DRUPAL IBERIA

Drupal @ scale @ dropsolid

Tales from a building a Drupal-centric platform

Linguistics addendum

Dramatis personae

Manuel Gomes helps people and systems work better together.

Has been a techie since the 20th century

Tells dad jokes (sometimes on purpose)

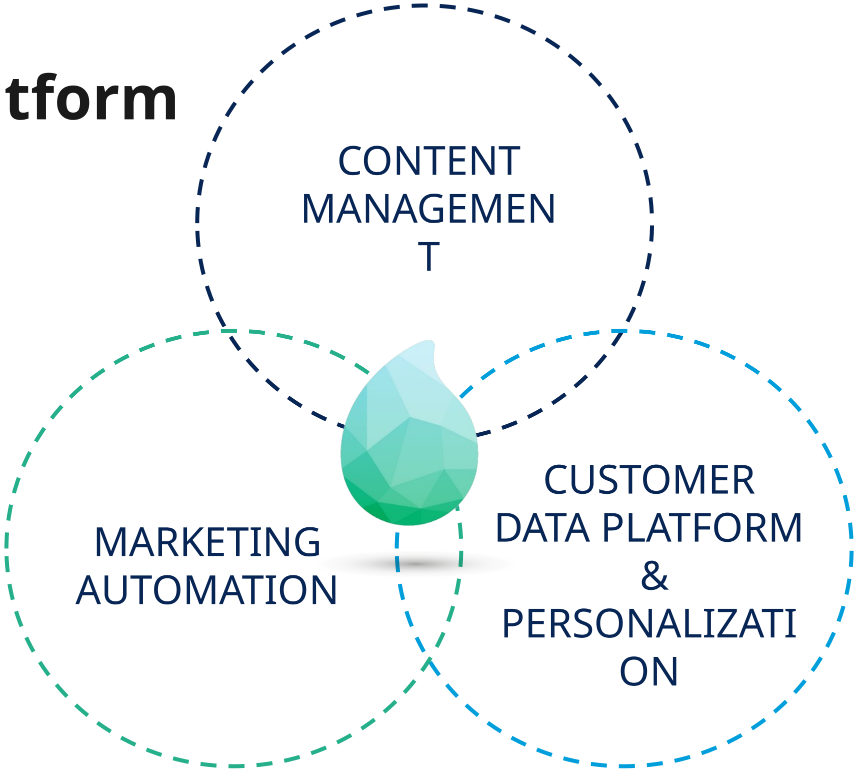
Is a product engineer at...

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.

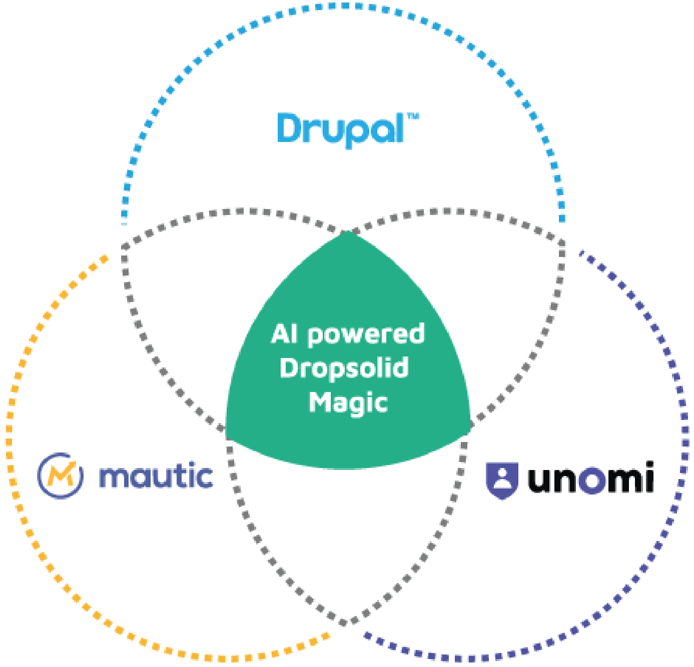
DXP: Digital eXperience Platform

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



Currently

> **500 clients** running > **550 production projects**

which under the hood means

~ **1400 environments** running on ~ **200 servers**

… you don’t want to manage **that** manually

DRUPAL IBERIA

dropsolid

So we built a platform

Today we'll be talking about language

Nouns

project
user
server
environment
membership
etc
organisation
backup
tenancy

Verbs

clone
delete
create
upgrade
rollback
etc
anonymize
archive
assign

Truth

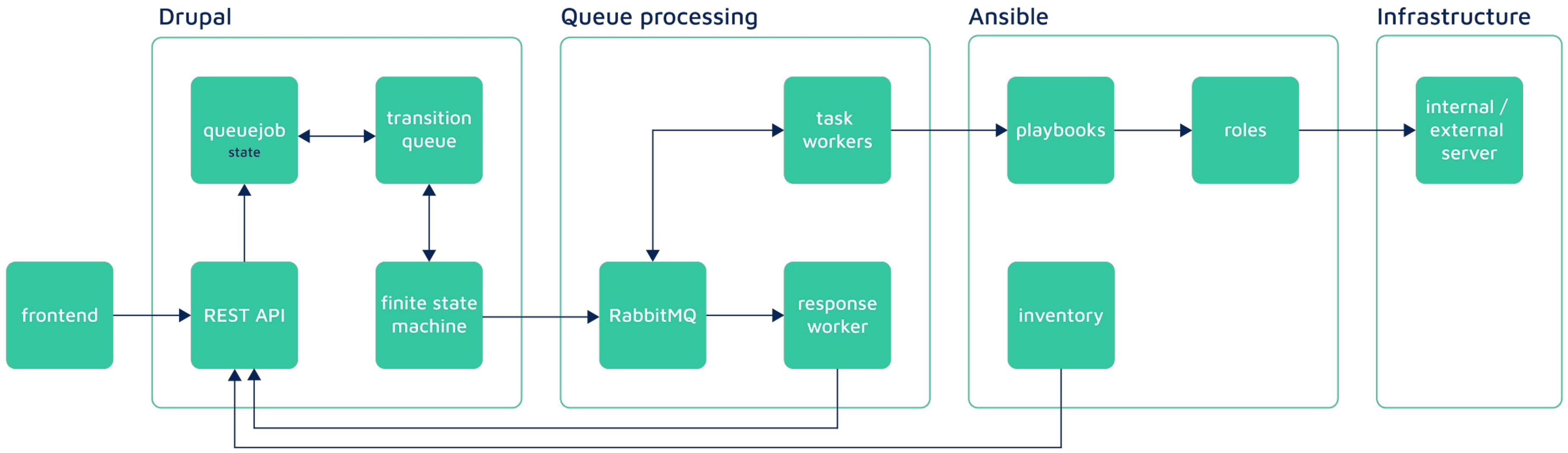
Truth the way

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

Action!


Drupal
Queue processing
Ansible
Infrastructure
frontend
REST API
queuejob
state
transition queue
finite state machine
RabbitMQ
task workers
response worker
playbooks
roles
inventory
internal / external server

It makes sense!

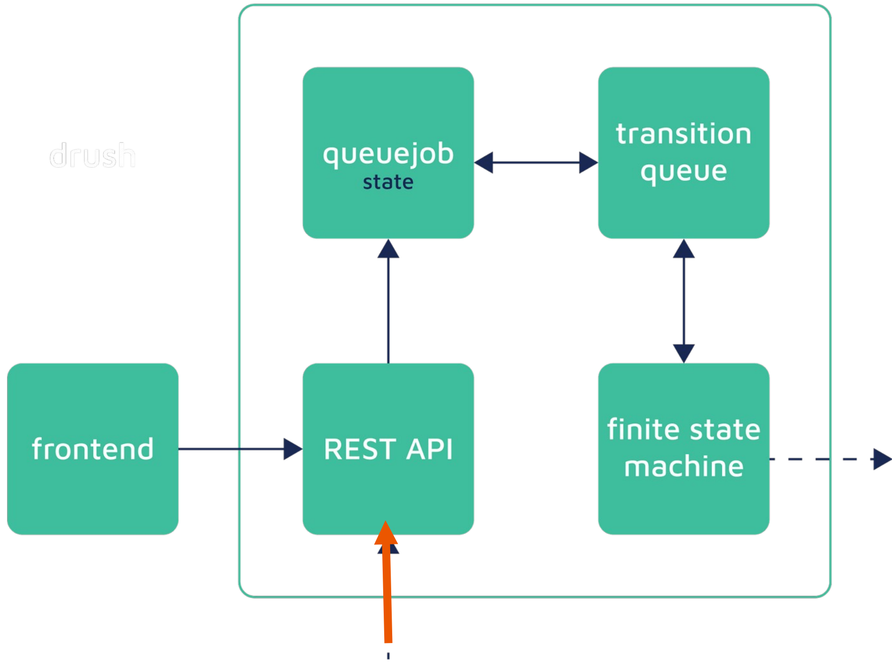
until...

DRUPAL IBERIA

Drupal is not an application platform

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

- High revision overhead
- Bad blob handling



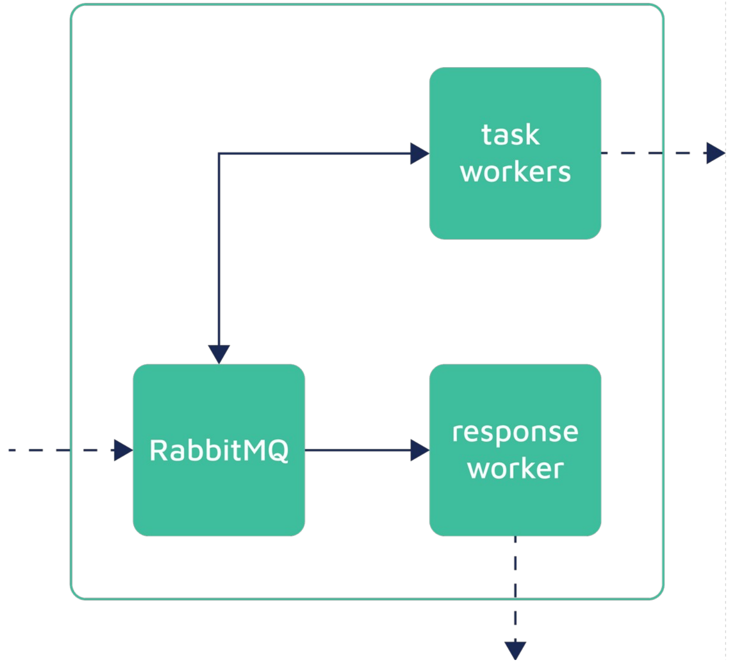
…lead to **deadlocks**

DRUPAL IBERIA

We defended with smarter queueing

Celery on top of **RabbitMQ**

Retries with smart backoff
Reliable interim state store

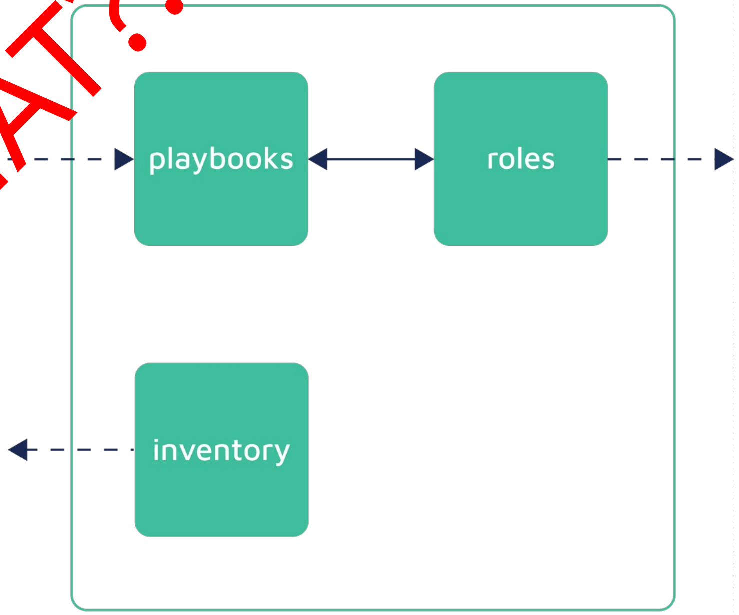


at-least-once delivery

While Ansible took care of most platform verbs

… yeah, we did it again! When all you have is a hammer…

Wait, WHAT?!



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

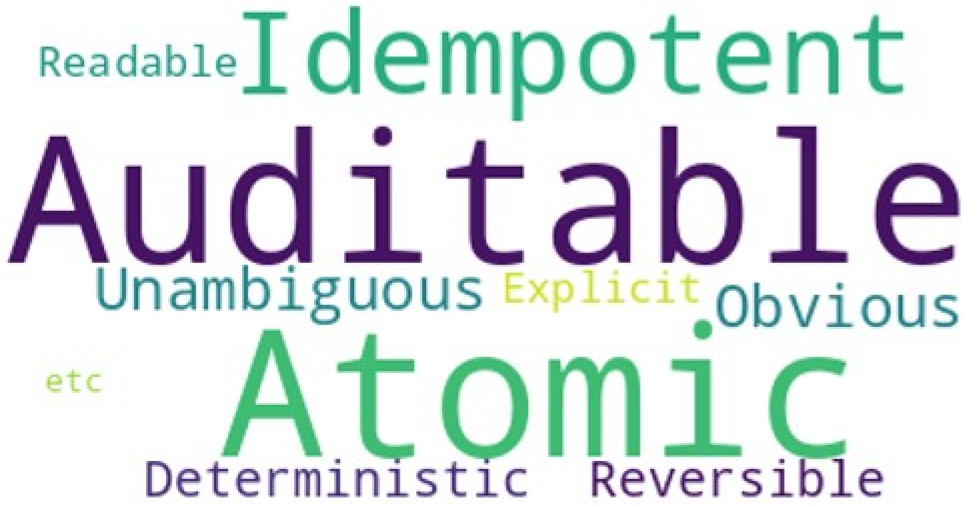
You can resume a workflow

Enables atomic “revert” options

We now have vocabulary, and a reliable grammar

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

These sentences should be



Many other things had to be handled differently as well!

For our customers’ benefit, for our platform’s performance, cost, security… and our own sanity

Backups

€€€ disk usage, difficult retention, too effort-intensive to operate and service - something had to give

Restic

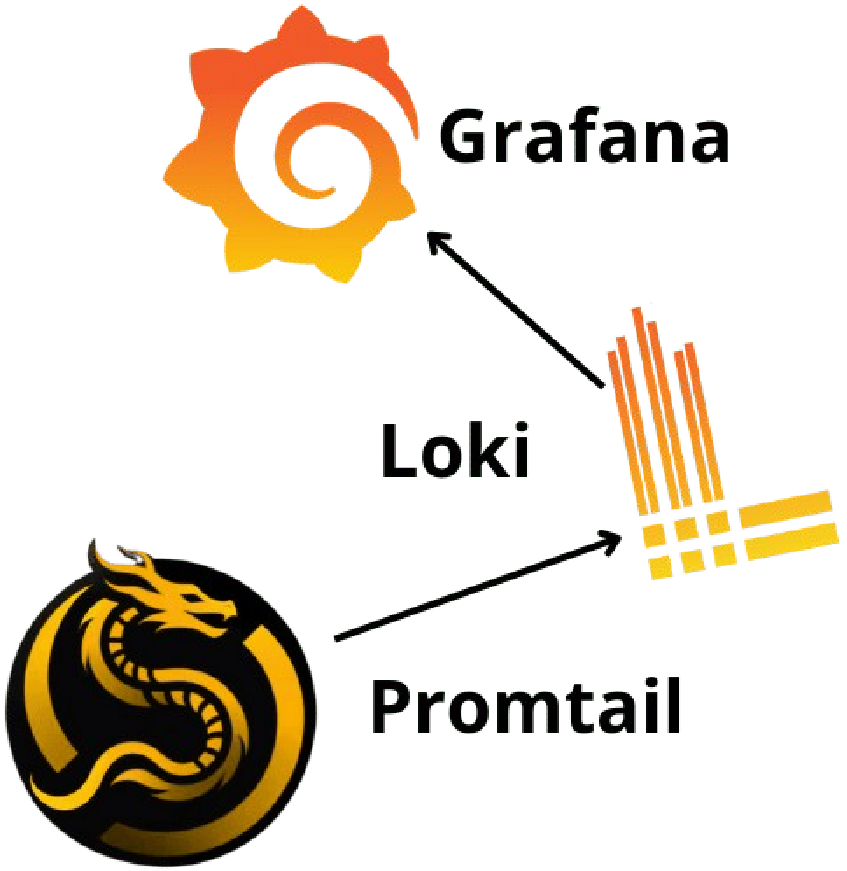
Backups done right!

It’s 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
- Take an early day - it Just Works!

Logging

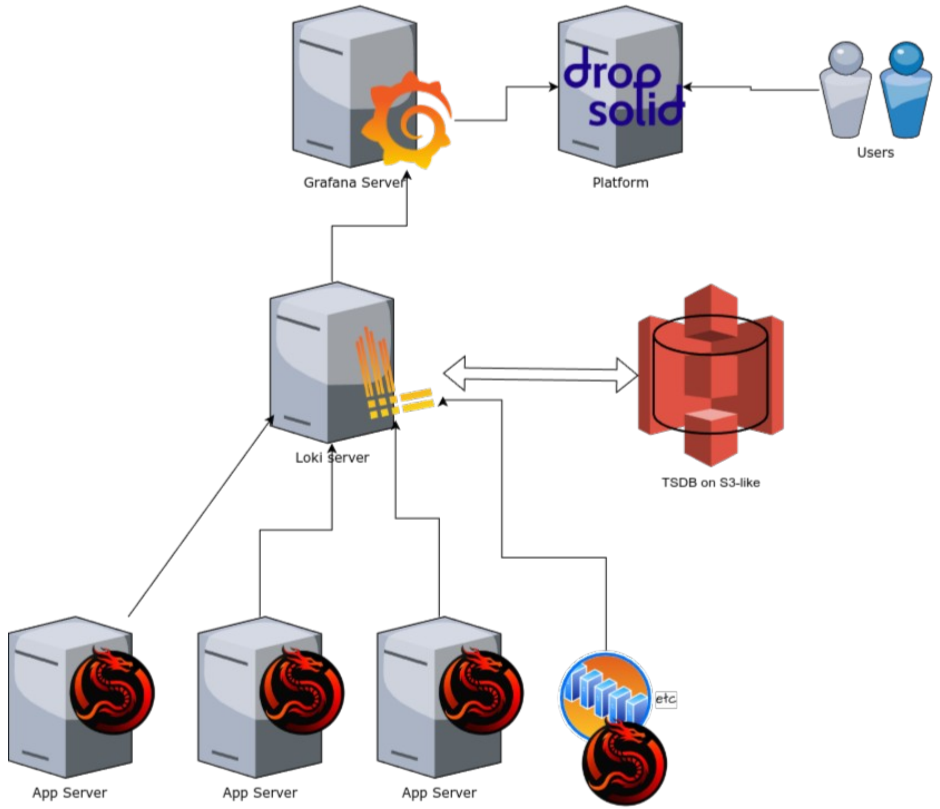
Shall we **not** have world+dog `ssh`ing into our servers to do `tail -f`? Or doing clunky `scp`s?



Logging stack

Promtail	An agent on each host that crawls logs and ships to Loki in real time
Loki	A log database using object store (TSDB) as a backend
Grafana	A front end providing (embeddable) views over Loki logs

A match made in … Asgard?



And as the growth continues...

Everything breaks at a scale

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

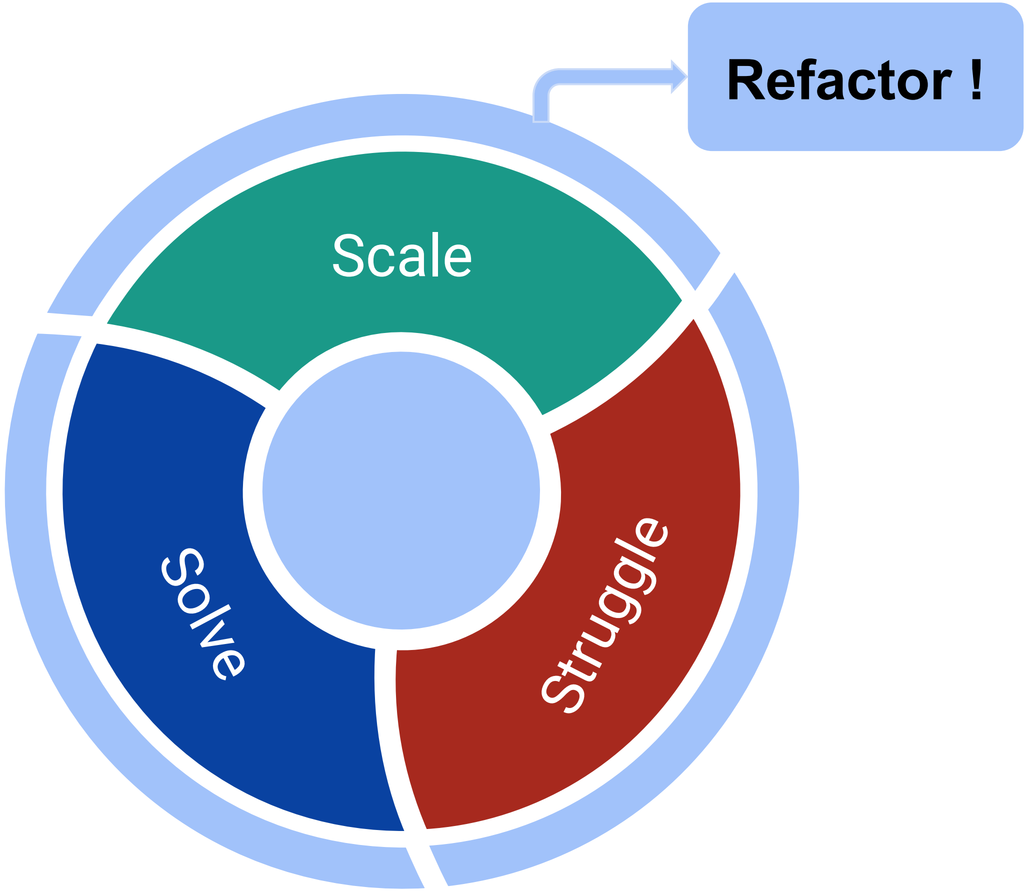
Many of our abstractions get leakier as we scale up

… 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**

It’s an iterative process

ssh
shell scripts
puppet
ansible
drupal app
docker
RabbitMQ
Celery
… kubernetes?



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

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

- Vocabulary
- Grammar
- Sentences

Maybe it rings a few bells...

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

Behaviour Driven Design scenarios, like in `behat`

Domain Driven Design's *ubiquitous language*

... coincidence?

This one simple trick (product managers hate 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”.

Why is that relevant?

Call it **Decoupling** and **Alignment**.

Good abstractions, good sentences, allow us to maintain and evolve our value delivery to our customers, while swapping out implementation and supporting infrastructure in whatever way necessary.

So 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…

So let’s keep talking about kubernetes

And let’s talk about **Verbs**

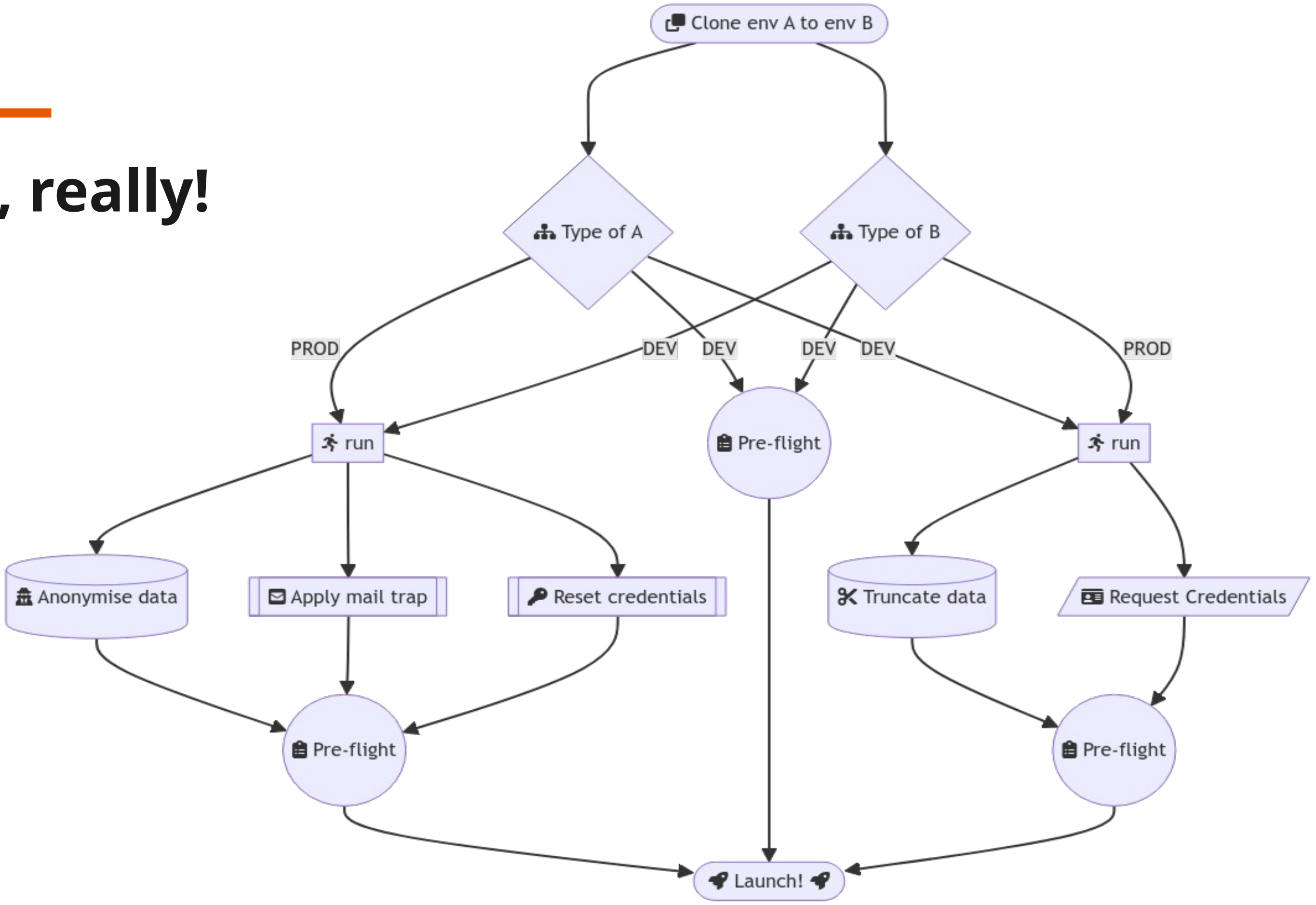
CRDs are fine, but what do do with them? The API only knows definitions, and we have many nouns to orchestrate!

Maybe we can learn from past experience

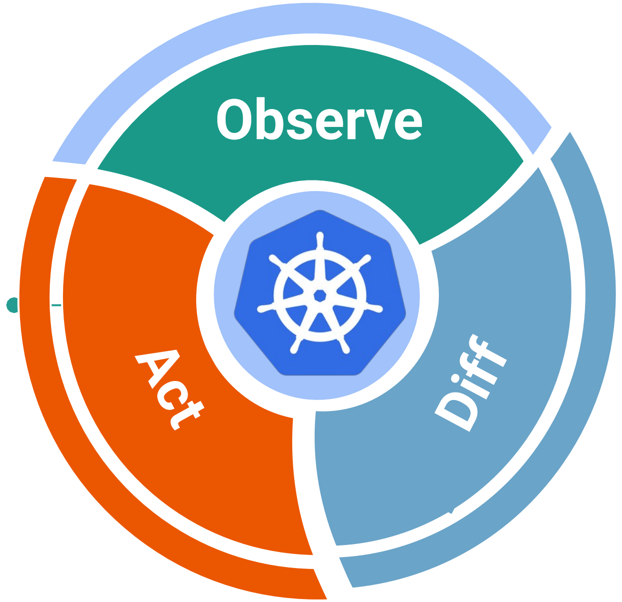
Remember the whole rant “Ansible is not a programming language or an application framework”?

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

Yes, really!


Clone env A to env B
Type of A
Type of B
PROD
DEV
DEV
DEV
DEV
PROD
run
Pre-flight
run
Anonymise data
Apply mail trap
Reset credentials
Truncate data
Request Credentials
Pre-flight
Pre-flight
Launch!

Kubernetes gives us operators



Domain-specific controllers that extends 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!

DRUPAL IBERIA

Thank you! Questions?