

The background is a solid dark blue color. It is decorated with various light blue geometric elements: several thin diagonal lines of varying lengths, and several circles of different sizes. Some of these circles are partially obscured by larger, semi-transparent, rounded rectangular shapes that also have a light blue tint. The overall aesthetic is modern and tech-oriented.

Gitless GitOps: Beyond Git

Agenda

01

Definition

Explore what Gitless GitOps is

02

Advantages

Compare a traditional GitOps approach VS a Gitless GitOps approach

03

Additional notes

Some relevant info regarding the Gitless approach

04

Demo

Show a demo

05

Bibliography

Content for further reading



01

Definition

Explore what Gitless GitOps is

What is GitOps?

GitOps

A deployment approach that continuously synchronizes system configuration with a version-controlled source of truth.

Set of principles for operating and managing systems in a declarative way.



Declarative

Desired system state is declared.
Configuration is not dependent on implementation.

Pulled automatically



Software agents can pull the system state configurations automatically.



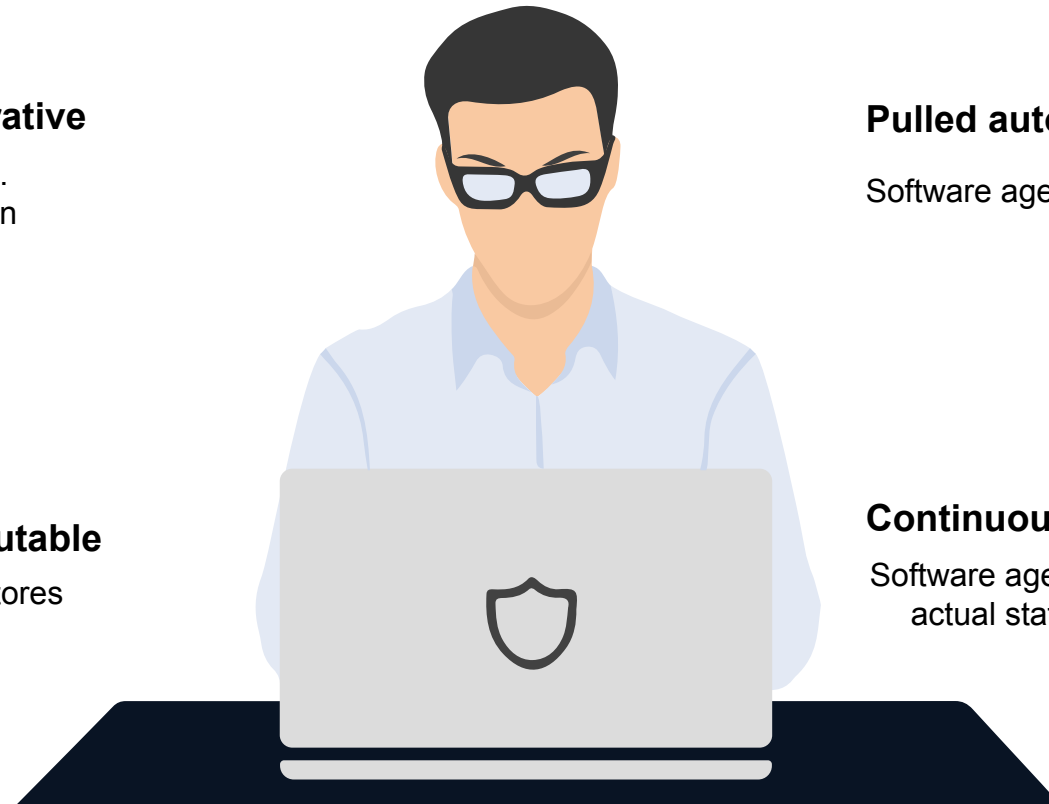
Versioned and immutable

Desired state is versioned and stores in an immutable way.

Continuous reconciling



Software agents continuously reconcile the actual state to the desired defined state.



GitOps

≠

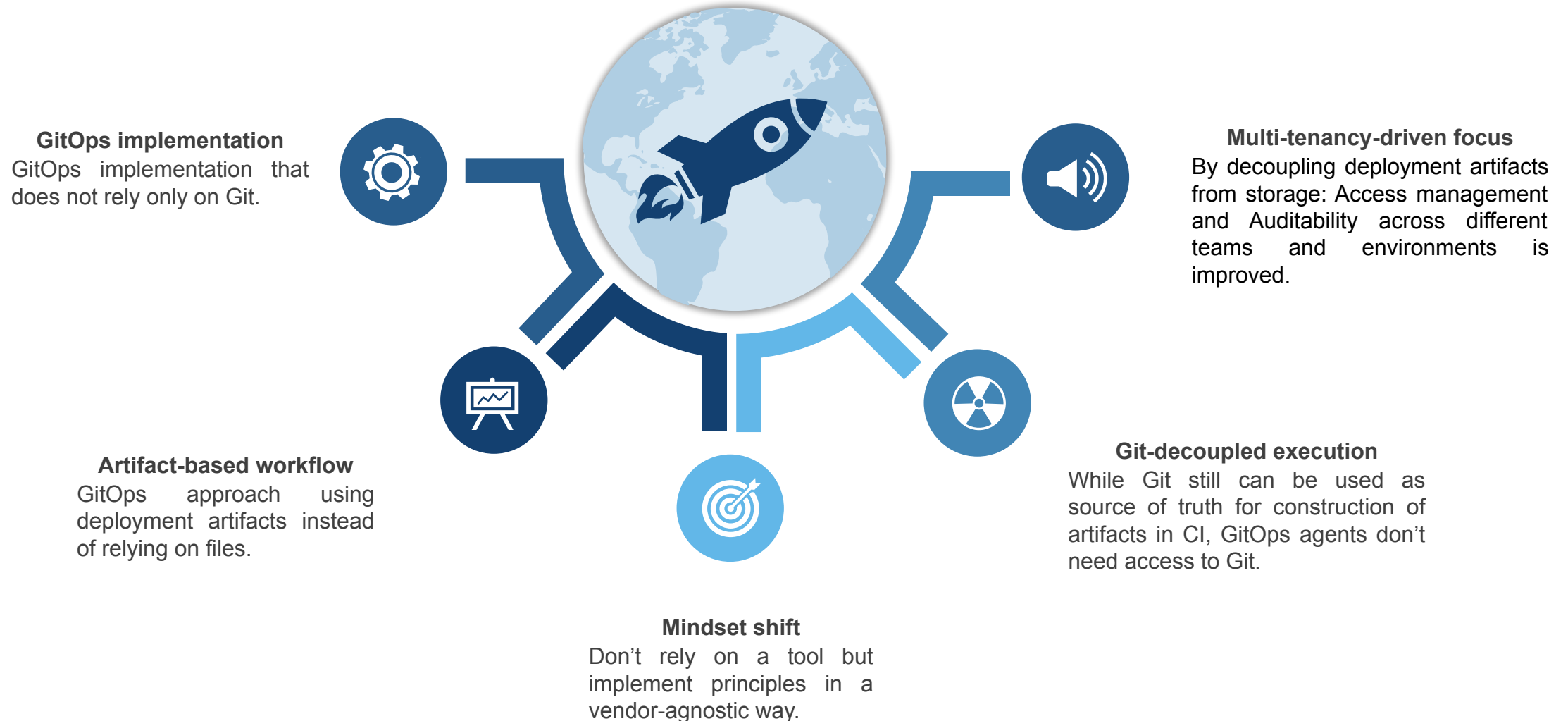
{
Git
Kubernetes
Argo
Flux
Fleet


...

Vendor-specific tools

*It's the
principles,
not the
tools*

What is Gitless GitOps?



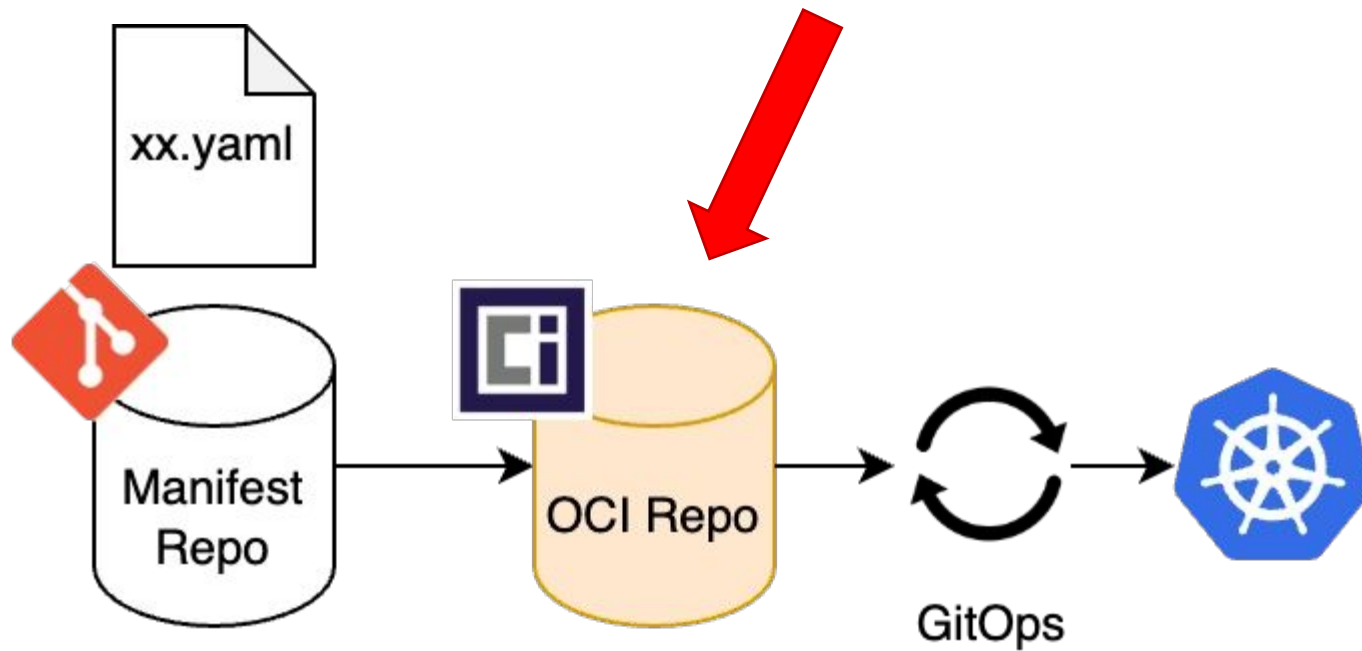
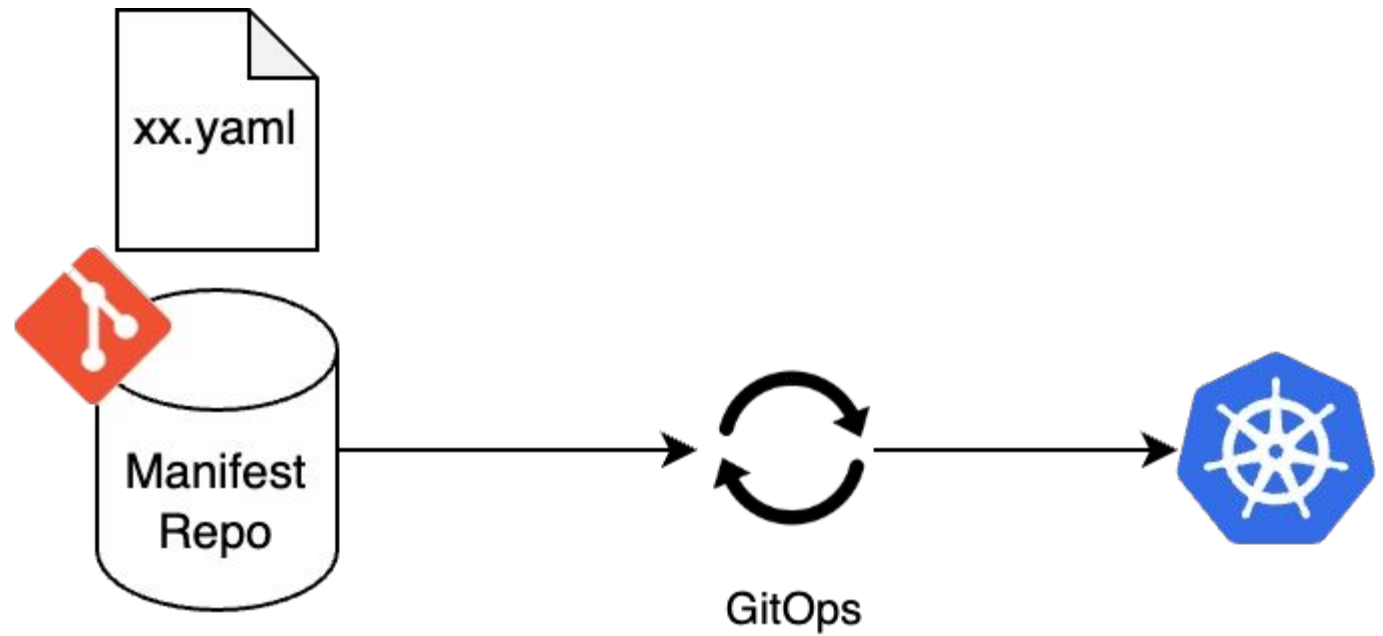
The background is a solid blue color with various abstract geometric shapes in lighter shades of blue. These include circles, lines, and rounded rectangles, some of which are semi-transparent, creating a layered effect. The shapes are scattered across the left and top portions of the slide.

02

Advantages

Compare a traditional GitOps approach VS a Gitless GitOps approach

Traditional GitOps



Gitless GitOps

Dimension	Traditional CI/CD	GitOps	Gitless GitOps
Deployment Model	Push-based: CI/CD pipeline executes deploy actions (e.g., kubectl apply)	Pull-based: GitOps agent pulls from Git repo and reconciles state	Pull-based: GitOps agent pulls from OCI registry and reconciles state
Source of Truth	Often implicit in pipeline definitions or multiple YAML repos	Git repository (e.g., main branch of config repo)	OCI artifact in container registry; Git is still upstream for audit/collab
Change Flow	Dev → CI/CD pipeline → Deploy	Dev → Git commit → GitOps pull → Deploy	Dev → Git commit → CI builds/signed artifact → OCI pull → Deploy
Configuration Processing	CI/CD pipeline renders/apply configs imperatively	GitOps agent may render templates at runtime (e.g., Kustomize/Helm)	CI does rendering and packaging; CD applies final, immutable artifact

Dimension	Traditional CI/CD	GitOps	Gitless GitOps
Auditability	Varies; may need external logging to trace deployments	Git commit history = audit trail	Git commit + artifact digest/signature = full traceability & provenance
Rollback Mechanism	Manual pipeline re-run or reversion of changes	Git revert to previous commit	Point GitOps agent to prior OCI artifact (version/digest) for atomic rollback
Compliance Support	Weak unless bolted-on; hard to enforce policies	Good audit trail but lacks artifact-level controls	Strong: supports signed artifacts, SBOMs, OIDC identity verification, SLSA compliance
Operational Complexity	High: pipelines encode both “what” and “how”; can drift from declared state	Moderate: clearer separation of concerns, but Git workflows can be Git-centric and heavyweight	Low: clean separation of CI (build) and CD (apply); declarative + immutable delivery artifacts

Dimension	Traditional CI/CD	GitOps	Gitless GitOps
Drift Correction	None – once deployed, no reconciliation unless pipeline re-run	Continuous reconciliation via Git watcher	Continuous reconciliation via OCI artifact watcher
Security Model	Relies on pipeline permissions; may push from CI with credentials	Requires Git credentials in-cluster (SSH/PAT)	No Git access needed in cluster; uses cloud-native OCI auth + artifact signing
Templating Logic Location	Inside CI scripts or CD steps	Inside GitOps controller (e.g., runtime Helm rendering)	Inside CI pipeline only; CD agents apply fully rendered manifests
Best Fit For	Small teams, ad-hoc automation, simple workflows	Teams wanting auditability, Git-centered workflows, and pull-based infra management	Organizations needing high assurance, compliance, scale, or operating in regulated/air-gapped environments

The background is a gradient of blue shades, from a darker blue on the left to a lighter blue on the right. It is decorated with various geometric elements: several thin, light blue diagonal lines; several circles of different sizes, some solid and some outlined; and several elongated, rounded rectangular shapes in various shades of blue, some overlapping each other.

03

Additional notes

Relevant info regarding the Gitless approach

Conceptual shifts



Deploy artifacts not Git

- You deploy a versioned, signed and immutable artifact, not only a Git folder.



Improved relevance for CI/CD

- ✓ CI integrates changes in a single codebase
- ✓ CD deploys the artifact created by CI



Security-first

- ✓ Artifact signatures are verified before deployment.
- ✓ No Git needed in-cluster.
- ✓ Aligns with **Zero Trust** and **SLSA** standards.

Strategic benefits



Supply chain security

- Promotes usage of cloud-native identity management.
- No Git secrets needed for state management.



Performance

- Possibility of caching.
- Easier pulls than diffs.



Better auditability

- More auditable steps.
- Artifact metadata improves traceability.



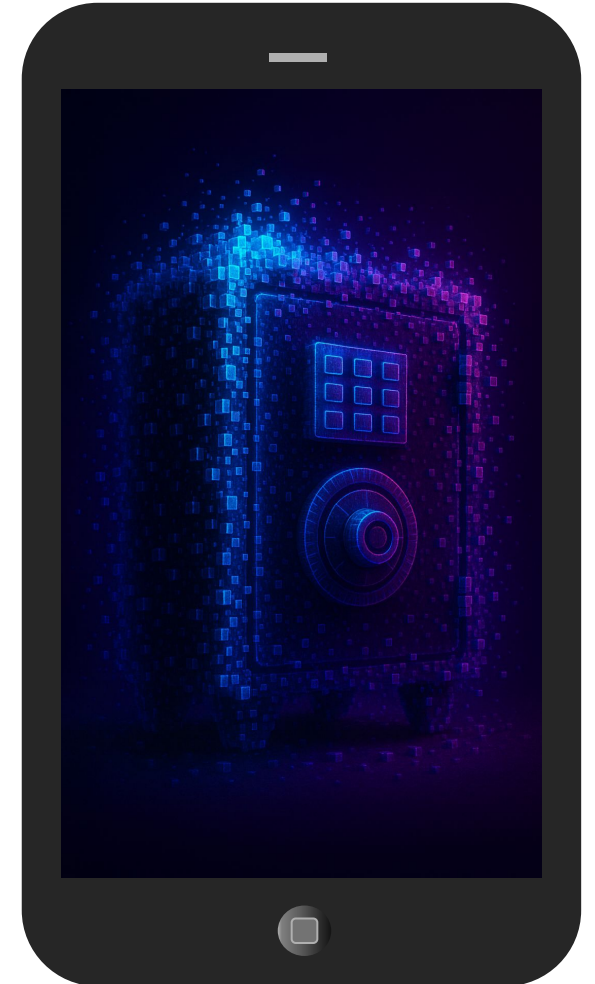
Modularity

- Artifacts can be created independently for different parts of system



Portability

- Storage can be easily changed



Additional considerations

Artifact management and provenance tracking

This approach allows for easy alignment with modern software supply chain management practices.

Immutable deployments

Using artifacts for deployments, significantly reduces the risk of "drift" between environments and promotes consistent, reproducible rollouts across multiple environments.



Increased operational overhead

Introducing a more complex architecture can result in operational challenges that lead to operational burden for teams

Improved compliance

This approach is ideal for industries that require compliance with standards like NIST, FedRAMP, or SOC 2 because of deployment's nature.

Good practices

Use Git as the source of record

This allows for human change approval and upstream traceability

Produce signed deployment artifacts

This allows for better deterministic build provenance and compliance enforcement

Prefer immutability for deployment

This helps prevent “drifts” and improves reliability as well as repeatability

Enforce signature verification for CD

Ensure a trusted delivery channel with proof of origin and easy traceability


Structure artifacts for logical granularity

This enables selective promotion, rollback, and audit

Adopt versioning standard for artifacts

Adopting a standard allows for easier lifecycle management and rollback procedures.

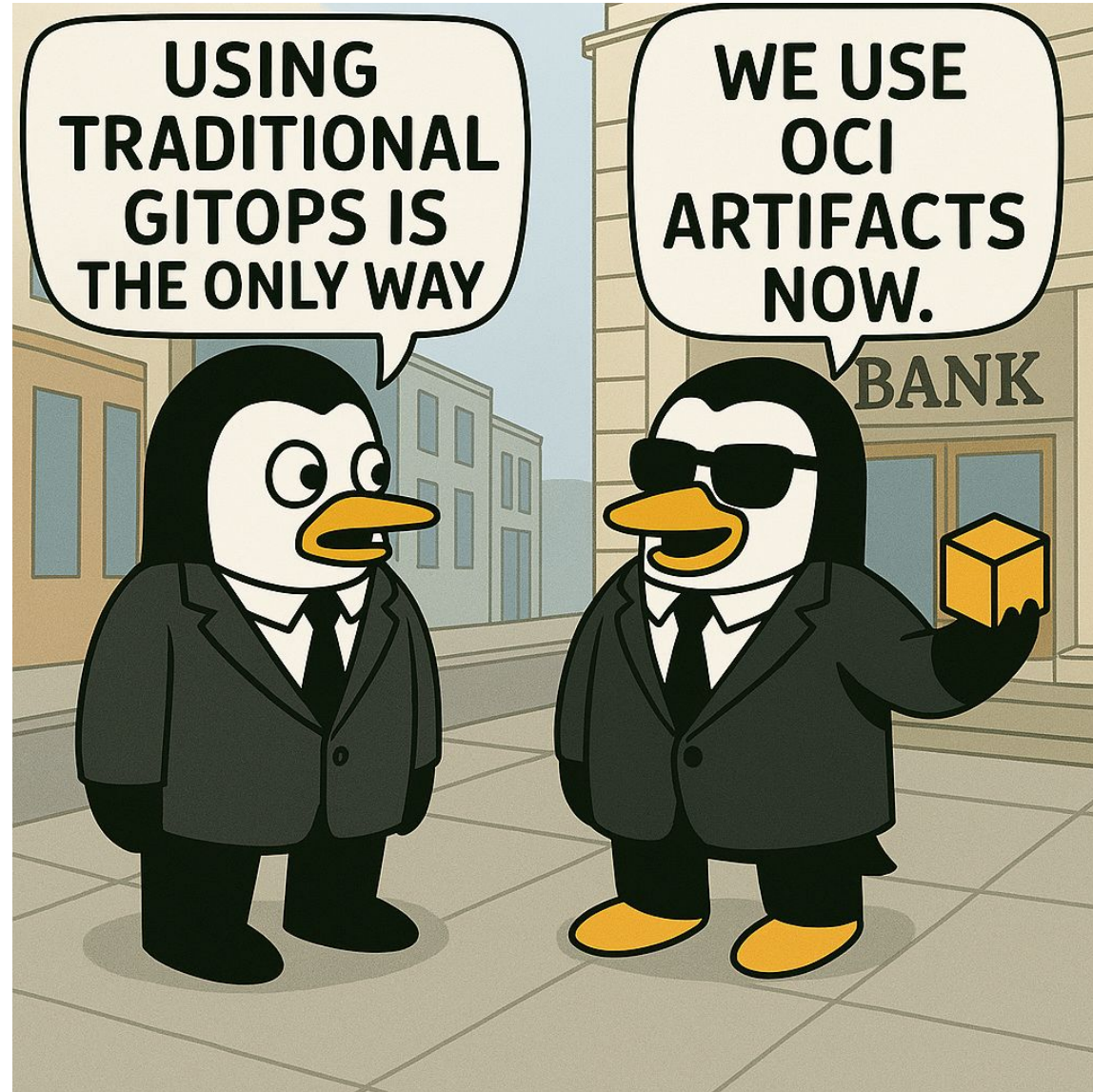




04

Demo

Show a demo





05

Bibliography

Content for further reading

Relevant bibliography 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:*

Relevant bibliography 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).