Using the GitOps Approach for Automating Deployments

Nigel Brown

@n_brownuk www.windsock.io

Module Outline

How does GitOps help to automate application deployments?

Coming up:

- The principles that govern GitOps
- GitOps in the wild
- Introduction to the Flux toolset
- Bootstrapping Flux into a cluster

GitOps

GitOps is a discipline governed by a set of principles, that:

1. **Uses version-controlled, declarative configuration to describe a desired state, and**
2. **Establishes the desired state in a target system through the use of software automation.**

The GitOps Principles

OpenGitOps CNCF Sandbox Project: https://opengitops.dev/

Principles defined by OpenGitOps project:

- Declarative

  A system managed by GitOps must have its desired state expressed declaratively.

- Versioned and immutable

  Desired state is stored in a way that enforces immutability, versioning and retains a complete version history.

- Pulled automatically

  Software agents automatically pull the desired state declarations from the source.

- Continuously reconciled

  Software agents continuously observe actual system state and attempt to apply the desired state.

Declarative

Describable
Abstractions and syntax

Understandable
Common view of desired state

Ingestible
Suitable for automation

Recoverable
Easier path to recovery

Versioned and Immutable

Version-controlled storage of the desired state, provides a single source of truth for all stakeholders

Immutable versions of the desired state, safeguard against the risk of accidental configuration drift

Rollback mechanisms are an inherent feature of systems that manage the storage of versioned code

A transaction log provides a history of change to desired state, including the rationale and the actors involved

Automatically Pulled

Software automation automatically pulls the desired state from the source, but this can be instigated in different ways.

Source polling

Software agent polls the source on a periodic basis.

Webhook trigger

A pull is triggered on the receipt of an external webhook.

Image reflection

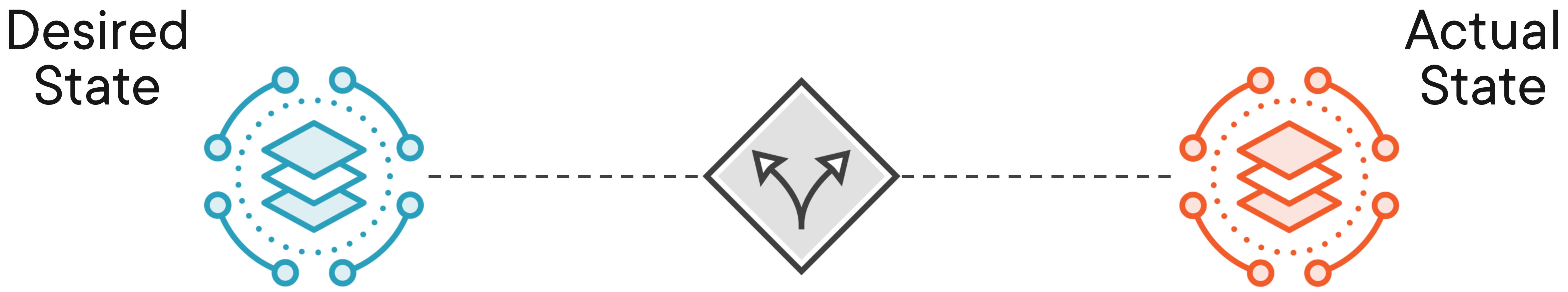
Detection of new image tag is reflected in the desired state.

Continuously Reconciled

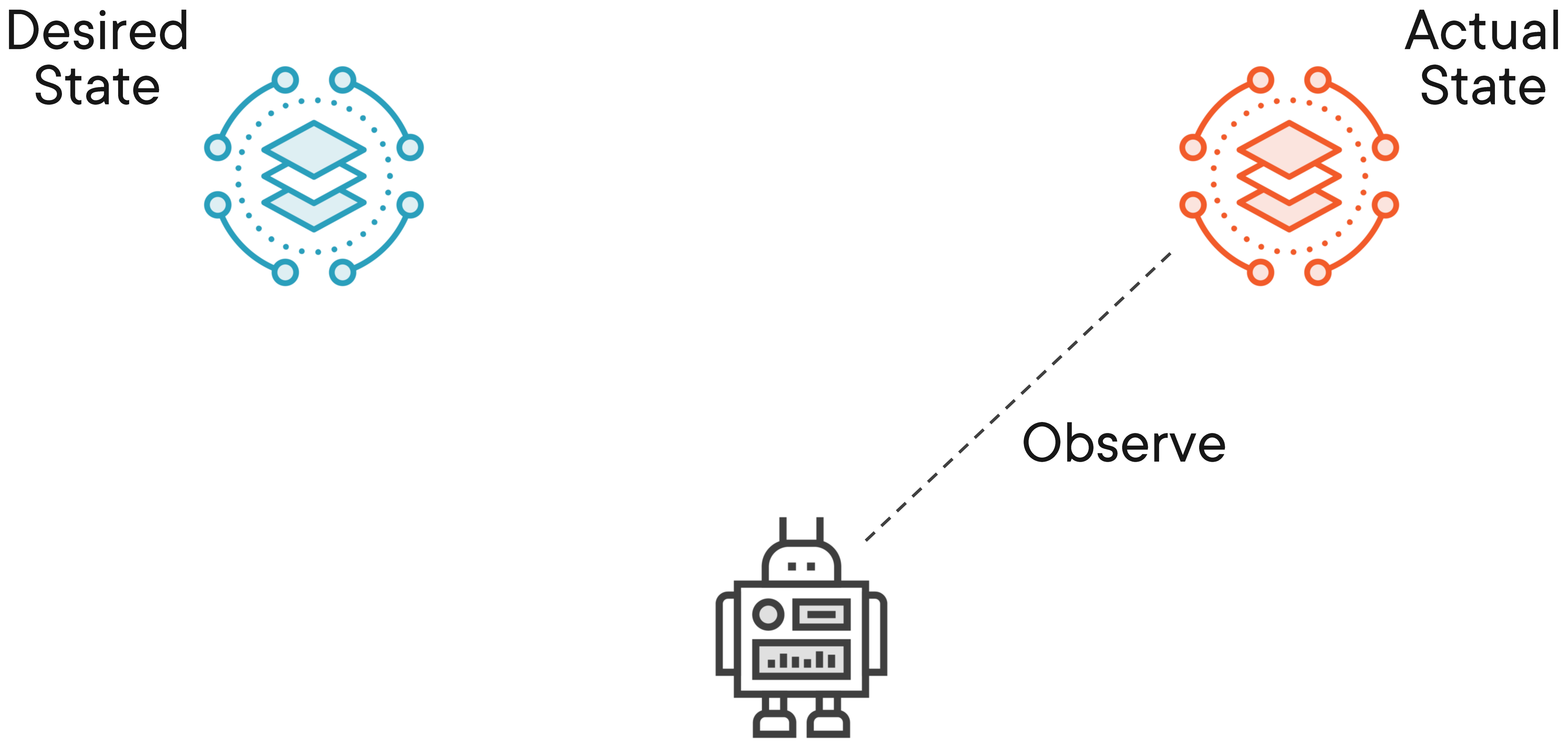
Desired State

Actual State

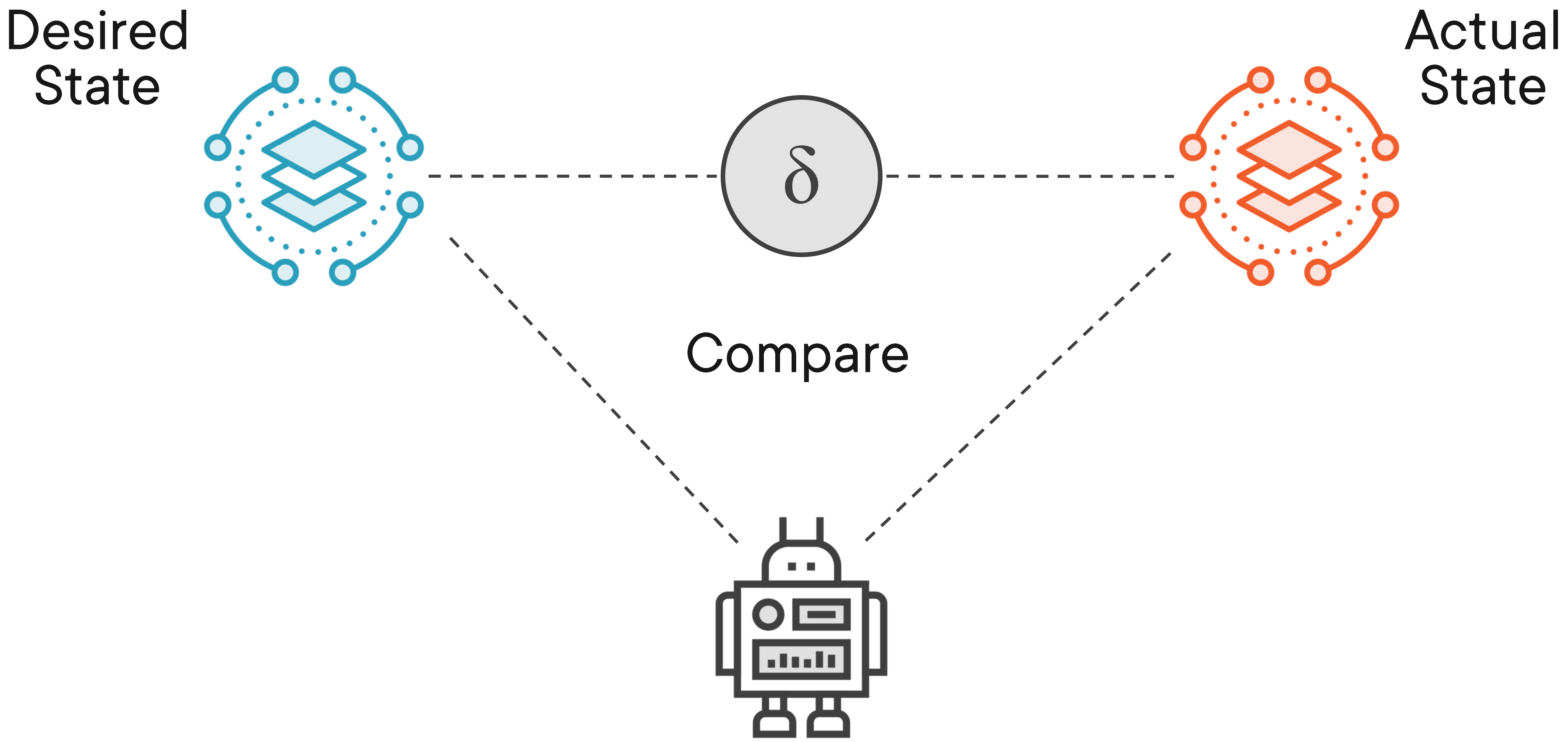
Continuously Reconciled



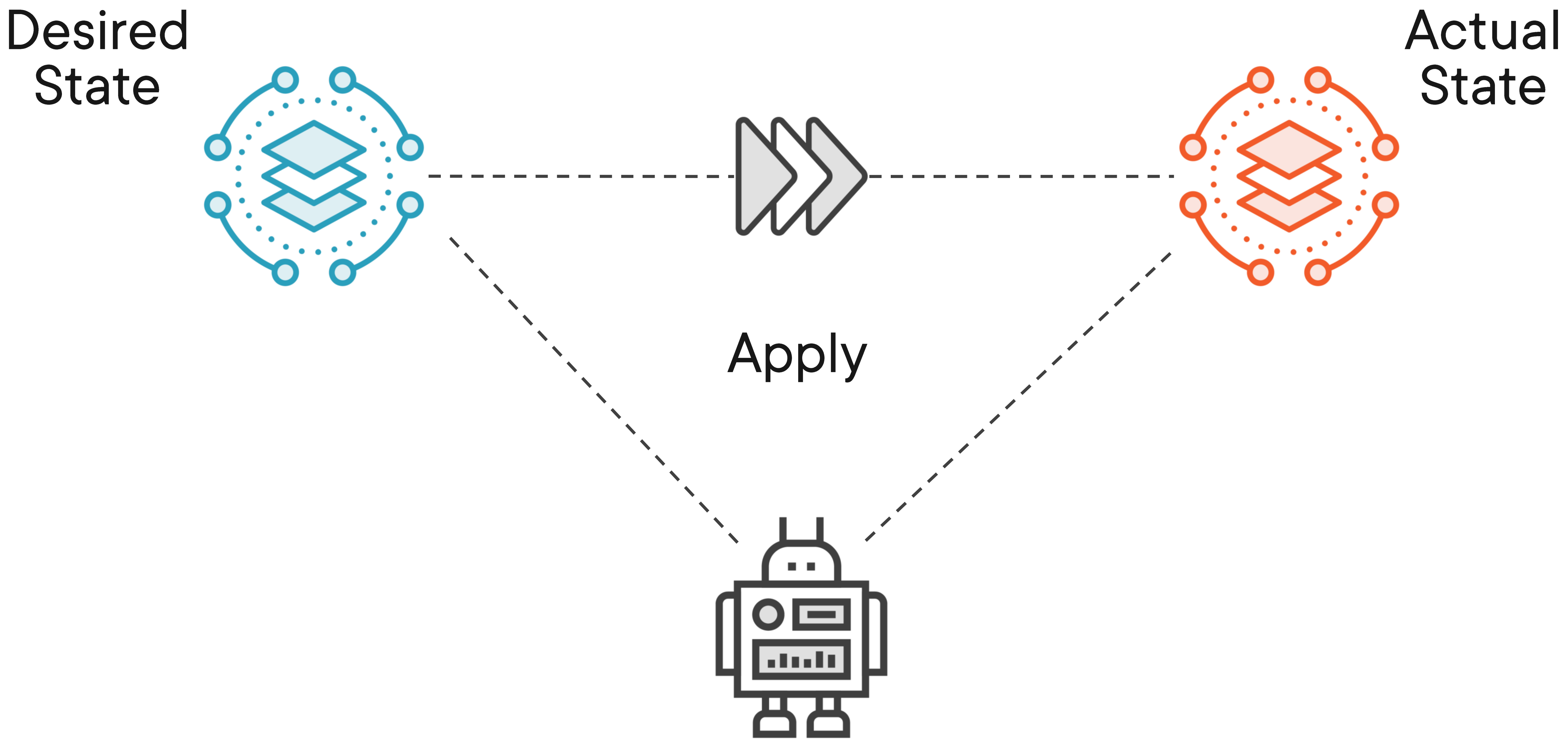
Continuously Reconciled



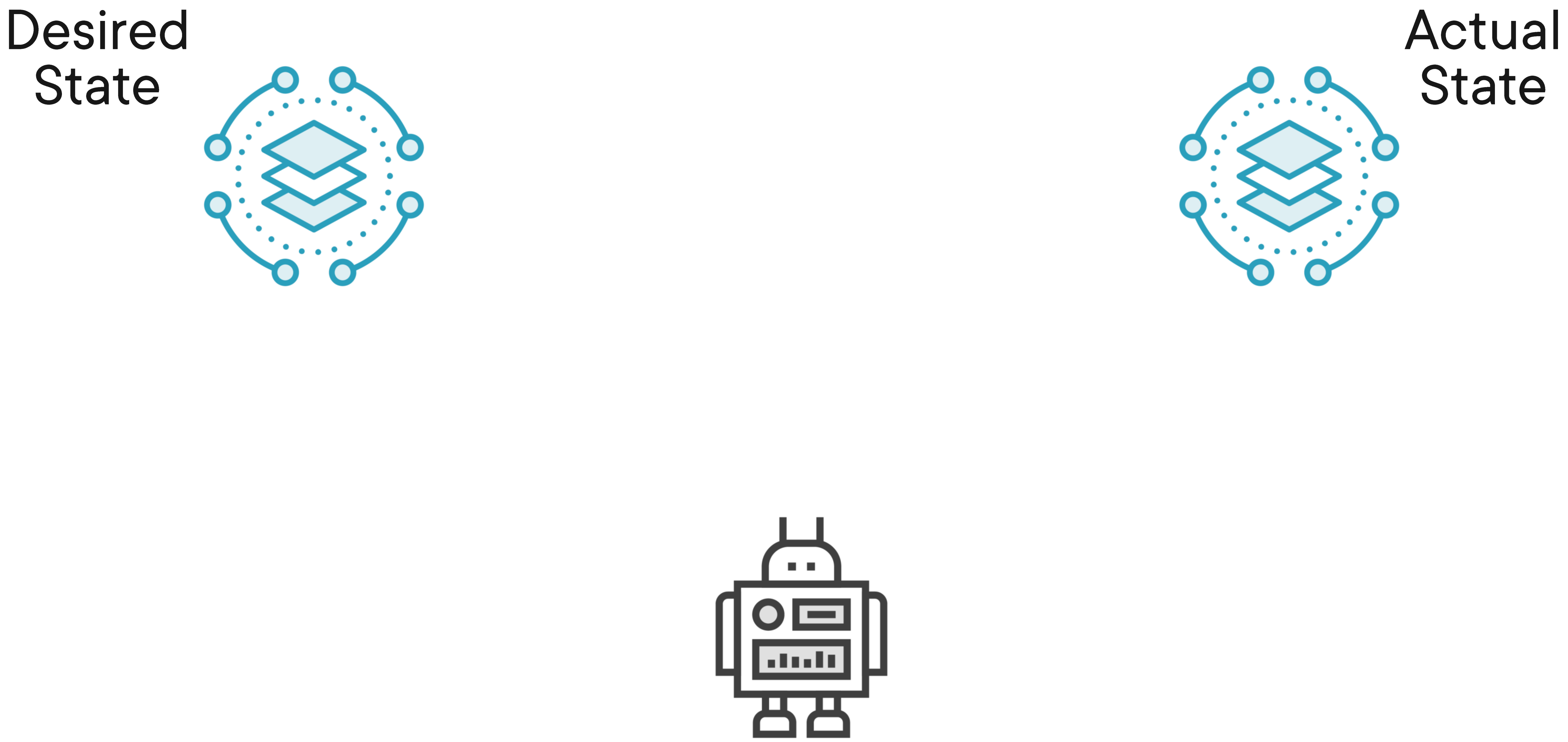
Continuously Reconciled



Continuously Reconciled



Continuously Reconciled



GitOps

GitOps is not opinionated about the choice of technology in a solution, but it does mandate use of the four GitOps principles.

GitOps in the Wild

Problem domain is not limited to Kubernetes

GitOps can be applied at all layers of the stack

Desired state storage does not need to be Git

Does not replace tasks within CI/CD pipelines

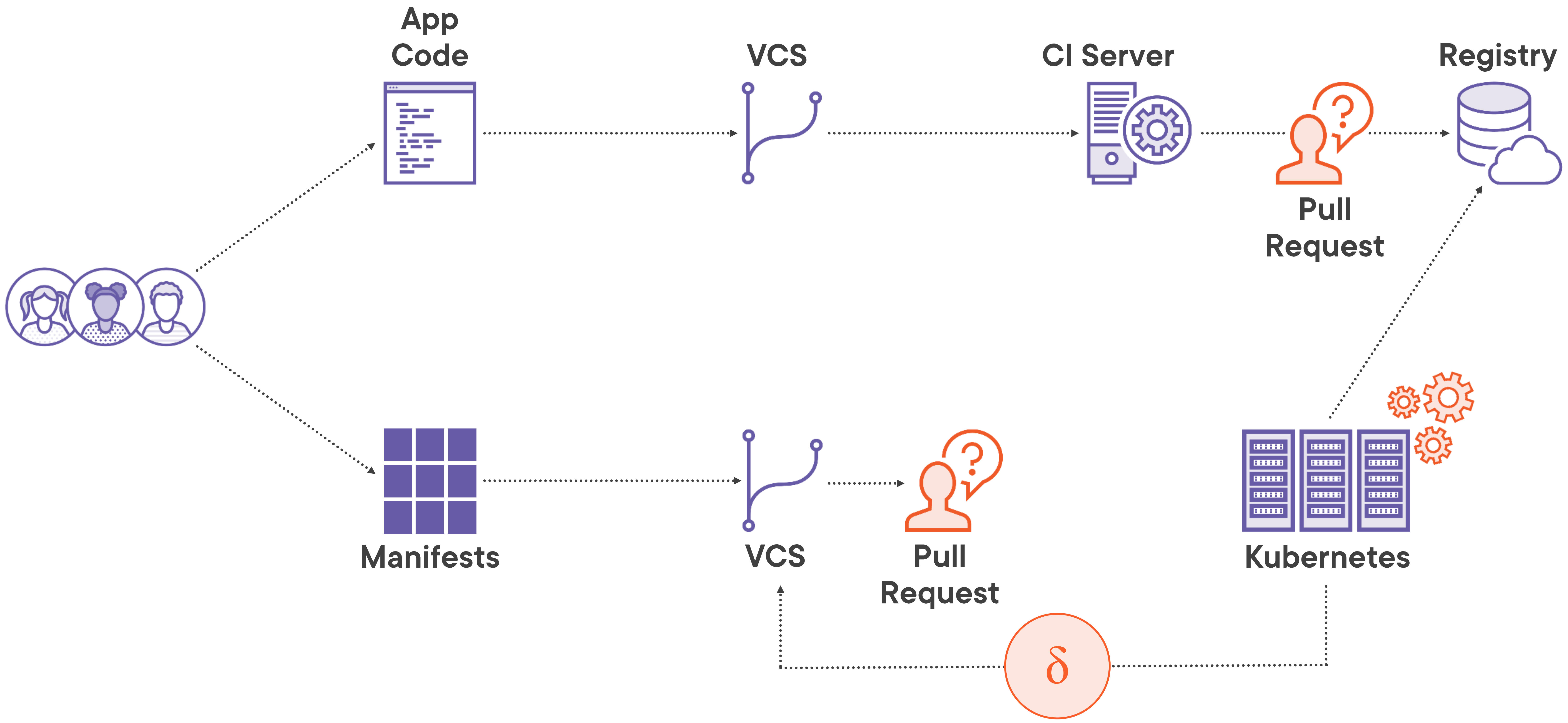
Deployments with CI/CD Pipelines

CI/CD pipelines can be used to trigger new deployments to a Kubernetes cluster. Why do we need GitOps?

Deployment Strategies

CI/CD	GitOps
Desired state ‘pushed’ by CI/CD tools	Desired state ‘pulled’ by GitOps agent
Requires Kubernetes API access from outside the cluster boundary	Accesses the Kubernetes API server from within cluster boundary
No visibility of state inside the cluster	Configured to view relevant API objects
Variety of traditional CI/CD tools can fulfil push automation	Limited and bound by smaller set of available tools

GitOps Workflow



GitOps Tooling

Jenkins X

https://jenkins-x.io/

ArgoCD

https://git.io/JJ7yc

Flux

https://fluxcd.io/

Introducing Flux

A set of Kubernetes controllers that implement the GitOps principles as defined by the OpenGitOps project

Handles configuration defined as raw YAML, Kustomize overlays, or packaged as Helm charts

Optional support for monitoring repositories in container registries for new application images

Can be used with other tools to support automated progressive deployments (e.g. canary releases)

Flux v2 is pre-dated by an earlier, different GitOps solution, called Flux v1.

GitOps Toolkit

The different controllers that comprise the Flux solution, are also collectively known as the GitOps Toolkit.

Toolkit Components

Source Controller

Kustomize Controller

Helm Controller

Notification Controller

Image Reflector Controller

Image Automation Controller

Flux Command Line

Fully-featured command line utility for working with the GitOps Toolkit, and for managing GitOps workflows.

Provision

Enables Flux to be bootstrapped into an existing cluster.

Manage

Allows for ongoing management of a Flux deployment.

Query

Provides a means for retrieving status information.

Demo

Installing Flux to a Kubernetes Cluster

- Using the Flux CLI
- Bootstrapping the GitOps Toolkit
- Establishing the ‘Single Source of Truth’

k3d is a lightweight wrapper to run k3s
https://k3d.io/

Chicken or the Egg?

Flux is a collection of software components

Can we manage Flux using GitOps principles?

Is it possible for Flux to manage itself?

Configuring Flux for Automated Deployments

Module Summary

What we covered:

- The GitOps principles
- GitOps is tool agnostic
- GitOps is an extension to CI/CD pipelines
- Flux is one of the main GitOps solutions
- Bootstrapping the GitOps Toolkit

Let's dig a bit deeper!