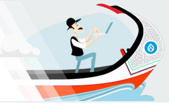
Drupal @ scale @ Dropsolid

Tales of building a Drupal-centric platform

manuel.gomes@dropsolid.com
https://manuelgomes.me



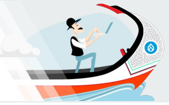
Dramatis personae

Manuel Gomes helps people and systems work better together.

- Has been a techie since the 90s
- Tells dad jokes (on purpose?)
- Product engineer at…

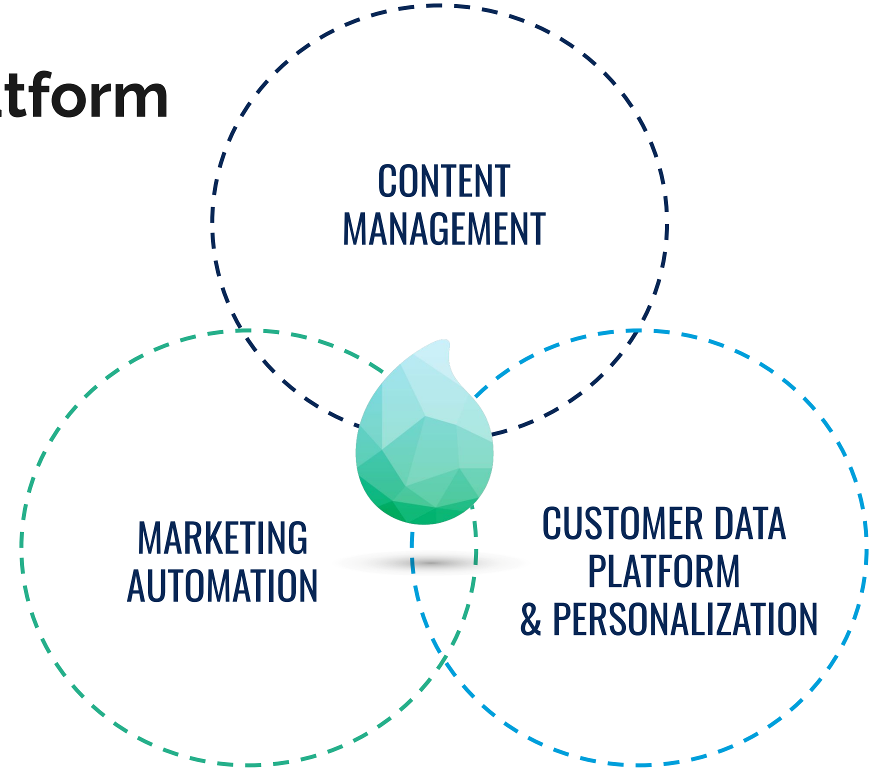
Dropsolid aims to make the best **digital experiences** accessible to everyone.

Driven by an **open culture** and with a passion for **open source**, we share our knowledge, our code, and our talent with our clients and communities.



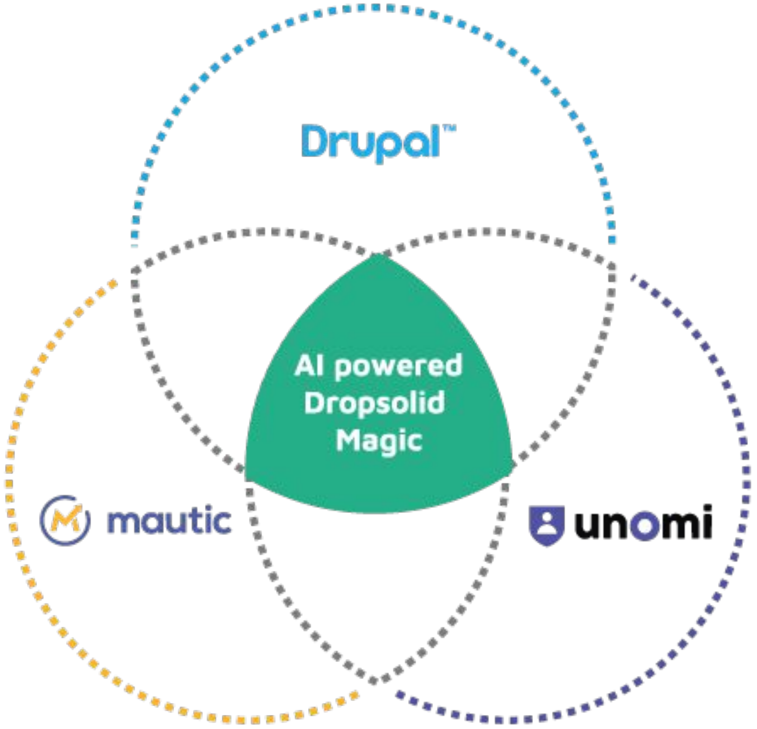
DXP: Digital eXperience Platform

Build, manage, deploy, and continually optimize digital experiences for all users across all channels, such as websites, emails, mobile apps, chat, etc.



Dropsolid Experience Cloud

- All Open Source
- Complete data sovereignty
- Security (ISO27001) and Privacy (GDPR) built-in
- Community Native



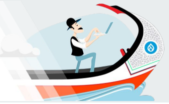
This is (mostly) a tech talk. From a weird angle

Stories we tell, and the language we use to tell them.

Nouns, *verbs*, *adjectives*, *grammar*, and *semantics*.

How language maps mind. And mind maps systems.

Tools that embody our lexicon well - at certain points of scale.



Our story starts with Dropsolid

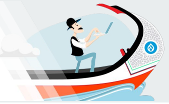
Awesome Drupal-centric company founded 11 years ago. It **grew**.

~700 projects ~1500 environments ~200 servers

(and that's just *production*)

… you don't want to manage that manually

So we built a platform



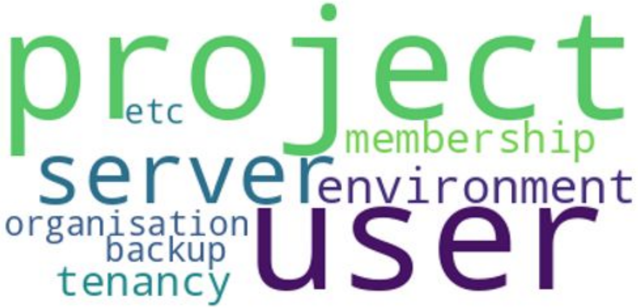
It's been quite a journey (and it's far from over)

> I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail.
>
> – Abraham Maslow

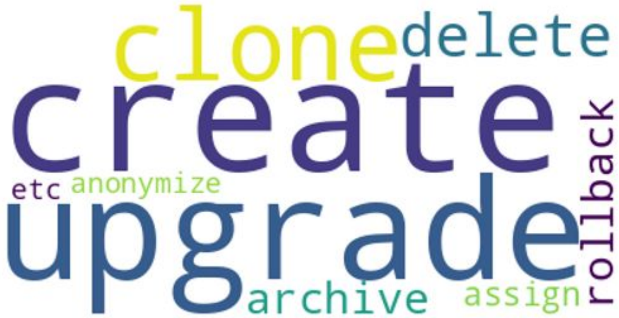
So obviously... we used Drupal to build it!

Domain modelling

Nouns



Verbs



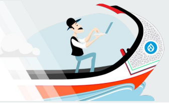
Nouns the Drupal Way

custom entities for nouns: projects, environments, servers, memberships, organisations, users, CDP, …

ACLs for memberships: Gitlab, OAuth2 Proxy, others

…

and **state**… but more on that later



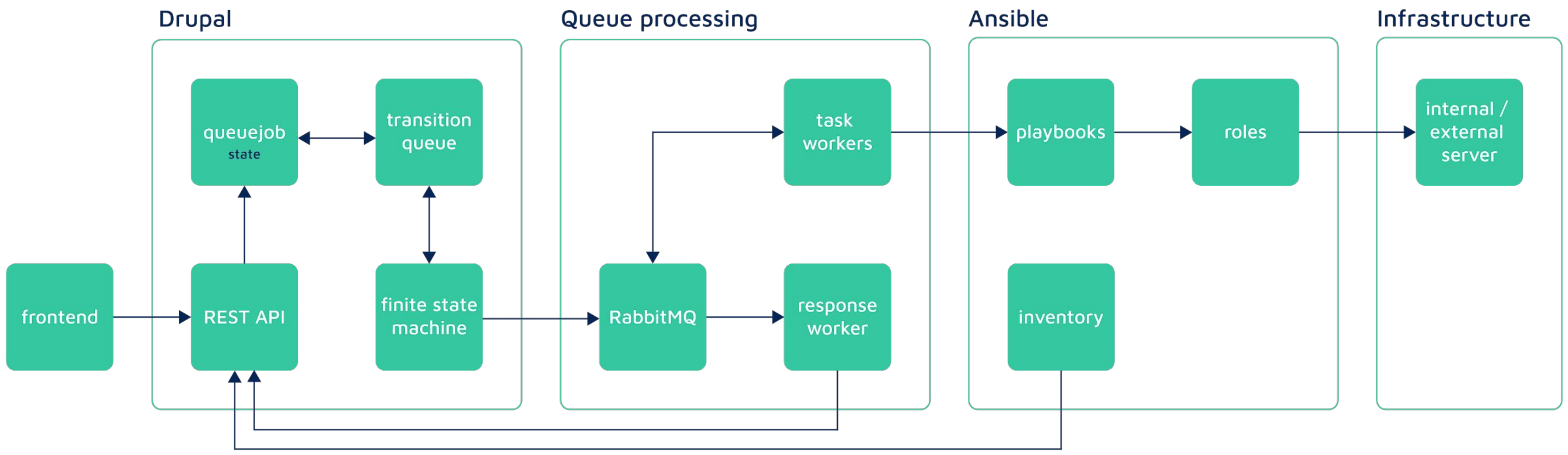
What about verbs?

We did everything in…

ANSIBLE

… but more on that later

Putting it all together`



It made sense.

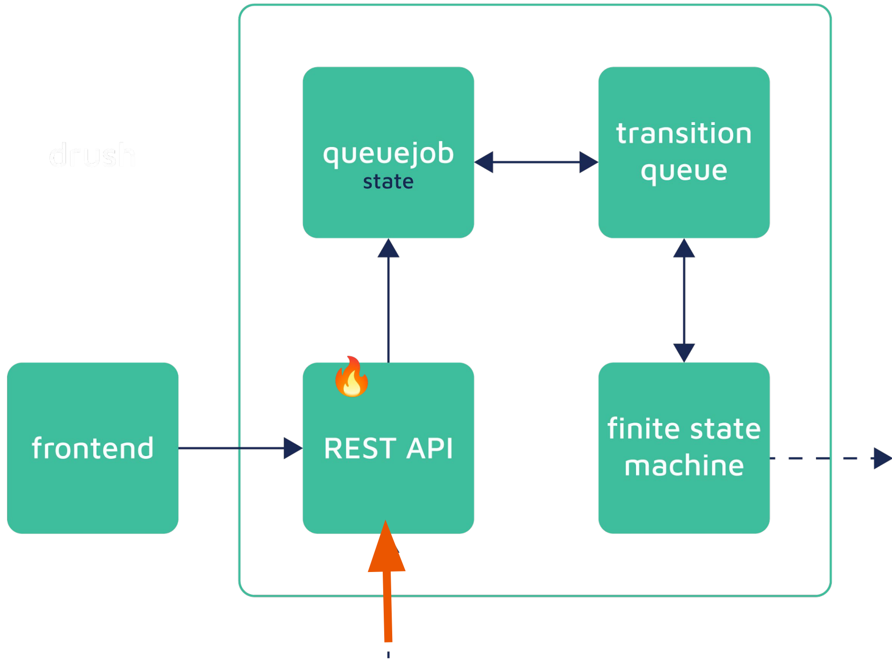
It worked pretty well.

until...

Drupal is not an application framework

Awesome CMS, but… not a high concurrency application building framework

High revision overhead!
Bad blob handling!

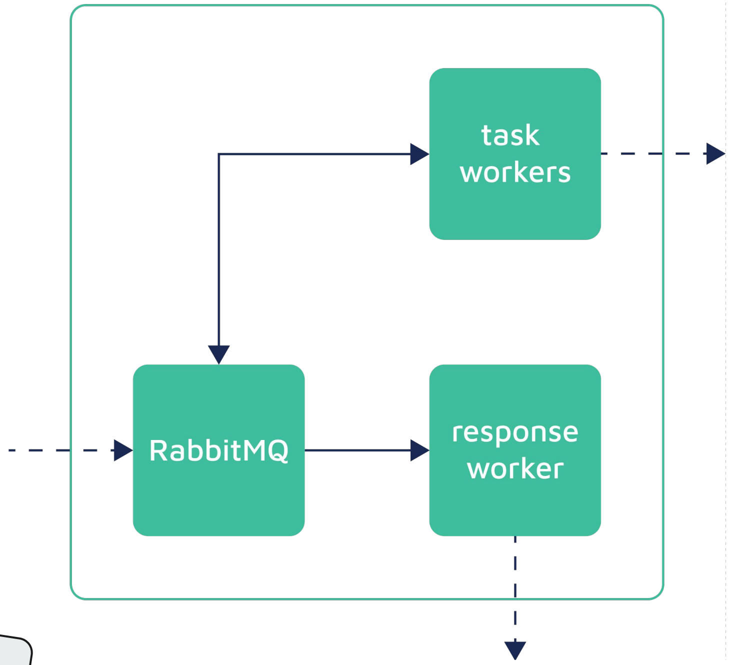


…lead to **deadlocks**

We defended with smarter queueing

Celery on top of RabbitMQ

Retries with smart backoff
Reliable interim state store



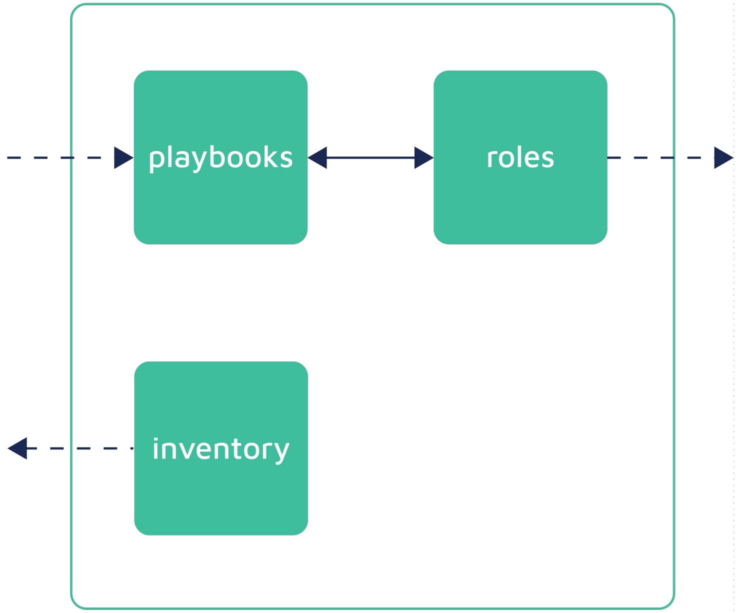
Semantics!

at-least-once
delivery

While Ansible takes care of most platform verbs

Which is great!

… mostly…



So let's talk about ANSIBLE

Sometimes we use it just right

Brilliant at "its thing":

Creating, provisioning, configuring servers

Broad Ecosystem

Flexible, extensible

… perhaps a little too much?

Some more Ansible

Great at (re)writing configuration files

creating, starting, stopping, restarting services

BUT

It has no notion of its own concurrency

It doesn’t really know “rollback”

Atomic writes are… as good as you make them

Too much Ansible!

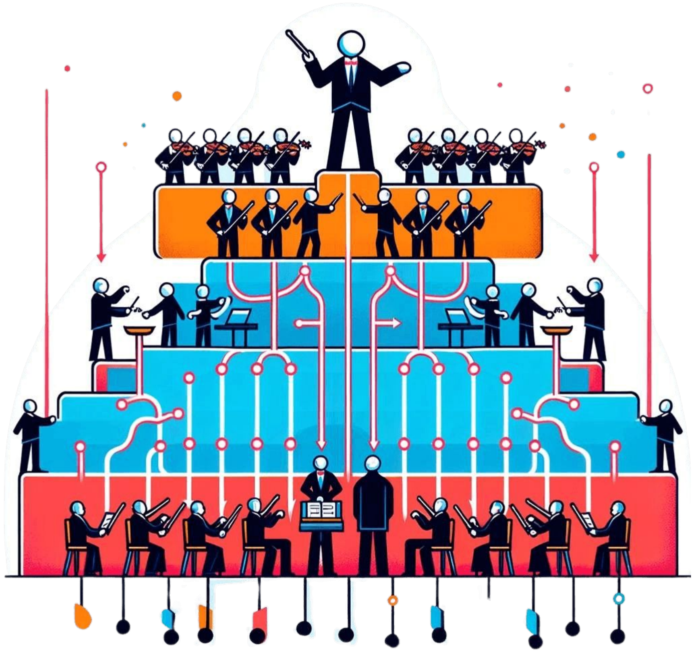
It is **not** a programming language

It is **not** an application framework

It can **run** applications made in frameworks of programming languages

But boundaries and separation of concerns become extra hard

That’s a lot of orchestration!



It’s very hard to keep track of it all

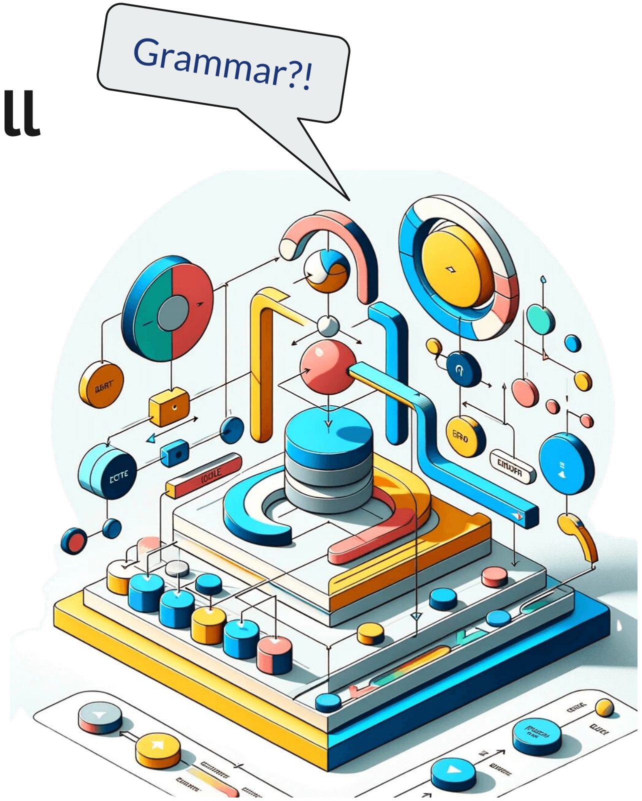
State machines to the rescue!

Deterministic success/failure

If something crashes, you know exactly what and where

You can resume a workflow

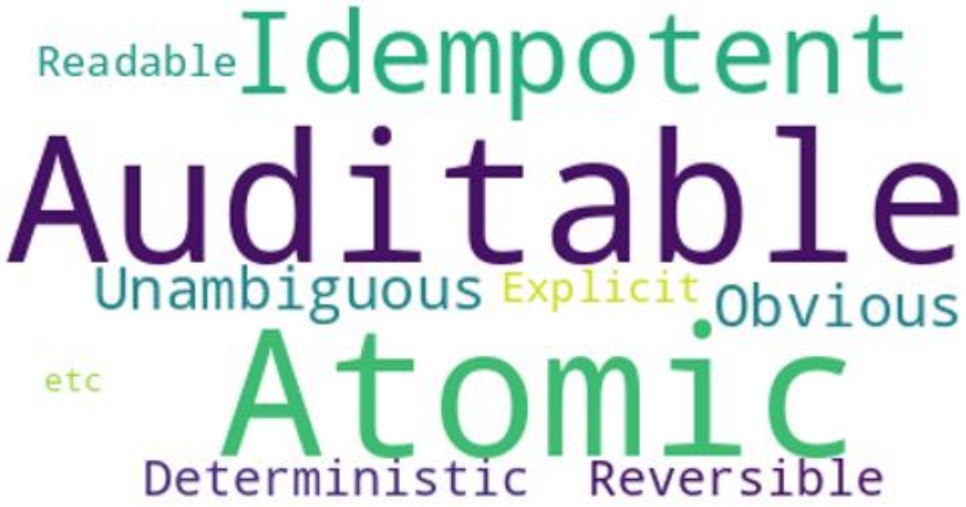
Enables atomic “revert” options



We have nouns, verbs, semantics, and grammar

With them, we can build **meaningful sentences** with which we articulate value

They **should** be



Maybe this rings a few bells...

As a `<persona>` I want to `<action>` so that `<outcome>`

Behaviour **D**riven **D**esign scenarios, like in *`behat`*

Domain **D**riven **Design**'s *ubiquitous language*

... coincidence?

Tending your abstractions for ~~fun~~ sanity and profit

We’ve talked about some unusual stuff for a tech conference

- Vocabulary
- Semantics
- Grammar
- Value narratives

Why are they relevant?

Alignment and **Decoupling**.

Good abstractions, good sentences, allow us to maintain a consistent narrative of our value delivery to our customers, while swapping out implementations and supporting infrastructure in whatever way necessary.

Everything breaks at a scale

As a general rule, the greater the scale, the greater the necessary level of abstraction.

Many abstractions get leakier as we scale up.

… but we shouldn’t get ahead of common sense **for our scale point**

How you start is often not as you finish. **And that’s OK**

One simple trick (product managers love it) !

> If your “platform sentence” sounds like something your customers say, you might be on the right track

Corollaries:

- A good platform makes writing frequent customer sentences easy
- If what you’re writing is both *necessary* and customer-*nonsensical*, you’re writing plumbing, not platform. “drivers”, not “userland”.

Manuel, can we get to the tech, please?

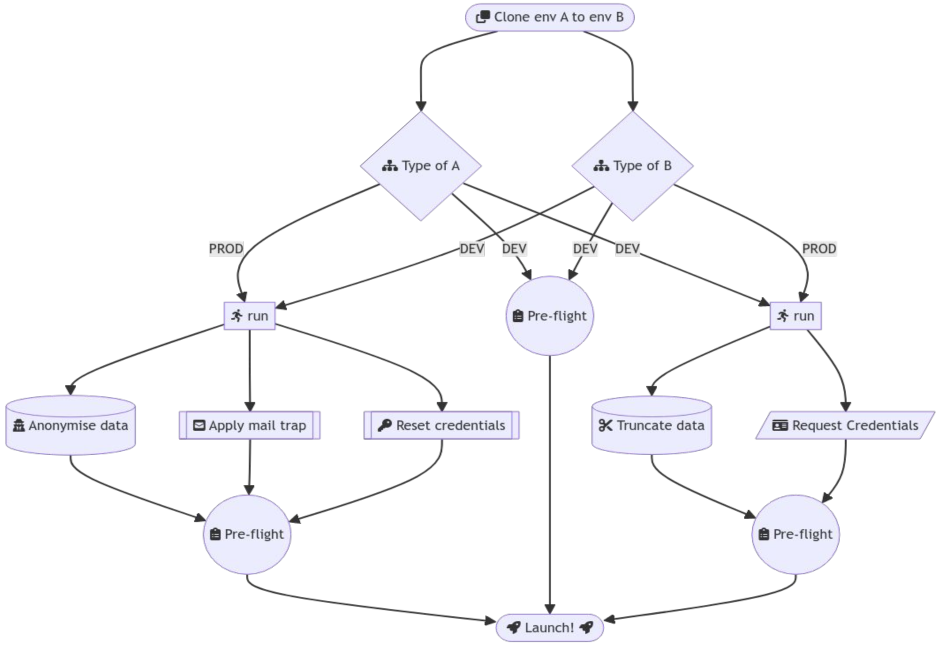
Ok, then, let's talk about Kubernetes

And let's talk about **Nouns**

Most **nouns** would do well… as **C**ustom **R**esource **D**efinitions…

… and we can let the Kubernetes API take care of the ***CRUD*** bits - those are **some** of the verbs

But we have far more complex sentences



Let's look at a simplified version of **Clone Environment**

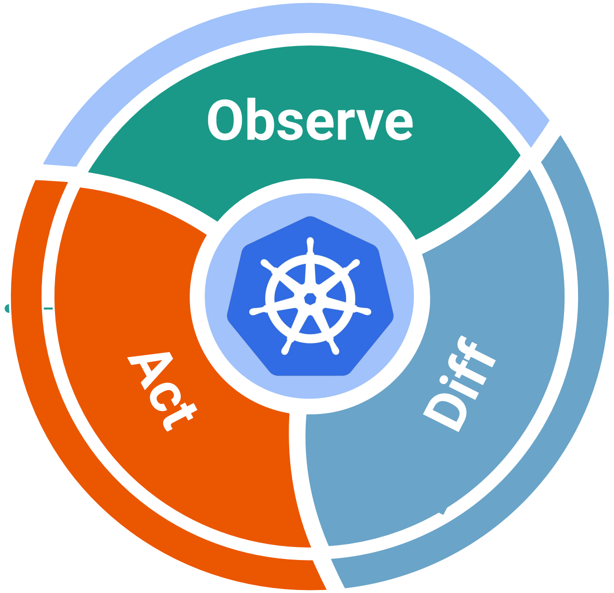
Not shown: testing, staging, QA, DQS, demo, *ad infinitum*

Maybe we can learn from past experience

Remember the whole rant “Ansible is not a programming language or an application framework”?

Perhaps it’s time to admit to ourselves that we **do** need an application to implement our grammar!

Kubernetes gives us operators



Domain-specific controllers that extend the Kubernetes API to *manage and automate* tasks based on the *specific needs* of the software they manage. They **encapsulate operational knowledge** into software that can be shared and reused.

You can build them with

GO HELM OPERATOR FRAMEWORK

And unavoidably... ANSIBLE

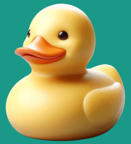
Or maybe skip the whole Kubernetes silliness…

and go full serverless!

It won’t matter

… as long as we’re speaking the

right language!


happy ducky for scale

Honorable mentions in tooling

Logging

Alloy: OpenTelemetry collector

Loki: log database backed by object store

Grafana: Front end for (embeddable!) view over loki logs and other Alloy entities

Backups (done right!)

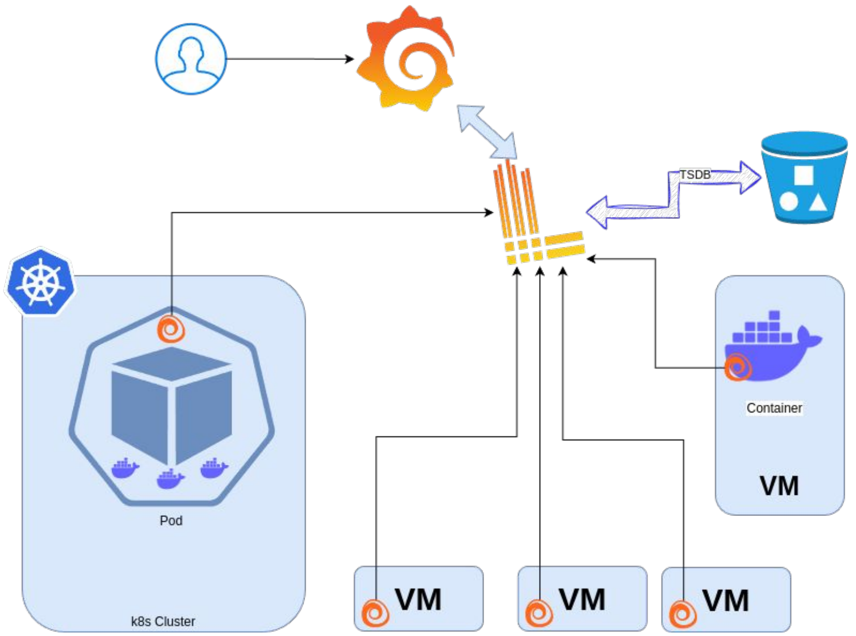
Restic

- Flexible
- Lightweight
- Simple

Restic is awesomely simple

- Establish a **repo** on: disk, NFS, MinIO, S3, GCS, Ceph…
- Define the **source**, plus any **exclusions**
- First run compresses and backs up everything
- Ensuing runs detect **differences**, compress and store the delta (versioned!)
- Restore snapshots fully, or paths within them, or mount them as FUSE file systems(!)
- Enforce retention in a cron-like syntax
- Take an early day - it Just Works!

Grafana stack? Not so simple



But worth it!

- Incredibly flexible and feature-complete
- Plays well with others, specifically Prometheus (metrics) - and other OpenTelemetry “citizens”
- Embeddable dashboards make it a full citizen of our product offering, not just ops stack.

Thank you!

Questions?

hopefully the right language..