

Implementing GitOps for Kubernetes Applications

Establishing a GitOps-driven Deployment Model



Praveenkumar Bouna

Cloud Instructor

linkedin.com/in/praveenkumarbouna



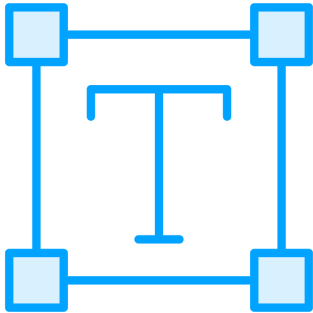


Introduction to GitOps Principles

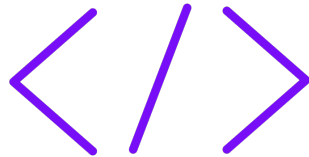
GitOps

A set of practices that uses Git repositories as the single source of truth for infrastructure and application deployments.

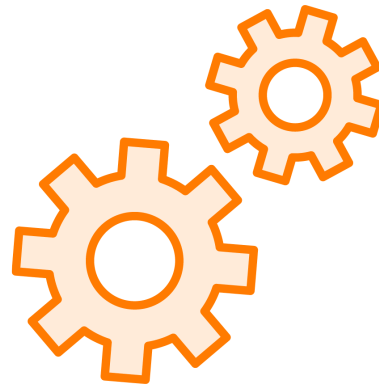
Core Principles of GitOps



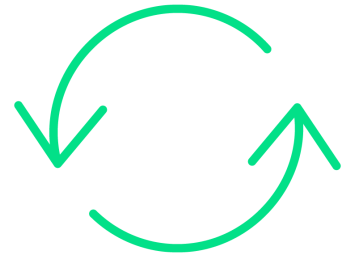
**Declarative
infrastructure**



Version controlled



**Automatically
applied**



**Automatic drift
detection**

Benefits for Cloud-native Applications

Improved security through declarative deployments

Reduced configuration drift and human error

Faster recovery through rollbacks

Enhanced collaboration through Git workflows

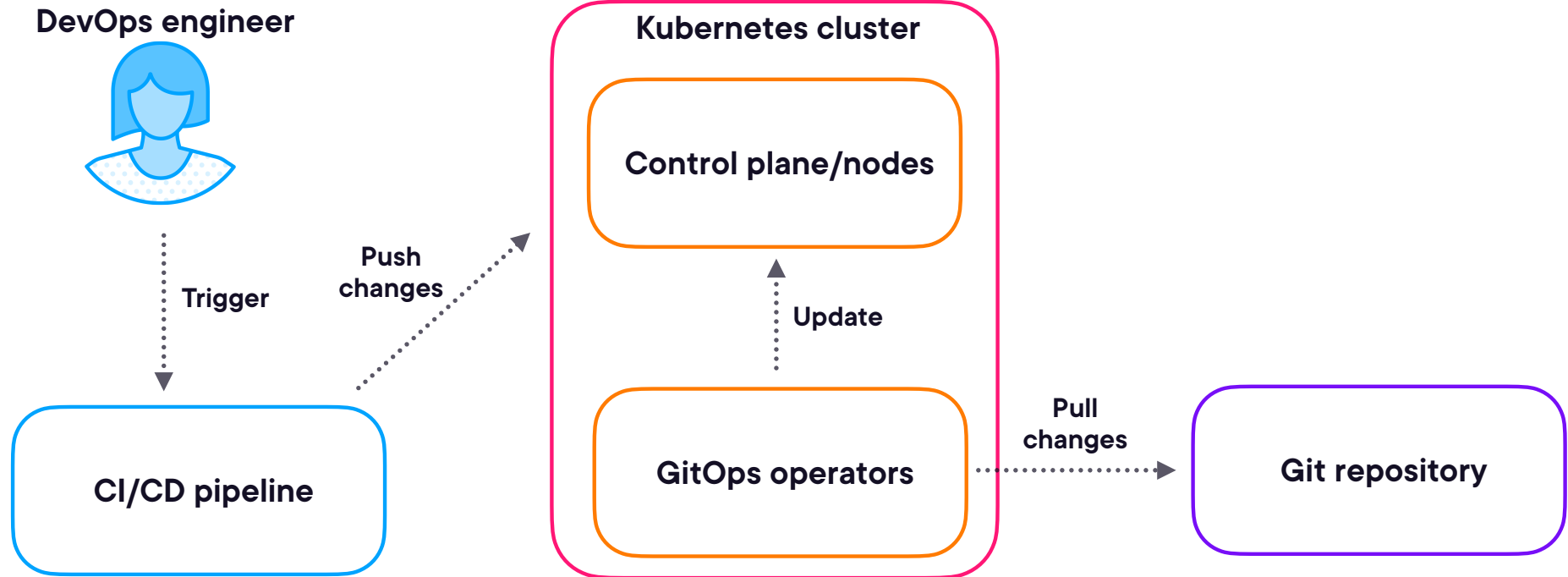


GitOps Pull-based Deployment

GitOps Pull-based Deployment

Traditional

GitOps





GitOps vs. Traditional DevOps Workflows

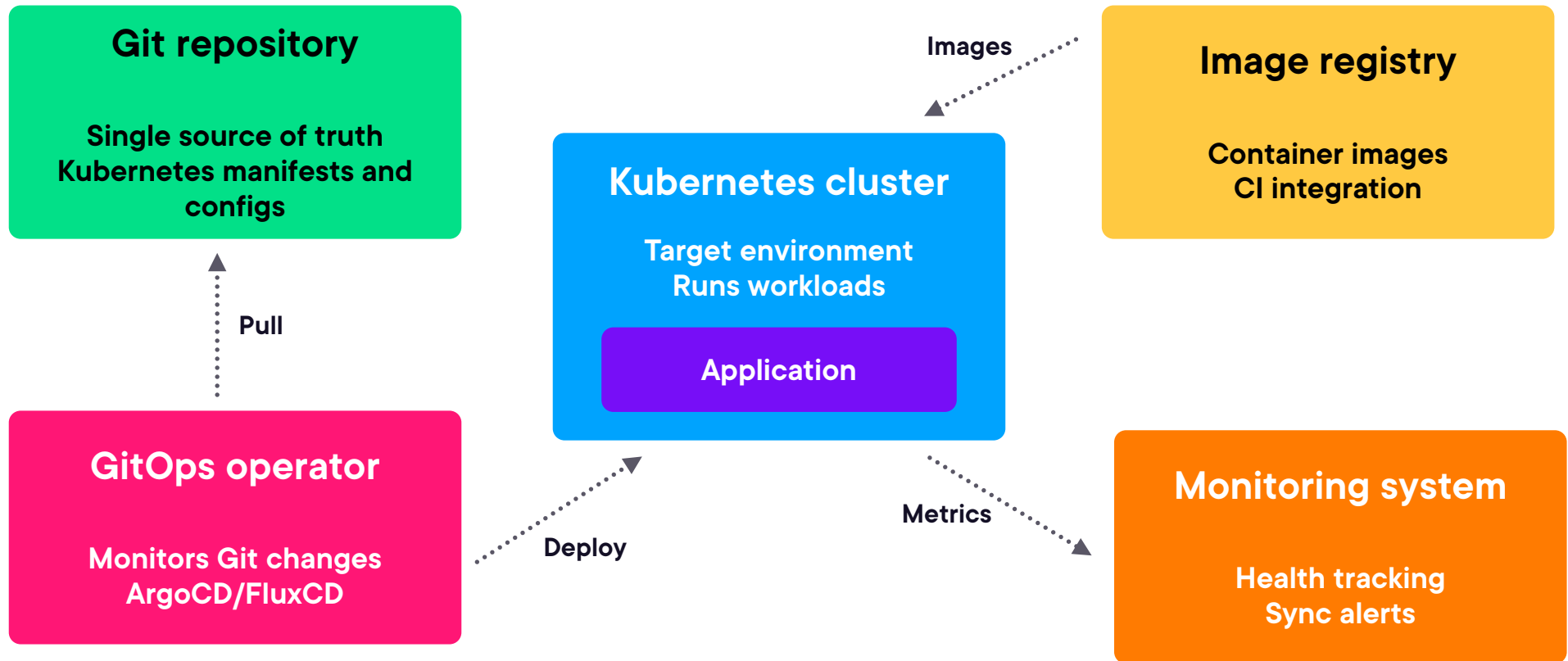
GitOps vs. Traditional DevOps

Aspect	Traditional DevOps	GitOps
Deployment	Push-based <ul style="list-style-type: none">• Requires cluster credentials• Direct cluster modification	Pull-based <ul style="list-style-type: none">• Agents pull from Git• No external credentials
Audit Trail	Limited <ul style="list-style-type: none">• Changes not tracked• Manual documentation	Complete <ul style="list-style-type: none">• Git history tracks all changes• PR reviews for approval
Drift Detection	Manual <ul style="list-style-type: none">• No automatic detection• Environment inconsistency	Automatic <ul style="list-style-type: none">• Continuous monitoring• Auto-correction of drift
Rollback	Manual <ul style="list-style-type: none">• Manual procedures• Error-prone process	Automated <ul style="list-style-type: none">• Git revert• Fast and reliable



GitOps Architecture Components

GitOps Architecture Components





Demo: Implementing Declarative Kubernetes Manifests