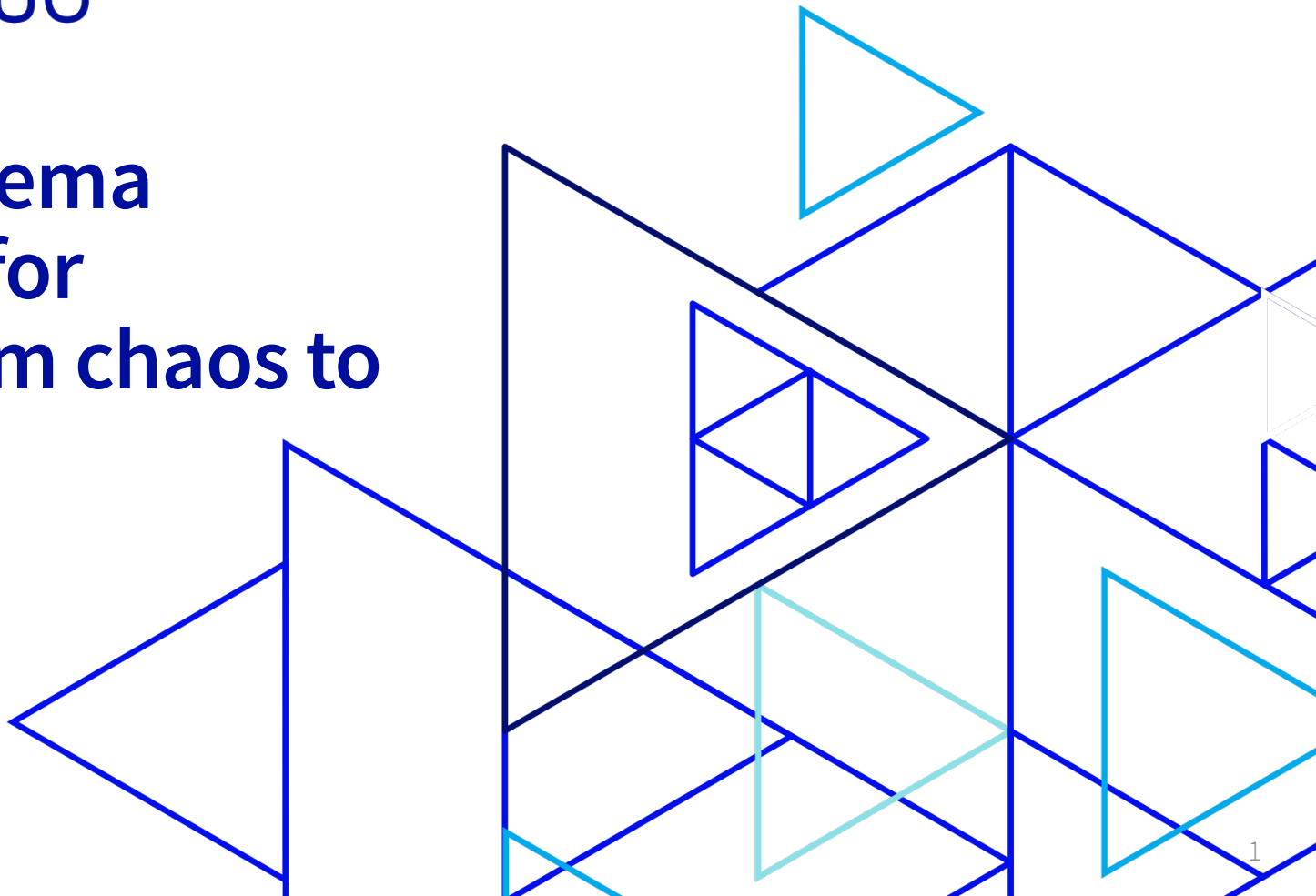
OVHcloud

Databases schema management for lazybones: from chaos to heaven

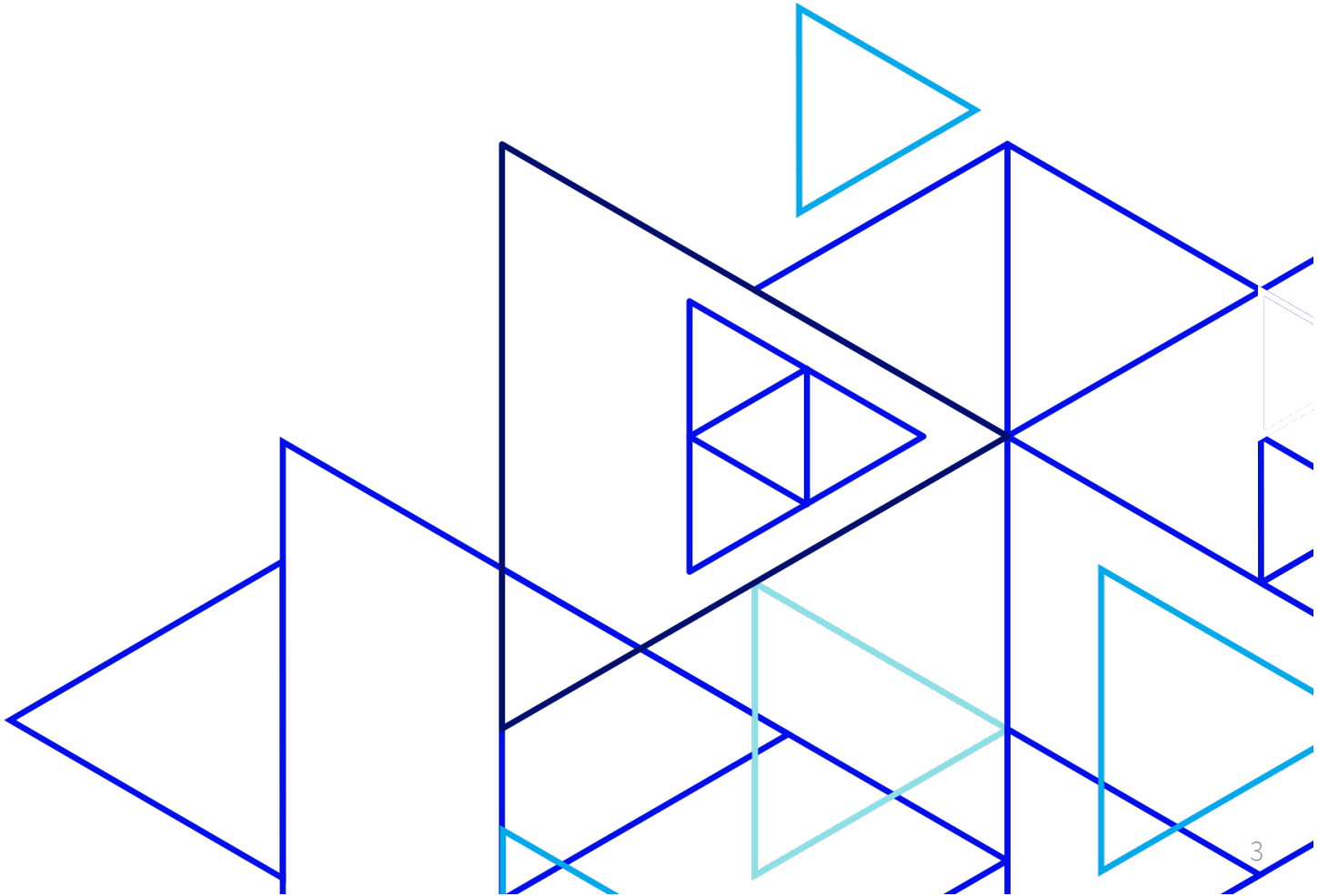
Julien Riou
FOSDEM
February 6, 2021



Speaker

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

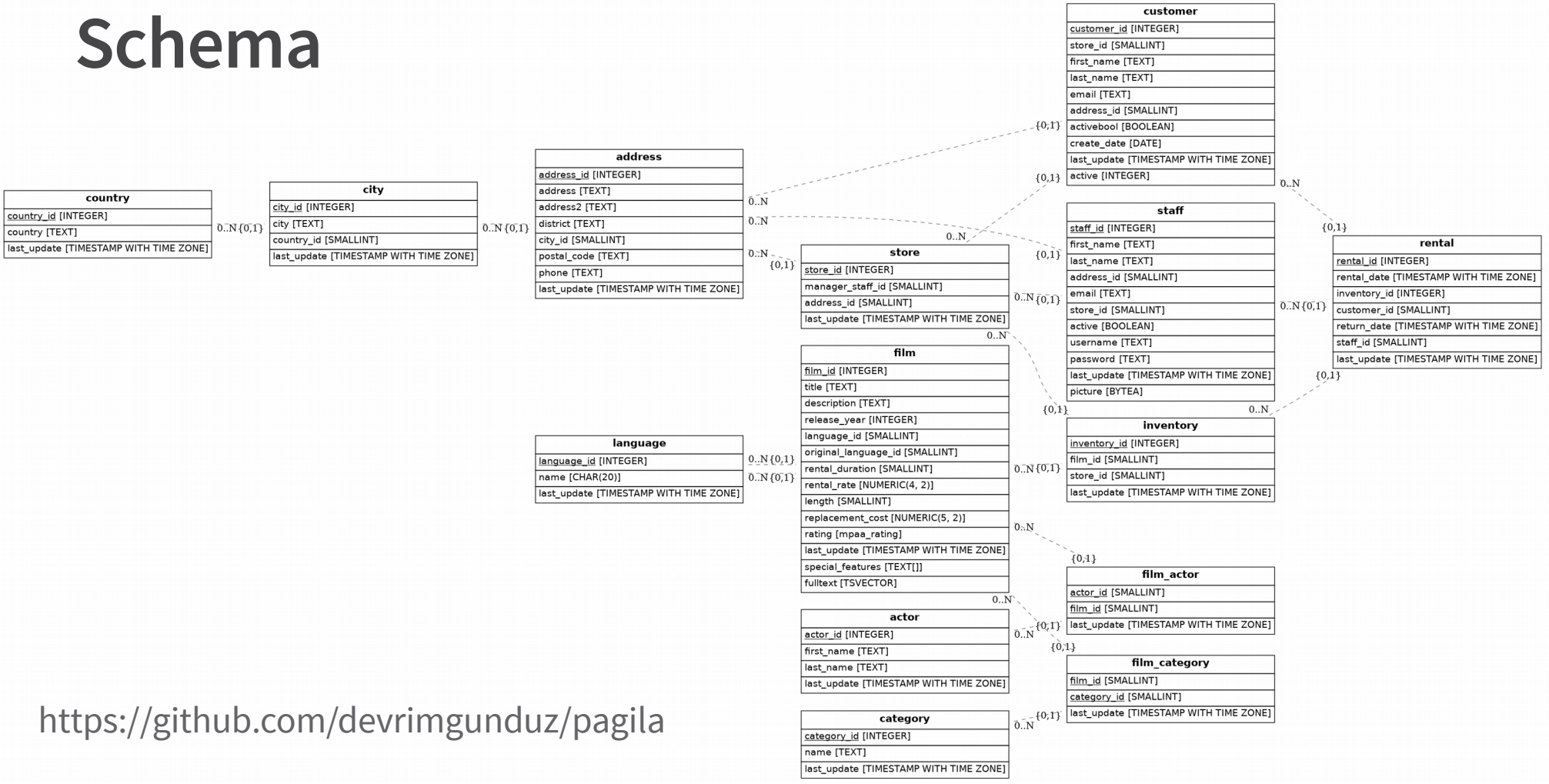
Definitions



FOSDEM
February 6, 2021

OVHcloud

Schema

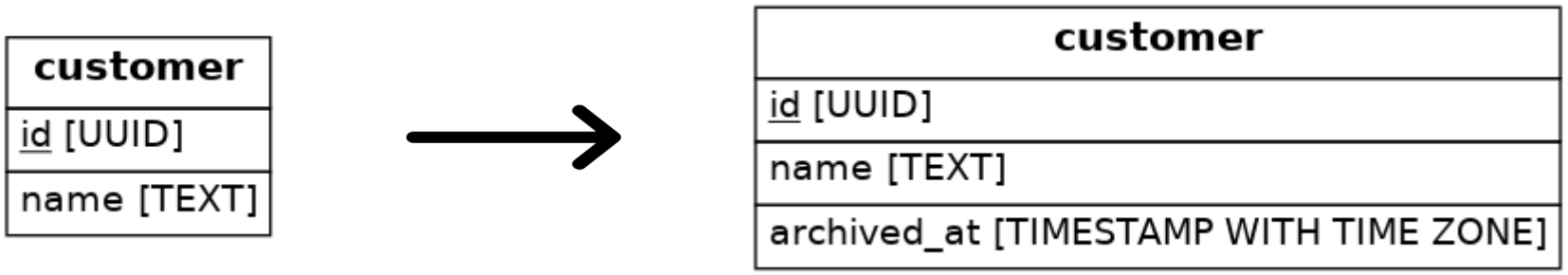


https://github.com/devrimgunduz/pagila

Schema changes over time

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

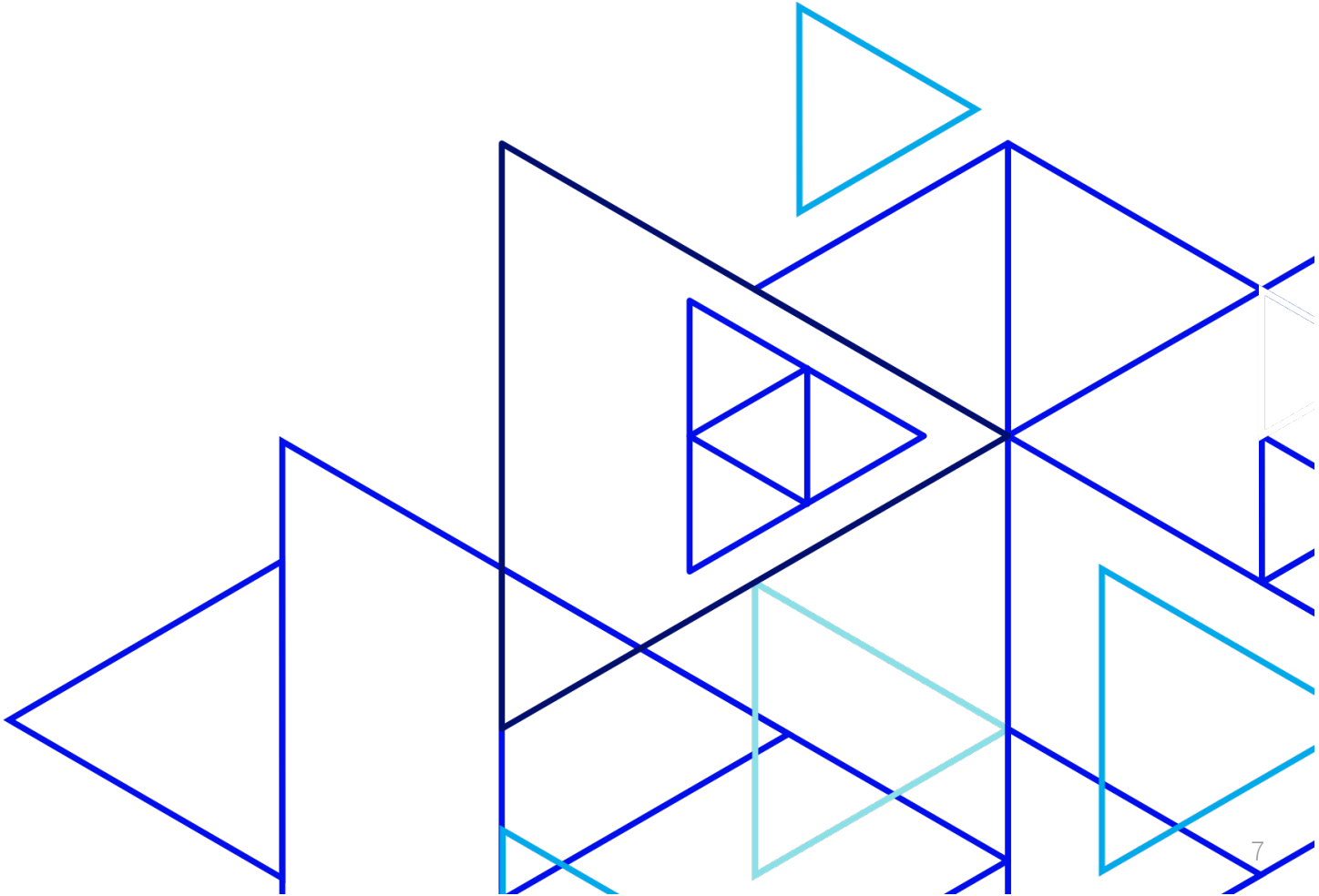
Schema migration



- Data definition language (DDL)

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

Context

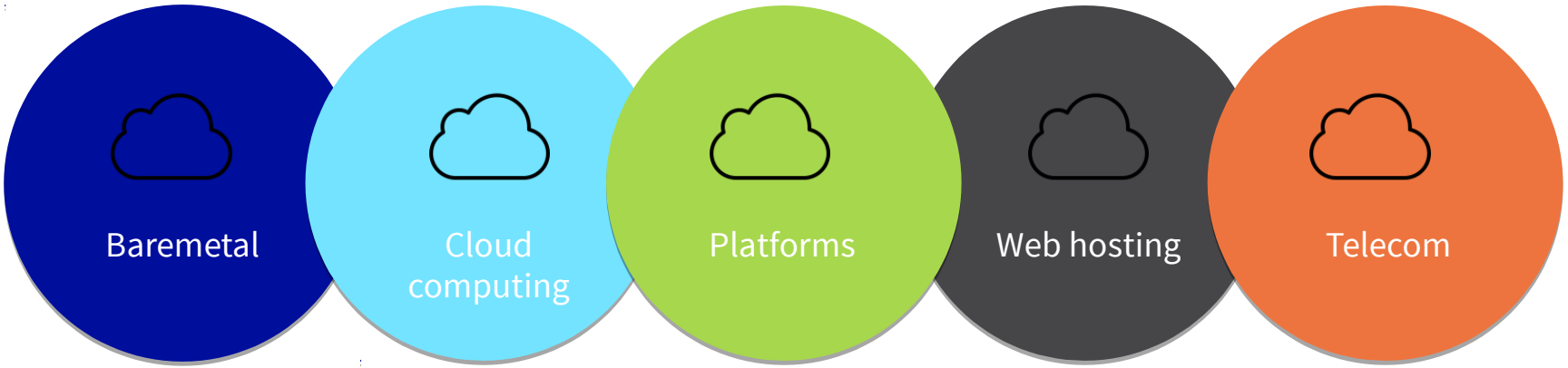


FOSDEM
February 6, 2021

OVHcloud


OVHcloud

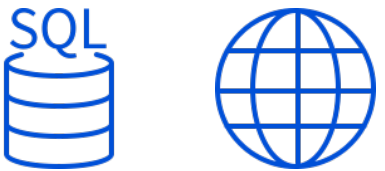
Critical services



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

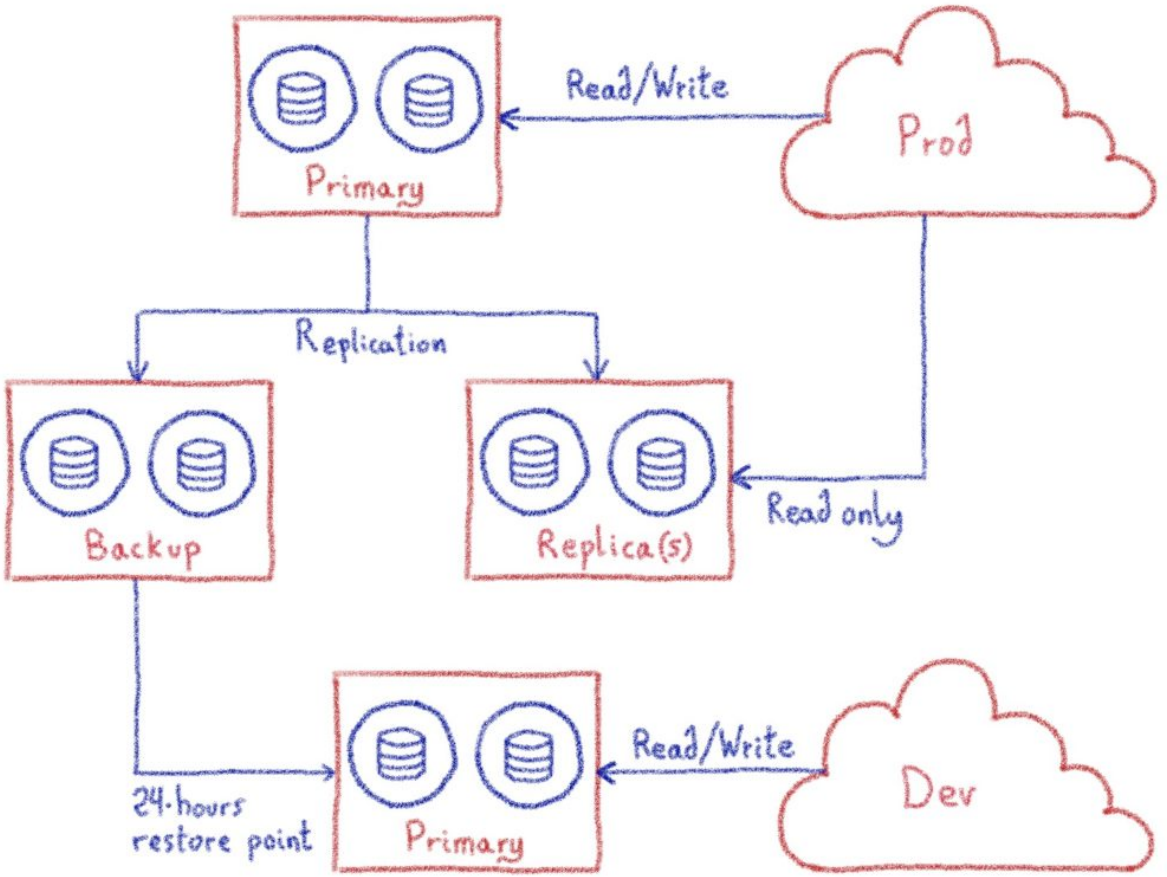
Internal databases

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

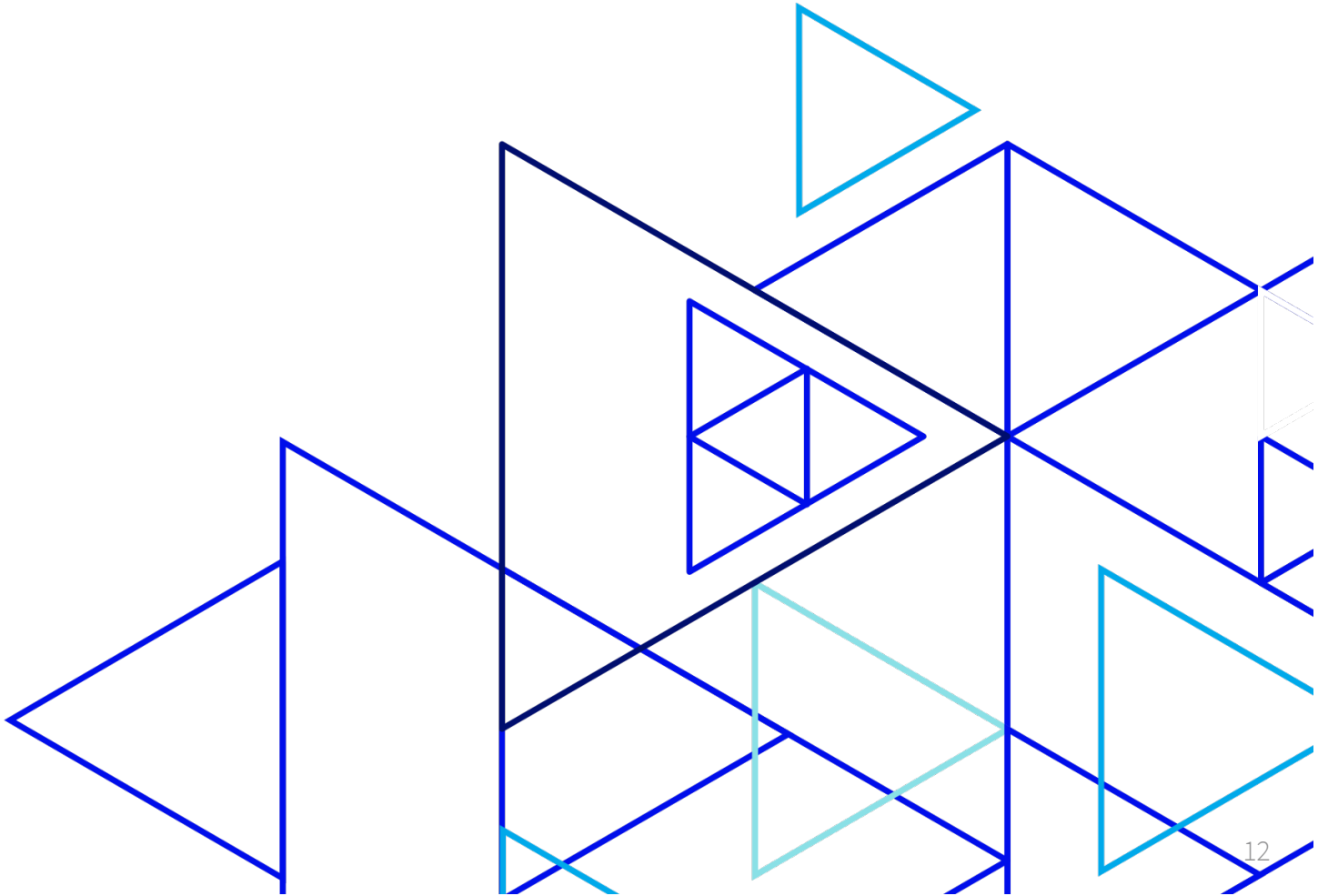


Internal cluster example

- MySQL
- PostgreSQL



Overview



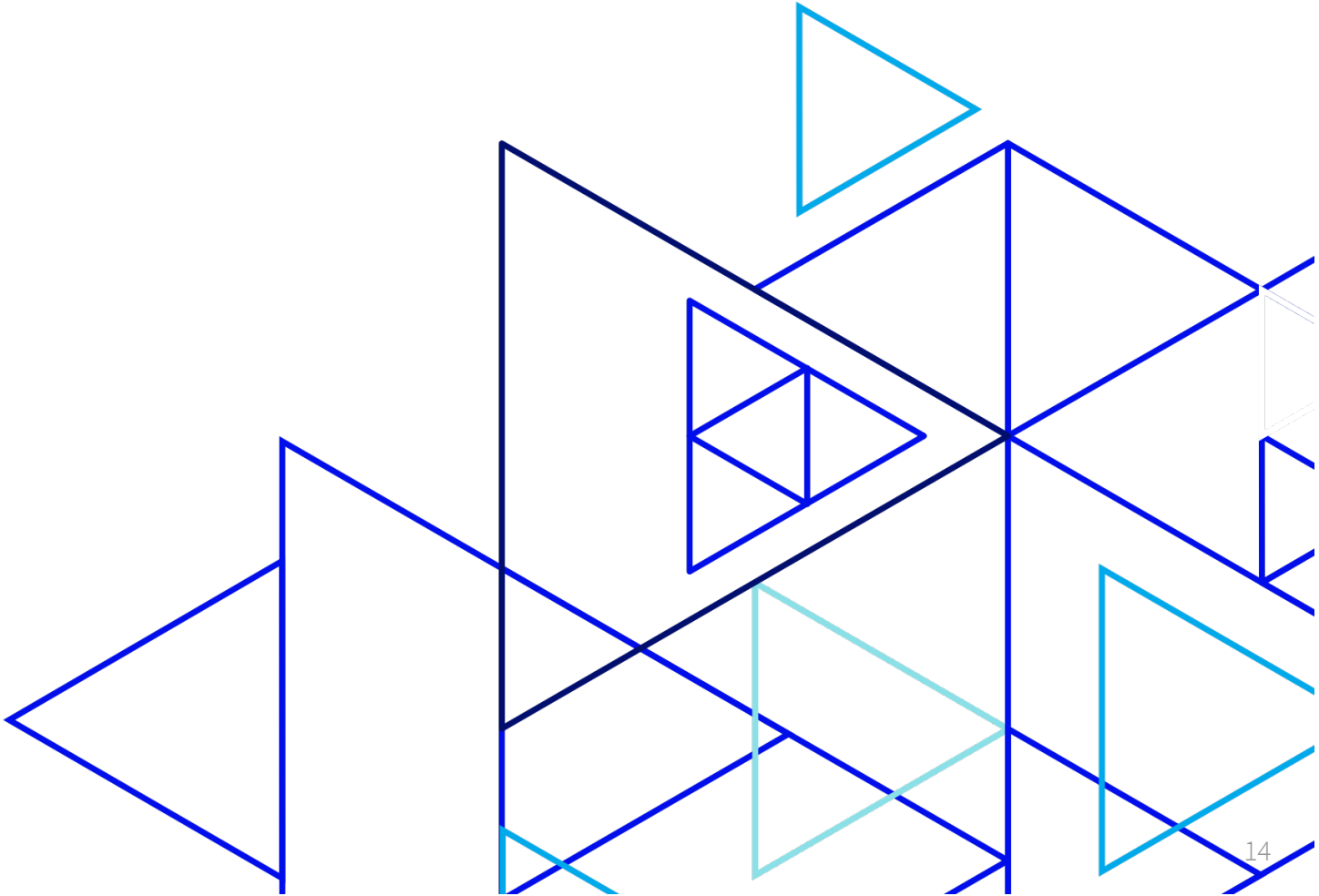
FOSDEM
February 6, 2021

OVHcloud

Overview

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

Chaos



FOSDEM
February 6, 2021

OVHcloud

Who can ALTER my database?

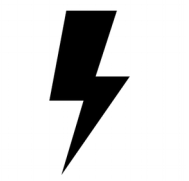
Startup mindset

- 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

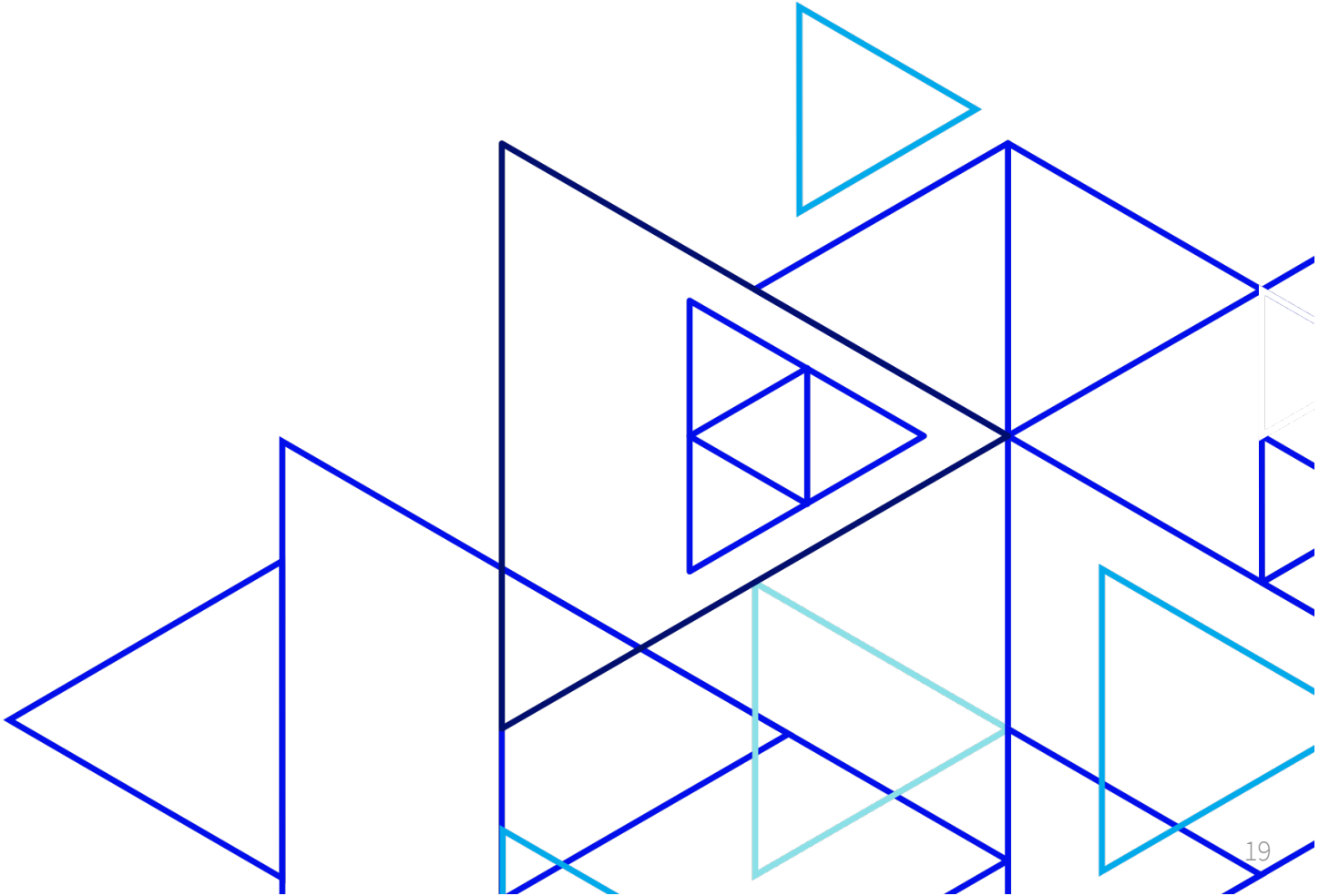
Startup mindset

- 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

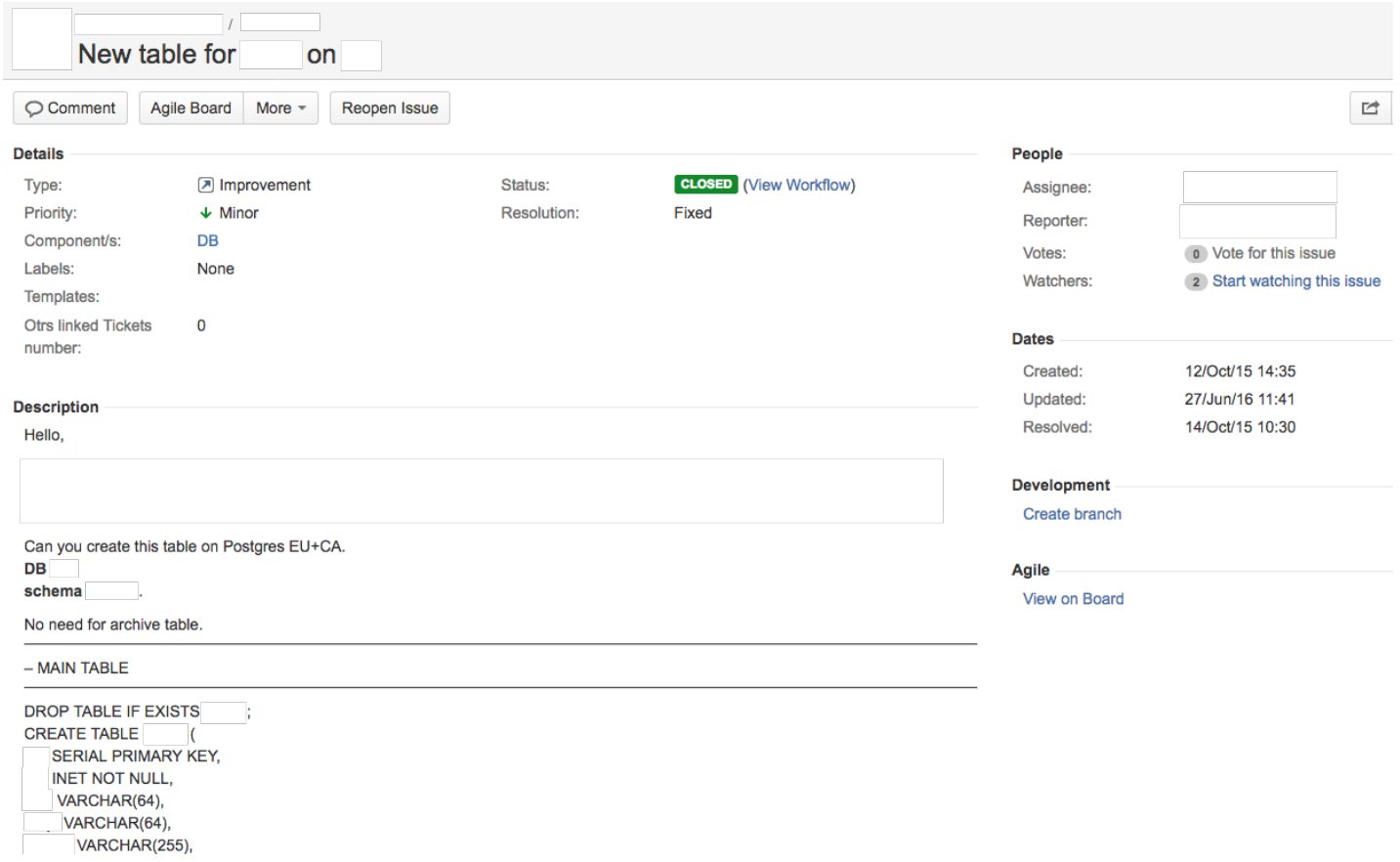


FOSDEM
February 6, 2021

OVHcloud

Let’s write migrations in a ticketing system

- JIRA



New table for on

Comment | Agile Board | More | Reopen Issue

Details

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

People

Assignee:

Reporter:

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

Development

Create branch

Agile

View on Board

Description

Hello,

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

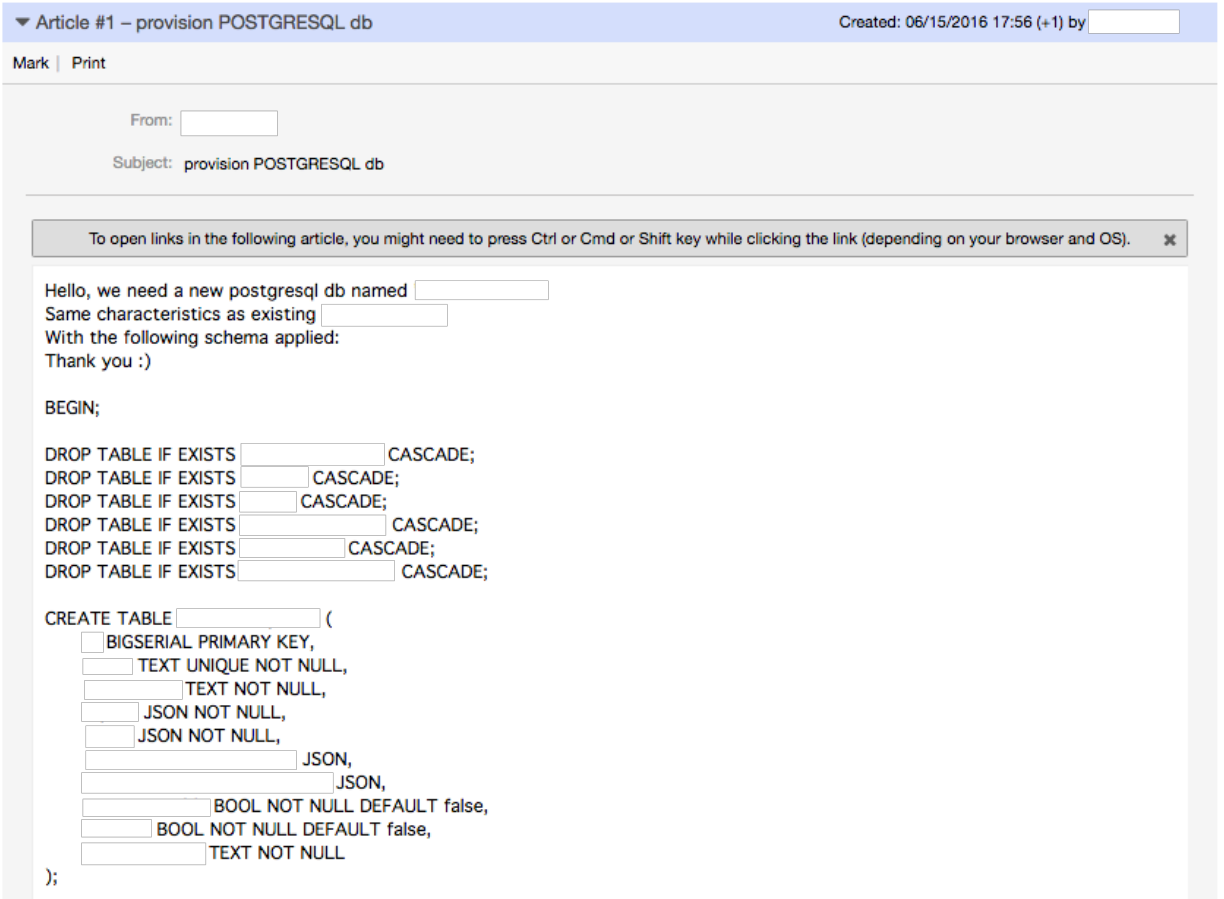
No need for archive table.

– MAIN TABLE

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

Let’s write migrations in a ticketing system

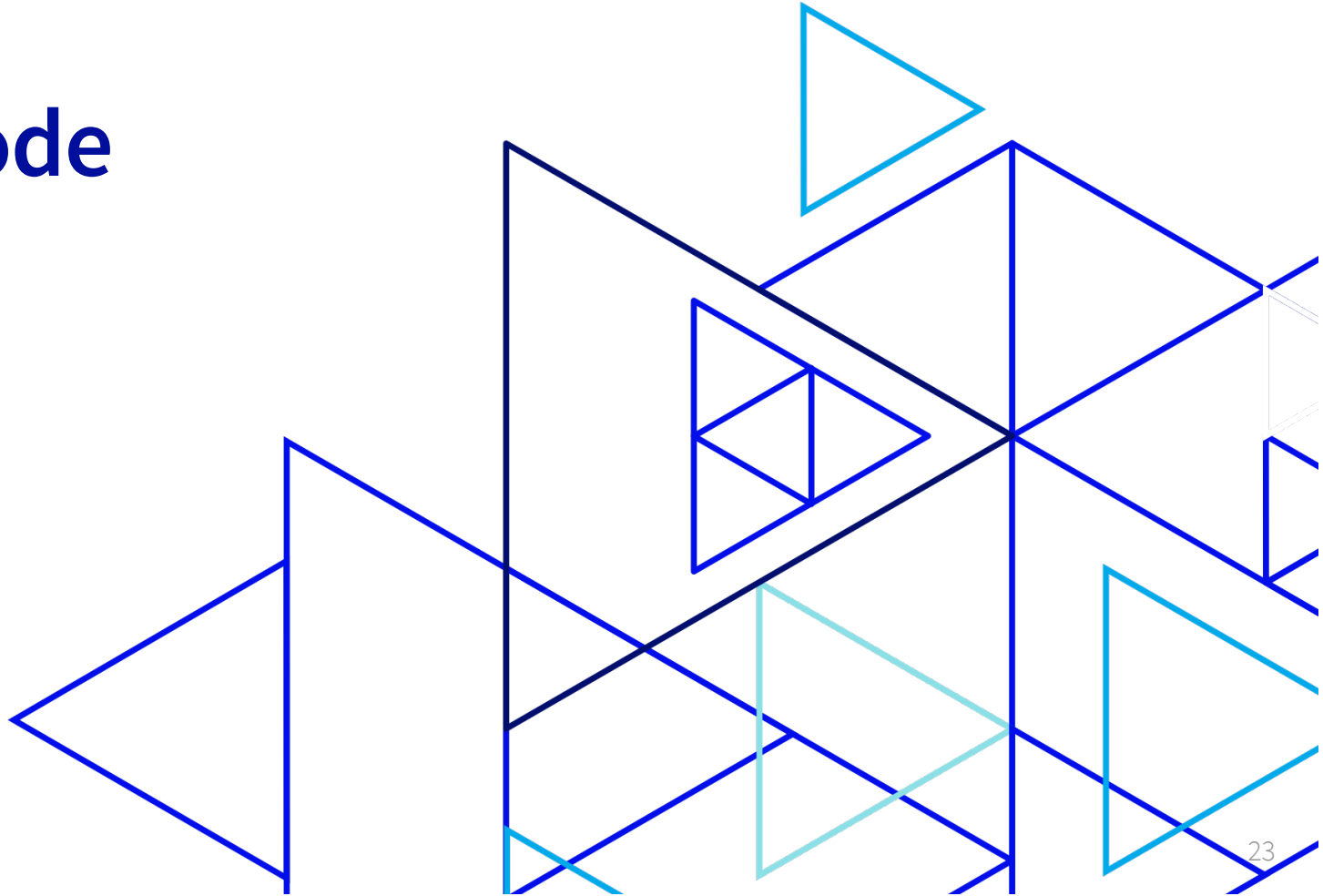
- OTRS



But

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

Schema as code



FOSDEM
February 6, 2021

OVHcloud

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  |
+-----------+---------------------------------------+
```

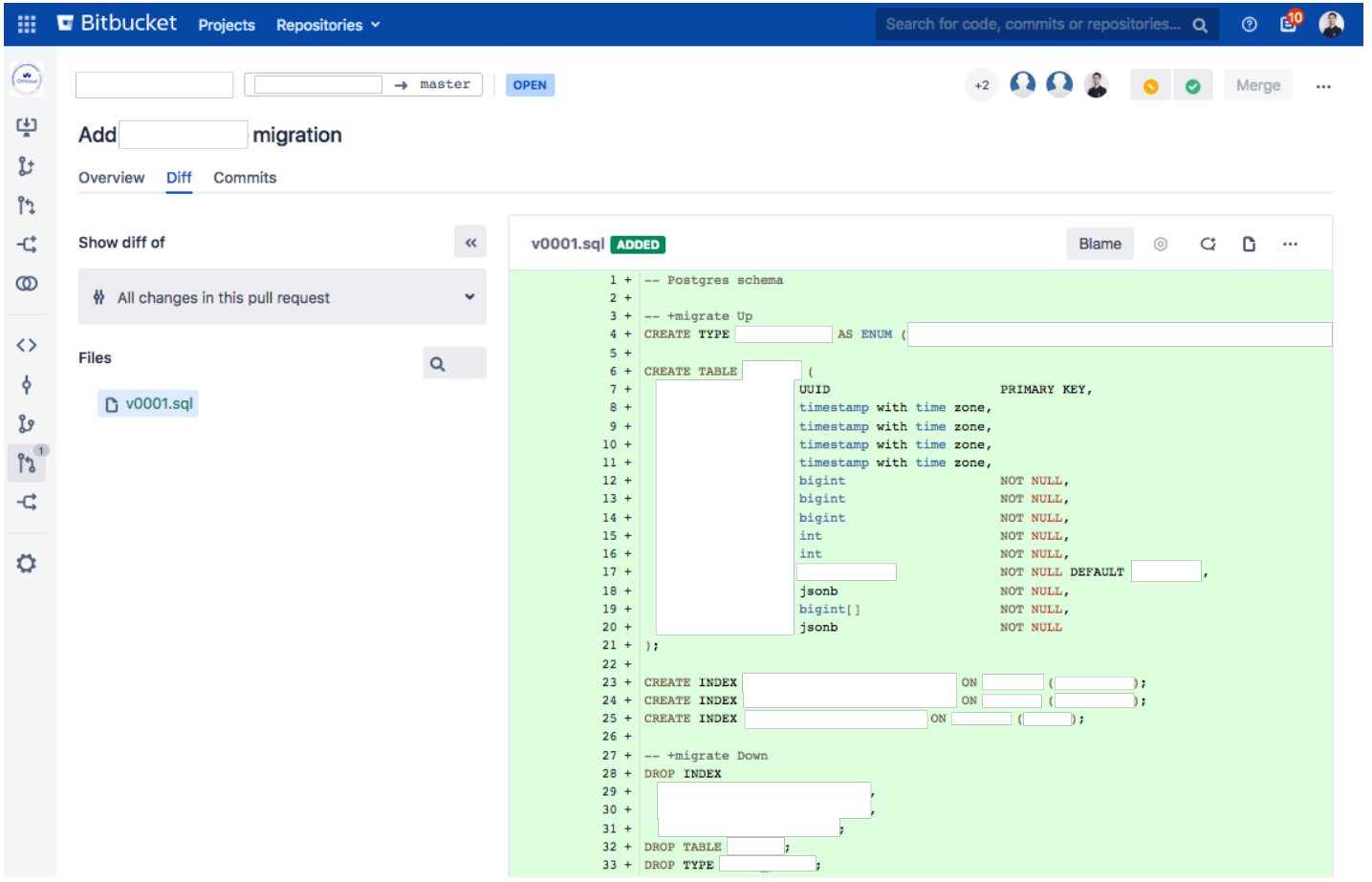
Schema migrations

- 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)

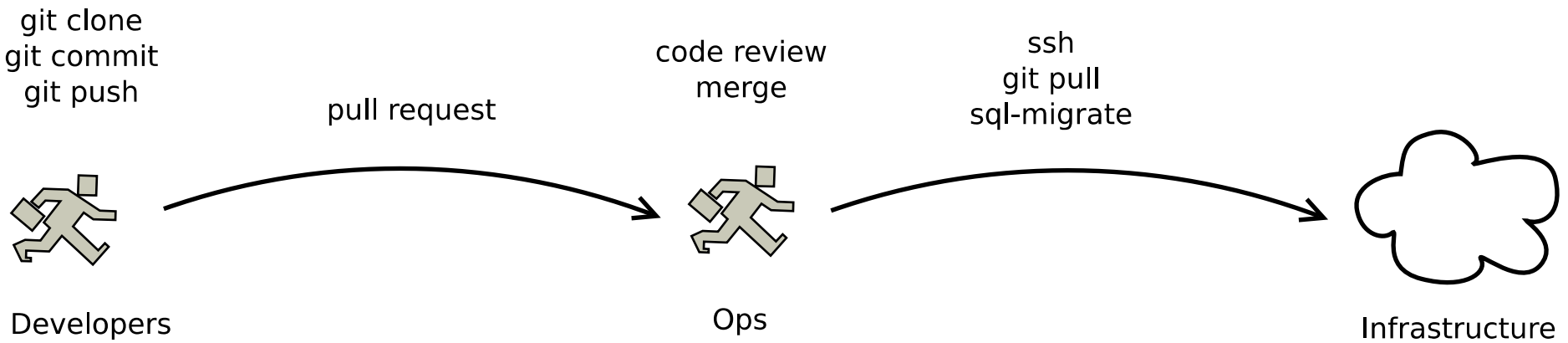
Code reviews

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

Code reviews with Bitbucket



Workflow


git clone
git commit
git push
pull request
code review
merge
ssh
git pull
sql-migrate
Developers
Ops
Infrastructure

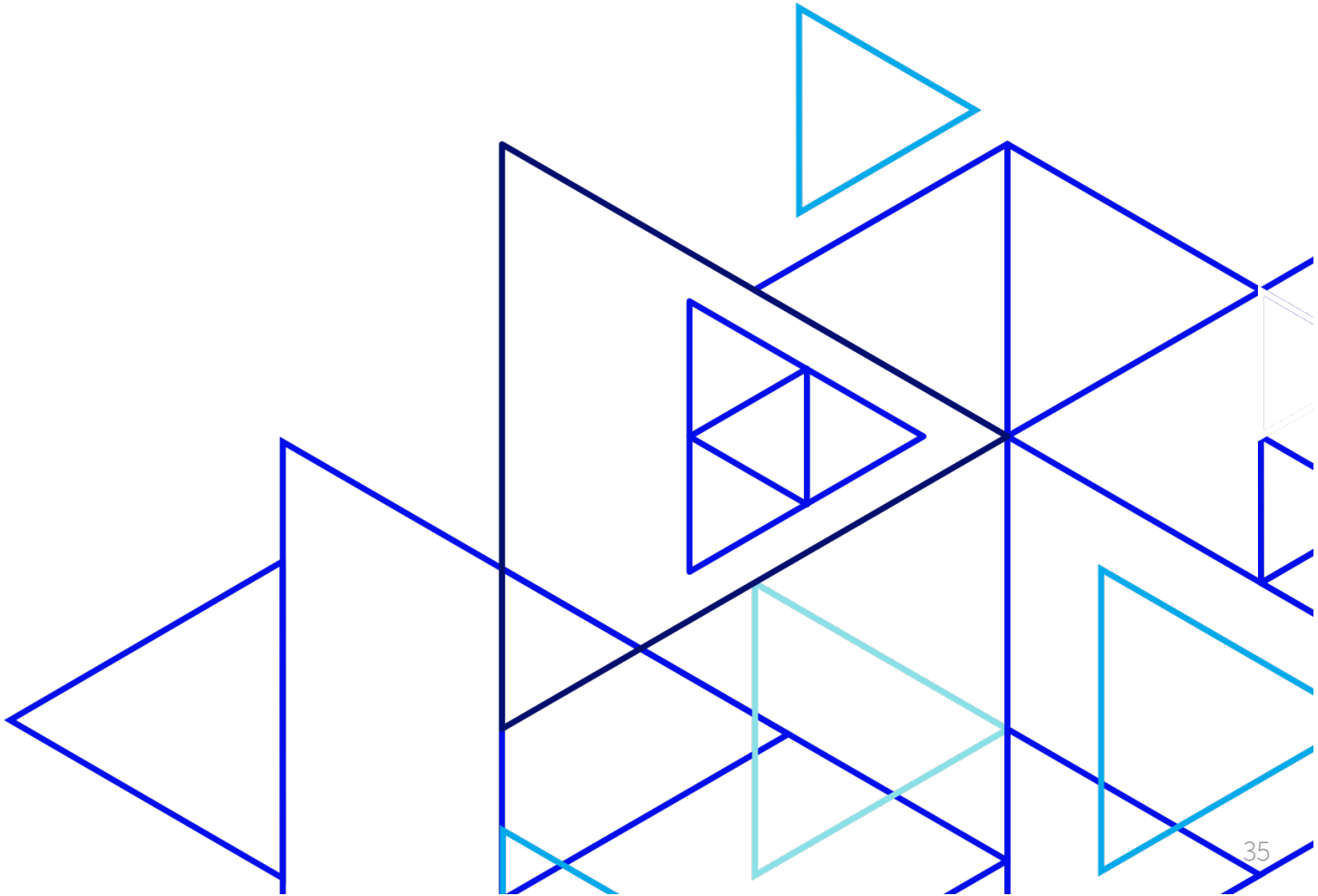
What's good

- 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

OVHcloud

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

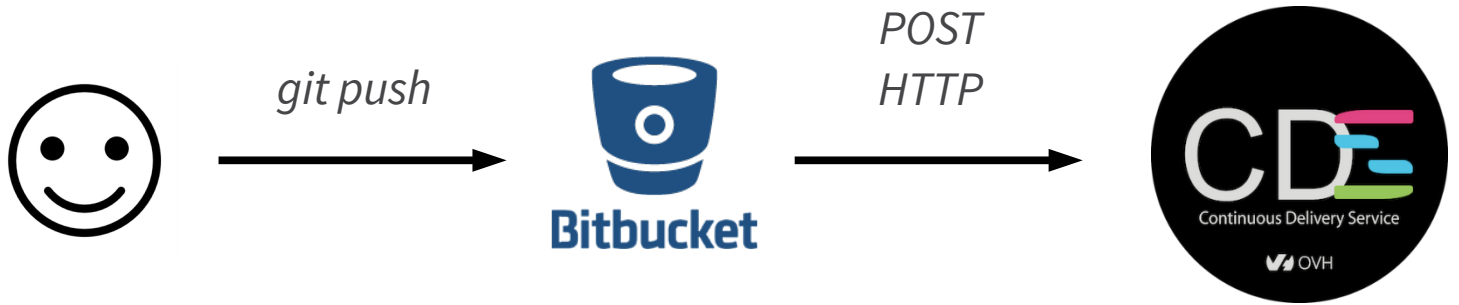
Continuous integration with CDS

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

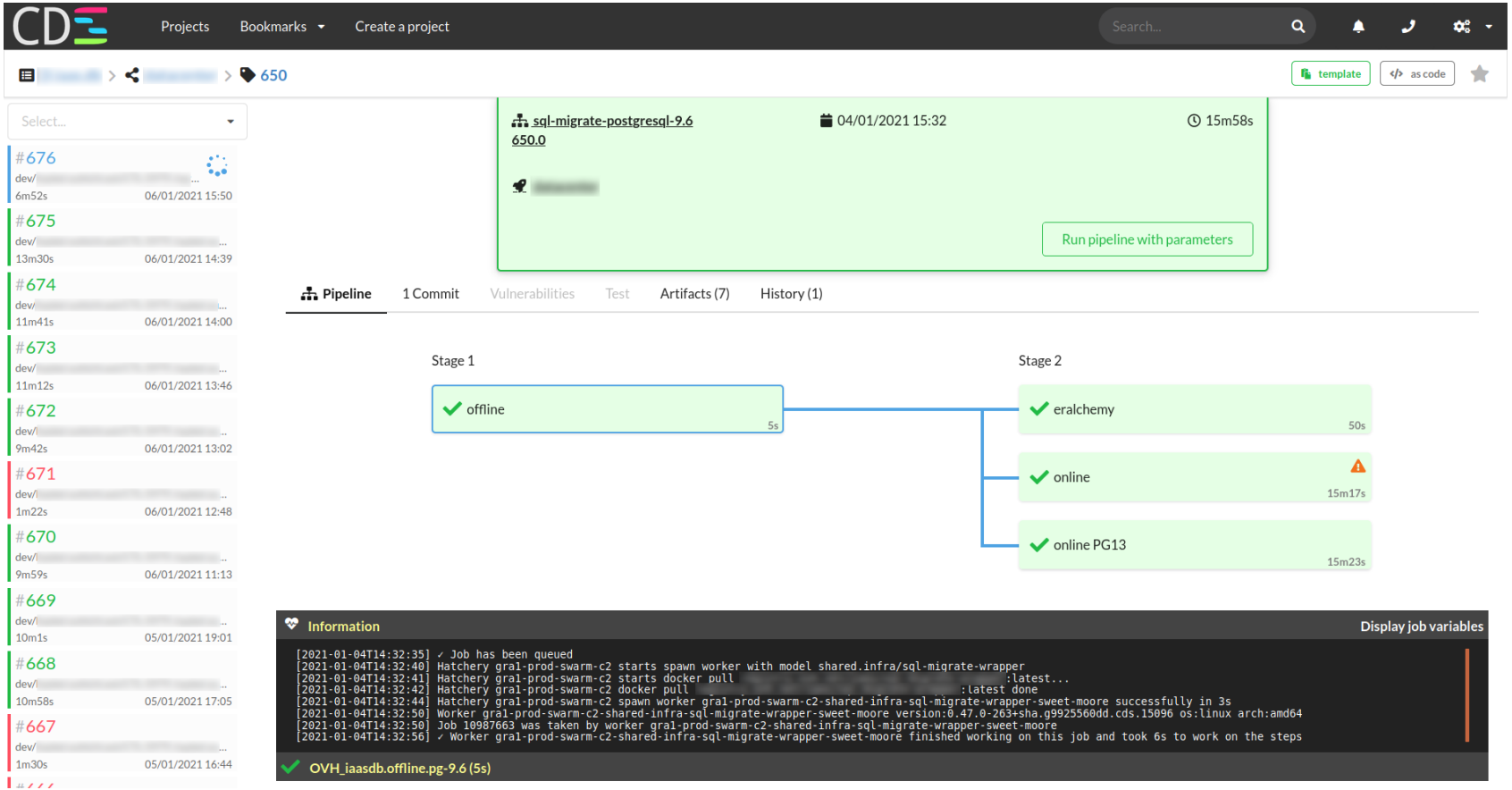
Continuous integration with CDS

- **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)

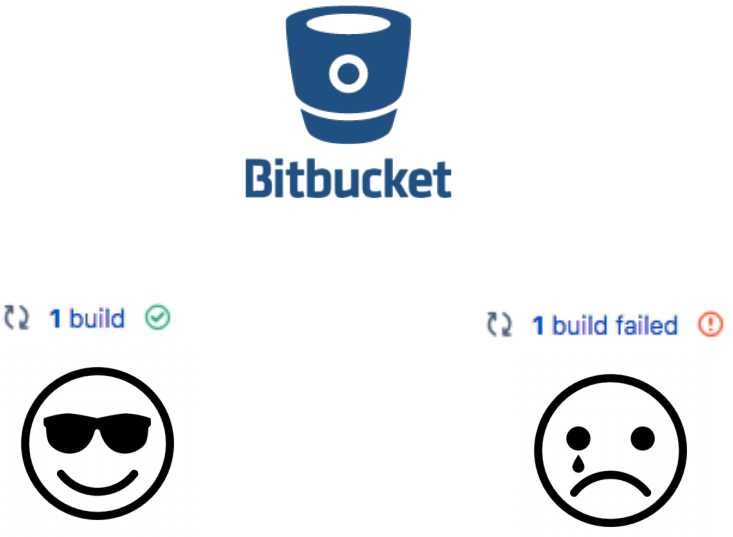
Continuous integration with CDS



Continuous integration with CDS



Continuous integration with CDS



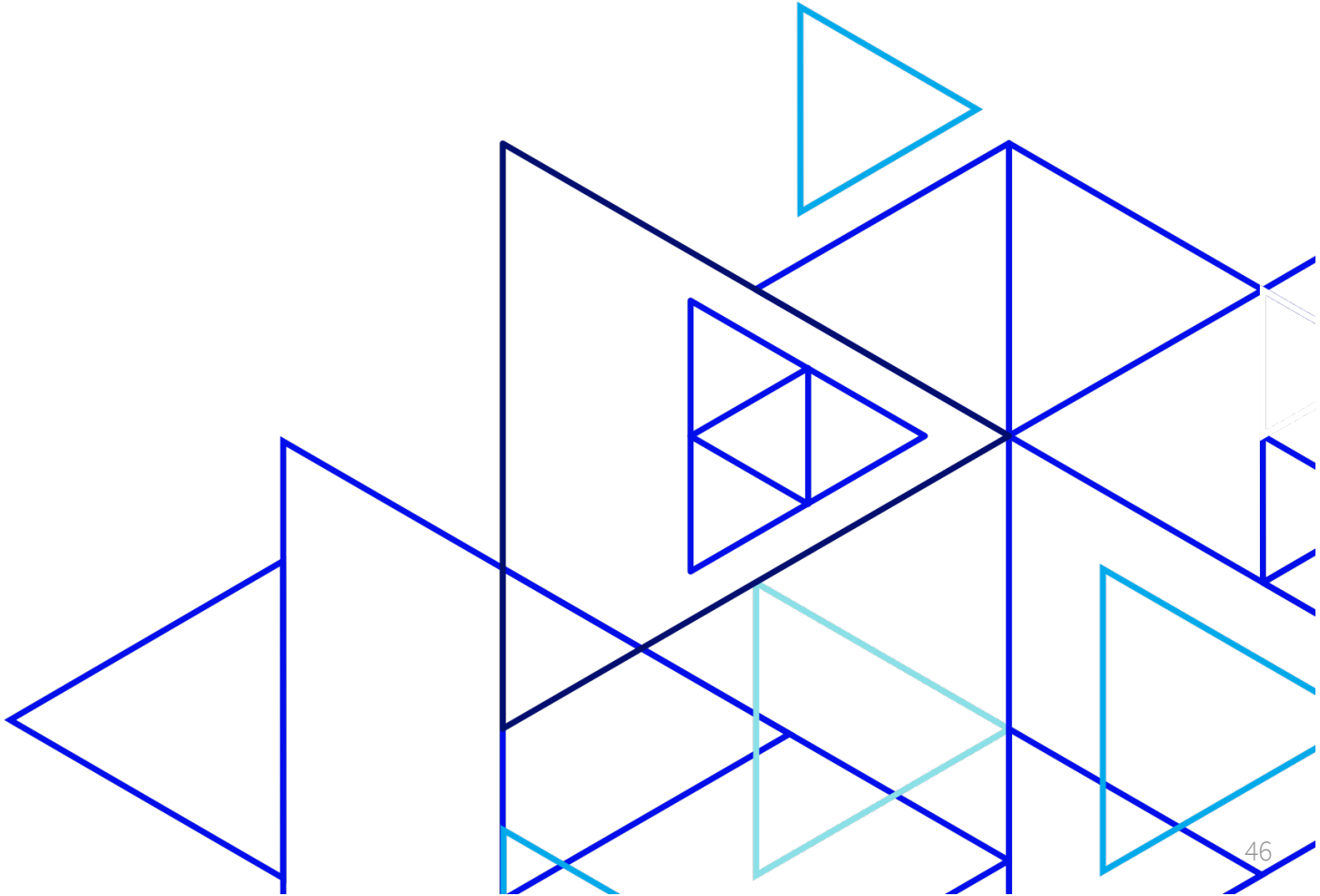
What’s good

- 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

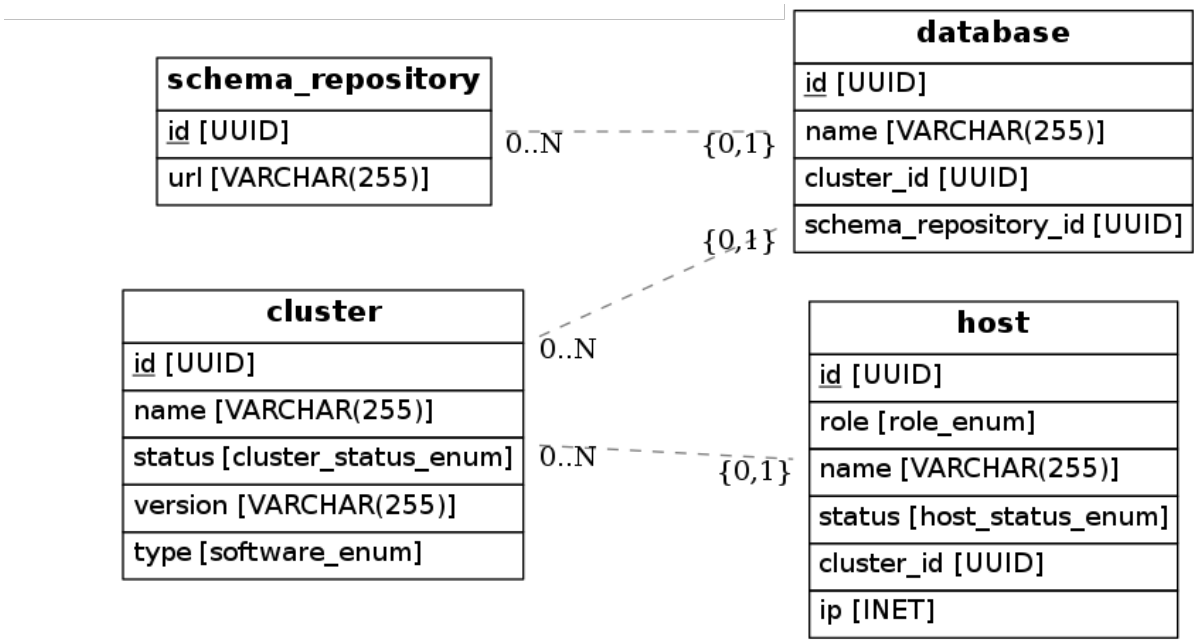
OVHcloud

Inventory

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

Inventory

- In a database of databases!



Inventory

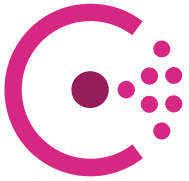
- With a REST API

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

Inventory, but…

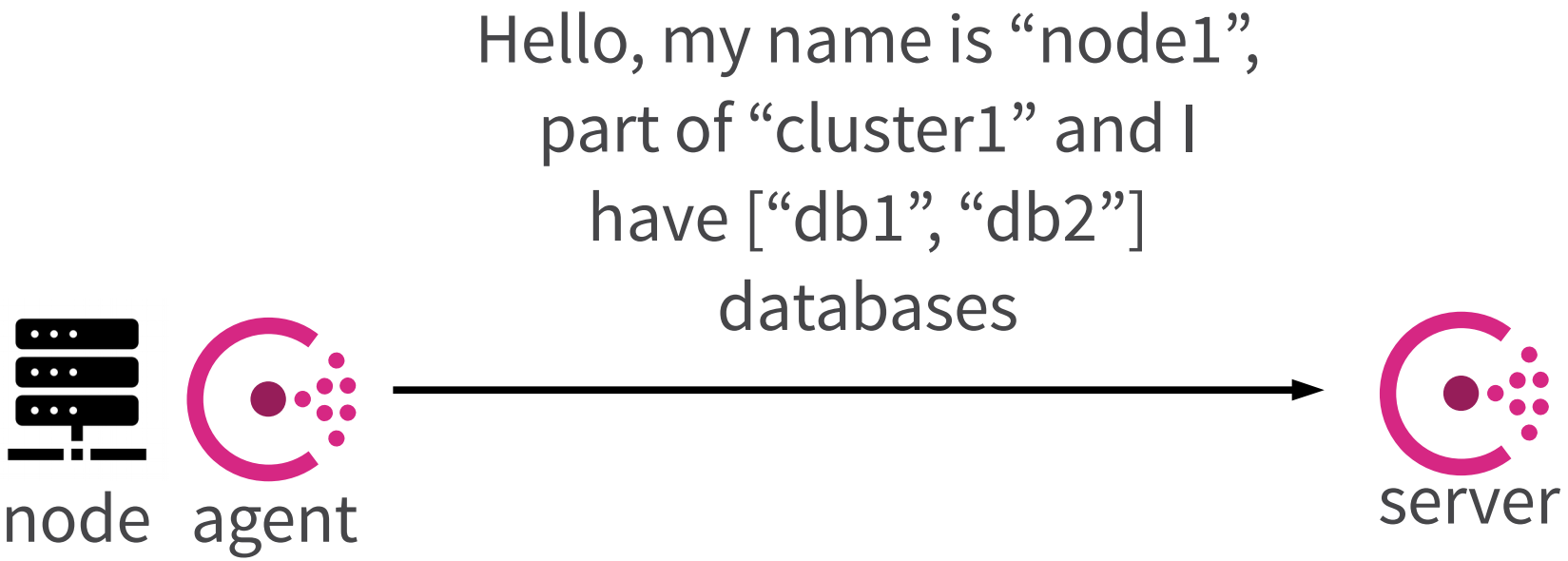
- Every change has to be declared
  - Add a cluster, host, database, git repository, ...
  - Remove a cluster, host, database, git repository, …
  - Easy to miss an event
- Fully home-made

Service discovery with Consul



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

Service discovery with Consul



Service discovery with Consul

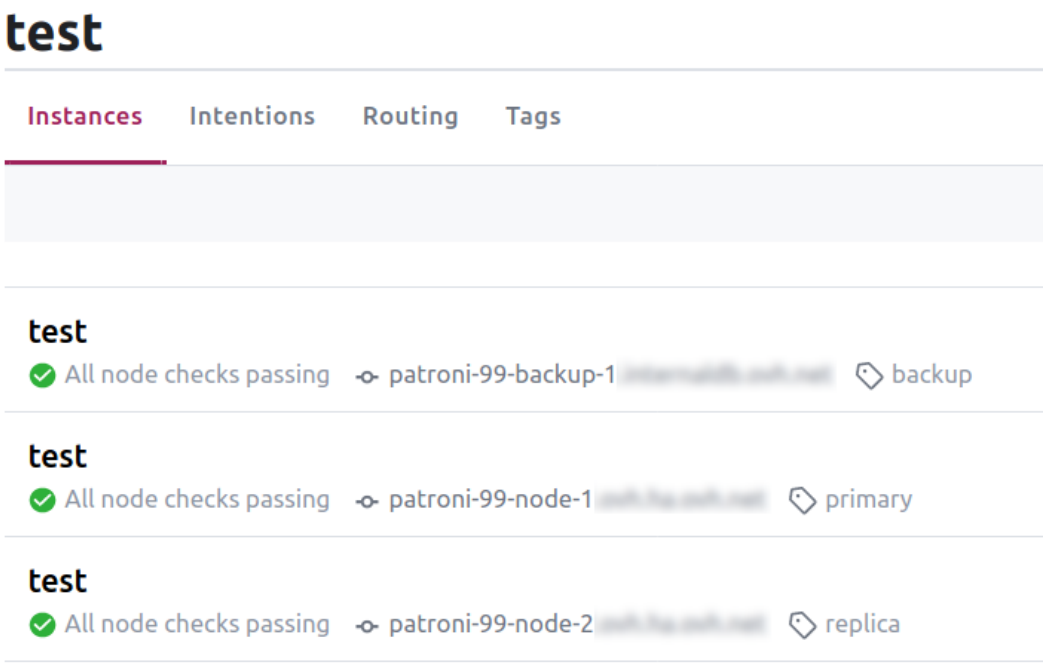
- **Nodes** → name, IP address, meta(data)
- **Services** → cluster, databases
- **Key/value store**
- Access control list (ACL)
- Encryption

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

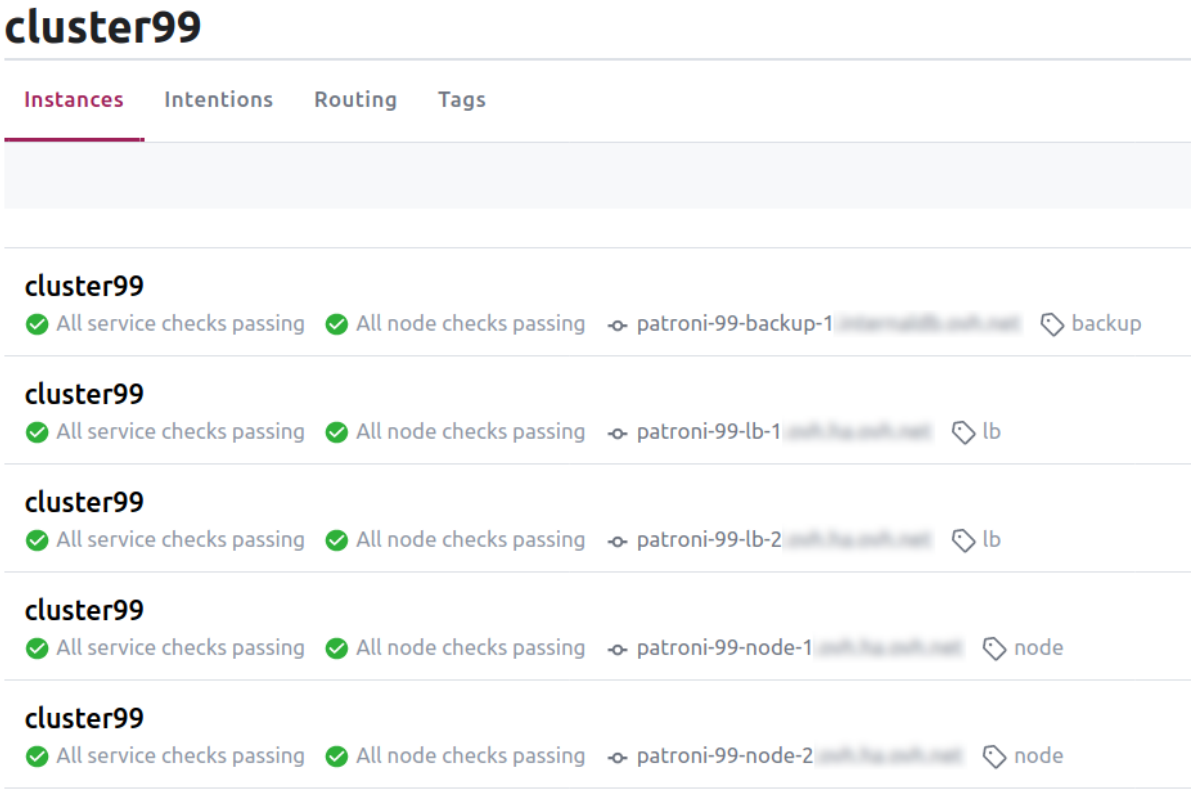
Consul – Nodes and services

- Where is my database?



Consul – Nodes and services

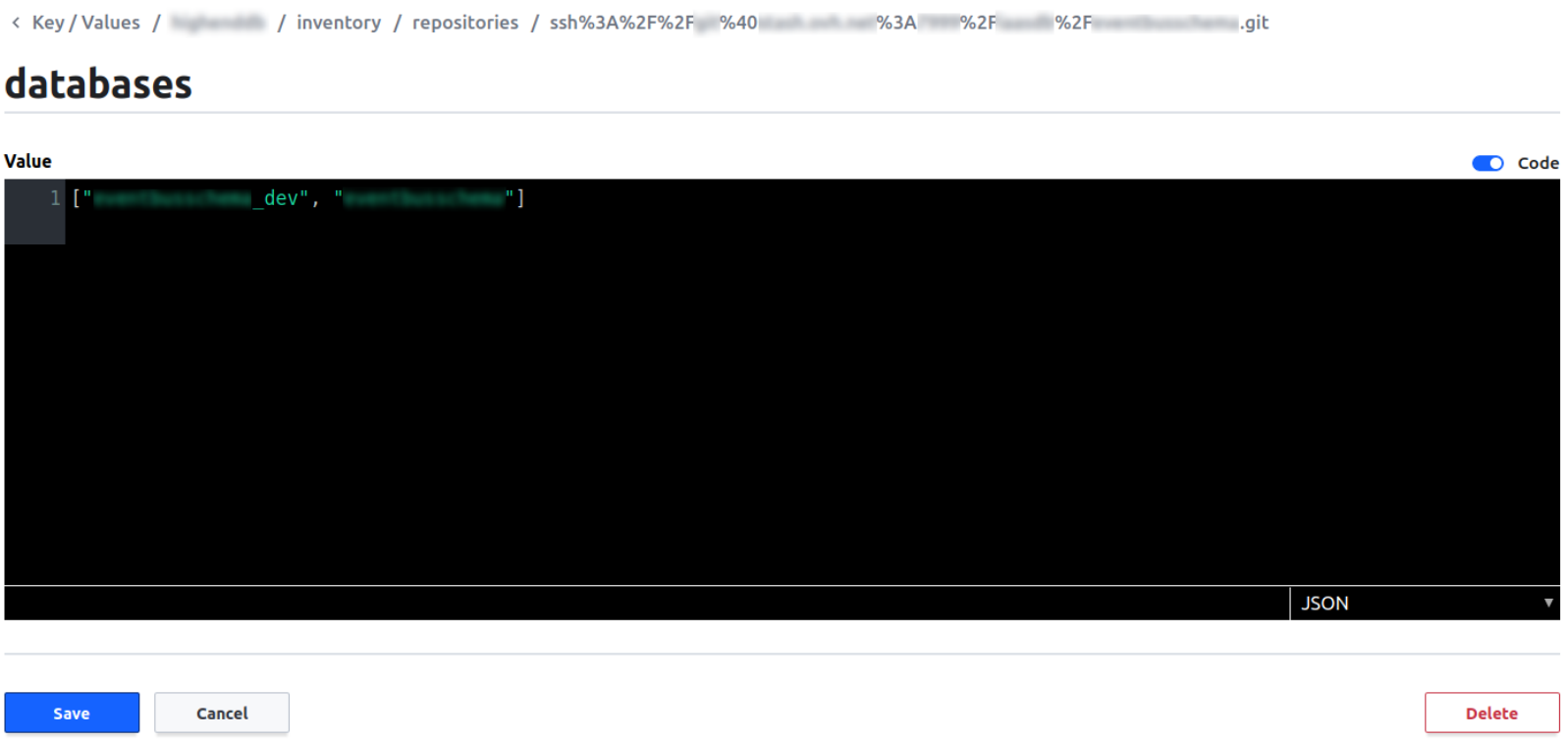
- Where is my cluster?



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



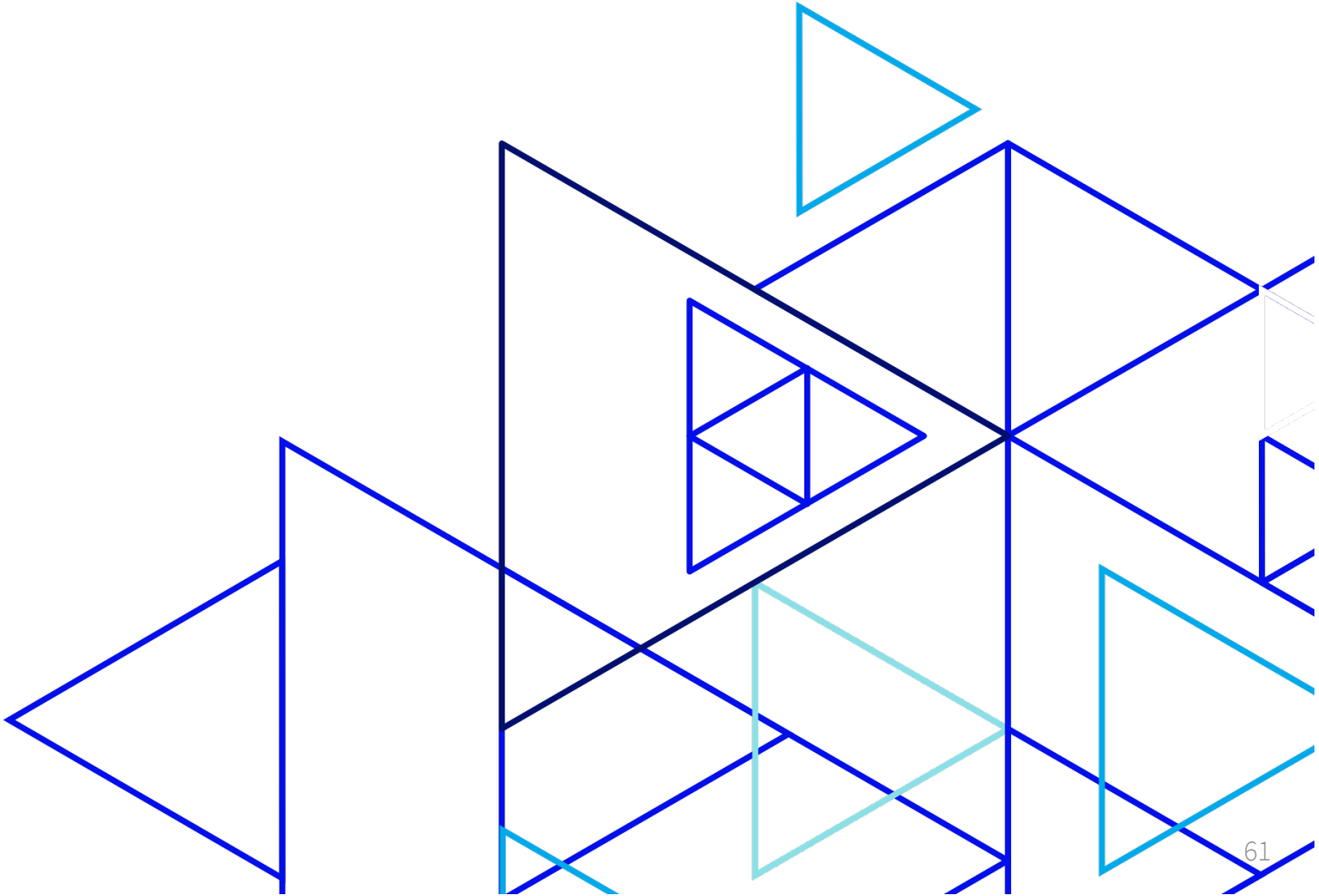
Consul – Automatic retention

- 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

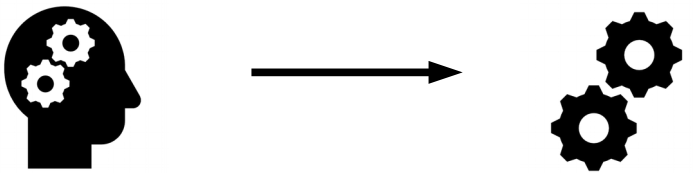


FOSDEM
February 6, 2021

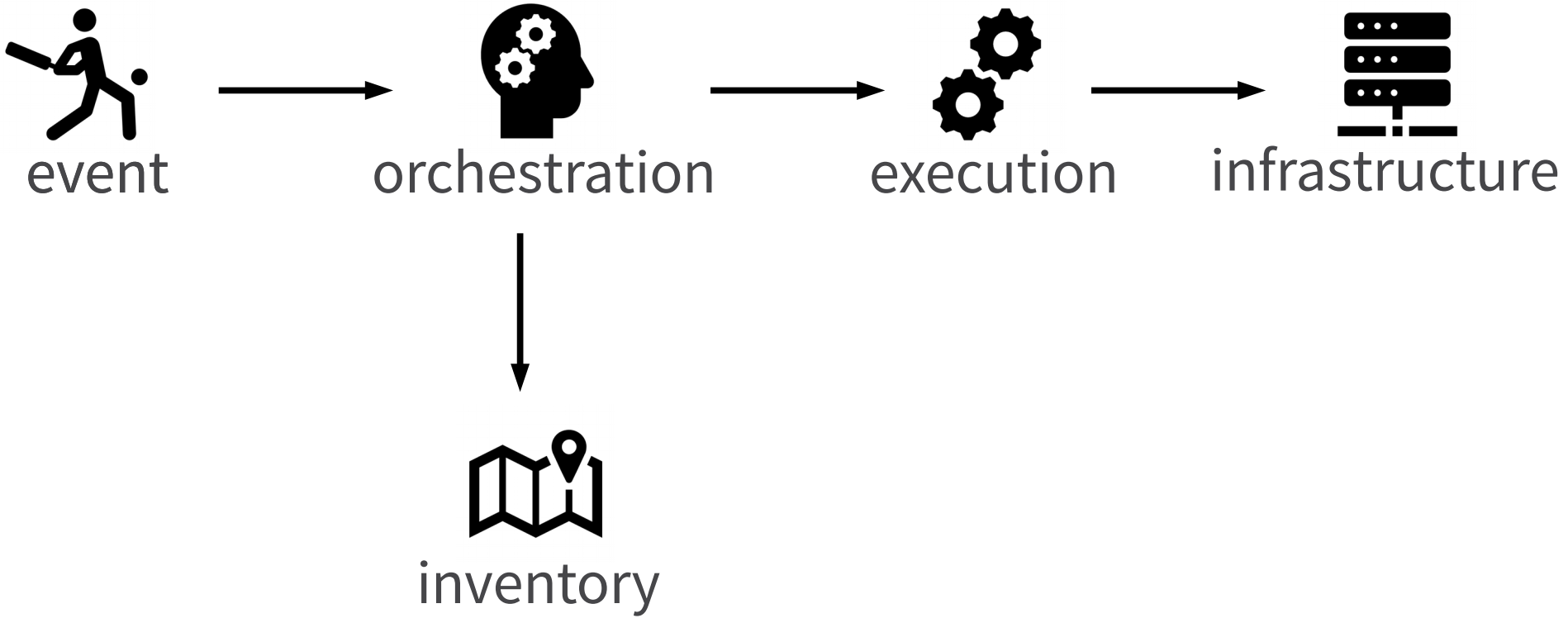
OVHcloud

Automation

- How migrations can be applied automatically?



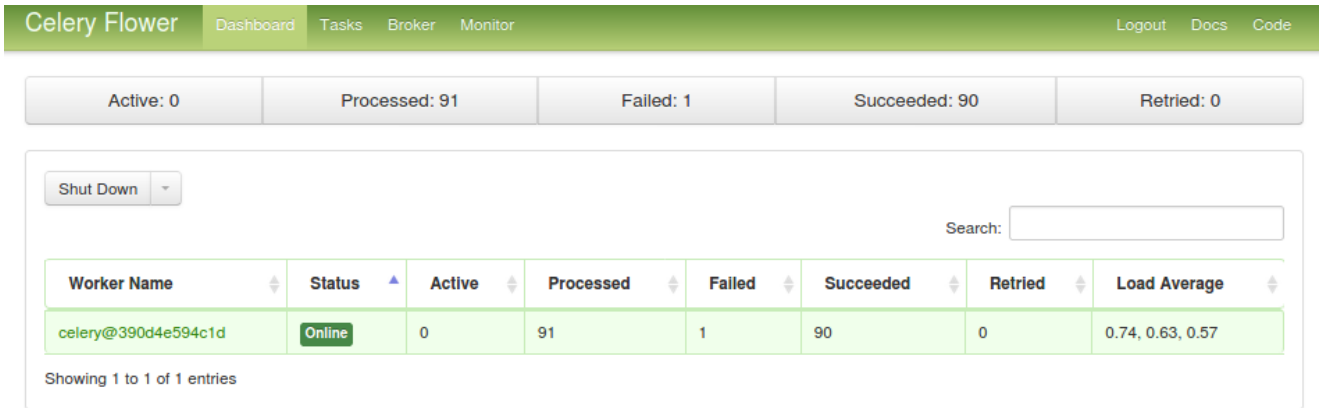
Automation


event
orchestration
execution
infrastructure
inventory

Orchestration

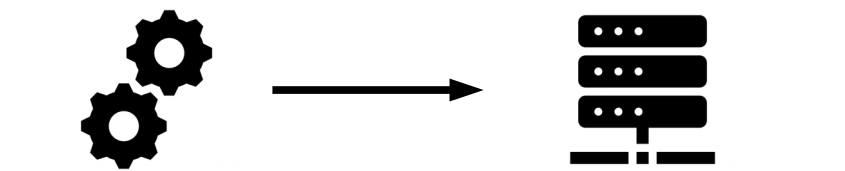
- Business logic
  - Look for databases → clusters → hosts → IP addresses
  - Talk to the “execution” to run schema migrations
- Custom development with standard libraries
  - HTTP API with **Flask**
  - Job scheduling with **Celery**
  - Web interface with **Celery Flower**

Orchestration



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 }}.yml
```

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

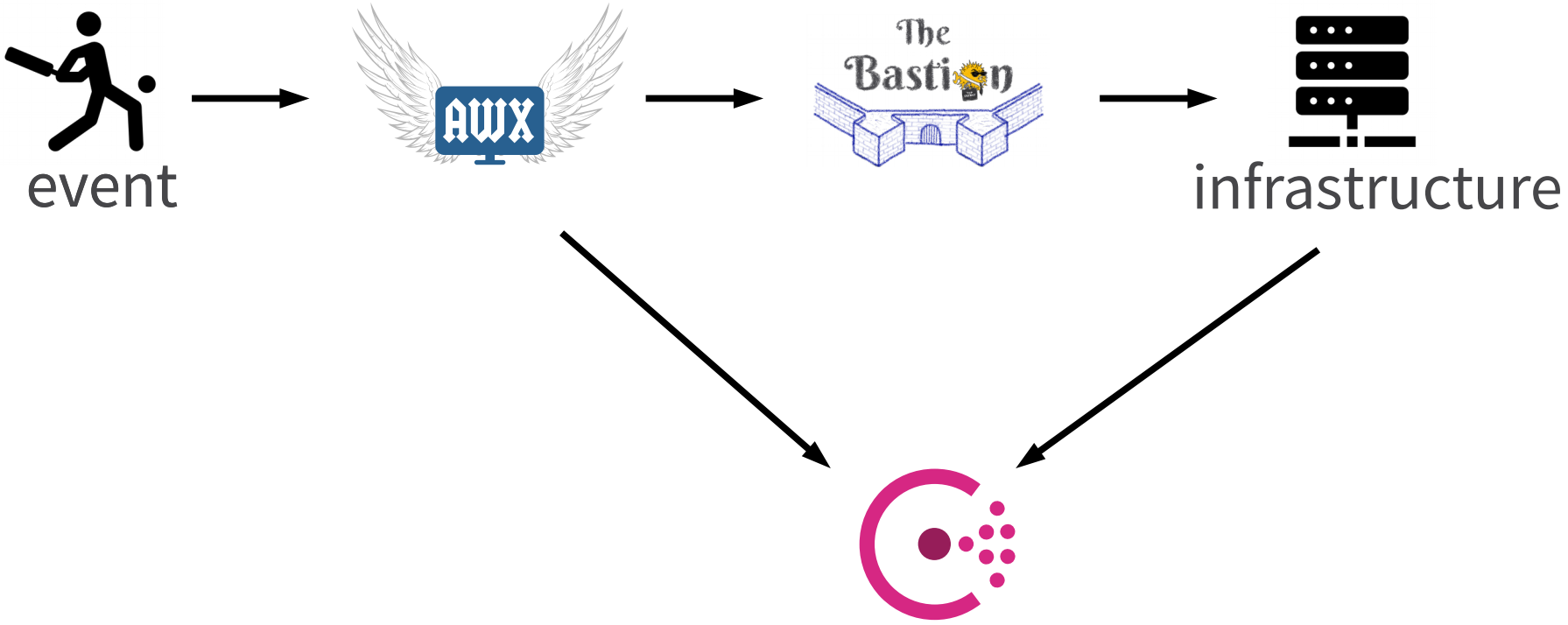
- Compatible with Ansible using SSH/SCP commands

```
[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
```

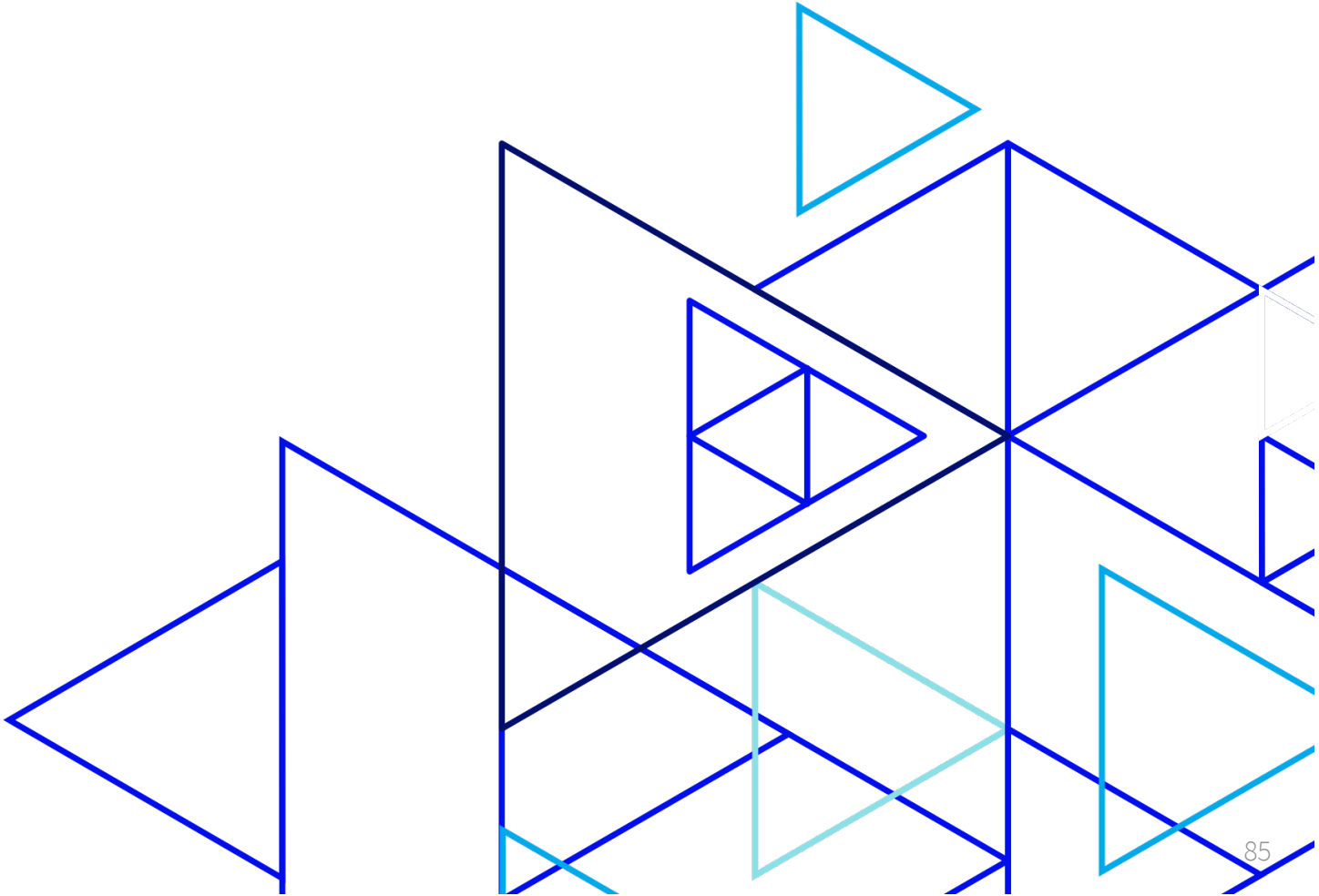
- Wrapper not opensource… yet!

Security

Architecture



Heaven



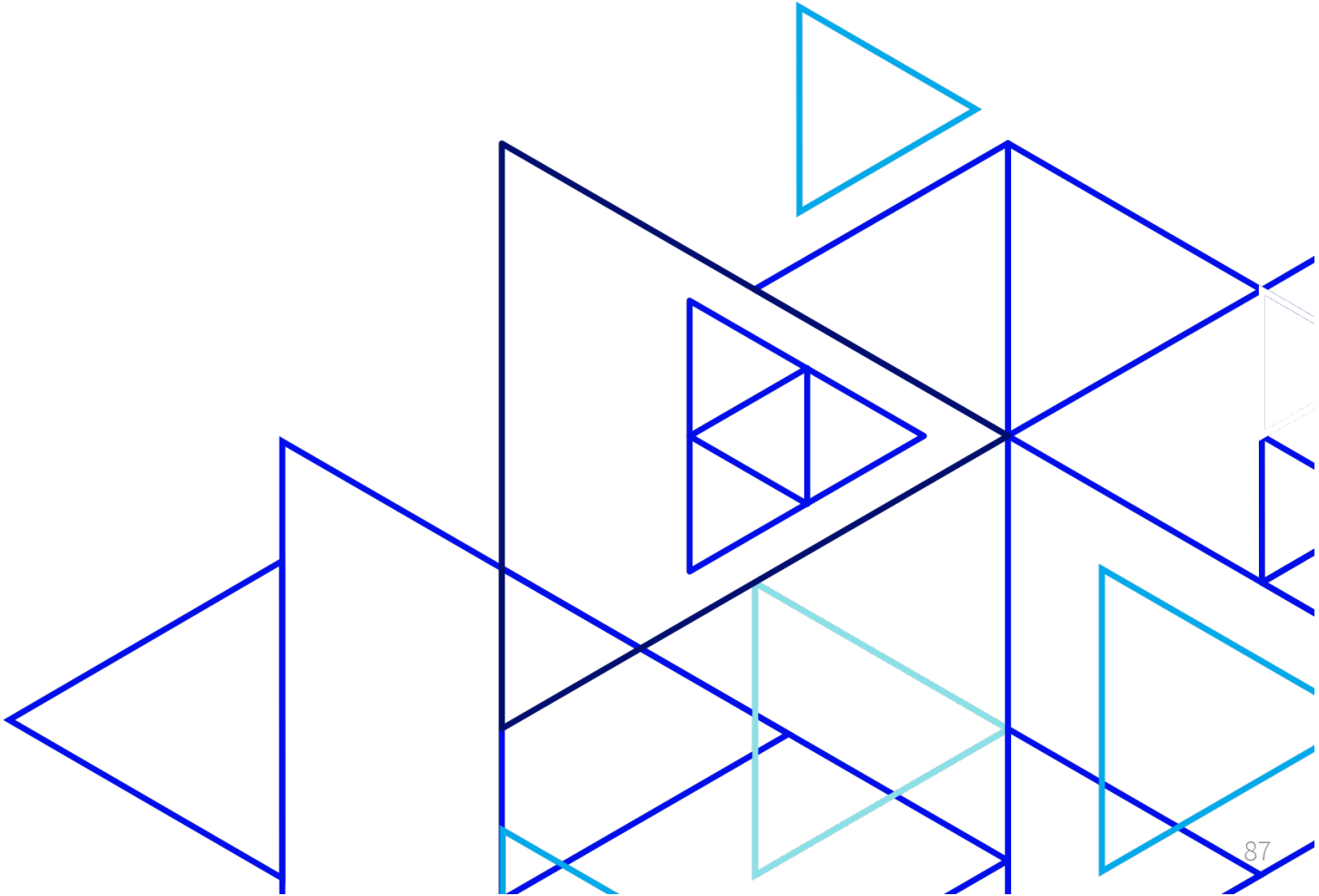
FOSDEM
February 6, 2021

OVHcloud

Heaven

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

What’s next?



FOSDEM
February 6, 2021

OVHcloud

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