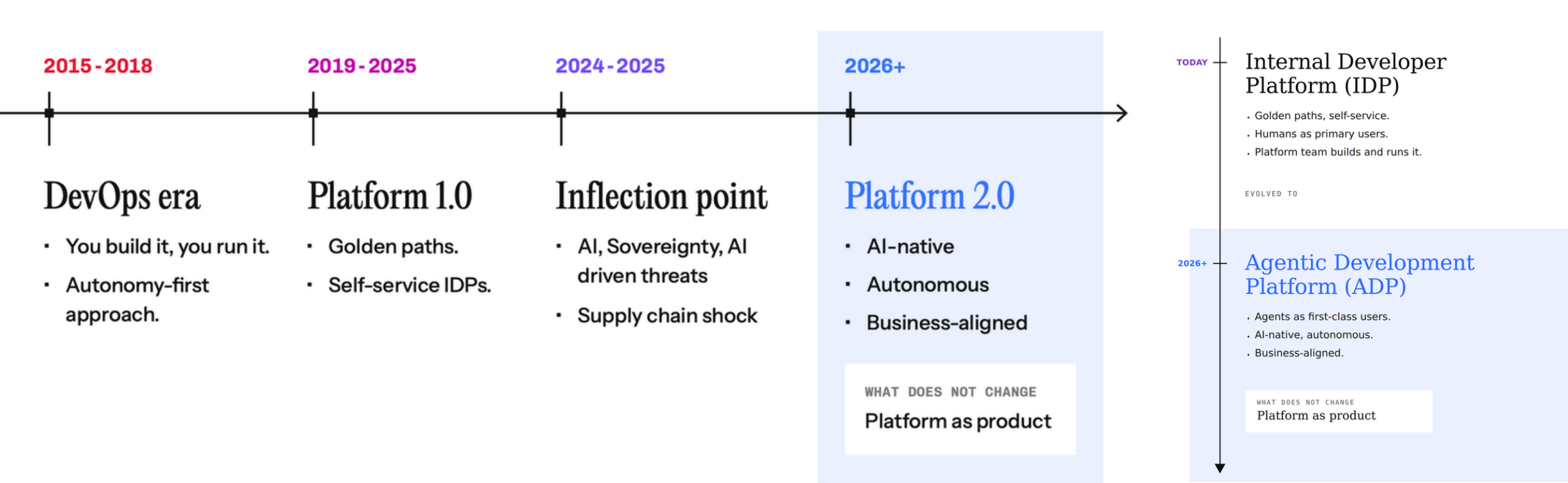
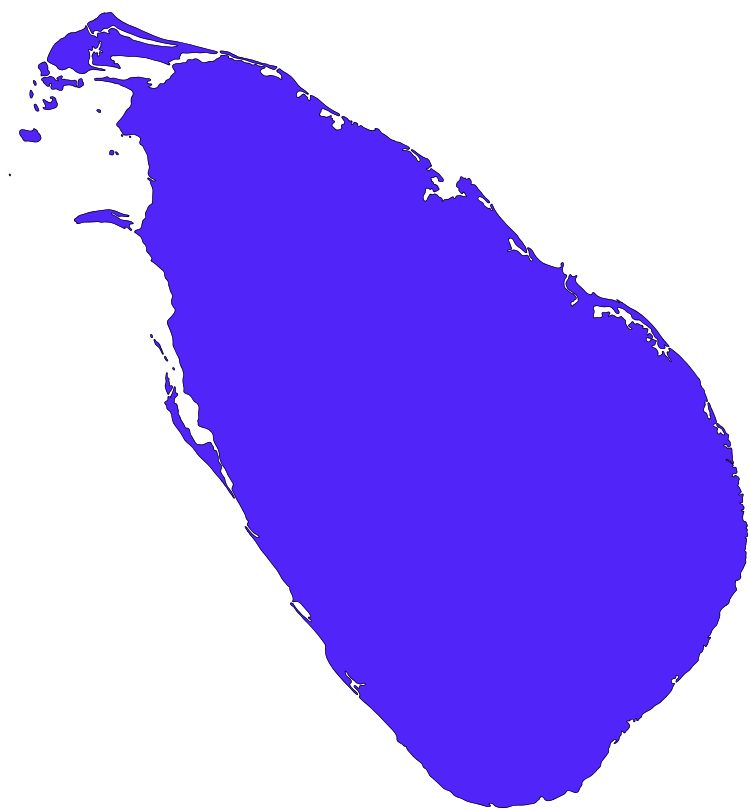




GitOps, Gitless, Agentic: The Evolution of Platform Engineering





Artem Lajko

Head of Platform Engineering

- ✓ CNCF Kubestronaut
- ✓ Ambassador
- ✓ Published Author



Mission



To understand the future, first we need understand the past.



To see what is possible today.



And to understand:
How Agentic fits
in.



How it all began.



In **2017**,
Alexis Richardson
gave it a name:



GitOps!



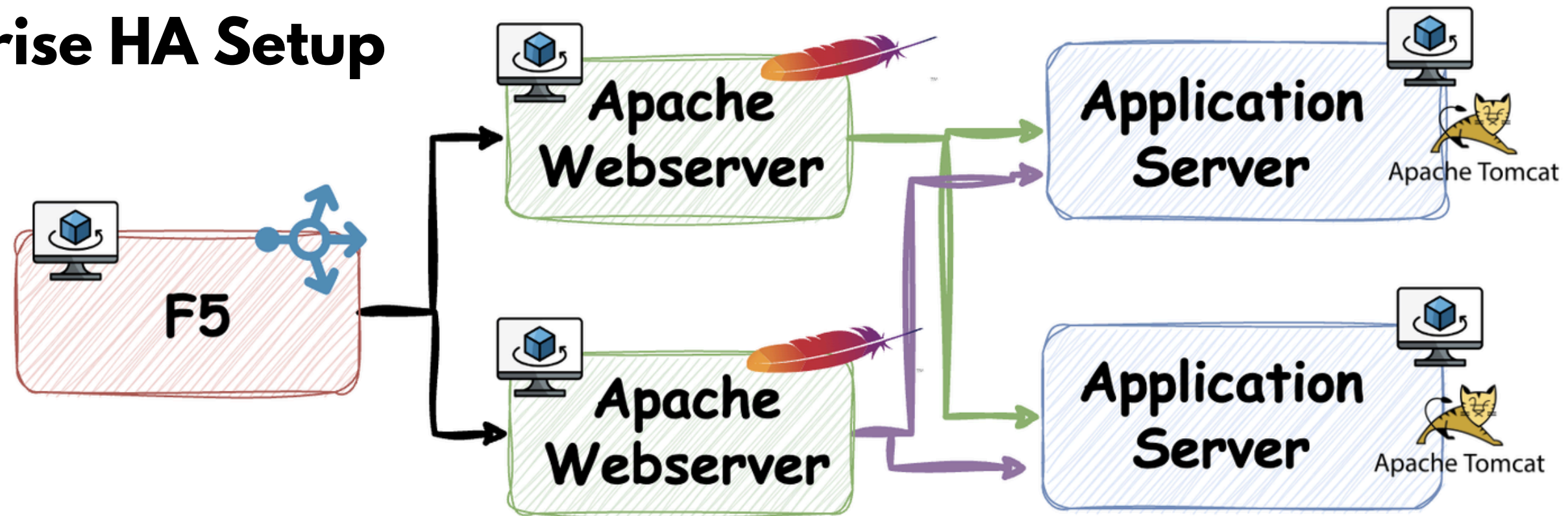
Alexis Richardson

CEO, ConfigHub 

What problem did we want to solve in 2017?

Classic enterprise HA was reliable, but were complex, manual, and fragile.

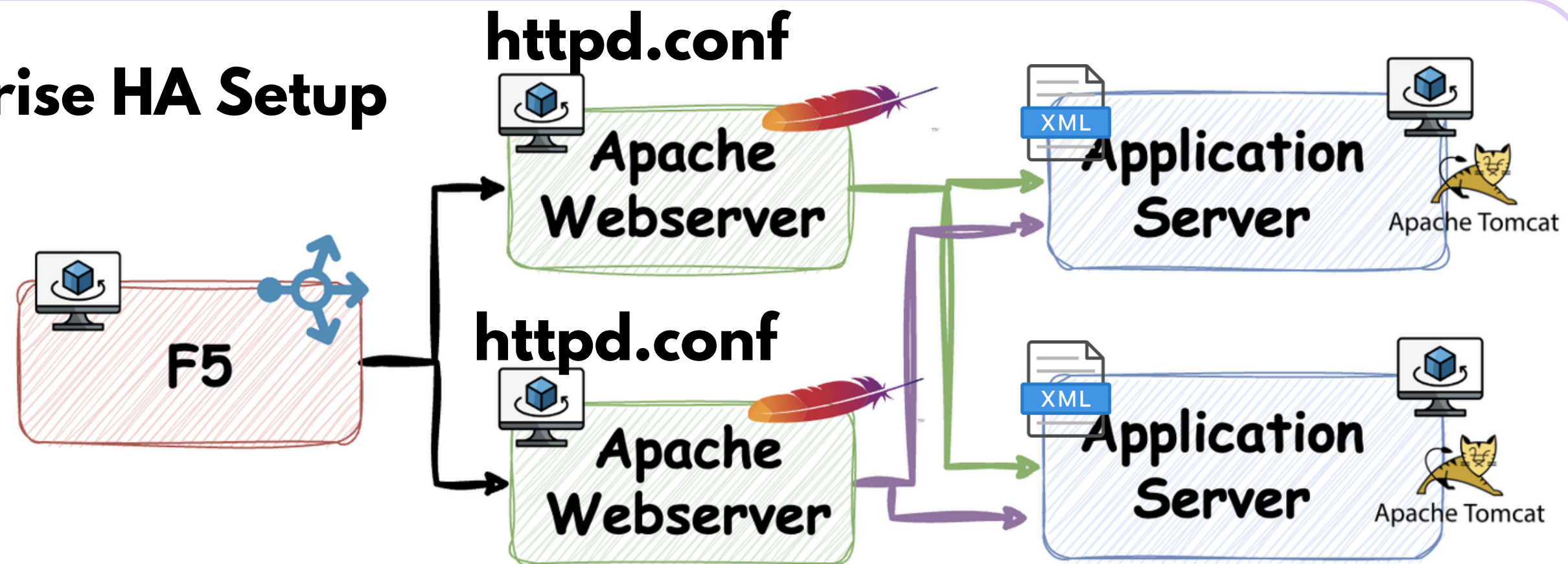
Enterprise HA Setup



Jumping from VM to VM via RDP or SSH.

Manual server-by-server changes caused drift, slow releases, and painful rollbacks.

Enterprise HA Setup



Doesn't scale!

And this led to Infrastructure as Code...



ANSIBLE



puppet



CHEF



HashiCorp

Terraform



SALTSTACK

...and everything becomes **as Code.**



ANSIBLE



puppet



CHEF



HashiCorp

Terraform



SALTSTACK

Infrastructure as Code



Network as Code



and so - - - -

Policy as Code



Configuration as Code



New tooling. Same challenge.

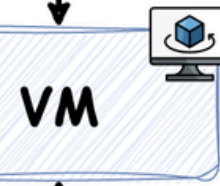
Still running changes from somebody's machine.

System Engineer:
Alex



1
execute changes on

LocalOps

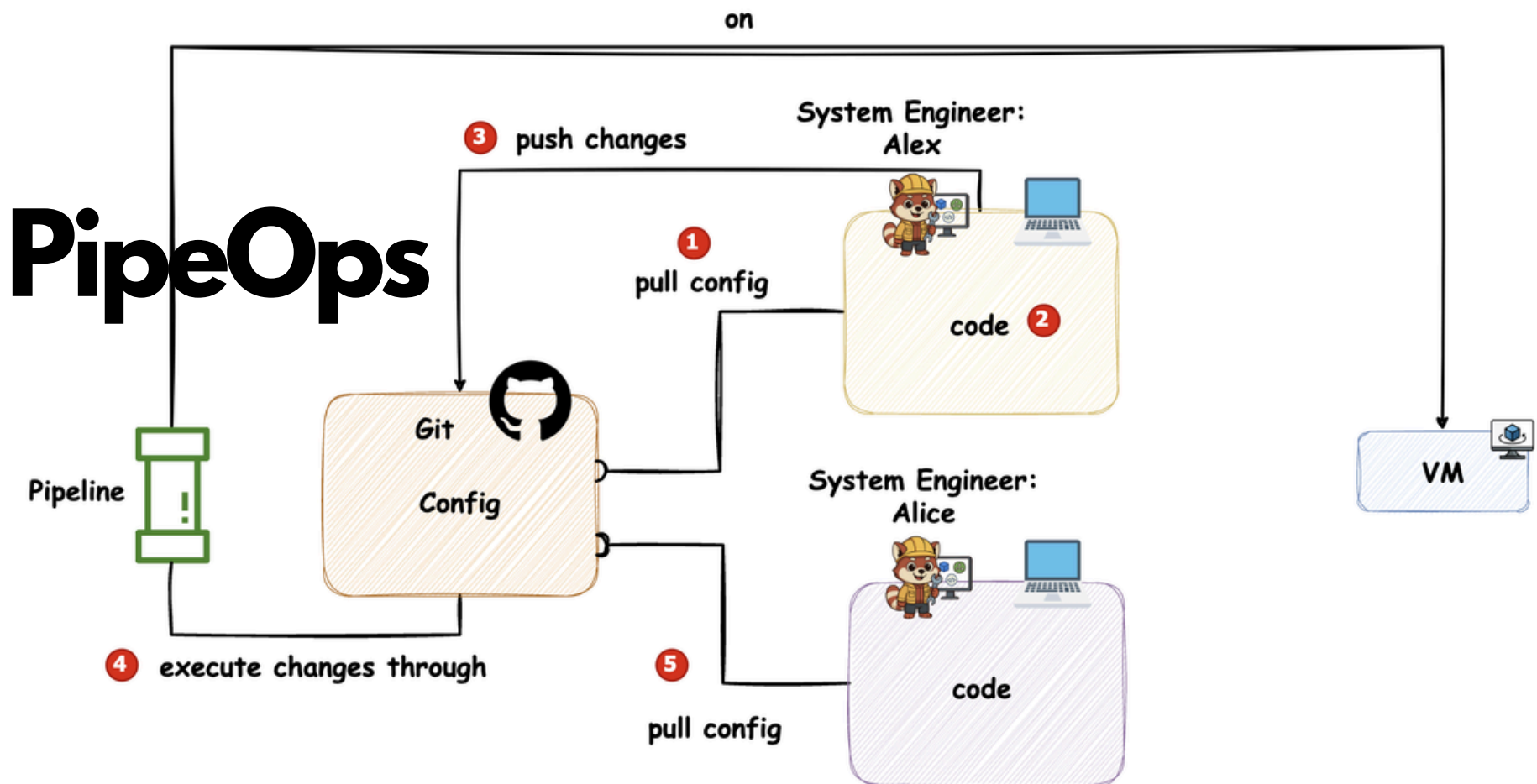


System Engineer:
Alice



2
execute changes on

PipeOps



Tools **doesn't** change Mindset.

From LocalOps, PipeOps to an Agent.

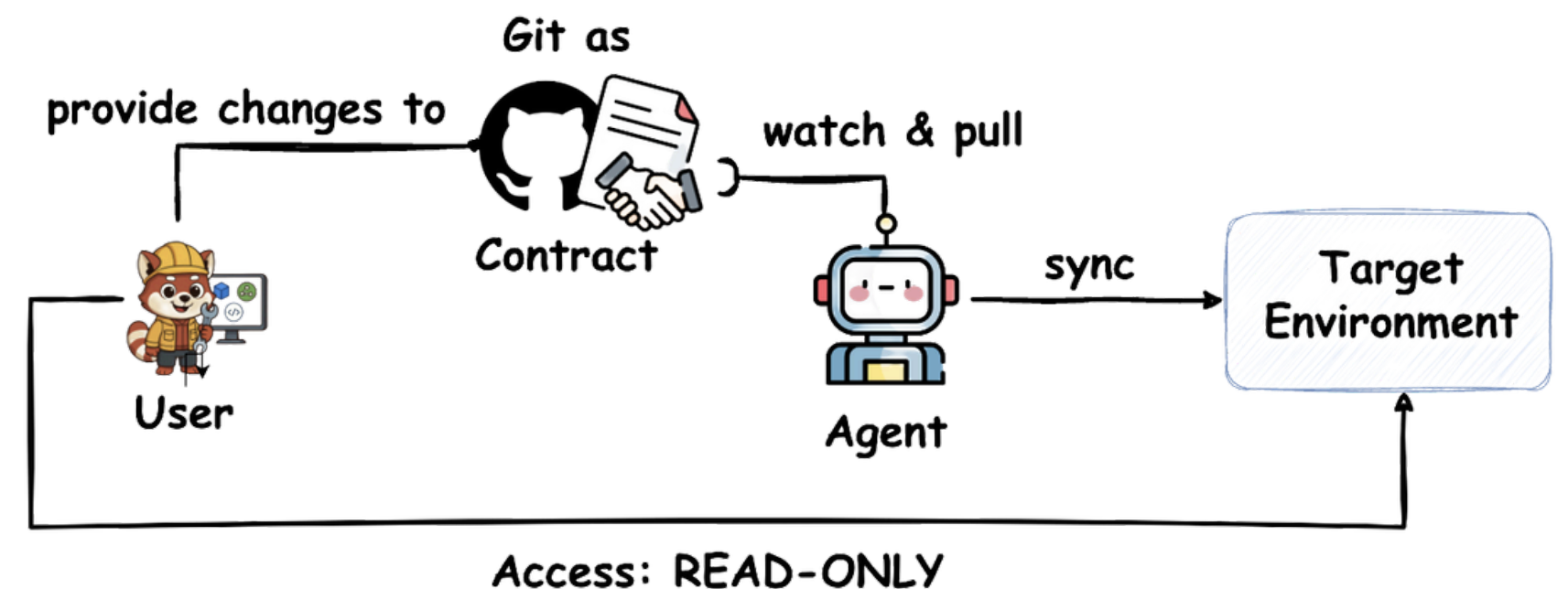
The user declares intent in Git. The agent keeps the target environment in sync.

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 intent in Git with the actual state of the target environment.

Agent keeps the environment in sync

Git is the contract. The agent watches, pulls, and syncs.



Basic Principles

Before we can evolve GitOps,
we need to agree on a few basic principles.



The 4 Core Principles of OpenGitOps.

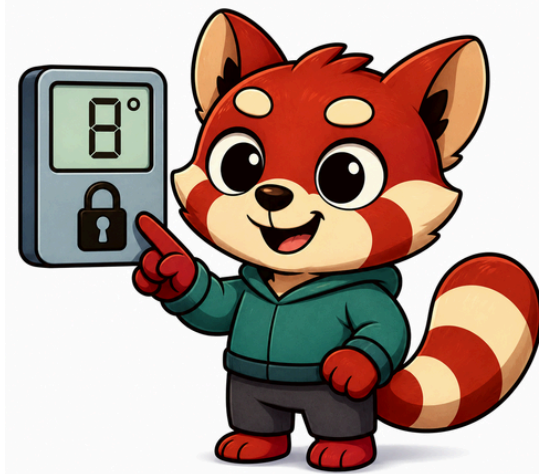


Declarative. Versioned. Pull-based. Continuously reconciled.

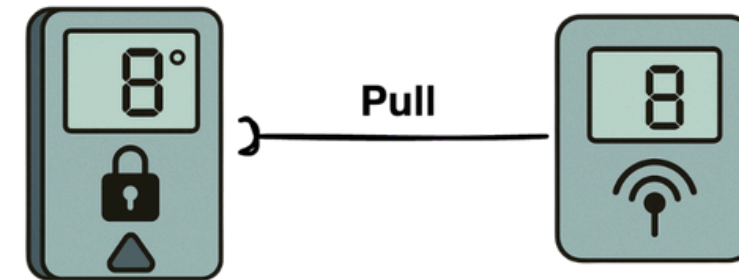
Declarative



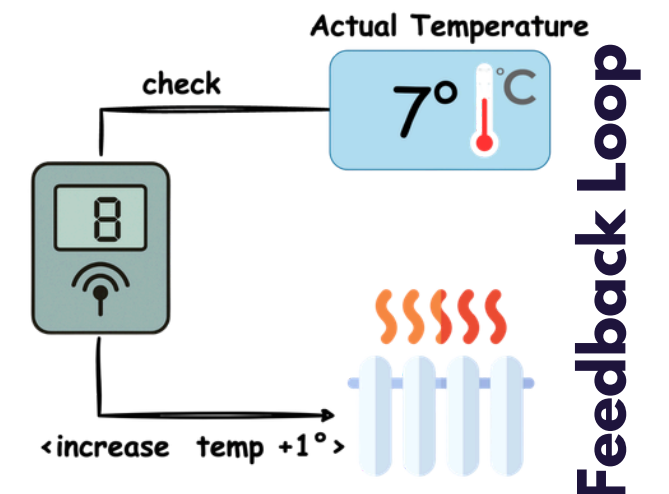
Versioned and Immutable



Pull-Based



Continuously Reconciled



These four constraints make GitOps predictable.



The use of GitOps allows us to:



Not just to manage a **single** ship...

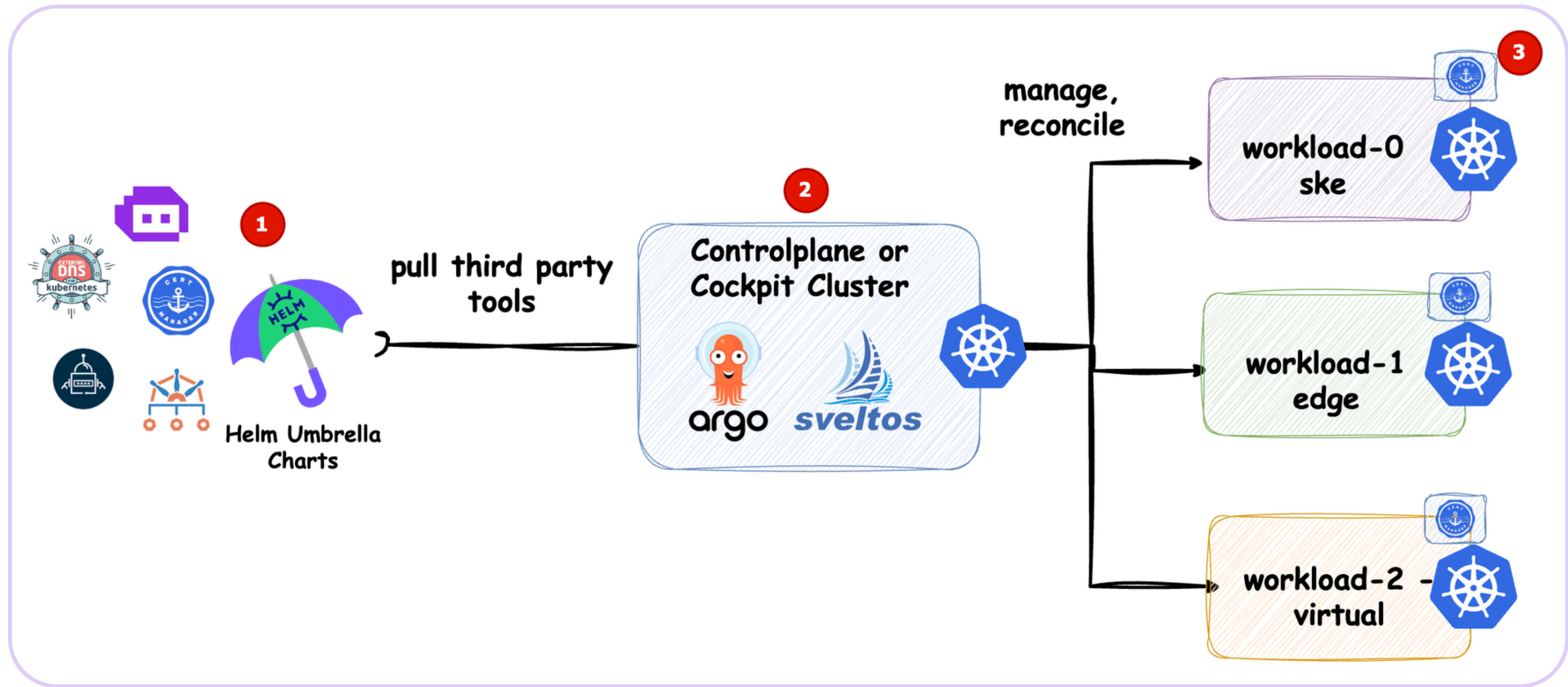


but a whole **fleet**.



By Creating a catalog...

Pull third-party tools into a control plane and reconcile workloads across clusters.

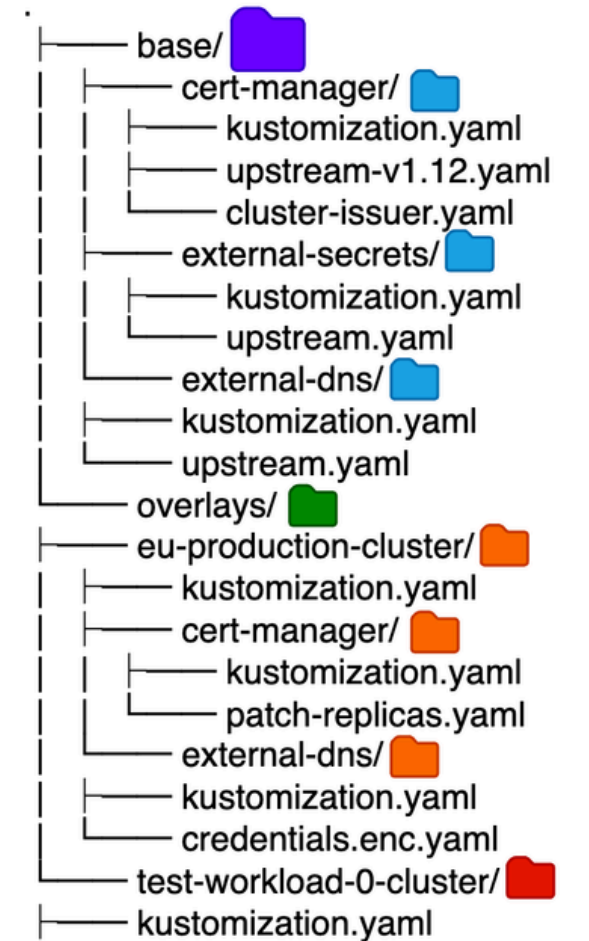
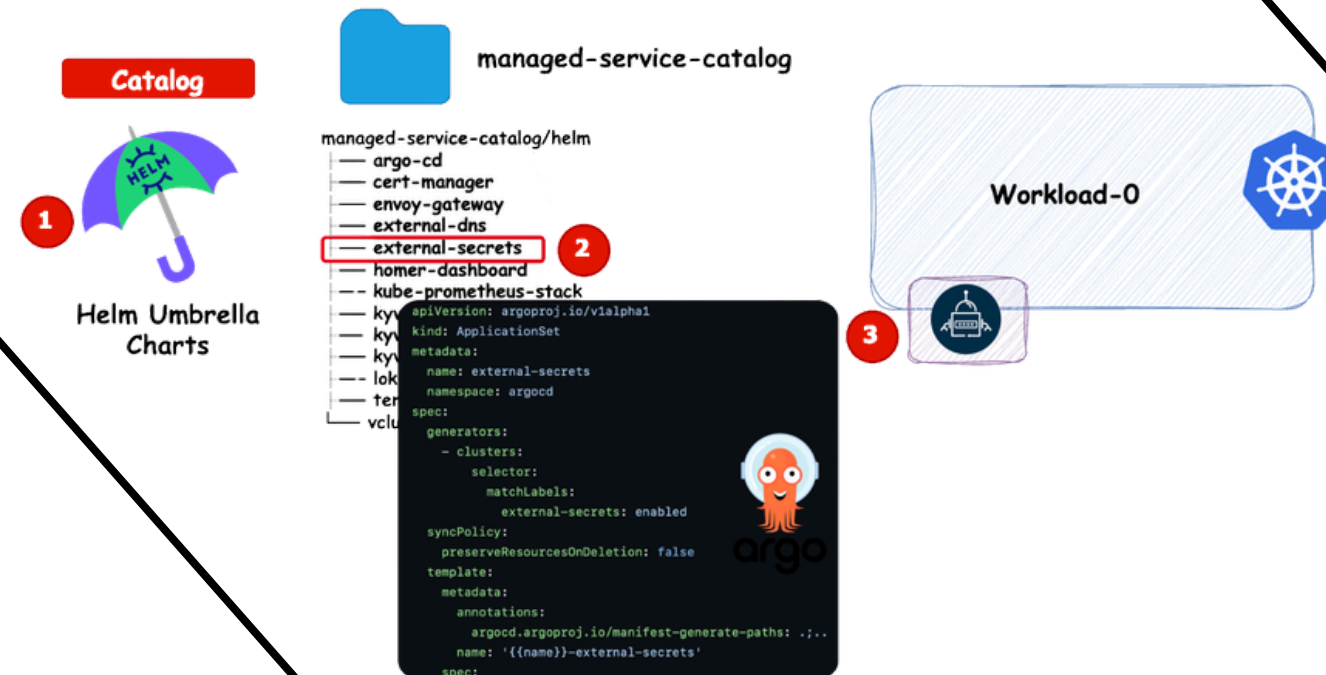
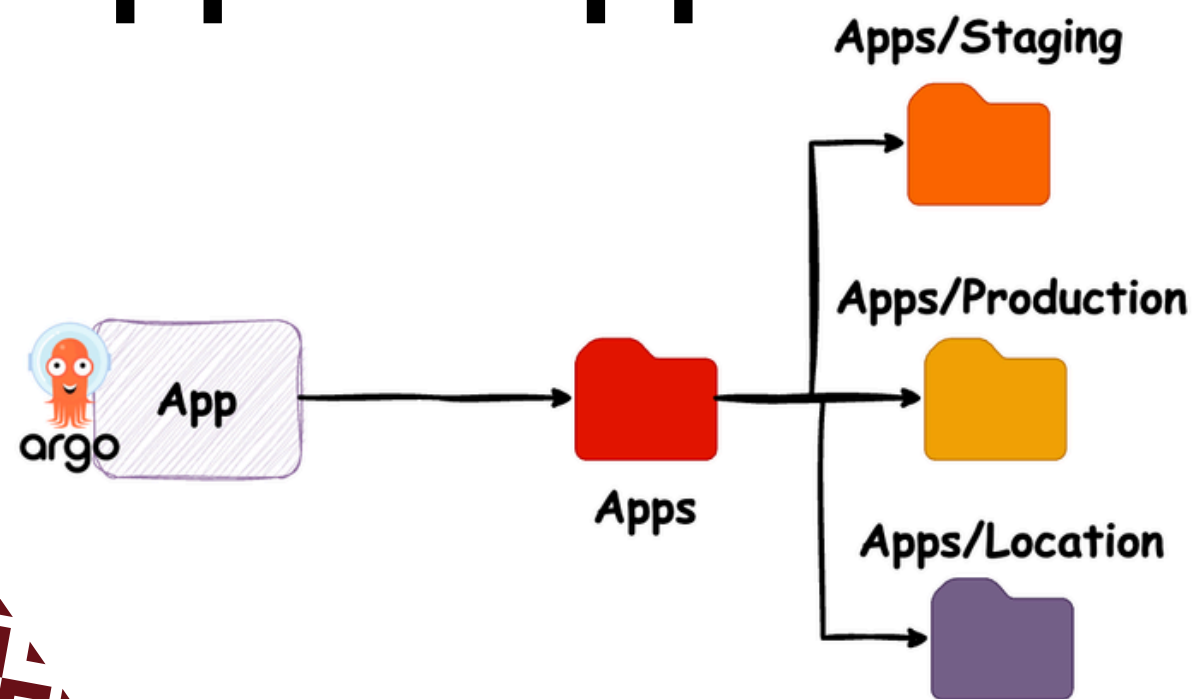


...and combining different approaches.

ApplicationSets + Clustergenerator

Overlays

App of Apps

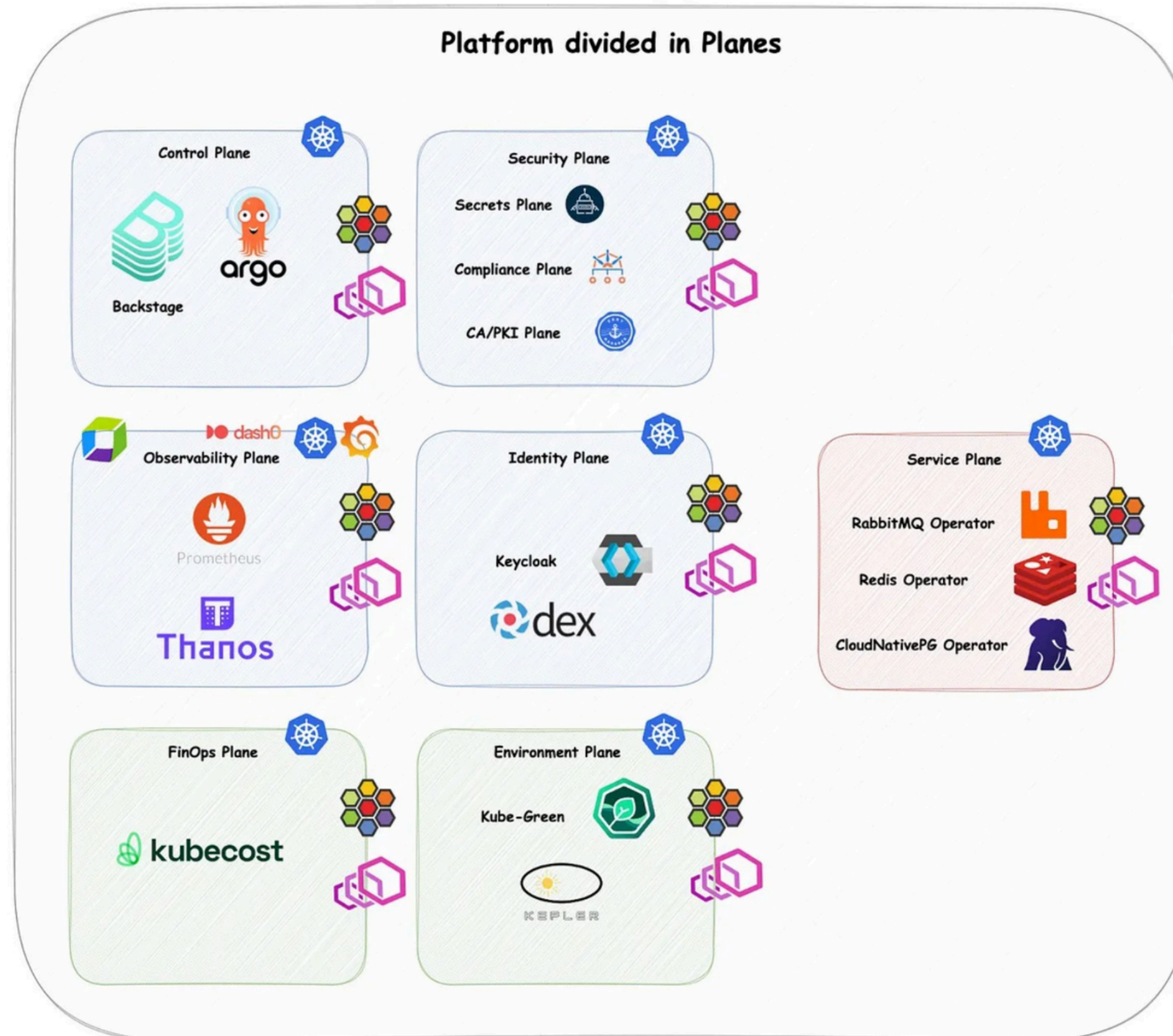


 **ustomize**

...and

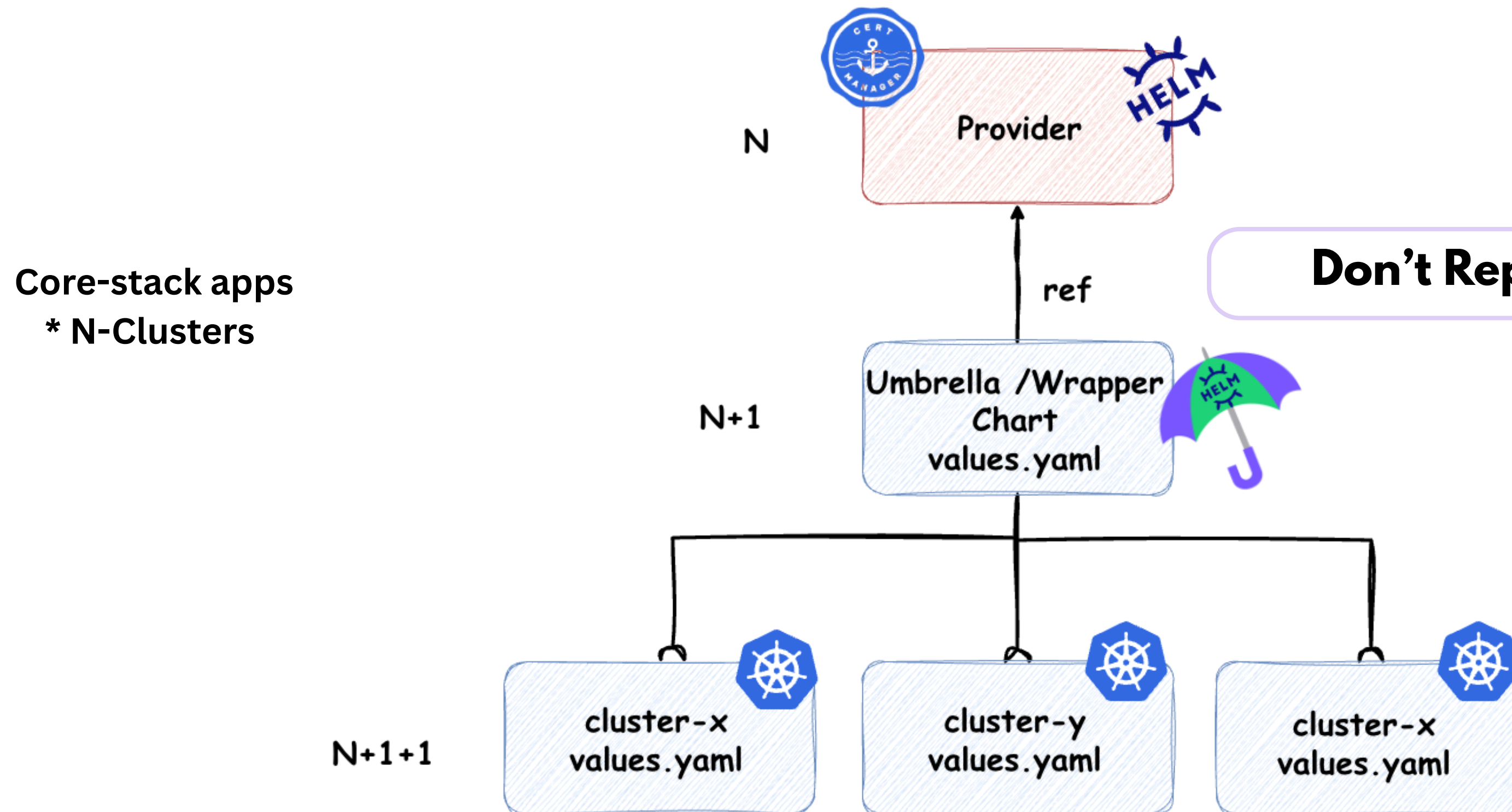
Thinking in Planes.

If you want real scale, cut the platform into planes.

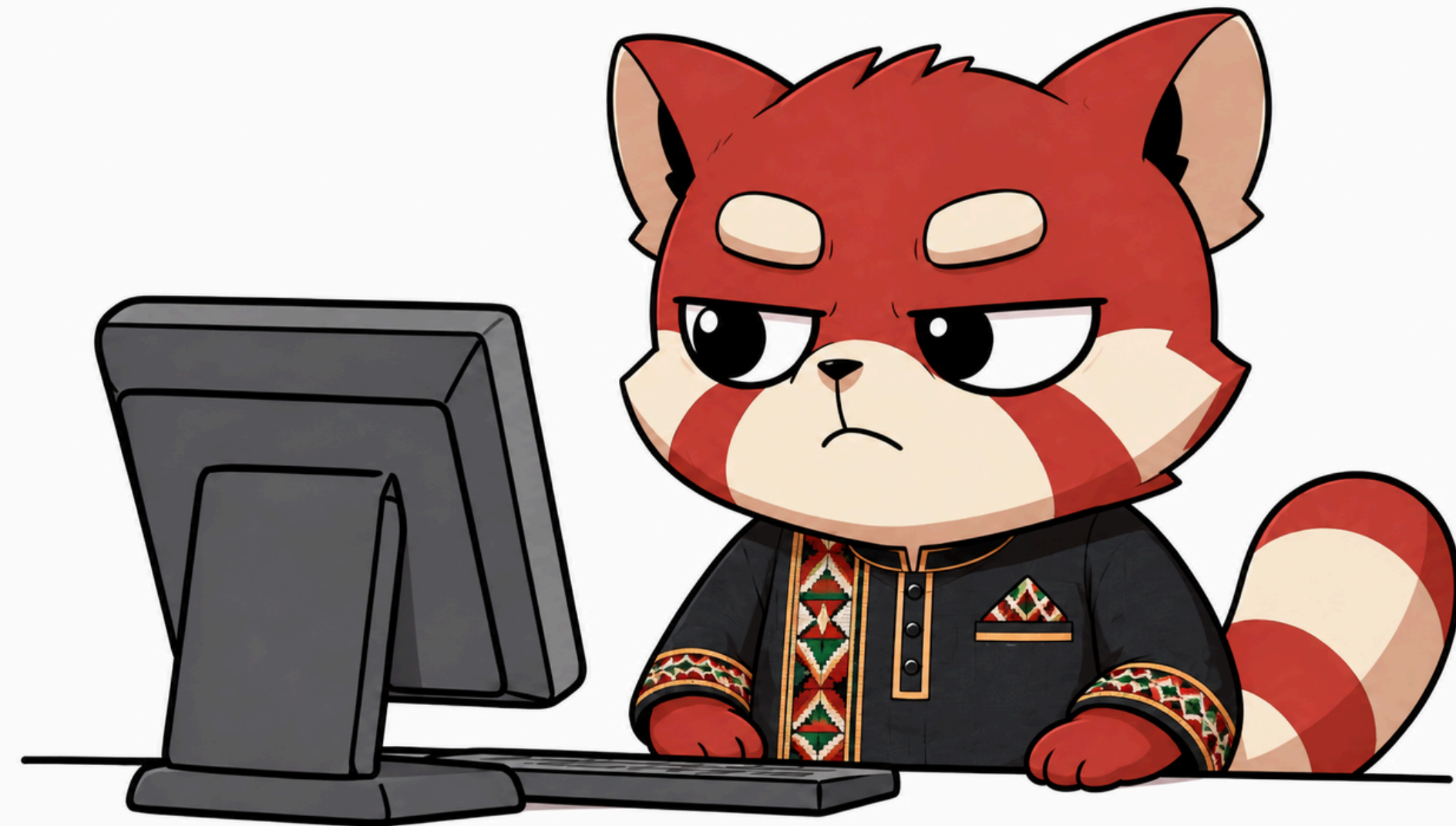


Different approaches. Same idea.

Avoid repeating the same configuration across every cluster.

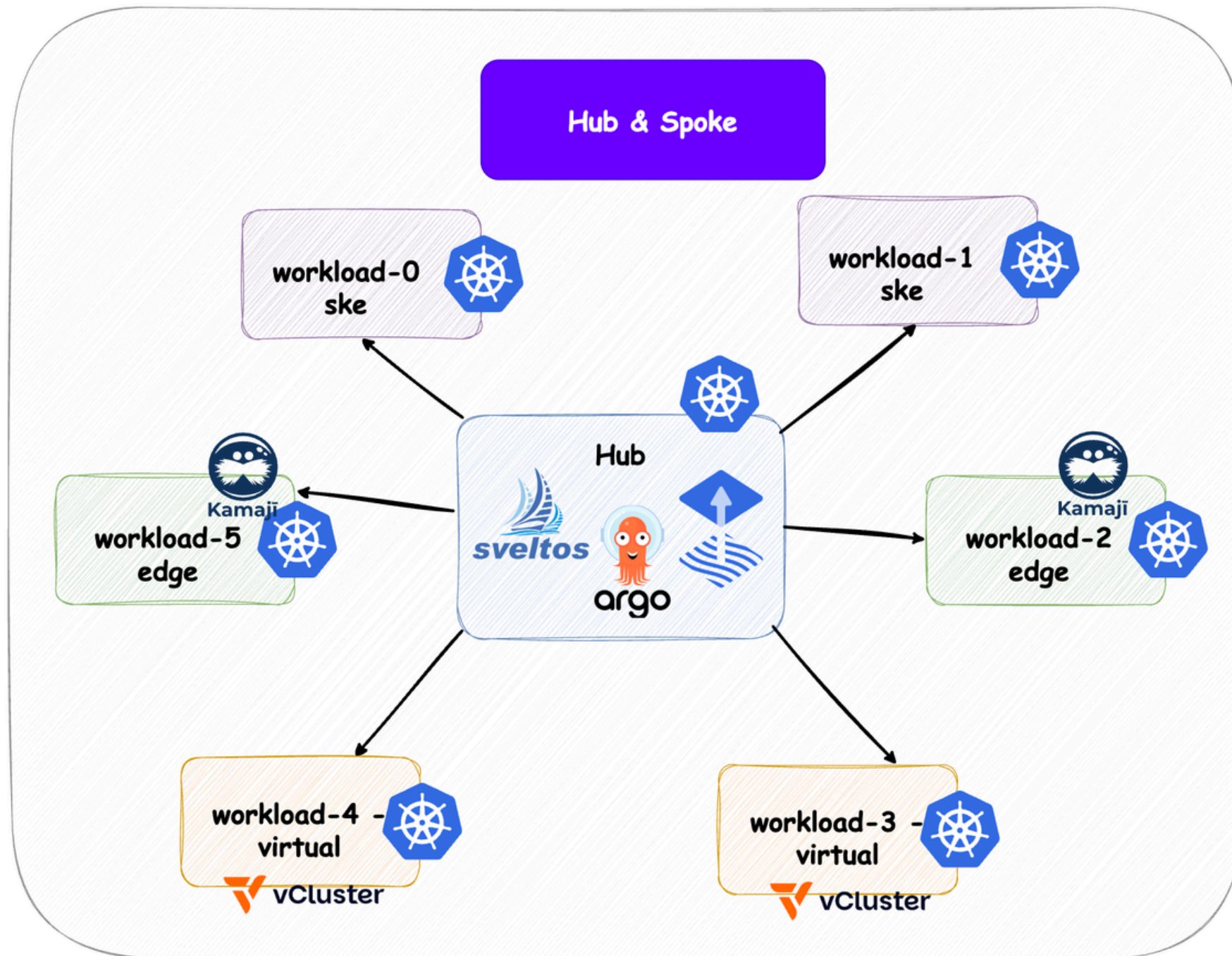


What is the best-fitting architecture?



We implement different architectures.

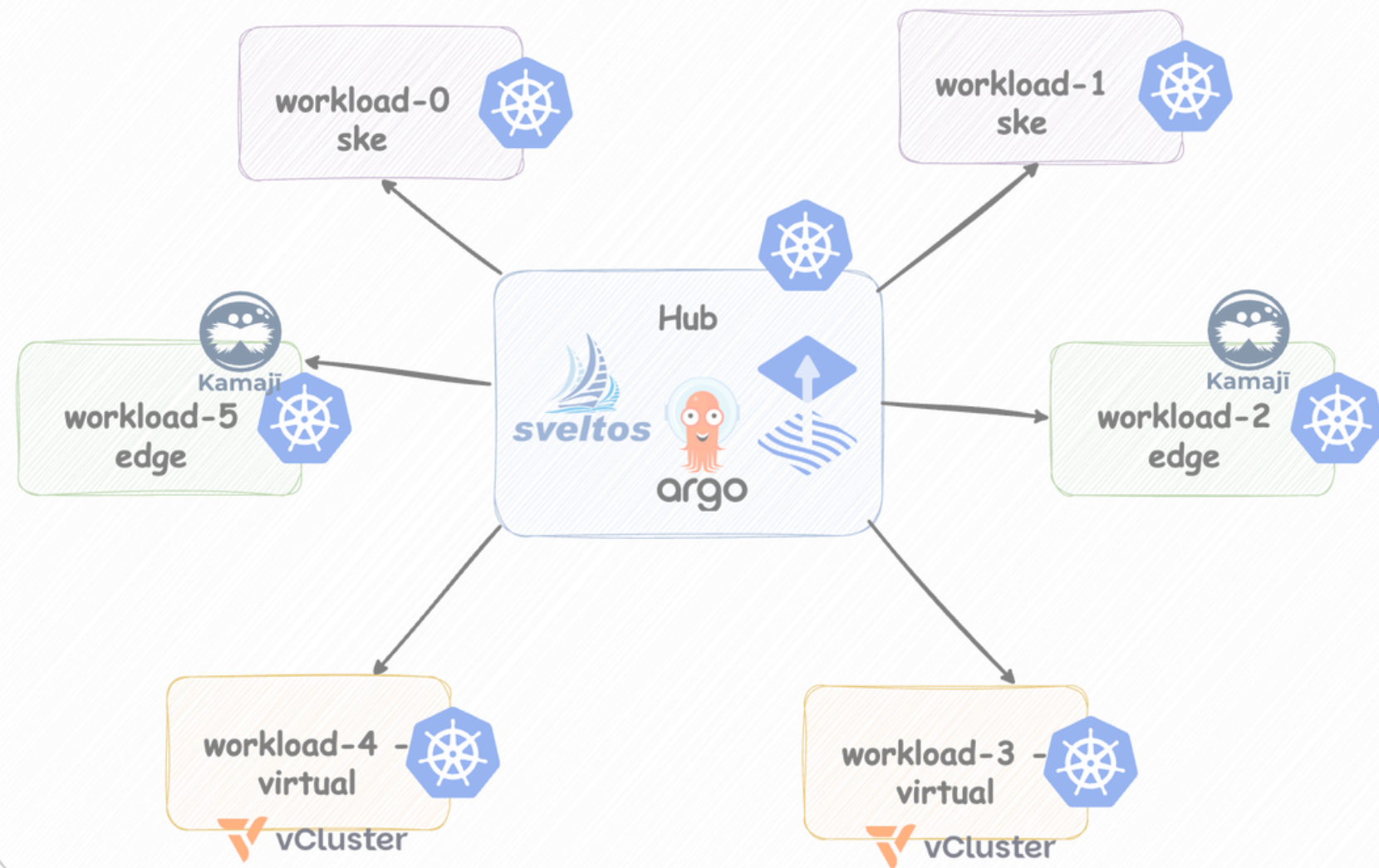
Example: Hub & Spoke



Or: dedicated instance per cluster.

Same target model - different operational trade-off.

Hub & Spoke

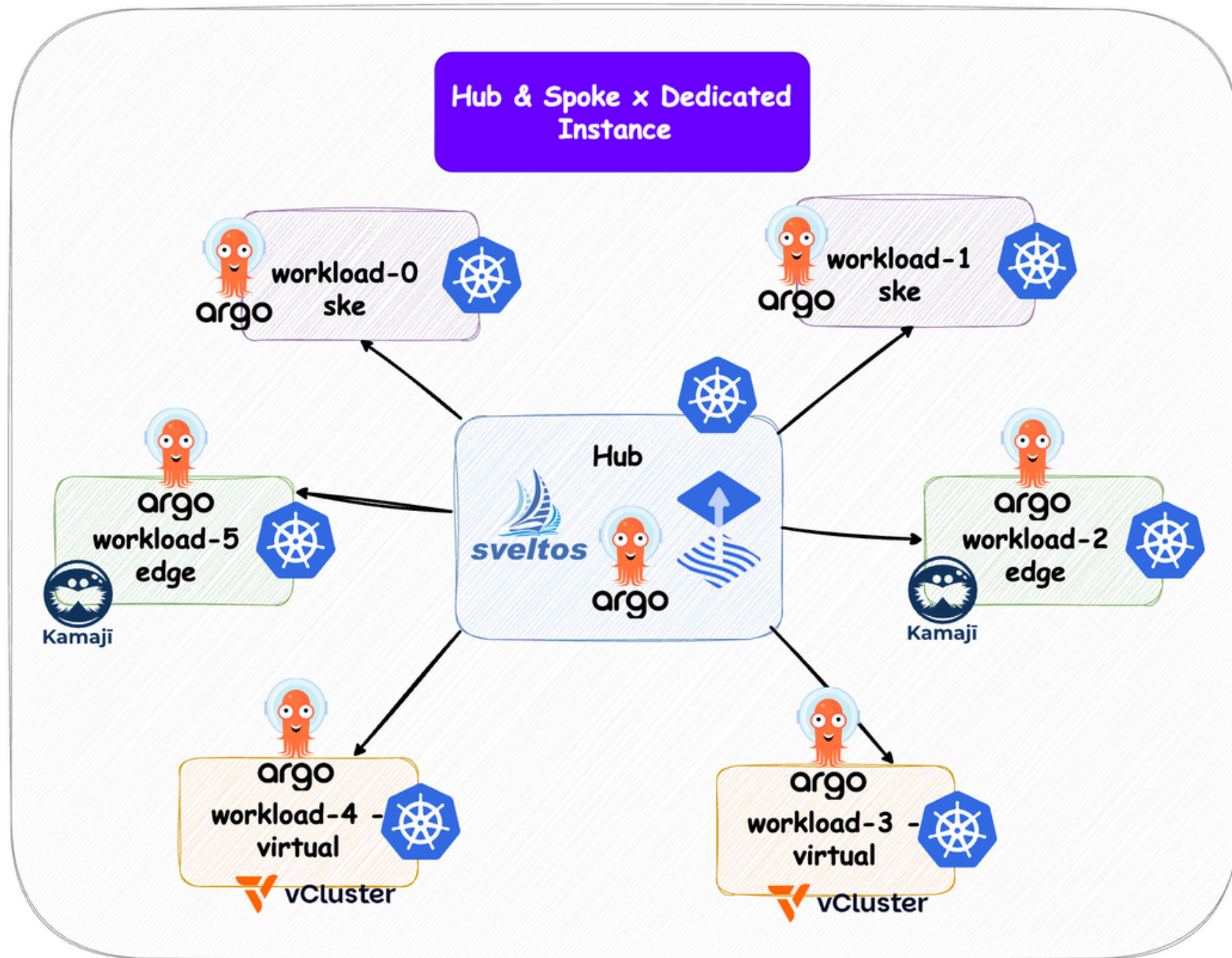


Dedicated Instance or Standalone



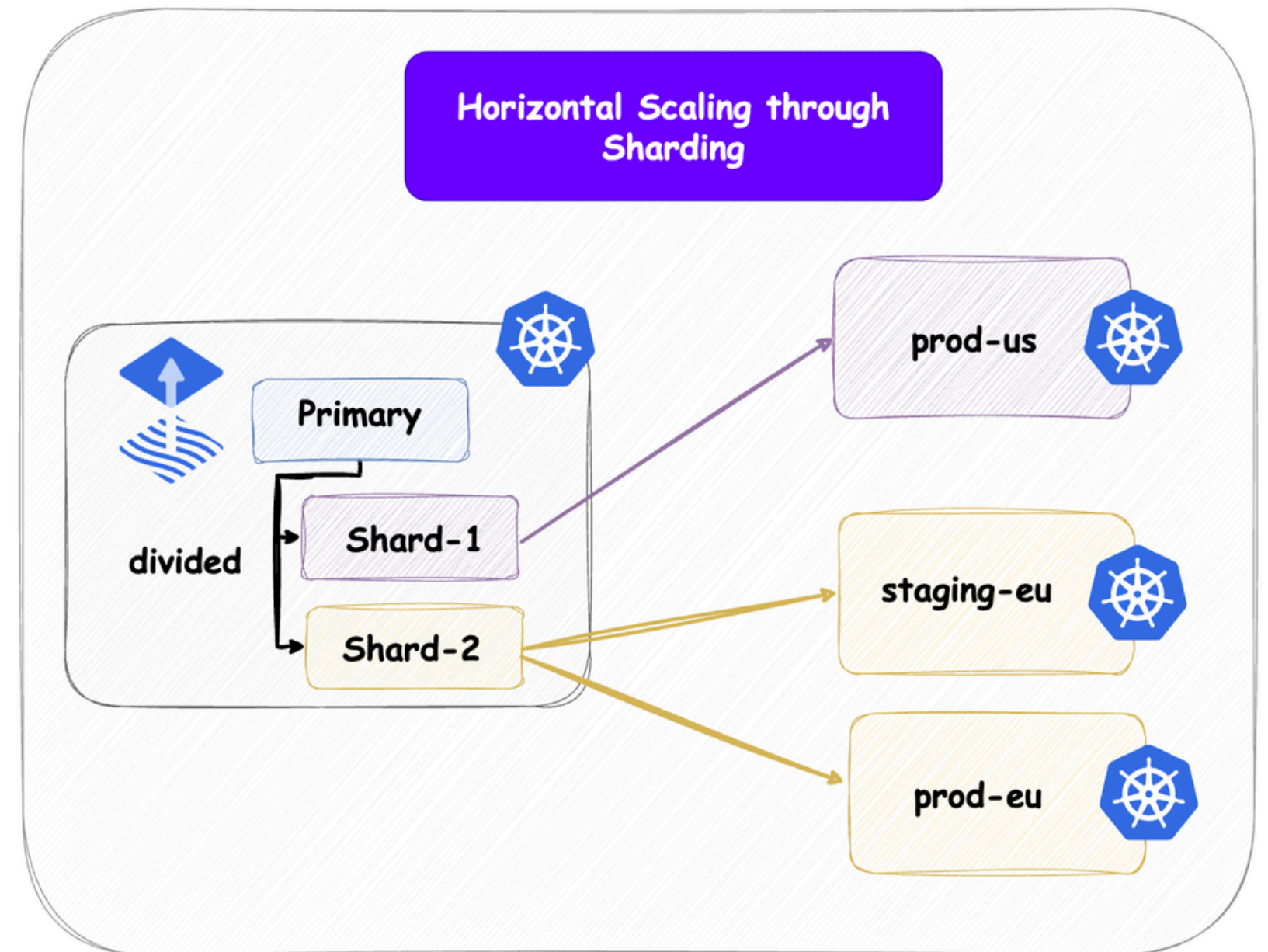
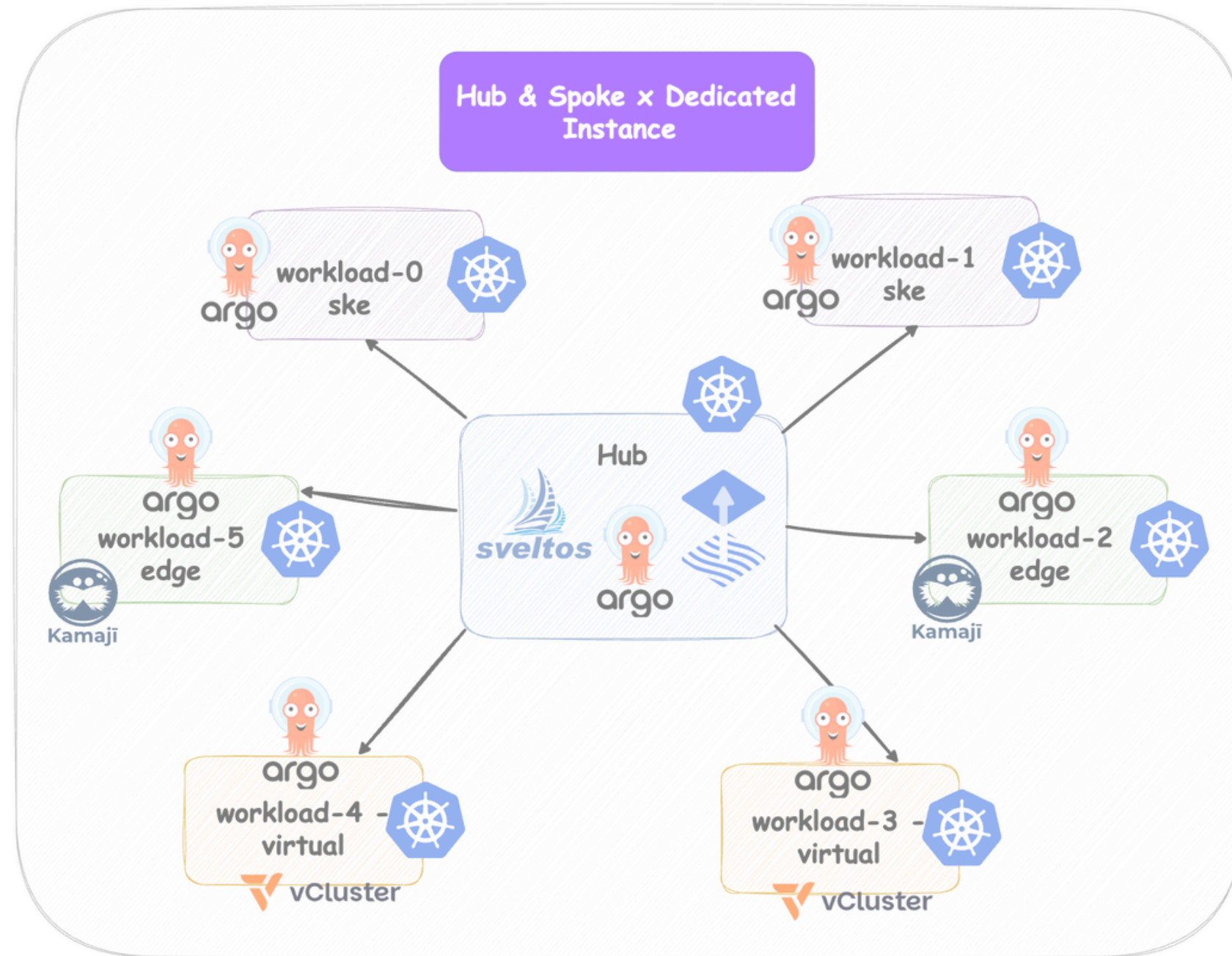
Or: combination of both architectures.

Hub & Spoke where it fits. Dedicated Instance where isolation matters.



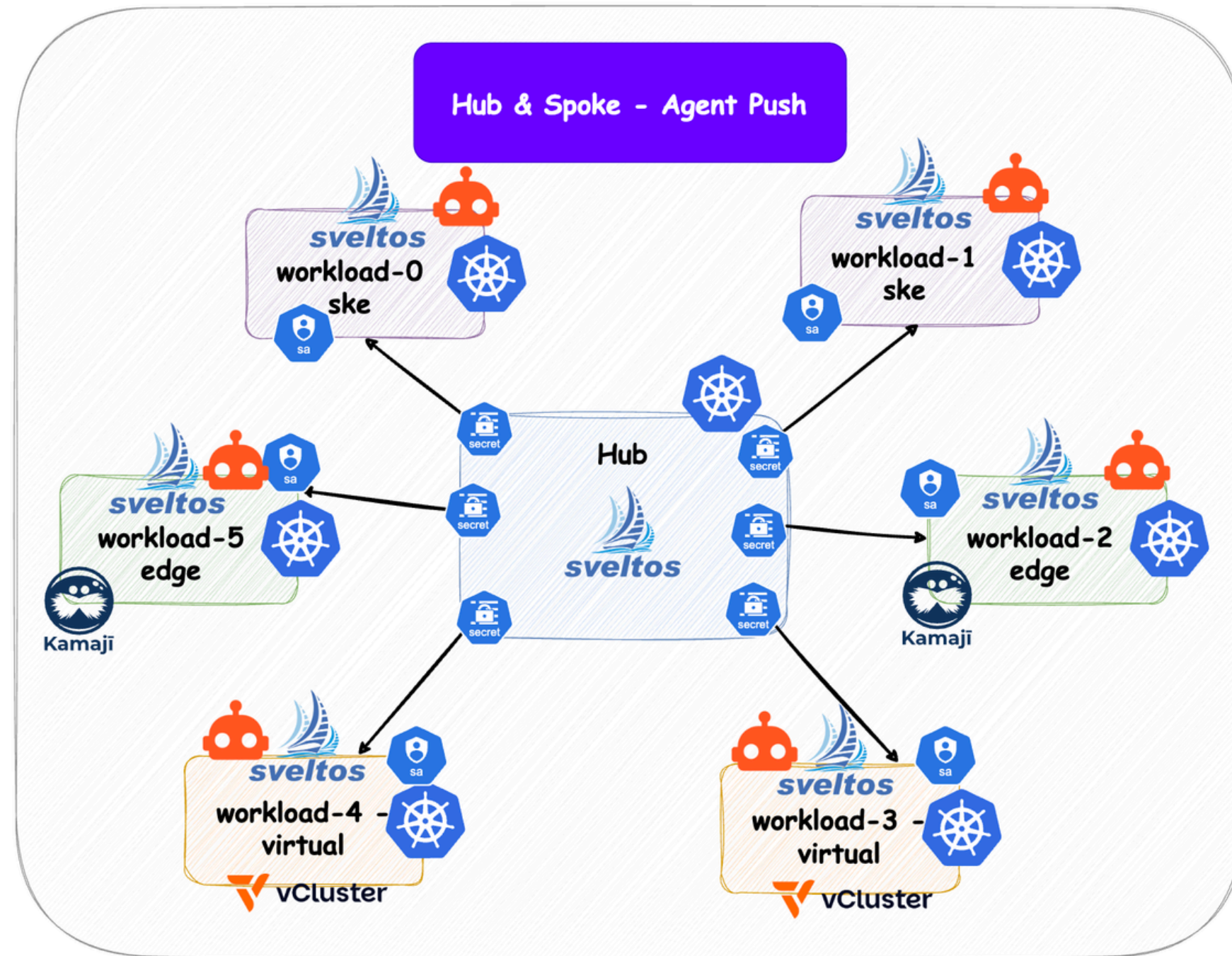
With scale, sharding becomes interesting.

Split reconciliation work across shards as the fleet grows.



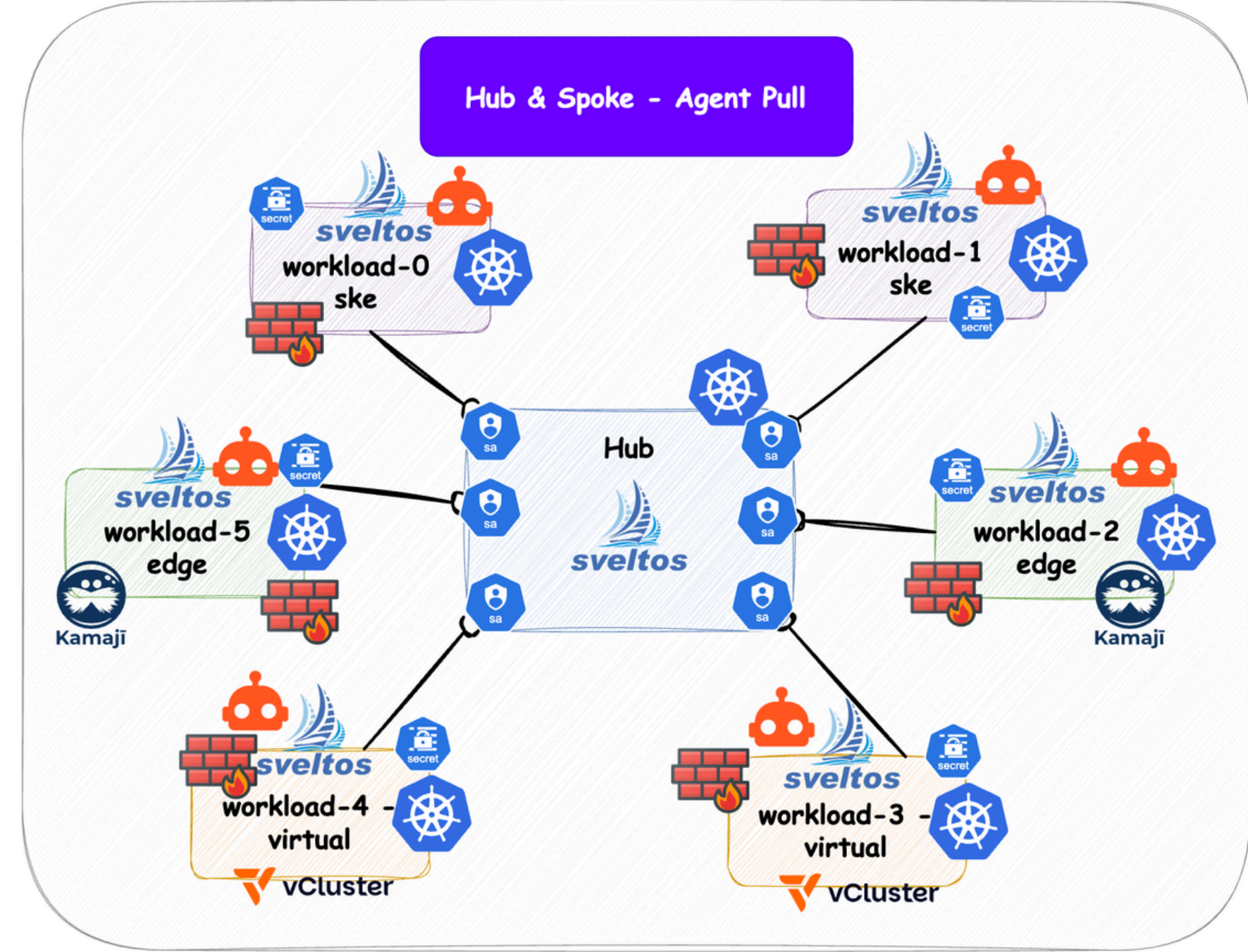
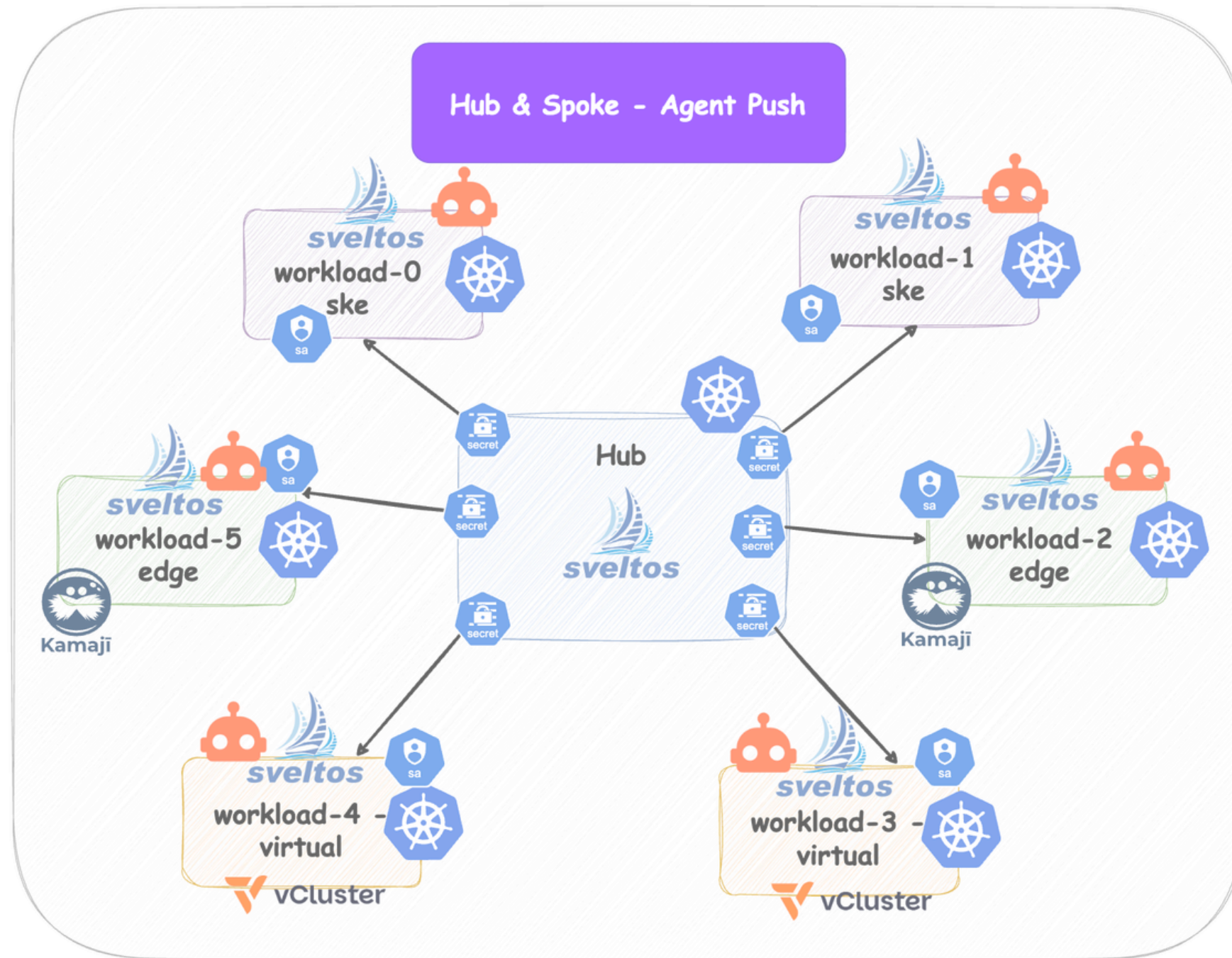
Agent-based Architectures evolved.

The hub stays central, while agents take ownership at the edge.



Pull-based Agents shine in restricted environments.

Instead of pushing into clusters, agents pull desired state from the control plane.

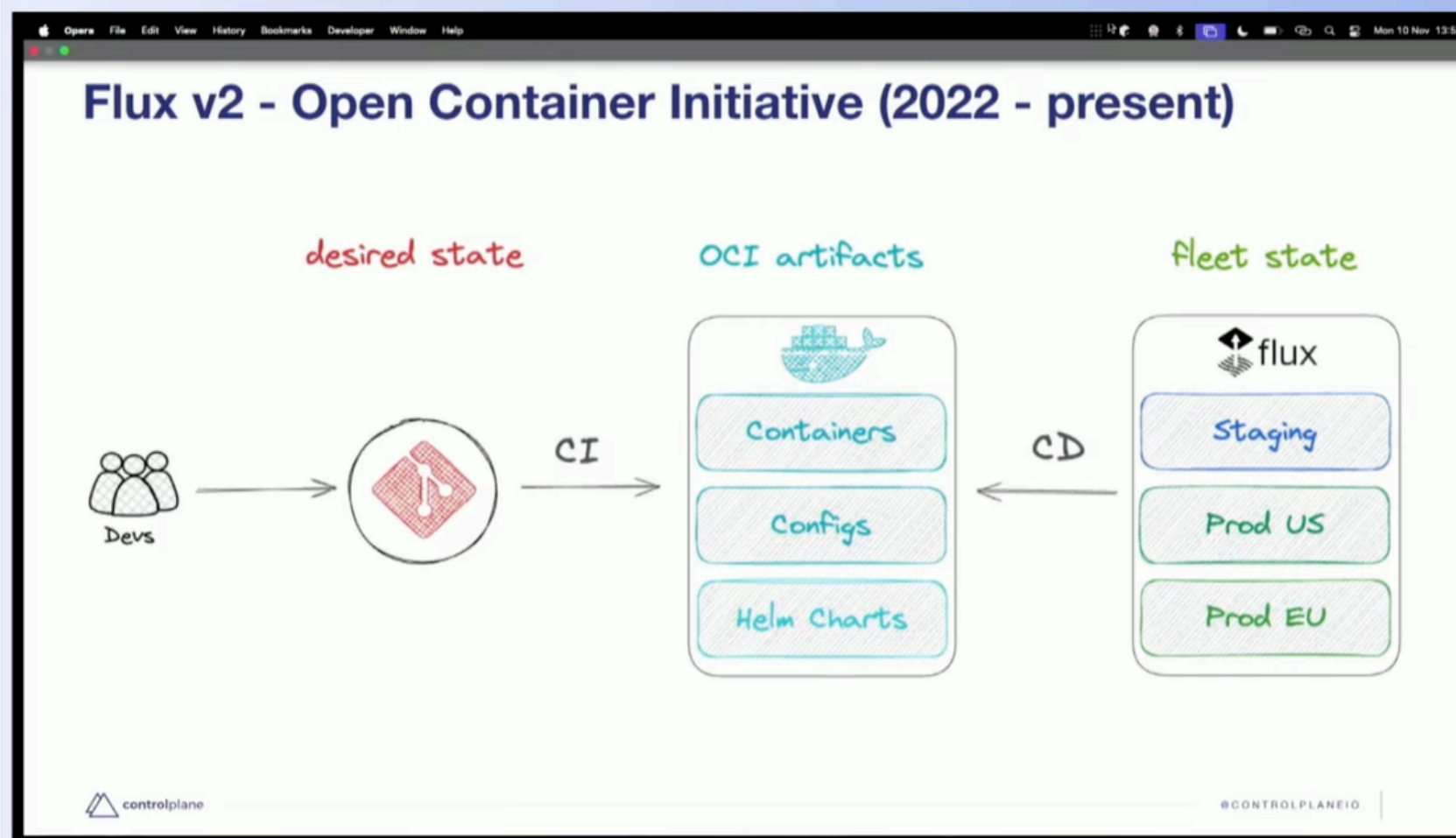


Gitless GitOps



Gitless GitOps with OCI rise

in 2022, desired state could move from Git-only workflows into OCI artifacts.



Git vs. OCI.

Git is great for human collaboration and intent.
OCI is built for immutable artifact distribution.



Git

Mechanism: Stores intent as version-controlled configuration (YAML, Helm, Kustomize).

Acts as a collaboration and change tracking system.

Best For: Capturing who changed what and why (audit trail, PR workflows).

Challenge: Not a state store and not the full Source of Truth. External dependencies (images, charts) live outside *Git* and cluster mutations can cause drift. Limited scalability due to frequent polling and repo sync.



OCI

Mechanism: Packages fully rendered manifests into immutable OCI artifacts and distributes them via registries.

Best For: Scalable distribution of deployment artifacts across many clusters.

Advantages:

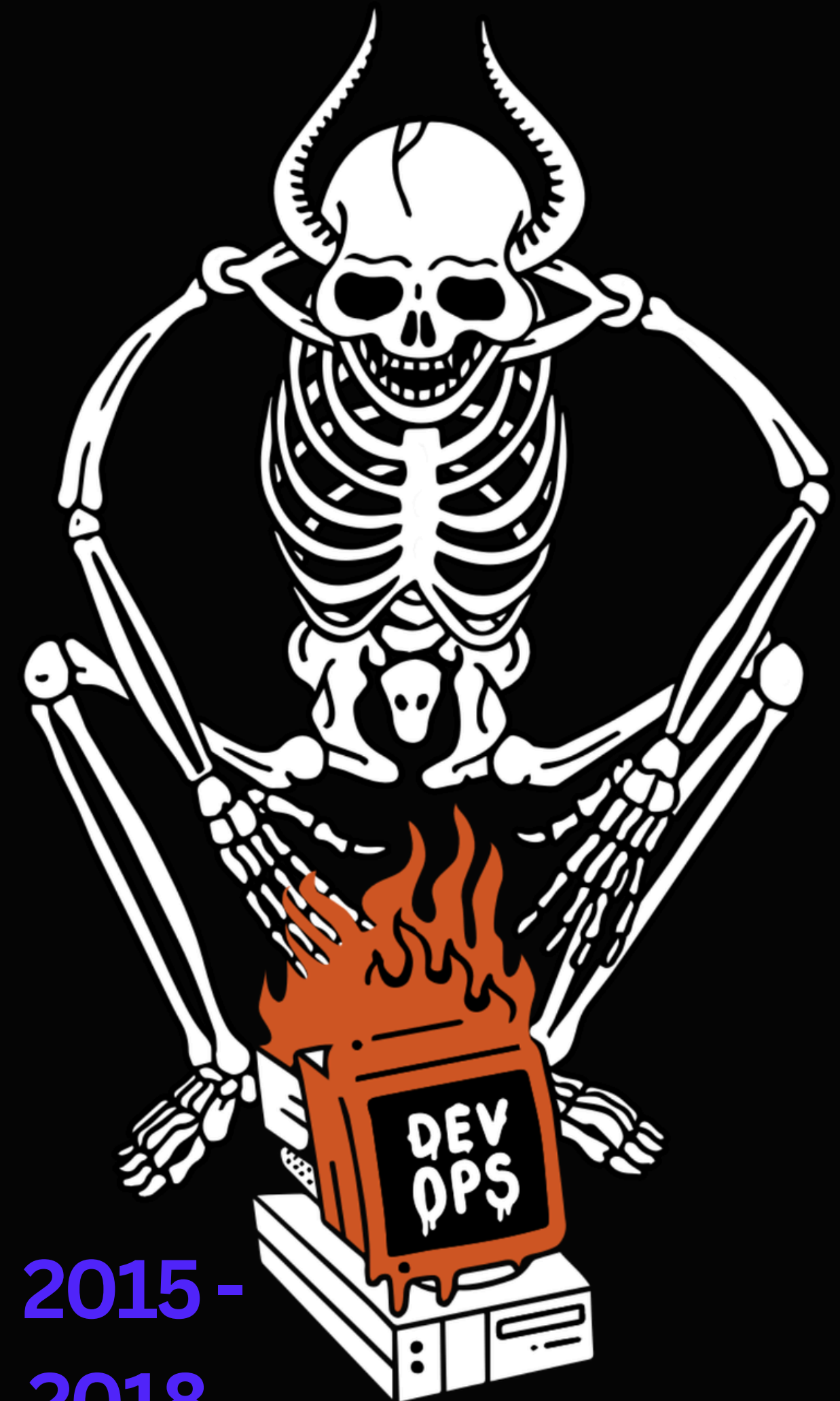
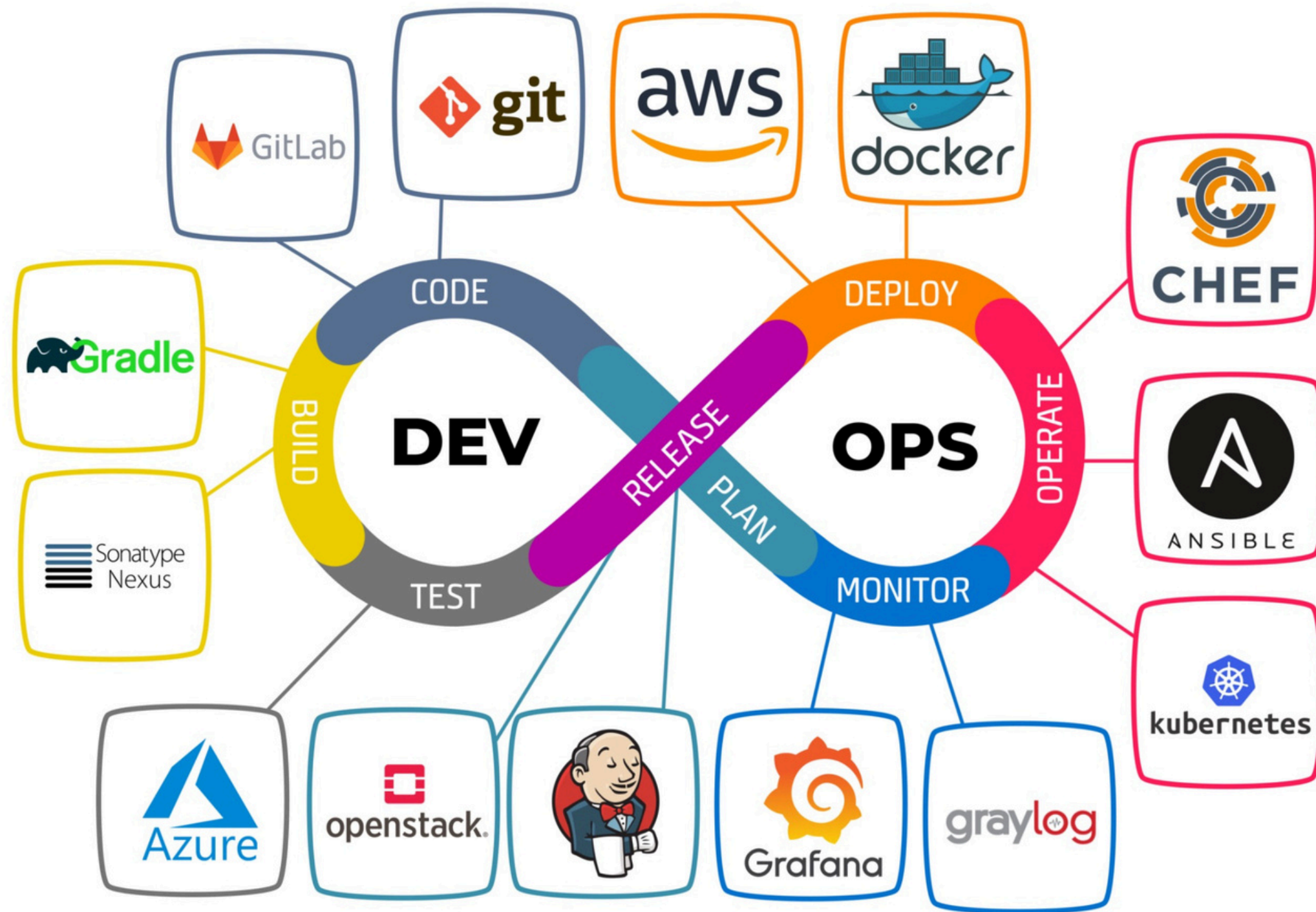
- **Faster sync:** pull artifact instead of cloning repos
- Immutable releases via digests
- Reproducible deployments



In the meantime.



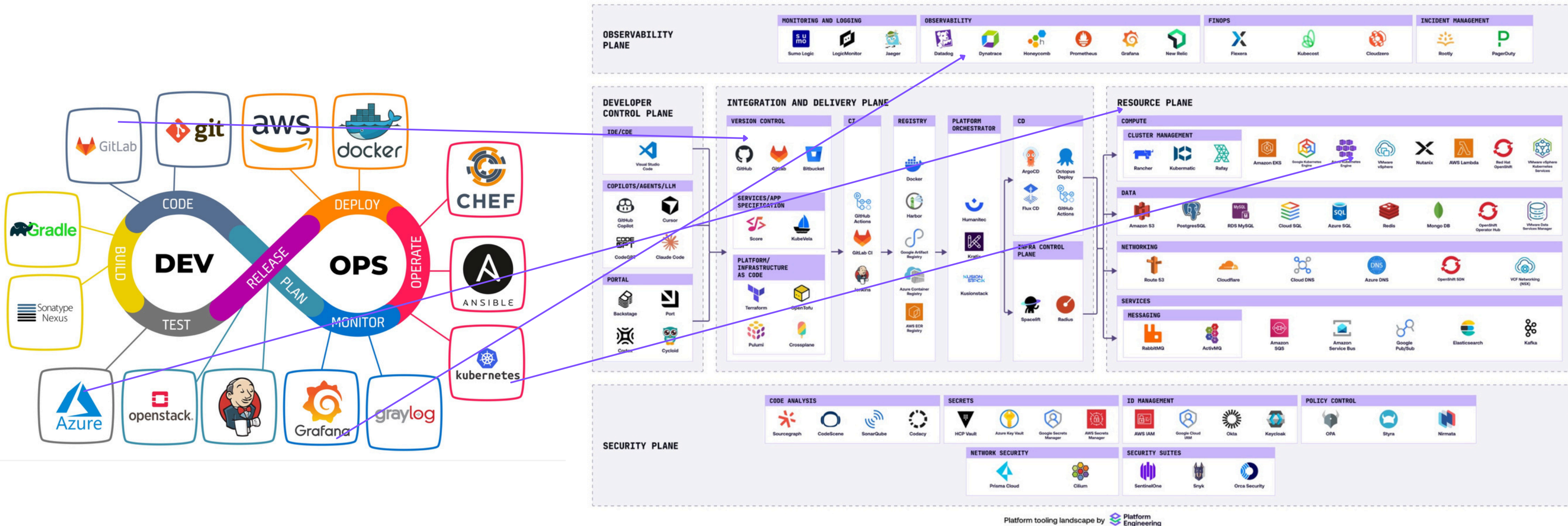
DevOps was declared dead.



2015 -
2018

Platform Engineering rises.

The DevOps toolchain expands into an internal platform ecosystem.



The challenge moved from individual tools to planes.

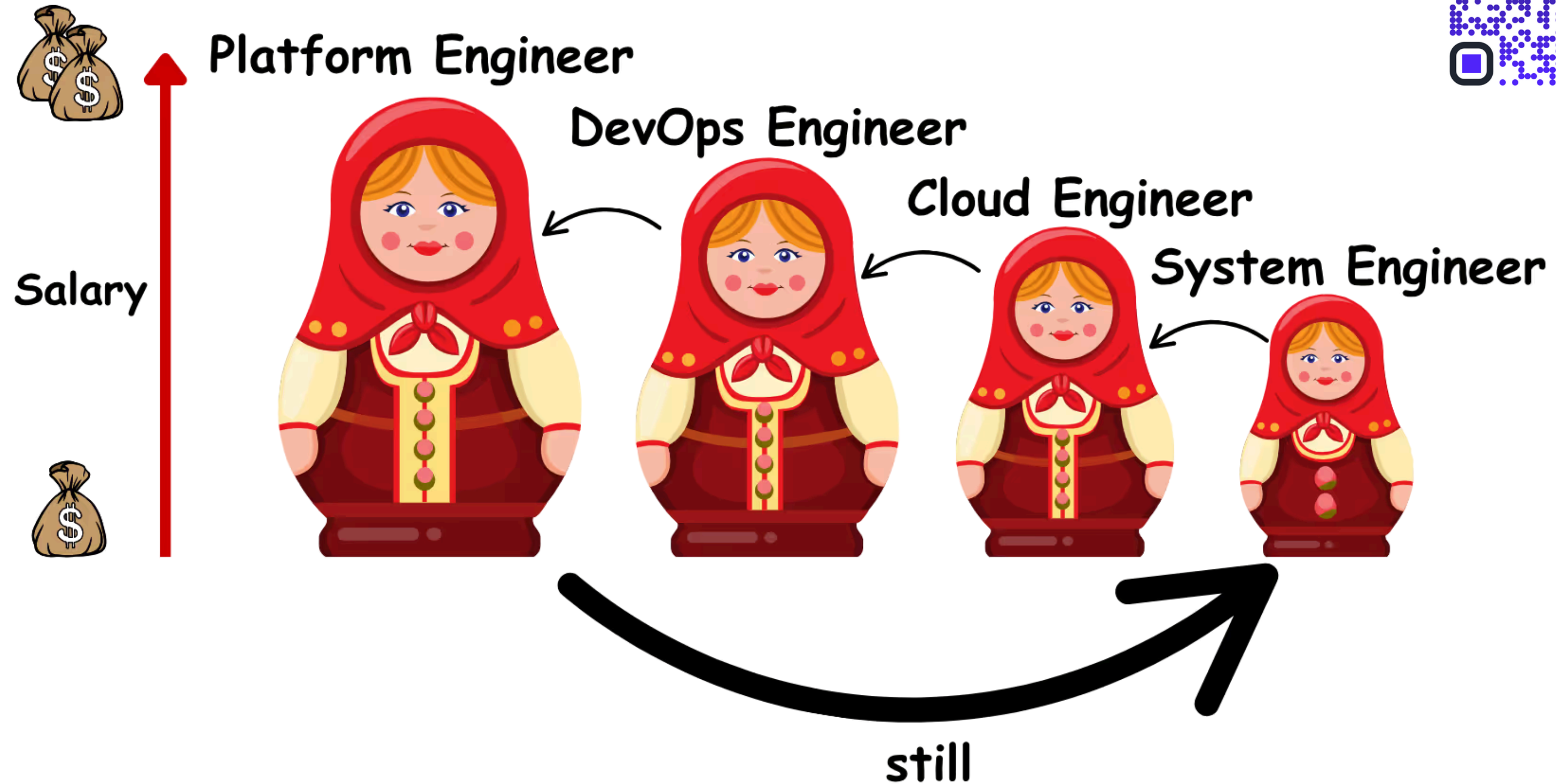


Central platform teams

are created.



Sorry...we mean relabeled.



Same Team. New Mission.

The relabeled DevOps team is now expected to build an Internal Developer Platform.



To *increase* Developer Experience...



**...and provide Self-
Service.**





Through the **right** abstraction layer(s).

Platform API

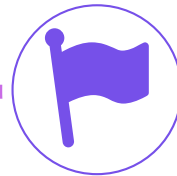
Abstractions

Golden Path

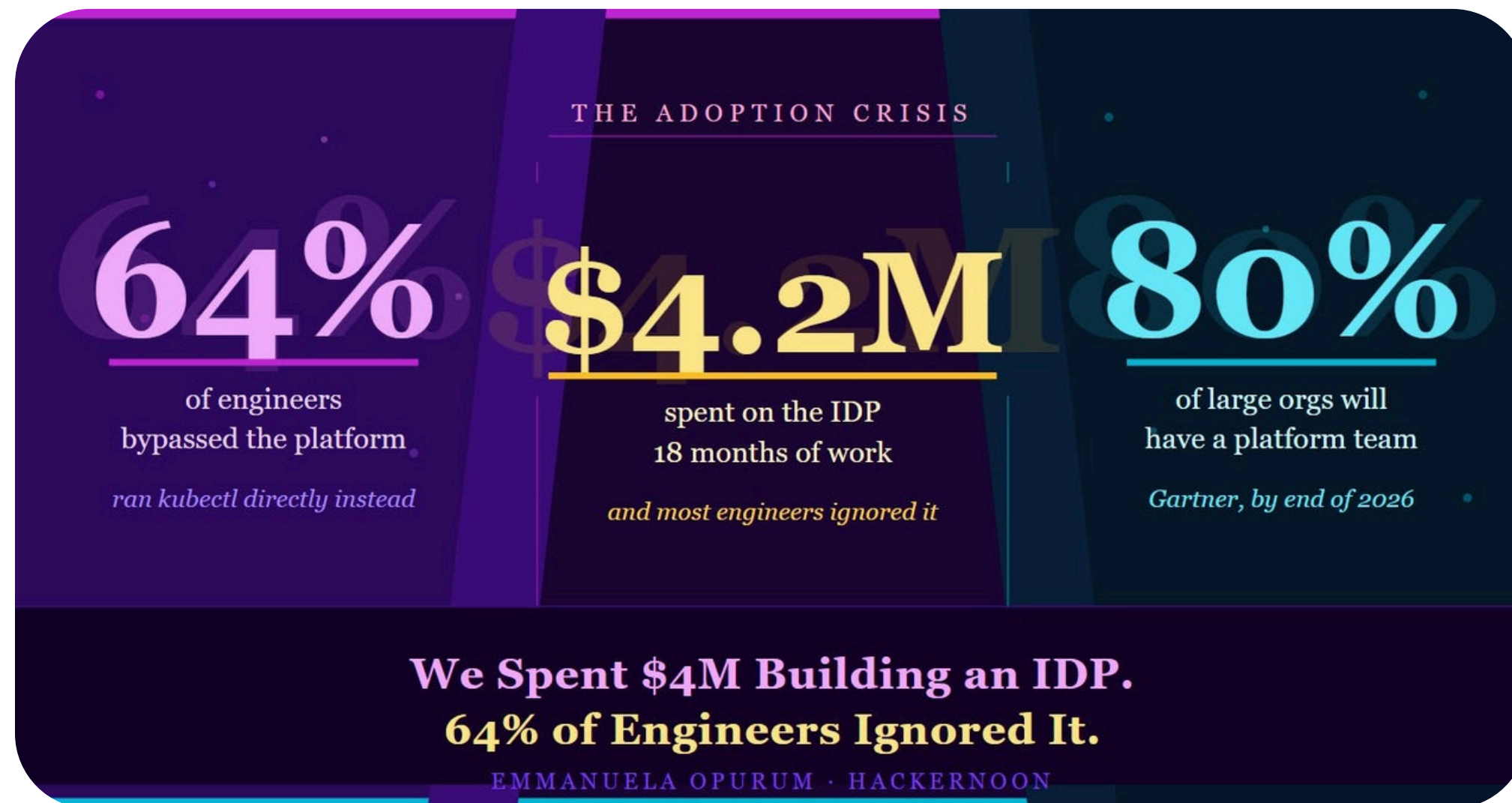
Developers



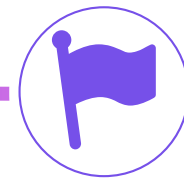
After years of failed attempts and millions spent...



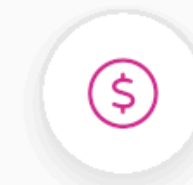
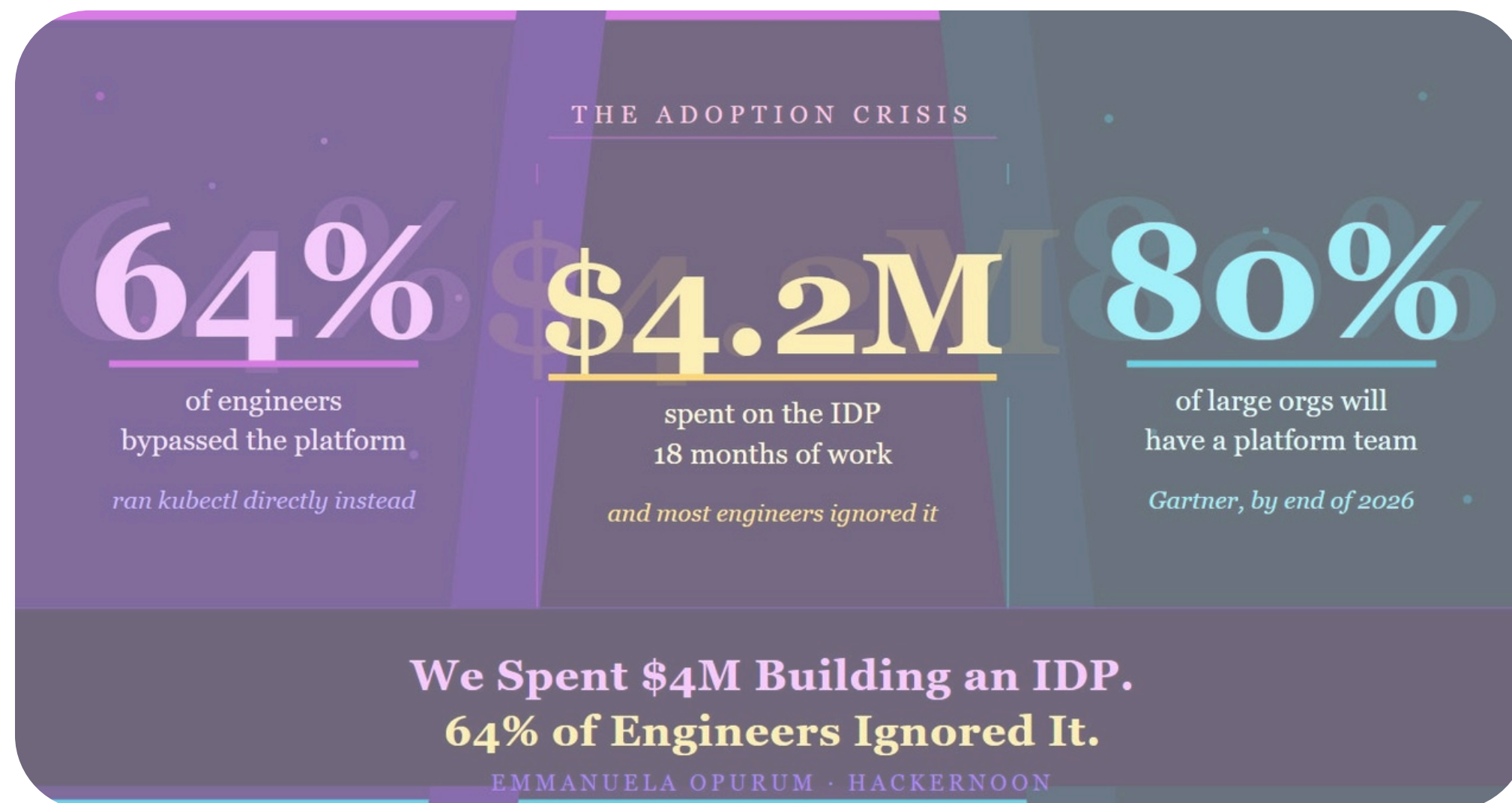
we still struggled to build platforms
developers would use voluntarily.



After years of failed attempts and millions spent...



we still struggled to build platforms
developers **would use voluntarily.**



SOFTWARE VALUE
\$19.2M



TOTAL CONTRIBUTORS
1,256
+291% vs. previous year



Deploying Backstage $\neq$ Building an IDP.

Backstage is a framework. The platform is still your job.

“Let’s just use Backstage—it’s production-ready, right?”

“Just deploy it, it’s just a Helm chart!”

“It’ll cover everything we need... just check the docs!”

“Wait, we only get example Node.js setups?”

“Where’s our stack? And who said we’re liking React?”

Portal framework

Not an out-of-the-box IDP

Heavy customization

Example Node.js setups

Your stack, you integration work

What you have



What you want



What you get



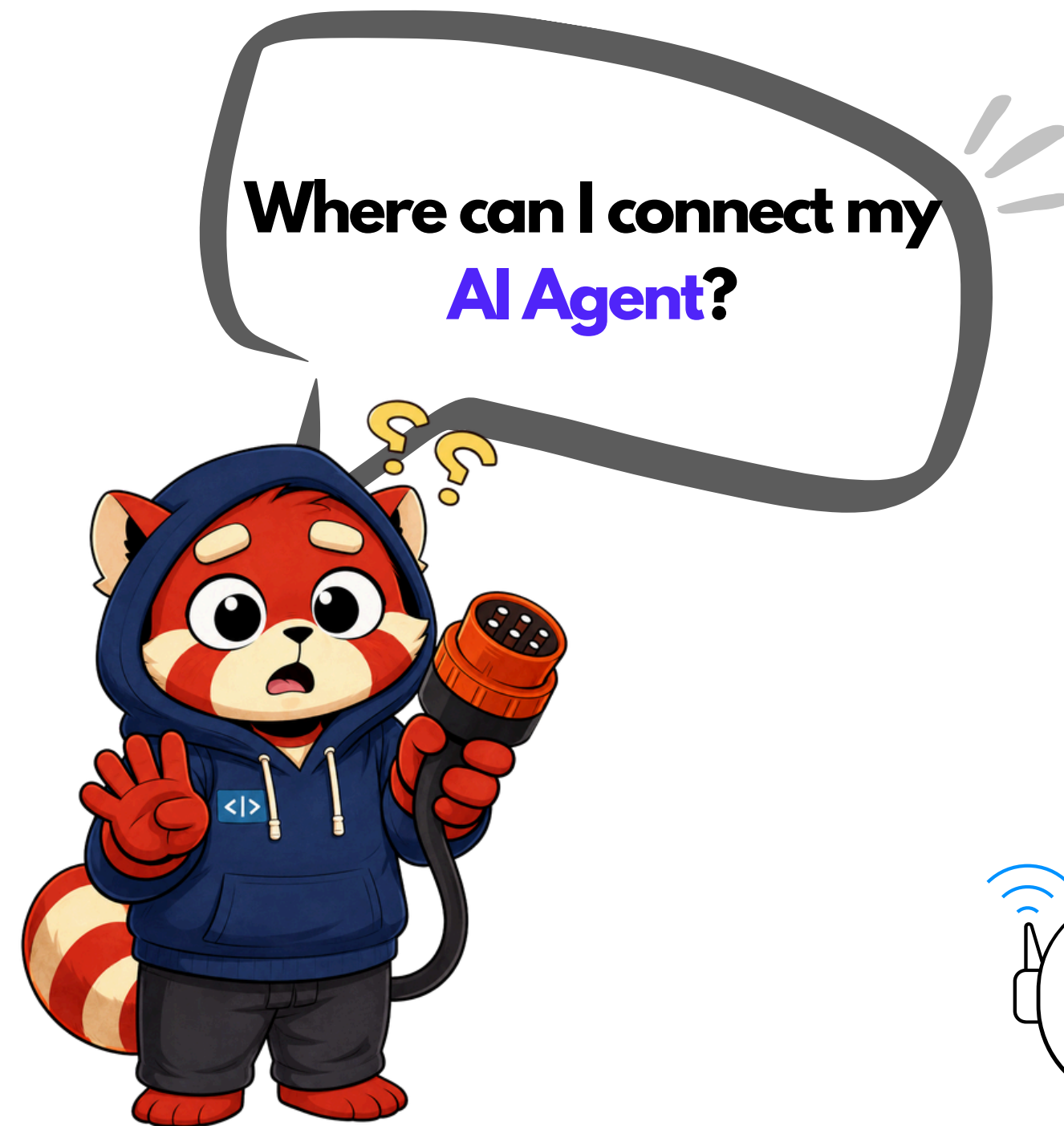
Out of context, Backstage is great—once you understand why Spotify built it in the first place.

After years of failure, we finally got adoption.

Developers started asking new questions.

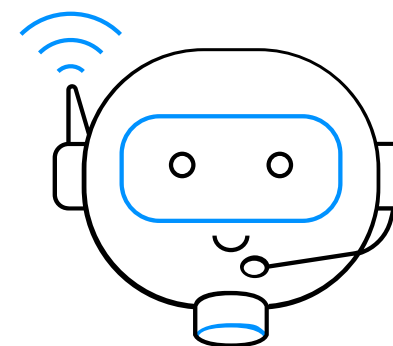


Questions like.



Developer

Wants to build great things with AI.



AI Agent

Needs to connect to platform system.
A2A. MCP.



Platform Engineer

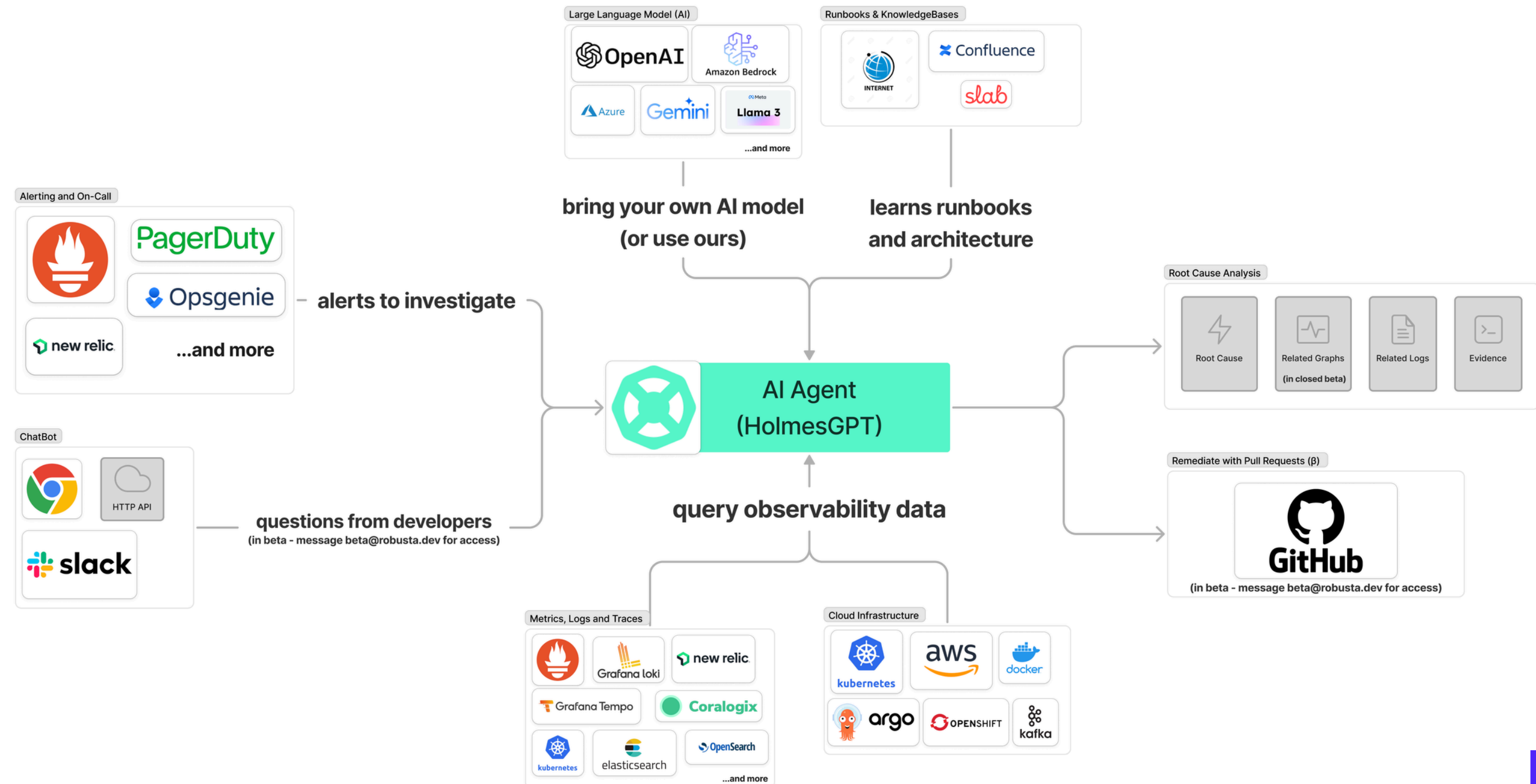
Builds the connections, guardrails and enablement.

Meanwhile...
AI is not just becoming
a citizen of your platform.



AI SRE Agents like Holmes GPT don't use your platform.

They **operate** it.



But...

You will...

...need to integrate them.

...need to guard them.

...need to give them an identity.

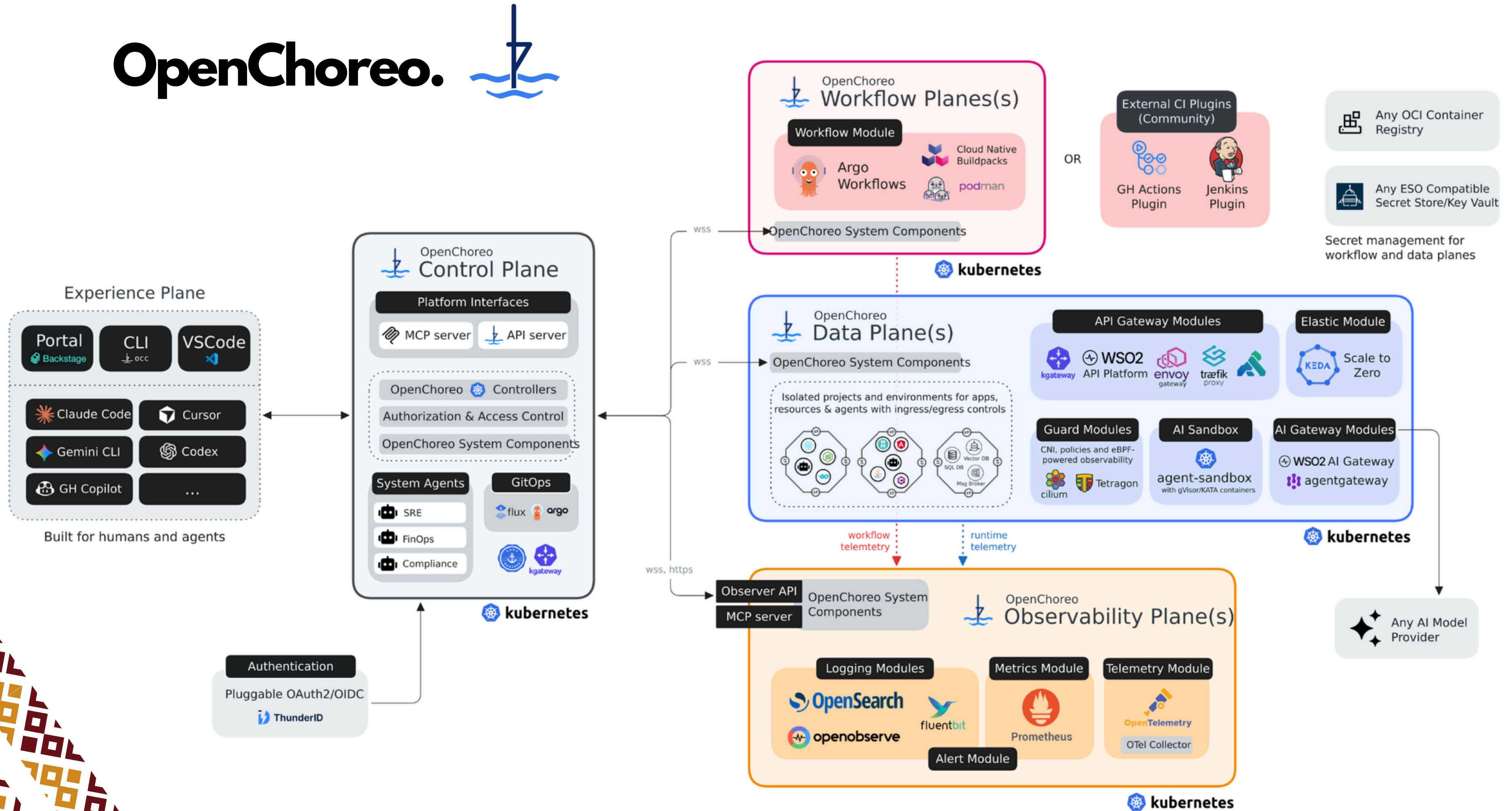
...need to sandbox them.

...need to give them capabilities.

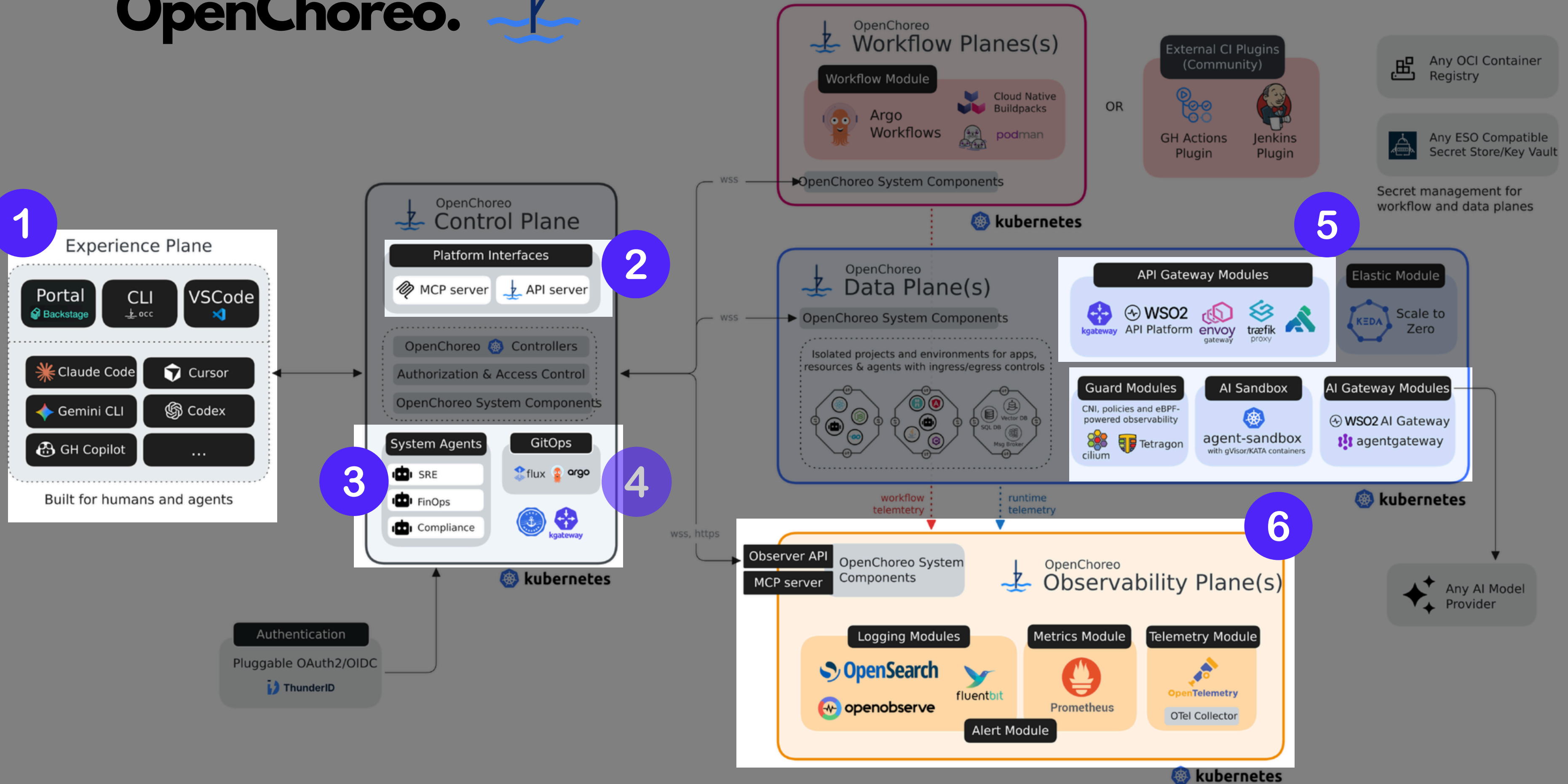
...need to observe them.



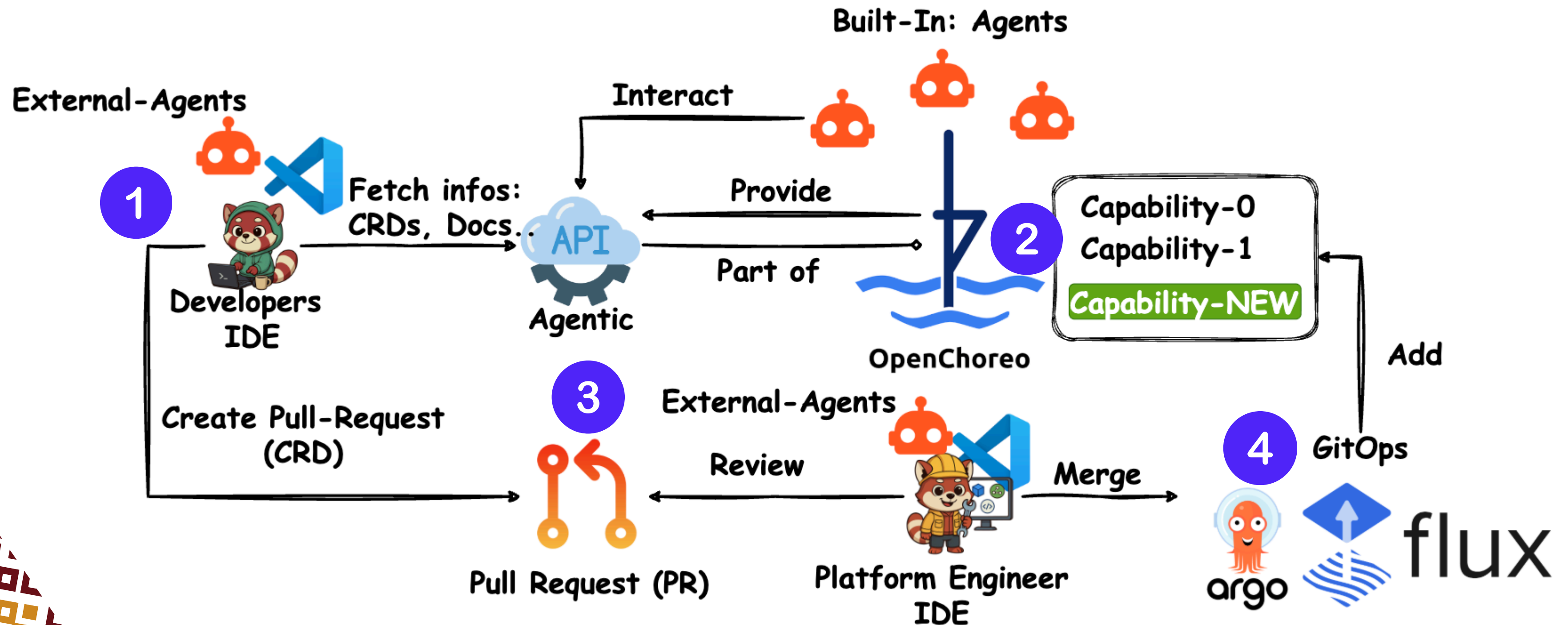
OpenChoreo.



OpenChoreo.



Put the Evolution together.



The Movement: From Internal Developer Platform (IDP) to **Agentic Development Platform (ADP)**.

*“For a decade we enabled developers.
Now we have to enable their agents, because they're the
ones using the platform.”*

Artem Lajko



Questions?



Not covered topics.
Happy to discuss.



Asante!

[Linkedin.com/in/lajko](https://www.linkedin.com/in/lajko)

