Transform your development workflow with thin cloning

Nick Meyer @ Academia.edu

PGConf Europe 2025, Riga Latvia

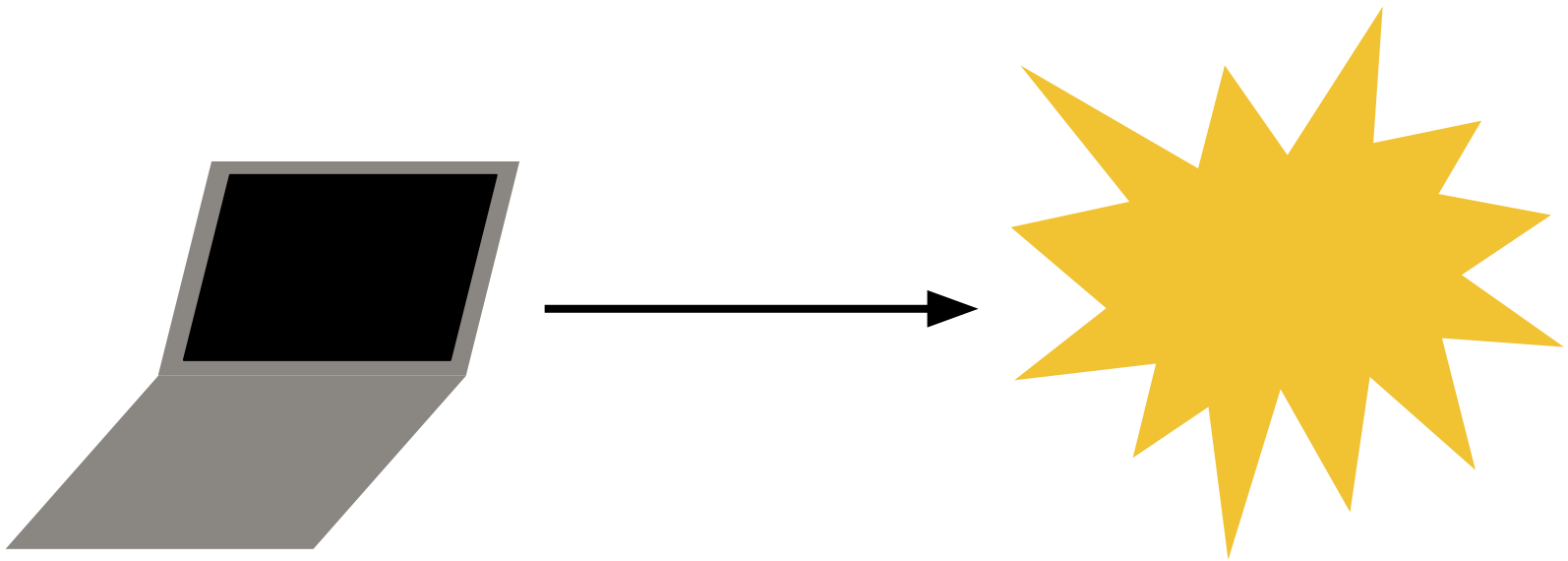
Transform your development workflow with thin cloning

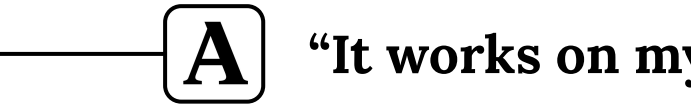
Nick Meyer @ Academia.edu

PGConf Europe 2025, Riga Latvia

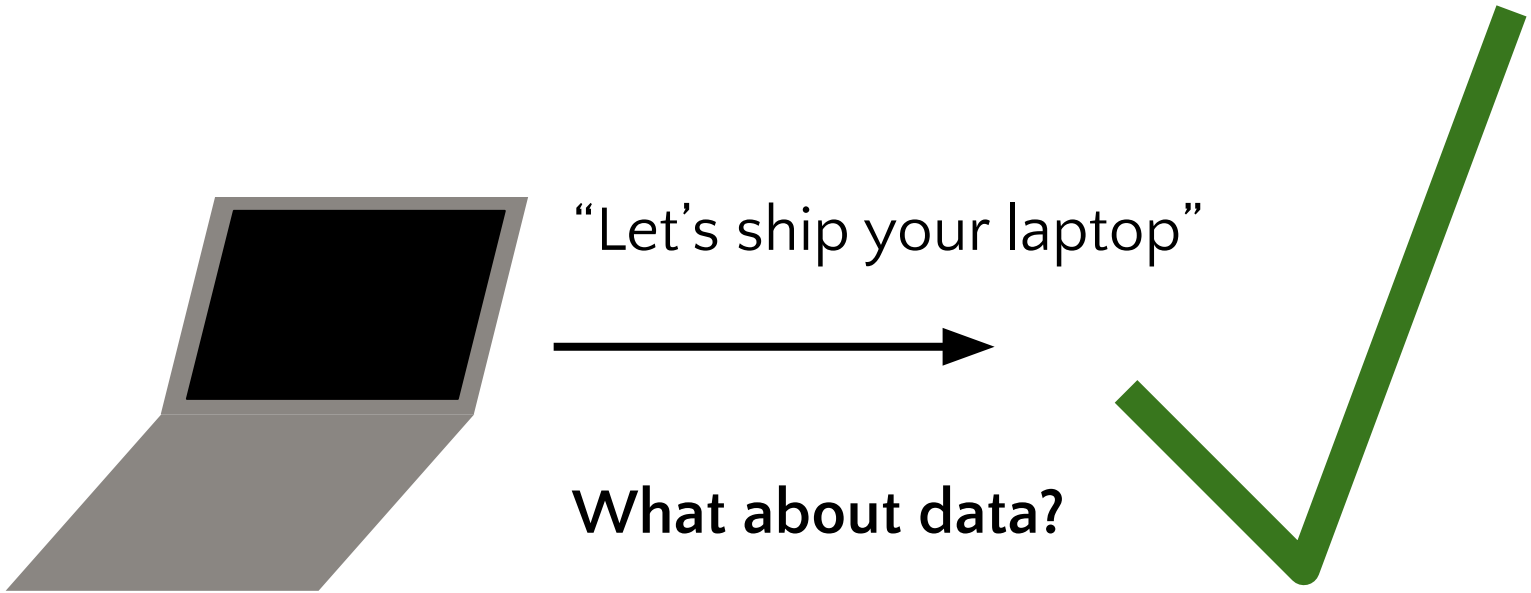
A

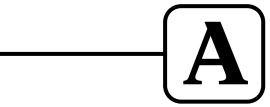
“It works on my laptop”





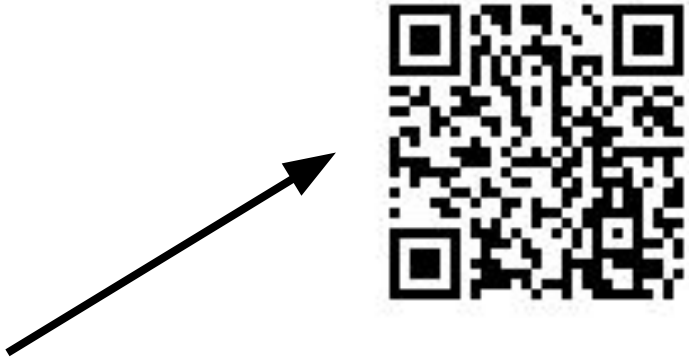
“It works on my laptop” and on prod





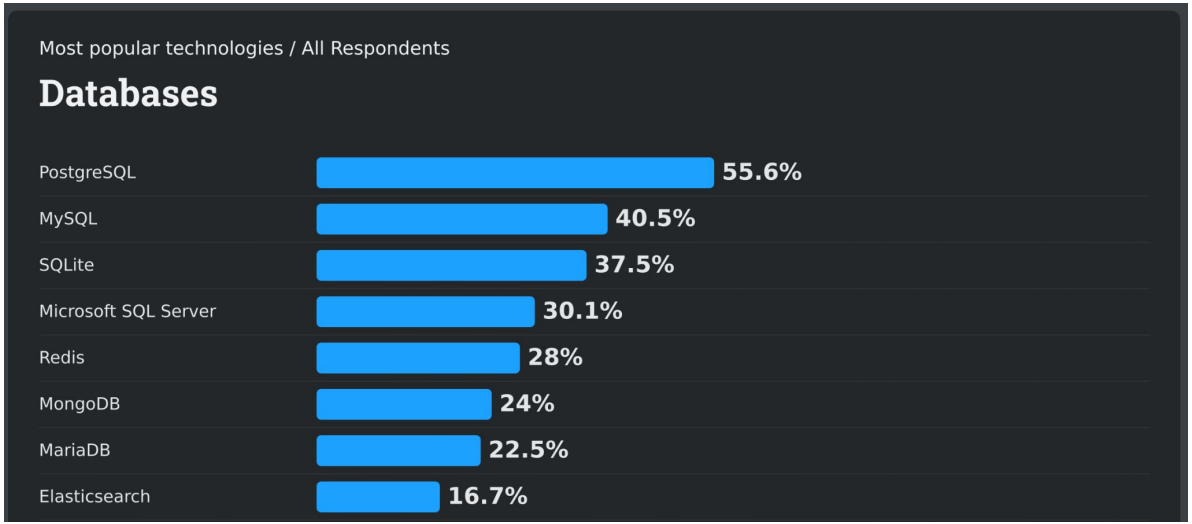
A bit about me (Nick Meyer)

- **Team lead of Platform Engineering**
- **@** Academia.edu
  - San Francisco, CA, USA
  - Accelerating the world's research
  - Open access
- https://github.com/aristocrates



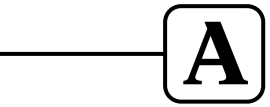
Slides: https://github.com/aristocrates/pgconf_eu_2025_talk

Happier devs move faster



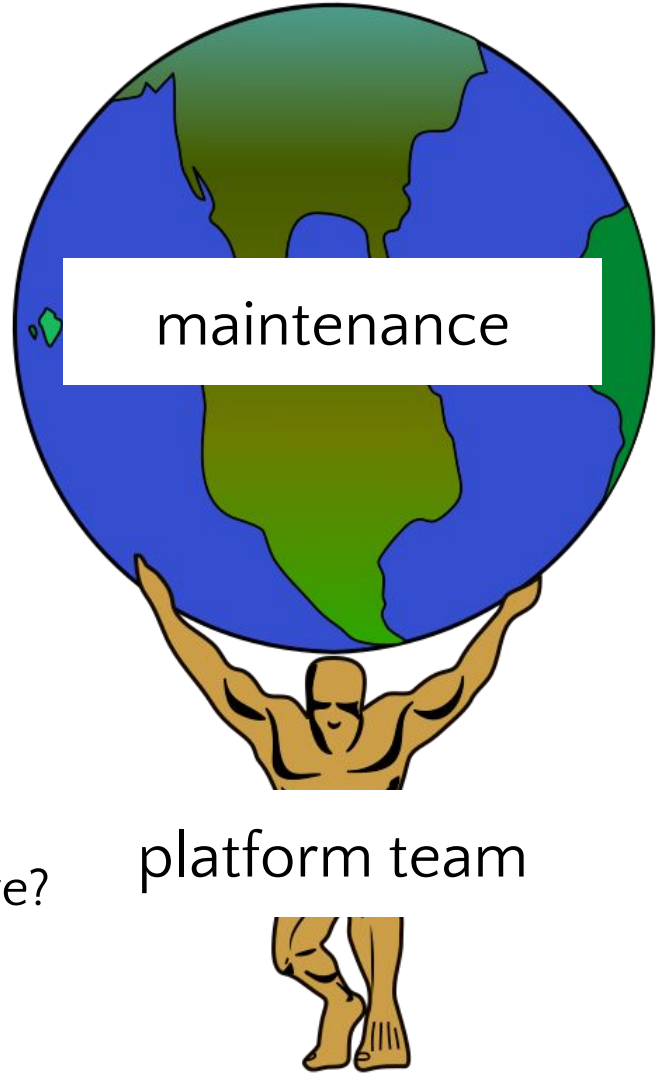
https://survey.stackoverflow.co/2025/technology#1-databases

The dev env struggle



Development data difficulties

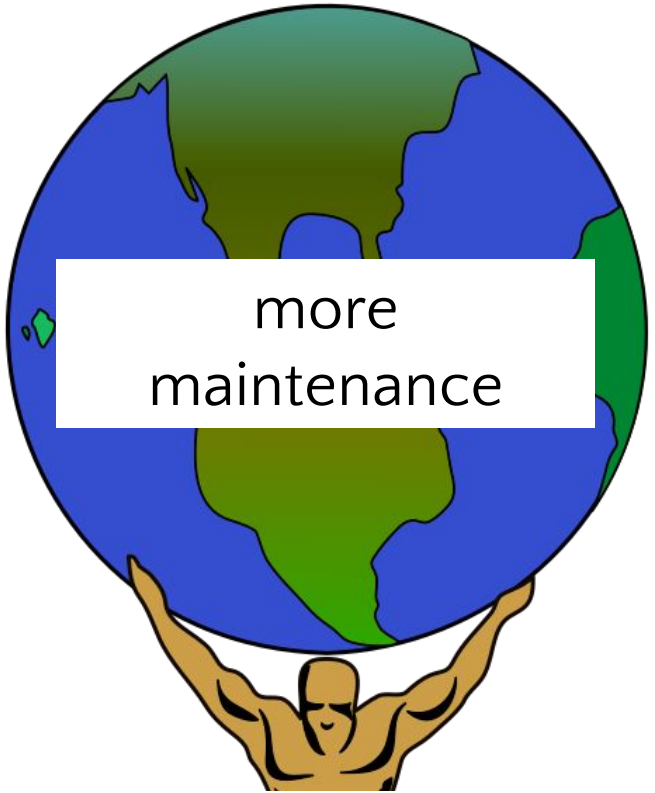
- **Shared vs many DBs**
- **Schema drift**
- **Realistic data**
  - Factories/fixtures
  - Subsetting + anonymization
- **Scale**
  - 10 TB vs 1 MB
  - How many uploads does one user have?



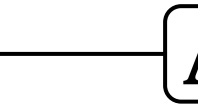
A Development data difficulties

To test any of [Team 1]'s changes:

- ✓ **Download some file from S3**
- ✓ **Set some ENV_VAR=1**
- ✓ **Run these 20 commands**



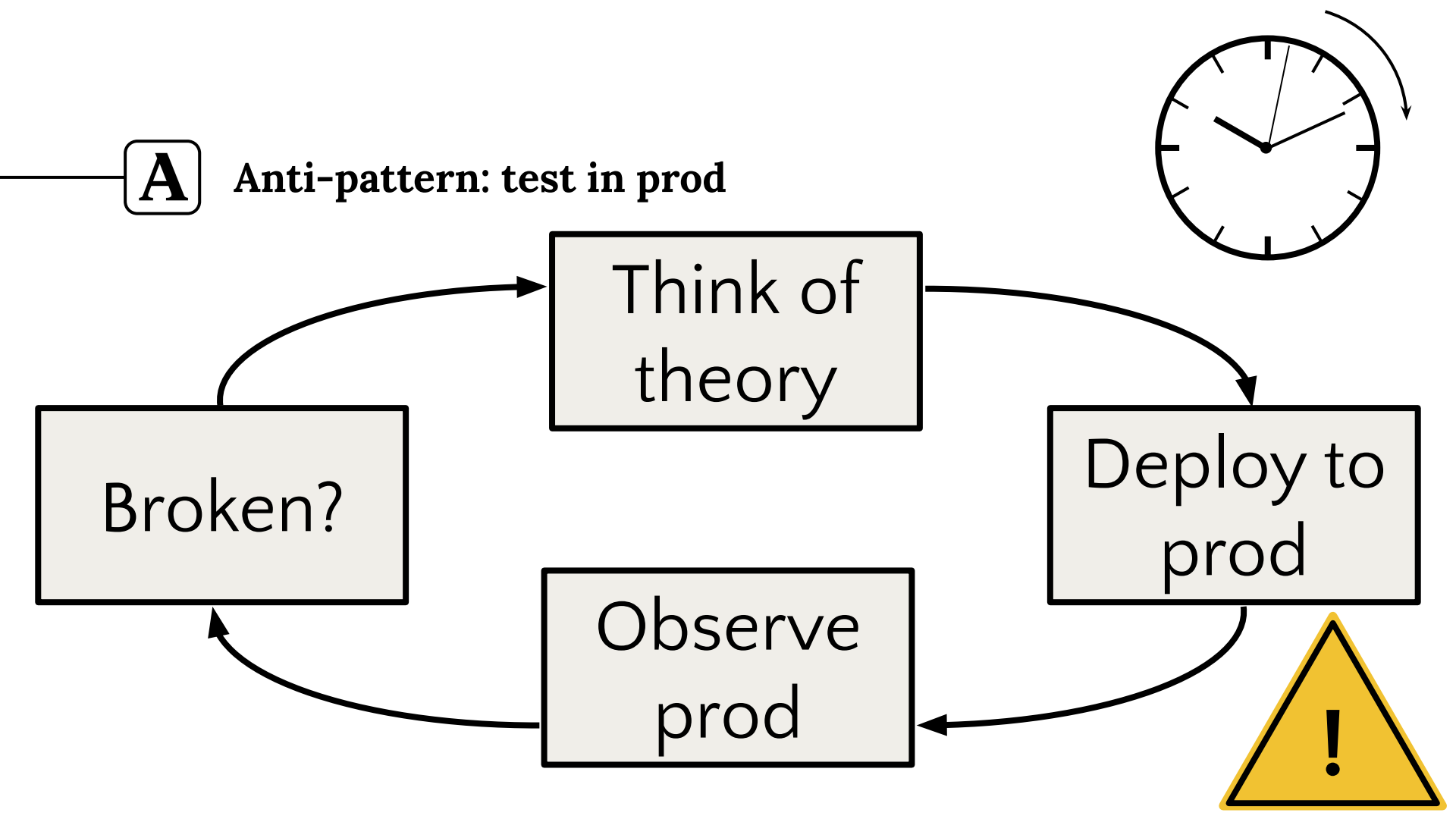
platform team



Staging environment

- **Cost (cutting)**
  - “Intentional differences” with prod
- **Shared DB?**
- **When it breaks, who fixes it?**

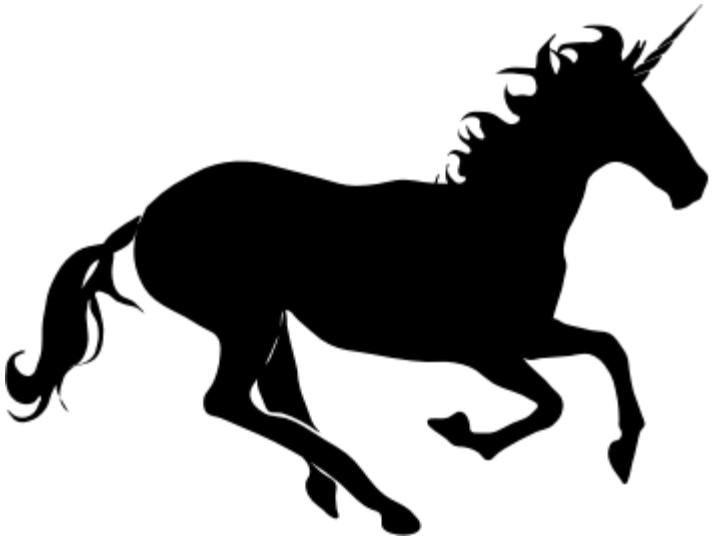
platform team


A
Anti-pattern: test in prod
Think of theory
Deploy to prod
Observe prod
Broken?
!

What do we want?

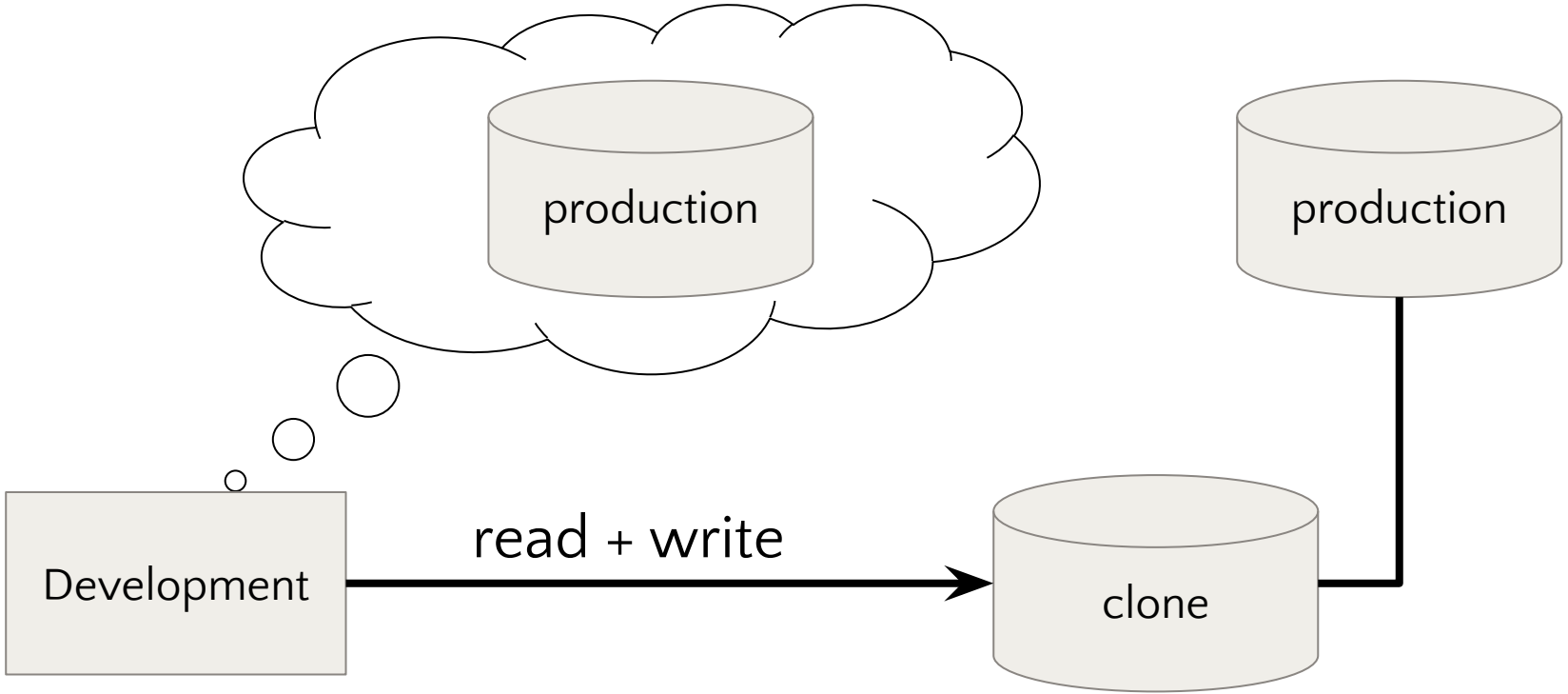
Database for development:

- Low cost
- Low maintenance
- Matches prod

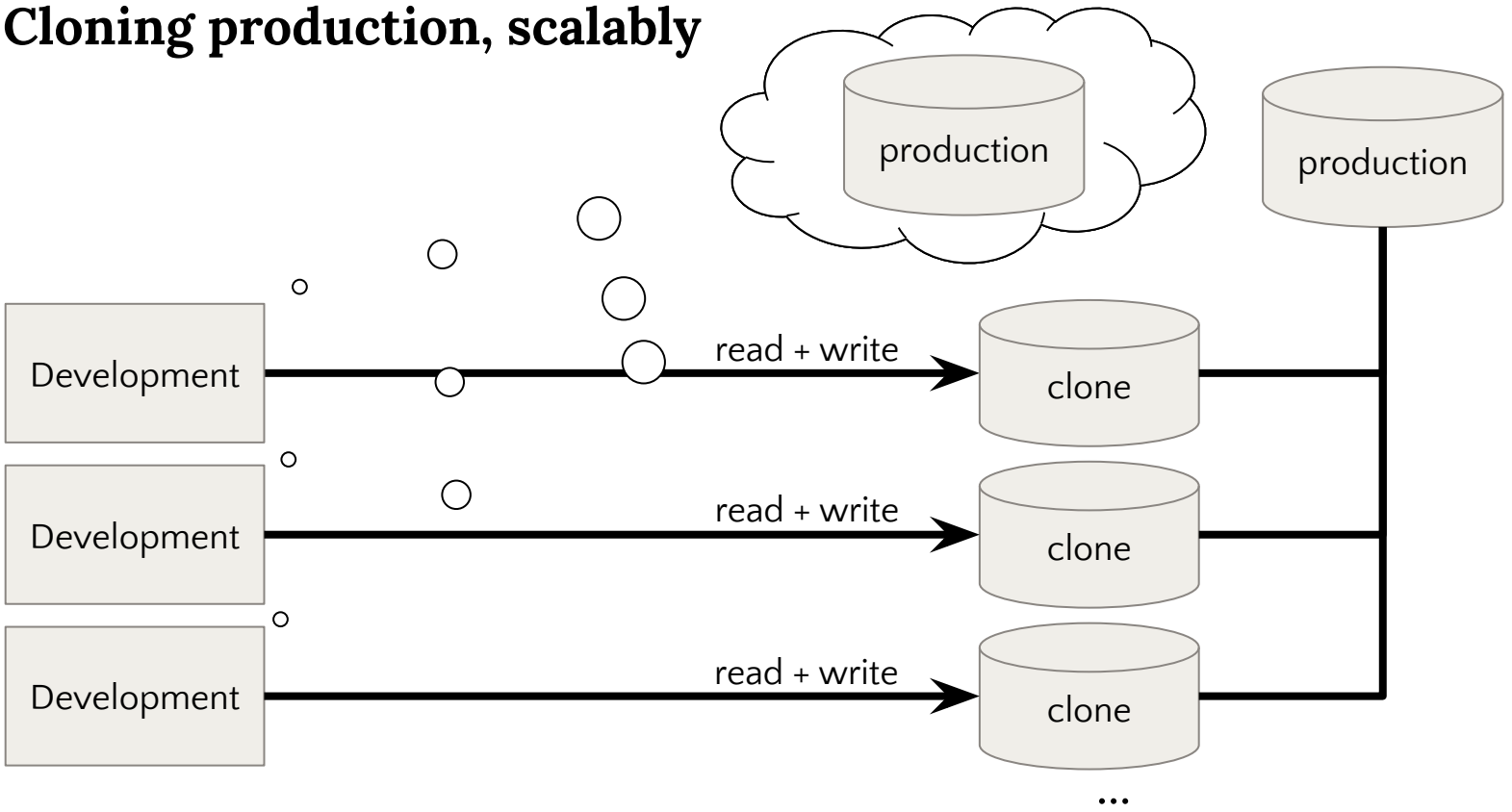


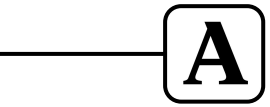
Solution:
DB thin cloning

A Cloning production



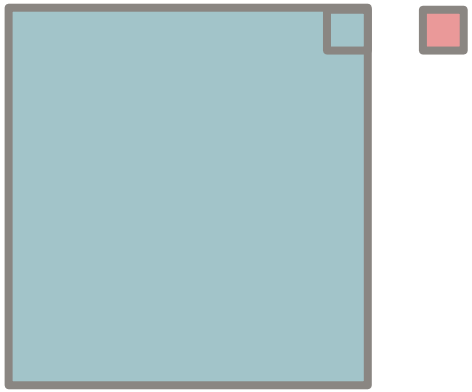
A Cloning production, scalably

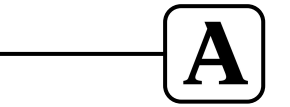




Copy on write

- **Storage or filesystem layer**
  - e.g. ZFS, LVM, k8s + OpenEBS
- **Copy on write clone of $PGDATA**
- **Run postgres off of the clone**
  - Separate compute/container
  - Start the server, promote
- **Get a connection string**

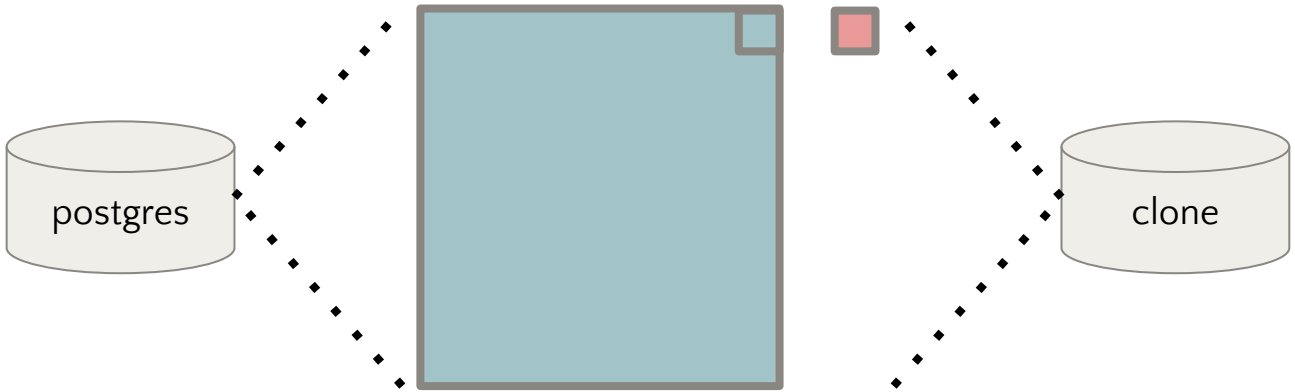




=> writable “thin clone” of the DB

Creating new clones:

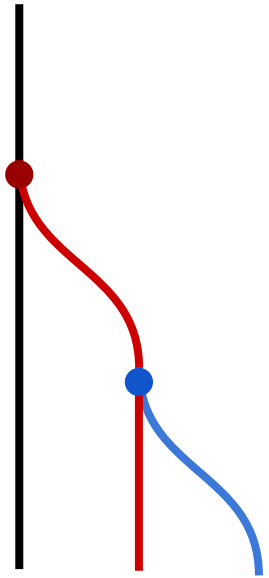
- Fast (5 TB DB, 2.5 seconds)
- Only takes more storage after writes



Postgres (or “postgres compatible”) thin cloning options

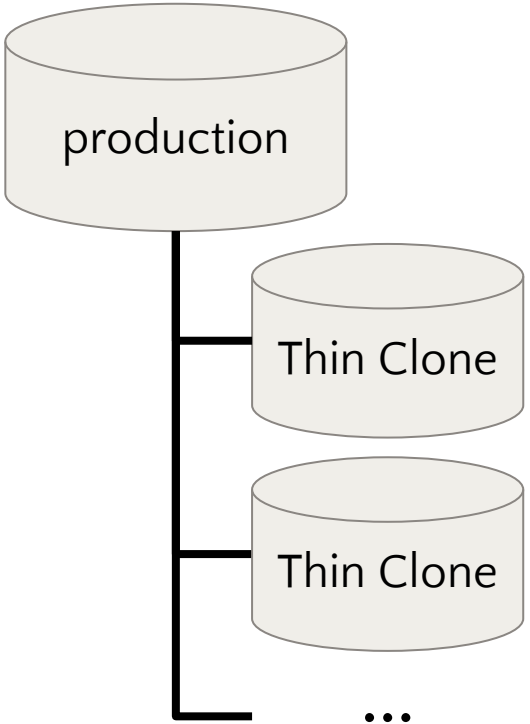
- DBLab Engine
- Xata
- Amazon Aurora
- Neon
- TigerData

“Branching”



Architecture - “native”

- Xata
- Neon
- Amazon Aurora
- TigerData

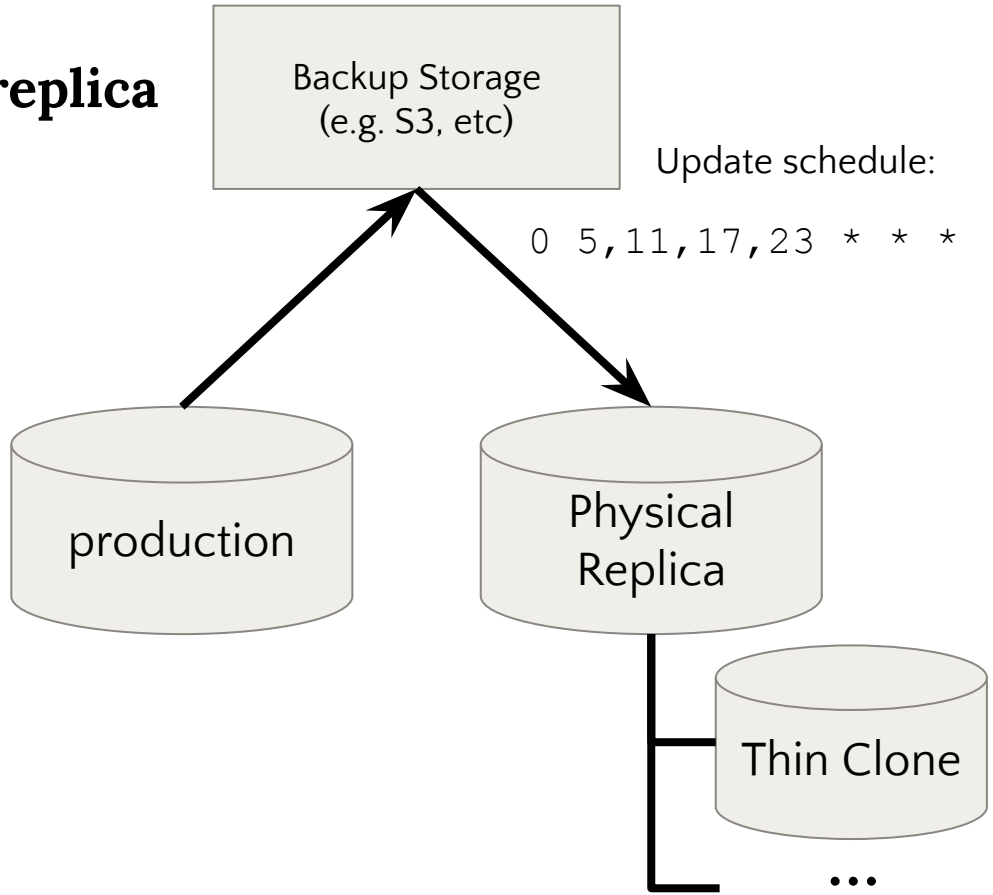


A Architecture - physical replica

- DBLab Engine

Advantages:

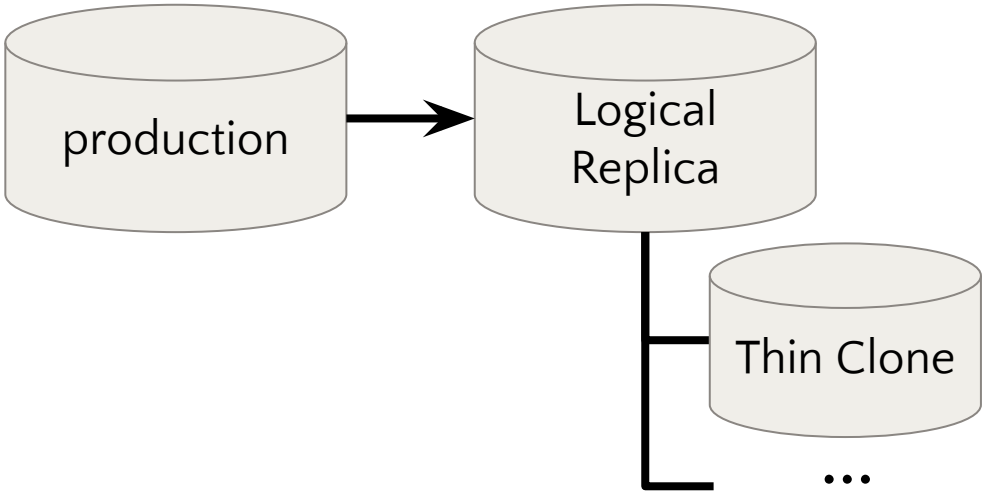
- Sync performance
- Exact state of DB
  - even corruption



A Architecture - logical replica

- **Xata**
  - Anonymization pgstream
- **DBLab Engine**

pg_dump
pgstream
pgcopydb



Advantages:

- **Flexibility**
- **Anonymization**

What Academia.edu uses

- DBLab Engine open source (physical approach)
- Aurora (testing DB upgrades)
- Testing Xata (logical approach)
- A script to get clones
  - Calls clone APIs, templates dev DB config
- An env var to use them

Simulated demo

```
$ script/dbclone --all
```

Simulated demo

```
$ script/dbclone --all
Requesting clone of main…
Requesting clone of bibliography...
bibliography cloned in 2.4 seconds
main cloned in 2.7 seconds
```

Simulated demo

```
$ DBCLONE=1 bundle exec rails c
Loading development environment (Rails)
1:(main)>
```

Simulated demo

```
$ DBCLONE=1 bundle exec rails c
Loading development environment (Rails)
1:(main)> Authorship.last.id
```

Simulated demo

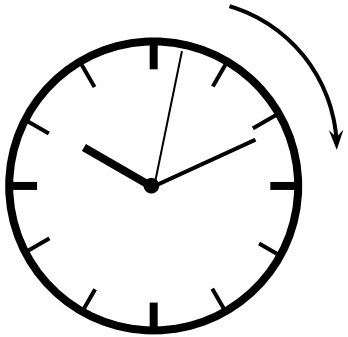
```
$ DBCLONE=1 bundle exec rails c
Loading development environment (Rails)
1:(main)> Authorship.last.id
(11.4ms) SELECT "authorships".* FROM
"authorships" ORDER BY "authorships"."id"
DESC LIMIT 1
=> 162_088_564
```

Simulated demo

```
$ DBCLONE=1 bundle exec rails c
2:(main)> Authorship.count
```

Simulated demo

```
$ DBCLONE=1 bundle exec rails c
2:(main)> Authorship.count
(...) SELECT COUNT(*) FROM "authorships"
```



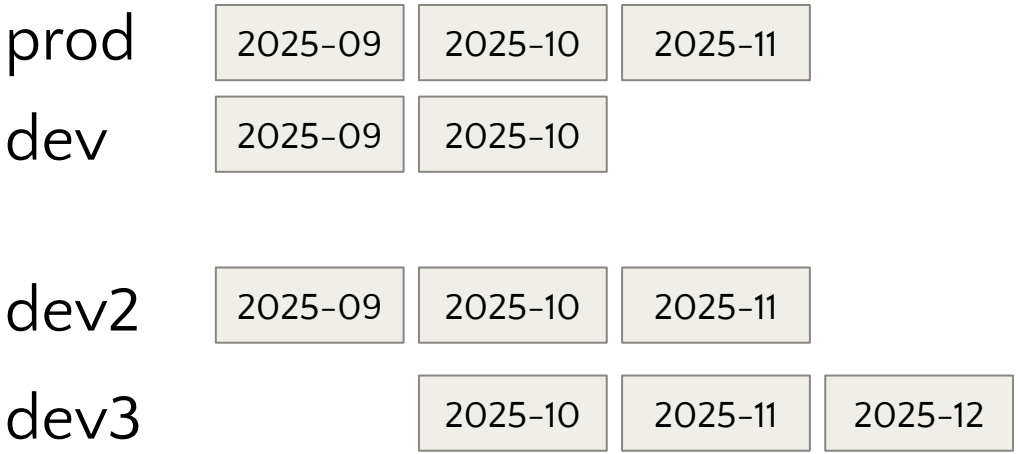
Schema differences

Partitioning

```
\d+ notifications

[...]

Partition key: RANGE (created_at)

Partitions: notifications_2025_09 FOR VALUES FROM ('2025-09-01
00:00:00') TO ('2025-10-01 00:00:00'),

            notifications_2025_10 FOR VALUES FROM ('2025-10-01
00:00:00') TO ('2025-11-01 00:00:00'),

            notifications_2025_11 FOR VALUES FROM ('2025-11-01
00:00:00') TO ('2025-12-01 00:00:00')
```

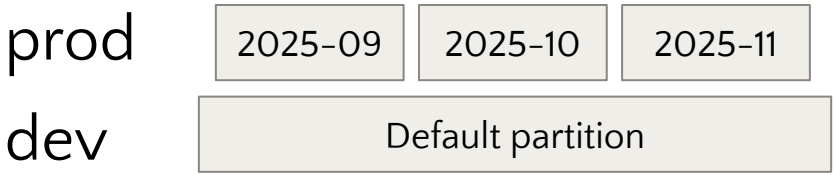
Partitioning in development

Cron?



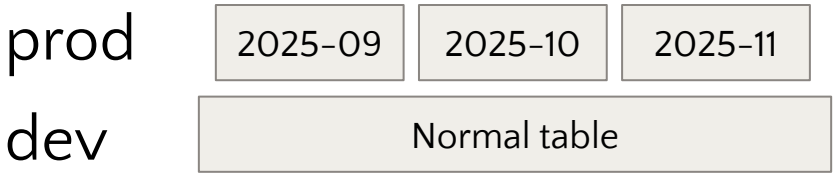
Partitioning in development

Default partition?



Partitioning in development

Just a normal table in dev?



Partitioning in development

Or: thin clone

prod	2025-09	2025-10	2025-11
dev	2025-09	2025-10	2025-11

Slow queries

Fast in dev, slow in prod

- Scale of data
- Bad query plans
- Bloat, VACUUM issues

Debugging a slow INSERT, UPDATE, or DELETE

```
EXPLAIN ANALYZE
DELETE FROM [table] WHERE [...] ;
```

Debugging a slow INSERT, UPDATE, or DELETE

```
BEGIN;
EXPLAIN ANALYZE
DELETE FROM [table] WHERE [...] ;
-- capture output
ROLLBACK;
```

Debugging a slow INSERT, UPDATE, or DELETE

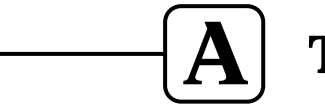
```
BEGIN;
EXPLAIN ANALYZE
DELETE FROM [table] WHERE [...] ;
-- capture output
ROLLBACK;
```

Risks:

- Locking
- Bloat, WAL writes
- Mistakes/autocommit

Thin cloning advantages: query performance

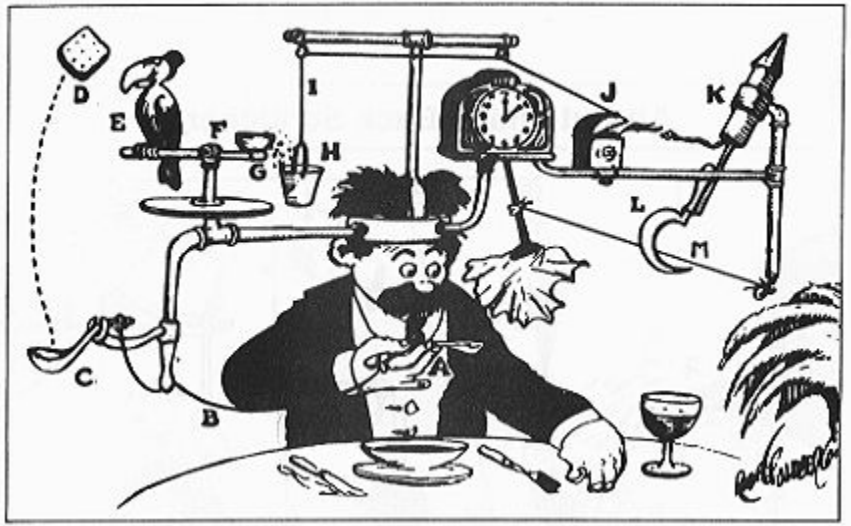
- **Safer `EXPLAIN ANALYZE` on DML**
- **What if you don’t have the query?**
  - ORM
  - Statement constructed at runtime
- **Go to web page in dev, prod queries**



Testing a fix

query planner

Statistics?
Index?
Config?



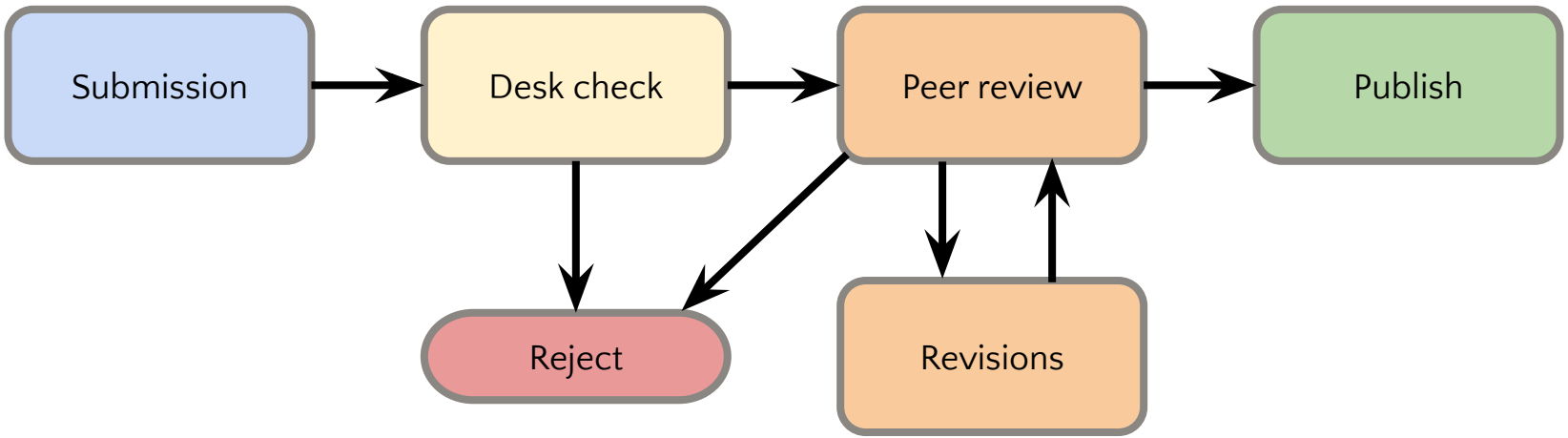
profit?

0 -> 1 Product

A The state of academic publishing

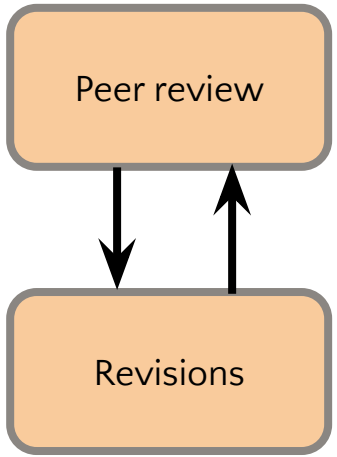
- 17th century tech in the internet age
- Slow peer review, slow time to publish
- How can we speed it up?

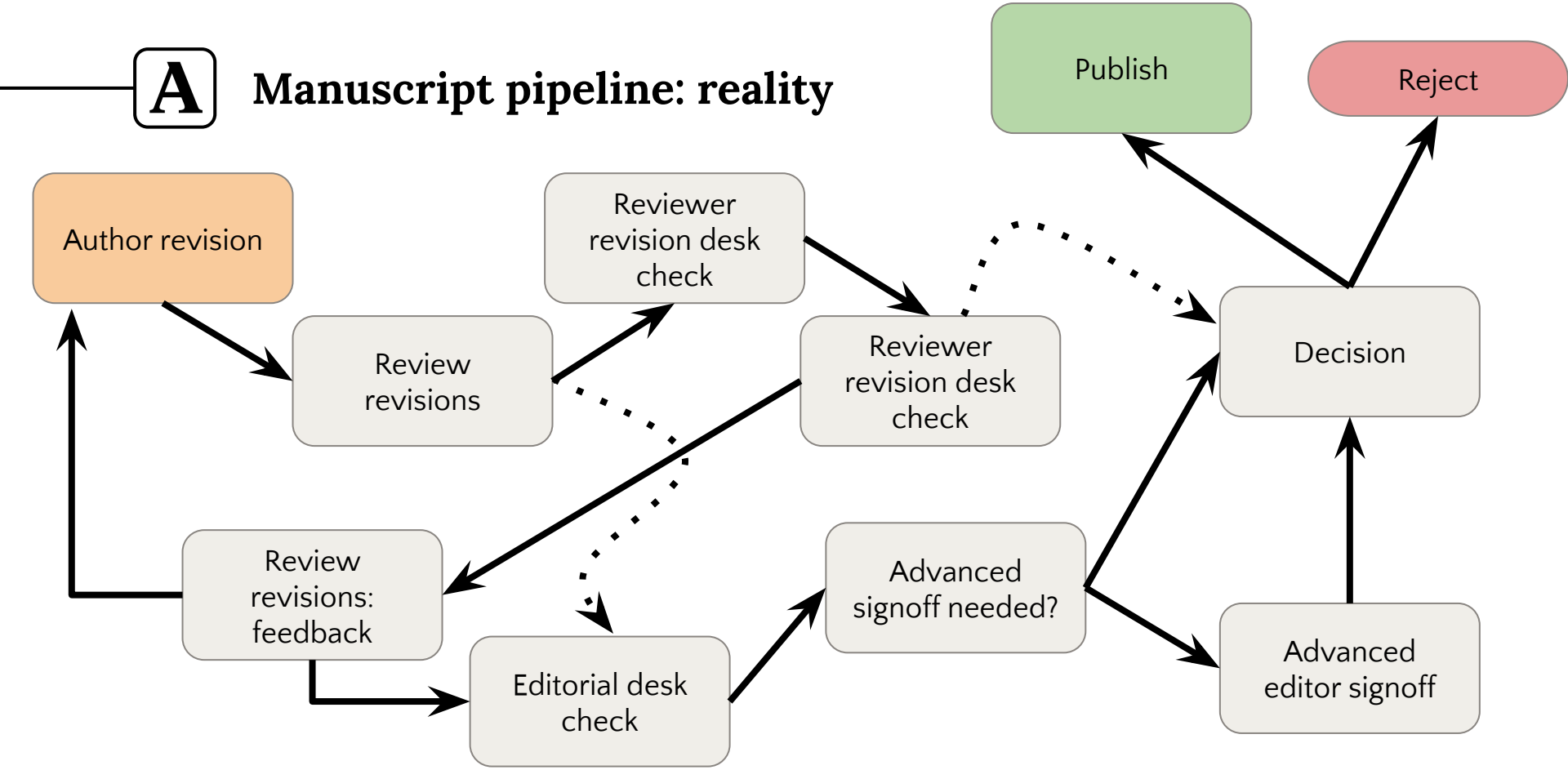
A Manuscript pipeline: idealized



A Manuscript pipeline: idealized

Just revise and resubmit




A
Manuscript pipeline: reality
Publish
Reject
Author revision
Reviewer revision desk check
Review revisions
Reviewer revision desk check
Decision
Review revisions: feedback
Advanced signoff needed?
Editorial desk check
Advanced editor signoff

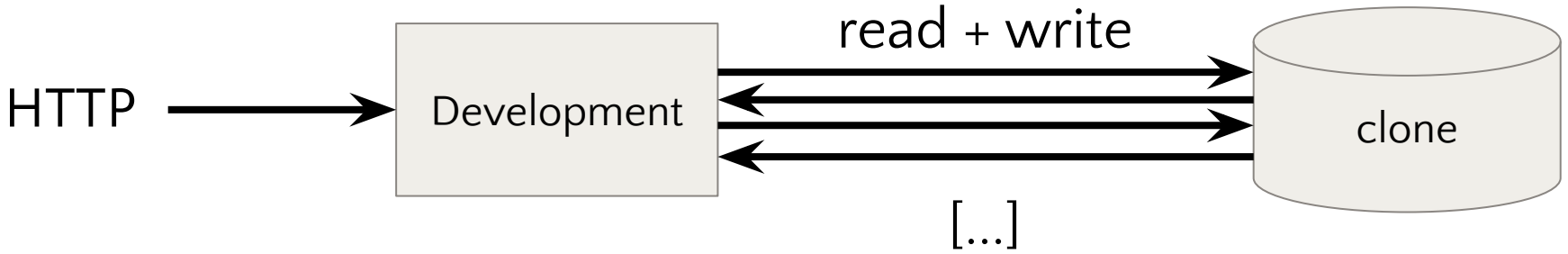
A Feedback cycles

Thin cloning has been revolutionary for us

- Replicating complex bugs
- Awareness of query performance
- Iterating faster

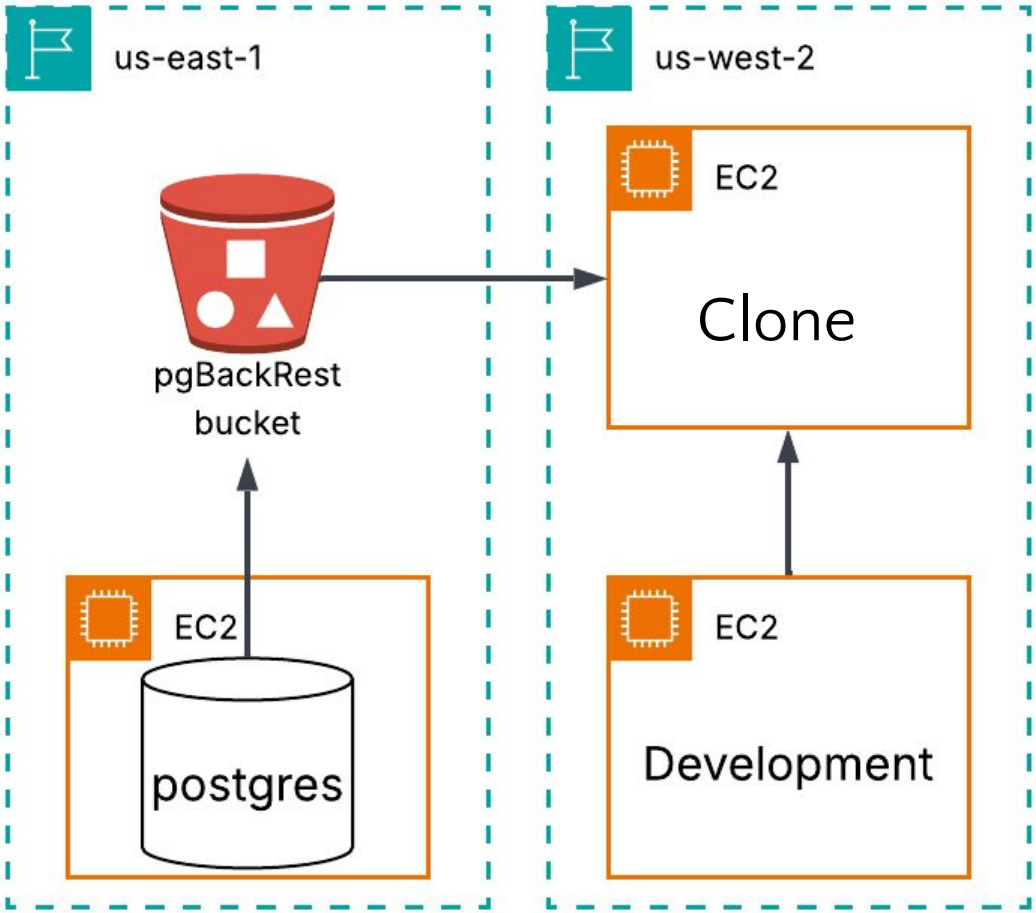
Challenges and advice

Latency is key



A Latency is key

Want dev app and dev db in the same region



Performance

- Has the dataset of prod
- Less compute power, memory than prod
- Some development workflows can be slower
- Thin clones are not ideal for benchmarking
- Some ZFS `recordsize` tuning

Data in places other than postgres

- S3
- Elasticsearch
- Redis
- Dynamo
- etc

```
select id,
file_name
from documents
limit 1;
[ RECORD 1 ]-+--------------
id           |      123456789
file_name    |lorem_ipsum.pdf
```

S3

Data in places other than postgres

- **Custom workarounds for dev**
  - “Intentionally different”
- **Pareto principle, 80/20**

PII + Compliance

“dev == prod” approach

- Lock down dev access
- Avoid DBs that are too sensitive

PII + Compliance

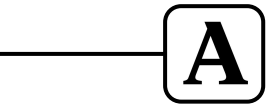
Data anonymization approach

- **Logical + anonymize before it reaches dev**
  - pgstream + anonymization
- **Physical + Extension**
  - PostgreSQL Anonymizer
  - Dynamic masking

Closing thoughts

A Why just postgres?

- I’m not aware of any key technical reasons
- Strong community + ecosystem
- I wish I could use this on every “database”



Summary

- Thin cloning was game-changing for us
- Questions?
- Feedback?

Feedback: https://www.postgresql.eu/events/pgconfeu2025/feedback/7102/

Appendix

A

A Image credits

https://commons.wikimedia.org/wiki/File:Atlas_%28mythology%29.svg

- Deed - CC0 1.0 Universal - Creative Commons

https://openclipart.org/detail/238859/magical-unicorn-silhouette-no-stars

- Deed - CC0 1.0 Universal - Creative Commons

https://en.wikipedia.org/wiki/File:Tyrannosaurus_Rex_Holotype.jpg

- Deed - Attribution-ShareAlike 3.0 Unported - Creative Commons

https://openclipart.org/detail/335938/political-map-of-the-countries-of-the-world-in-2018-neutral-colors

- Deed - CC0 1.0 Universal - Creative Commons

https://en.wikipedia.org/wiki/File:Rube_Goldberg%27s_%22Self-Operating_Napkin%22_(cropped).gif

- Public domain