# An Introduction to GitOps and Secrets Management

## Managing your cloud native ecosystem securely

Andrew Block
Distinguished Architect



Red Hat

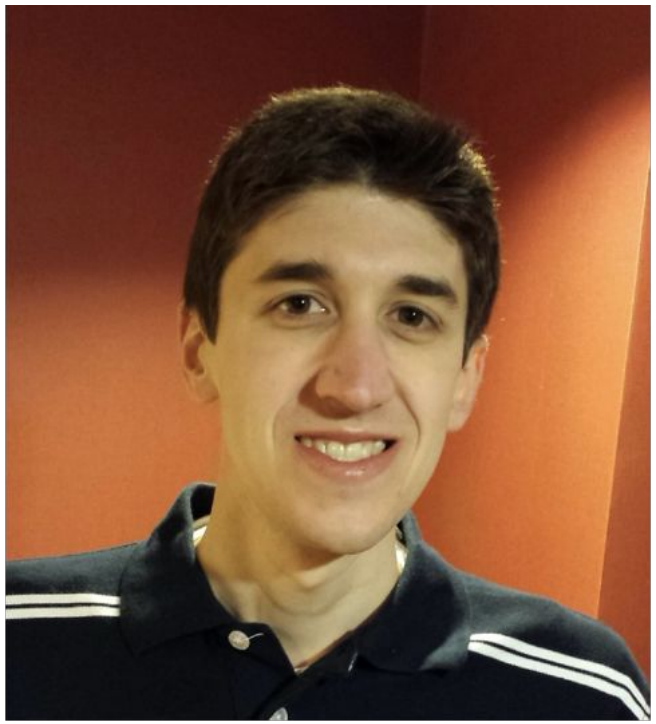
# $ git log --author="Andrew Block" --patch

Author: Andrew Block <andrew.block@redhat.com>
Date: Tue July 14 14:00:00 2022 -0500

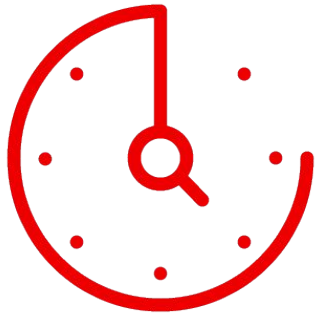
Added speaker bio

diff --git a/speaker_bio.txt b/speaker_bio.txt
new file mode 100644
index 0000000..23c2ed6
--- /dev/null
+++ b/speaker_bio.txt
@@ -0,0 +1,7 @@

+ **Distinguished Architect at Red Hat**
+
+ **Specialize in Security, CI/CD and Automation**
+
+ **Author:**
+. **- Learn Helm**
+ **- Kubernetes Secrets Management**
+
+ **Helm Maintainer**
+
+ **Lead Helm integration on sigstore project**



# What we'll discuss today



- Introduction to GitOps
- OpenShift GitOps
- Secret Management with GitOps

# Adopting a Cloud Native Approach

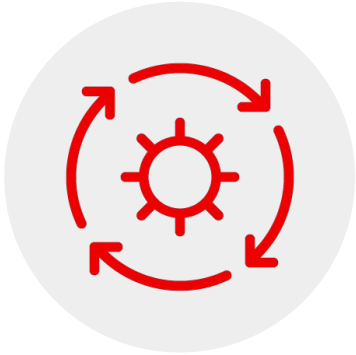
## Top Considerations

### Speed

Increase developer productivity and ship quality applications faster

### Security

Application and supply chain security from start to production



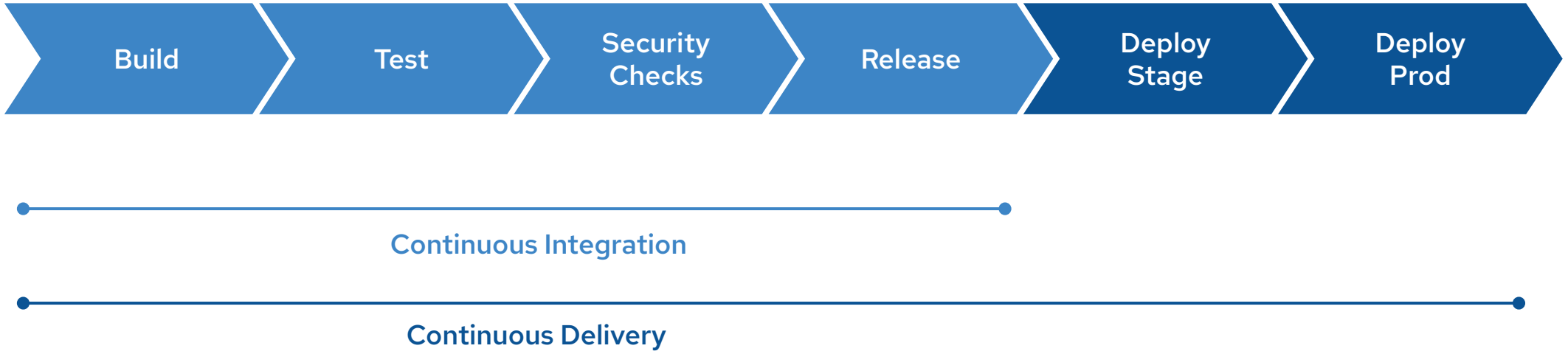
### Scale

Automate and scale application delivery on hybrid cloud infrastructure

Innovate your new future and optimize what you have

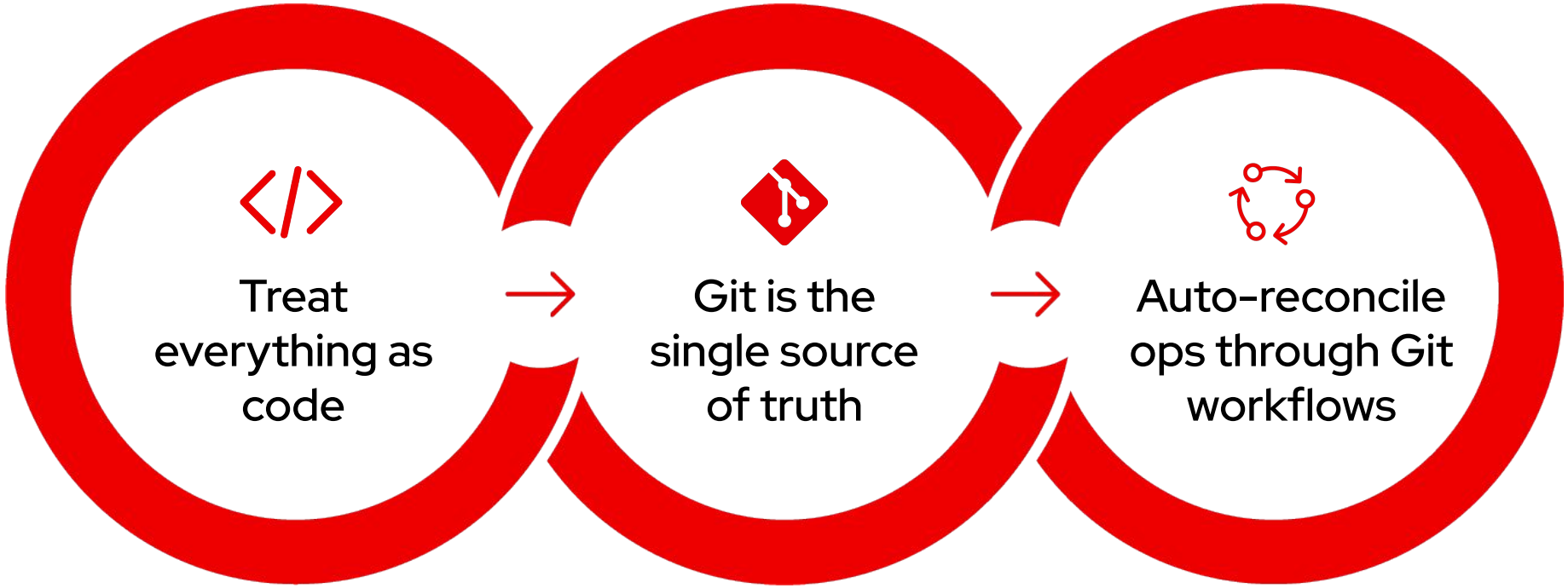
# Continuous Integration(CI) & Continuous Delivery (CD)

A key DevOps principle for automation, consistency and reliability



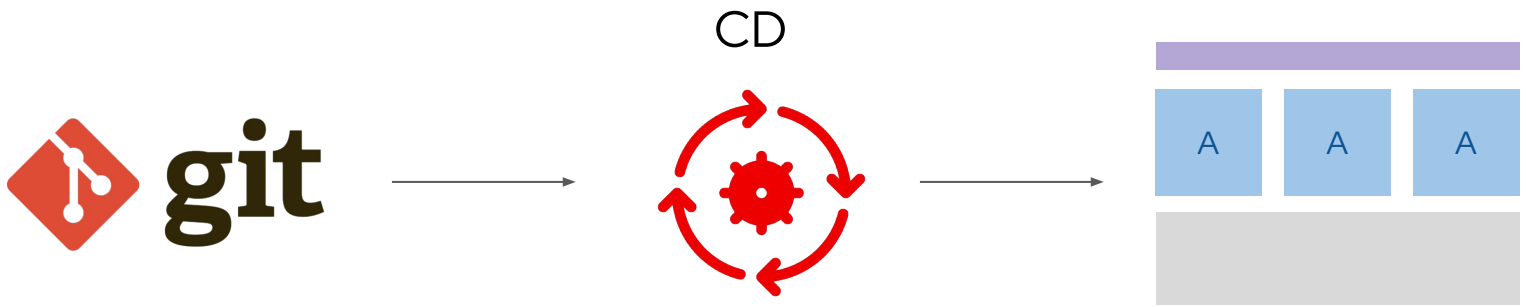
# What is GitOps?

An developer-centric approach to Continuous Delivery and infrastructure operation



# GitOps Workflow

## a declarative approach to application delivery




What you want
(desired state)

What you have
(current state)

# Why GitOps?

## Standard Workflow

Familiar tools and Git workflows from application development teams

## Enhanced Security

Review changes beforehand, detect configuration drifts, and take action

## Visibility and Audit

Capturing and tracing any change to clusters through Git history

## Multi-cluster consistency

Reliably and consistently configure multiple Kubernetes clusters and deployment

# The GitOps Evolution

## Adopting GitOps Principles is a Departure from How Infrastructure and Applications are Traditionally Managed

### Traditional Models

- Emphasizing access management against individuals instead of systems
- Actions performed manually
- Dedicated teams responsible for managing infrastructure and applications
- Change request and approvals

### GitOps

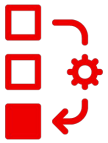
- Actions automated
- Infrastructure codified
- Participation by both infrastructure and application teams
- Changes applied automatically
- Self healing

# OpenShift GitOps

# OpenShift GitOps

## Multi-cluster config management

Declaratively manage cluster and application configurations across multi-cluster OpenShift and Kubernetes infrastructure with Argo CD



## Automated Argo CD install and upgrade

Automated install, configurations and upgrade of Argo CD through OperatorHub

## Opinionated GitOps bootstrapping

Bootstrap end-to-end GitOps workflows for application delivery using Argo CD and Tekton with GitOps Application Manager CLI

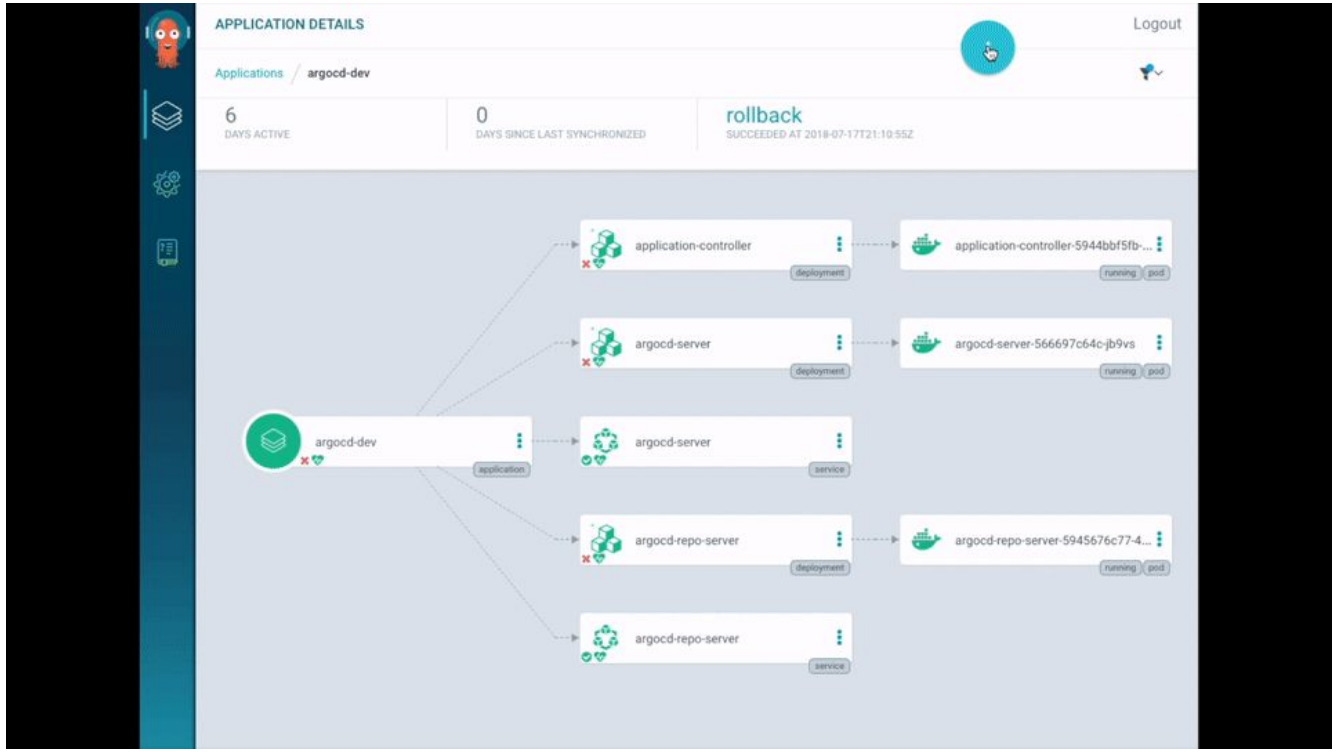
## Deployments and environments insights

Visibility into application deployments across environments and the history of deployments in the OpenShift Console

# Argo CD - What is It?

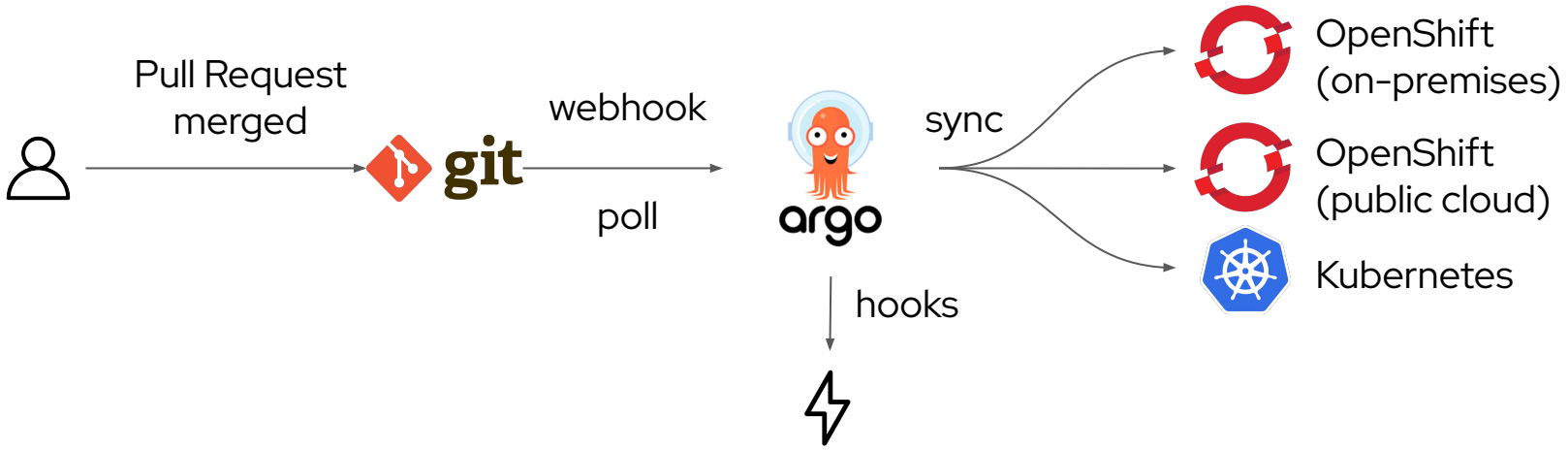
*Argo CD is a declarative, GitOps continuous delivery tool for Kubernetes.*

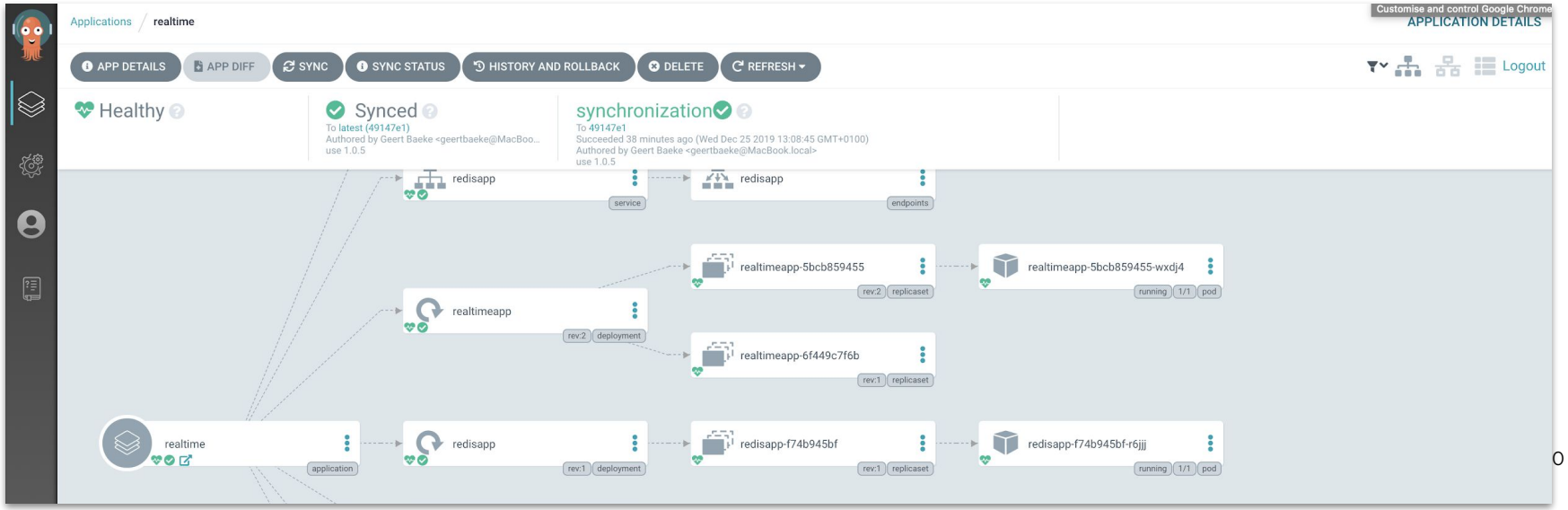
- Easily deploy applications in a declarative way
- Synchronizes cluster state with git repos
- Works with a variety of Kubernetes deployment tools including:
  - Helm
  - Kustomize
  - Ksonnet/Jsonnet
  - Directories of yaml
- It is **not** a CI tool!



# Git Workflow with Argo CD



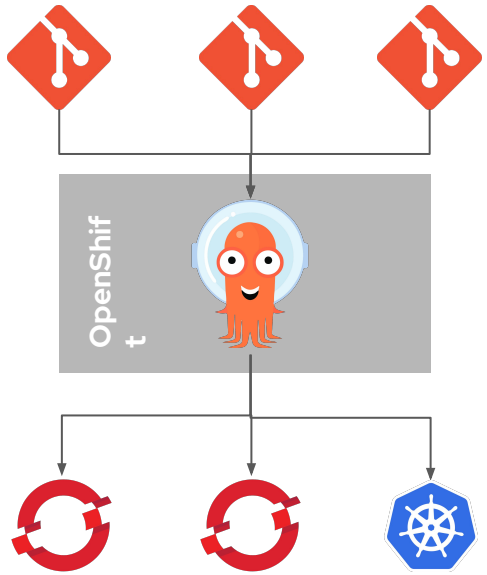




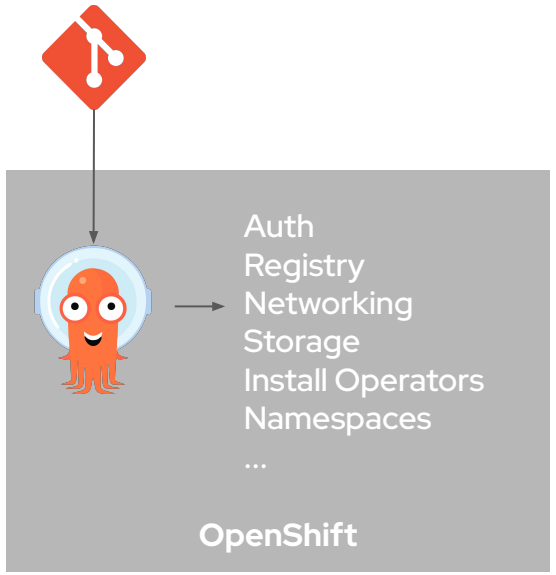

Red Hat

# Flexible Deployment Strategies



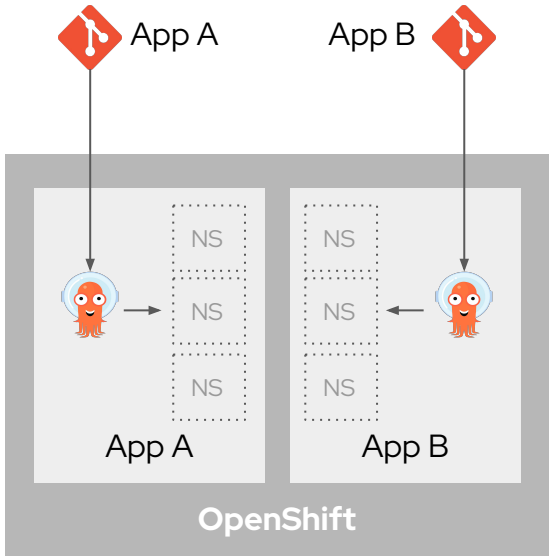

## Central Hub (Push)

A central Argo CD pushes Git repository content to remote OpenShift and Kubernetes clusters




## Cluster Scoped (Pull)

A cluster-scope Argo CD pulls cluster service configurations into the OpenShift cluster




## Application Scoped (Pull)

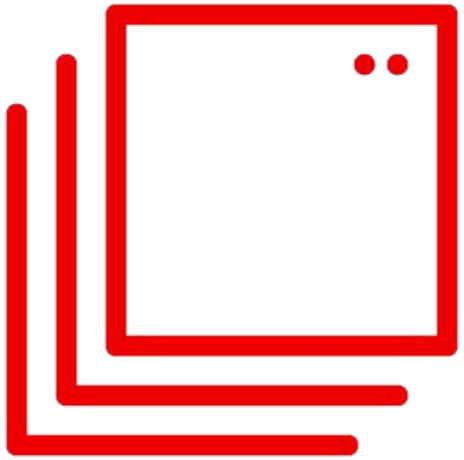
An application scoped Argo CD pulls application deployment and configurations into app namespaces

# What is an Argo CD Application?

- *Application* is a Custom Resource (CR) is a custom resource provided by Argo CD
- Application definition includes:
  - Name
  - Cluster
  - Git repository
  - Synchronization Policy
- Application is the unit of work
- Applications can be deployed from Argo CD GUI or CLI (argocd or kubectl or oc)

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
 name: product-catalog-dev
 namespace: argocd
spec:
 destination:
  namespace: argocd
  server: https://kubernetes.default.svc
 project: product-catalog
 source:
  path: manifests/app/overlays/dev-quay
  repoURL: https://github.com/gnunn-gitops/product-catalog.git
  targetRevision: main
 syncPolicy:
  automated:
   prune: false
   selfHeal: false
```

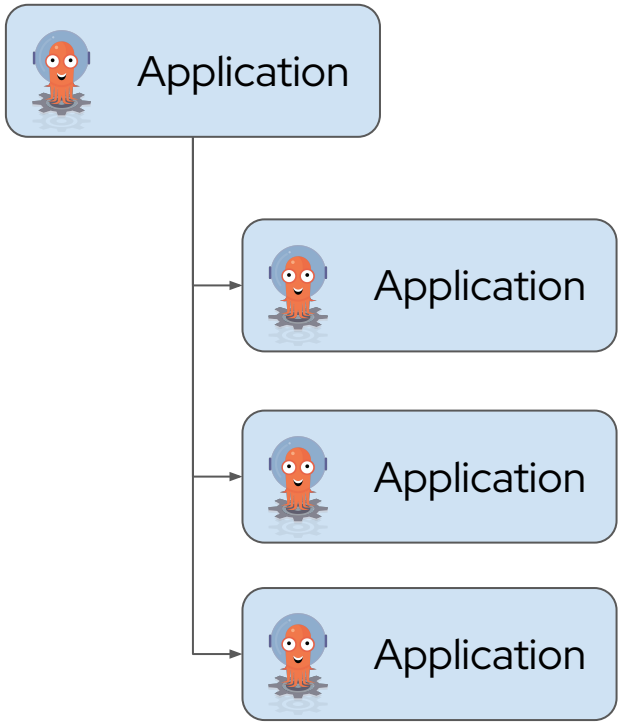
# Application Scope of Deployment



- Argo CD Application can span multiple Kubernetes Namespaces
- In Argo CD, the Application is the “center of the world”
- Argo CD has multi tenancy capabilities with Projects
  - These differ from OpenShift Projects
  - Is not 1:1 with Namespaces/Projects
- Argo CD Application deployed to an Argo CD Project can be explicitly be deployed across multiple Namespaces/Projects

# Argo CD “App of Apps”

- App of Apps is a common pattern where one Argo CD Application points to a repo that only contains other Argo CD Applications
- Very useful to be able to provision and manage a group of related applications or items together in a single shot
- Pattern has evolved over time, expected to be supplanted by new ApplicationSets feature




https://argoproj.github.io/argo-cd/user-guide/application-set

# ApplicationSets

- Use a single manifest to target multiple clusters
- Deploy Application from a single or multiple git repos
- Manages lifecycle of the Application CR
- Can be thought of as sort of a “factory” for building Application CRs
  - You still end up with Application definitions, it’s just managed now from a central configuration.

# Managing Secrets in GitOps

# GitOps and Secrets

## GitOps Commonly Involves the Use of Sensitive Assets

### Source Code Repositories

Credentials to obtain access to the content driving GitOps

### Application Configurations

Properties that are utilized by applications

### Infrastructure Management

Configurations for managing the infrastructure

### Access to Platform Resources

Assets to communicate to platform components

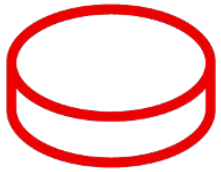
# Managing Secrets the Wrong Way

- **DO NOT** store sensitive resources in plain text
  - Committed to Git repositories
  - Within associated tooling
- Not understanding the capabilities and limitations of the tools being used
- Implementing “security theater”
  - Security by obscurity


https://argoproj.github.io/argo-cd/user-guide/application-set

# Properly Handling Secrets in GitOps

## GitOps Commonly Involves the Use of Sensitive Assets

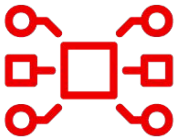


### Storing Secrets

Methods for secure storage of sensitive assets

### Accessing Secrets

Obtaining previously stored values



### Consuming Secrets

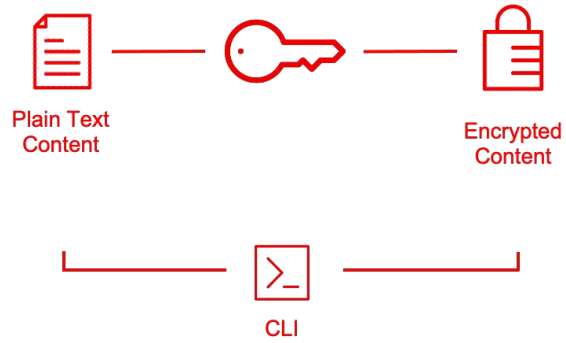
Mechanisms for which target components make use of sensitive assets

# Storing Secrets

Two primary approaches for storing sensitive assets within source code repositories

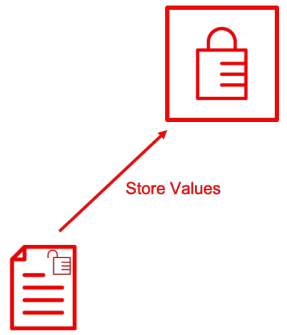
## Encrypted Content

Sensitive values are encrypted and stored within repository




## External Reference

Sensitive values are stored within another location. Reference is provided within repository content.



# Tools for Accessing Secrets

## Various utilities are available retrieve previously stored sensitive assets

### Command Line Interface

- Interactive methods when working on a local machine
- Standalone or integrated into existing tooling

### CI/CD

- Popular solutions (Jenkins, GitHub, GitLab) support such integrations
- Accelerates DevSecOps processes

### Plugins

- Extension of existing tooling
- Abstracts interactions supporting secrets management
- Lifecycle can integrated within tool or handled by plugin manager
  - helm, kubectl, etc

```
$ <CLI_TOOL> <PLUGIN> <COMMANDS>
```

# Kubernetes Native Secret Access

## Popular tooling to support Secrets Management in Kubernetes

### Sealed Secrets

- Controller containing private keys to encrypt/decrypt sensitive values

### External Secrets

- Retrieve values stored within external Secrets Management utilities.

### Vault Agent Sidecar Injector

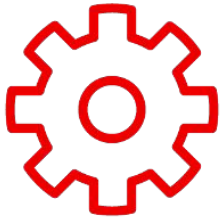
- Feature rich tool for HashiCorp Vault
- Injects sidecars to applications
- Manages lifecycle of secrets within cluster

```
apiVersion: external-secrets.io/v1alpha1
kind: ExternalSecret
metadata:
  name: aws-creds
  namespace: gitops
spec:
  refreshInterval: "60s"
  secretStoreRef:
    name: vault
    kind: ClusterSecretStore
  target:
    name: aws-creds
  data:
  - secretKey: aws_secret_access_key
    remoteRef:
      key: kv/data/mycluster
      property: aws_secret_access_key
  - secretKey: aws_access_key_id
    remoteRef:
      key: kv/data/mycluster
      property: aws_access_key_id
```

AWS credentials stored in Vault and integrated into Kubernetes using External Secrets project

# Consuming Secrets

## Common methods of consuming values from within applications



### Environment Variables

Values are exposed to applications through Operating System environment variables

### File System

Values are included within files mounted into the containers for access by applications

# Kubernetes Secrets

Kubernetes Secrets are a common storage facility utilized after obtaining values from secrets stores

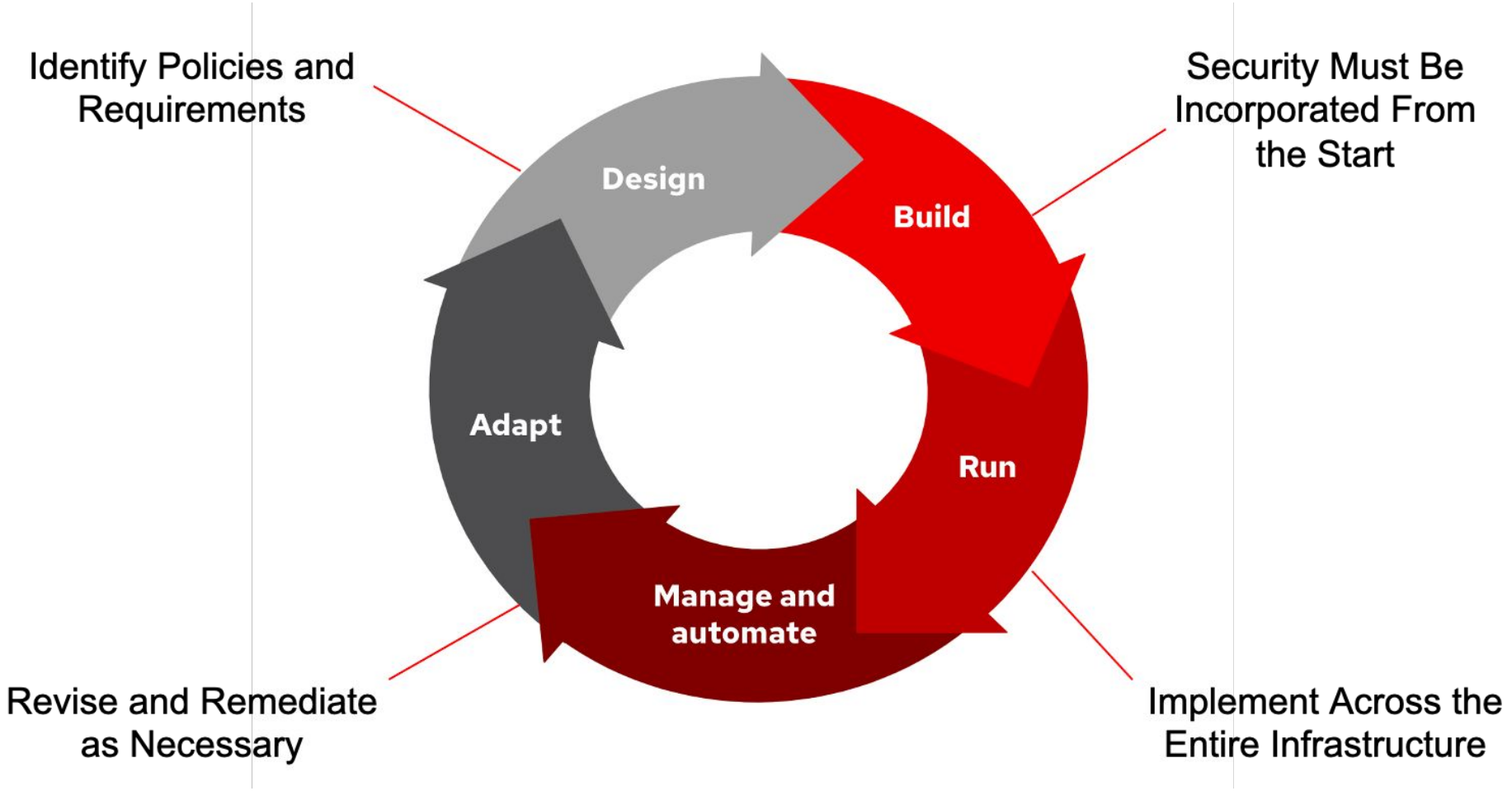
# Kubernetes Secrets

Kubernetes Secrets Store CSI Driver interacts with external secret stores while leveraging the native CSI capabilities provided by Kubernetes


## Why Create Kubernetes Secrets at All?

- Pluggable and supports multiple secrets management backends
  - AWS, Azure, GCP, Vault
- Extendable to enable user defined providers
- Applications performs standard volume mounting using `csi` volume type
- Avoids storing sensitive assets as Kubernetes Secrets

# Security is Continuous



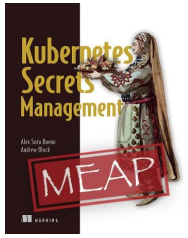
# Resources

## GitOps Principles

- OpenGitOps - Open source standards and best practices for GitOps
  - https://opengitops.dev

## OpenShift GitOps

- Hands on training and self paced exercises
  - https://learn.openshift.com




## Publications

- Kubernetes Secrets Management
  - Comprehensive look at secrets management in Kubernetes
  - Publisher Site

## Presentations

- GitOpsCon
  - Sharing knowledge around GitOps concepts and practices
  - Recordings