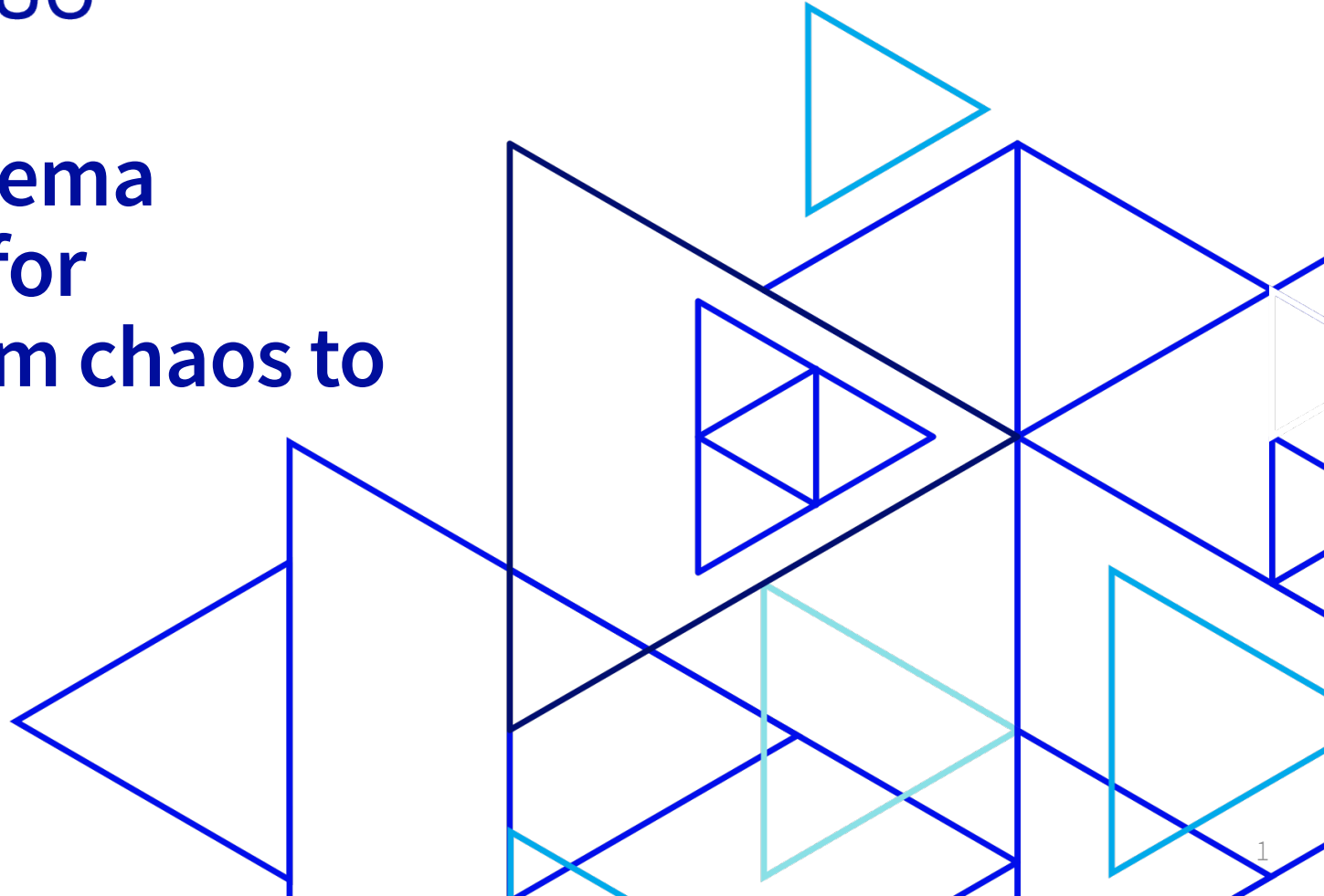




Databases schema management for lazybones: from chaos to heaven

Julien Riou
FOSDEM
February 6, 2021



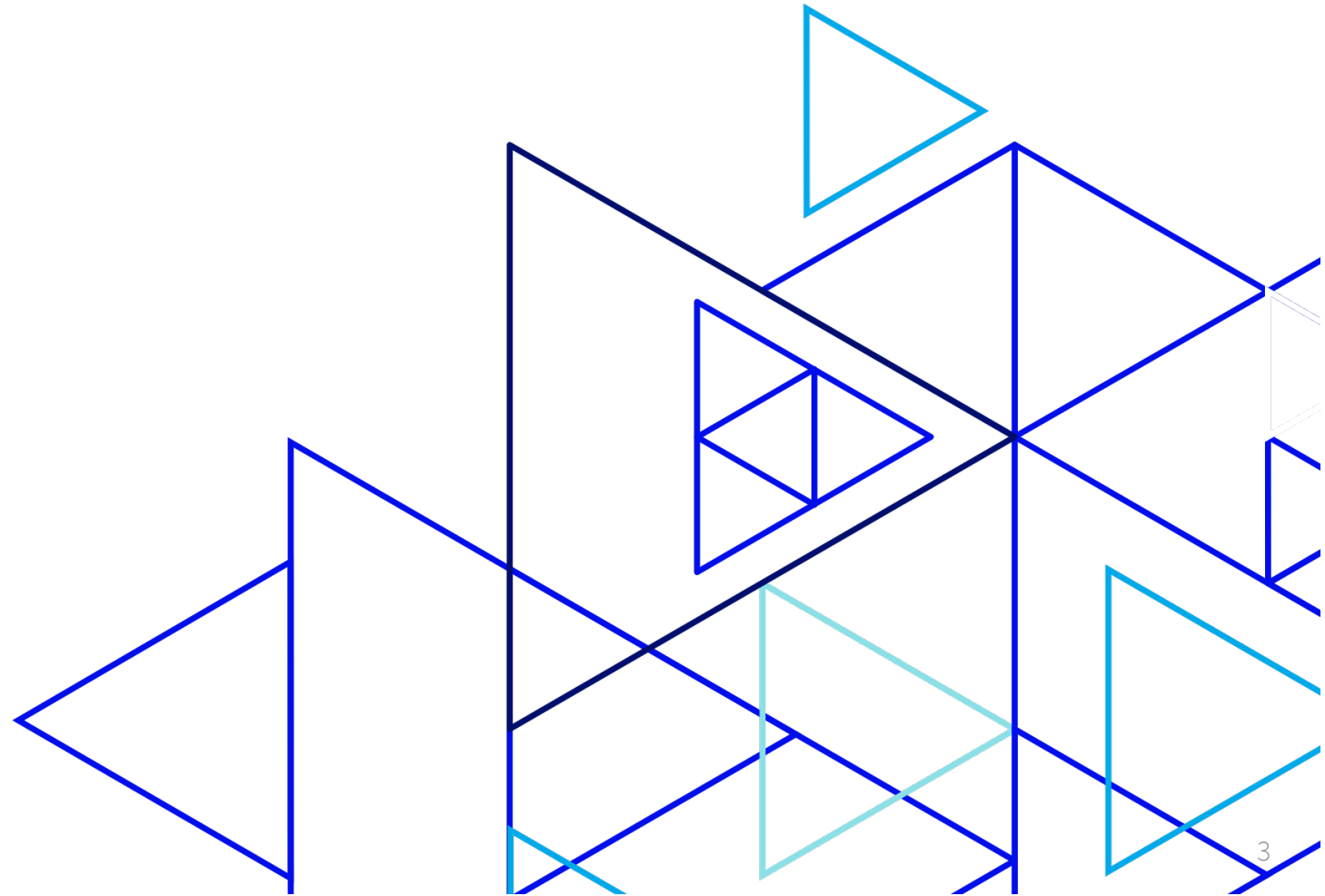
Speaker



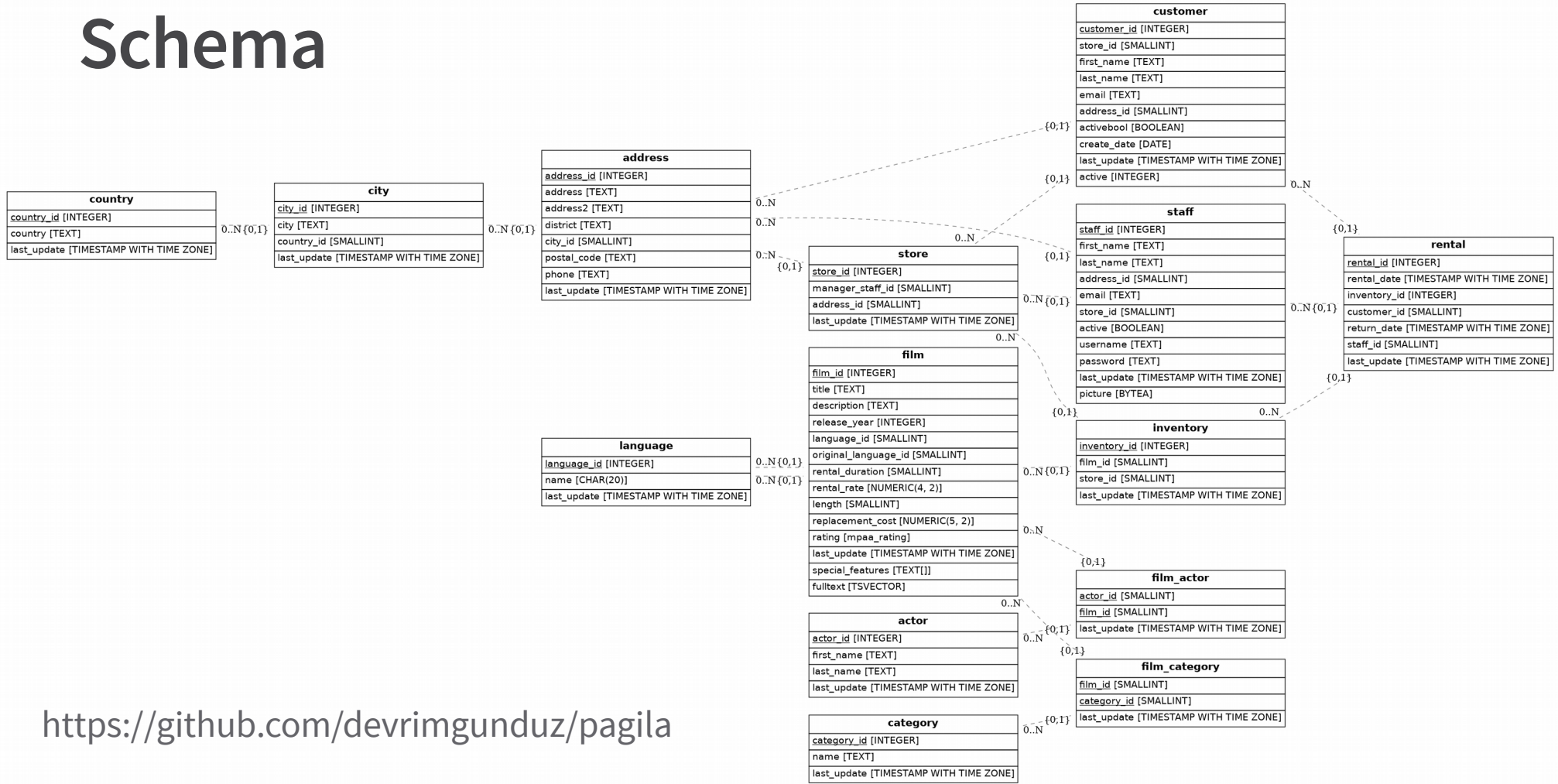
- ▶ Julien Riou
- ▶ DBA since 2012
- ▶ Tech lead in the databases team at OVHcloud since 2015
- ▶ <https://julien.riou.xyz/>

Definitions

FOSDEM
February 6, 2021



Schema

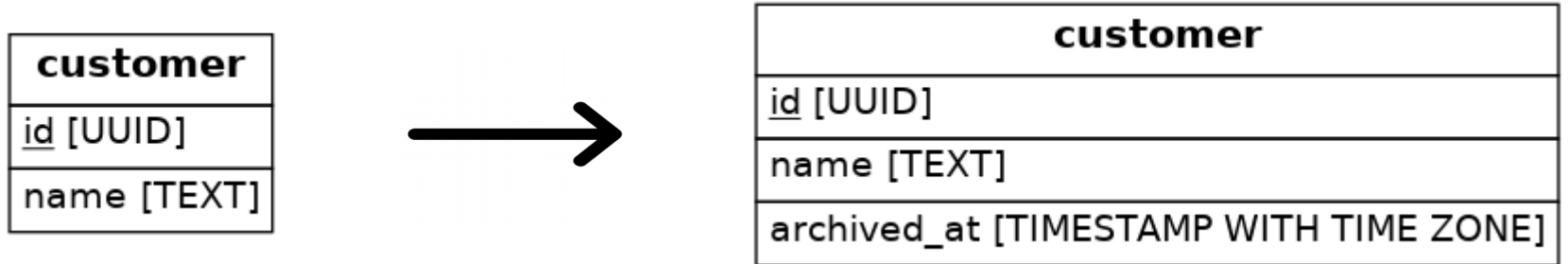


<https://github.com/devrimgunduz/pagila>

Schema changes over time

- ▶ New tables
- ▶ New columns
- ▶ New constraints
- ▶ New everything

Schema migration

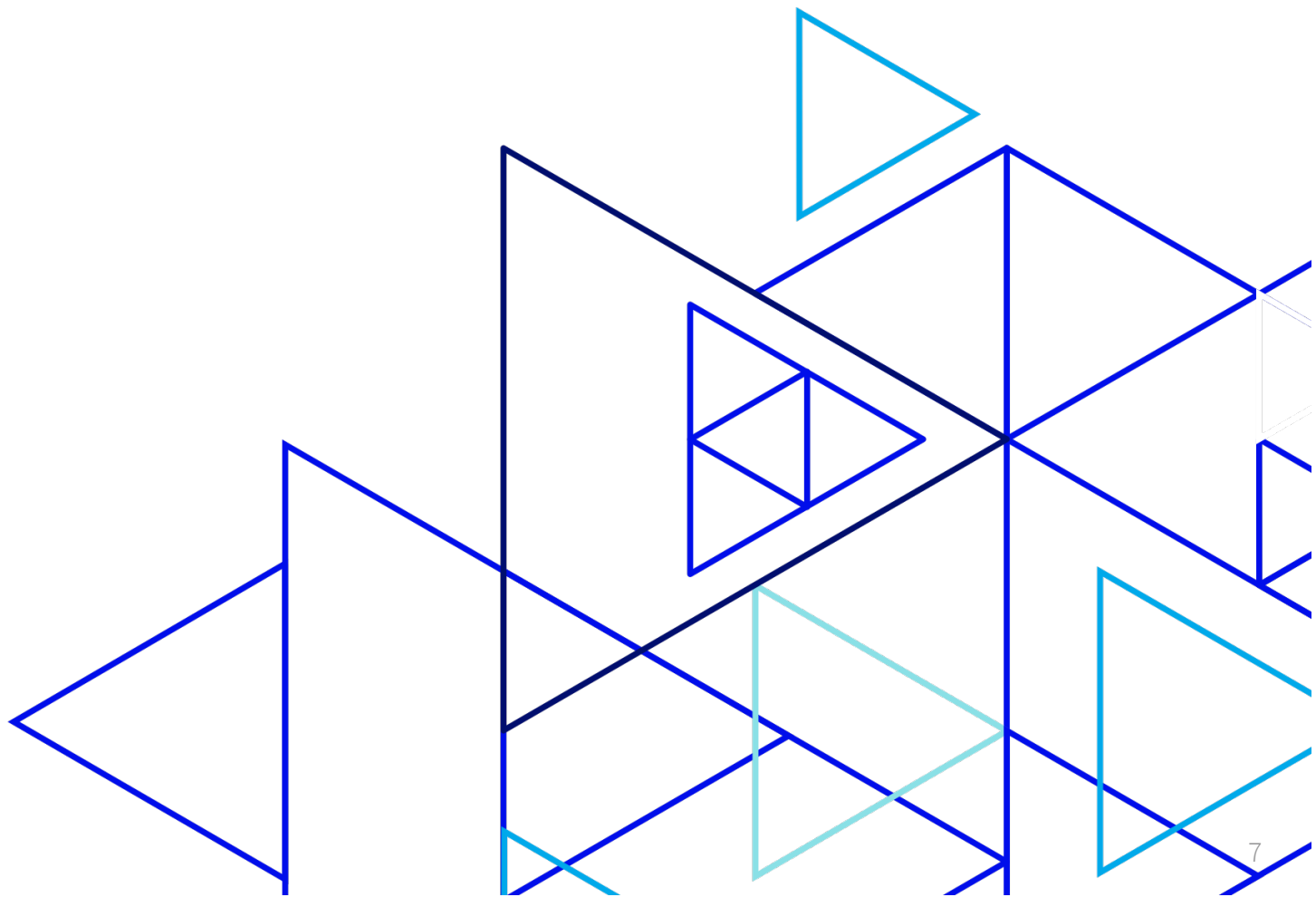


► Data definition language (DDL)

```
ALTER TABLE customer  
  ADD COLUMN archived_at TIMESTAMP WITH TIME ZONE;
```

Context

FOSDEM
February 6, 2021

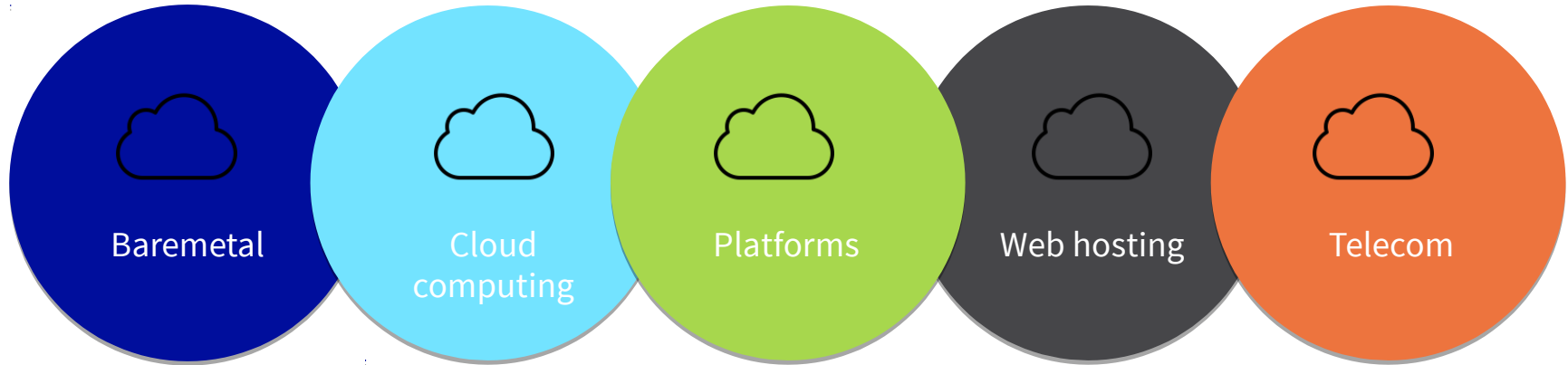




OVHcloud



Critical services



They all rely on **internal databases** for control plane!

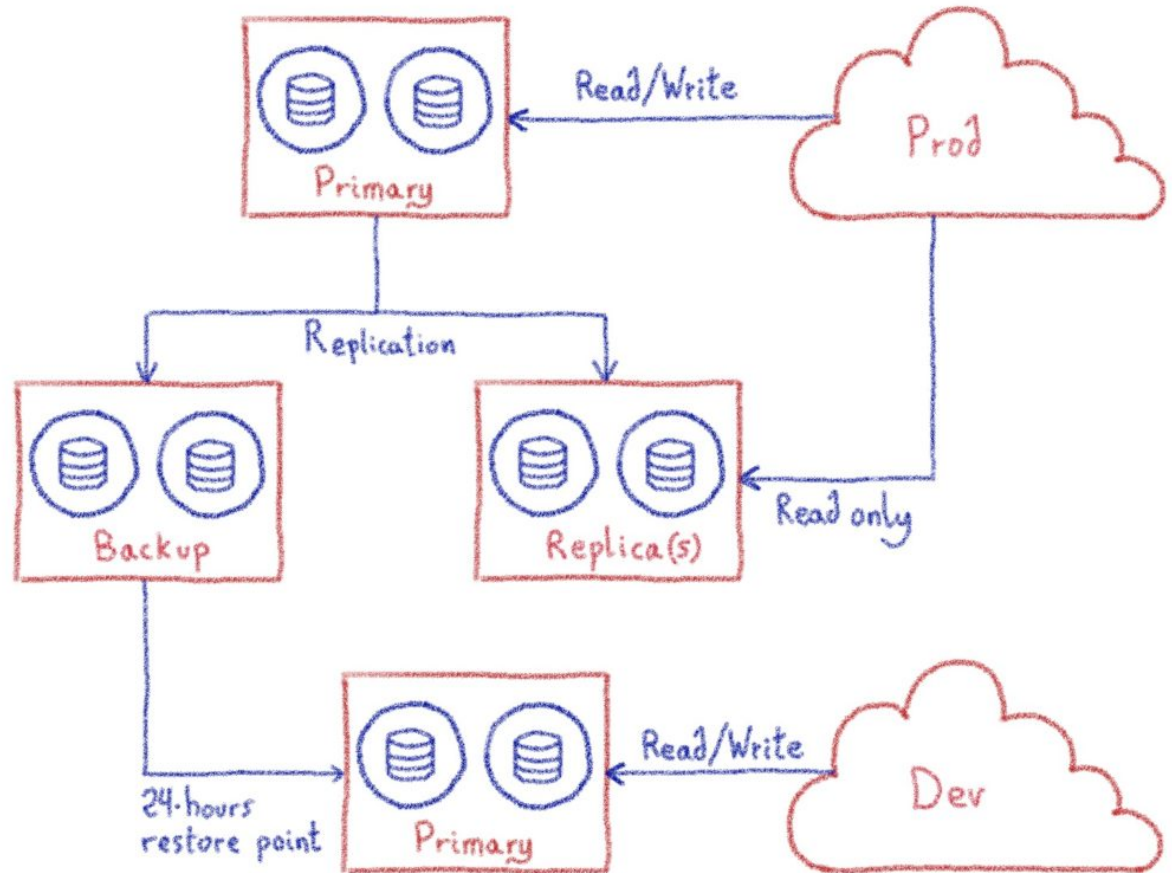
Internal databases

- ▶ 60 clusters
- ▶ MySQL and PostgreSQL
- ▶ 3000 applications
- ▶ 700 users
- ▶ 500 databases
- ▶ Worldwide



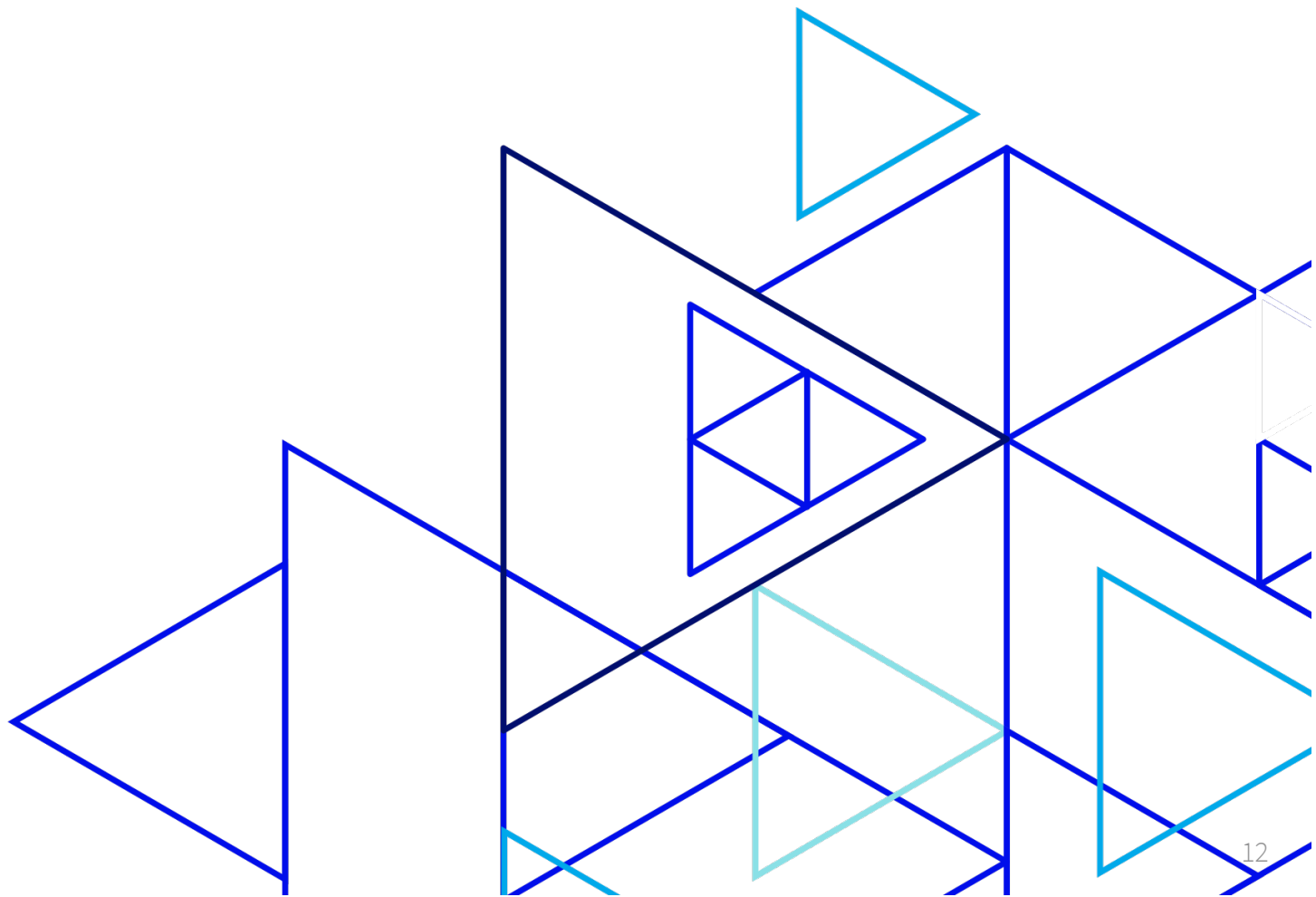
Internal cluster example

- ▶ MySQL
- ▶ PostgreSQL



Overview

FOSDEM
February 6, 2021

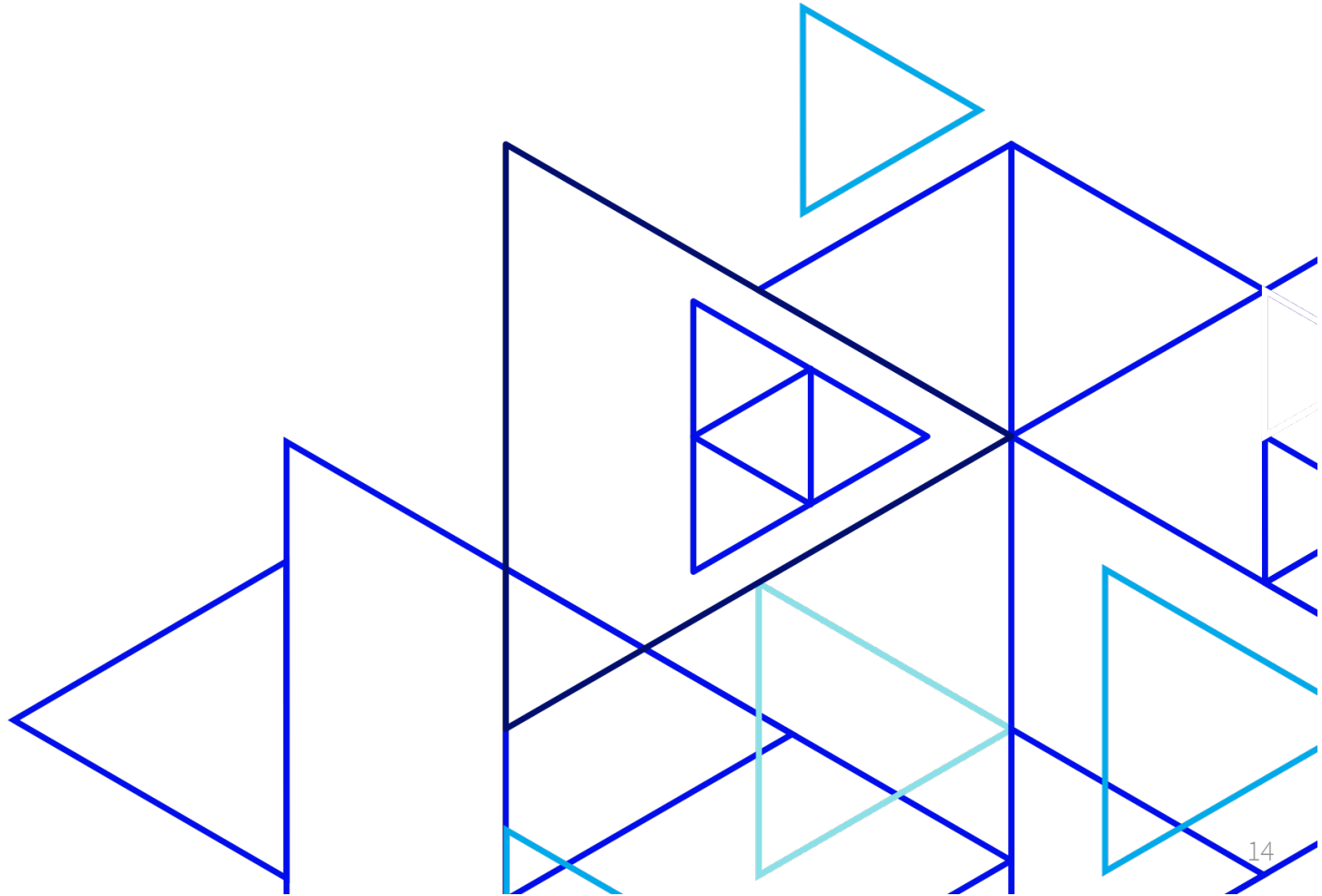


Overview

- ▶ Chaos
- ▶ Ticketing
- ▶ Schema as code
- ▶ CI/CD
- ▶ Inventory
- ▶ Automation
- ▶ What's next?

Chaos

FOSDEM
February 6, 2021



Who can ALTER my database?



Startup mindset

- ▶ Employees are all in the same openspace
- ▶ Developer come to the ops desk
- ▶ “Can you do X on my database, please?”
- ▶ Ops SSH to the production database host
- ▶ `psql` database
- ▶ Translate X to queries and run them
- ▶ Job done

Startup mindset

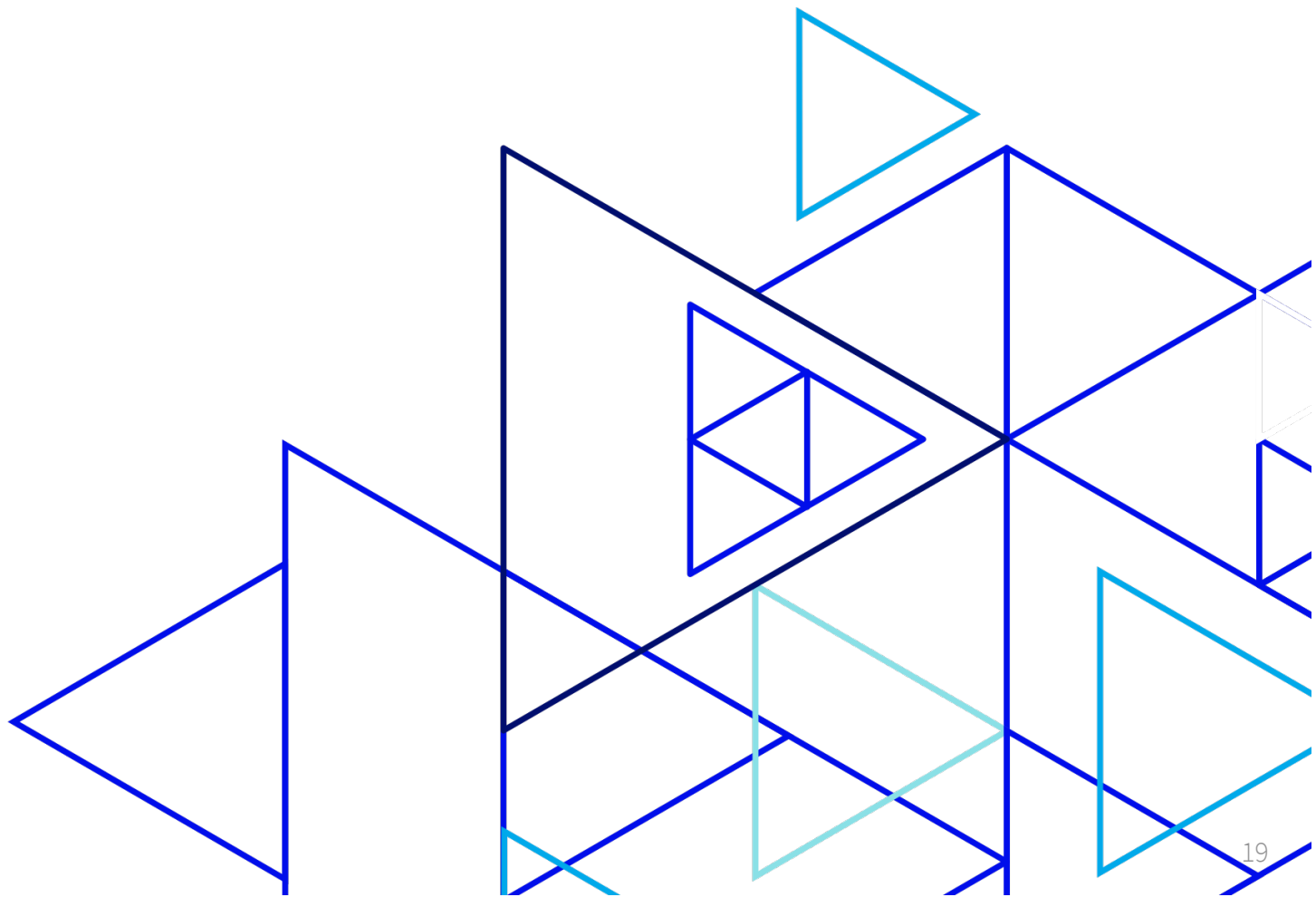
- ▶ Ops forgot on which host the database is
- ▶ Ops misunderstood what developer really wanted
- ▶ Ops executed the query on EU but forgot about Canada
- ▶ Ops forgot to create a transaction and one query failed in the middle
- ▶ Ops started a transaction but forgot to commit
- ▶ Transaction runs for too long and locks production
- ▶ Ops forgot “SET ROLE” and permissions are broken

Blame the ops!



Ticketing

FOSDEM
February 6, 2021



Let's write migrations in a ticketing system

► JIRA

The screenshot shows a JIRA ticket interface. At the top, the ticket title is "New table for [redacted] on [redacted]". Below the title are buttons for "Comment", "Agile Board", "More", and "Reopen Issue".

Details

- Type: Improvement
- Priority: Minor
- Component/s: DB
- Labels: None
- Templates:
- Otrs linked Tickets number: 0
- Status: **CLOSED** (View Workflow)
- Resolution: Fixed

People

- Assignee: [redacted]
- Reporter: [redacted]
- Votes: 0 Vote for this issue
- Watchers: 2 Start watching this issue

Dates

- Created: 12/Oct/15 14:35
- Updated: 27/Jun/16 11:41
- Resolved: 14/Oct/15 10:30

Description

Hello,

[redacted]

Can you create this table on Postgres EU+CA.
DB [redacted]
schema [redacted].

No need for archive table.

– MAIN TABLE

```
DROP TABLE IF EXISTS [redacted];
CREATE TABLE [redacted] (
  [redacted] SERIAL PRIMARY KEY,
  [redacted] INET NOT NULL,
  [redacted] VARCHAR(64),
  [redacted] VARCHAR(64),
  [redacted] VARCHAR(255),
```

Let's write migrations in a ticketing system

► OTRS

▼ Article #1 – provision POSTGRESQL db Created: 06/15/2016 17:56 (+1) by

Mark | Print

From:

Subject: provision POSTGRESQL db

To open links in the following article, you might need to press Ctrl or Cmd or Shift key while clicking the link (depending on your browser and OS). ✕

Hello, we need a new postgresql db named
Same characteristics as existing
With the following schema applied:
Thank you :)

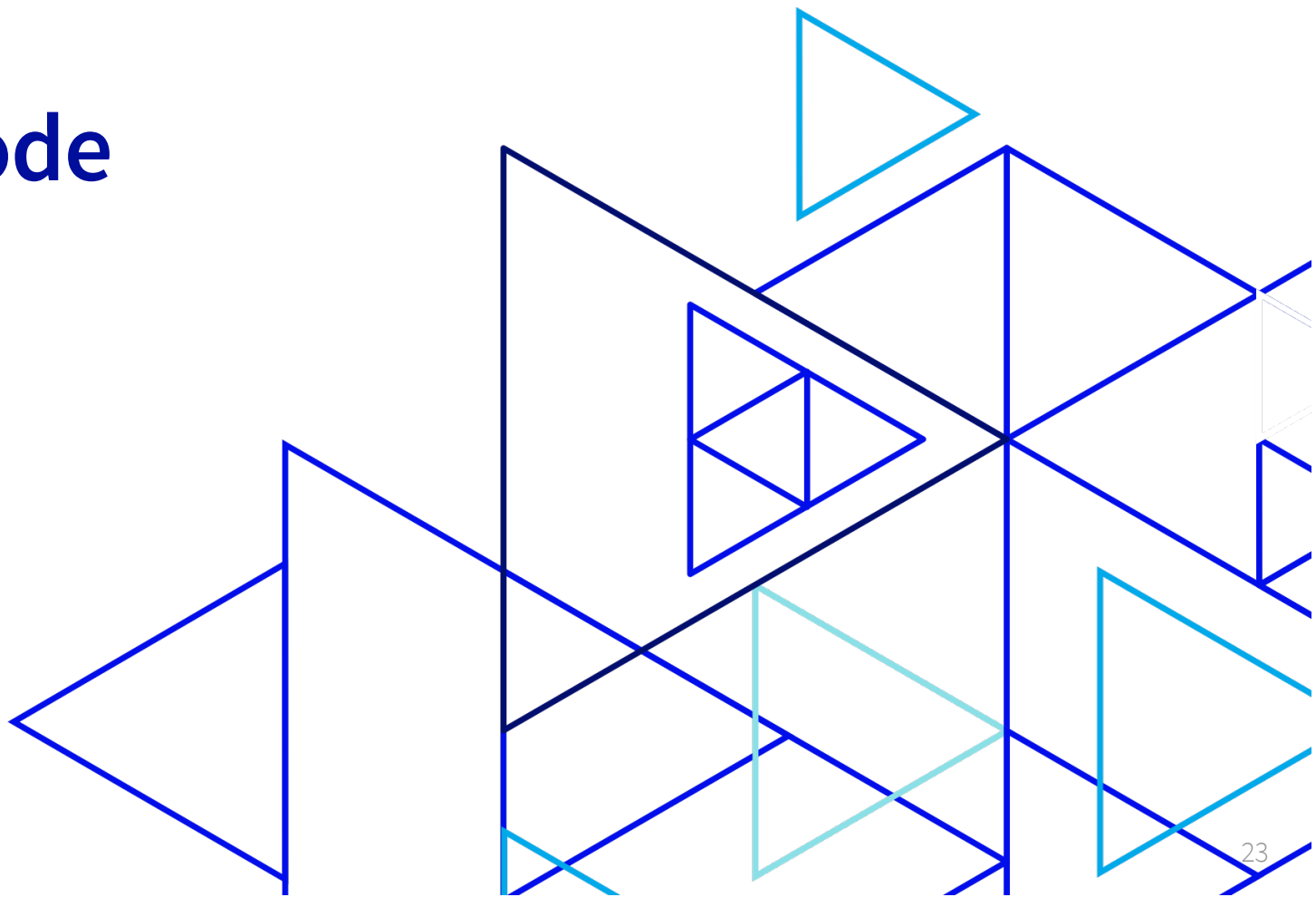
```
BEGIN;  
  
DROP TABLE IF EXISTS  CASCADE;  
DROP TABLE IF EXISTS  CASCADE;  
DROP TABLE IF EXISTS  CASCADE;  
DROP TABLE IF EXISTS  CASCADE;  
DROP TABLE IF EXISTS  CASCADE;  
DROP TABLE IF EXISTS  CASCADE;  
  
CREATE TABLE  (  
   BIGSERIAL PRIMARY KEY,  
   TEXT UNIQUE NOT NULL,  
   TEXT NOT NULL,  
   JSON NOT NULL,  
   JSON NOT NULL,  
   JSON,  
   JSON,  
   BOOL NOT NULL DEFAULT false,  
   BOOL NOT NULL DEFAULT false,  
   TEXT NOT NULL  
);
```

But

- ▶ Multiple ticketing systems for different teams
- ▶ SQL statements are not well formatted
- ▶ Doesn't prevent from bad copy pastes
- ▶ Poor reviewing system

Schema as code

FOSDEM
February 6, 2021



Version control

- ▶ **Git**

- ▶ One repository for one or more databases

- ▶ Well known by developers

Schema migrations

- ▶ **sql-migrate**
- ▶ “SQL schema migration tool for Go”
- ▶ <https://github.com/rubenv/sql-migrate>
- ▶ MIT license

Schema migrations

- ▶ Developers manage code

```
-- +migrate Up
```

```
create table x (...);
```

```
-- +migrate Down
```

```
drop table x;
```

Schema migrations

- ▶ Ops deploy the change with a CLI command
- ▶ Wrapper created to handle “SET ROLE”
- ▶ Two migration paths:
 - “admin” → run as superuser to create extensions
 - “normal” → run with DDL privileges to create objects

Schema migrations

```
# sql-migrate-wrapper status -config /etc/sqlmigrate/database.json
INFO[2021-01-06T15:01:56+01:00] Processing 'database' database
INFO[2021-01-06T15:01:56+01:00] Processing admin migration path
+-----+-----+
| MIGRATION |          APPLIED          |
+-----+-----+
| v0001.sql | 2017-09-12 12:35:44.175971 +0200 CEST |
+-----+-----+
INFO[2021-01-06T15:01:56+01:00] Processing normal migration path
+-----+-----+
| MIGRATION |          APPLIED          |
+-----+-----+
| v0001.sql | 2017-09-12 12:35:44.175971 +0200 CEST |
| v0002.sql | 2017-09-12 12:35:44.175971 +0200 CEST |
| v0003.sql | 2017-09-12 12:35:44.175971 +0200 CEST |
...
| v0042.sql | 2020-12-16 12:40:44.013301 +0100 CET   |
+-----+-----+
```

Schema migrations

- ▶ One file → one migration
- ▶ One migration → one transaction
- ▶ Implicit transaction by default
- ▶ But can be disabled with “notransaction”:
 - ALTER TYPE... ADD VALUE...
 - CREATE INDEX CONCURRENTLY...
- ▶ We focus on schema (DDL), not data (DML)

Code reviews



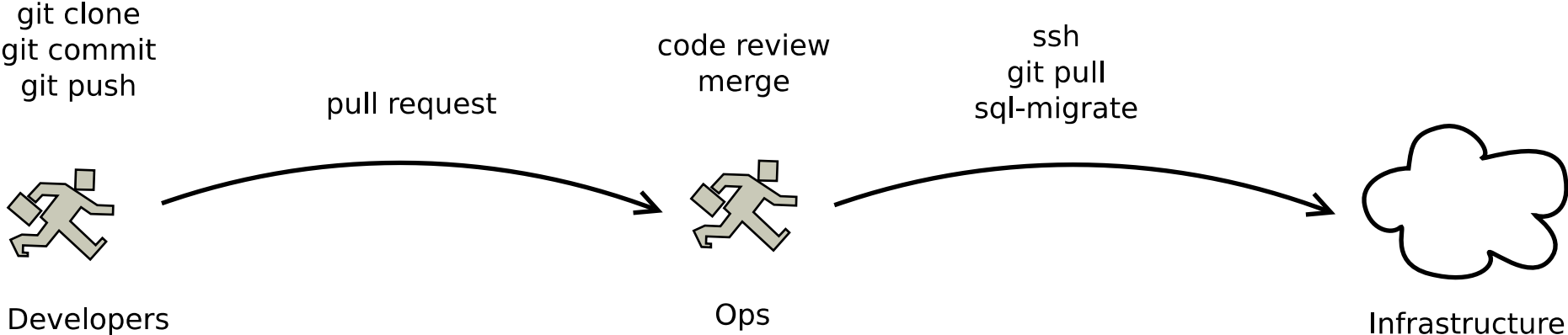
- ▶ Git based collaboration tool
- ▶ <https://bitbucket.org>
- ▶ Adopted by the company
- ▶ Not opensource

Code reviews with Bitbucket

The screenshot displays the Bitbucket web interface for a pull request. The top navigation bar includes the Bitbucket logo, 'Projects', 'Repositories', and a search bar. The main content area shows a pull request titled 'Add [redacted] migration'. The 'Diff' tab is active, showing a comparison of the 'v0001.sql' file. The diff view highlights the changes in a light green background. The code is a PostgreSQL migration script with the following structure:

```
1 + -- Postgres schema
2 +
3 + -- +migrate Up
4 + CREATE TYPE [redacted] AS ENUM ([redacted])
5 +
6 + CREATE TABLE [redacted] (
7 +     [redacted] UUID PRIMARY KEY,
8 +     [redacted] timestamp with time zone,
9 +     [redacted] timestamp with time zone,
10 + [redacted] timestamp with time zone,
11 + [redacted] timestamp with time zone,
12 + [redacted] bigint NOT NULL,
13 + [redacted] bigint NOT NULL,
14 + [redacted] bigint NOT NULL,
15 + [redacted] int NOT NULL,
16 + [redacted] int NOT NULL,
17 + [redacted] [redacted] NOT NULL DEFAULT [redacted],
18 + [redacted] jsonb NOT NULL,
19 + [redacted] bigint[] NOT NULL,
20 + [redacted] jsonb NOT NULL
21 + );
22 +
23 + CREATE INDEX [redacted] ON [redacted] ([redacted]);
24 + CREATE INDEX [redacted] ON [redacted] ([redacted]);
25 + CREATE INDEX [redacted] ON [redacted] ([redacted]);
26 +
27 + -- +migrate Down
28 + DROP INDEX
29 + [redacted]
30 + [redacted]
31 + [redacted]
32 + DROP TABLE [redacted];
33 + DROP TYPE [redacted];
```

Workflow



What's good

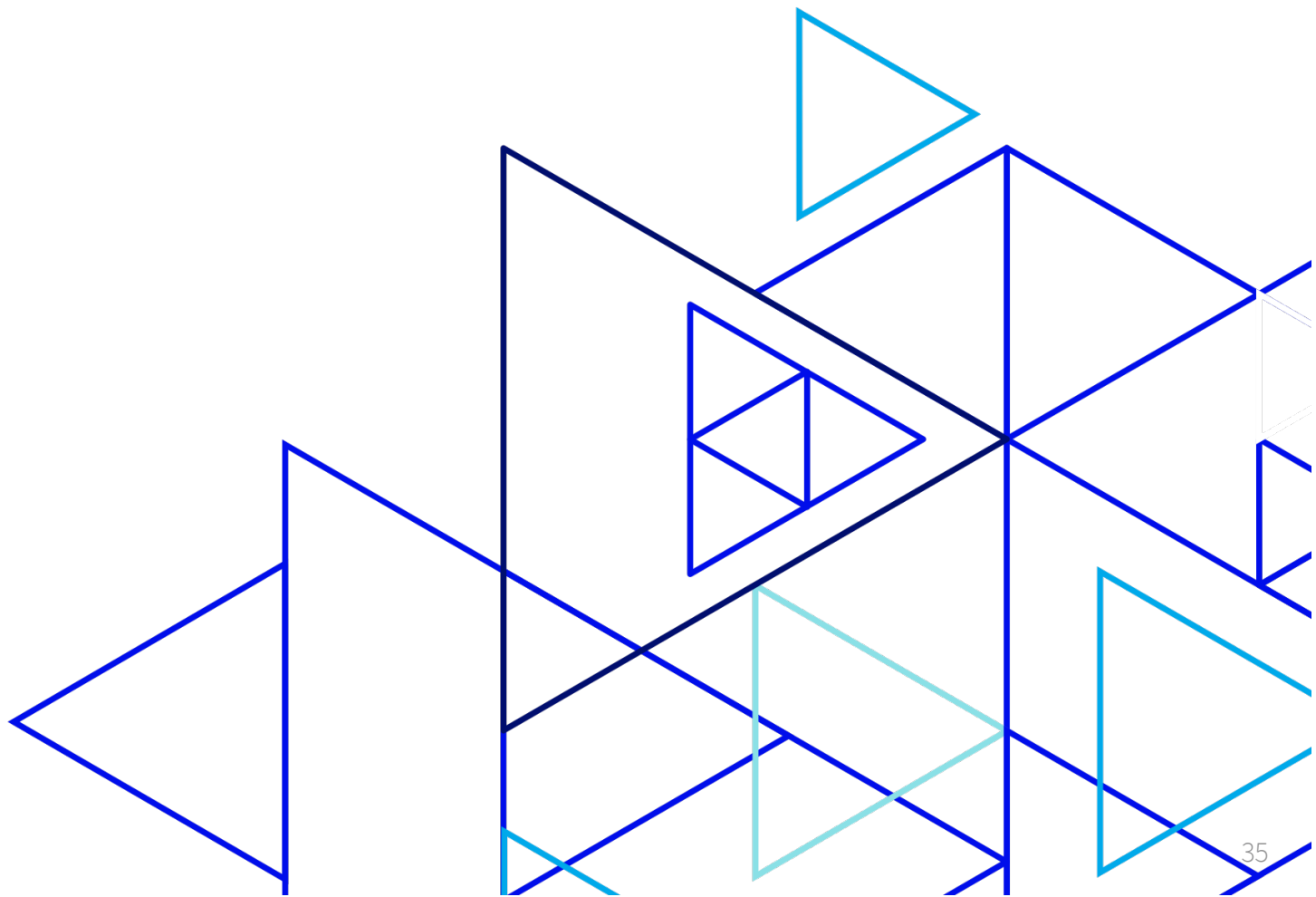
- ▶ No more copy/paste typos
- ▶ Developers are familiar with version control and code reviews
- ▶ Better change history
- ▶ Better reviewers system
- ▶ Better centralization (one tool)

What's bad

- ▶ Multiple migration paths are hard to manage
- ▶ Highly concurrent repositories conflict a lot
- ▶ Focus on “**how** to go from version 1 to version 2” instead of “**what** is the expected schema”
- ▶ Lots of errors seen at execution time

CI/CD

FOSDEM
February 6, 2021



Continuous integration

- ▶ Test locally
- ▶ Ensure software is deliverable
- ▶ Merge changes to the main branch

Continuous delivery

- ▶ Build a release

Continuous deployment

- ▶ Build a release and deploy it
- ▶ No manual intervention

Continuous integration with CDS

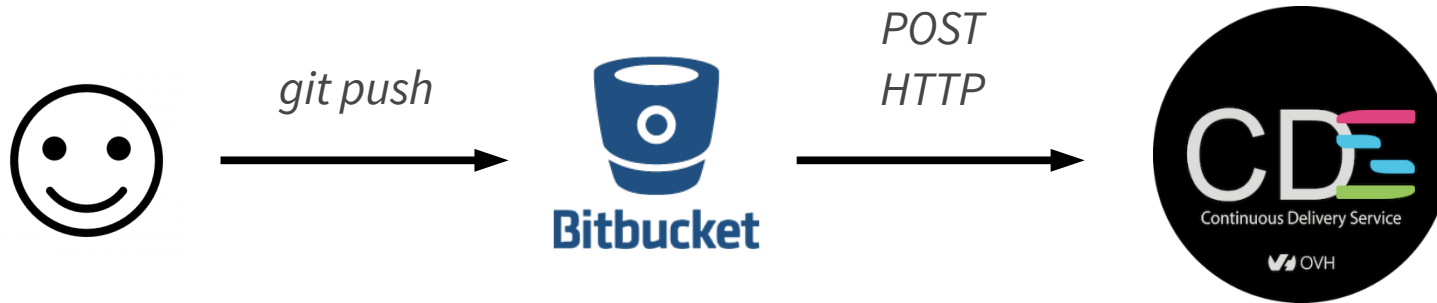


- ▶ “CDS is an Enterprise-Grade Continuous Delivery & DevOps Automation Open Source Platform.”
- ▶ <https://ovh.github.io/cds/>
- ▶ Open source!

Continuous integration with CDS

- ▶ **Project** → all schema tests at the same place
- ▶ **Application** → one git repository (schema)
- ▶ **Pipeline** → run sql-migrate on PostgreSQL 13
- ▶ **Workflow** → link an application to pipeline(s)

Continuous integration with CDS



Continuous integration with CDS

The screenshot displays the CDS web interface. At the top, there's a navigation bar with 'Projects', 'Bookmarks', and 'Create a project'. A search bar is on the right. Below the navigation, a breadcrumb trail shows the current project path. On the left, a list of pipeline runs is shown, with the most recent one (#676) highlighted. The main area shows the details of a specific pipeline run (#650.0) for 'sql-migrate-postgresql-9.6', which started on 04/01/2021 at 15:32 and took 15m58s. A 'Run pipeline with parameters' button is visible. Below the pipeline details, a diagram shows the pipeline structure: Stage 1 (offline, 5s) leads to Stage 2, which consists of three parallel tasks: 'eralchemy' (50s), 'online' (15m17s, with a warning icon), and 'online PG13' (15m23s). At the bottom, an 'Information' panel shows the execution logs, including job queuing, worker spawning, and successful completion of the migration steps.

CD Projects Bookmarks Create a project Search... template as code star

Select... #676 dev/ 6m52s 06/01/2021 15:50 #675 dev/ 13m30s 06/01/2021 14:39 #674 dev/ 11m41s 06/01/2021 14:00 #673 dev/ 11m12s 06/01/2021 13:46 #672 dev/ 9m42s 06/01/2021 13:02 #671 dev/ 1m22s 06/01/2021 12:48 #670 dev/ 9m59s 06/01/2021 11:13 #669 dev/ 10m1s 05/01/2021 19:01 #668 dev/ 10m58s 05/01/2021 17:05 #667 dev/ 1m30s 05/01/2021 16:44

sql-migrate-postgresql-9.6 04/01/2021 15:32 15m58s Run pipeline with parameters

Pipeline 1 Commit Vulnerabilities Test Artifacts (7) History (1)

Stage 1 ✓ offline 5s

Stage 2 ✓ eralchemy 50s ✓ online 15m17s ✓ online PG13 15m23s

Information Display job variables

```
[2021-01-04T14:32:35] ✓ Job has been queued
[2021-01-04T14:32:40] Hatchery gral-prod-swarm-c2 starts spawn worker with model shared.infra/sql-migrate-wrapper
[2021-01-04T14:32:41] Hatchery gral-prod-swarm-c2 starts docker pull :latest...
[2021-01-04T14:32:42] Hatchery gral-prod-swarm-c2 docker pull :latest done
[2021-01-04T14:32:44] Hatchery gral-prod-swarm-c2 spawn worker gral-prod-swarm-c2-shared-infra-sql-migrate-wrapper-sweet-moore successfully in 3s
[2021-01-04T14:32:50] Worker gral-prod-swarm-c2-shared-infra-sql-migrate-wrapper-sweet-moore version:0.47.0-263+sha.g9925560dd.cds.15096 os:linux arch:amd64
[2021-01-04T14:32:56] Job 10987663 was taken by worker gral-prod-swarm-c2-shared-infra-sql-migrate-wrapper-sweet-moore
[2021-01-04T14:32:56] ✓ Worker gral-prod-swarm-c2-shared-infra-sql-migrate-wrapper-sweet-moore finished working on this job and took 6s to work on the steps
```

✓ OVH_iaasdb.offline.pg-9.6 (5s)

Continuous integration with CDS



🔄 1 build ✓



🔄 1 build failed ❌



What's good

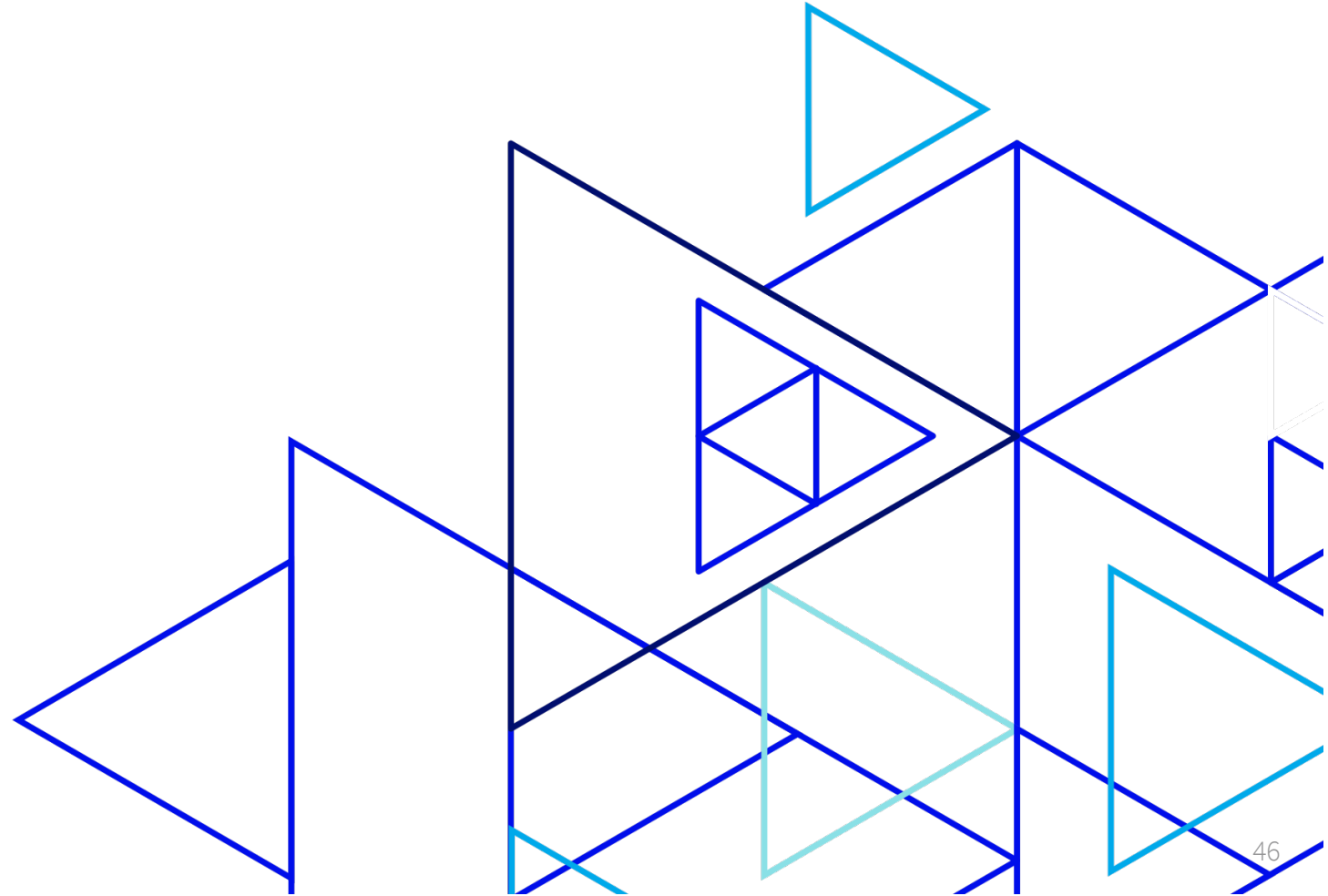
- ▶ Test early
- ▶ Time and efforts saved

What's bad

- ▶ Under active development
- ▶ Not enough trust to use continuous deployment feature on production databases

Inventory

FOSDEM
February 6, 2021



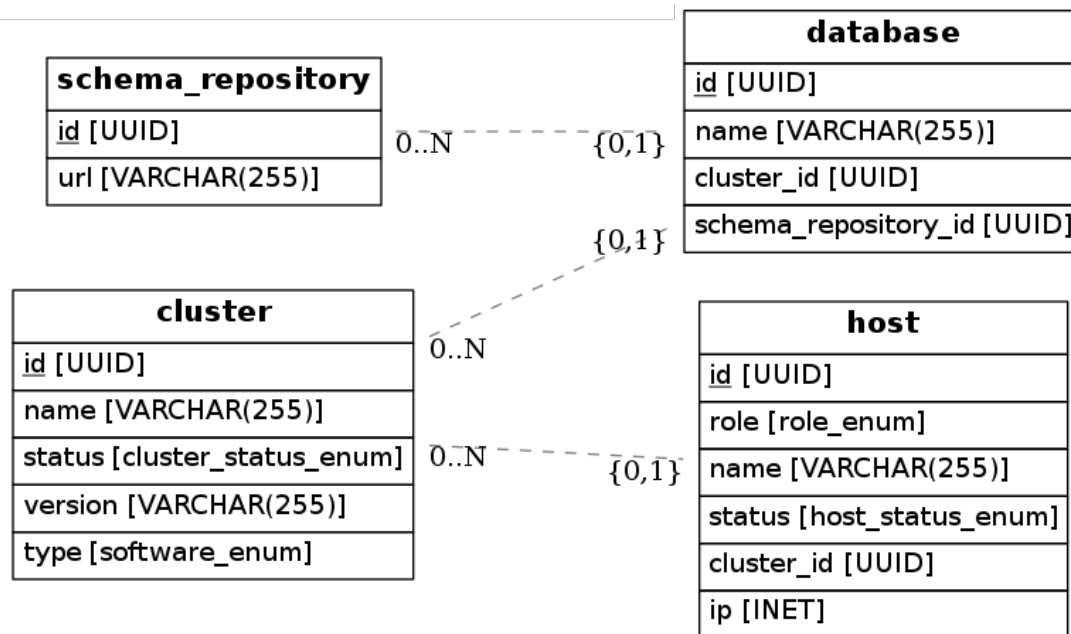
Inventory

- ▶ What databases are behind this git repository?
- ▶ Where is the database?



Inventory

- In a database of databases!



Inventory

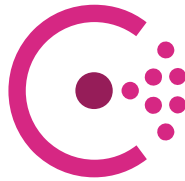
► With a REST API

```
$ curl https://<api>/database?name=test
{
  "code": 200,
  "data": [
    {
      "schemarepository": "<uuid>",
      "name": "test",
      "id": "<uuid>",
      "cluster": "<uuid>"
    }
  ]
}
```

Inventory, but...

- ▶ Every change has to be declared
 - Add a cluster, host, database, git repository, ...
 - Remove a cluster, host, database, git repository, ...
 - Easy to miss an event
- ▶ Fully home-made

Service discovery with Consul



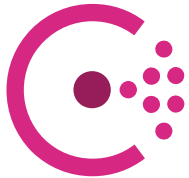
- ▶ Distributed and highly-available data store
- ▶ Local agent
- ▶ Open source!
- ▶ <https://github.com/hashicorp/consul>

Service discovery with Consul

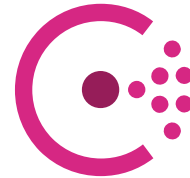
Hello, my name is “node1”,
part of “cluster1” and I
have [“db1”, “db2”]
databases



node



agent



server

Service discovery with Consul

- ▶ **Nodes** → name, IP address, meta(data)
- ▶ **Services** → cluster, databases
- ▶ **Key/value store**
- ▶ Access control list (ACL)
- ▶ Encryption

Consul – Nodes and services

- ▶ Static configuration
 - Node meta
 - Cluster service
- ▶ Dynamic configuration (local discovery)
 - Node “subrole” (primary, replica)
 - Database services
 - Reload agent configuration on changes

Consul – Nodes and services

- ▶ Where is my database?

test

Instances Intentions Routing Tags

test

✓ All node checks passing ↗ patroni-99-backup-1 [redacted] ⚙ backup

test

✓ All node checks passing ↗ patroni-99-node-1 [redacted] ⚙ primary

test

✓ All node checks passing ↗ patroni-99-node-2 [redacted] ⚙ replica

Consul – Nodes and services

► Where is my cluster?

cluster99

Instances Intentions Routing Tags

cluster99

✓ All service checks passing ✓ All node checks passing ↗ patroni-99-backup-1 10.10.10.10:5050 ⚙ backup

cluster99

✓ All service checks passing ✓ All node checks passing ↗ patroni-99-lb-1 10.10.10.10:80 ⚙ lb

cluster99

✓ All service checks passing ✓ All node checks passing ↗ patroni-99-lb-2 10.10.10.10:80 ⚙ lb

cluster99

✓ All service checks passing ✓ All node checks passing ↗ patroni-99-node-1 10.10.10.10:5050 ⚙ node

cluster99

✓ All service checks passing ✓ All node checks passing ↗ patroni-99-node-2 10.10.10.10:5050 ⚙ node

Consul – Key/value store

- ▶ Used to manage **repository / databases relationship**
- ▶ File containing databases list in each repository
- ▶ Loop over bitbucket repositories, parse this file then update the key/value store
- ▶ Key = repository URL, value = databases list
- ▶ Run as cron job

Consul – Key/value store

< Key/Values / `inventory` / repositories / `ssh%3A%2F%2F%40%3A%2F%2F.git`

databases

Value

Code

```
1 ["_dev", ""]
```

JSON

Save

Cancel

Delete

Consul – Automatic retention

- ▶ Focus on live state
- ▶ Deregister dead nodes after 72 hours (not configurable)
- ▶ Deregister dead services
- ▶ Except the key/value store

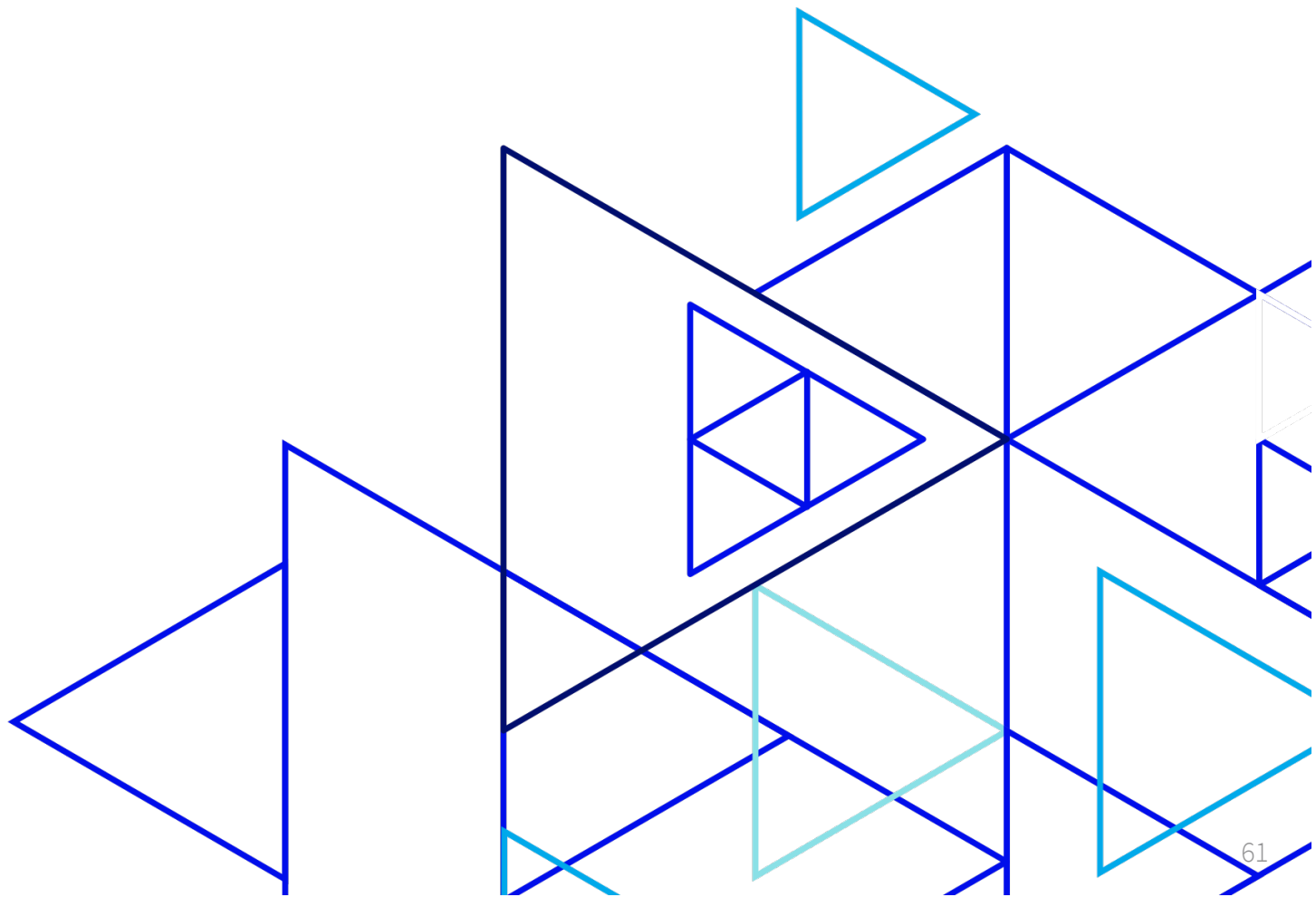
Service discovery

- ▶ No more manual intervention
- ▶ Only one source of truth



Automation

FOSDEM
February 6, 2021

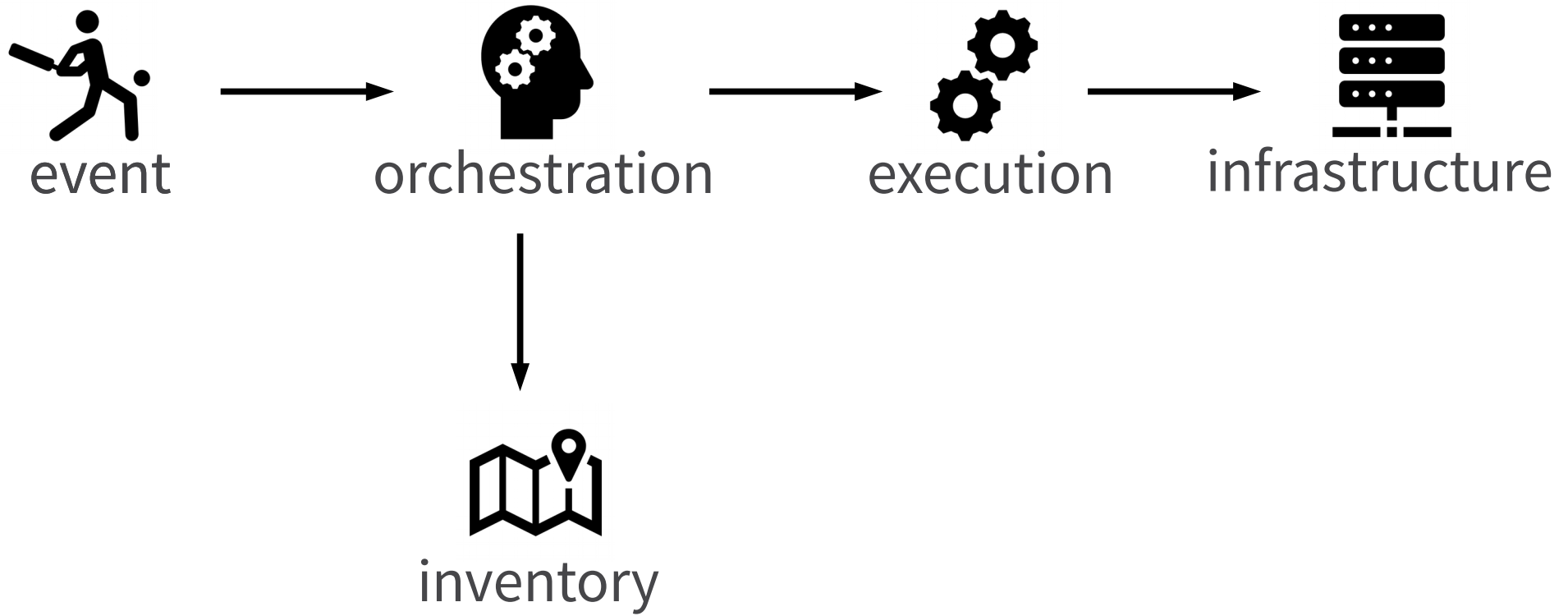


Automation

- ▶ How migrations can be applied automatically?



Automation



Orchestration

- ▶ Business logic
 - Look for databases → clusters → hosts → IP addresses
 - Talk to the “execution” to run schema migrations
- ▶ Custom development with standard libraries
 - HTTP API with **Flask**
 - Job scheduling with **Celery**
 - Web interface with **Celery Flower**

Orchestration

Celery Flower [Dashboard](#) [Tasks](#) [Broker](#) [Monitor](#) [Logout](#) [Docs](#) [Code](#)

Active: 0 Processed: 91 Failed: 1 Succeeded: 90 Retried: 0

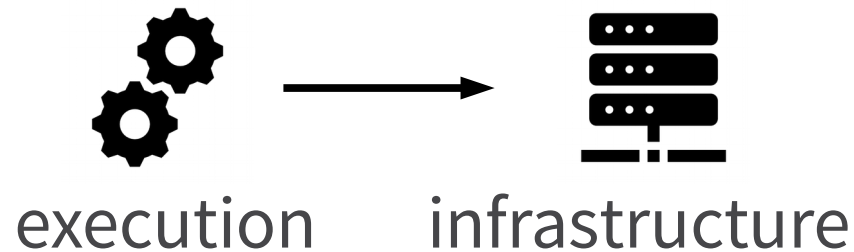
Shut Down

Worker Name	Status	Active	Processed	Failed	Succeeded	Retried	Load Average
celery@390d4e594c1d	Online	0	91	1	90	0	0.74, 0.63, 0.57

Showing 1 to 1 of 1 entries

Execution

- ▶ Execute **Ansible** playbooks
- ▶ Similar stack as the orchestration
 - HTTP API, job scheduling, web interface
- ▶ Closed to the infrastructure for command latency



Ansible playbook example – Overview

```
---  
- name: update database to the latest schema migration  
  hosts: all  
  pre_tasks:  
    - name: check variable schema_url      # fail fast  
    - name: check variable database_name # fail fast  
    - name: gather host facts  
  tasks:  
    - name: update schema  
      when: subrole == "primary"  
      block:  
        - name: create sql-migrate directories  
        - name: create sql-migrate configuration file  
        - name: clone schema  
        - name: run migration
```

Ansible playbook example – Details (1/2)

- name: **create sql-migrate directories**
file:
 path: "{{ item }}"
 state: directory
loop:
 - /etc/sqlmigrate
 - /var/lib/sqlmigrate
- name: **create sql-migrate configuration file**
template:
 src: "./templates/sqlmigrate/database.json.j2"
 dest: "/etc/sqlmigrate/{{ database_name }}.json"

Ansible playbook example – Details (2/2)

- name: **clone schema repository**
git:
 repo: "{{ schema_url }}"
 dest: "/var/lib/sqlmigrate/{{ database_name }}"
 version: "{{ branch|default('main') }}" # branch or tag
 force: true
environment:
 TMPDIR: /run
- name: **run migration**
command: /usr/bin/sql-migrate-wrapper up -config /etc/sqlmigrate/{{ database_name }}.yaml

Ansible playbook command

```
$ ansible-playbook \  
  --extra-vars \  
    '{"database_name":"database",  
     "schema_url":"ssh://git@hostname:port/project/repository.git"  
    }' \  
  schema-update.yml
```

What's good

- ▶ Fully automated
- ▶ Asynchronous
- ▶ Scalable
- ▶ It works
 - Used by OVHcloud Enterprise Cloud Databases
 - Used by OVHcloud internal databases

What's bad

- ▶ Hard to maintain (home made)
- ▶ Lots of components
- ▶ Ansible is hard to secure
 - Variable script name at execution (sudo “ALL”)
 - Direct SSH connection by default

Let's improve...

- ▶ The inventory management
- ▶ The “execution” stack
- ▶ The “orchestration” stack
- ▶ The Ansible security

Ansible + Consul

- ▶ Build Ansible inventory using Consul
- ▶ Support node meta, service and service tags
 - Meta: “*key_value*” (“*subrole_primary*”)
 - Service: “*name*” (“*customers*”), “*name_tag*” (“*customers_primary*”)
- ▶ Parse booleans (“*replica*” instead of “*replica_true*”, absent when value is “*false*”)
- ▶ <https://github.com/wilfriedroset/consul-awx> (MIT)

Ansible + Consul

```
---  
- name: update database to the latest schema migration  
  hosts: subrole_primary  
  pre_tasks:  
    - name: check variable schema_url    # fail fast  
    - name: check variable database_name # fail fast  
  tasks:  
    - name: create sql-migrate directories  
    - name: create sql-migrate configuration file  
    - name: clone schema  
    - name: run migration
```

Inventory management



Ansible AWX

- ▶ Ansible orchestration
- ▶ REST API, web interface, CLI
- ▶ Notifications
 - Compatible with OpsGenie for alerting
 - Compatible with Webex Teams for instant message
- ▶ Open source!
- ▶ <https://github.com/ansible/awx>



Ansible AWX

- ▶ Organization, projects, teams, users, privileges
- ▶ Inventory source
- ▶ Source Control (Git) and Machine (SSH) credentials
- ▶ Job templates
- ▶ Scheduled jobs
- ▶ Notification templates

Ansible AWX

► Manually update a schema

```
$ awx -f human job_templates launch --monitor \  
  --extra-vars \  
    '{"database_name": "database",  
     "schema_url": "ssh://git@hostname:port/project/database.git"}' \  
  database-primary-schema-update
```

► Show job logs from the past

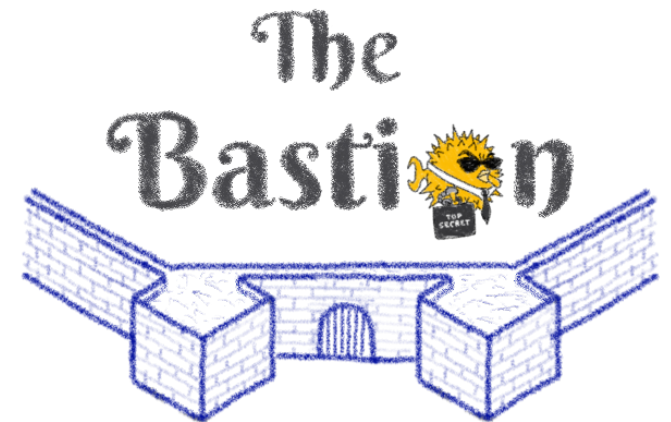
```
$ awx -f human jobs list --filter 'id, name, status, started, finished, extra_vars'  
$ awx job stdout <id>
```

Orchestration and execution



Ansible + The Bastion

- ▶ Administration via SSH gateway
- ▶ Fine-grained access to infrastructure
- ▶ Sessions can be recorded
- ▶ Used in secure environments
- ▶ Already used by humans
- ▶ <https://github.com/ovh/the-bastion> (Apache 2.0)



Ansible + The Bastion

- ▶ Compatible with Ansible using SSH/SCP commands

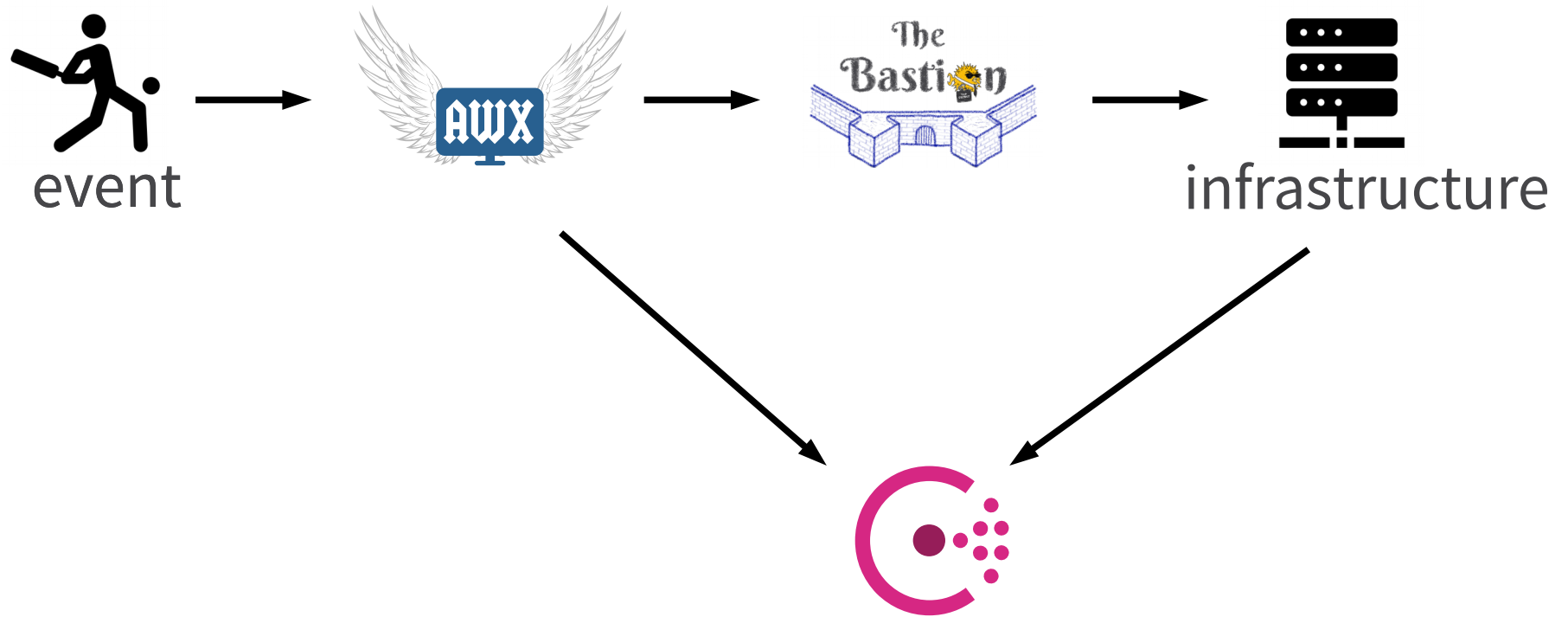
```
[ssh_connection]
scp_if_ssh = True
pipelining = True
private_key_file = ~/.ssh/id_rsa
ssh_executable = ./extra/bastion/sshwrapper.py
scp_executable = ./extra/bastion/scpbastion.sh
transfer_method = scp
```

- ▶ Wrapper not opensource... yet!

Security

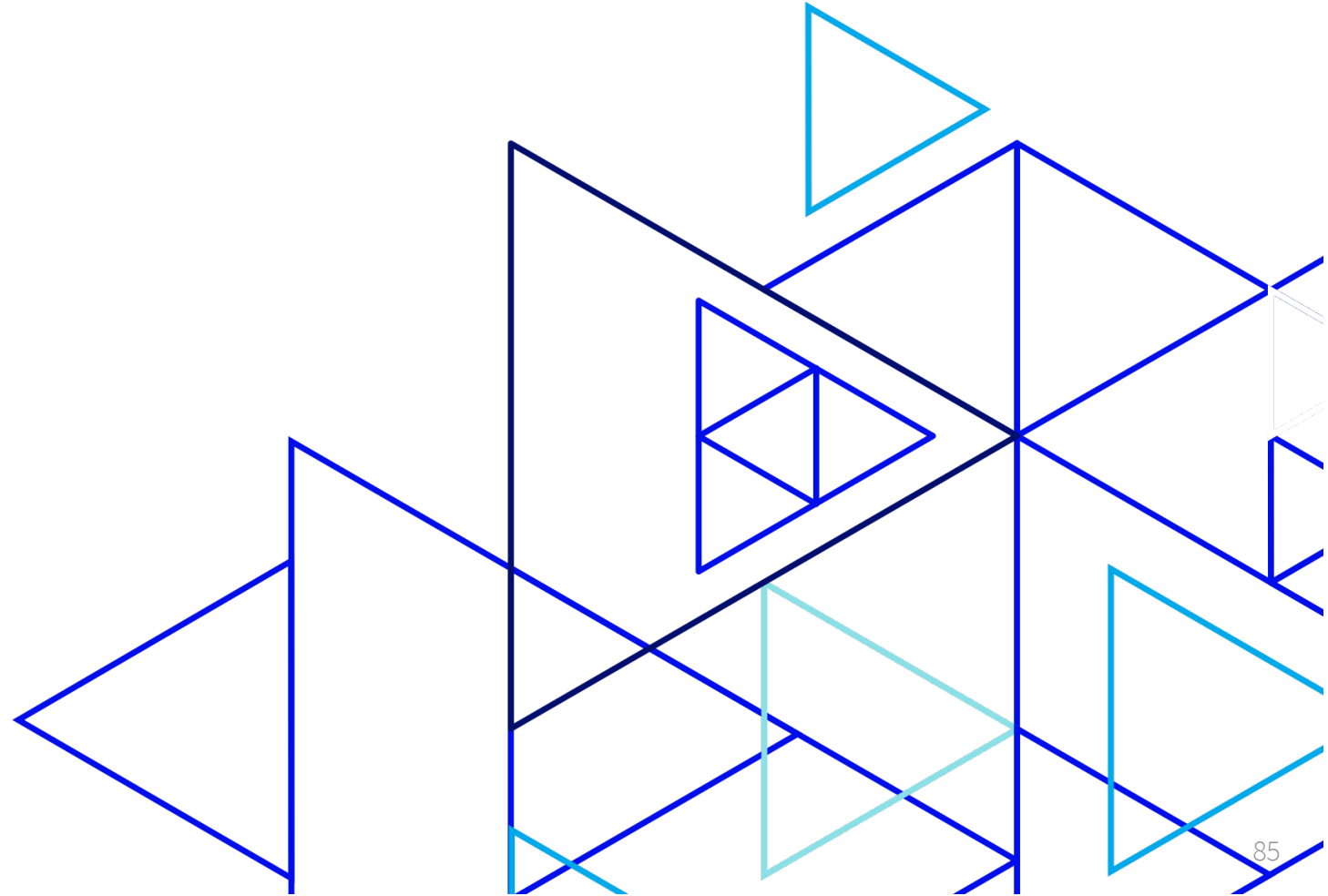


Architecture



Heaven

FOSDEM
February 6, 2021



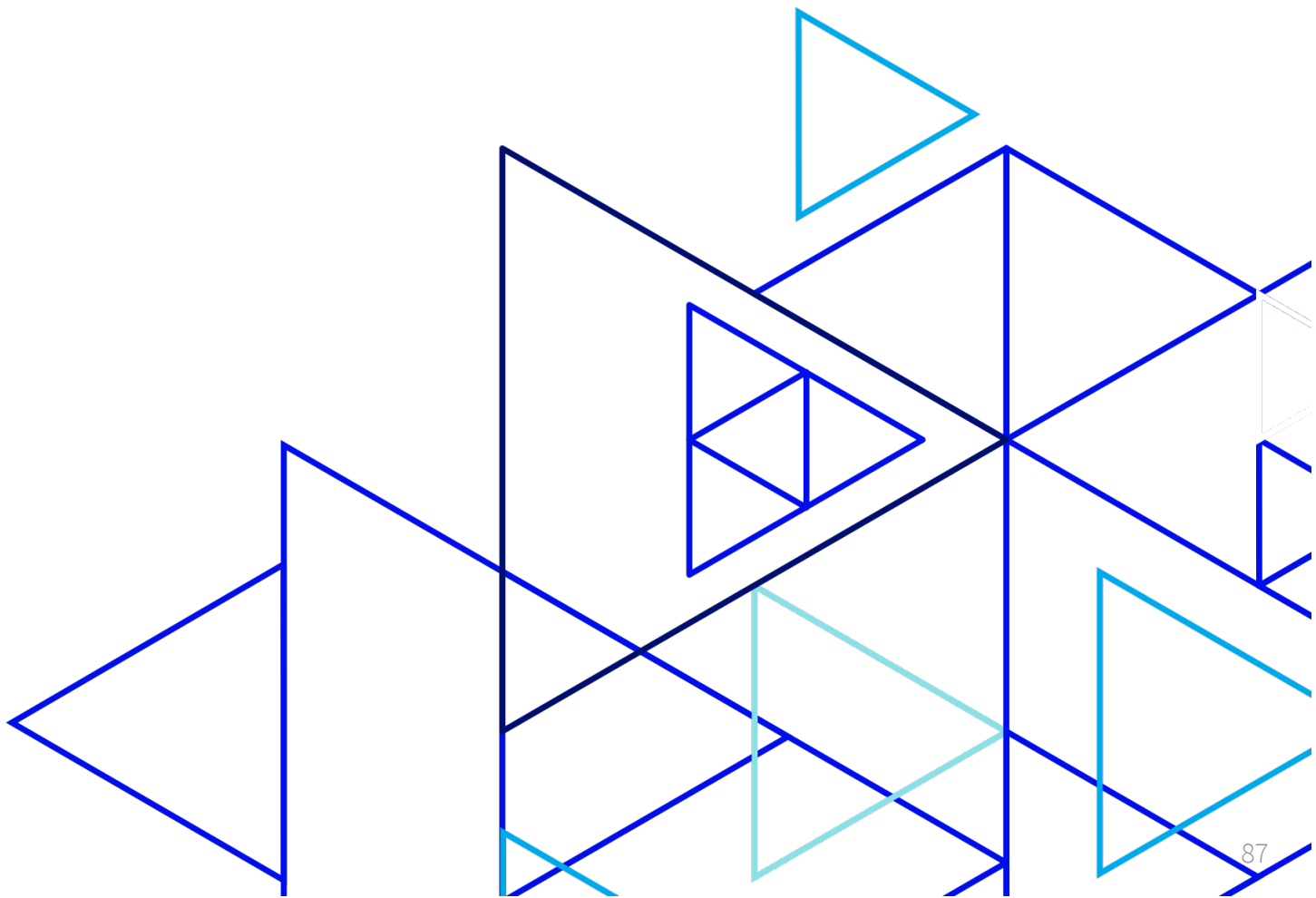
Heaven

- ▶ Code review
- ▶ Click on “merge”
- ▶ Job done



What's next?

FOSDEM
February 6, 2021



The future

- ▶ Use more standard products
 - Replace sql-migrate-wrapper by sql-migrate
- ▶ Solve the repository concurrency problem
 - Use a “declarative” schema migration tool
- ▶ Reduce long-running migrations lock time
- ▶ Use only one DBMS

In other communities



- ▶ Database schema management for MySQL
- ▶ Version control style like “git”
- ▶ <https://github.com/skeema/skeema>

In other communities



PERCONA
Toolkit

pt-online-schema-change

- ▶ Online schema change for MySQL
- ▶ Based on triggers
- ▶ Controllable
- ▶ “ALTER TABLE” fully supported
- ▶ <https://www.percona.com/doc/percona-toolkit/3.0/pt-online-schema-change.html>

In other communities



- ▶ Online schema change for MySQL
- ▶ Based on logical replication
- ▶ Highly controllable
- ▶ “ALTER TABLE” fully supported
- ▶ <https://github.com/github/gh-ost>

Useful resources

- ▶ “Automating schema migration flow with Github Actions, skeema & gh-ost”
https://archive.fosdem.org/2020/schedule/event/mysql_github_schema/
- ▶ “Changing your huge table's data types in production”
https://fosdem.org/2021/schedule/event/postgresql_changing_your_huge_tables_data_types_in_production/
- ▶ “Challenges of Concurrent DDL - Robert Haas”
<https://www.youtube.com/watch?v=kbtKKh9B7eo>
- ▶ “Transparent Logical DDL Replication (pgl_ddl_deploy)”
https://github.com/enova/pgl_ddl_deploy
- ▶ Schema comparison tool “pgquarrel” <https://github.com/eulerto/pgquarrel>



OVHcloud

Thank you

