Navigating the Verbose Maze of Technical Jargon (skip this part)

Definition of GitOps

GitOps is a modern software delivery methodology that harnesses the power of version control systems, typically Git, to manage and automate the deployment and operation of applications and infrastructure. In the GitOps paradigm, the desired state of the system is declaratively defined and stored in a Git repository. Automated processes then continuously reconcile the actual state of the system with the declared state, ensuring consistency, reliability, and traceability in the deployment and management processes.

Definition of Continuous Integration (CI)

Continuous Integration (CI) is a software development practice where code changes from multiple contributors are frequently integrated into a shared repository. Each integration triggers an automated build and test process, helping to identify and rectify integration issues early in the development cycle. The goal of CI is to ensure that code changes are regularly and smoothly integrated, leading to a more stable codebase and reducing the chances of integration conflicts.

Definition of Continuous Deployment (CD)

Continuous Deployment (CD) is an extension of Continuous Integration (CI) where code changes that pass automated tests are automatically deployed to production environments. This approach allows for rapid and frequent delivery of new features, bug fixes, and enhancements to end-users. CD aims to minimize manual intervention and streamline the release process, enabling teams to deliver software changes quickly and reliably, while maintaining a high level of quality and stability in production.

GITOPS

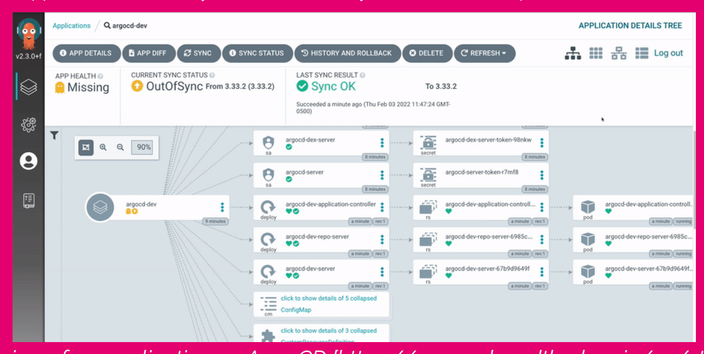
ARGOCD THE WAY WE CHOOSE

Les Filles & Les Garçons DE LA TECH

Benoît GARÇON
benoit@fgtech.fr

Discovering GitOps with Argo CD

Argo CD is an open-source platform tailored for Continuous Deployment in Kubernetes environments, driven by GitOps principles. It offers automated, declarative management, Helm support, a user-friendly UI, robust security features, audit capabilities.



Overview of an application on Argo CD [https://argo-cd.readthedocs.io/en/stable

How does Argo CD works ?

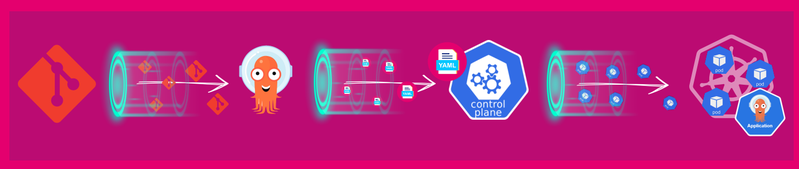
Argo CD adds four main types of resources.

Application: Define where and when a configuration (Chart, manifest, etc.) is deployed.

AppProject: Define authorization for all applications that use this AppProject.

Repository (secret): Define how to reach the git repository (URL and credentials).

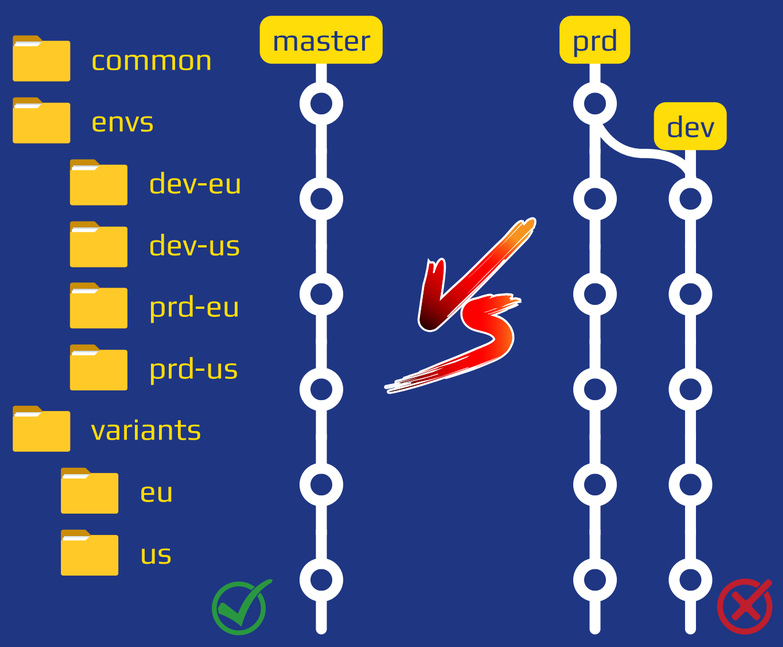
ApplicationSet: Define rules to automate the generation of Application (kind of "meta-Application").



Focus on stage 3

Master Your Environments Like a Pro

Runtime environment configurations should not be managed via branches, but rather through folders or repositories to leverage Git functionalities and simplify daily operations.



Focus on stage 4

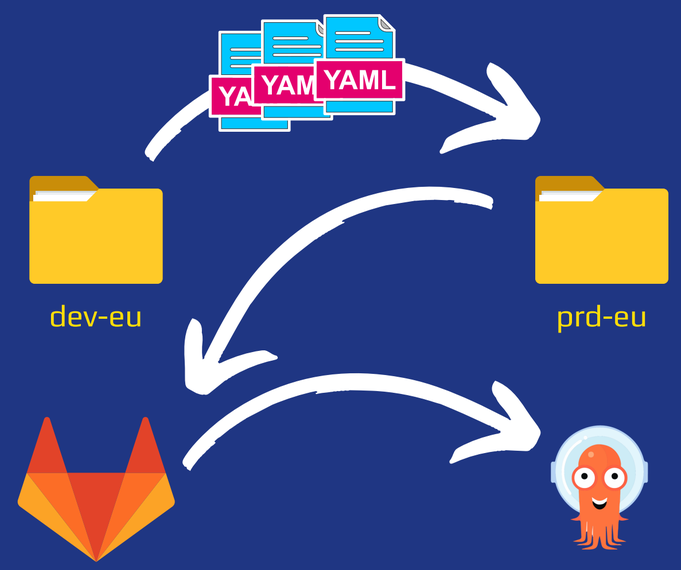
Use Secrets in a Declarative Way

Secrets should not be exposed in plaintext within configuration management; rather, it is imperative to manage them in a centralized and version-controlled manner.

Focus on stage 6

Declarative CI to the Rescue of CD

If a well-designed architecture is in place, operations such as promotion become simple copy-paste tasks that are easy to automate.



GITOPS ROADMAP

Deploy Argo CD

Distributed architecture with one ArgoCD by cluster rather than one centralized ArgoCD. During this stage, the journey kicks off with the deployment of Argo CD. Setting up this GitOps tool lays the foundation for streamlined application delivery and management.

STAGE 1

Tip: Deploying an instance of Argo CD per environment helps minimize the security impact of a fault on a target using a pull strategy rather than push.

Package apps for GitOps

As we proceed, stage two focuses on packaging applications for the GitOps approach. This involves encapsulating configurations within version-controlled repositories, paving the way for consistent and repeatable deployments.

STAGE 2

Tip: The application should be properly divided into at least three Git repositories: source code, Helm packaging, and Argo CD configuration.

App-of-Apps pattern

Advancing to stage five, the focus shifts to the App-of-Apps pattern. Argo CD's ability to manage multiple applications as a single entity simplifies complex deployments, reducing operational complexity. It enables teams to deploy with limited permissions.

STAGE 5

Tip: Implementing an App-of-Apps pattern, enables Ops to more easily manage the application landscape while granting flexibility to Devs.

Deal with secrets

In stage four, attention turns to handling secrets securely. Argo CD's integrations with secret management systems empower us to manage sensitive information without compromising on confidentiality. Using a plugin or an operator is possible.

STAGE 4

Tip: To inject secrets into Kubernetes clusters, use operators like Sealed Secret or HashiCorp Vault.

Manage environments

Moving forward, stage three shines a light on environment management. With Argo CD, orchestrating different environments becomes effortless, ensuring smooth transitions from development to testing and production.

STAGE 3

Tip: The Git repository with GitOps configuration should have a folder for each environment, rather than a branch per environment.

Back to imperative CI

The journey concludes at stage six, circling back to imperative actions when necessary. Argo CD's flexibility allows us to address unique scenarios, enabling both declarative GitOps and imperative adjustments when needed.

STAGE 6

Tip: Configure automation through continuous integration (CI) for the application promotion process to normalize daily operations.