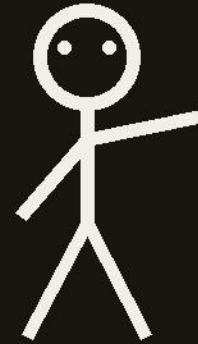
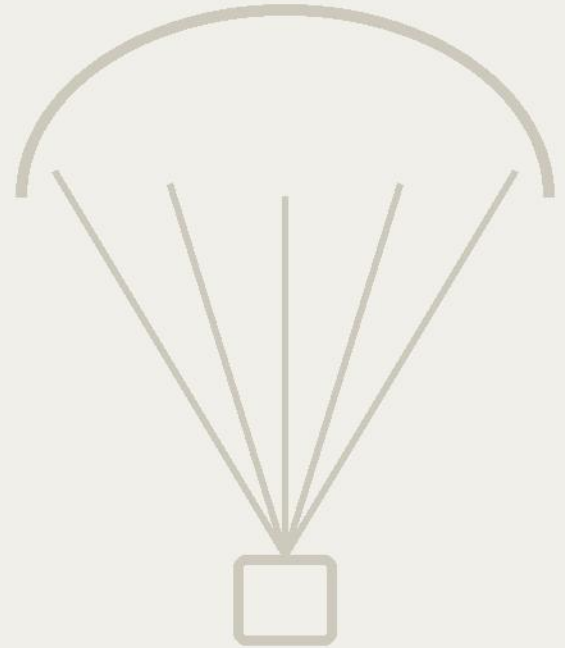


ONE FLAG, NO DEPLOY

Forcing your own outage on a platform of
one



A backup you have
never forced is a
backup you have
never seen work.



never opened

Mine had never been seen to take over.

On 24 September I pulled the plug. On the 25th I filmed it.

ON THE WAY TO THE RECORDING



- 1** A flag read on every request,
without slowing anything down.



- 2** A middle setting
that watches only my own account.



- 3** Why the backup waits
for four failures, not one.



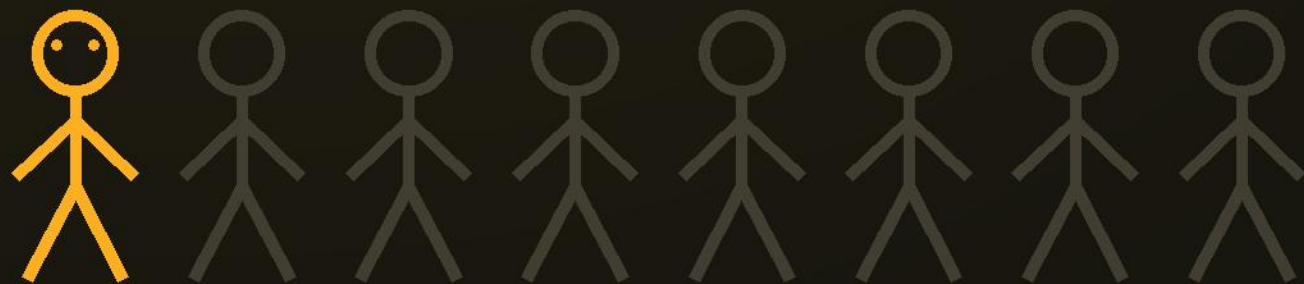
- 4** The test itself.
What it caught, and what I changed.

EVERYTHING ELSE IS THE TWO SWITCHES THAT MADE IT POSSIBLE.

THE SCALE, HONESTLY

Two production apps.

ONE ENGINEER.



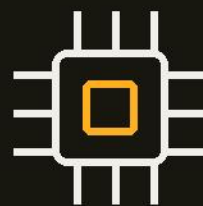
No platform team. No on-call rota. No second pair of eyes.



workers



postgres



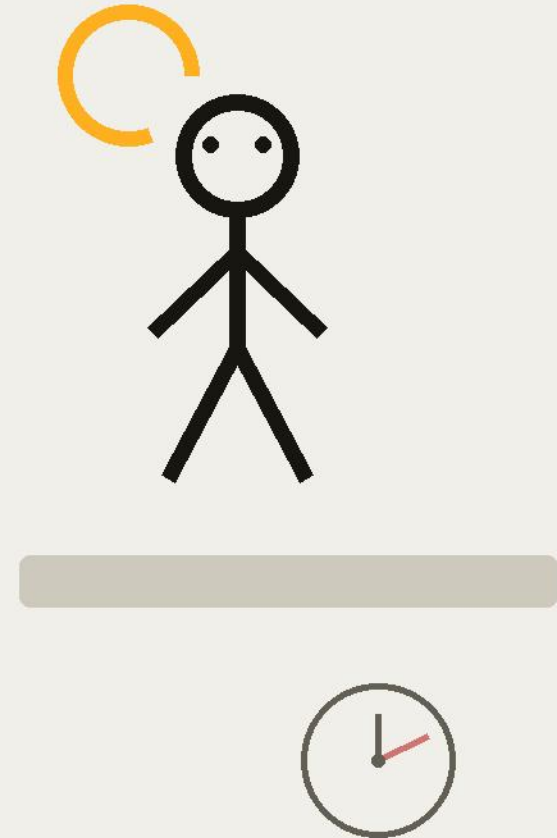
the model

THE CONSTRAINT

A deploy takes minutes.

Minutes is fine on a Tuesday afternoon.

It is not fine while a user
watches a spinner.



AND THE OTHER HALF

The trace holds the user's own answers
and the script written for them.

WORD FOR WORD.

Leaving that at rest because it might be useful one day is
not a decision.



so

Diagnostics I can raise and lower in seconds, from a laptop, without shipping anything.

That is a flag.

Not a feature flag. Nobody is shown a new button. The same mechanism, pointed at a different problem.



One key in a store. No deploy.



off

nobody.
the default, and what
a failed read means



admin

only me.



all

everybody, briefly,
on purpose.



Traces expire after 24 hours.

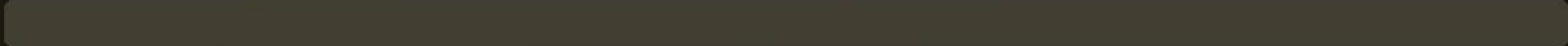
```
$ wrangler kv key put --binding STATUS_KV --remote \  
  trace_mode admin
```

No build. No deploy. Live in under a minute.

EVERY REQUEST READS IT

A flag read at start-up is not a switch.
It is a deploy with extra steps.

the KV read  cached at the edge

the model call  seconds

KV is built for high-volume reads. Next to a language model, the read is negligible.

THE TRADE-OFF

KV is eventually consistent.

Change the flag, and another location can still see the old value for a while.

Up to 60 seconds
to reach every edge.

A cheap switch with some delay beats turning a debugging flag into another deployment.



60 s

admin is the state that
stops you doing something
at midnight you would
not do at midday.

Two states make every question a choice between blind
and reckless.



00:00

TWO DETAILS

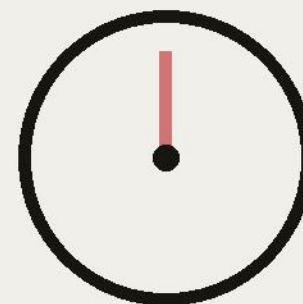
A read failure means off.

No throw. No retry. One log line, and carry on. You can lose the diagnostics. Never the request.



Traces expire after 24 hours.

I will forget one day. Everyone forgets. The expiry means forgetting costs a day.



24 h

NOT ALL LOGGING BELONGS BEHIND THE SWITCH

ALWAYS ON

- which provider answered
- how many tokens
- that a retry happened

no user content

BEHIND THE SWITCH



- the prompts
- the replies
- the finished script

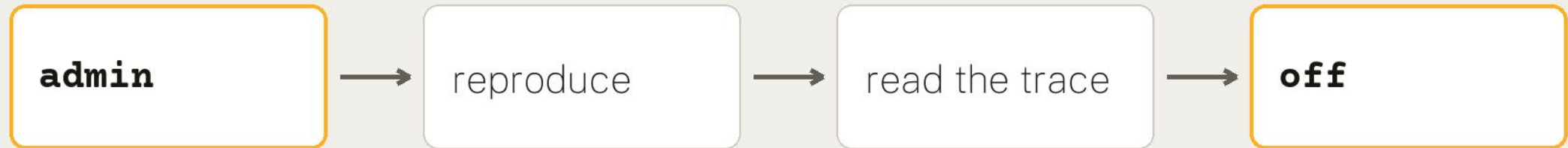
private

If your flag is hiding your operational logs, it is on the **wrong** door.

THE STATE I ACTUALLY USE

Not broken. Wrong.

No error, no operational log, because the thing worked. You need what went in and what came out.



Minutes. And at no point did I look at **anyone else's** writing.

WHAT IT COSTS

Admin only helps when I can reproduce the problem.

If the fault lives in one user's data, only all would help. And all records everyone's generations while it is on.

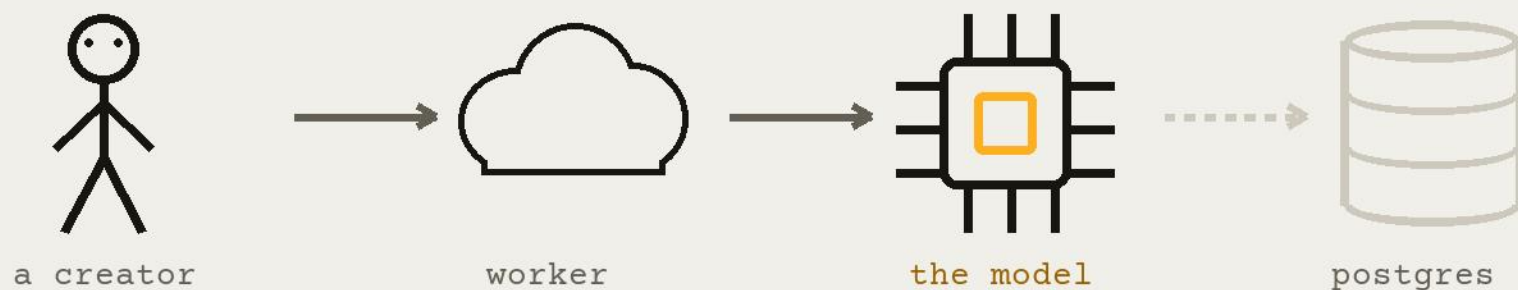
Not needed yet. If ever: minutes, the 24 hour expiry, a note in a file, and I tell them.

A third state does not remove the hard case.
It takes the easy case out of the hard pile.



SWITCH TWO · THE WRITER

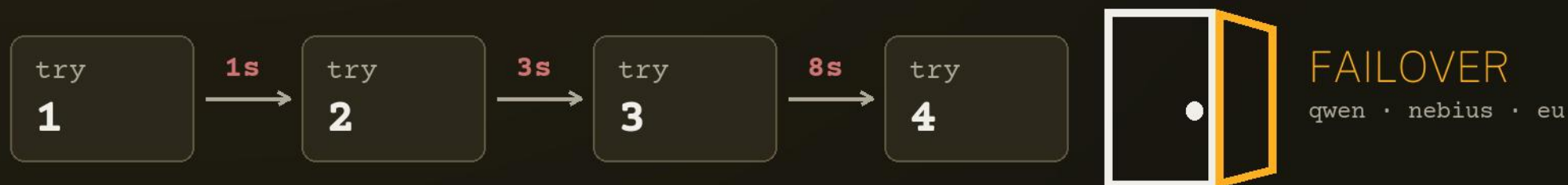
One writer. Somebody else's API.



If Anthropic is unavailable, nobody gets a script. That is the whole product, stopped, because somebody else's API is having an afternoon.

THE LADDER

Four attempts, then the door.



Jitter a quarter either side, so concurrent requests do not hit a recovering API on the same beat.

MOST FAILURES ARE NOT OUTAGES. FAIL OVER ON THE FIRST ONE AND YOU LIVE ON THE FALLBACK.

THE EXCEPTION THAT NEARLY WRECKED THE TEST

A definite 400 is **never** retried.

Bad request. Bad key. That is my bug, not their outage.

That exception matters more than it looks.



never reached

THE FALLBACK

Qwen3 235B on Nebius. EU.



anthropic · us



nebius · eu



03:00 nightly ping
40 tokens, real path

Different model.

Different company.

Different cloud.

A FALLBACK IN ANOTHER REGION OF THE SAME PROVIDER IS A FALLBACK FOR A DATA CENTRE FIRE.

CHOSEN BY MEASUREMENT, NOT BY LEADERBOARD

Qwen3 235B



5 to 8s · three tenths of a penny · every broken rule visible



Kimi K3



10 to 15 min · 40x the cost · invented a football scoreline

DeepSeek / GLM / MiniMax



70 to 90s on hidden reasoning, or nothing at all

Slow is annoying. Confidently wrong is **disqualifying**.

QWEN IS NOT AS GOOD AS SONNET

And I do not pretend otherwise.

- ✗ It parrots example phrases out of the prompt.
- ✗ It sharpens loose numbers into precise ones.
- ✗ Its endings are softer.



Degraded but alive.

A stand-in for an outage, not a replacement. Knowing exactly how it is worse is the point of measuring it.

IT TELLS ME, AND IT CHECKS

A log line

provider, model, tokens, and why the primary failed

One email

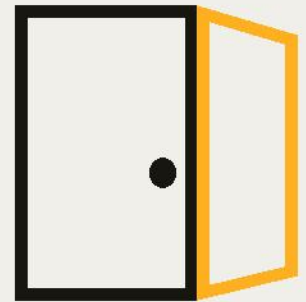
throttled to one an hour

Every night at 3

forty tokens through the real failover path, under a penny a year

It proves the door opens.

Not that anything is behind it.



SO HOW DO YOU FORCE AN OUTAGE

Bad API key?



401 is on the never-retry list. The failover is never reached.

Edit the code, record, revert?



Then the talk describes a system that did not exist.

A THIRD SWITCH.

Off by default. Every attempt treated as an overload, no request made.



THE OUTAGE IS THE ONLY THING THAT IS FAKED. EVERYTHING AFTER IT IS REAL.

WHAT CHANGES AFTER THIS

One failover in a script should count for the **whole** script.



The next script still tries Anthropic fresh, so a recovery is never missed.

78 seconds becomes something closer to **40**.

FIVE RULES

-  **1** Read the flag per request. A failed read means off.
-  **2** Three states, not two.
-  **3** Write down every flip, and when you turned it back.
-  **4** Count your calls. One failover speaks for the request.
-  **5** Put a gate downstream of every fallback.

None of this makes the system reliable.

It makes it **legible**.

THE PART I OWE YOU

When the writer fails, my users still get a worse script than they came for.

The flag tells me it happened, keeps the product alive, and lets me see inside without shipping anything.



Next

Qwen as the main writer, Sonnet as the backup

The risk

a switch that can degrade the writer for everyone if I leave it on

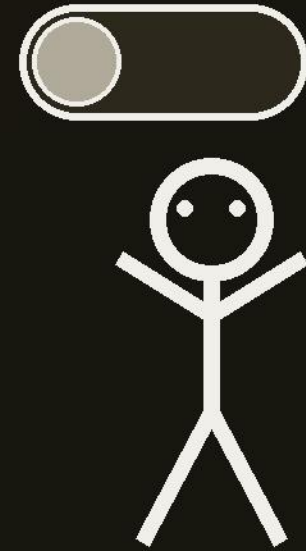
My work is not done.

THANK YOU

Fuad Laguda

linkedin.com/in/fuad-laguda-13ba217b

dev.to/lagudafuad



THE LOG SAYS FORCED

```
[model-retry] writer attempt 1/4: HTTP 529 overloaded_error  
      (FORCED by writer_fault) - retrying
```

A log that cannot tell the difference between an outage you suffered and one you caused will lie to somebody eventually.

Probably you. In six months.



WHAT ACTUALLY HAPPENED

It is not one model call.

IT IS FOUR.



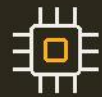
probe

4 tries · 12s
then handover



writer

4 tries · 12s
then handover



correction

4 tries · 12s
then handover



director

4 tries · 12s
then handover

Forty eight seconds of waiting. Not twelve.

78.4 s END TO END. 28.8 s ON A NORMAL RUN.

THE BIT I DID NOT EXPECT

The fallback broke one of my rules.

```
content gates failed, correcting:
```

```
punctuation=true
```



The gate caught it. The correction pass fixed it.

A flag that routes users to a worse model is only safe if something downstream is checking.

THE NUMBERS

	FORCED	NORMAL
End to end	78.4 s	28.8 s
Cost a script	\$0.004	\$0.060
Admin emails	1	0
Band 300-750 words	300	375



The expensive model makes things up too.
It just makes up different things.

1 sent
3 suppressed