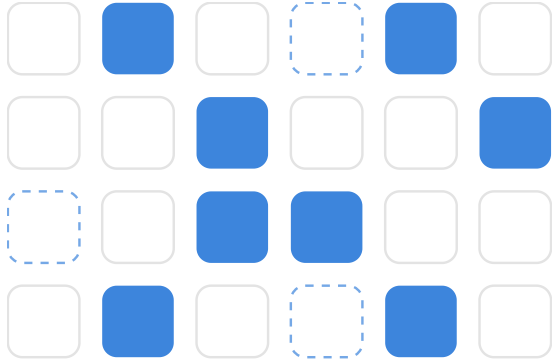
cecil.wöbker × remberg

CONTAINERDAYS HAMBURG — 3 SEPTEMBER 2026

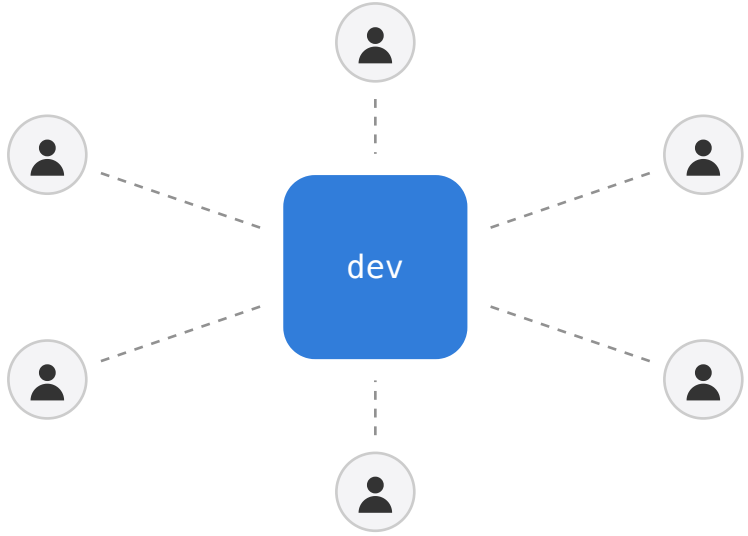
The Full Stack Preview

Kubernetes, ArgoCD, and ephemeral databases



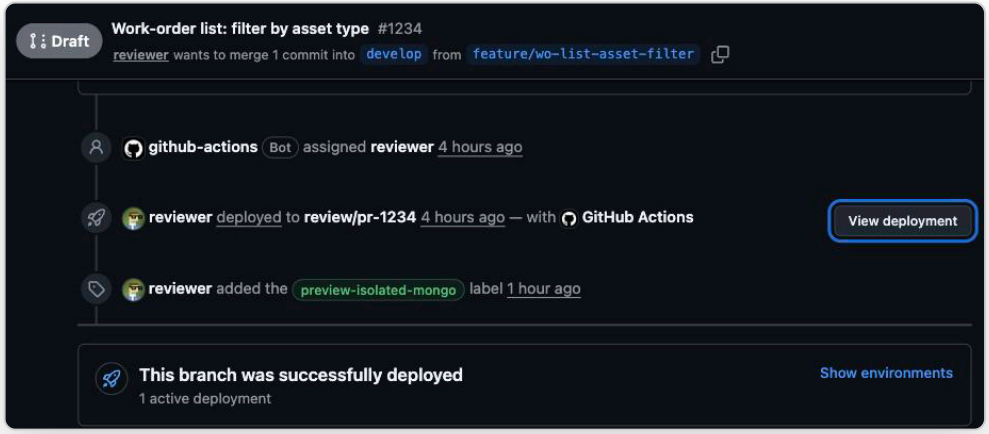
Cecil Wöbker
Co-Founder & CTO, remberg
@cwoebker

One dev environment, everyone in it



The promise

Every pull request gets its own environment. Reviewers click a link instead of pulling a branch.

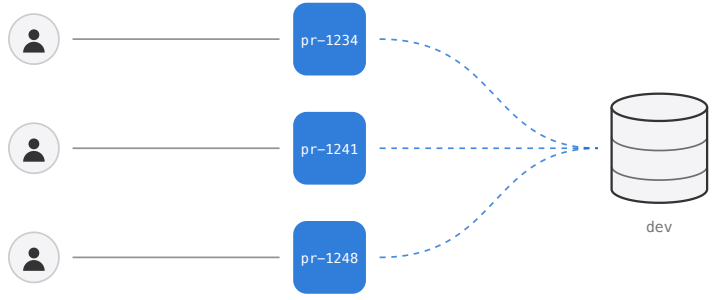


The whole reviewer workflow: one button on the pull request

Preview environments — v1

One Application per open PR: api, web, workers, ingress. Click a link, see the branch. It works.

```
MONGO_URL: mongodb+srv://dev.a1b2c.mongodb.net
```



600 pull requests a month

~70 open at any moment, climbing — writing code is not the constraint anymore but validation

~15 engineers. Merged pull requests per month, July and August 2026.

Deleting a file in a preview deletes it from dev.

— engineering velocity grows → side effects of shared state grow

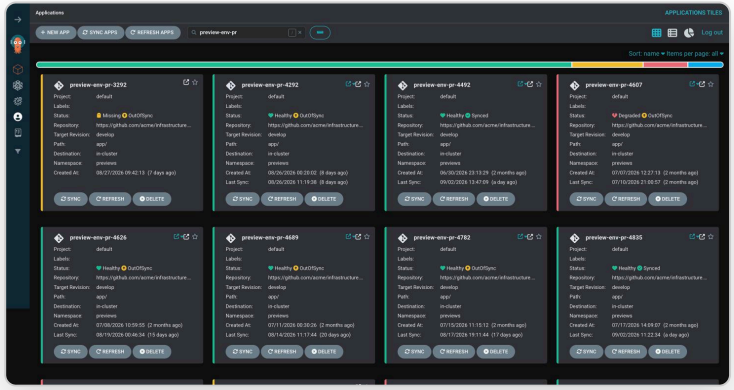
01

The Preview Environment

ApplicationSet, Pull Request generator

ArgoCD watches the repo and templates one Application per open PR. Automatically, no CI.

```yaml
generators:
  - pullRequest:
      github:
        owner: acme
        repo: platform
      requeueAfterSeconds: 120
```



ArgoCD — one Application per open pull request, 76 of them this morning

Open a PR, get an environment.

Close it, Argo prunes it.

What one PR provisions

```
previews/                one namespace, all previews
  preview-pr-1234        one Application per PR
    deployments          api, web, workers
    secrets              generated per release
    ingress              pr-1234.preview.internal
```

Teardown handled via Argo.

Full prune of the deployment.

ArgoCD network view — Ingress, one Service, and one Pod per workload

State is a list, not a database

Six managed services that need to be potentially isolated, with three possible answers: **isolate**, **namespace**, or **share**.

MongoDB ?

whose data does a preview write?

Redis ?

whose cache does it flush?

RabbitMQ ?

who consumes what it publishes?

S3 ?

whose files does it delete?

Outbound email ?

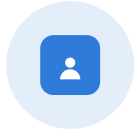
who receives what it sends?

Auth (SSO) ?

who can log in?

What we actually wanted

Isolation that goes all the way down.



One environment per PR

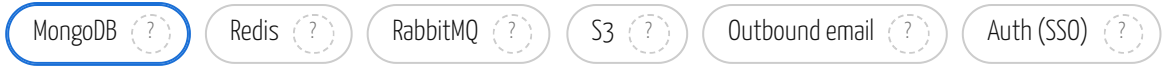
Its own database

A bill that follows use

← shift left: bugs surface **before** staging

02

The database


MongoDB ?
Redis ?
RabbitMQ ?
S3 ?
Outbound email ?
Auth (SSO) ?

One `mongod` per pull request

REJECTED

Managed cluster per preview
via a cloud operator, takes time to provision, & a SaaS dependency in the loop

Shared cluster, per-PR prefixes
application changes needed, prone to errors

BUILT

Single-member replica set

An Atlas-local image

Running data migrations

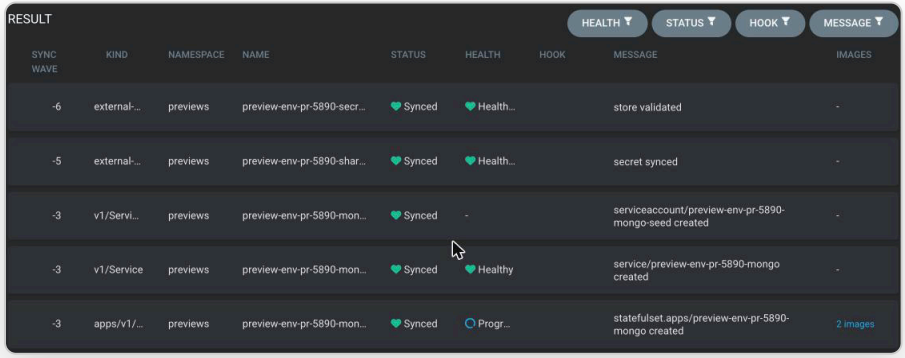
Re-generating **search indexes**

App unchanged: only `MONGO_URL` differs

Ship it dark

🚩 `PREVIEW_ISOLATION_ENABLED=true`

The seed pipeline



Sync waves: secrets at –6 and –5, the database at –3 — nothing else starts until the StatefulSet is healthy, and healthy means seeded

Dump

nightly CronJob, dev → S3, three-day expiry

Restore

in the mongod pod, readiness gated on a seed marker

Migrate

a PostSync job runs the PR's own migration image

Index

search indexes built explicitly

The seed was carrying dead weight

159,000 orphaned documents swept

12 of 20 GB left out of the seed

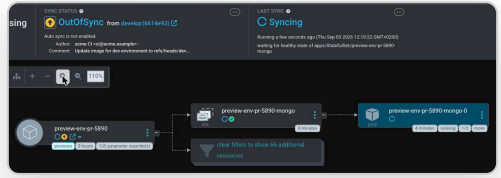
60 GB of dangling files found in S3

Nobody looks at a database that "just works".
It's nice to clean it up every once in a while so everyone profits.

Startup Measurements

STAGE	2 JULY	3 SEP
mongod to Ready	36 s	1 s
fetch dump from S3	41 s	28 s
`mongorestore`	**826 s**	**351 s**
migrate job: delta, flush, indexes	68 s	16 s

993 seconds, 83 % of it one command. After the diet: **396**.



Mid-sync: the mongod pod at 1/2 while the seed sidecar restores — every later wave waits

Storage that dies with the pod

REJECTED

A PVC per PR — created by the StatefulSet, never tracked by ArgoCD: an orphaned EBS volume per closed PR

EBS is AZ-bound — a rescheduled pod trades data loss for a scheduling failure

BUILT

`emptyDir` — the data lives exactly as long as the pod

The pod **reseeds** itself on start (losing a pod costs a few minutes)

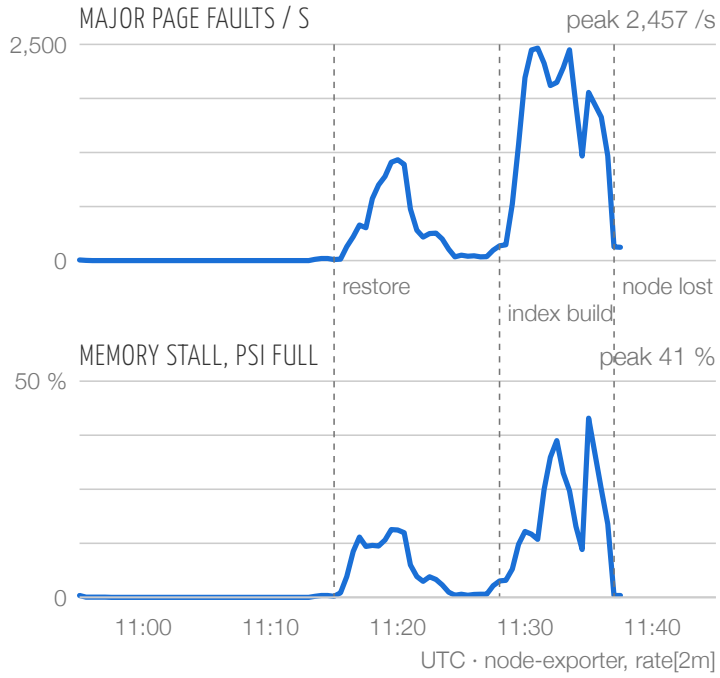
Nothing to bill, nothing to reap


MongoDB
ISOLATE
Redis
?
RabbitMQ
?
S3
?
Outbound email
?
Auth (SSO)
?

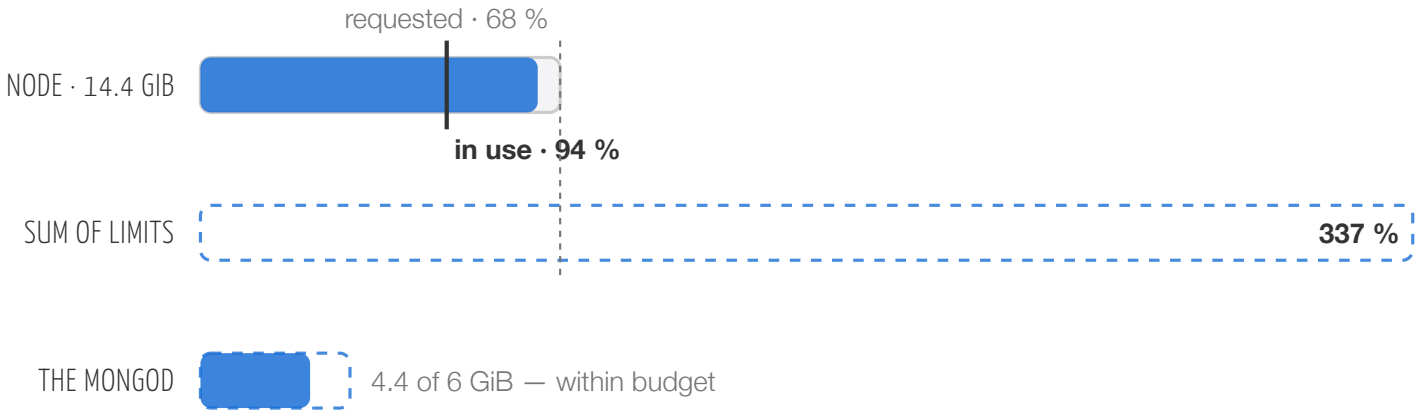
03

What broke

A preview killed its node



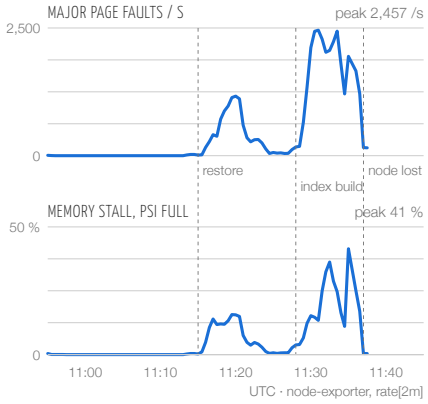
The container stayed within budget. The node died.

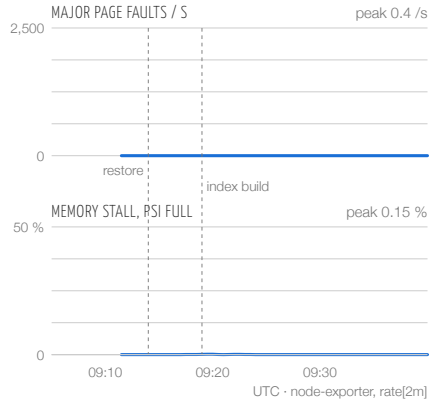


Limits govern a container's memory.

Nothing governs its appetite for page cache.

Give state its own nodes — then measure





	JULY, A SHARED 16 GIB NODE	AUG, THE DEDICATED POOL
major page faults	2,457 /s	**0.4 /s**
memory stall, PSI full	41 %	**0.15 %**
the node	50 pods, 337 % of limits	**10 Gi reserved per mongod, several per node**

04

The Rest

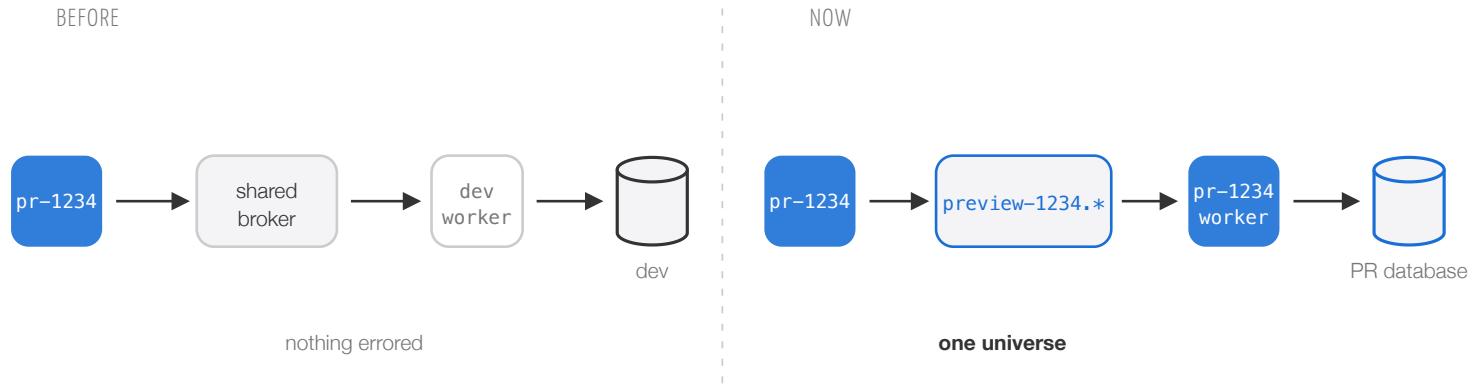

MongoDB
ISOLATE
Redis
?
RabbitMQ
?
S3
?
Outbound email
?
Auth (SSO)
?

Redis: namespace, don't isolate

The client library adds the prefix. Isolation happens transparently for everyone.

MongoDB ISOLATE | Redis NAMESPACE | RabbitMQ ? | S3 ? | Outbound email ? | Auth (SSO) ?

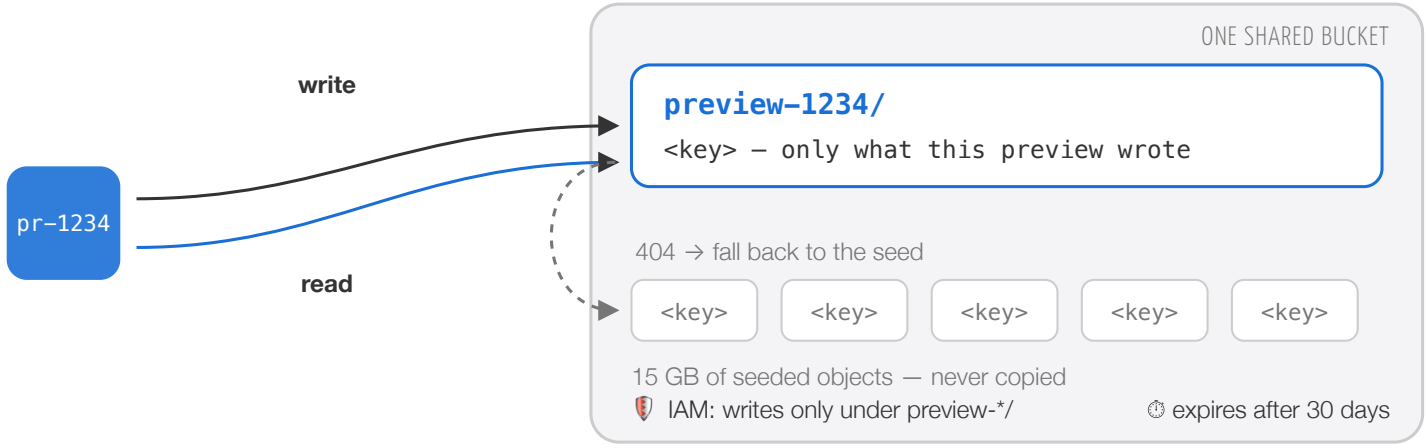
RabbitMQ: the messages went somewhere



No errors. The workload simply happened in the **wrong universe**.

MongoDB ISOLATE | Redis NAMESPACE | RabbitMQ NAMESPACE | S3 ? | Outbound email ? | Auth (SSO) ?

S3: copy-on-write with a key prefix



Fifteen gigabytes never copied — one `HeadObject` per read.

MongoDB ISOLATE · Redis NAMESPACE · RabbitMQ NAMESPACE · **S3 NAMESPACE** · Outbound email ? · Auth (SSO) ?

The two deliberate shares

SHARE

Real mail gets send

SHARE

Logging in is enough for us

Sharing is fine when it is a real decision instead of a default.

MongoDB ISOLATE | Redis NAMESPACE | RabbitMQ NAMESPACE | S3 NAMESPACE | Outbound email SHARE | Auth (SSO) SHARE

The scorecard

MongoDB	ISOLATE	own mongod & nightly seed
Redis	NAMESPACE	shared cluster & prefixed keys
RabbitMQ	NAMESPACE	per-preview exchanges, per-preview workers
S3	NAMESPACE	shared bucket, prefixed writes, reads fall-back
Outbound email	SHARE	dev's sender domain — previews can send real mail
Auth (SSO)	SHARE	mostly read-only — login is the test

05

The bill

2.40 € per pull request

a tenth of a shared database node | lives as long as the PR

† ≈0.11 €/h per database = r7g.2xlarge on-demand eu-central-1 ($0.5168/h) + 200 GiB gp3 at 6,000 IOPS / 600 MiB/s (≈$0.08/h), shared five ways, at 0.92 €/$; ≈0.16 €/h when fragmented two to an xlarge ($0.2584/h). × 18.3 h mean PR lifetime (last 100 merged, Sep 2026) ≈ 2.00–2.90 €. Median PR (5.3 h): ≈0.60–0.85 €.

What money decided

On-demand, not spot

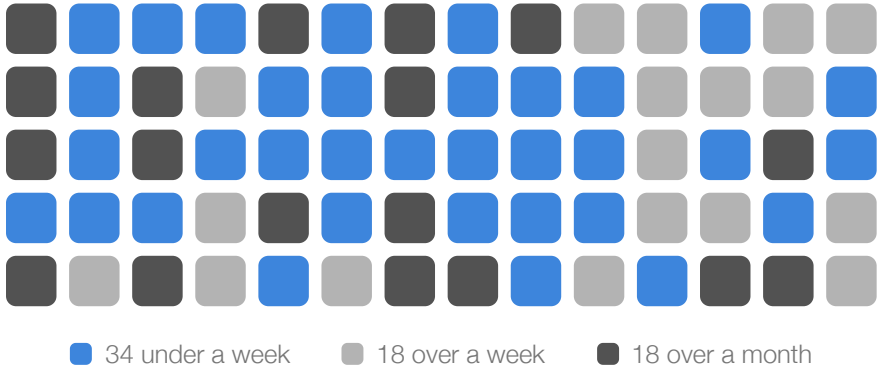
emptyDir — nothing to bill, nothing to reap

Honest reservations — several databases share a node

Idle previews shut down — 14 days without activity, database included

Merging fast is cheap. *Lingering* is not.

Why the default hasn’t flipped yet



95 % of the preview-hours belong to the **36 older than a week**.

Idle previews shut down, database included.

† 70 open PRs on 3 Sep 2026, drafts included. At 0.11–0.16 €/h, 70 always-on previews ≈ 6,500 €/month — 95 % of it on idle PRs.

What this was really about

Engineering velocity

600 PRs a month, 15 engineers

AI parallelises the work

ideas become real in hours
validation became the bottleneck

Validation that keeps pace

an environment that tells the truth
@ 2.40 €

State is not just the database — **decide every row**.

thanks.

@cwoebker

cwoebker.com/talks/the-full-stack-preview