Configuring Flux for Automated Deployments

Nigel Brown

@n_brownuk www.windsock.io

Module Outline

Coming up:

- Getting alerted through notifications
- Establishing a trusted identity
- Patterns for structuring repositories
- Defining sources and kustomizations
- Automating an application deployment

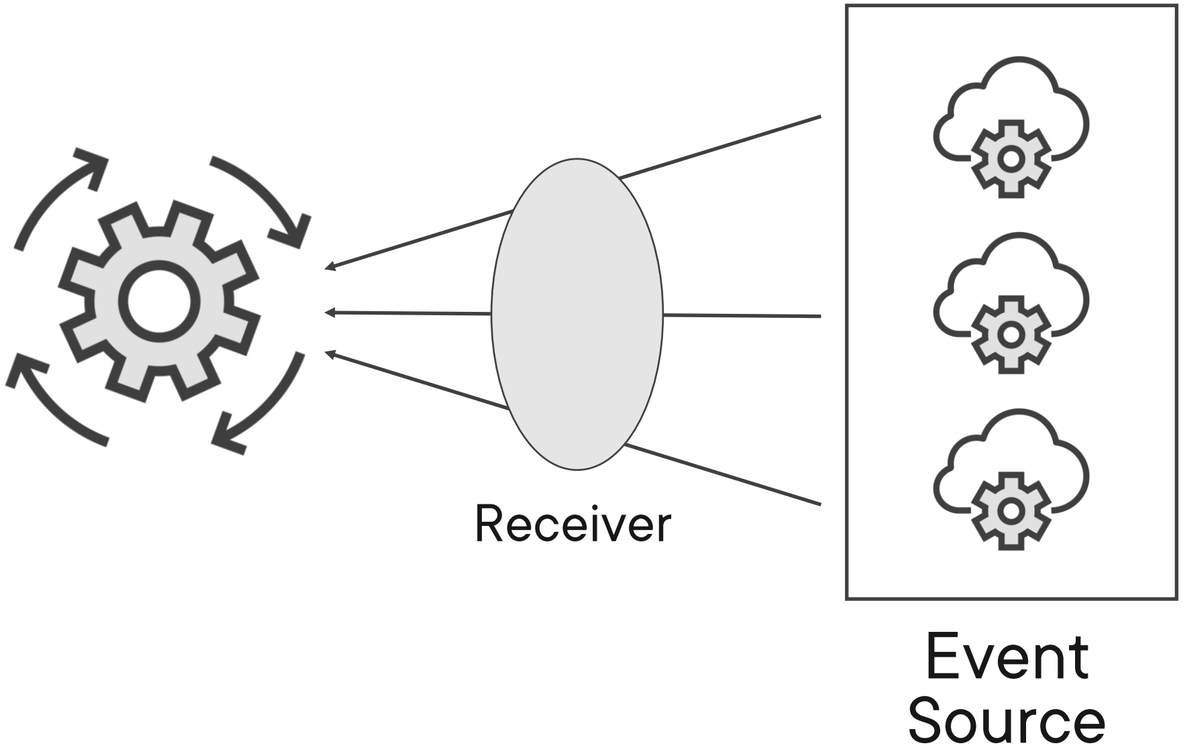
GitOps Visibility

With automation playing a key role in GitOps solutions, how do we know what is happening inside a cluster? Notifications provide visibility.

Notification Controller

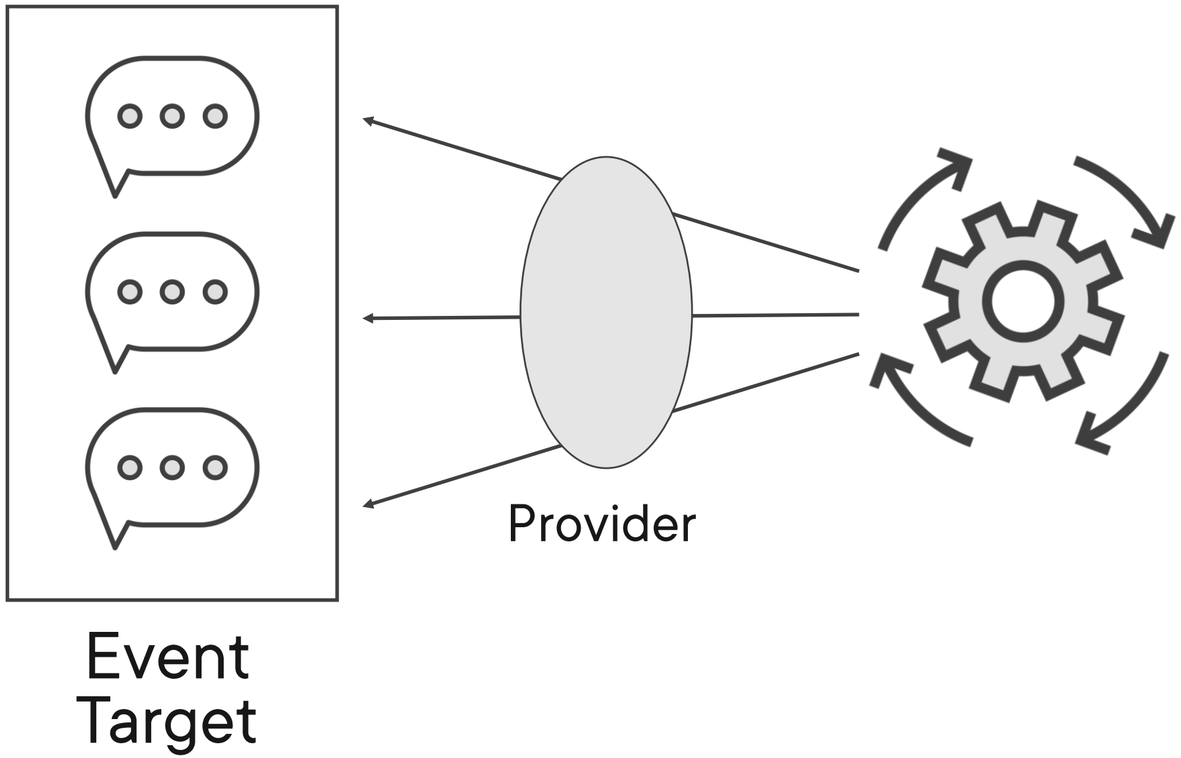
Receives events from external sources and components of the GitOps Toolkit, and dispatches events to external communications platforms.

Event Receivers



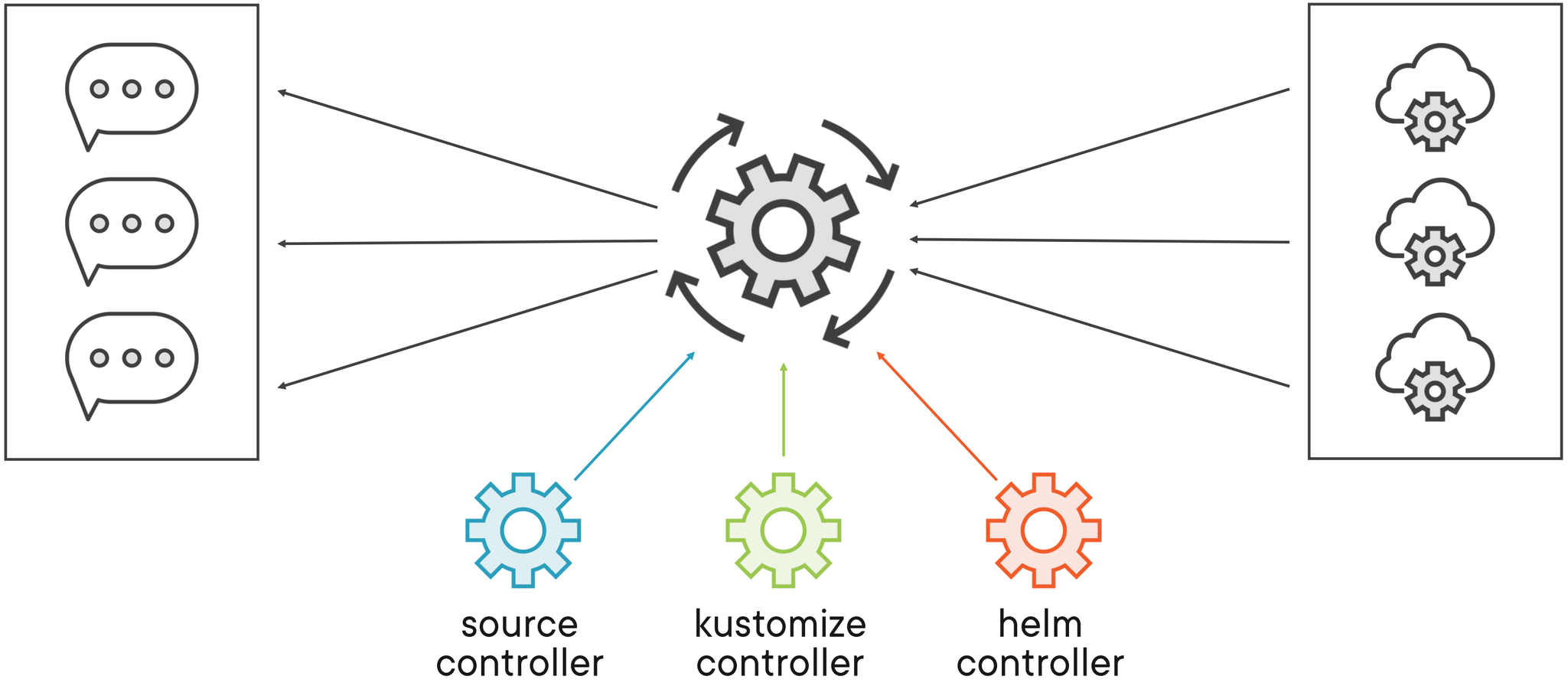
Receivers: GitHub, GitLab, Bitbucket, Gitea, Docker Hub, Harbor, Nexus ...

Event Providers



Providers: MS Teams, Google Chat, Slack, Discord, Matrix, Rocket, WebEx ...

Controller Events



Provider API

```yaml
---
apiVersion: notification.toolkit.fluxcd.io/v1beta1
kind: Provider
metadata:
  name: discord
  namespace: default
spec:
  type: discord
  channel: gitops-lab
  username: FluxBot
  secretRef:
    name: discord
```

```
$ kubectl create secret generic discord \
    --from-file=address="${PWD}/discord-webhook-address“
secret/discord created
```

Secret Reference for Provider

Contains the provider webhook URL

- `address=https://discord.com/api/webhooks/xxxxxxxxxxxxxxxxxx/xxxxx...`

Alert API

```
---
apiVersion: notification.toolkit.fluxcd.io/v1beta1
kind: Alert
metadata:
  name: discord-bot-alert
  namespace: default
spec:
  providerRef:
    name: discord
  eventSources:
  - kind: GitRepository
    name: '*'
  - kind: Kustomization
    name: ‘*’
  eventSeverity: info
```

Demo

Setting Up Alerting for Discord

- Create a bot with a webhook URL
- Establish Provider and Alert resources
- Use GitHub to store new configuration

Trust

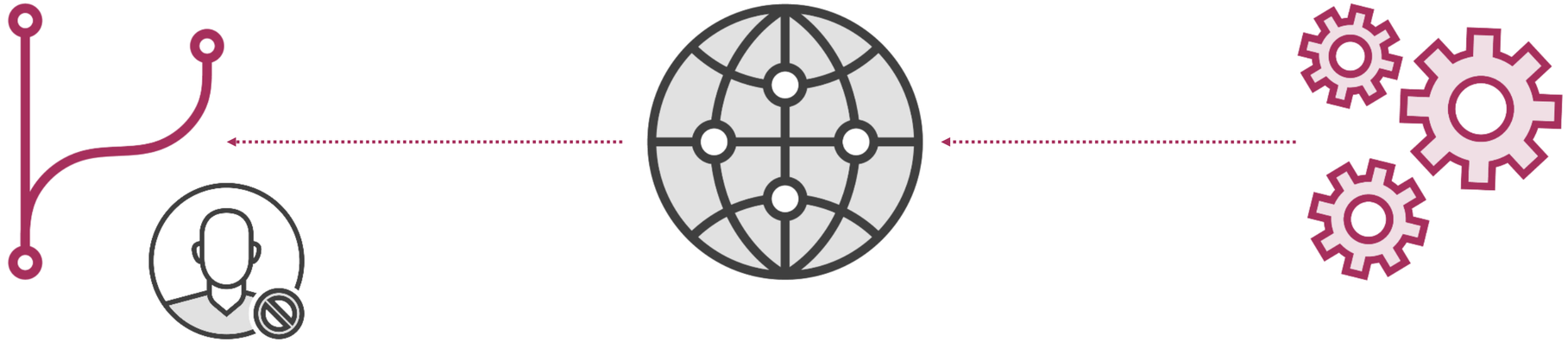
Flux configured trust for its own source repo

Achieved with permissive personal access token

Authentication

Flux needs to authenticate with remote source host providers, in order to establish trusted communication.

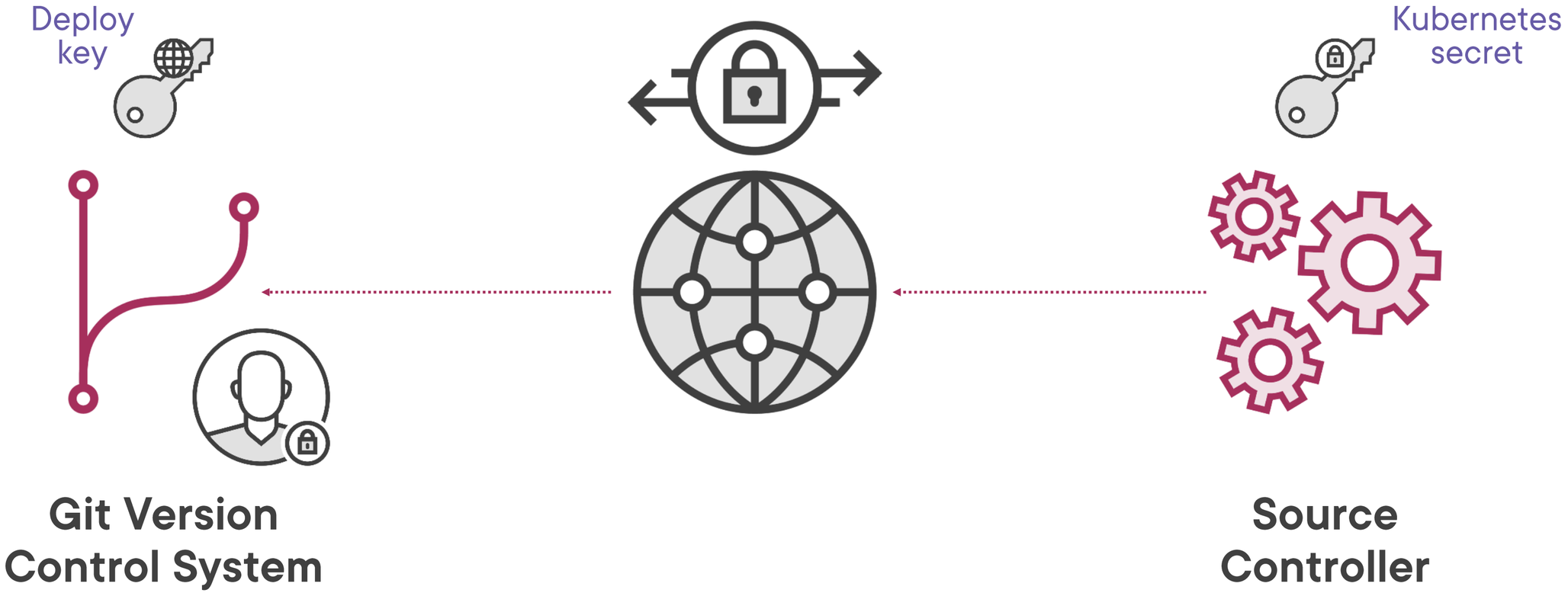
Trusted Communication



Git Version Control System

Source Controller

Trusted Communication



Flux CLI can be used to generate an SSH secret

```
$ flux create secret git gitops-lab-deploy-auth \
    --url=ssh://github.com/nbrownuk/gitops-lab-deploy \
    --namespace=default
deploy key: ecdsa-sha2-nistp384 AAAAE2VjZHNhLXNoYTItbmlzdHAz ...

git secret ‘gitops-lab-deploy-auth’ created in the ‘default’ namespace
```

Creating an Authentication Secret

Contains public/private key elements, along with ‘known_hosts’ content for remote provider

Basic authentication can be used in place of SSH where appropriate

Adding a Repository Deploy Key

Deploy keys / Add new

Title

flux-deploy-key

Key

```
ecdsa-sha2-nistp384
AAAAE2VjZHNhLXNoYTItbmlzdHAzODQAAAAIbmlzdHAzODQAAABhBBeON4djd5mGu9JF8Ekvfl3JkhVd9NAPAFhPCUznz
U1JP5KRt8QFCpxUznHrVJmXT7HMi8ALTwVIa0tIkM0pQE1pTnhlrI6LtJeTUX7krceyQ3i5Sr3Br7/fs2Jg4IPZIw==
```

☑ **Allow write access**

Can this key be used to **push** to this repository? Deploy keys always have pull access.

Add key

Factors That Affect Repo Structure

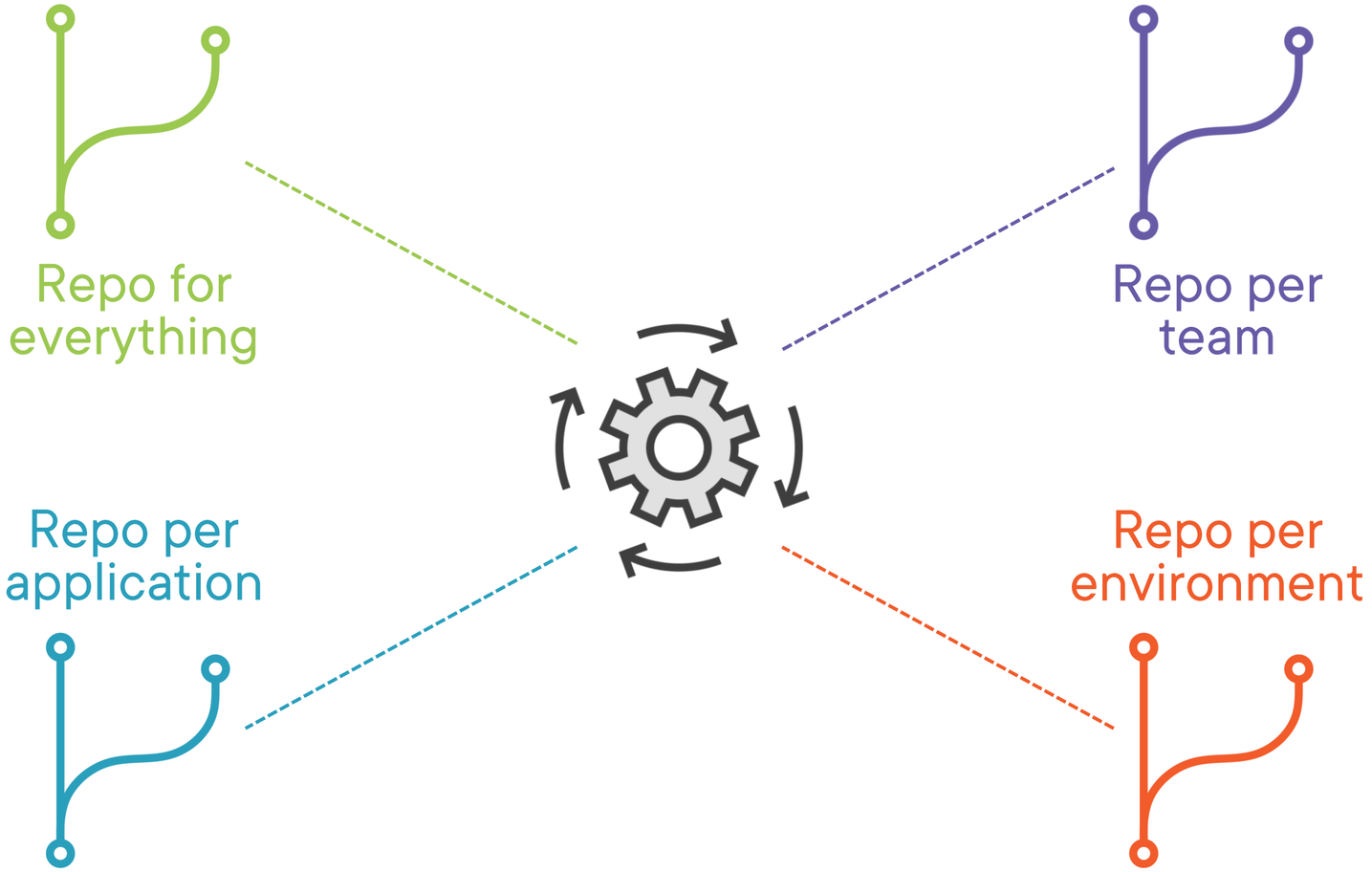
GitOps environments comprise of many repos

Applications, teams, environments, namespaces

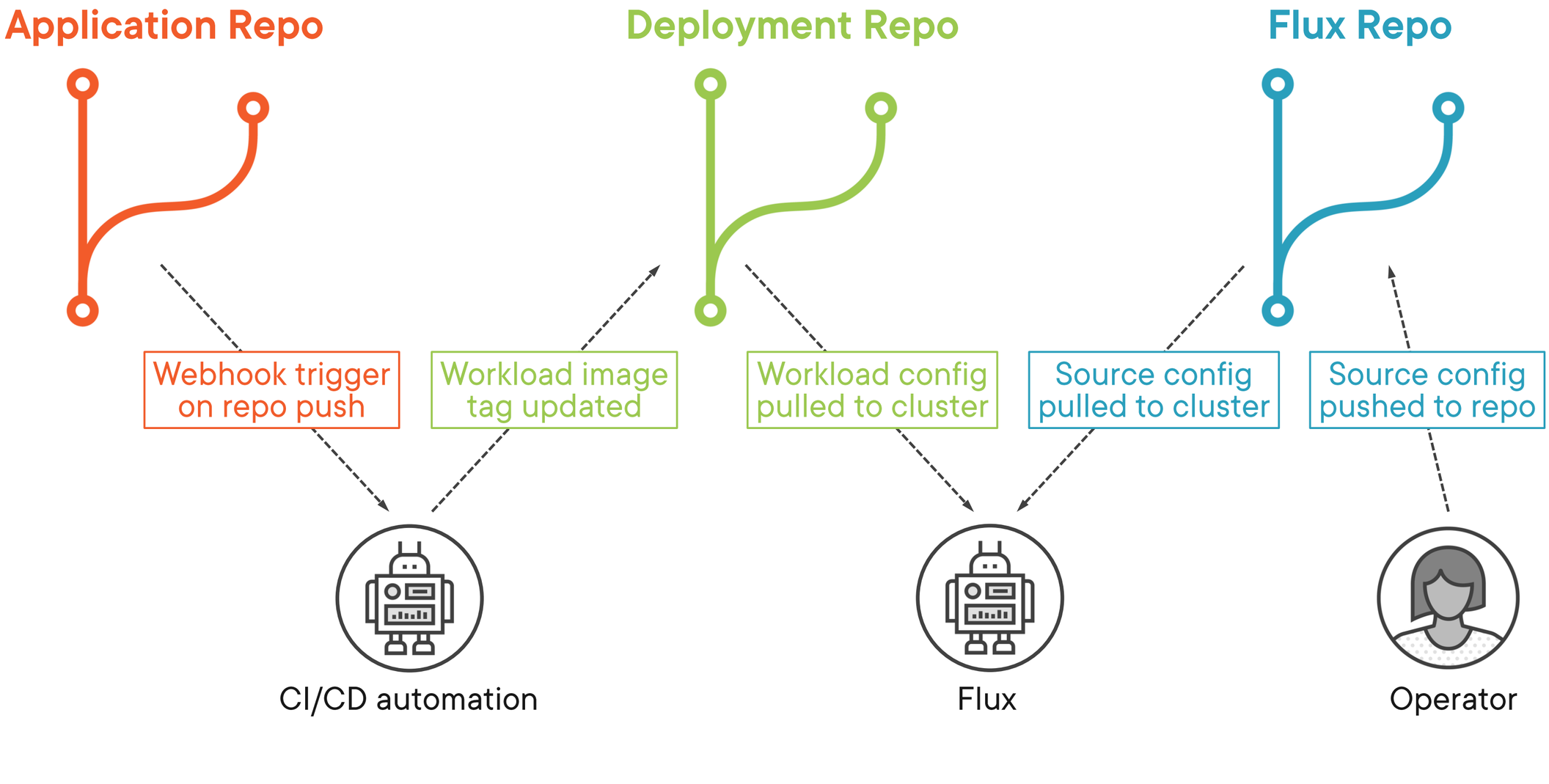
Need to strike the right balance for each factor

The repo structure should accommodate the way you work within your organization.

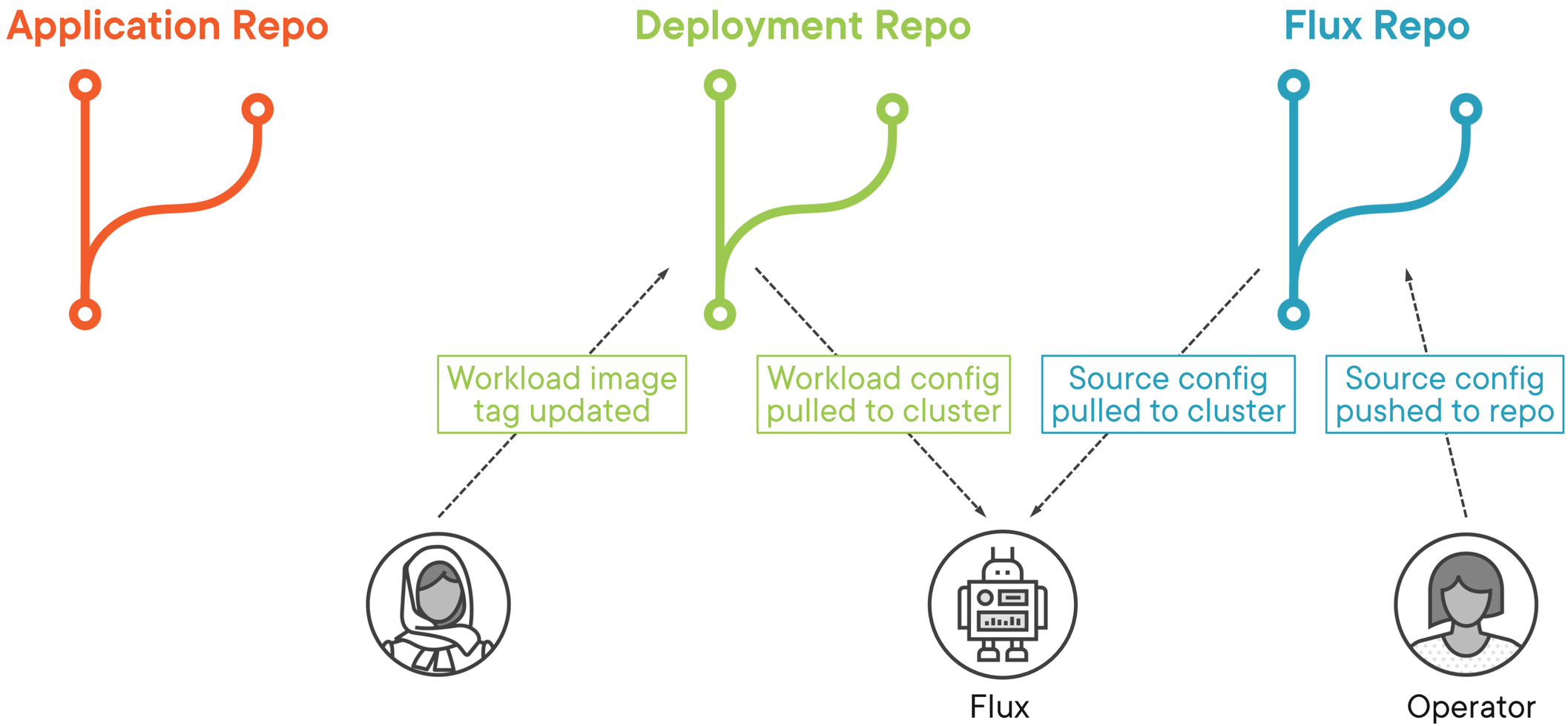
Source Repo Structure



Repo Structure for Demos



Simplified Repo Structure



Demo

Configuring Authentication with GitHub

- Create an SSH secret with the Flux CLI
- Use the public key as the deploy key
- Clone the remote deployment repo

Flux Custom Resources

Getting applications automatically deployed by Flux, requires the definition of some specific custom resources.

GitRepository API

```
---
apiVersion: source.toolkit.fluxcd.io/v1beta2
kind: GitRepository
metadata:
  name: nginxhello
  namespace: default
spec:
  url: ssh://github.com/nbrownuk/gitops-lab-deploy
  ref:
    branch: main
  secretRef:
    name:  gitops-lab-deploy-auth
  interval: 1m0s
```

Kustomization API

```yaml
---
apiVersion: kustomize.toolkit.fluxcd.io/v1beta2
kind: Kustomization
metadata:
  name: nginxhello
  namespace: default
spec:
  sourceRef:
    kind: GitRepository
    name: nginxhello
    namespace: default
  path: ./deploy
  interval: 1m0s
  prune: true
  targetNamespace: default
```

Kustomize configuration management utility: https://kustomize.io

Kustomization API

```yaml
---
apiVersion: kustomize.toolkit.fluxcd.io/v1beta2
kind: Kustomization
metadata:
  name: nginxhello
  namespace: default
spec:
  sourceRef:
    kind: GitRepository
    name: nginxhello
    namespace: default
  path: ./deploy
  interval: 1m0s
  prune: true
  targetNamespace: default
```

Set namespace in source config, or use ‘targetNamespace’

Demo

Automating an Application Deployment

- Create appropriate custom resources
- Push revised desired state to Flux repo
- Check Flux reconciles custom resources
- Observe deployment of application
- Monitor Discord for notification events

Handling Application Updates with Image Automation

Module Summary

What we covered:

- Configured Flux's notification system for Discord
- Provided Flux with an identity for GitHub using SSH
- Explored Flux's use of custom resource definitions