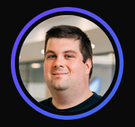
anny

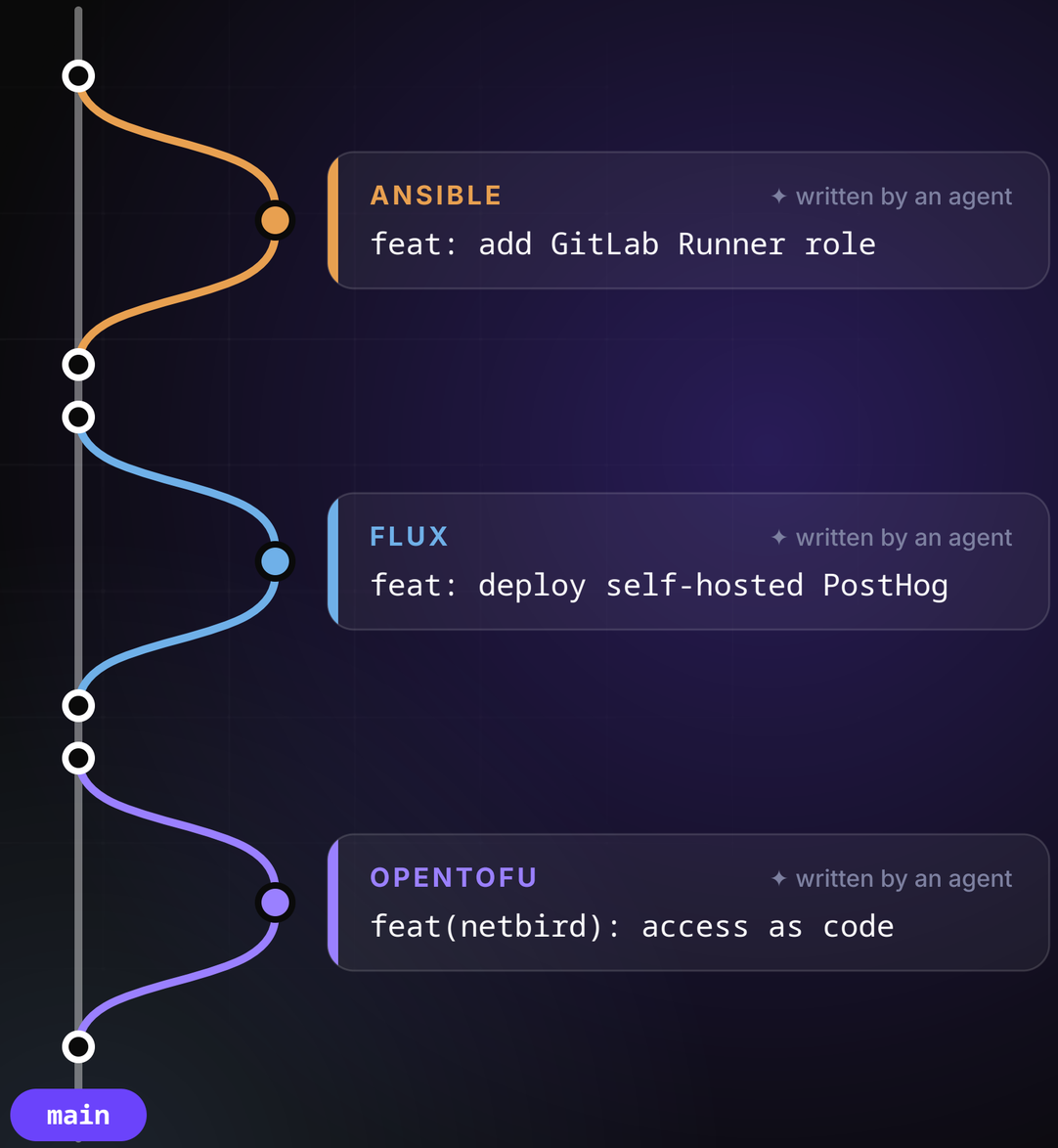
EU TECH MEETUP · DIGITALHUB AACHEN

GitOps, powered by AI

How anny moves six years of clicks into code



Adriaan Wind
Co-Founder & CTO at anny





ABOUT ME

Hi, I'm Adriaan

Co-Founder & CTO at **anny**, the booking platform for desks, rooms, events and library seats.

> **Building scalable software on Kubernetes.**

Kubernetes · DevOps · GitOps

linkedin.com/in/adriaan-wind

Who here has set up their cloud environment by clicking around in the dashboard?

PART 1

2020: ship the product, click the rest

PART 1 · WHERE THINGS LIVED

Some YAML in Git. The rest we clicked.

Apps on Kubernetes	YAML in Git, applied by hand with kubectl	HALF IN GIT
Cluster add-ons	Flux, but only on some clusters	IN GIT
Kubernetes clusters	Cloud dashboard	CLICKED
Databases	Cloud dashboard	CLICKED
Object storage	Cloud dashboard	CLICKED
DNS, CDN & firewall	Provider dashboards	CLICKED
Servers & CI runners	SSH, by hand	CLICKED
VPN & access	Admin UI	CLICKED

Where clicking started to hurt

Who changed this?

Something is different and nobody knows when, who or why.

No linting, no tests

Nothing catches a typo in a firewall rule before it's live. You find out when it breaks.

Can we rebuild it?

Only from memory, a few screenshots and whoever clicked it in the first place.

Every extra environment

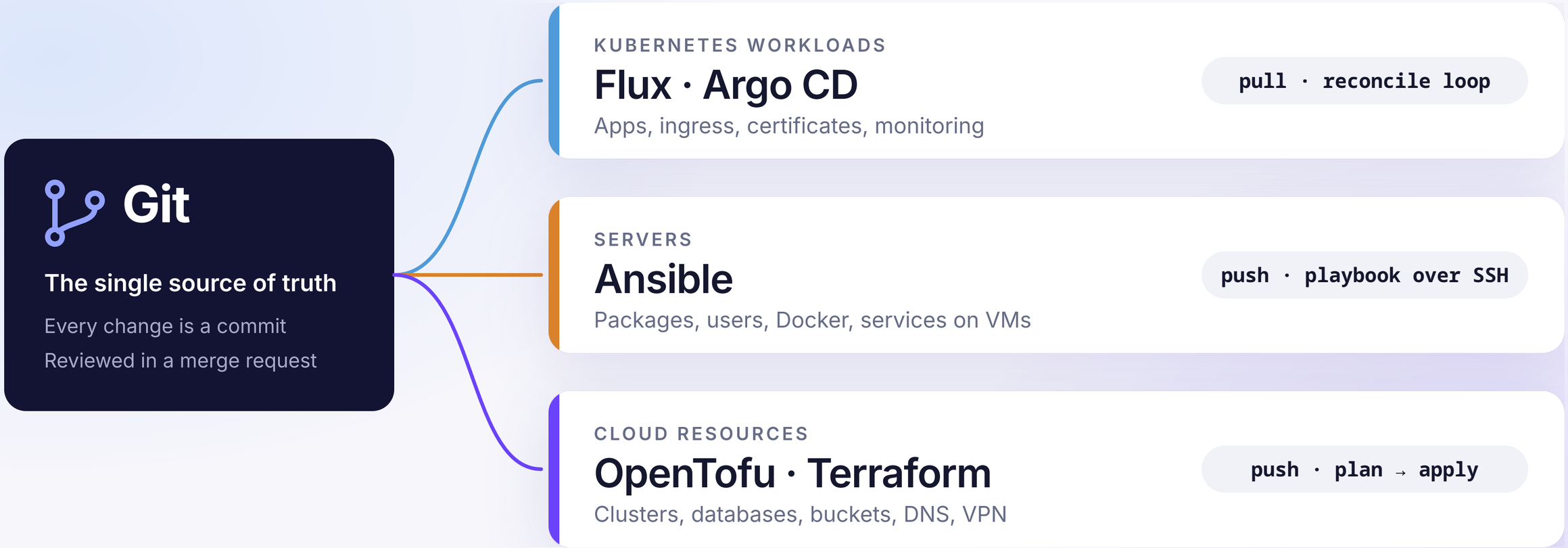
Staging, production, a second region. Each one clicked again by hand, and they slowly drift apart.

02

PART 2

GitOps in three pieces

What GitOps means for us



PART 2 · CLOUD RESOURCES

OpenTofu / Terraform

You describe your resources in HCL. `plan` shows you what would change, `apply` does it. OpenTofu is the open-source fork of Terraform, started after the 2023 licence change.

At anny

- Kubernetes clusters and databases
- DNS and CDN
- Monitoring and alerting
- Access control and VPN

anny-services/cluster.tf

```
resource "scaleway_k8s_cluster" "anny_services" {
  name    = "anny-services"
  type    = "kapsule"
  version = "1.36"
  cni     = "cilium"
  region  = "nl-ams"

  auto_upgrade {
    enable                = true
    maintenance_window_day = "sunday"
  }
}
```

PART 2 · SERVERS

Ansible

Ansible logs into your machines over SSH and sets them up: packages, users, Docker, services. Nothing to install on the server, and a second run doesn't change anything.

At anny, since February 2026

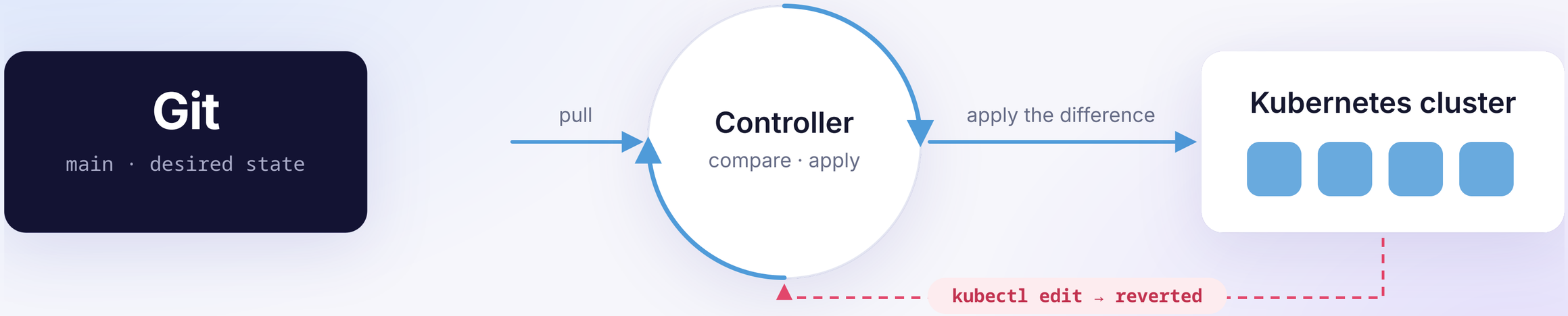
- Base setup and hardening for every server
- CI runners
- Monitoring

ansible/site.yml

```yaml
- name: Configure all servers
  hosts: all
  roles:
    - ssh
    - base
    - docker
    - beszel-agent

- name: Deploy CI Runner servers
  hosts: ci_runner
  roles:
    - gitlab-runner
```

Flux / Argo CD: the cluster pulls from Git



Flux or Argo CD?

	Flux WHAT WE USE	Argo CD
HOW YOU USE IT	CLI and YAML, no built-in UI	Web UI first, with a CLI too
BUILDING BLOCKS	Small controllers you combine	One app server with Applications
HELM CHARTS	Real Helm releases, helm list works	Renders the chart, applies plain YAML
MANY CLUSTERS	Usually one Flux per cluster	One Argo CD can run many clusters
ACCESS CONTROL	Plain Kubernetes RBAC	Own RBAC, SSO and projects
IMAGE UPDATES	Built in, writes new tags to Git	Separate add-on
PICK IT IF	**Your team lives in the terminal, and so do your agents**	**People want to see and click the sync state**

PART 3

Worth it, but a lot of work

PART 3 · WHY IT'S WORTH IT

Why we wanted this anyway

Maintainability

Shared modules instead of copy-paste. Fix it once and every cluster using it gets the fix.

Review before change

Someone looks at every change before it hits production, not after the incident.

Compliance & audit trail

Who changed what, when and why. That's what auditors ask for, and Git just has it.

Reproducible

Rebuild a cluster or a whole region from code instead of from memory.

No drift

Hand edits get reverted. What's in Git is what's running.

Documentation for free

The repo is the docs, and the comments say why.

PART 3 · THE PRICE, BEFORE AI

Two minutes in the dashboard, half a day in code

IN THE DASHBOARD ≈ 2 min

1. Open the dashboard
2. Fill in the form
3. Click "Create"

AS CODE ≈ half a day

1. Find the provider docs
2. Learn the resource schema
3. Write the HCL
4. Import what already exists
5. Plan, fix the diff, plan again
6. Merge request, wait for review
7. Apply

Rough numbers for one new resource, from my own experience. Not a benchmark.

PART 3 · THE WORST PART

Importing what you already clicked

```
$ terraform import scaleway_k8s_cluster.services nl-ams/1f3c…
Error: resource address "scaleway_k8s_cluster.services"
does not exist in the configuration.

# write the block by hand, guess every argument, retry
$ terraform import scaleway_k8s_cluster.services nl-ams/1f3c…
Import successful!

$ terraform plan
  # scaleway_k8s_cluster.services must be replaced
      ~ cni = "cilium" -> "calico" # forces replacement
Plan: 1 to add, 0 to change, 1 to destroy.

# …and 8 more resources to go
```

1. **Find the ID**
   Every provider has its own format: region/uuid, zone/record, projects/.../clusters/...
2. **Write it blind**
   import only fills the state. You guess the config, one argument at a time.
3. **One resource at a time**
   Our services cluster alone: 9 resources, 9 IDs, 9 imports.
4. **Chase the diff**
   Copy values from state show into HCL until plan finally stops complaining.
5. **One wrong value**
   forces replacement, and suddenly plan wants to destroy production.

Since Terraform / OpenTofu 1.5: `import {}` blocks and `plan -generate-config-out`. Better, but you get every attribute, hard-coded IDs and no references. Still a lot of cleanup.

May 2022: we tried to import our Cloudflare DNS.

```
$ cd infra/terraform/cloudflare
$ ls
cf-terraforming main.tf booking_buddy_de.tf joboerse_ac.tf …
$ git log
fatal: your current branch 'main' does not have any commits yet
```

So for years, the dashboard won.

PART 4

Letting AI write the code

Agents are much better at writing code than at clicking through UIs.

PART 4 · WHY CODE

Why agents are good at this

It's all text

Models read and write text. A web dashboard is pixels and mouse clicks, built for human eyes.

I can review it

I read what the agent wrote before anything happens. A click is just gone.

It's repeatable

The same file builds the next cluster and the next region. A clicking agent starts from zero every time.

plan checks it

validate, plan and CI catch mistakes before production. Nobody checks a click.

What the agent actually does for us

01

Write

OpenTofu, Ansible roles, HelmReleases. In our conventions, with comments that explain why.

02

Read

Looks at what's live with CLIs like scw and kubectl, and spots what drifted.

03

Check

Runs validate and plan, explains the diff, and fixes whatever plan complains about.

PART 4 · CLICK-OPS TO GIT

One component at a time

1

Pick something

Something we clicked: a cluster, a DNS zone, a VPN policy.

2

Look it up

The agent checks it with the provider's CLI or API.

3

Write the code

Resource blocks plus import blocks, the way we write them.

4

Run plan

Again and again, until plan has nothing left to change.

5

Review & merge

Someone reviews the merge request. From now on it lives in Git.

NEXT COMPONENT

```
$ tofu plan
No changes. Your infrastructure matches the configuration.
```

PART 4 · IMPORT IN PRACTICE

What adopting a clicked resource looks like

Something already exists because someone clicked it. The code gets two parts: a resource block that describes it, and an import block that points at the real thing.

```
$ tofu plan
Plan: 1 to import, 0 to add, 0 to change, 0 to destroy.
```

Nothing gets created or destroyed. After the apply, the import block can go. Same pattern for clusters, databases or buckets.

dns.tf

```
resource "cloudflare_dns_record" "app" {
  zone_id = var.zone_id
  name    = "app.example.com"
  type    = "A"
  content = "203.0.113.10"
  proxied = true
}

# Created by hand in the dashboard.
# Adopt it instead of creating it again.
import {
  to = cloudflare_dns_record.app
  id = "<zone-id>/<record-id>"
}
```

PART 4 · GETTING CONTEXT

Give the agent your CLIs

MCP server

- − Every tool definition sits in the context, whether you use it or not
- − You get whole objects back and filter after they are already in the context
- − A schema the model has never seen before
- − Yet another server to run, update and give a token

CLI OUR DEFAULT

- + The model already knows it from years of docs and examples
- + Pipes into jq, grep or head, so only the lines that matter reach the model
- + --help is the manual, -o json is basically the API
- + Runs with my login and my permissions, and I can rerun the same command myself

```
$ kubectl get pods -A -o json | jq -r '.items[] | select(.status.phase != "Running") | .metadata.name'
# only a few lines reach the model, not the JSON of every pod in the cluster
```

PART 4 · GETTING CONTEXT

Example: a volume someone grew by hand

SATURDAY EVENING

Database volume grown 60 → 70 GB by hand in the dashboard

SINCE THEN

Every plan wants to shrink it back. Scaleway volumes only grow: apply fails

SUNDAY

The agent reads the live size with scw and writes it into the code

AGENT SESSION

```
$ scw rdb instance get <id> region=nl-ams -o json \
    | jq '.volume.size'
70000000000

$ ./tofu.sh plan
  ~ volume_size_in_gb = 70 -> 60 # a shrink
```

COMMIT · ANNY-SERVICES

fix(psql): record the volume's out-of-band growth to 70 GB

```
- volume_size_in_gb = 60
+ # Grown 60 -> 70 by hand on 2026-08-29 (manual
+ # backup taken first). Scaleway volumes only grow.
+ volume_size_in_gb = 70
```

"This records reality; applying it changes nothing live."

✦ Co-Authored-By: Claude

PART 4 · GETTING CONTEXT

Where MCP still makes sense

No CLI exists

Lots of SaaS tools only have an API. Then an MCP server beats the agent writing curl calls.

SaaS behind OAuth

Tickets, error tracking, docs. Log in once with scoped access, no token lying around in a dotfile.

No shell

Chat apps and sandboxed agents can't run a CLI. For them MCP is the only way in.

A hard allowlist

When the agent should be able to call exactly these five actions and nothing else.

Our do’s and don’ts

Do

- Everything goes through a merge request, the agent’s changes too
- The agent can plan. Applying to production is a human decision
- Keep secrets in a secret manager, never in the state, the repo or the prompt
- Write your conventions down: AGENTS.md, CLAUDE.md, skills
- One component per merge request

Don’t

- ✕ Give the agent production apply rights
- ✕ Paste tokens into the chat
- ✕ Approve a plan you haven’t read. Search it for `destroy`
- ✕ Import fifty resources at once
- ✕ Trust argument names the model remembers. `validate` and `plan` decide

05

PART 5

One component after another

PART 5 · ANNY TODAY

How we got into Git

2020 – 2025

4 STEPS IN 5 YEARS · BY HAND

- Sep 2020
  Kubernetes manifests in Git
- Sep 2021
  Flux for cluster add-ons
- Jan 2023
  First OpenTofu: CDN, DNS, status page
- Feb 2025
  Edge rules and alerting as code

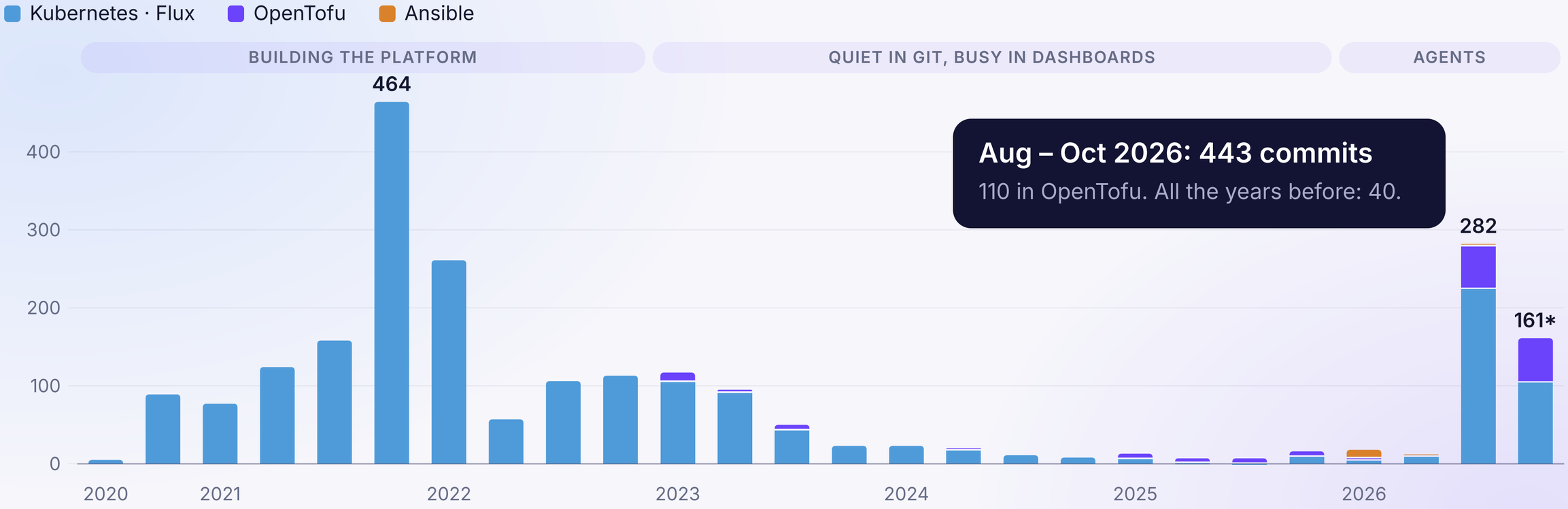
2026

5 STEPS IN 8 MONTHS · WRITTEN BY AGENTS

- Feb 2026
  Every server configured with Ansible
- Feb 2026
  Services cluster imported into OpenTofu
- Apr 2026
  A Flux platform repo for the newer clusters
- Sep 2026
  VPN access as code: groups, people, policies
- Sep 2026
  LLM inference box, provisioned as code

PART 5 · ANNY TODAY

What happened in our infra repos



Commits per quarter across anny's infra repos, human authors only. * Q4 2026 until 25 October. Since August a dedicated platform engineer works on these repos too.

PART 5 · ANNY TODAY

The last three months

43
infra commits in all of 2025

443
infra commits, Aug – Oct 2026

1 in 2
with a Claude co-author trailer

GitOps was always a good idea. AI just made it cheap enough that we actually do it.

WRAPPING UP

Key takeaways

01

Import, don't rebuild

Bring in what you clicked. When plan says no changes, that part is done.

02

The agent writes, we merge

Everything goes through a merge request. No production credentials for the agent.

03

Start with CLIs

Give the agent the CLIs you already use. Add MCP where there is no CLI.

04

Write your rules down

AGENTS.md, CLAUDE.md, skills. That's how the agent learns your setup.



Adriaan Wind

Co-Founder & CTO · anny.co

linkedin.com/in/adriaan-wind

THE SLIDES

devtig.dev/talks