Production-to-Dev in One Button

K8s-native PII anonymization for Percona XtraDB Cluster Backups

Percona Live 2026 · Amsterdam

Yannick Dixken · dixken.de

> whoami

> Yannick Dixken · Berlin · ~~system~~ ~~devops~~ *platform engineering*

> "grew up" with Percona MySQL at former employers (SysEleven, ScaleCommerce..)

> Independent contractor since 2019, moving *.yaml for a living

> Long-term customers in e-commerce, public cloud, federal defense, finance (..)

> Spare time: CTFs, breaking things when my ADHD kicks in

> Writing about data privacy & battles with companies after reported vulnerabilities

> Sometimes I like to build the neglected stuff

Mumble audibly if you've encountered .. in the past

> [!!] Dev Environments running with prod data · purpose limitation ⇒ gone

> [!!] Debug logs with exposed PII · data minimisation ⇒ gone

> [!!] Everyone who ever needed access still has it ⇒ least privilege, gone

> [!!] Debug dumps are created from prod & kept / passed over · retention ⇒ gone


I DON'T KNOW HOW TO PUT THIS, BUT..
GDPR IS KIND OF A BIG DEAL
imgflip.com

GDPR, read as an engineering spec

> [OK] **Data minimization** · dev does not need real emails

> [OK] **Purpose limitation** · the data was collected to ship orders, not to debug

> [OK] **Least privilege** · dump access is PII access

> [OK] **Auditability** · "who saw what?"

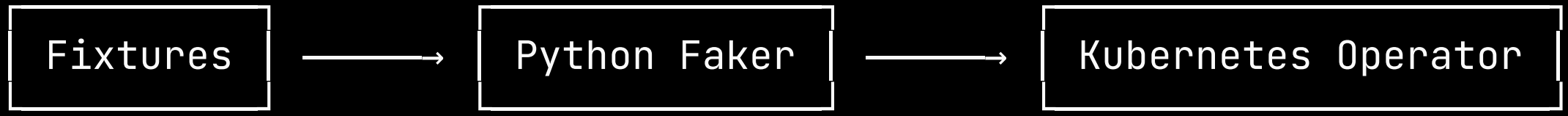
... and, IMO:

> [OK] **Repeatability** · a one-off script isn't

Question

Why does dev need production data at all?

My Journey: from fixtures to an operator



The first naive attempts

attempt #1: hand-curated fixtures

> [!!] **Drift** · the schema moves, fixtures do not

> [!!] **Unrealistic shapes** · 500 rows where prod has n millions

attempt #2: mysqldump | sed

> [!!] **Slow** · logical restore of a multi-hundred-GB dataset

> [!!] **Breaks constraints** · UNIQUE emails, FK chains

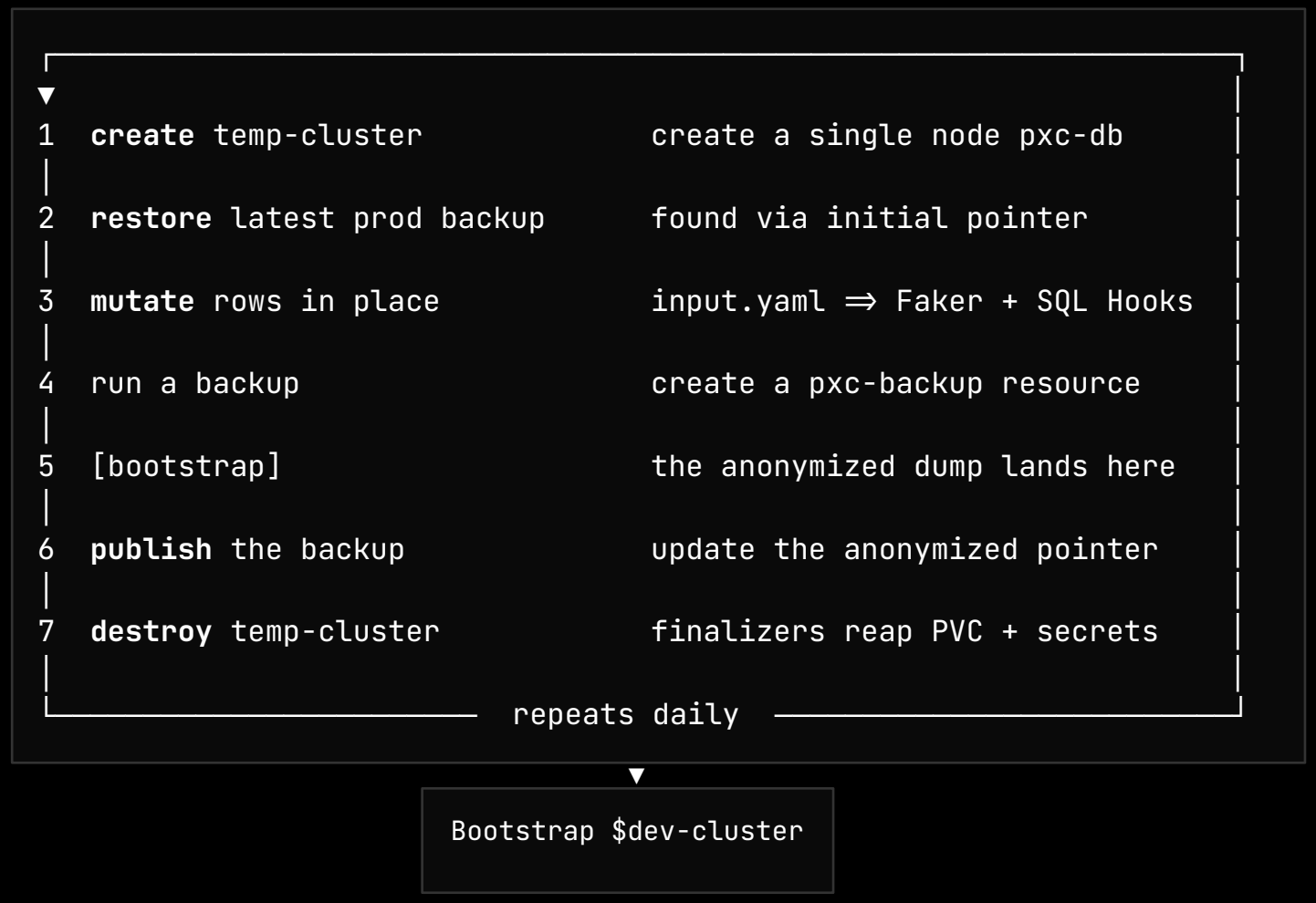
> [!!] **The worst part** · everyone with dump access has already seen the PII

It might work for a while, but let's do this right and repeatable:
Anonymise regularly based on rules, without a logical restore and multi-threaded

IDEA: THE EPHEMERAL SANDBOX

create · restore · mutate · publish · destroy

THE EPHEMERAL SANDBOX




```
{
"name": "$CLUSTER-$DATE",
"destination": "s3://prod-backup/$CLUSTER-
$DATE"
}
```

▼

```
[input.yaml]
users:
  - table: user
    columns:
      email: ascii_email
      given_name: first_name
      birthday: date
      phone: phone_number
    - file:
        path: "static/users/hook.sql"
        mode: "post"
```

▼

```
{
"name": "$TEMPCLUSTER-$DATE-anonymized",
"destination": "s3://bootstrap/$TEMPCLUSTER-
$DATE-anonymized"
}
```

Boiling it down: Operational pattern

> Ephemeral Sandbox // restore, mutate, republish, destroy based on the *latest* PerconaXtraDBClusterBackup

> Latest Pointers // "latest" is a pointer created after a successful backup

> Bootstrapping a fresh cluster // based on the latest anonymised backup

Create an ephemeral pxc-db cluster resource

```
name = "temp-cluster-" + os.urandom(4).hex()
spec = {
    "crVersion": "1.16.1",
    "tls": {"enabled": False},
    "unsafeFlags": {"tls": True, "pxcSize": True, "proxySize": True},
    "upgradeOptions": {"apply": "disabled"},
    "pxc": {"size": 1, "persistence": {"size": "500Gi"}},
    "finalizers": ["finalizers.percona.com/delete-pxc-pvc",
                   "finalizers.percona.com/delete-ssl"],
}
```

> [OK] Optional: Single node, no TLS, no HA · HA not needed, faster to setup, faster to work with

> [!!] Mass UPDATEs on 3 nodes · certification and flow control would tax every anonymised batch

> [OK] No SST partners to sync · the fresh backup is the only artifact we keep

> [OK] Finalizers · deleting the CR reaps the 400Gi PVC, cleanup is the operator's job

Physical restore, tuned hard

```
create restore CR ──> xbcloud ──> xbstream ──> xtrabackup ──> wait for "Succeeded"

backupSource.destination =
    s3://percona-xtrabackup-prod/percona-cluster-2026-08-01-20:00:14-full
```

> [OK] Tune your restore performance, test this!

> [OK] Reduced to ~15 minutes for a restore of the customers dataset ~200GB

> [??] logical load of the same dataset · Measured hours, not minutes

> [OK] Yay! You automatically test your backup - every day!

Mutate in place

```yaml
users:
  - table: user
    columns:
      email: ascii_email
      given_name: first_name
      birthday: date
      phone: phone_number
```

input.yaml

```yaml
users:
  - table: user
    condition: "WHERE email LIKE '%@shop.internal'"
```

Ignore Conditions

> SELECT · Fake · UPDATE · Commit · no magic involved

> Generators need to respect the structure · UNIQUE indexes, NULL stays NULL (...)

> Ignore Conditions are respected · e.g. SQL WHERE's, short reviewed list

> Pre & Post Hooks · static queries to support the process

Check: https://faker.readthedocs.io/en/master/ | https://github.com/brianvoe/gofakeit

Prepare the clone, some examples

> [OK] Payment providers → sandbox · the clone must never charge a card

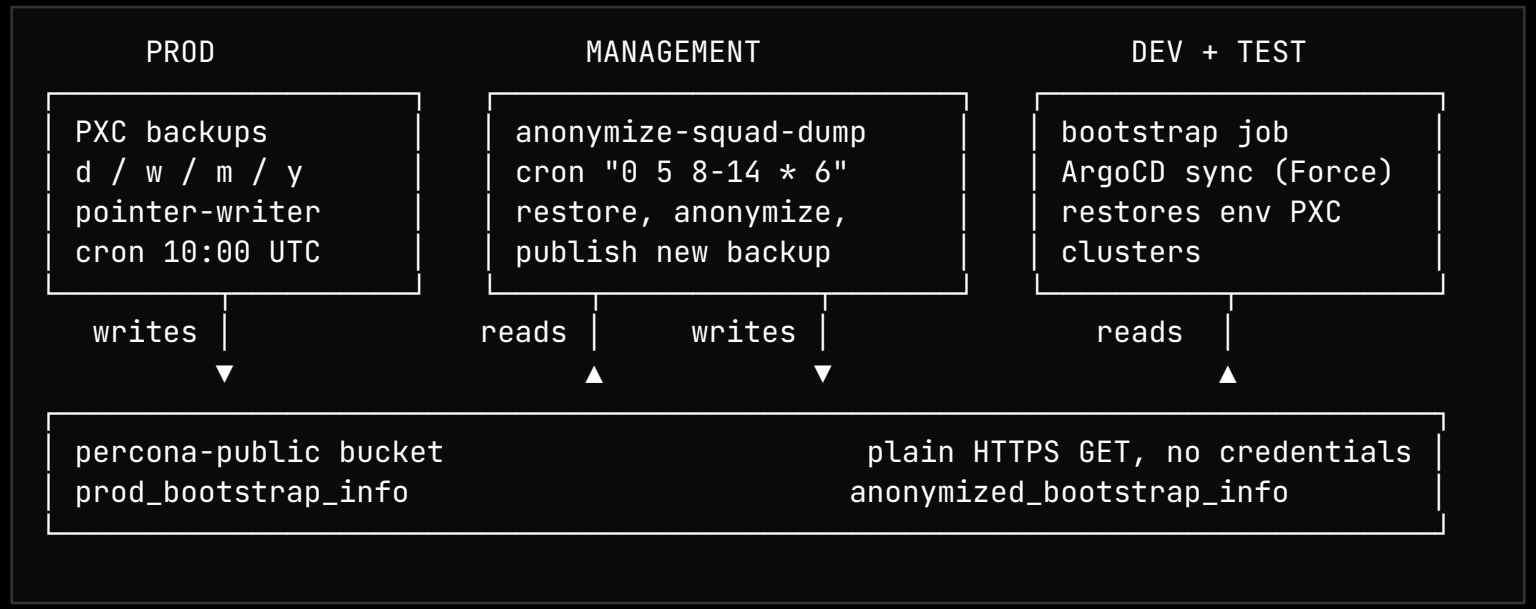
> [OK] TRUNCATE the mail queue and schedulers · the clone must never email a customer

> [OK] DROP app users · Crossplane (or else) re-provisions per-env credentials anyway

> [OK] ... whatever you need to run in 'pre' & 'post'.

This is - of course - highly specific to the actual use case.

Patterns 2 + 3: putting it together



Downstream never touches prod storage:
It follows one JSON pointer to retrieve the latest anonymised backup

PATTERN 2 · THE LATEST POINTER

"latest" is a document

```
{
  "name": "cron-percona-cluster-daily-...",
  "destination": "s3://percona-xtrabackup-prod/percona-cluster-2026-08-01-20:00:14-full"
}
```

> [OK] a JSON file with two keys · keep it simple

> [OK] daily 10:00 UTC · writer finds the newest *Succeeded* PerconaXtraDBClusterBackup

> [??] "latest" is a document, not a tag · consumers decoupled from backup naming

IDEA: LET'S PACKAGE THIS

kubebuilder can I haz operator plz?

Three Jobs become 5 CRs

pointer-writer CronJob	→	BackupPointer
input.yaml + SQL files	→	AnonymizationPolicy
anonymize-squad-dump Job	→	AnonymizationRun
its cron, "0 5 8-14 * 6"	→	AnonymizationSchedule
bootstrap-percona-cluster Job	→	Bootstrap

The pattern itself did not change - but we're now "native"

An example: AnonymizationPolicy

```
steps: [{name: sandbox-pay, configMapKeyRef: {name: hooks, key: pay.sql}}]
databases:
  - name: shop_users
    post: [{name: sandbox-pay}]
    tables:
      - {name: user, ignore: {where: "email LIKE '%@shop.internal'"},
         columns: [{name: email, strategy: email, consistent: true},
                   {name: phone, strategy: phone, params: {locale: de}}]}
      - {name: session_tickets, action: Truncate}
```

> [OK] 29 strategies included right now · email, iban, vatId, address, phone (...)

> [OK] Typos fail at admission · random_letters: Unsupported value, no NULLed column

> [OK] Common Actions included · e.G Truncate

> [OK] consistent: true · one HMAC per value, the same pseudonym in every table, re-roll on collision

Failure is now a condition

```
kind: AnonymizationRun
spec:
  source: {backupPointerRef: {name: latest-backup}, restore: {...}}
  policyRef: {name: shop}
  tempCluster: {crVersion: "1.20.0", storage: {size: 400Gi, ...}, ...}
  output: {objectStorage: {bucket: anonymized, ...}}
  runner: {workers: 4, pageSize: 5000, dryRun: true}
  timeouts: {clusterReady: 30m, restore: 3h, anonymize: 6h,
             backup: 3h, publish: 10m, cleanup: 30m}
```

> [OK] Phases · Completed, Failed; nine phases before the run

> [OK] Failure has a reason · SchemaMismatch, PermissionDenied, UniqueViolation, Timeout

> [OK] Status: audit trail · source.backupName, policyHash, startedAt, completedAt (...)

> [OK] The runner Job · ORDER BY pk, checkpoint per page, verdict via termination message

How this looks like in action

```
00:00:19  SourceResolved    FromPointer         the restore source is snapshotted
00:01:48  TempClusterReady  Ready               the temporary cluster is ready
00:04:18  Restored          Succeeded           the PXC restore finished successfully
00:04:21  Anonymized        Succeeded           the runner completed successfully
00:05:01  BackedUp          Succeeded           the output backup succeeded
00:05:01  Published         Uploaded            the output pointer was uploaded
00:05:01  CleanedUp         TempClusterDeleted  the temporary cluster and owned credentials are absent
00:05:01  Complete          Succeeded           the pipeline and temporary resource cleanup finished
```

> [OK] BackupPointer · Ready, Fresh, 5 backups seen, key prod_bootstrap_info

> [OK] AnonymizationPolicy · Valid, hash sha256:688c1cae…, snapshotted into the Run

> [OK] AnonymizationRun · Completed in 4m43s, 5000 rows per page, temp cluster deleted

> [OK] AnonymizationSchedule · 0 6 * * 0 UTC, next 2026-09-13T06:00Z

> [OK] Bootstrap · downstream restored in 6m26s, 9 Crossplane objects paused, resumed

What you want

> [!!] **Identify the prerequisites** · Schema, Tables, Columns, Strategies, Relationships (...)
> [!!] **Config validation** · Fail early if possible
> [!!] **Per-table checkpoints** · any late failure otherwise retries the whole run
> [!!] **Multi-threading** · spawn child processes per table, maybe even consider breaking up the tables in equal parts

I released the sanitised version, feel free to steal

https://github.com/ydixken/pxc-anonymizer

Q&A

WEB dixken.de/blog (Slides will be here)
GIT git(lab|hub).com/ydixken
LINK linkedin.com/in/ydixken
FEDI @spektrum@chaos.social
MAIL yannick@dixken.de · 0x938EE3DC

Questions?