Platform Engineering
UNIVERSITY

GITOPS FOR PLATFORM ENGINEERING

Introduction and Motivation

MODULE 01

Artem Lajko

- Head of Platform Engineering at iits
- Ambassador for Platform Engineering
- CNCF Kubestronaut
- Published Author

Book: Implementing GitOps with Kubernetes

What is this course about?

01

Introduction and Motivation

02

GitOps Fundamentals and Core Principles

03

GitOps Architecture, Patterns, and Anti-Patterns

04

GitOps Tooling 101 – Argo CD, Flux CD and Sveltos

05

GitOps for Enterprises – Scaling and Security

06

Outlook and Trends – GitOps is a Sprawl of Configs

Why GitOps

Why GitOps?

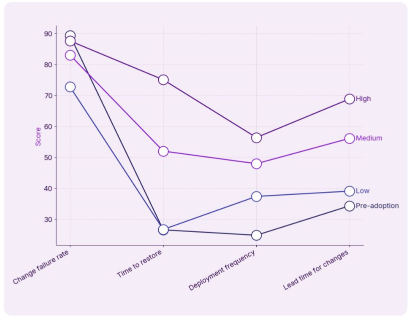
1000 Reasons + 1

- Git becomes your deployment API
- Fully versioned infrastructure
- Perfect reproducibility
- Bulletproof auditability
- Developer-friendly, Ops-safe
- Lower risk, higher velocity

> GitOps is a **contract** between **Agents** and **Humans**!

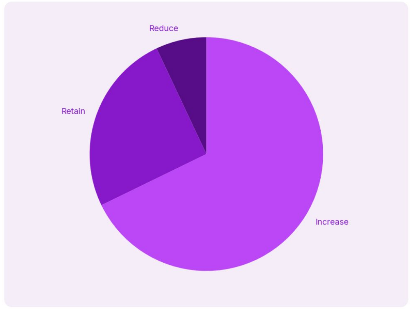
Why GitOps? Let’s look at the numbers (I)

1. Better software delivery



High-performing GitOps teams were more likely to exhibit high software delivery performance as measured by the DORA 4 key metrics, change failure rate, time to restore, deployment frequency, and lead time for changes.

4. Adoption is increasing

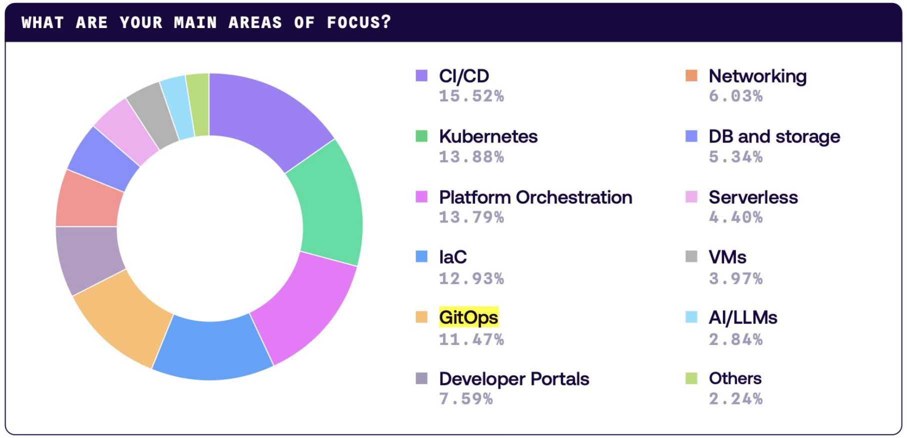


Most organizations (93%) plan to continue or increase their GitOps adoption. Organizations that have yet to fully adopt GitOps practices are more likely to reduce or stop their rollout.

https://octopus.com/publications/state-of-gitops-report

Main focus of platform engineers' work

For those primarily working in platform engineering, the main areas of focus are diverse, reflecting the range of responsibilities in the field. CI/CD tops the list at 15.52%, followed closely by Kubernetes (13.88%) and Platform Orchestration (13.79%). Infrastructure as Code (IaC) is also significant at 12.93%, alongside GitOps (11.47%). Other areas like Developer Portals, Networking, and Serverless see lower engagement, but each represents crucial elements of platform engineering. Emerging fields like AI/LLMs also make a small yet growing appearance at 2.84%.



WHAT ARE YOUR MAIN AREAS OF FOCUS?

CI/CD	15.52%	Networking	6.03%
Kubernetes	13.88%	DB and storage	5.34%
Platform Orchestration	13.79%	Serverless	4.40%
IaC	12.93%	VMs	3.97%
GitOps	11.47%	AI/LLMs	2.84%
Developer Portals	7.59%	Others	2.24%

https://platformengineering.org/reports/state-of-platform-engineering-vol-3

Why GitOps? Let’s look at the numbers (II)

50%

Infrastructure Costs

REDUCED

<5 MIN

Incident Response Time

MEDIAN

99.95%

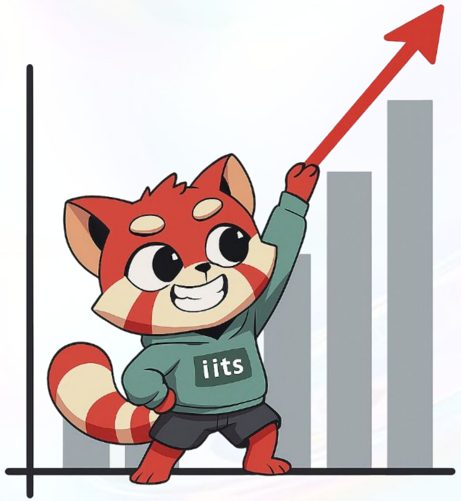
Service Level Agreement

FROM 98% TO 99.95%

0 MIN

Release Downtime

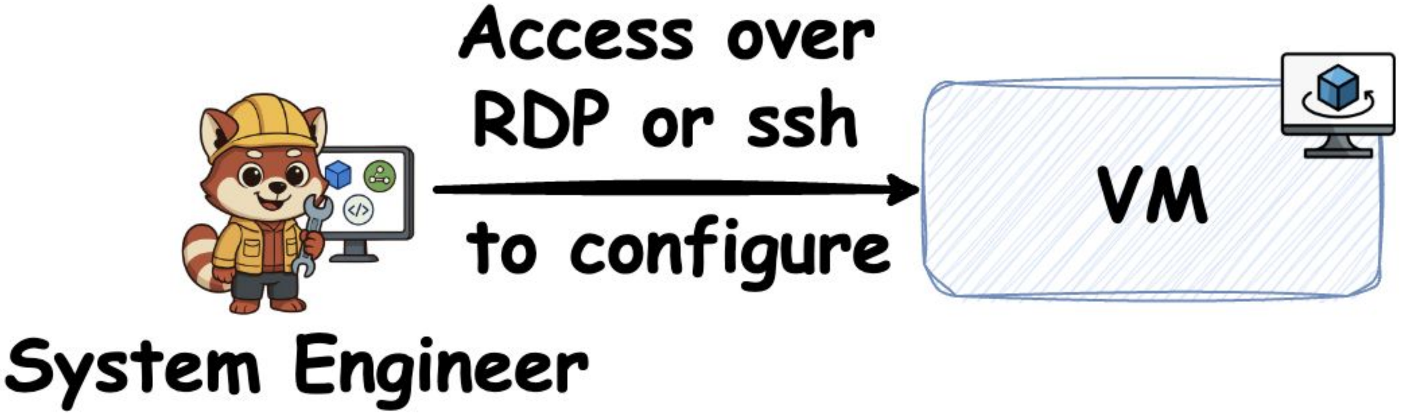
FROM 180 MIN TO 0



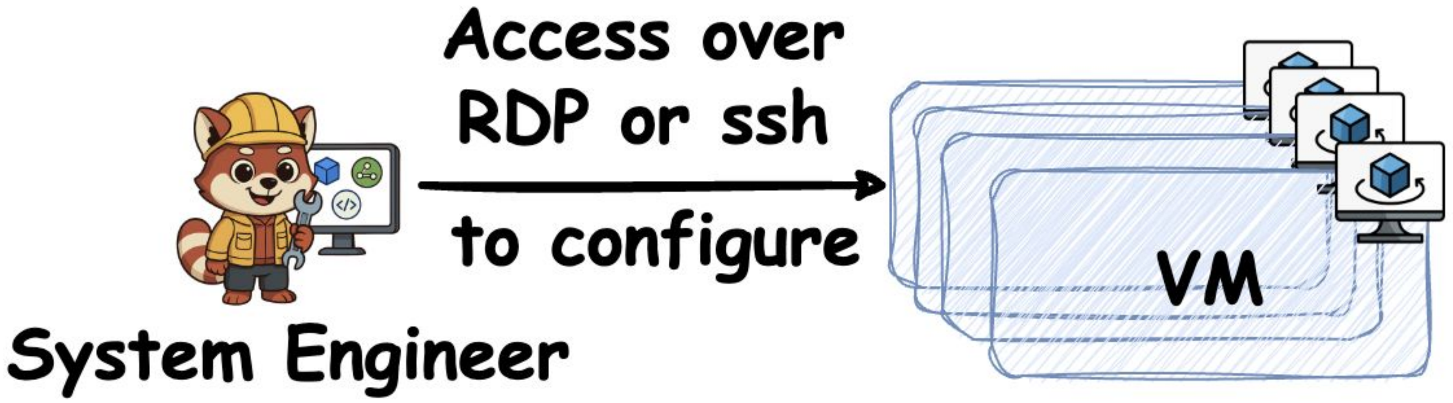
iits-consulting: Migration from VMS and Rancher (RKE) to GitOps with Argo CD (KumoOps Stack)

The Wild West Era – Before DevOps and GitOps

The Wild West Era — Before DevOps and GitOps (I)



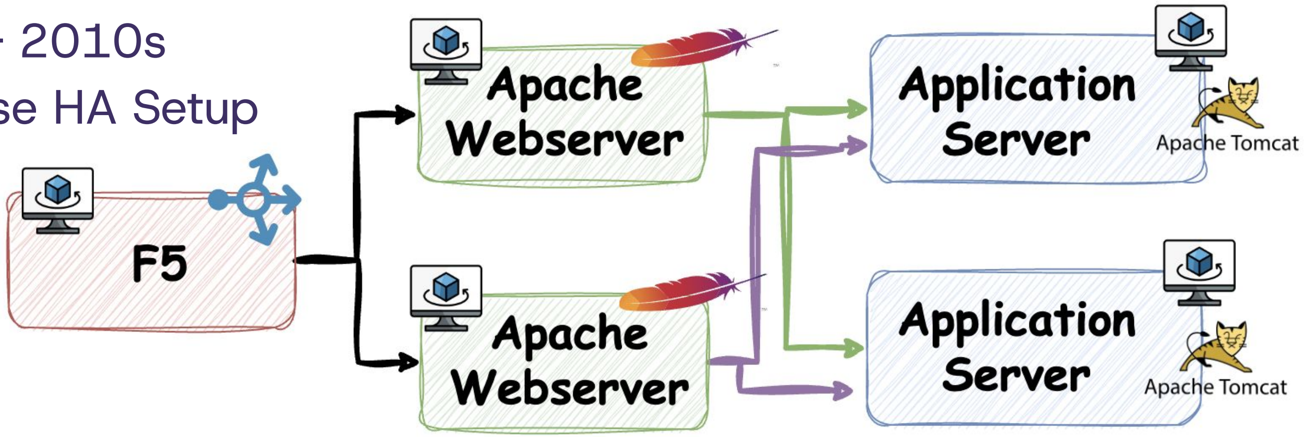
The Wild West Era — Before DevOps and GitOps (II)



The Wild West Era — Before DevOps and GitOps (III)

2000s – 2010s

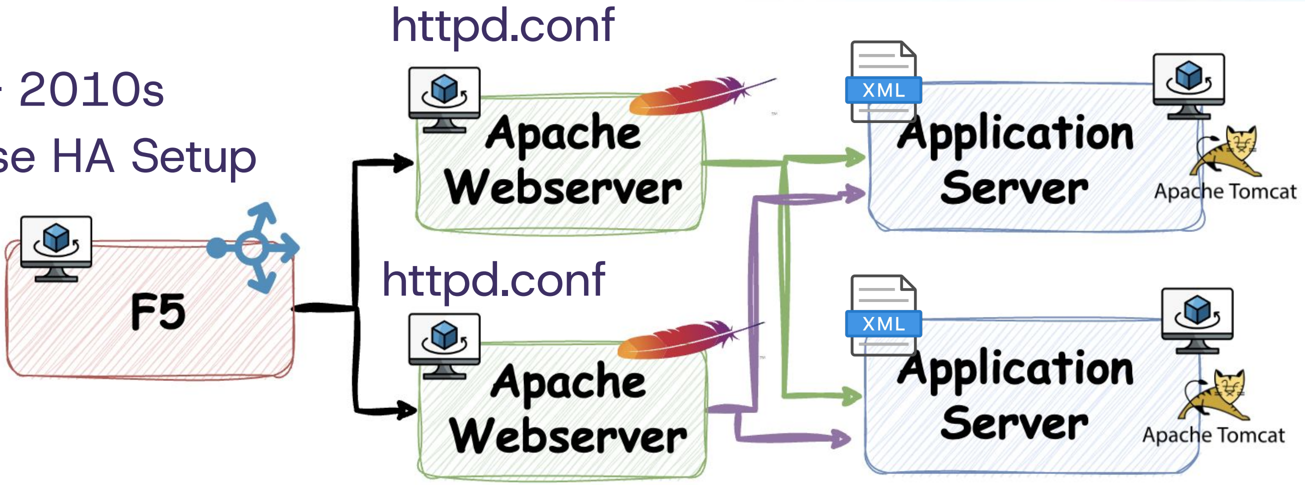
Enterprise HA Setup



The Wild West Era — Before DevOps and GitOps (IV)

2000s – 2010s

Enterprise HA Setup



Doesn’t **scale**!

The Birth of Infrastructure as Code

The Birth of Infrastructure as Code (IaC) – I

ANSIBLE

puppet

CHEF

HashiCorp Terraform

SALTSTACK

The Birth of Infrastructure as Code (IaC) – II

Everything as Code

Infrastructure as Code

Policy as Code

Network as Code

Configuration as Code

and so - - - - -

Everything becomes **declarative**!

New Tooling…Similar Challenge… Still Running on My Machine

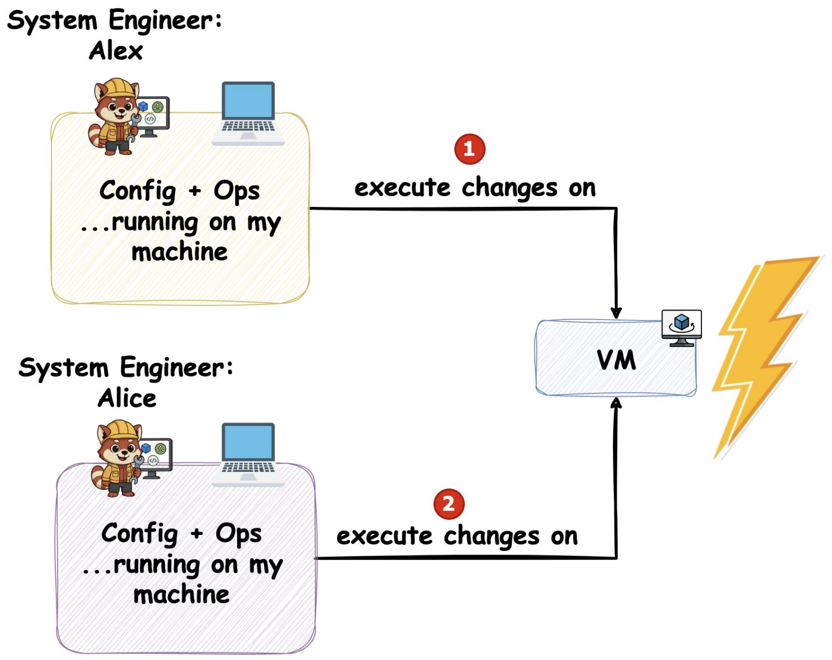
Config and Ops Still Run on My Machine

We now use “X as Code” (let’s call it config), but it still runs on engineers’ machines.

If Alex changes the VM first, there is no shared state—because not every tool supports state.

Then Alice may apply her change, but it can overwrite Alex’s change and likely break the system.

Config and ops still run on the local machine.



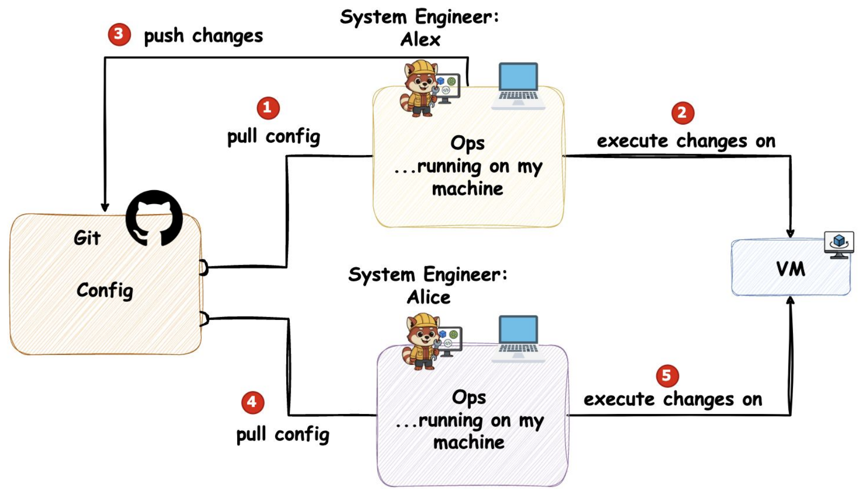
Local Ops!

Ops still running on my Machine!

We have now split config and ops. Config lives in a central Git repo, but operations still happen on engineers' machines.

Alex makes a change on the VM and commits the config to Git. Alice pulls the change, merges it with her own, and runs the change on the VM.

- Config is in Git
- Ops still run on the local machine



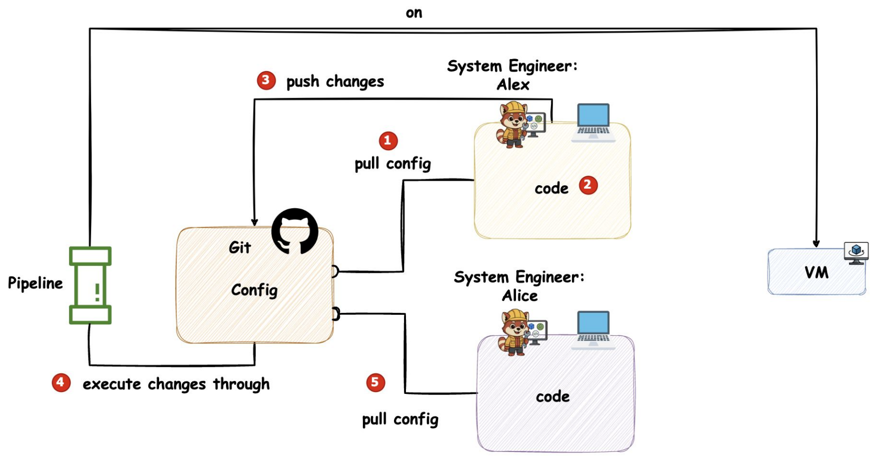
Distributed Ops!

Pipeline Ops: Code still running on my Machine!

We have now split config and ops. Config lives in a central Git repo, and operations are executed through a pipeline.

Alex pulls the config, makes a change, and commits it. This triggers an event that runs the pipeline. The pipeline applies the changes to the VM.

- Config is in Git
- Ops are through pipelines



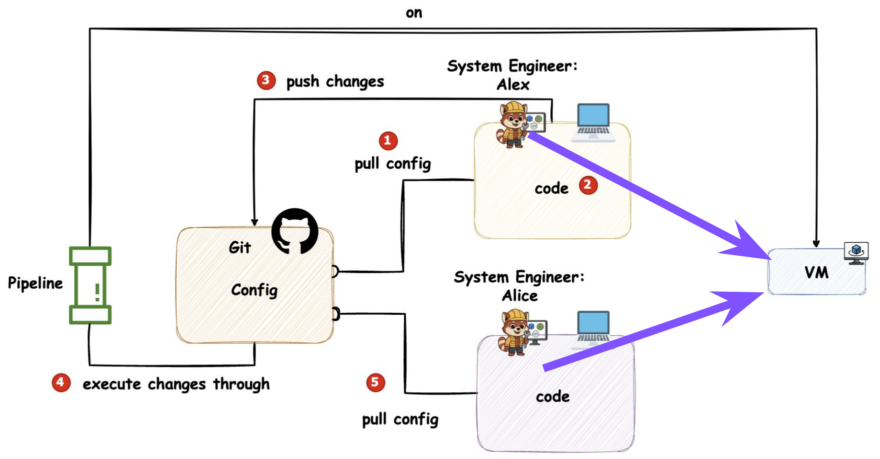
Pipeline Ops!

Pipeline Ops: Code still running on my Machine!

We have now split config and ops. Config lives in a central Git repo, and operations are executed through a pipeline.

Alex pulls the config, makes a change, and commits it. This triggers an event that runs the pipeline. The pipeline applies the changes to the VM.

- Config is in Git
- Ops are through pipelines

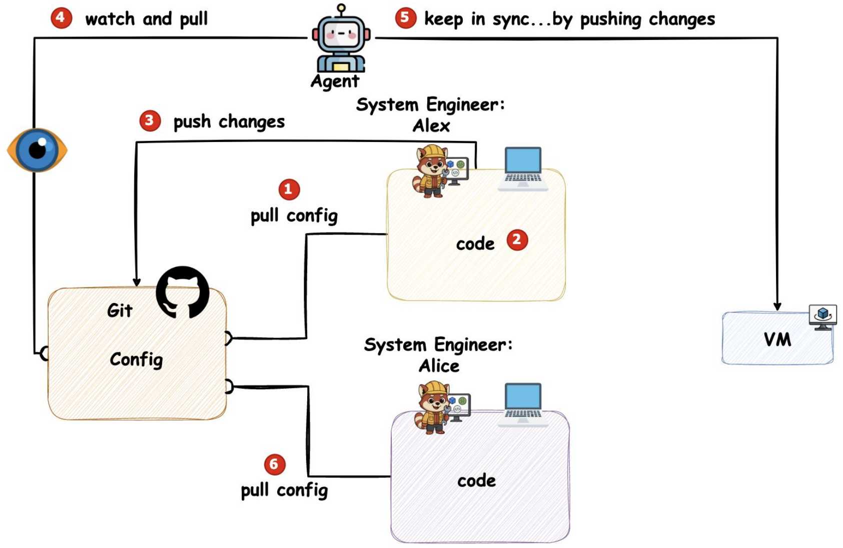


Event? 1 Day, 1 Week, 1 Month...?
Drift...

Agent Ops: Code still running on my Machine!

We have now split config and ops. Config lives in a central Git repo. An agent watches the repo's desired state for changes. If it detects drift, it pulls the config and reconciles the target environment to match the actual state.

- Config is in Git
- Ops are running through an Agent (Machine)!



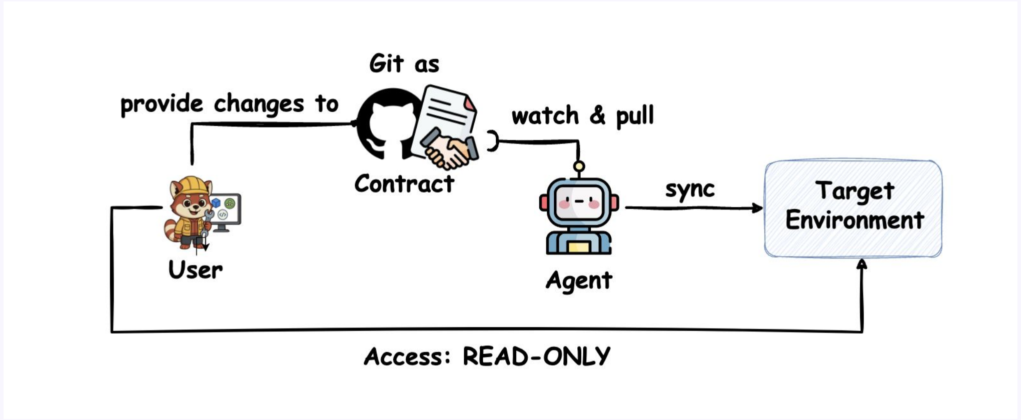
Execute Ops through Git as a declarative contract!

You Have a Contract with the Agent

The user has a contract with the agent. The agent promises to keep the target environment in sync.

User: I put everything in Git. Git becomes the single source of truth for the contract.

Agent: I fulfill the contract by continuously reconciling the desired state in Git with the actual state of the target environment.



Why Do We Want GitOps?

Why Do We Want GitOps?

- ❏ If you build a platform, GitOps is the glue for continuous delivery
- ❏ Single source of truth (visibility, history, …)
- ❏ Git as a common tool to store config as data, with one shared language
- ❏ Git as a contract between humans and agents
- ❏ Disaster recovery and backups built in
- ❏ **Security-friendly:** strong audit trail
- ❏ **AI-ready:** it can explain what happened, why, when, and how

Recap and Learning Objectives

A quick look at what's coming next.

↓

What YOU will learn

- What is GitOps
- How to manage a fleet of 1,000 clusters
- How to build your own platform distribution
- Which GitOps topologies exist
- A comparison of GitOps tools
- Why GitOps is more than just tooling

…and more!

→

How YOU will learn

- Understand the core ideas of GitOps
- See a real enterprise example
- Hands-on exercises with examples based on Kubara
- Learn how to choose the right solution
- Deep dive into the most popular GitOps tools
- Understand the culture behind GitOps

Let's not spoil too much yet.