Пора тестировать миграции

А нужно ли тестировать миграции?

Надежный пайплайн

Надежный пайплайн

CR-1

Надежный пайплайн

CR-1

CR-2

Надежный пайплайн

CR-1

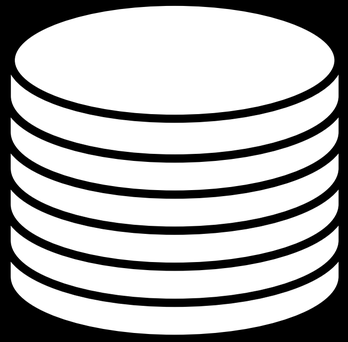
CR-2

Deployment

Надежный пайплайн

CR-1

CR-2



Deployment

Надежный пайплайн

CR-1

CR-2

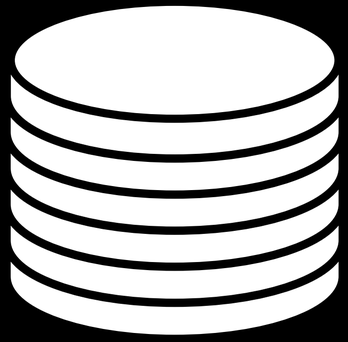
Deployment


Дамы и господа
базы данных
больше нет

В чем проблема?

CR-1

CR-2



* Deployment *

В чем проблема?

1. Больно ревьюить
2. Деградация пайплайна
3. Postmortem

Надежный пайплайн

Code of Conduct

```
1  -- migrate:up  << Это upgrade-блок
2  SELECT * FROM *;
3
4  -- migrate:down << Это downgrade-блок
5  SELECT * FROM *;
6
```

Пора тестировать миграции

Давайте автоматизируем up-down

Column	Type	Constraints	Default
id	SERIAL	PRIMARY KEY	
username	TEXT	NOT NULL	
current_mood	mood	NOT NULL	'ok'
updated_at	TEXT	NOT NULL	

```
| Column        | Type    | Constraints                | Default  |
|---------------|---------|----------------------------|----------|
| id            | SERIAL  | PRIMARY KEY                |          |
| username      | TEXT    | NOT NULL                   |          |
| current_mood  | mood    | NOT NULL                   | 'ok'     |
| updated_at    | TEXT    | NOT NULL                   |          |

```

```
-- migrate:up
ALTER TYPE mood ADD VALUE 'happy' AFTER 'ok';

-- migrate:down

```

Column	Type	Constraints	Default
id	SERIAL	PRIMARY KEY	
username	TEXT	NOT NULL	
current_mood	mood	NOT NULL	'ok'
updated_at	TEXT	NOT NULL	

- up
- down

```
-- migrate:up
ALTER TYPE mood ADD VALUE 'happy' AFTER 'ok';

-- migrate:down

```

```
| Column        | Type    | Constraints                | Default  |
|---------------|---------|----------------------------|----------|
| id            | SERIAL  | PRIMARY KEY                |          |
| username      | TEXT    | NOT NULL                   |          |
| current_mood  | mood    | NOT NULL                   | 'ok'     |
| updated_at    | TEXT    | NOT NULL                   |          |
```

✅ up
✅ down
❌ up

```
-- migrate:up
ALTER TYPE mood ADD VALUE 'happy' AFTER 'ok';

-- migrate:down

```

Выводы

1. up-down не проверяет идемпотентность

Выводы

1. up-down не проверяет идемпотентность
2. down важнее up, так как отвечает за приведение к инварианту

Staircase-тест

1. накатываем все миграции

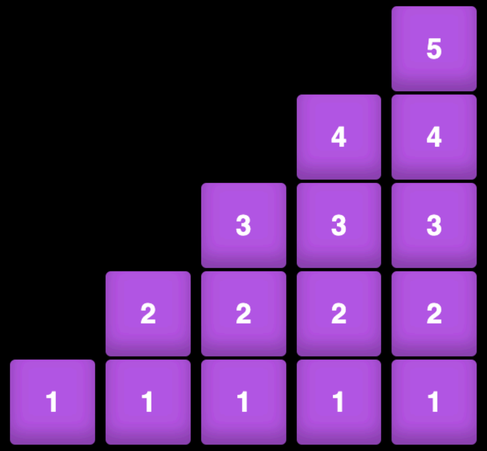
Staircase-тест

1. накатываем все миграции
2. итеративный down > up > down
до depth: int = 0

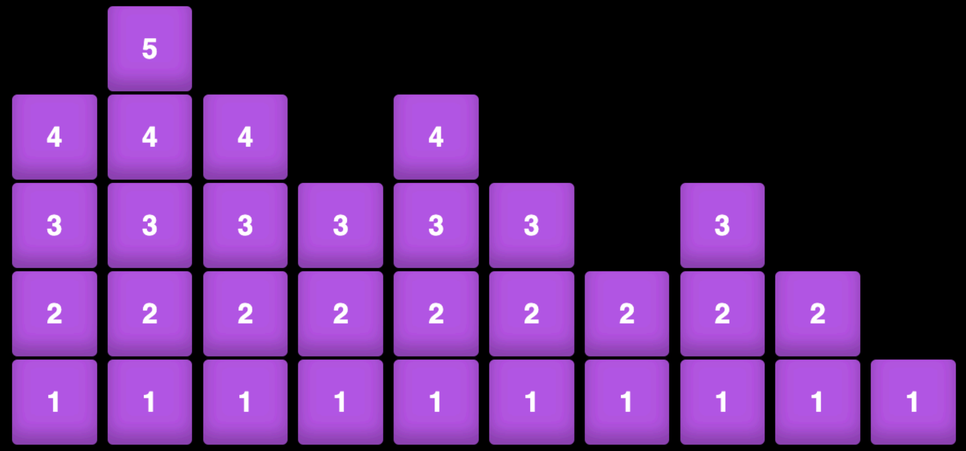
Staircase-тест

1. накатываем все миграции
2. итеративный down > up > down
 до depth: int = 0
3. накатываем откаченное

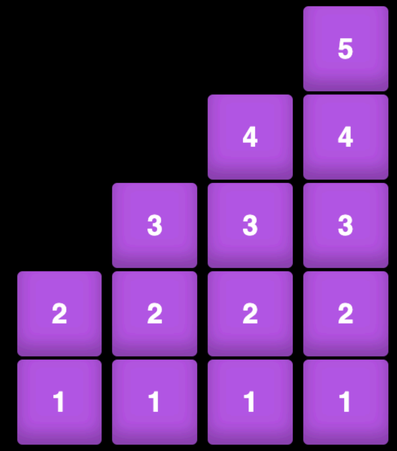
Staircase-test



1. Actualize



2. down > up > down



3. Re-actualize

Есть проблема

```sql
-- migrate:up
DO $$
BEGIN
    IF NOT EXISTS (
        SELECT 1
        FROM pg_enum
        WHERE enumlabel = 'happy'
          AND enumtypid = 'mood'::regtype
    ) THEN
        ALTER TYPE mood ADD VALUE 'happy' AFTER 'ok';
    END IF;
END;
$$;

-- migrate:down

```

Миграция **идемпотентна**, но **инвариант** ломает

Снапшоты схем

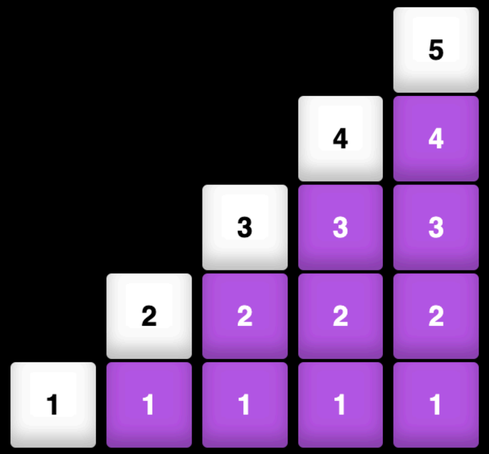
- snap1
- up
- down
- snap2

Снапшоты схем

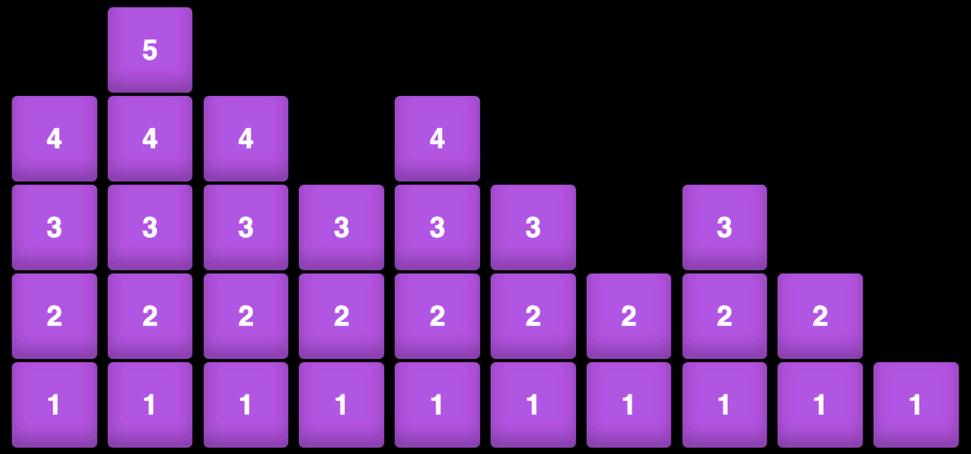
- snap1
- up
- down
- snap2

→ snap1==snap2

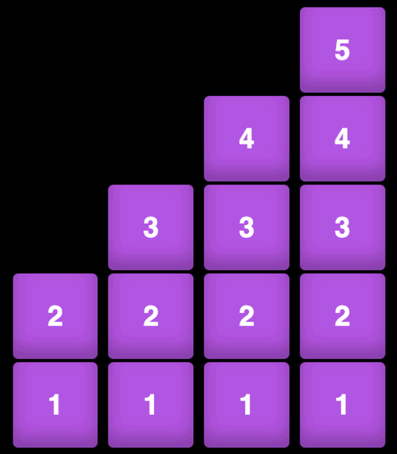
Staircase + Снапшоты



1. Actualize



2. down > up > down



3. Re-actualize

белые — эталоны

Column	Type	Constraints	Default
id	SERIAL	PRIMARY KEY	
username	TEXT	NOT NULL	
current_mood	mood	NOT NULL	'ok'
updated_at	TEXT	NOT NULL	

- ✅ up
- ✅ down
- ❌ сравнение схем

```sql
-- migrate:up
ALTER TYPE mood ADD VALUE 'happy' AFTER 'ok';

-- migrate:down

```

Как брать snapshot-ы

Как брать snapshot-ы

INFORMATION_SCHEMA

ISO/IEC 9075-11

Как брать snapshot-ы

tables

constraints

pk, fk

indexes

triggers

INFORMATION_SCHEMA

sequences

types

functions

views, m-views

ISO/IEC 9075-11

Как брать snapshot-ы

tables

constraints

pk, fk

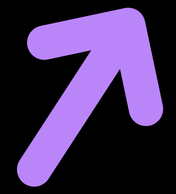
indexes

triggers

INFORMATION_SCHEMA

sequences

types



functions

views, m-views

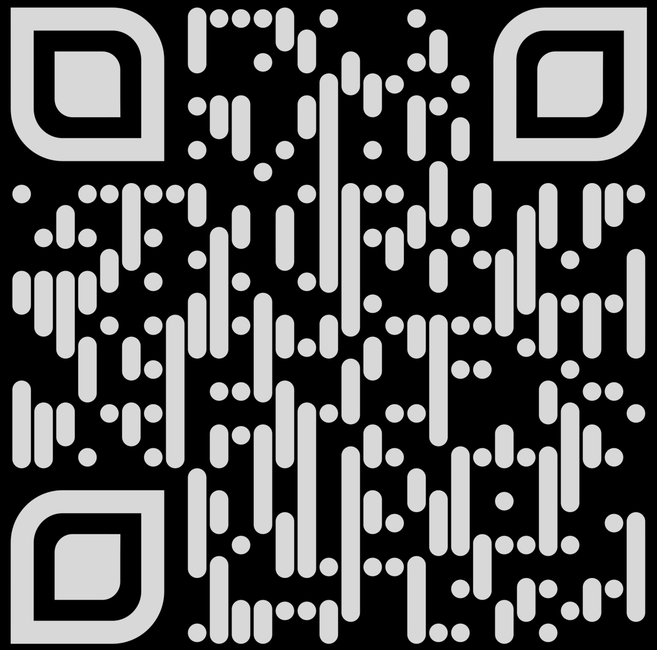
Python / Go & pg-driver

Сериализуем!

ISO/IEC 9075-11

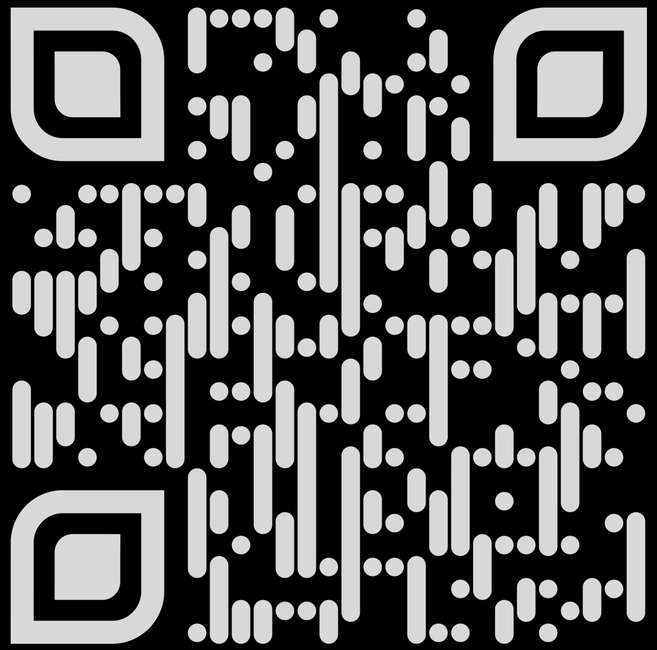
Как делать staircase

Как делать staircase



realkarych/seqwall

Как делать staircase



```
2025/05/03 01:07:42 Step 3 (up-down-up) completed successfully!
2025/05/03 01:07:42
🎉 Staircase test completed successfully!
❯
```

```
@@ -376,8 +376,8 @@
         },
         {
           "ColumnName": "updated_at",
-          "DataType": "timestamp with time zone",
-          "UDTName": "timestamptz",
+          "DataType": "timestamp without time zone",
+          "UDTName": "timestamp",
           "DateTimePrecision": {
             "Int64": 6,
             "Valid": true

❯
```

realkarych/seqwall

Промежуточные итоги

1. идемпотентность != соблюдение инварианта

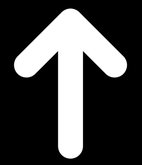
Промежуточные итоги

1. идемпотентность != соблюдение инварианта
2. используй снапшот-тесты

Всё? Победа?

Что не так?

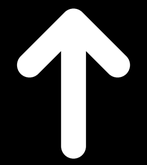
id	username	current_mood	updated_at
1	alice	happy	2025-03-14 12:00:00+03
2	bob	sad	2025-03-14 14:30:00+03
3	charlie	ok	2025-03-14 09:15:00+03



```
1  -- migrate:up
2  ALTER TABLE users
3    ALTER COLUMN updated_at TYPE timestamp
4    USING updated_at::timestamp;
5
6  -- migrate:down
7  ALTER TABLE users
8    ALTER COLUMN updated_at TYPE text
9    USING updated_at::text;
10
```

Что не так?

id	username	current_mood	updated_at
1	alice	happy	2025-03-14 12:00:00+0
2	bob	sad	2025-03-14 14:30:00+0
3	charlie	ok	2025-03-14 09:15:00+0



```
1  -- migrate:up
2  ALTER TABLE users
3    ALTER COLUMN updated_at TYPE timestamp
4    USING updated_at::timestamp;
5
6  -- migrate:down
7  ALTER TABLE users
8    ALTER COLUMN updated_at TYPE text
9    USING updated_at::text;
10
```

Что не так?

id	username	current_mood	updated_at
1	alice	happy	2025-03-14 12:00:00+0
2	bob	sad	2025-03-14 14:30:00+0
3	charlie	ok	2025-03-14 09:15:00+0

```sql
-- migrate:up
ALTER TABLE users
  ALTER COLUMN updated_at TYPE timestamp
  USING updated_at::timestamp;

-- migrate:down
ALTER TABLE users
  ALTER COLUMN updated_at TYPE text
  USING updated_at::text;

```

Пу-пу-пу, timestamptz...

А что там с данными?

Генерируем данные

Генерируем данные

Генерируем данные

- используем faker / greenmask / hypothesis / regex

Генерируем данные

- используем faker / greenmask / hypothesis / regex
- не репрезентативно

Генерируем данные

- используем faker / greenmask / hypothesis / regex
- не репрезентативно
- данные устаревают

Генерируем данные

- используем faker / greenmask / hypothesis / regex
- не репрезентативно
- данные устаревают
- больно с внешними фидами

Генерируем данные

- используем faker / greenmask / hypothesis / regex
- не репрезентативно
- данные устаревают
- больно с внешними фидами
- невозможно на больших и связанных схемах


PROGRESS
user_id
UUID • PK
team_id
UUID • PK
task_code
TEXT • PK
status
TEXT
user_id + team_id
team_id + task_code
MEMBERSHIPS
user_id
UUID • PK
team_id
UUID • PK
joined_at
TIMESTAMP
TEAM_TASKS
team_id
UUID • PK
task_code
TEXT • PK
title
TEXT


Hydra.

Сэмплируем данные

- тащим данные с прода

Сэмплируем данные

- тащим данные с прода
- нужна инфраструктура

Сэмплируем данные

- тащим данные с прода
- нужна инфраструктура
- хорошо сэмплировать — сложно

Сэмплируем данные

- тащим данные с прода
- нужна инфраструктура
- хорошо сэмплировать — сложно
- скорость тестов VS объем данных VS актуальность данных

Сэмплируем данные

- тащим данные с прода
- нужна инфраструктура
- хорошо сэмплировать — сложно
- скорость тестов VS объем данных VS актуальность данных
- используем greenmask / свой велосипед

Генерация VS Сэмплы

Генерация VS Сэмплы

- Генерируем, если:
  - нет возможности сэмплировать
  - мало схем, слабые связи
  - данные семантически-очевидны (хеши, почты, имена, id...)

Генерация VS Сэмплы

- Сэмплируем, если:
  - много схем
  - сложные связи, составные ключи
  - внешние фиды и неочевидные констрейнты

Данные нашли,
а что дальше?

А какие бывают миграции?

А какие бывают миграции?

1. **возвратные** — после up-down
   данные те же

А какие бывают миграции?

1. **возвратные** — после up-down данные те же
2. **невозвратные** — up влечет потерю данных

Data Snapshots

Data Snapshots

- snap1
- up
- down
- snap2

Data Snapshots

- snap1
- up
- down
- snap2

→ snap1==snap2

Профит!!

id	username	current_mood	updated_at
1	alice	happy	2025-03-14 12:00:00+03
2	bob	sad	2025-03-14 14:30:00+03
3	charlie	ok	2025-03-14 09:15:00+03

```
-- migrate:up
ALTER TABLE users
  ALTER COLUMN updated_at TYPE timestamp
  USING updated_at::timestamp;

-- migrate:down
ALTER TABLE users
  ALTER COLUMN updated_at TYPE text
  USING updated_at::text;

```

Поймаем ошибку на сравнении данных до и после up-down

Как брать snapshot-ы

Как брать snapshot-ы

```
1  pg_dump --data-only --column-inserts —file=<before/after>.sql db
2  diff <before>.sql <after>.sql
```

Как брать snapshot-ы

```
1  pg_dump --data-only --column-inserts –file=<before/after>.sql db
2  diff <before>.sql <after>.sql
```

realkarych/seqwall *To be continued...

Выводы

- тестировать без данных — сомнительно

Выводы

- тестировать без данных — сомнительно
- отлавливаем невозвратные миграции

Выводы

- тестировать без данных — сомнительно
- отлавливаем невозвратные миграции
- для сэмплов используем pg_dump + diff / самописный диффер

Ну теперь-то точно победа?

Вспоминаем прошлый пример

id	username	current_mood	updated_at
1	alice	happy	2025-03-14 12:00:00+03
2	bob	sad	2025-03-14 14:30:00+03
3	charlie	ok	2025-03-14 09:15:00+03

```
1  -- migrate:up
2  ALTER TABLE users
3    ALTER COLUMN updated_at TYPE timestamp
4    USING updated_at::timestamp;
5
6  -- migrate:down
7  ALTER TABLE users
8    ALTER COLUMN updated_at TYPE text
9    USING updated_at::text;
10
```

Вспоминаем прошлый пример

id	username	current_mood	updated_at
1	alice	happy	2025-03-14 12:00:00+03
2	bob	sad	2025-03-14 14:30:00+03
3	charlie	ok	2025-03-14 09:15:00+03

```
1  -- migrate:up
2  ALTER TABLE users
3    ALTER COLUMN updated_at TYPE timestamp
4    USING updated_at::timestamp
5
6  -- migrate:down
7  ALTER TABLE users
8    ALTER COLUMN updated_at TYPE text
9    USING updated_at::text;
10
```

access exclusive lock

Ловим локи

Ловим локи в рантайме

pg_locks & pg_stat_activity

```sql
SELECT
  l.locktype,
  l.mode,
  l.granted,
  a.query,
  a.state,
  a.wait_event_type,
  a.wait_event,
  a.query_start,
  a.pid
FROM pg_locks l
JOIN pg_stat_activity a ON l.pid = a.pid
WHERE NOT l.granted OR a.state != 'idle';

```

Ловим локи статикой

squawk

```
❯ squawk example-migration.sql
warning[constraint-missing-not-valid]: By default new constraints require a table scan and
 --> example-migration.sql:2:24
  |
2 | ALTER TABLE table_name ADD CONSTRAINT field_name_constraint UNIQUE (field_name);
  |                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  |
warning[disallowed-unique-constraint]: Adding a `UNIQUE` constraint requires an `ACCESS EXC
 --> example-migration.sql:2:28
  |
2 | ALTER TABLE table_name ADD CONSTRAINT field_name_constraint UNIQUE (field_name);
  |                            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  |
  = help: Create an index CONCURRENTLY and create the constraint using the index.

Find detailed examples and solutions for each rule at https://squawkhq.com/docs/rules
Found 2 issues in 1 file (checked 1 source file)
```

Copy

А вот такое видели?

```
1 Select a, b, c from
2 some_table as ST WHERE
3 st.c = st.a Order by b
4
```

А вот такое видели?

```
1 Select a, b, c from
2 some_table as ST WHERE
3 st.c = st.a Order by b
4
```

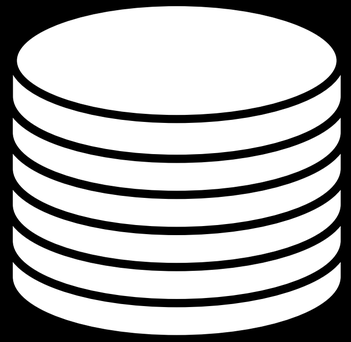
Линтеры

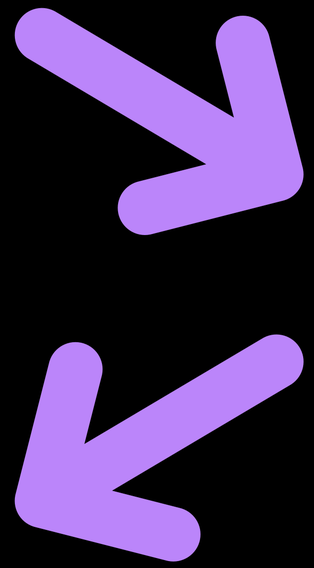
```
$ pip install sqlfluff
$ echo "  SELECT a  +  b FROM tbl;  " > test.sql
$ sqlfluff lint test.sql --dialect ansi
== [test.sql] FAIL
L:   1 | P:   1 | LT01 | Expected only single space before 'SELECT' keyword.
                      | Found '  '. [layout.spacing]
L:   1 | P:   1 | LT02 | First line should not be indented.
                      | [layout.indent]
L:   1 | P:   1 | LT13 | Files must not begin with newlines or whitespace.
                      | [layout.start_of_file]
L:   1 | P:  11 | LT01 | Expected only single space before binary operator '+'.
                      | Found '  '. [layout.spacing]
L:   1 | P:  14 | LT01 | Expected only single space before naked identifier.
                      | Found '  '. [layout.spacing]
L:   1 | P:  27 | LT01 | Unnecessary trailing whitespace at end of file.
                      | [layout.spacing]
L:   1 | P:  27 | LT12 | Files must end with a single trailing newline.
                      | [layout.end_of_file]

All Finished 📜 🎉!
```

Линтеры

```
$ pip install sqlfluff
$ echo "  SELECT a  +  b FROM tbl;  " > test.sql
$ sqlfluff lint test.sql --dialect ansi
== [test.sql] FAIL
L:   1 | P:   1 | LT01 | Expected only single space before 'SELECT' keyword.
                       | Found '  '. [layout.spacing]
L:   1 | P:   1 | LT02 | First line should not be indented.
                       | [layout.indent]
L:   1 | P:   1 | LT13 | Files must not begin with newlines or whitespace.
                       | [layout.start_of_file]
L:   1 | P:  11 | LT01 | Expected only single space before binary operator '+'.
                       | Found '  '. [layout.spacing]
L:   1 | P:  14 | LT01 | Expected only single space before naked identifier.
                       | Found '  '. [layout.spacing]
L:   1 | P:  27 | LT01 | Unnecessary trailing whitespace at end of file.
                       | [layout.spacing]
L:   1 | P:  27 | LT12 | Files must end with a single trailing newline.
                       | [layout.end_of_file]

All Finished 📜 🎉!
```

Надежный пайплайн





Надежный пайплайн

CR-1

CR-2

* Delivery *

Надежный пайплайн

CR-1

CR-2

* Delivery *

Роадмап

1. Пишем интеграционные тесты

Роадмап

1. Пишем интеграционные тесты
2. Включаем жестокий стат.анализ и линтеры

Роадмап

1. Пишем интеграционные тесты
2. Включаем жестокий стат.анализ и линтеры
3. Тестируем схему, локи

Роадмап

1. Пишем интеграционные тесты
2. Включаем жестокий стат.анализ и линтеры
3. Тестируем схему, локи
4. Тестируем данные

Роадмап

1. Пишем интеграционные тесты
2. Включаем жестокий стат.анализ и линтеры
3. Тестируем схему, локи
4. Тестируем данные
5. Тестируем многофазные миграции, регрессии перфоманса, ...

Пора тестировать миграции

фидбэк →

← слайды