunar is in a position to make disaster recovery a day to day operation. Listen as Henrik shares the successes, key learnings, and challenges we faced along the way.

LUNAR®

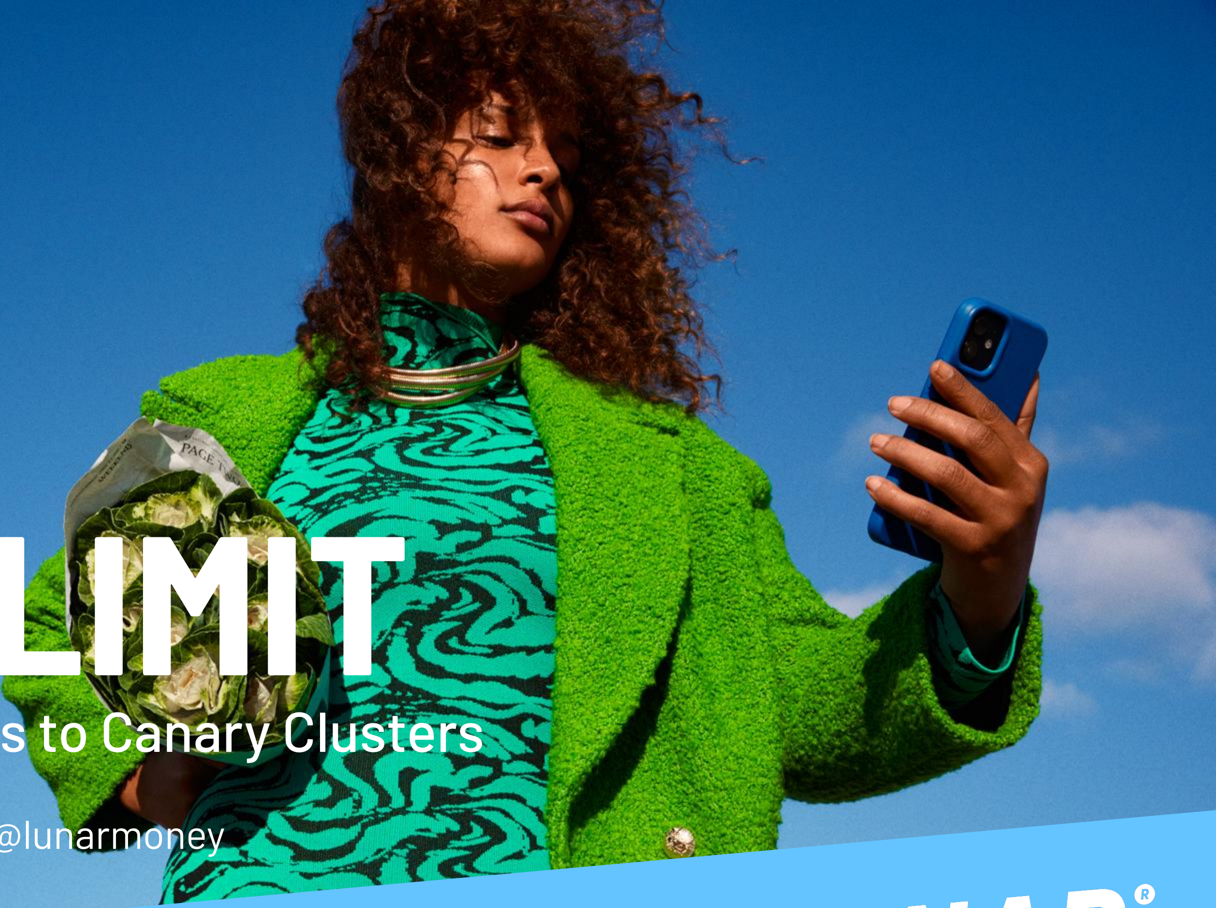
Metaslide

Bio

Henrik Høegh is a Cloud Native Co-organizer in Cloud Native Aarhus where he contributes to the community with event planning and talks. He works as Platform Engineer at Lunar maturing, developing the platform and giving support to its users.

He is currently focused on maturing Lunars failover capabilities and onboarding new developers to the platform. He has been using Kubernetes since early 2016 and has done countless talks on Kubernetes for beginners. Before joining Lunar Henrik worked as a consultant implementing a Cloud Native edge computing platform for one of the largest wind turbine companies in the world.

Metaslide

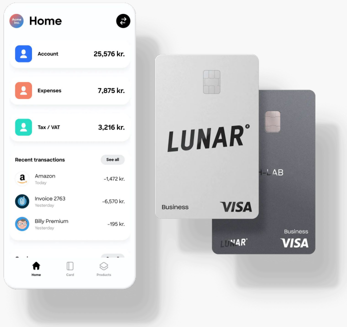


PUSH IT TO THE LIMIT

From Canary Deployments to Canary Clusters

Henrik Høegh - Platform Engineer @lunarmoney
@HenrikHoegh

LUNAR



15,000

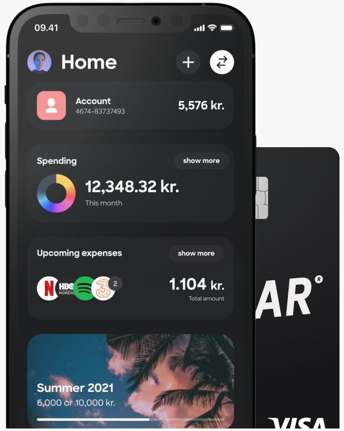
Total number of Business Customers

650

Employees

European Banking License issued in Denmark

We have offices in these locations



500,000

Customers in total

Company founded in 2015

€345m

Total amount raised

Series D ✓

Recently closed our Series D of €210m

UNICORN

APPARENTLY WE ARE NOW A UNICORN x 2

Photo by mark glancy: https://www.pexels.com/photo/boston-terrier-wearing-unicorn-pet-costume-1564506/

WHO?

HENRIK RENÉ HØEGH

PLATFORM ENGINEER

@HenrikHoegh

Co-organizer in Cloud Native Aarhus

Occasional speaker at Meetups, Conferences

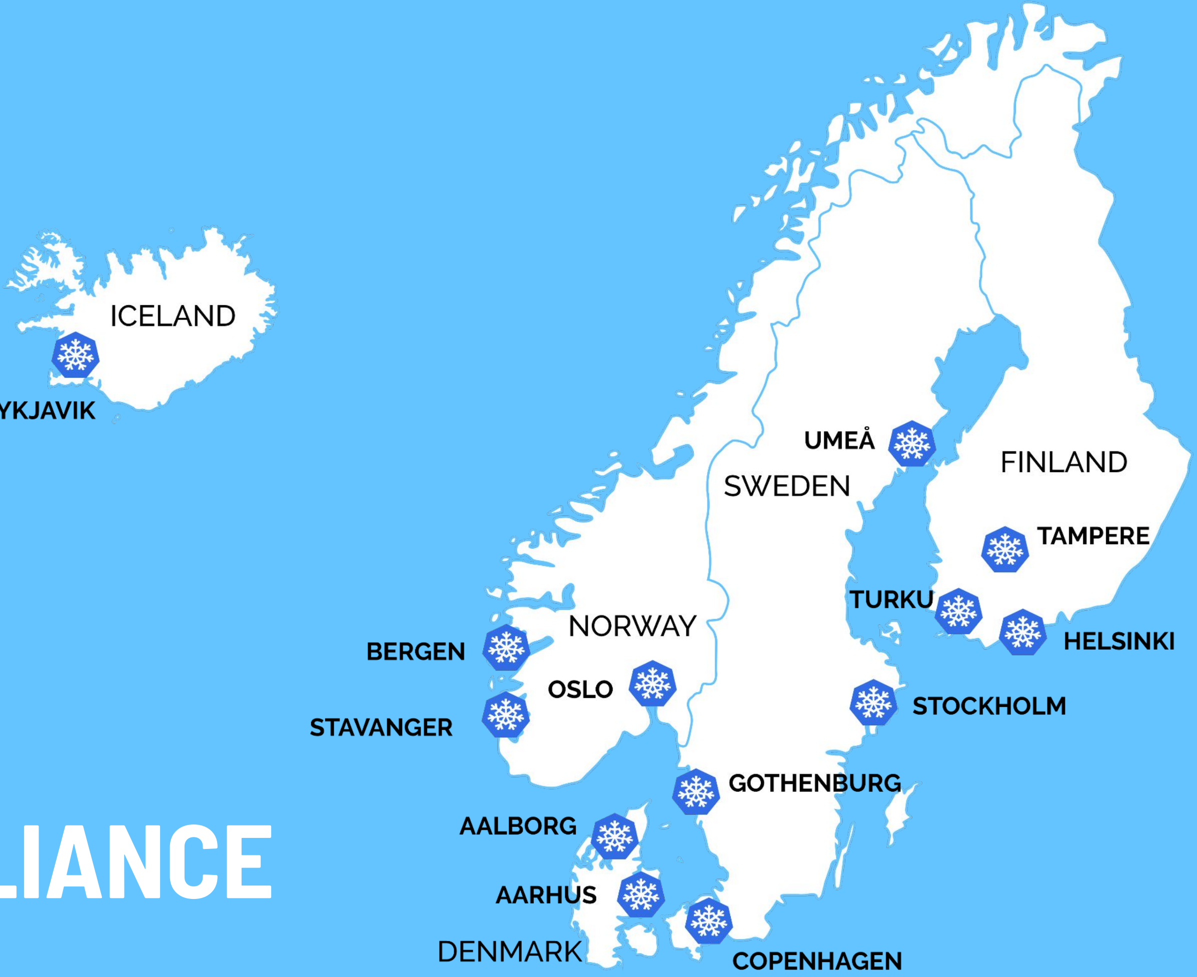
Working as a consultant for more than 14 years

Hobby : Dungeon & Dragons

CLOUD NATIVE
NORDICS

NORDIC

MEETUP ALLIANCE



Agenda

- Our tech stack
- How we did failovers
- Key changes for speed
- Future

Our failover journey starts

- 3 years of hard work
- From monolith to microservices
- From deployment pipelines to GitOps

NOW

OUR TECH STACK

Kubernetes

GitOps

Flux v1

AWS RDS database

Rabbit MQ

External DNS

NOW

OUR TECH STACK - HOME BREW

Shuttle

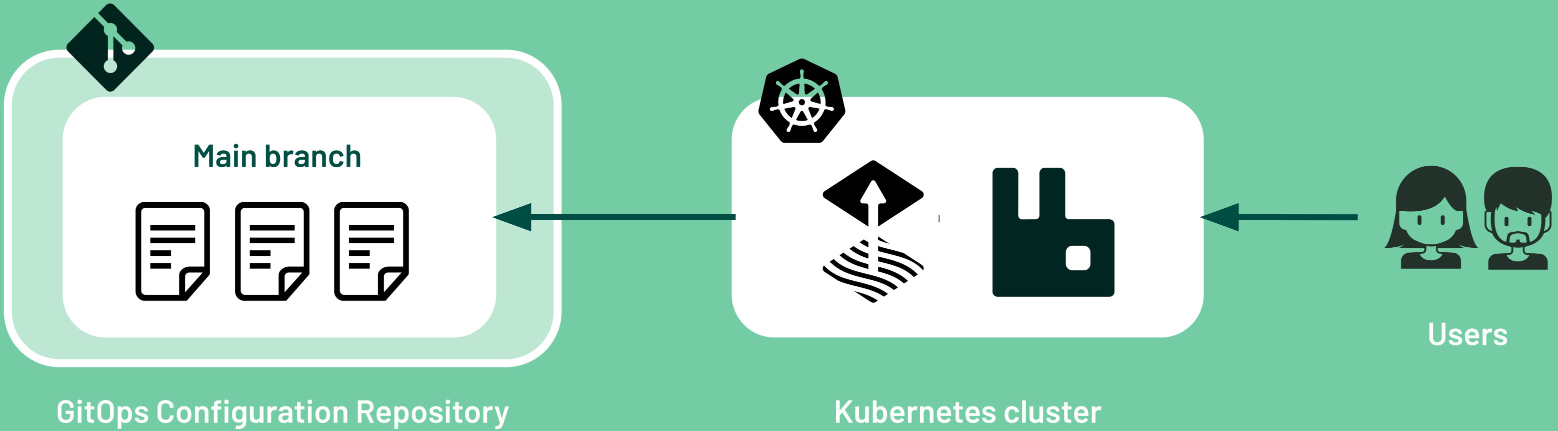
Release-manager

FAILOVER

1. GENERATION

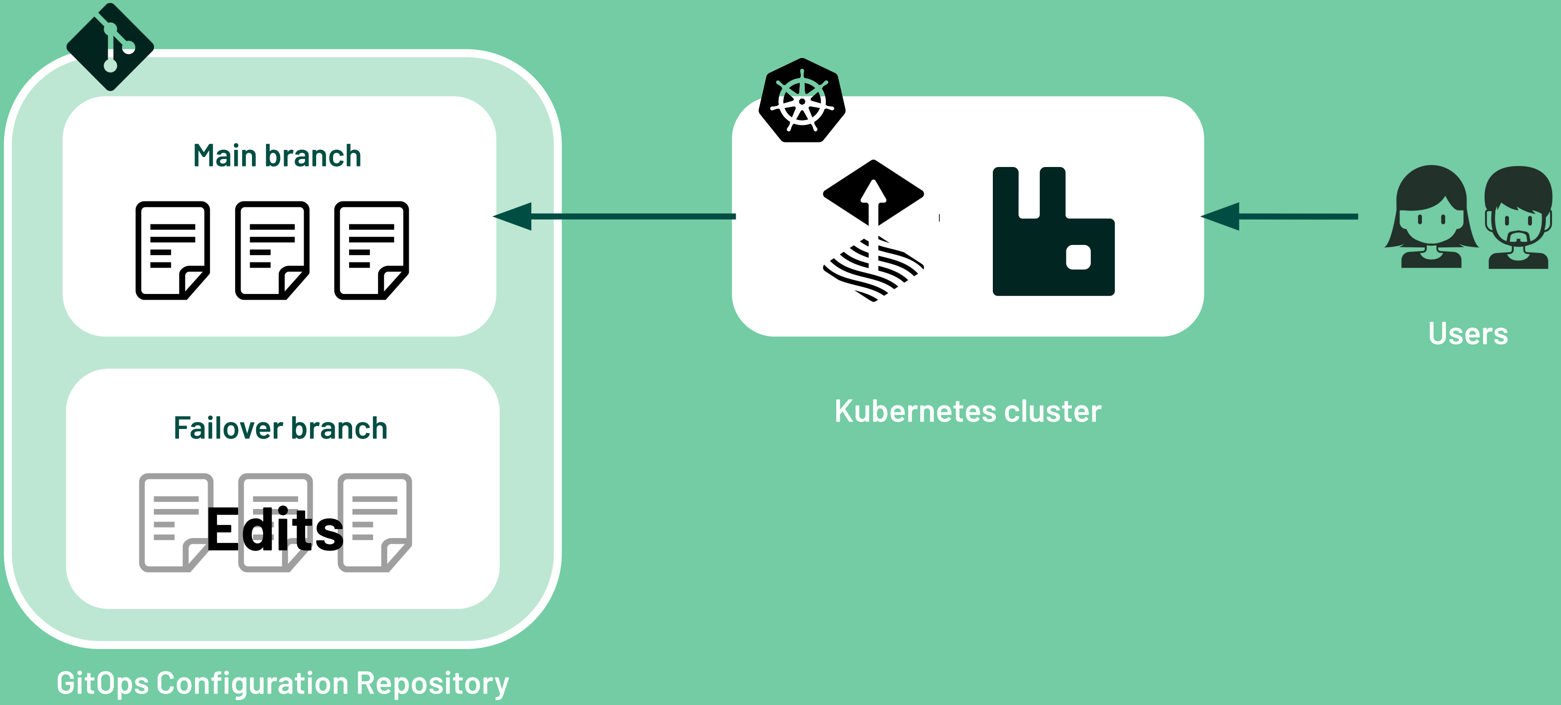
FAILOVER

1. GENERATION



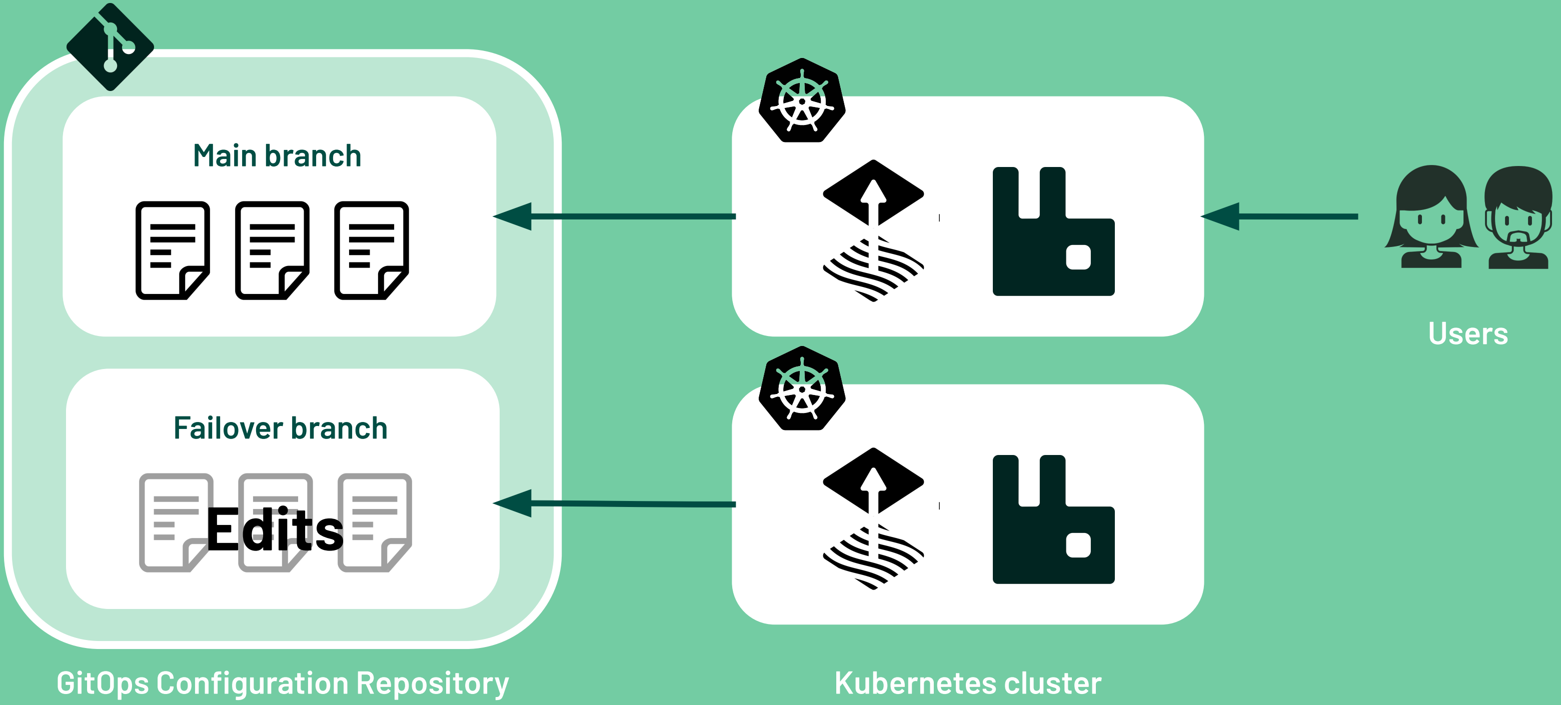
FAILOVER

1. GENERATION



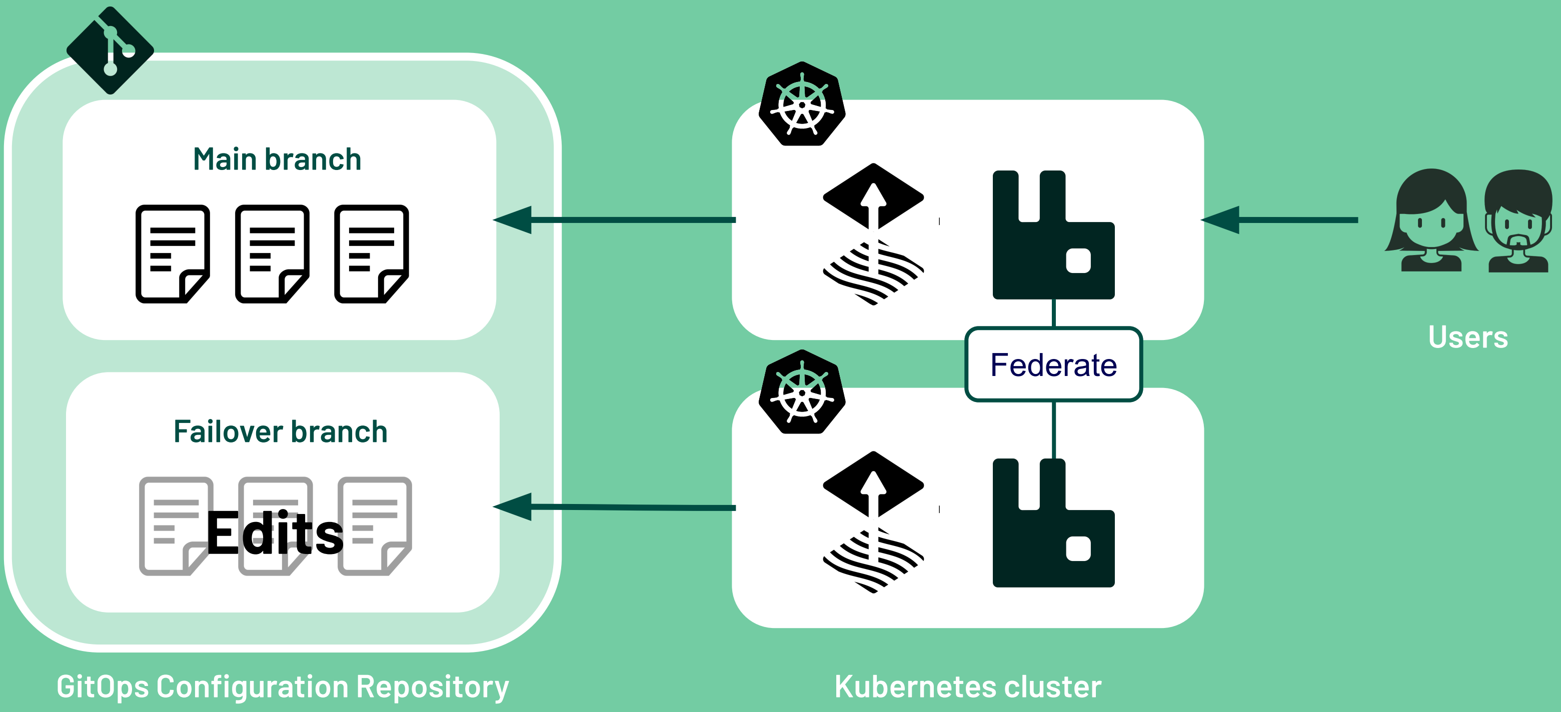
FAILOVER

1. GENERATION



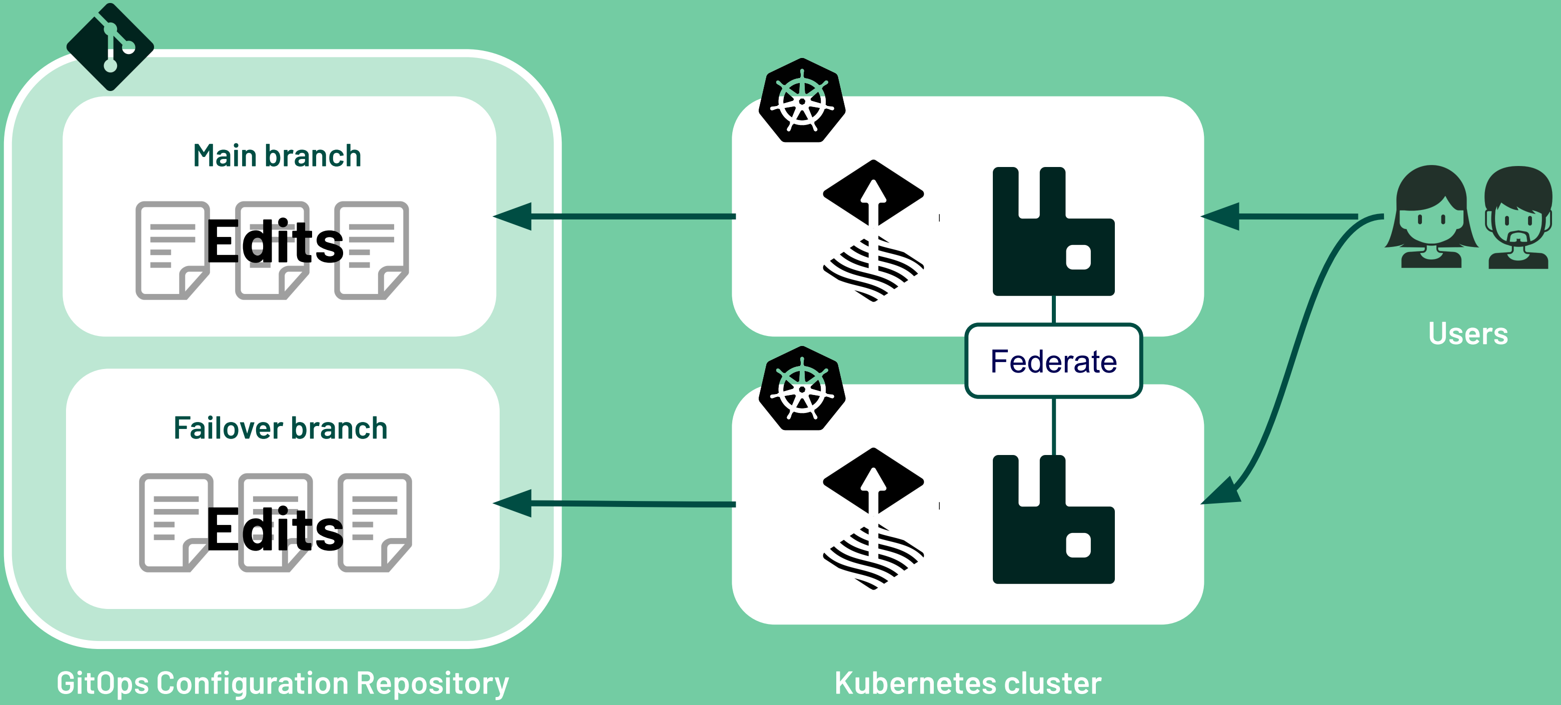
FAILOVER

1. GENERATION



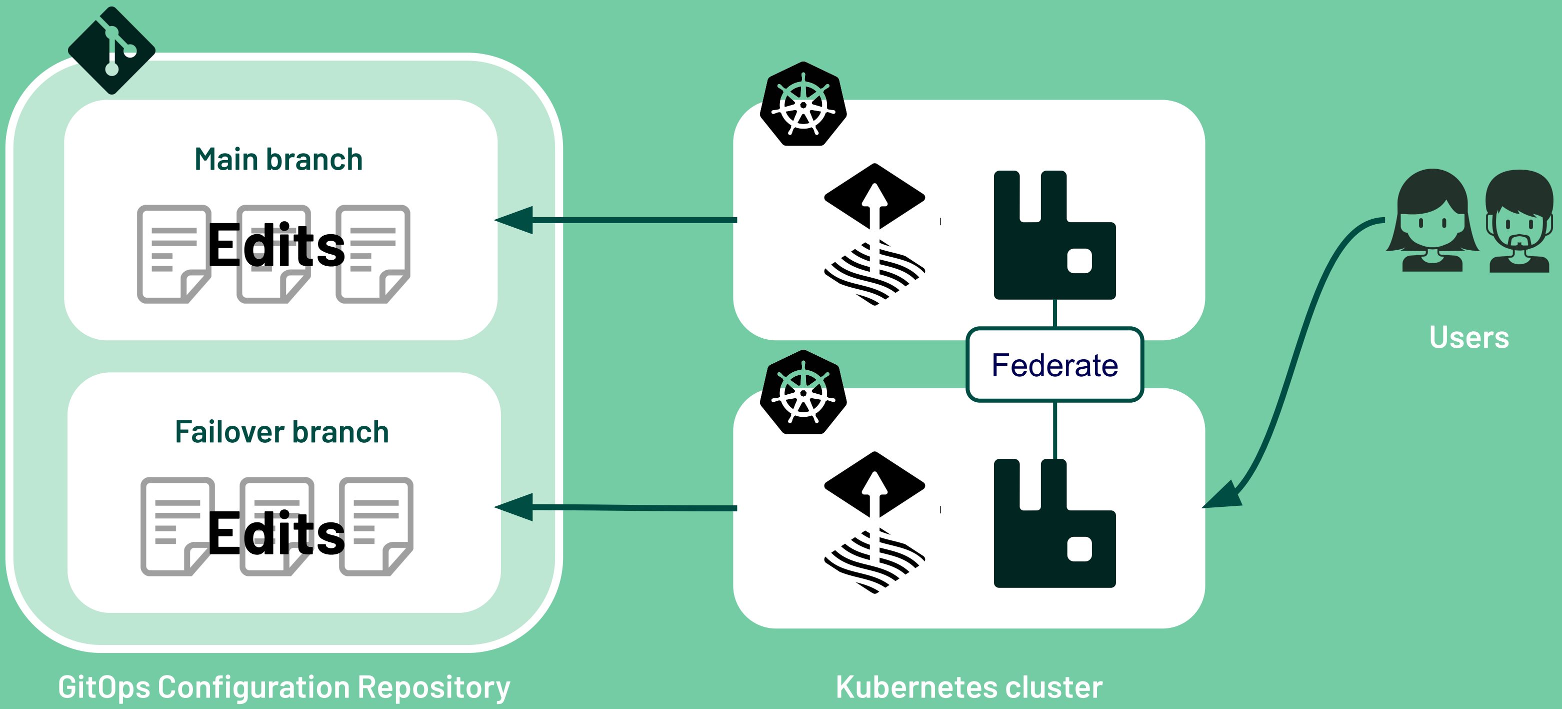
FAILOVER

1. GENERATION



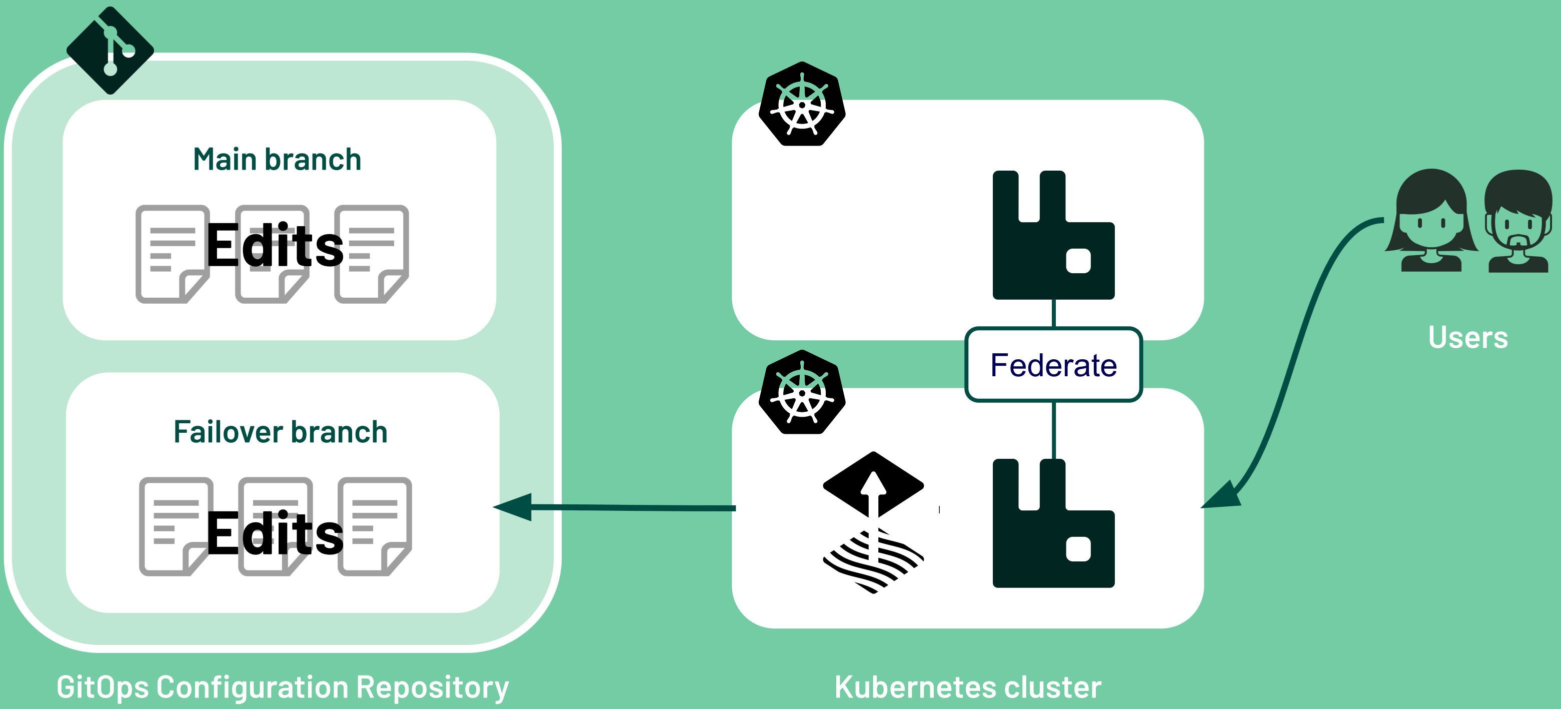
FAILOVER

1. GENERATION



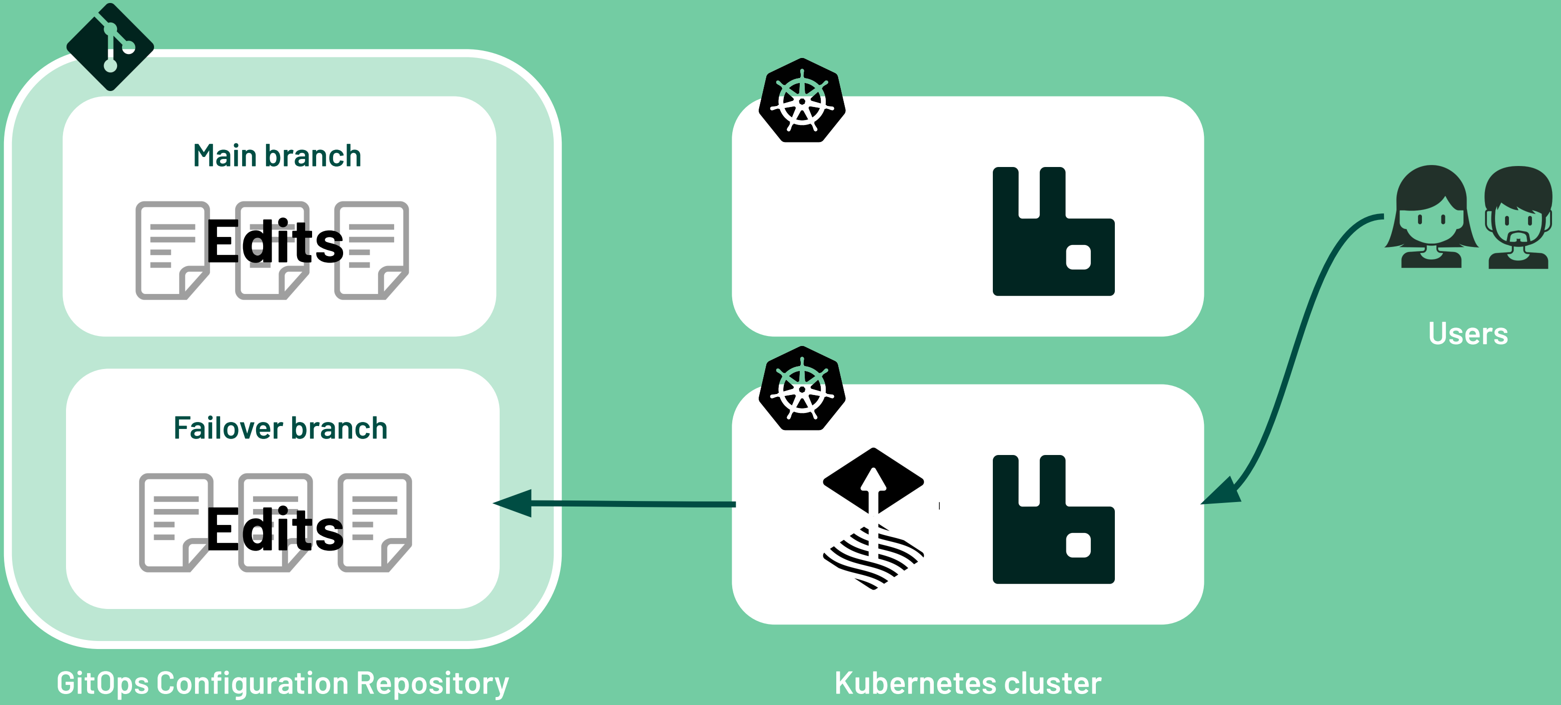
FAILOVER

1. GENERATION



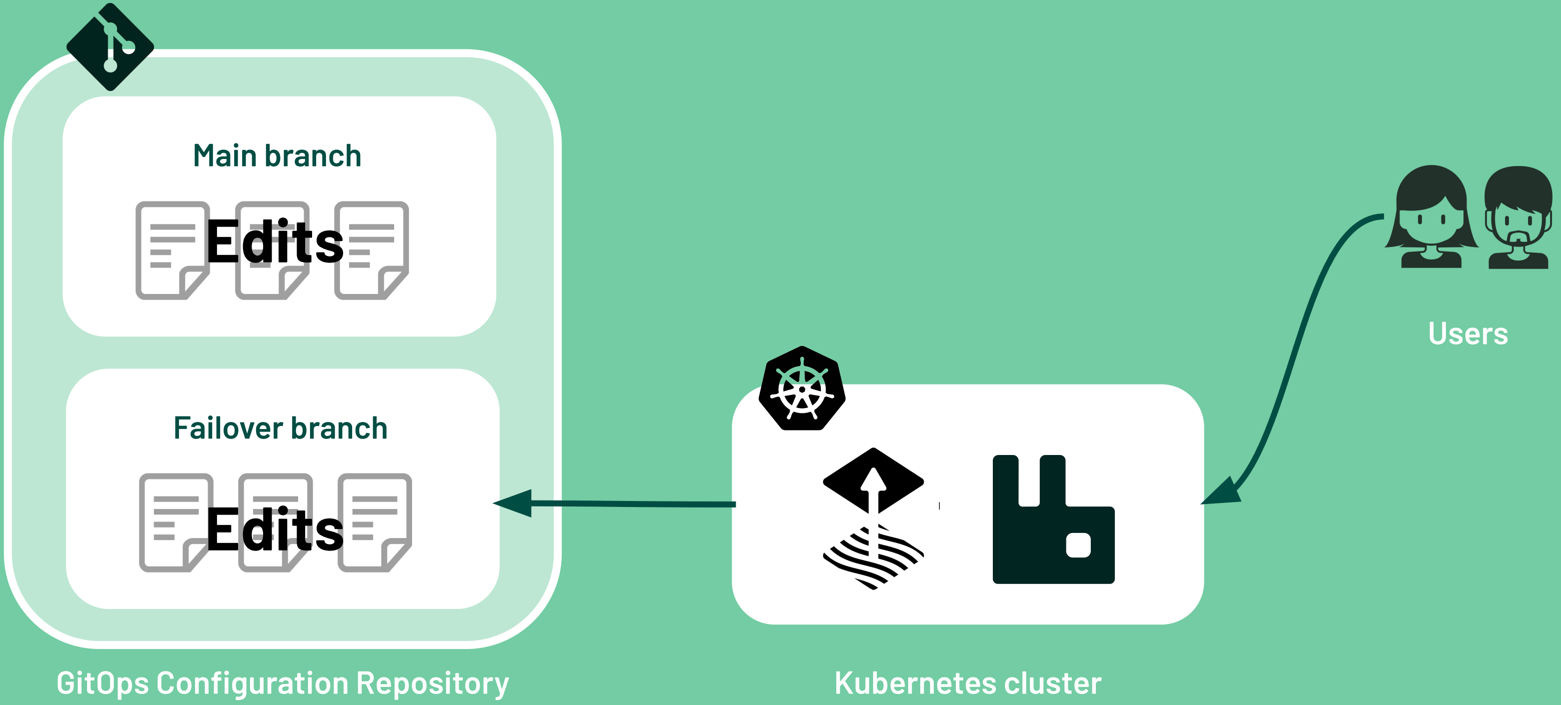
FAILOVER

1. GENERATION



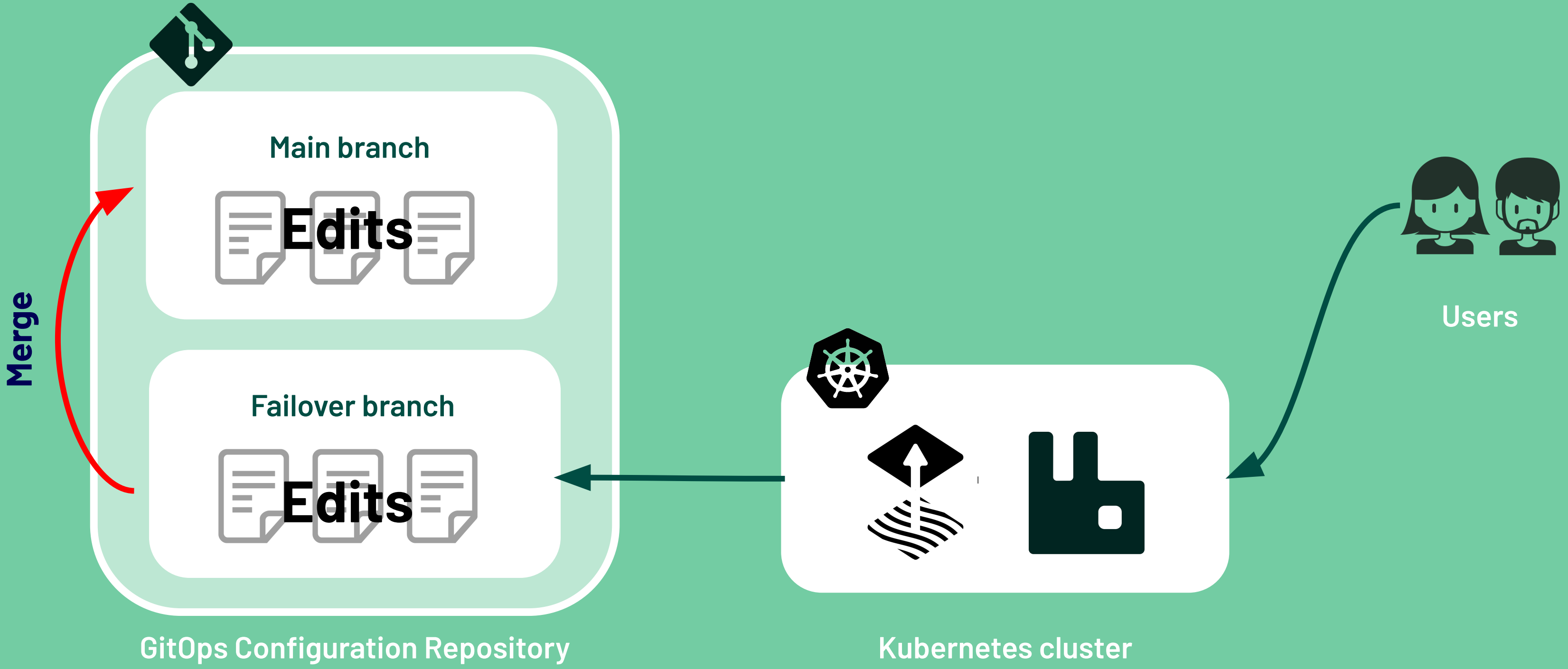
FAILOVER

1. GENERATION



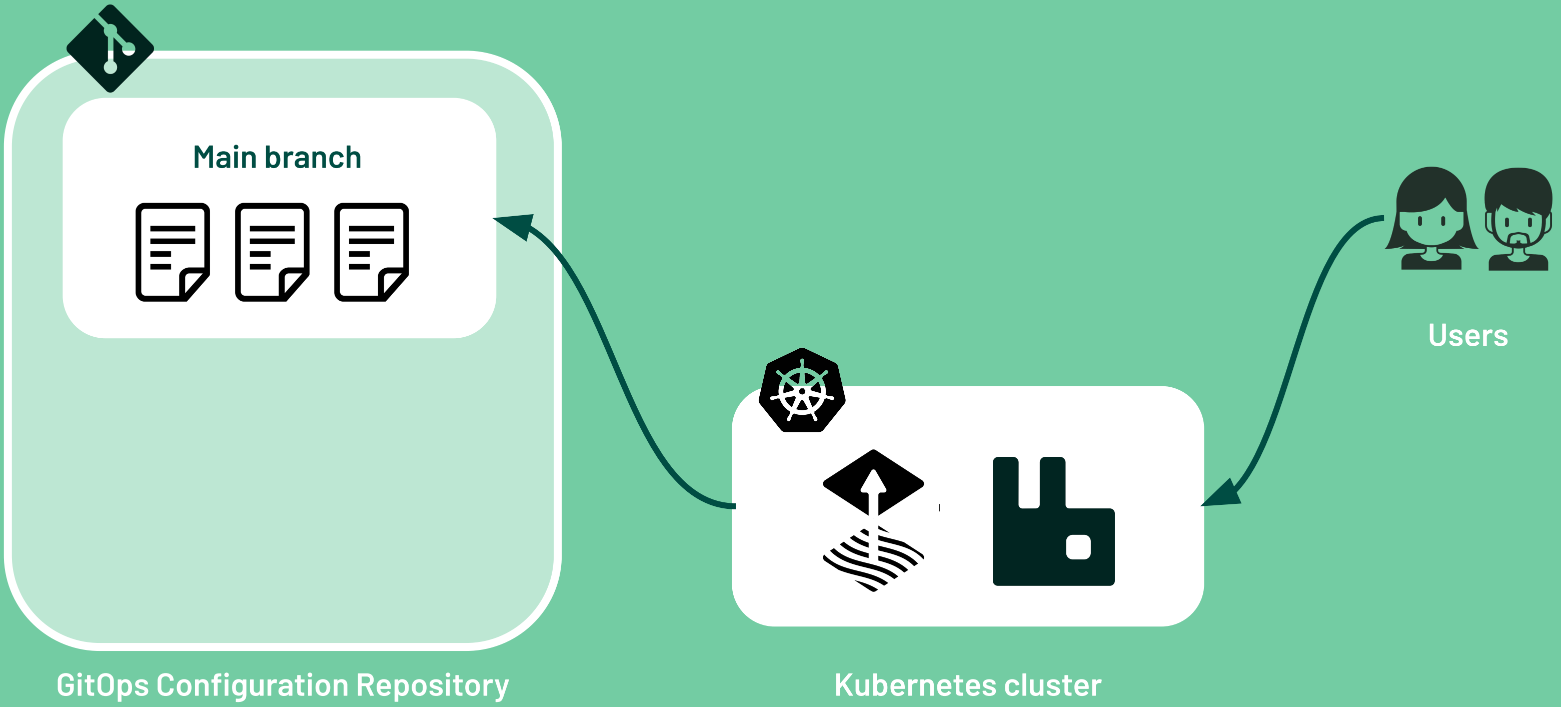
FAILOVER

1. GENERATION



FAILOVER

1. GENERATION



FIRST

GENERATION CHALLENGES

A lot of **merge complexity** in our GitOps repository

New deployments will **stale** if released after branching

Most people **felt uncomfortable** doing a failover

Not in the **spirit of GitOps**

FIRST

GENERATION OBSERVATIONS

Most edits in the GitOps repository was “**cluster name**”

Fluent Bit logs

AWS-iam-authenticator

External DNS annotations

FAILOVER

2. GENERATION

SECOND

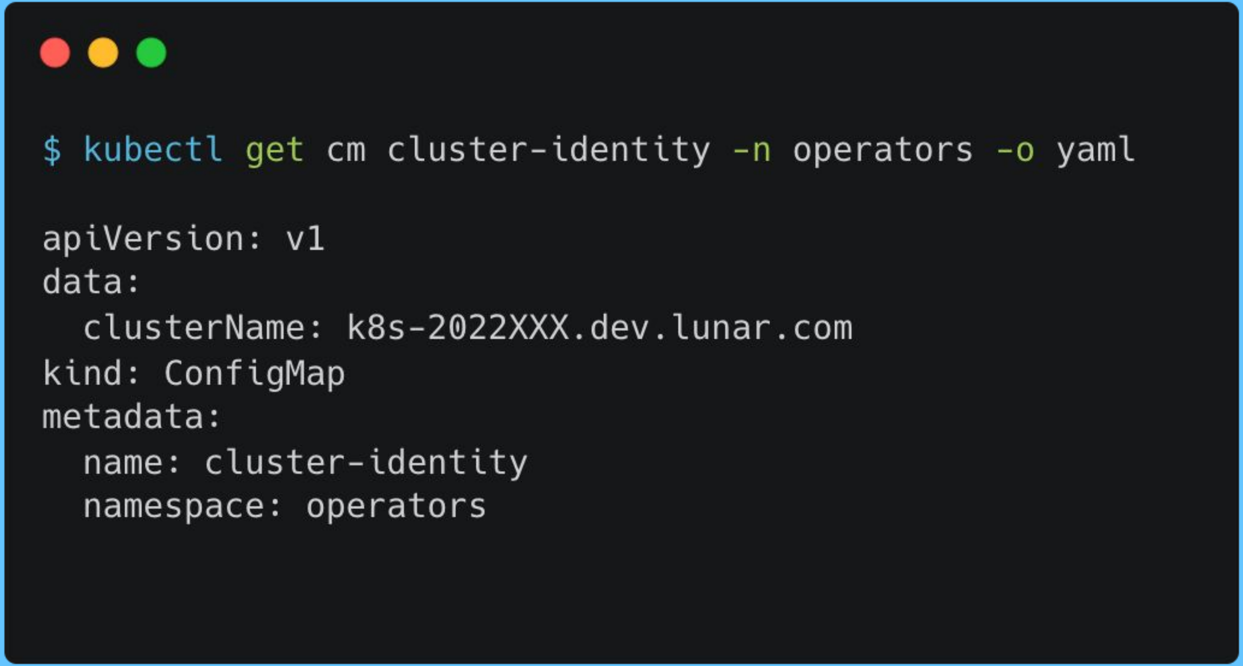
GENERATION - TWO NEW CONTROLLERS

Cluster identity controller

Routing Weight controller

SECOND

GENERATION - CLUSTER IDENTITY

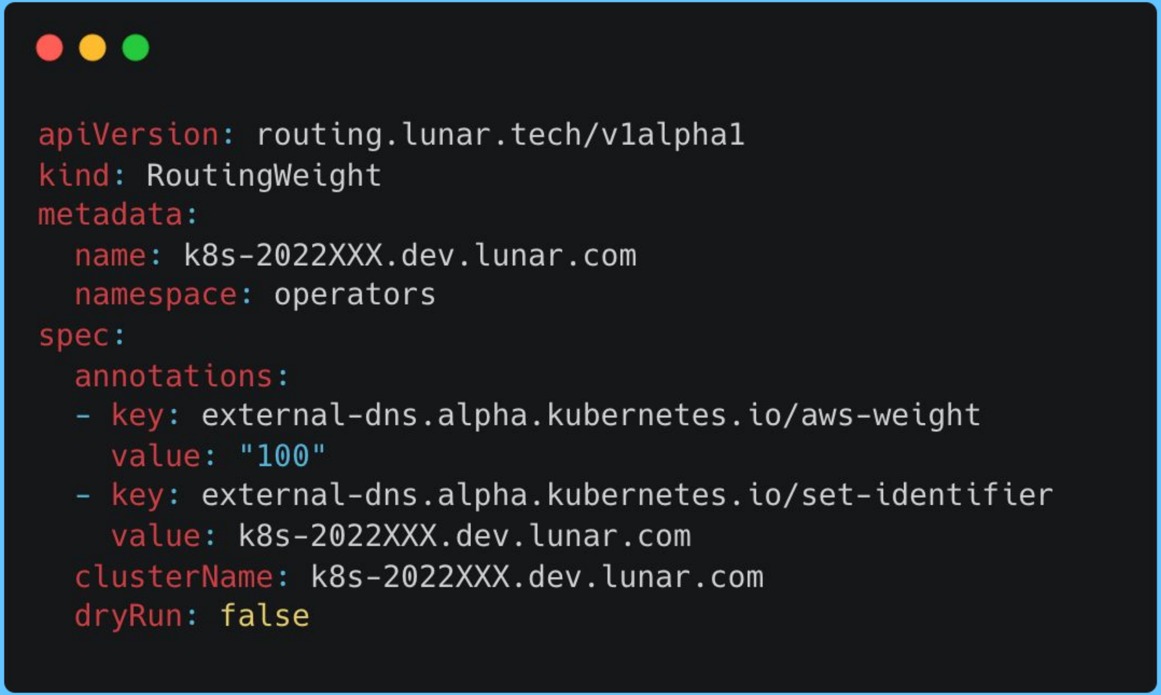


```
$ kubectl get cm cluster-identity -n operators -o yaml

apiVersion: v1
data:
  clusterName: k8s-2022XXX.dev.lunar.com
kind: ConfigMap
metadata:
  name: cluster-identity
  namespace: operators
```

SECOND

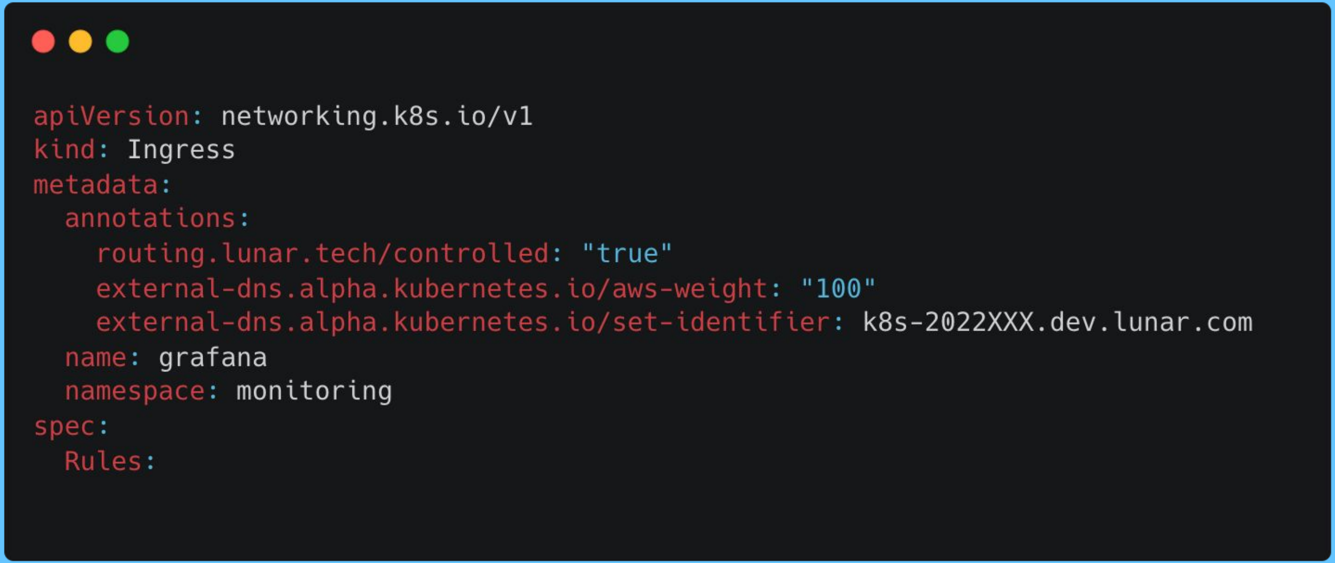
GENERATION - ROUTING WEIGHTS



```yaml
apiVersion: routing.lunar.tech/v1alpha1
kind: RoutingWeight
metadata:
  name: k8s-2022XXX.dev.lunar.com
  namespace: operators
spec:
  annotations:
  - key: external-dns.alpha.kubernetes.io/aws-weight
    value: "100"
  - key: external-dns.alpha.kubernetes.io/set-identifier
    value: k8s-2022XXX.dev.lunar.com
  clusterName: k8s-2022XXX.dev.lunar.com
  dryRun: false
```

SECOND

GENERATION - ROUTING WEIGHTS



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    routing.lunar.tech/controlled: "true"
    external-dns.alpha.kubernetes.io/aws-weight: "100"
    external-dns.alpha.kubernetes.io/set-identifier: k8s-2022XXX.dev.lunar.com
  name: grafana
  namespace: monitoring
spec:
  Rules:
```

SECOND

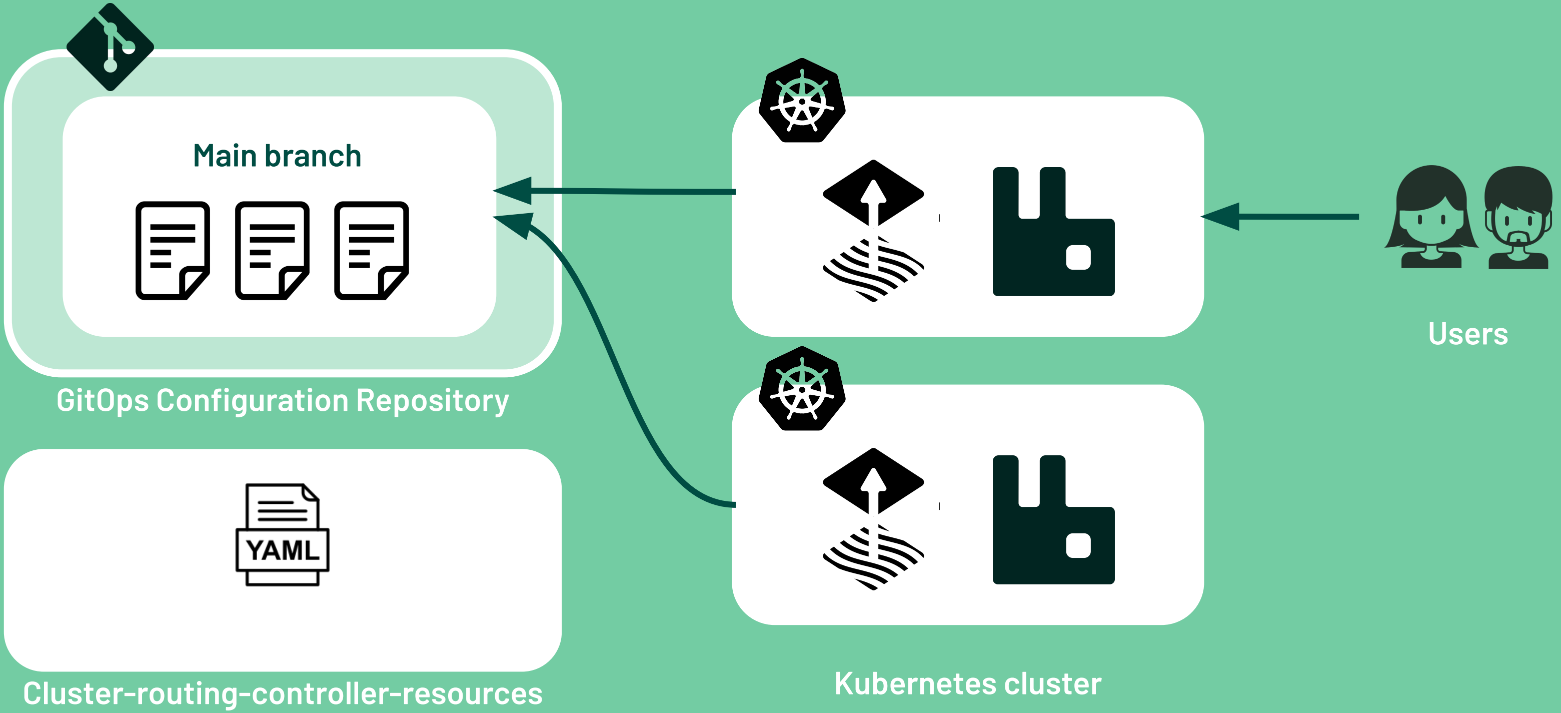
GENERATION - ROUTING WEIGHTS

```
$ shuttle run delete_routing_weight
$ shuttle run add_routing_weight
$ shuttle run adjust_routing_weight
```

```
k8s:
  namespace: operators
  clusters:
    dev:
      weights:
        - clusterName: k8s-2022xx.dev.lunar.com
          routingWeight: 100
```

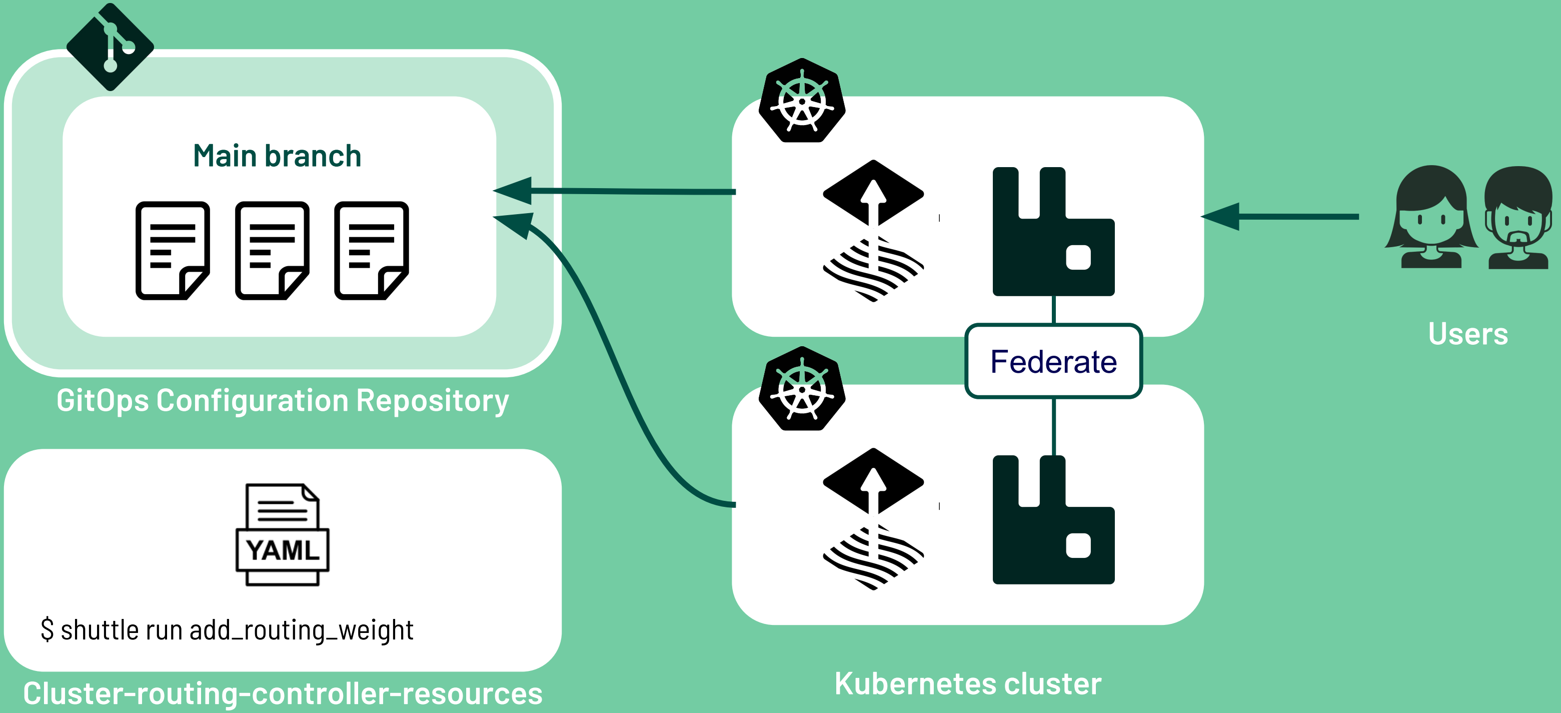
FAILOVER

2. GENERATION


Main branch
GitOps Configuration Repository
YAML
Cluster-routing-controller-resources
Kubernetes cluster
Users

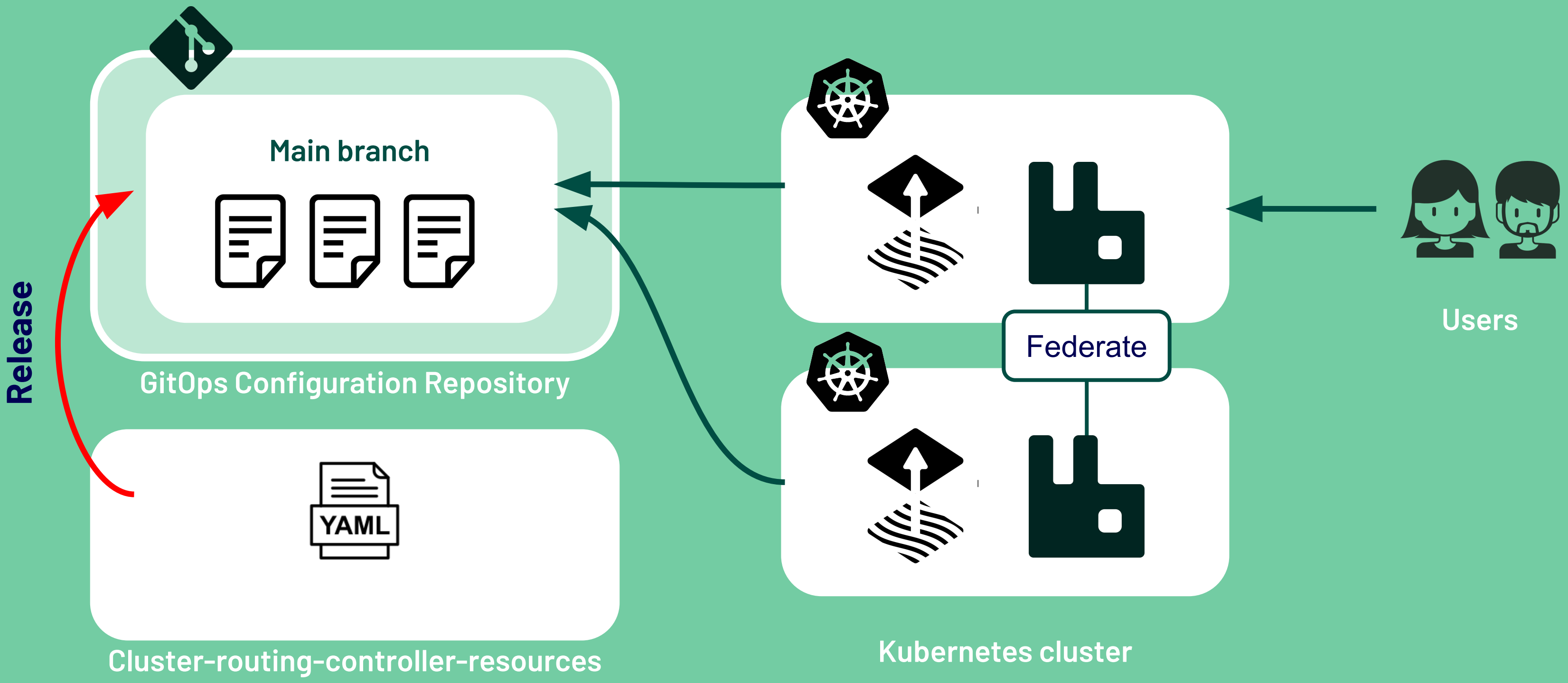
FAILOVER

2. GENERATION


Main branch
GitOps Configuration Repository
YAML
$ shuttle run add_routing_weight
Cluster-routing-controller-resources
Federate
Kubernetes cluster
Users

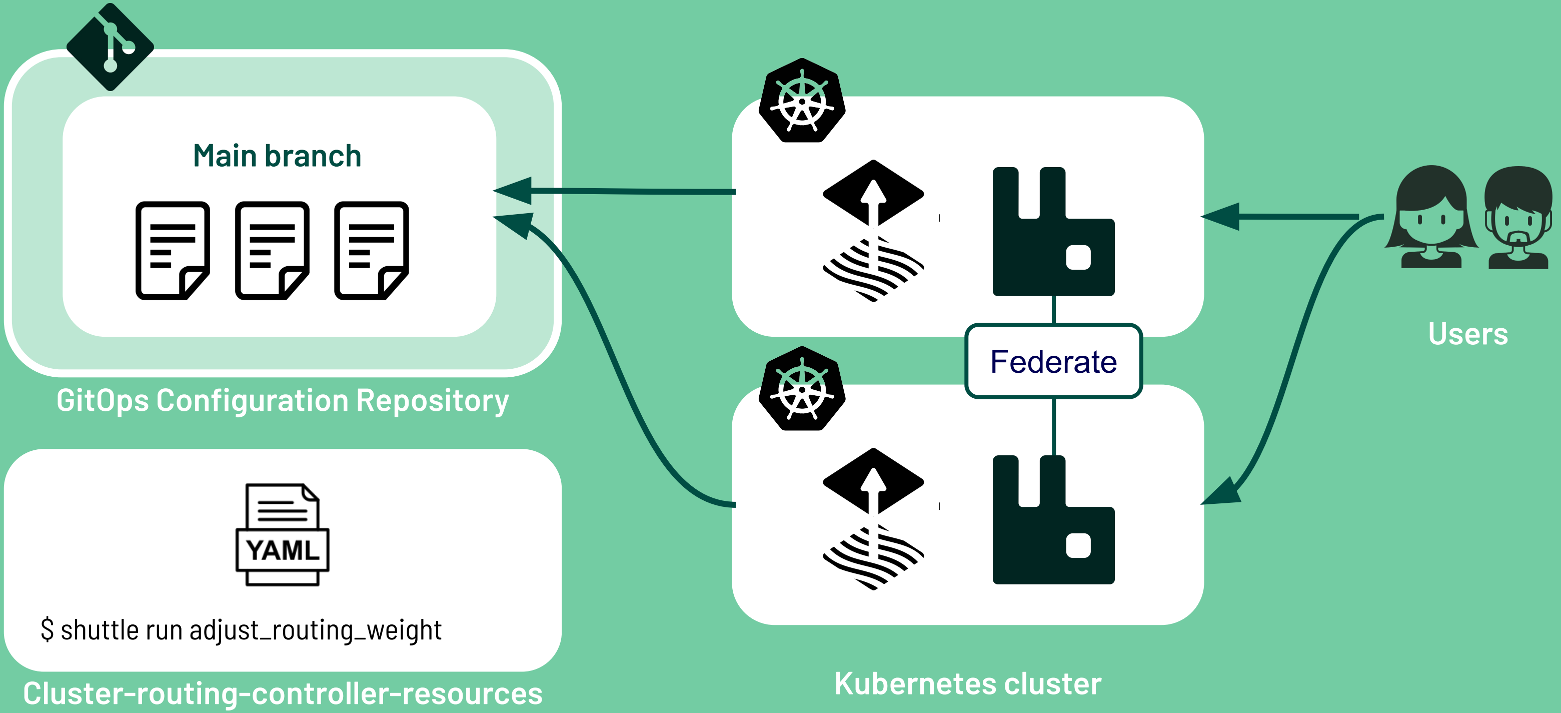
FAILOVER

2. GENERATION


Main branch
GitOps Configuration Repository
Release
YAML
Cluster-routing-controller-resources
Federate
Kubernetes cluster
Users

FAILOVER

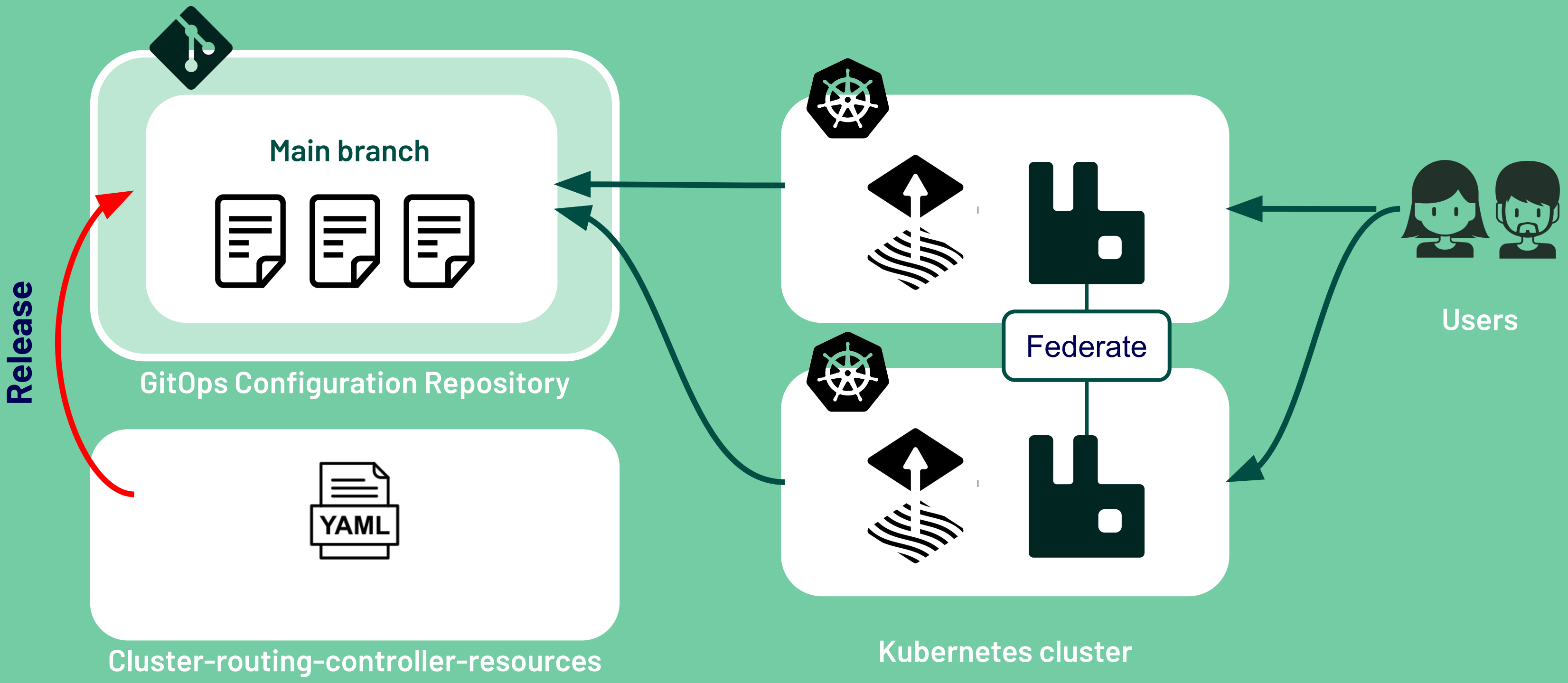
2. GENERATION


Main branch
GitOps Configuration Repository
YAML
Federate
Users
Kubernetes cluster
Cluster-routing-controller-resources

```
$ shuttle run adjust_routing_weight
```

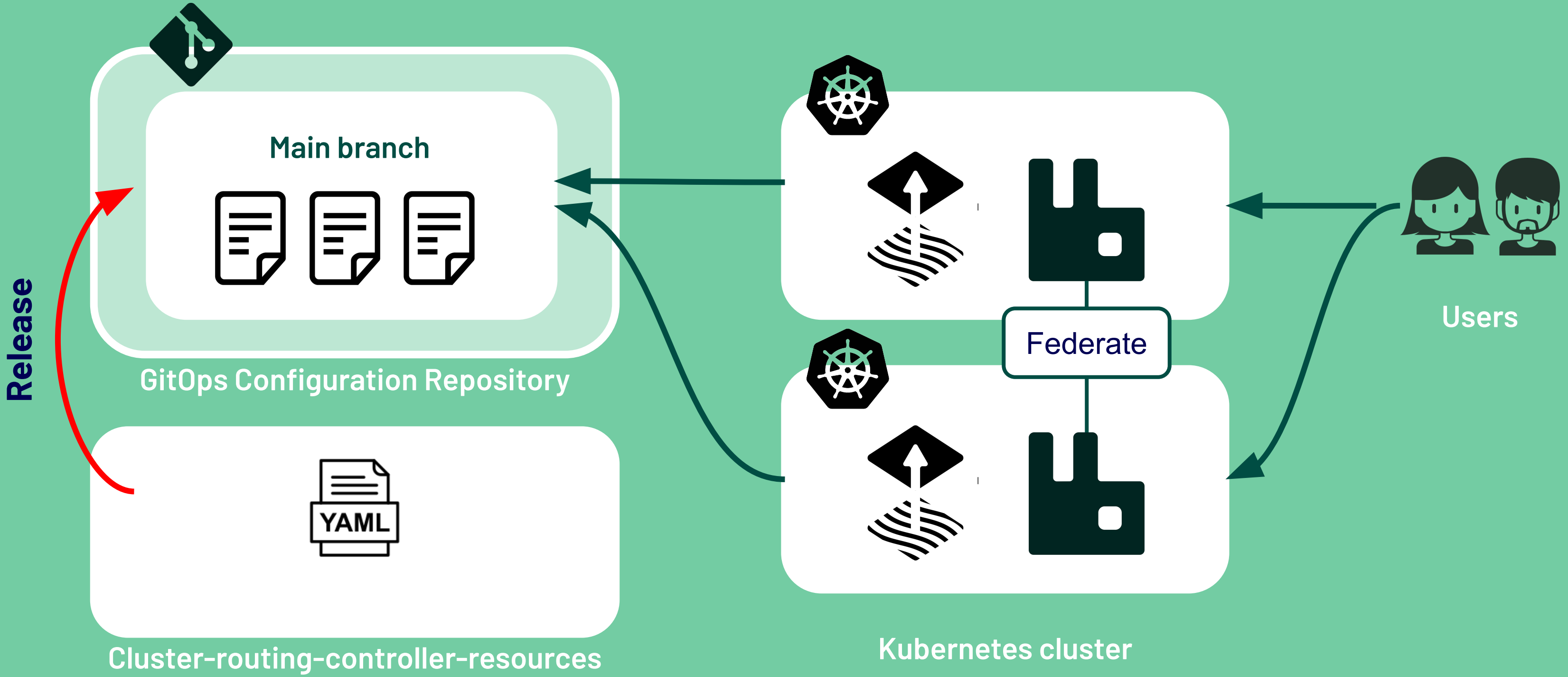
FAILOVER

2. GENERATION

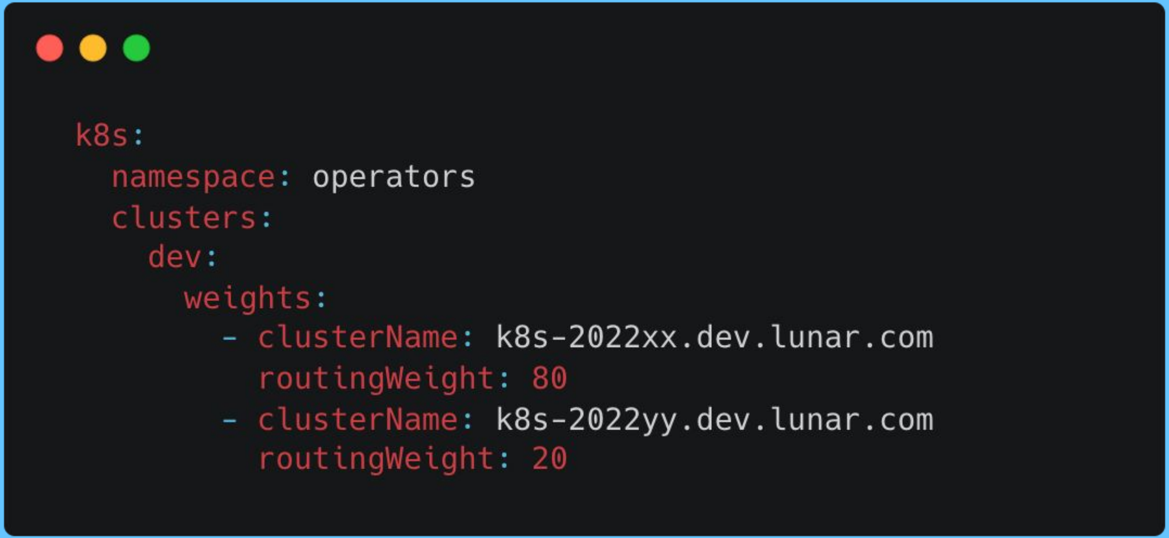

Main branch
GitOps Configuration Repository
Release
YAML
Cluster-routing-controller-resources
Federate
Kubernetes cluster
Users

FAILOVER

2. GENERATION


Main branch
GitOps Configuration Repository
YAML
Cluster-routing-controller-resources
Release
Federate
Kubernetes cluster
Users

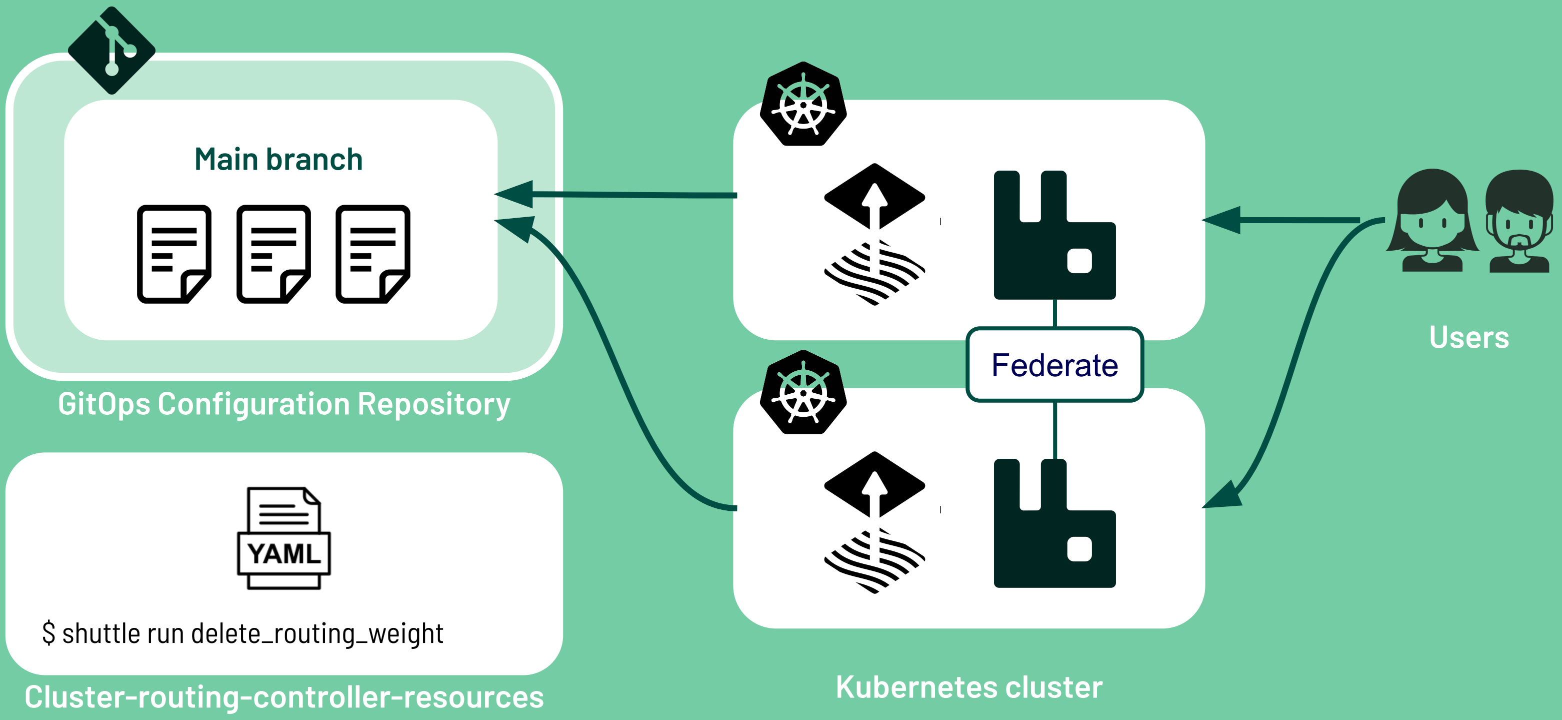
Routing Weight Resources



```yaml
k8s:
  namespace: operators
  clusters:
    dev:
      weights:
        - clusterName: k8s-2022xx.dev.lunar.com
          routingWeight: 80
        - clusterName: k8s-2022yy.dev.lunar.com
          routingWeight: 20
```

FAILOVER

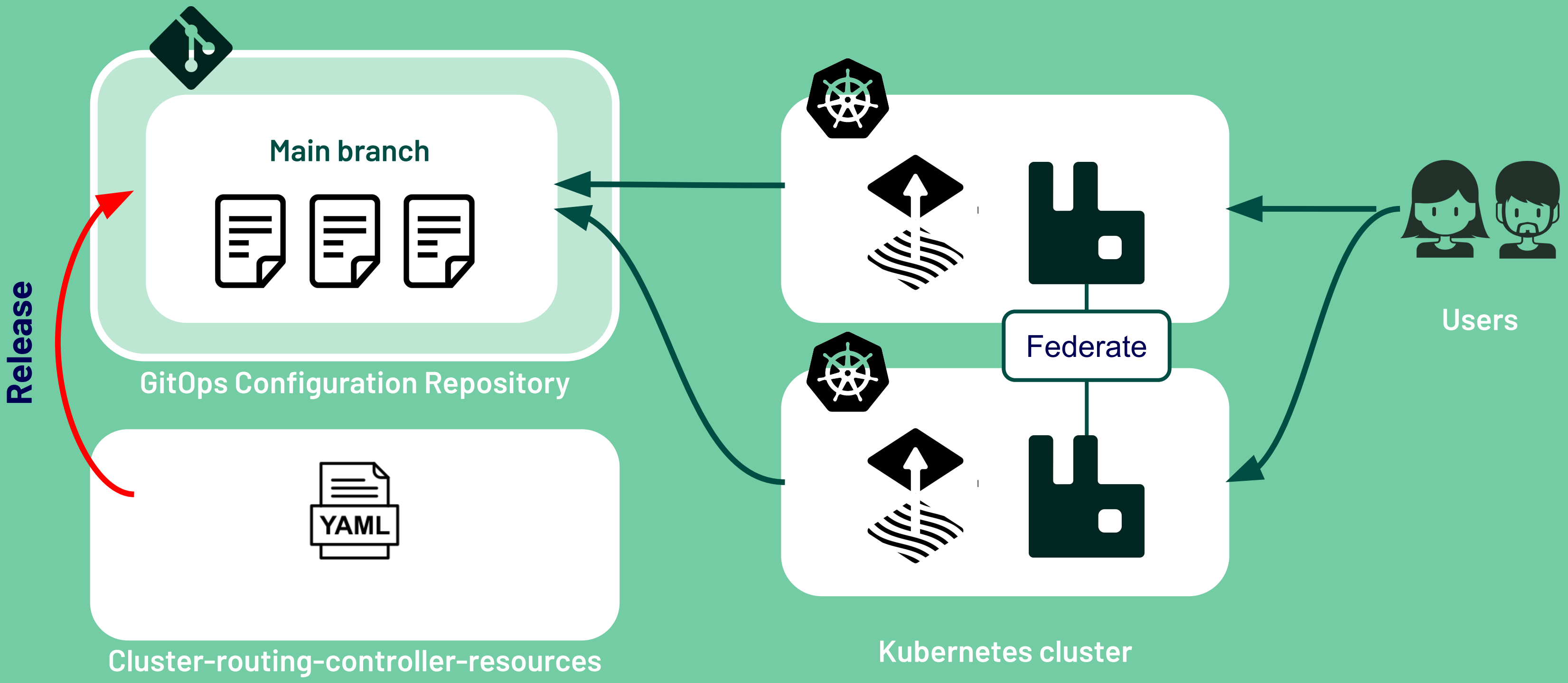
2. GENERATION


Main branch
GitOps Configuration Repository
YAML
Federate
Users
Kubernetes cluster
Cluster-routing-controller-resources

```
$ shuttle run delete_routing_weight
```

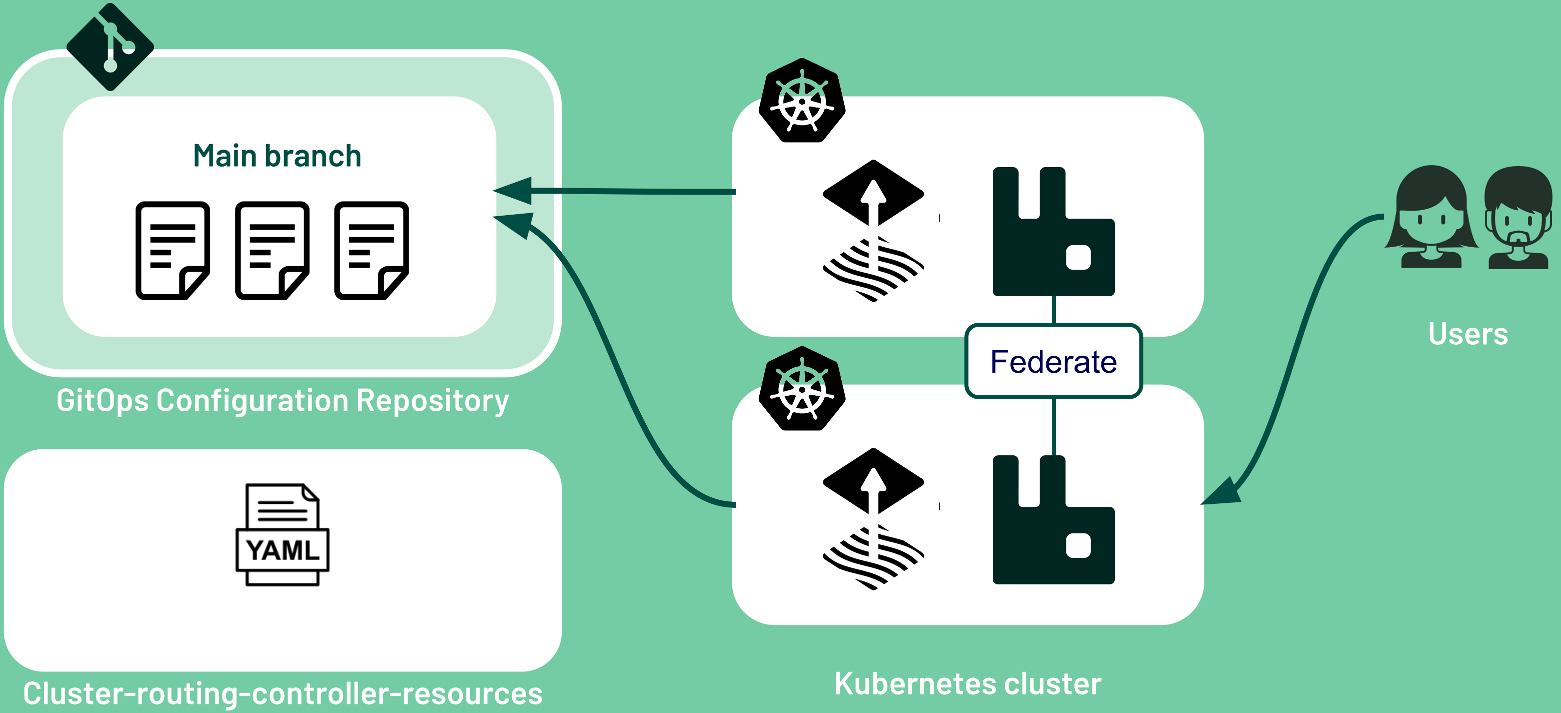
FAILOVER

2. GENERATION


Main branch
GitOps Configuration Repository
Release
YAML
Cluster-routing-controller-resources
Federate
Kubernetes cluster
Users

FAILOVER

2. GENERATION


Main branch
GitOps Configuration Repository
YAML
Cluster-routing-controller-resources
Federate
Kubernetes cluster
Users

SECOND

GENERATION EFFORT

Coding two Kubernetes operators

17 failover runs in Dev in a 2 month time period

Every iteration let to improvements

FIRST

GENERATION RESULTS

Everyone in Squad Odyssey can do a failover in production

The failover operation is down to 5 automated steps

No GitOps branching

From spending 4 hours on a failover to 40 minutes !

FIRST

GENERATION PROBLEMS

Our Cluster Identity controller has to have “strategies” to find the clusterName.

FUTURE

GOALS

Use dependency feature in Flux v2

Migrate from Kops to ClusterApi

We want to migrate our Terraform to Crossplane

FIRST

GENERATION RESULTS

Plan

Do

Study

Act

Open source links :

Shuttle: **github.com/lunarway/shuttle**
Release-manager: **github.com/lunarway/release-manager**

Contacts :

LinkedIn: **linkedin.com/in/hoeghh**
Twitter: **twitter.com/HenrikHoegh**
E-Mail: **her@lunar.app**

THANK YOU

LUNAR®

LUNAR®