GitOps Reinvented

Correcting Kubernetes Drift Before It Happens

Damien Dassieu

GitOps principles



Declarative



Versioned and Immutabled



Pulled automatically

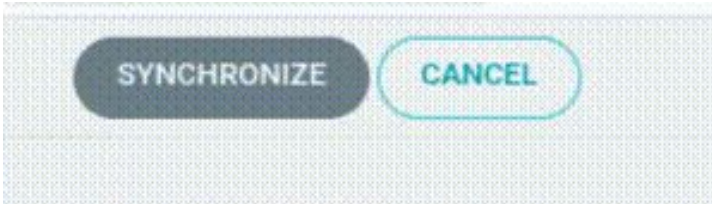


Continuously reconciled

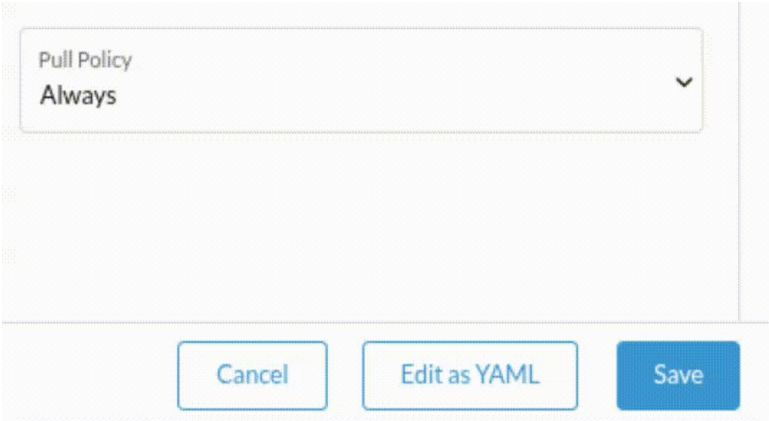
Why do we need to reinvent GitOps?

What about real world production clusters?

Auto-reconciliation disabled



Execute actions through an UI



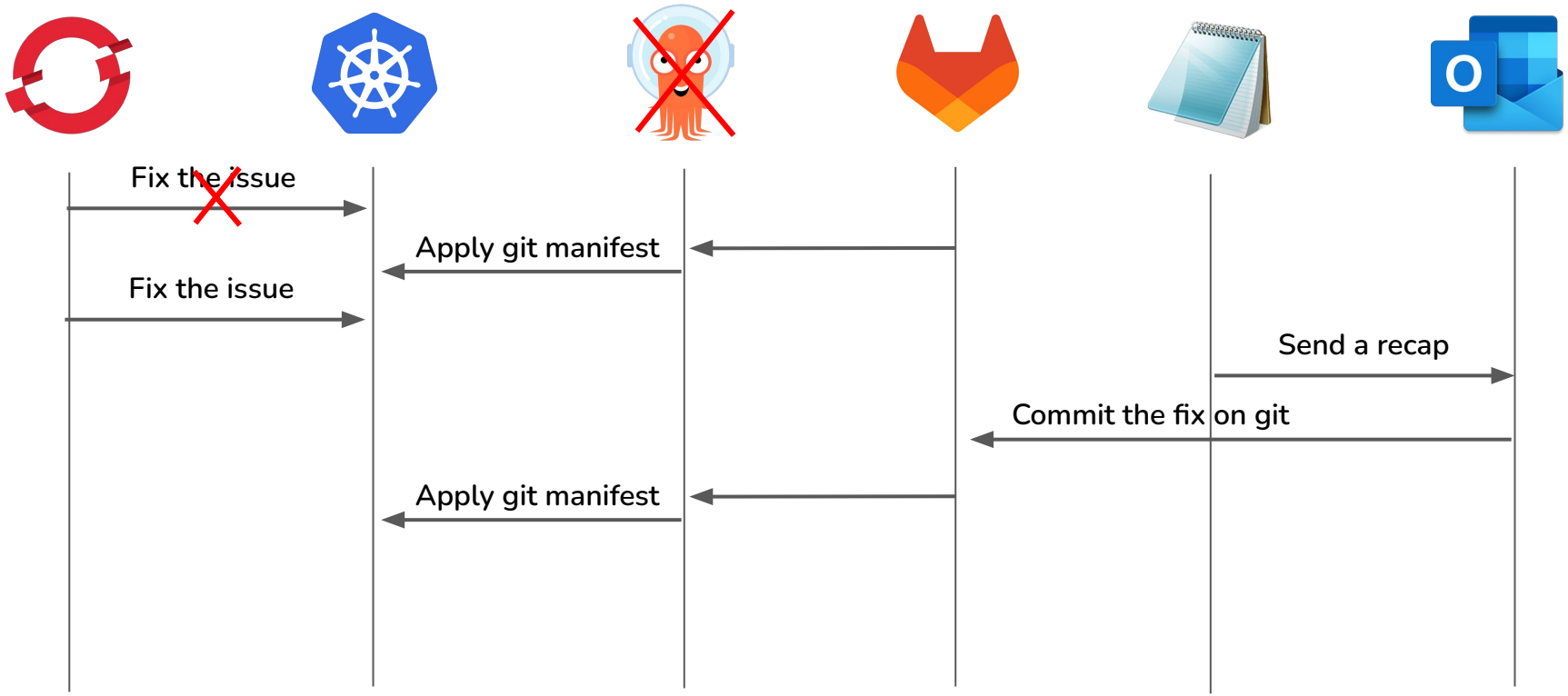
Execute actions with kubectl

```
damien@arch ~> kubectl scale deployment my-deployment --replicas=2
```

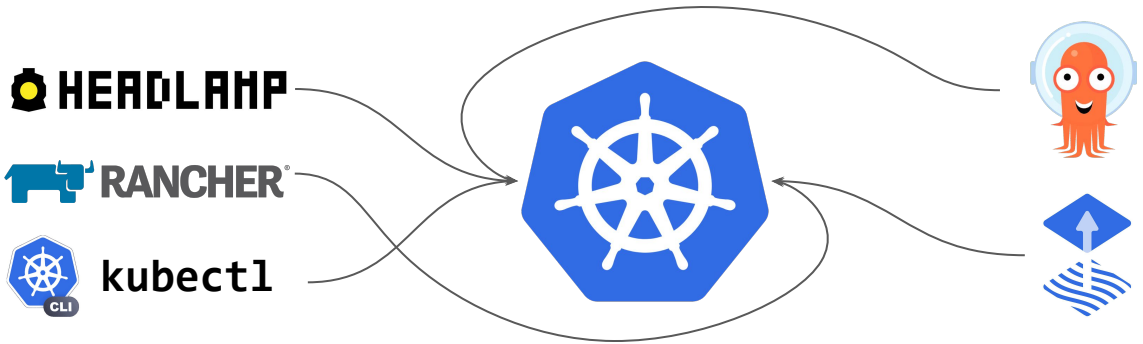
YAML files of 1000+ lines

```yaml
spec:
---
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: InfrastructureMachineTemplate
metadata:
  name: prod-controlplane-machinetemplate
  namespace: default
spec:
  template:
    spec:
      providerID: "controlplane-template-id"
---
apiVersion: bootstrap.cluster.x-k8s.io/v1beta1
kind: KubeadmConfigTemplate
metadata:
  name: prod-worker-bootstrap-template
  namespace: default
spec:
  template:
    spec:
      joinConfiguration:
        nodeRegistration:
          kubeletExtraArgs:
            cloud-provider: external
---
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: InfrastructureMachineTemplate
metadata:
  name: prod-worker-machinetemplate
  namespace: default
spec:
  template:
    spec:
      providerID: "worker-template-id"
```

Emergency case example



ClickOps & GitOps
mix and twist



How can we bridge this gap?

Solution 1

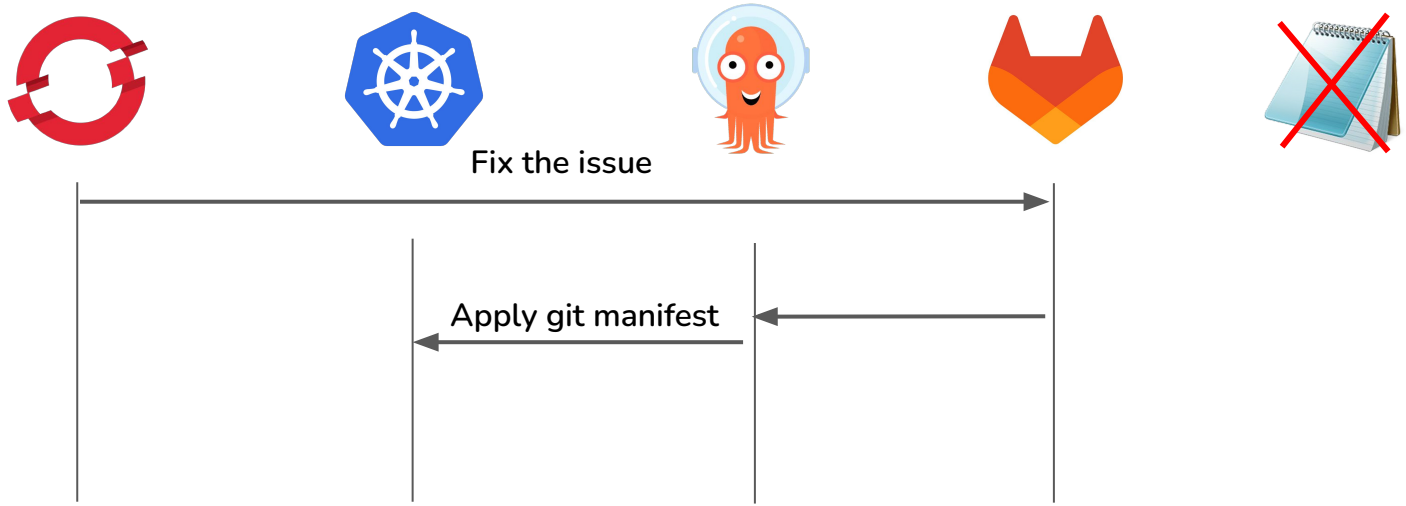
Force users to only use GitOps



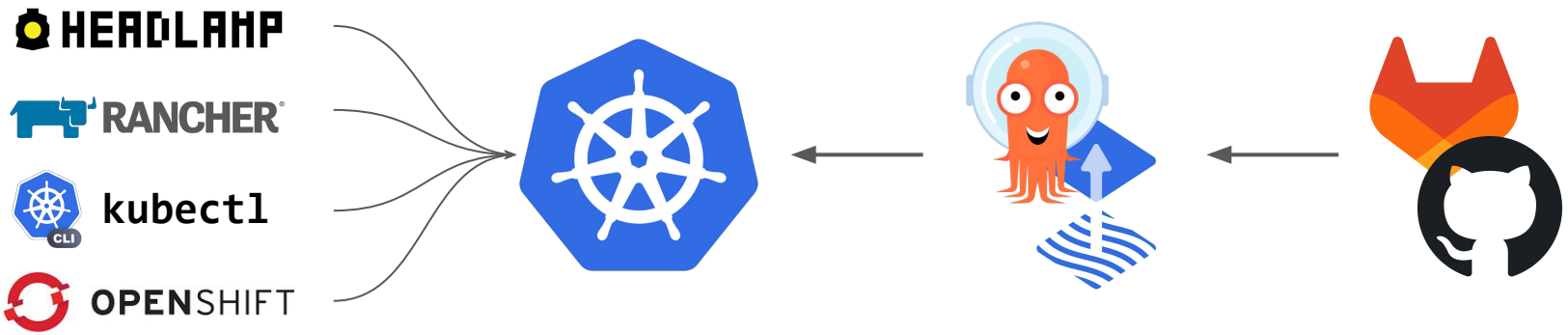
Solution 2
Merge ClickOps and GitOps



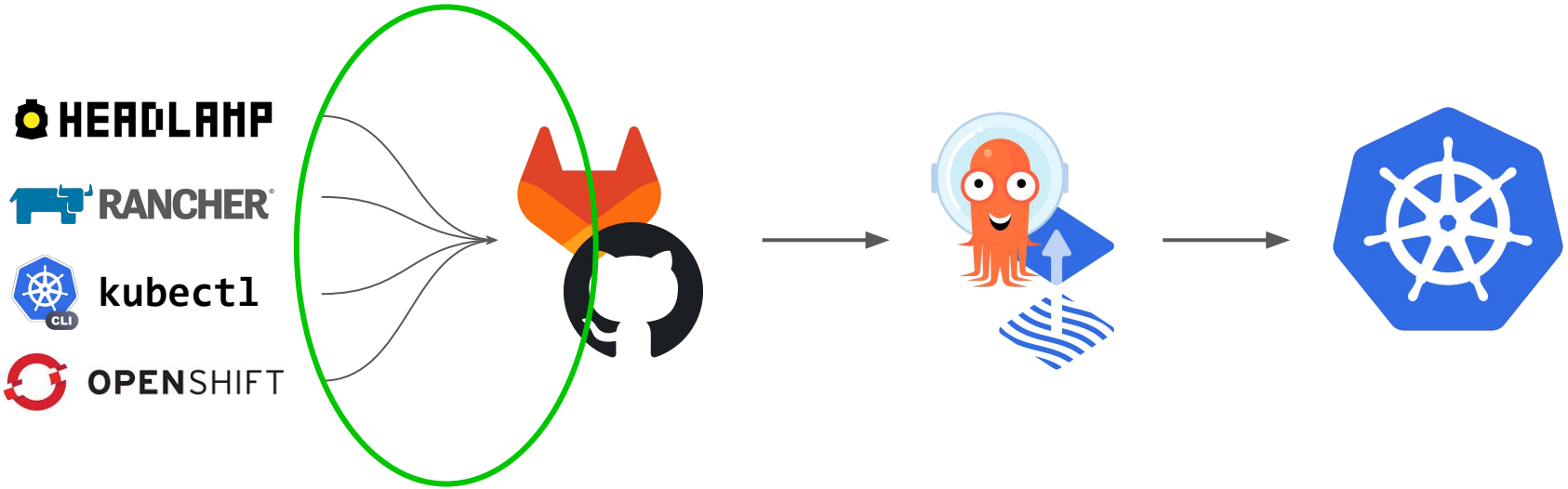
Emergency case example



How to merge ClickOps and GitOps

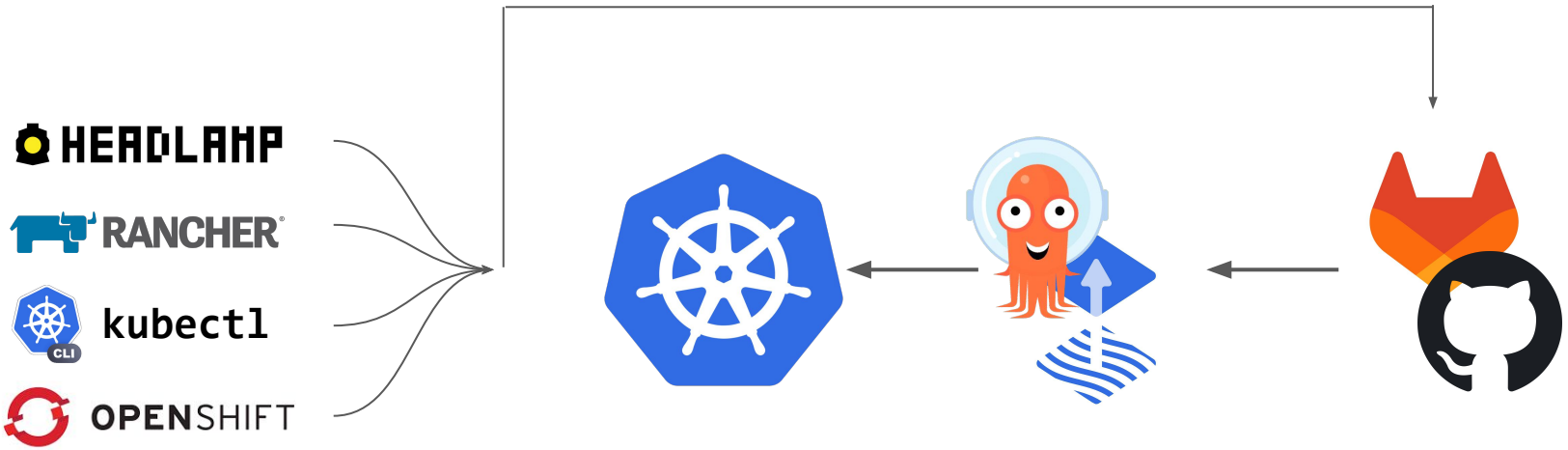


How to merge ClickOps and GitOps

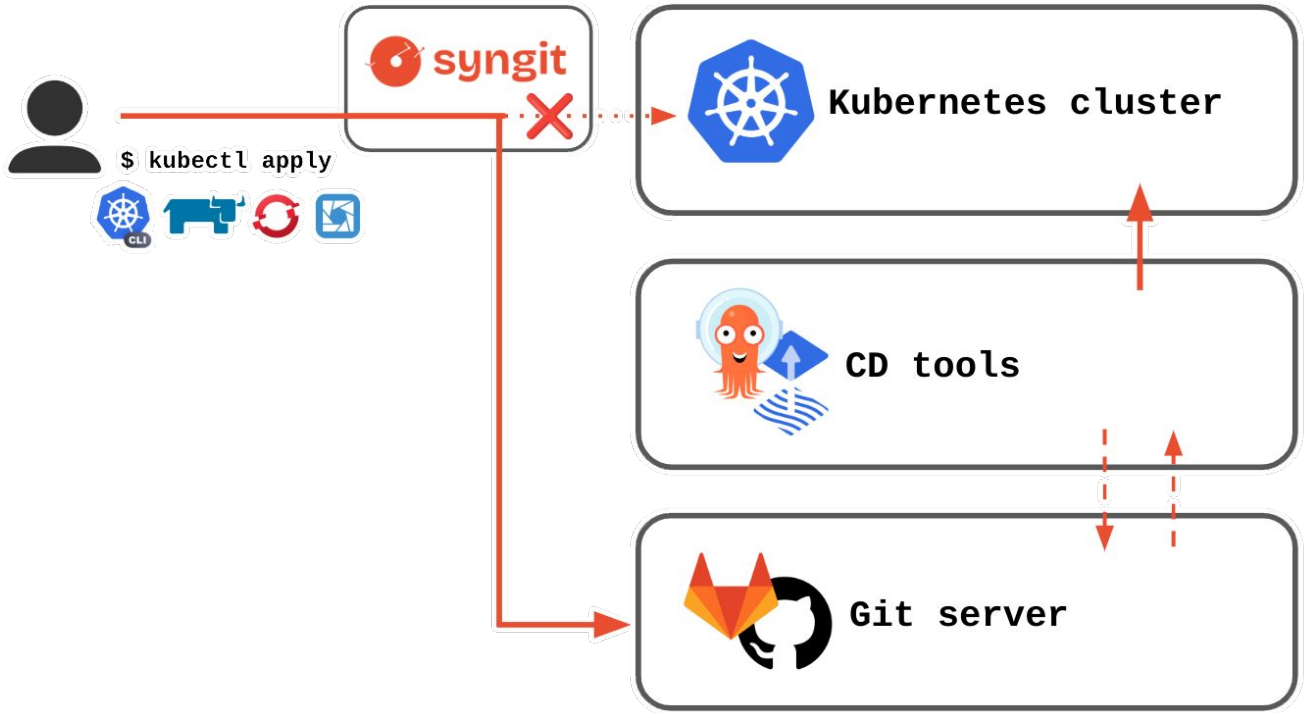


Should we make an extension for each of them?

How to merge ClickOps and GitOps



Syngit

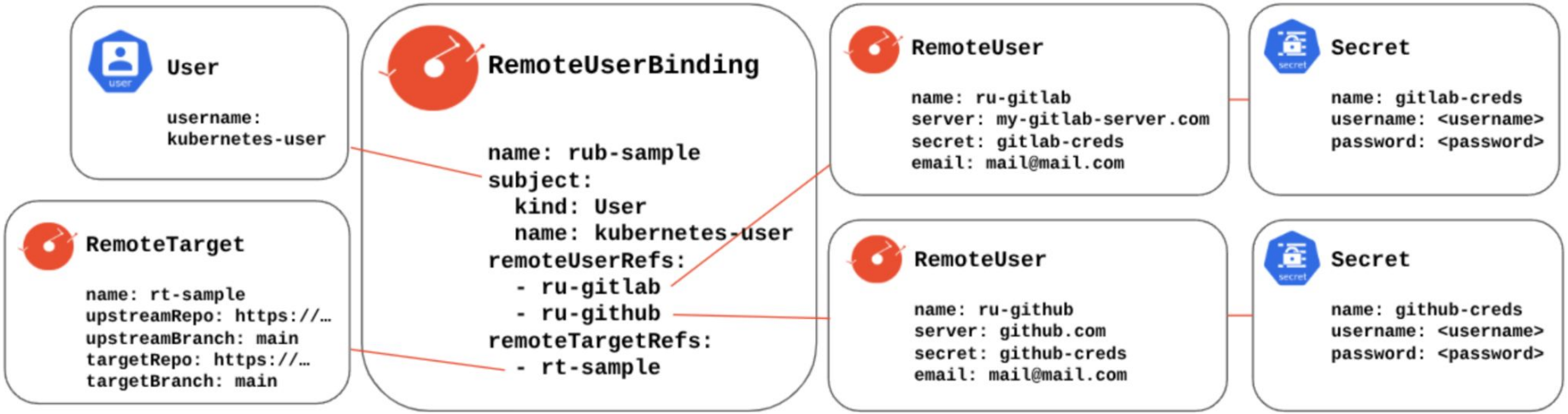

syngit
$ kubectl apply
Kubernetes cluster
CD tools
Git server

How does it work?

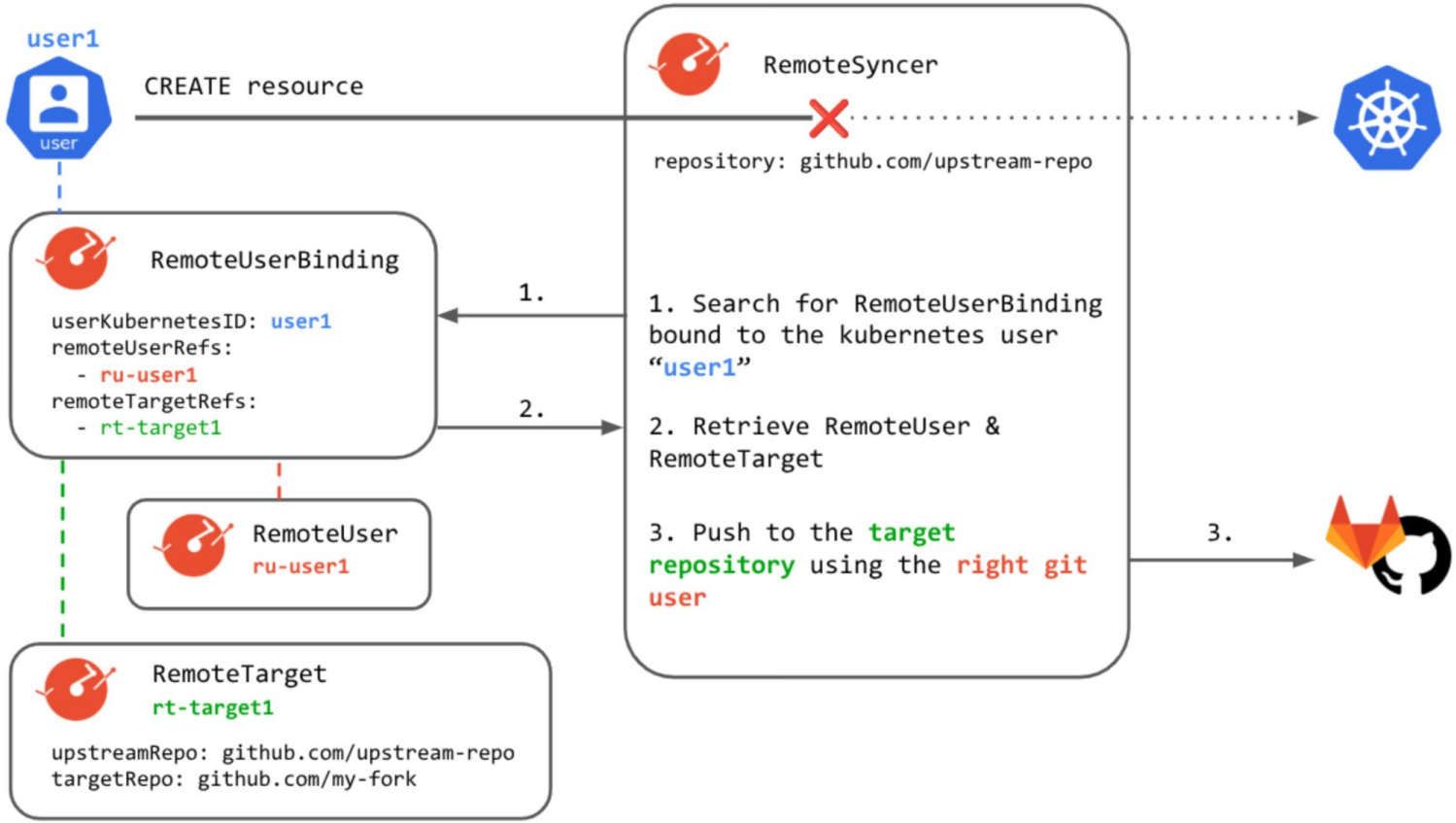
4 new CRDs

`RemoteUser` - Representation of the git user
`RemoteUserBinding` - Bind between the git user and the Kubernetes user
`RemoteSyncer` - Scope of the resources and define the strategy
`RemoteTarget` - Define the target repo and branch

RemoteUser - RemoteUserBinding - RemoteTarget



Full process



Demo time 🎉

Deep dive in the configuration ⛏️

Can we go further?

Dynamic targets - Done

- `syngit.io/remotetarget.pattern.user-specific: one-user-one-branch`
- `syngit.io/remotetarget.pattern.user-specific: one-user-one-fork`

RBAC management - Done

Git providers for Syngit

- Test user authentication - Done
- Fork auto-creation - In progress
- PR / MR auto-creation - In progress

Push on multiple branches at the same time - Done

Any questions?



`syngit-org/syngit`

if you like it, please leave a ⭐ ;)

Sources

OpenGitOps: `https://opengitops.dev/`

Tools:

- Kubernetes
- Syngit
- ArgoCD
- FluxCD
- Rancher
- kubectl
- Openshift
- Gitlab
- Github