

The background is a solid dark blue color. It is decorated with various light blue geometric elements: several thin diagonal lines, some circles of different sizes, and larger, semi-transparent rounded rectangular shapes that overlap each other. These elements are scattered across the entire frame, creating a modern, tech-oriented aesthetic.

Gitless GitOps: Más allá de Git

Agenda

01

Definición

Cómo podemos entender Gitless GitOps?

02

Ventajas

Comparar un enfoque GitOps tradicional y el enfoque Gitless GitOps

03

Notas adicionales

Información adicional sobre el enfoque

04

Demo

Pequeña demostración

05

Bibliografía

Contenido usado como bibliografía



01

Definición

Cómo podemos entender Gitless GitOps?

¿Qué es GitOps?

GitOps

Un enfoque de implementación que sincroniza continuamente la configuración de un sistema con una fuente de verdad controlada por versiones.

Conjunto de principios para operar y gestionar sistemas de forma declarativa.



Declarativo

Se declara el estado deseado del sistema. La configuración no depende de la implementación.



Versionado e inmutable

El estado deseado está versionado y se almacena de forma inmutable.



Estado extraído automáticamente



Agentes de software pueden extraer automáticamente las configuraciones del estado del sistema.



Reconciliado continuamente

Agentes de software reconcilian continuamente el estado actual con el estado deseado definido.

GitOps

≠



Git

Kubernetes

Argo

Flux

Fleet

...

Vendor-specific tools

*Son los
principios,
no las
herramientas.*

¿Qué es Gitless GitOps?

Implementación de GitOps
Implementación de GitOps que no depende de Git.

Flujo de trabajo basado en artefactos

Implementación de GitOps que utiliza artefactos en lugar de solamente archivos.

Cambio de mentalidad

Es importante implementar principios sin depender de implementaciones.

Enfoque multiusuario

Al desacoplar los artefactos del almacenamiento, se mejora la gestión de acceso y la auditoría entre diferentes equipos y entornos.

Ejecución desacoplada de Git

Mientras que Git puede ser usado como fuente de la verdad para la construcción de los artefactos en la etapa del CI, los agentes no requieren acceso a Git.



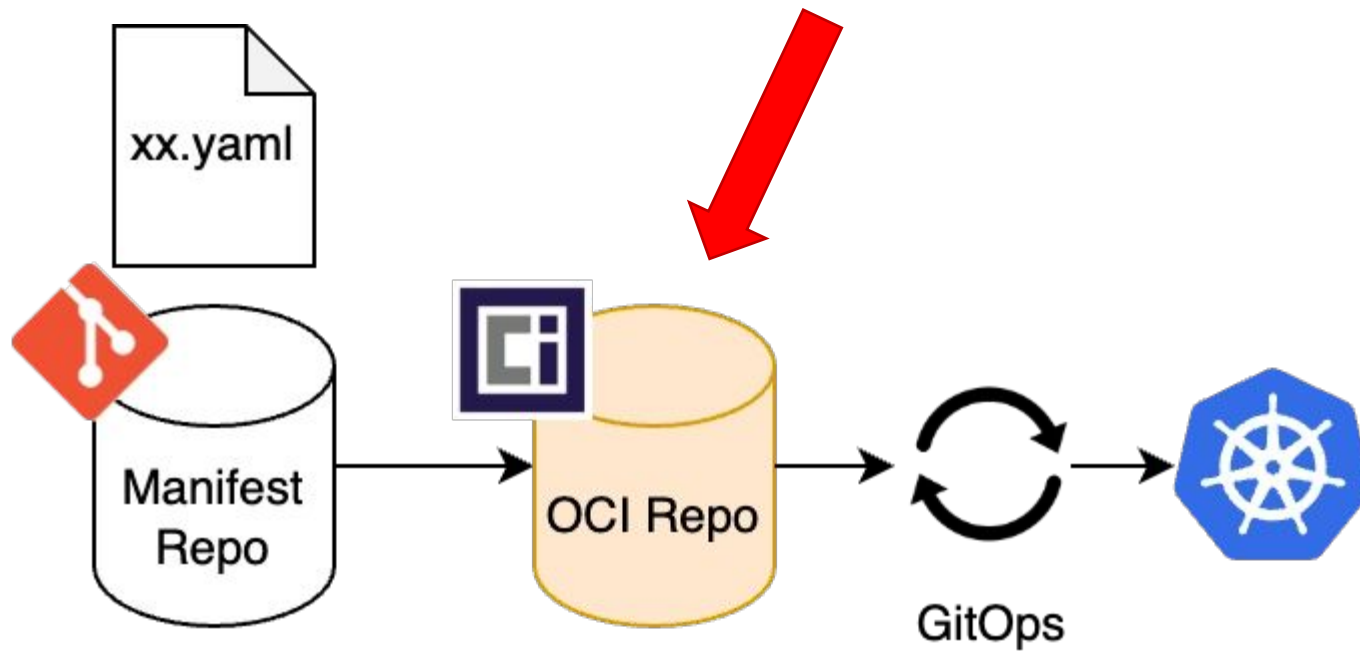
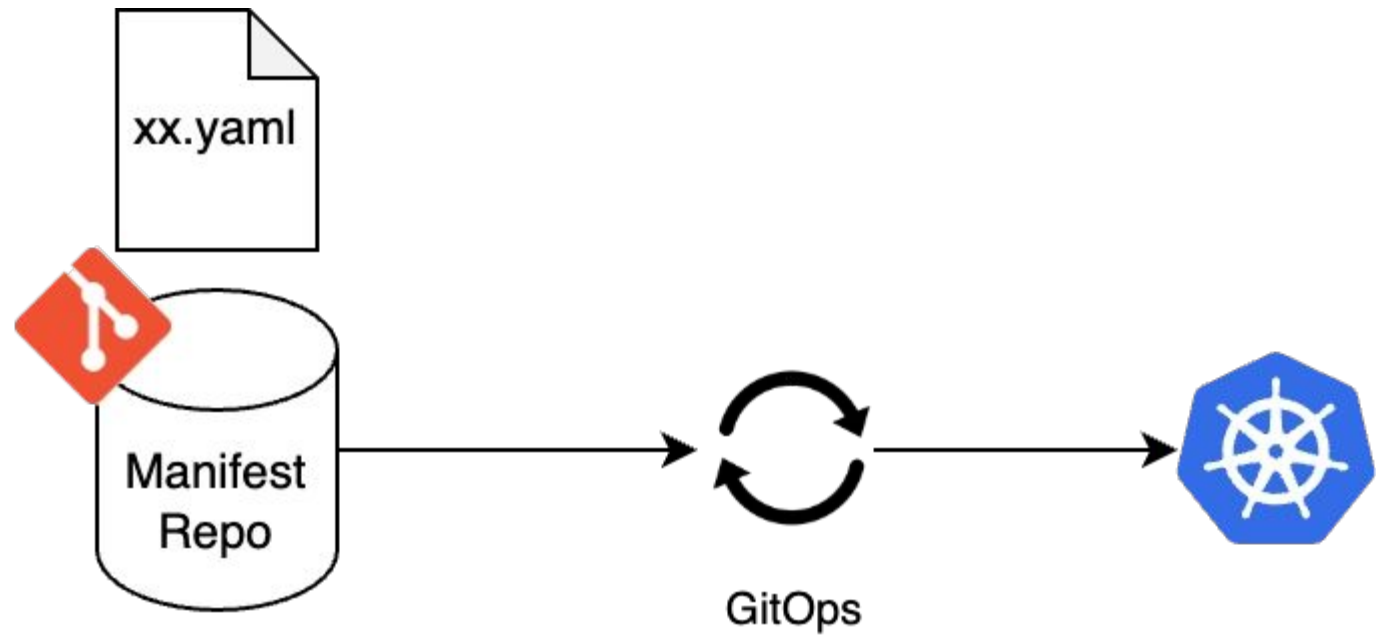


02

Ventajas

Comparar un enfoque GitOps tradicional y el enfoque Gitless GitOps

GitOps tradicional



Gitless GitOps

Dimension	CI/CD tradicional	GitOps tradicional	Gitless GitOps
Modelo de despliegue	Push-based: El pipeline de CI/CD ejecuta las acciones de despliegue	Pull-based: El agente de GitOps extrae la configuración de Git y reconcilia el estado	Pull-based: El agente de GitOps extrae del repositorio OCI y reconcilia el estado
Fuente de la verdad	Implícito en definición de pipeline o compartido con múltiples aplicaciones	Repositorio de Git	Artefacto OCI almacenado en el registro; Git puede utilizarse perfectamente como fuente remota para auditabilidad o colaboración
Flujo de cambios	Desarrollo → CI/CD pipeline → Despliegue	Desarrollo → Commit → Agente de GitOps → Despliegue	Desarrollo → Commit → Construcción y firma de artefacto → Agente de GitOps → Despliegue
Procesado de la configuración	Pipeline de CI/CD aplica las configuraciones imperativamente	El agente de GitOps puede renderizar plantillas en tiempo de ejecución	El CI se encarga del renderizado y empaquetado, mientras el CD aplica el artefacto inmutable final

Dimension	CI/CD tradicional	GitOps tradicional	Gitless GitOps
Auditabilidad	Varía; puede requerir un registro externo para rastrear los despliegues.	El historial de commits en Git equivale a una pista de auditoría.	Commit de Git + resumen/firma del artefacto = trazabilidad y procedencia completas.
Mecanismo de reversión	Reejecución manual del pipeline o reversión de cambios.	Revertir en Git a un commit anterior.	Apuntar el agente GitOps al artefacto OCI anterior (versión/resumen) para una reversión atómica.
Soporte de cumplimiento normativo	Débil a menos que se integre como complemento; difícil de aplicar políticas.	Buena pista de auditoría, pero carece de controles a nivel de artefacto.	Fuerte: soporta artefactos firmados, SBOMs, verificación de identidad OIDC y cumplimiento con SLSA.
Complejidad operacional	Alto: los pipelines codifican tanto el “qué” como el “cómo”; pueden desviarse del estado declarado.	Moderado: separación de responsabilidades más clara, pero los flujos de trabajo en Git pueden ser demasiado centrados en Git y pesados.	Bajo: separación clara entre CI (compilación) y CD (despliegue); artefactos de entrega declarativos e inmutables.

Dimension	CI/CD tradicional	GitOps tradicional	Gitless GitOps
Corrección de desviaciones	Ninguna: una vez desplegado, no hay reconciliación a menos que se reejecute el pipeline.	Reconciliación continua mediante un observador (watcher) de Git.	Reconciliación continua mediante un observador (watcher) de artefactos OCI.
Modelo de seguridad	Depende de los permisos del pipeline; puede realizar push desde CI con credenciales.	Requiere credenciales de Git dentro del clúster (SSH/PAT).	No se necesita acceso a Git dentro del clúster; utiliza autenticación nativa en la nube para OCI y firma de artefactos.
Ubicación de la lógica de plantillas	Dentro de los scripts de CI o pasos de CD.	Dentro del controlador GitOps.	Solo dentro del pipeline de CI; los agentes de CD aplican manifiestos completamente renderizados.
Mejor para	Equipos pequeños, automatización ad-hoc, flujos de trabajo simples.	Equipos que buscan auditabilidad, flujos de trabajo centrados en Git y gestión de infraestructura basada en pull.	Organizaciones que requieren alta garantía, cumplimiento normativo, escalabilidad o que operan en entornos regulados o aislados (air-gapped).



03

Notas adicionales

Información adicional sobre el enfoque

Cambios conceptuales



Desplegar artefactos, no Git.

- ★ Despliegas un artefacto versionado, firmado e inmutable, no solo una carpeta de Git.



Mayor relevancia para CI/CD

- ✓ CI integra los cambios en una única base de código.
- ✓ CD despliega el artefacto creado por CI.



Seguridad primero

- Las firmas de los artefactos se verifican antes del despliegue.
- No se necesita Git dentro del clúster.
- Se alinea con los estándares **Zero-Trust** y **SLSA**.

Beneficios estratégicos



Seguridad de la cadena de producción

- ❑ Fomenta el uso de gestión de identidad nativa en la nube.
- ❑ No se requieren secretos de Git para la gestión del estado.



Desempeño

- ❖ Posibilidad de almacenamiento en caché.
- ❖ Extracciones (pulls) más sencillas que diferencias (diffs).



Auditabilidad mejorada

- Pasos más auditables.
- Los metadatos del artefacto mejoran la trazabilidad.



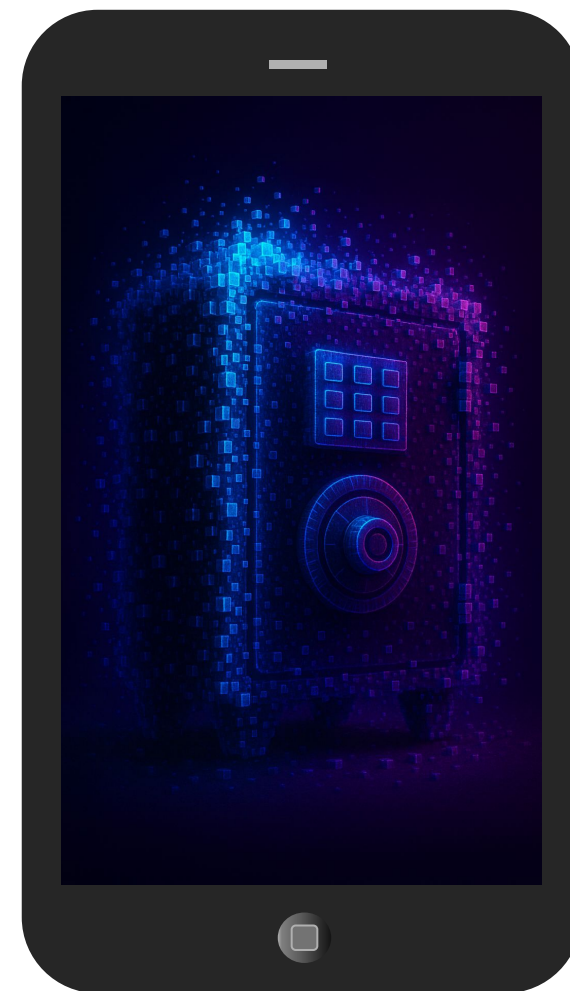
Modularidad

- ★ Los artefactos pueden crearse de forma independiente para diferentes partes del sistema.



Portabilidad

- ★ El almacenamiento puede cambiarse fácilmente.



Consideraciones adicionales

Gestión de artefactos y seguimiento de procedencia

Este enfoque permite una fácil alineación con las prácticas modernas de gestión de la cadena de suministro de software.

Despliegues inmutables

El uso de artefactos para despliegues reduce significativamente el riesgo de “drift” entre entornos y promueve implementaciones consistentes y reproducibles en múltiples entornos.



Mayor carga operativa

Introducir una arquitectura más compleja puede generar desafíos operativos que se traducen en una carga operativa para los equipos.

Mejor cumplimiento normativo

Este enfoque es ideal para industrias que requieren cumplimiento con estándares como **NIST, FedRAMP o SOC 2** debido a la naturaleza del despliegue.

Buenas prácticas

Uso de Git como fuentes de registro

Esto permite la aprobación humana de cambios y la trazabilidad ascendente.

Producción de artefactos firmados

Esto permite una mejor procedencia determinista de la compilación y el cumplimiento normativo.

Prefiere inmutabilidad para despliegues

Esto ayuda a prevenir “drifts” y mejora la fiabilidad, así como la repetibilidad.

Fuerza verificación de firmado para CD

Garantizar un canal de entrega confiable con prueba de origen y trazabilidad sencilla.


Alcanza una granularidad lógica

Esto permite la promoción selectiva, la reversión y la auditoría.

Adopta un estándar de versionado

Adoptar un estándar facilita la gestión del ciclo de vida y los procedimientos de reversión.

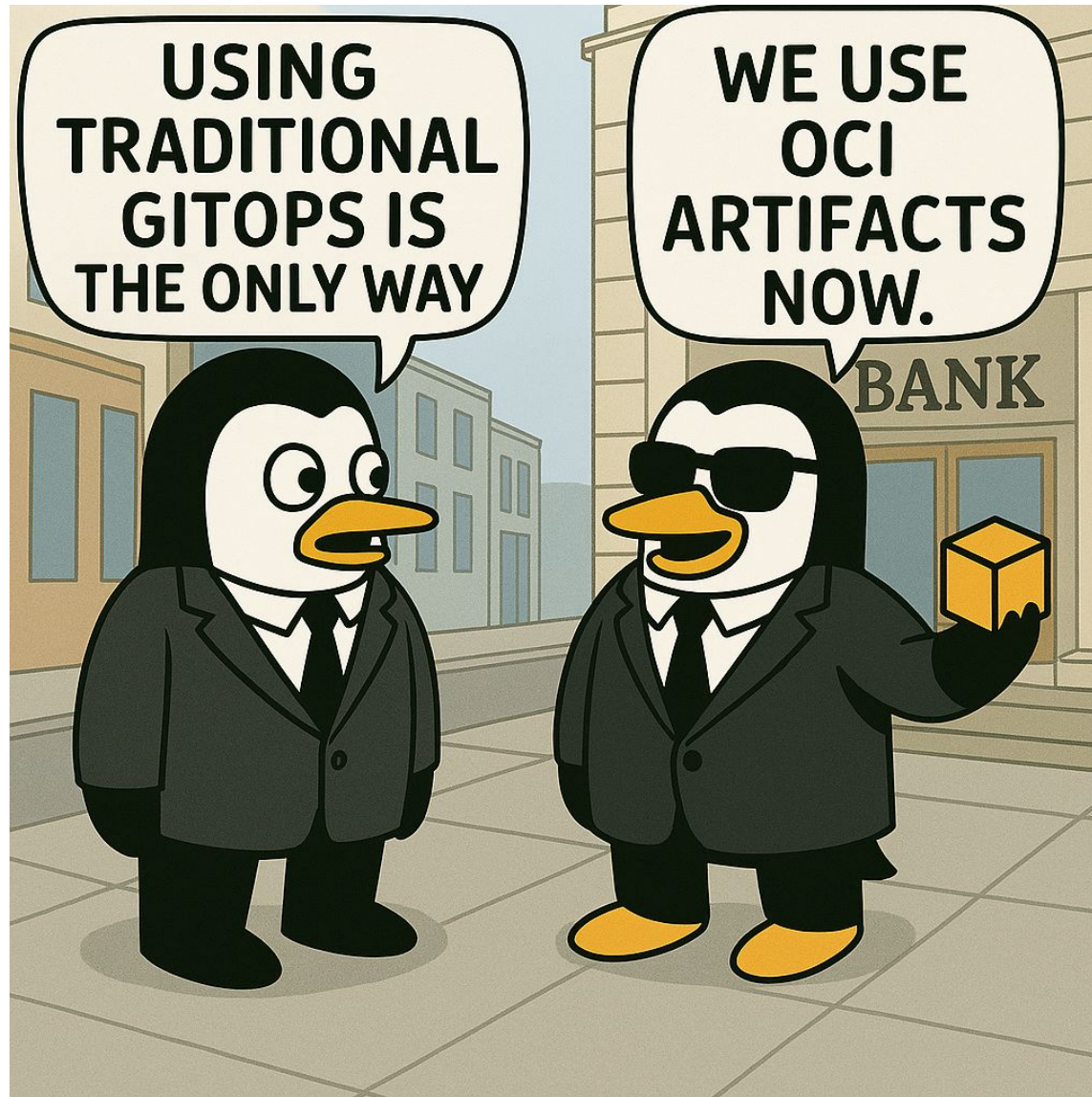




04

Demo

Pequeña demostración





05

Bibliografía

Contenido usado como bibliografía

Bibliografía relevante I

- *What is GitOps (2025) OpenGitOps. Available at:*
- Kikuchi, T. (2025) *Introduction to Gitless GitOps: A new OCI-centric and secure architecture, DEV Community. Available at:*

- *Flux D2 reference architecture – Gitless GitOps for secure multi-tenancy (2025) ControlPlane. Available at:*
- ControlPlane (2025) *D2 reference architecture, D2 Reference Architecture - ControlPlane Enterprise for Flux CD. Available at:*

- ControlPlane (2024) *D1 reference architecture, D1 Reference Architecture - ControlPlane Enterprise for Flux CD. Available at:*

Bibliografía relevante II

- *Argo Project (2025) Argo CD - Declarative GitOps CD for Kubernetes. Available at:*
- *ControlPlane Enterprise for Flux CD (2024) GitHub. Available at:*
- *The Kubernetes Authors (2025) kind. Available at:*
- Codefresh (2024) Argo CD vs. Flux: 6 key differences and how to choose. Available at:
- Levan, M. (2025) *Understanding GitOps Pros and cons, Devtron Blog. Available at:*
- 'CNCF GitOpsCon Europe' (2025).