PGSession 17

Usual Suspects

Que faire lorsque la base de données présente des signes inquiétants ?

Une histoire sur 3 jours..

pix

Qui sommes nous ?

Anne-Marie ESTEVES

Développeuse fullstack

pix

Equipe SRE

Pierre TOP

Développeur fullstack

tl;dr

Lorsque la base de données présente des **signes inquiétants**, les développeuses et développeurs ne savent pas toujours comment réagir.

Lorsqu’elle est administrée par un DBA, c'est vers lui qu'on se tourne. Mais si nous n’en avons pas ?

L’équipe SRE de Pix a pu, avec quelques notions PostgreSQL et un peu d’outillage sur le PaaS Scalingo, faire un premier diagnostic - et 🤞 résoudre le problème.

Attention à ne pas se précipiter sur le premier coupable venu…

Il y a un an..

Septembre est aux plateformes éducatives, comme Pix, ce qu'est décembre est aux grands magasins : un mois un peu fébrile où, avec les pics d'activité saisonniers, un imprévu peut prendre des proportions significatives.

En théorie, tout devrait bien se passer :

- côté front, les applications sont des SPA, donc l'activité est dans le navigateur client - les serveurs front sont de **simples nginx servant du js**;
- côté back, l'API REST **NodeJs** est stateless et scalée horizontalement avec l'**auto-scaler** de notre PaaS Scalingo;
- côté BDD, la seule brique stateful, on utilise un **plan PostgreSQL largement taillé**.

Aussi, lorsque la **base de données refuse les connexions** en une fin d'après-midi de septembre 2023, **l'équipe SRE** est intriguée.

C'était le premier symptôme d'un problème **qui allait durer 3 jours..**

Mardi soir : le signe avant-coureur

Constat

17h45: le nombre de connexions active augmente, puis la BDD refuse des connexions.

Si les connexions sont refusées, c'est parce que le nombre de connexions ouvertes a atteint le quota. Comme le nombre de connexions ouvertes est généralement très en dessous du quota, nous pensons tout de suite à une ressource (par exemple une table) qui n'a pas été libérée, et que la plupart des connexions attendent.

FAITS

- une MEP a démarré à 17h41
- la BDD ne répond plus (healthcheck) à 17h45

Réflexion

La requête de migration est en cours depuis 20 minutes. Son exécution devrait être rapide car elle ne modifie pas les enregistrements.

“*The DROP COLUMN form does not physically remove the column, but simply makes it invisible to SQL operations. Subsequent insert and update operations in the table will store a null value for the column. Thus, dropping a column is quick.*”

Mais pour démarrer, cette requête a besoin d’un accès exclusif sur la table. Il est probable qu’elle l’attende encore, et qu’elle bloque les requêtes suivantes.

```
 ?column?  | session_id |                    query                    | query_started_at | query_duration
-----------+------------+---------------------------------------------+------------------+----------------
 session=> |       6123 | alter table "users" drop column "lastLoggedAt" | 15:44:00         | 00:18:52
```

Outillage

La mise en production chez Pix est conçue pour minimiser les interruptions de service : les **modifications de schéma de base de données sont exécutées sans arrêter les applications**.

Dans certains cas, si au moins deux tables sont modifiées et qu'un même appel API utilise ces deux tables, un deadlock peut survenir. Cela ne s'est produit qu'une fois, et la solution de contournement a été documentée : arrêter l'API puis relancer la mise en production.

On a collectivement décidé de ne PAS s’outiller pour bloquer les merge. Par contre, une alerte est envoyée par Slack si un deadlock survient.

Action !

Côté Scalingo : Nous décidons d'arrêter tous les conteneurs API pour mettre fin aux requêtes SQL en cours. Cela n’a pas l’effet attendu !

Côté Postgres : Les requêtes SQL sont toujours actives sur la base de données, alors que les connexions réseau avec les clients ont été perdues. C'est un comportement documenté de PostgreSQL, **mais que nous ne connaissions pas**. Nous forçons l'arrêt (pg_terminate_backend) de toutes les requêtes SQL, puis relançons la mise en production. Ouf, tout se passe bien.

One Team

Nous sommes à ce moment-là près de **15 personnes** dans ce Meet.

L'objectif chez Pix est que les **mises en production soient des non-évènements :** les PO décident de les lancer, via Slack, sans aucune assistance de développeur.

Par contre, si le monitoring détecte une situation anormale, comme ici sur la BDD, l'équipe SRE, ainsi que d’autres développeurs intéressés, se rendent sur le **Meet dédié à la production**.

Bien qu'il ait été plus confortable de finir sa journée de travail, des développeurs de toutes les équipes, des PO et le support utilisateur nous ont rejoint.

Cela a permis d'explorer plusieurs pistes en parallèle, et de communiquer efficacement auprès des utilisateurs.

Mercredi 20: Batch, (a necessary) evil

Constat

Nous remarquons deux activités anormales la veille.

FAITS

- 17h15 : activité I/O multipliée par 4 sur BDD
- 17h30 - 17h45 : imports XML en échec
- 17h45: la BDD ne répond plus

↓ DATE	HOST	HTTP STATUS CODE	DURATION	URL PATH
Sep 19 17:45:06.576	router	499	59.97s	/api/organizations/11985/sco-organizati…
Sep 19 17:43:14.261	router	499	59.98s	/api/organizations/11985/sco-organizati…

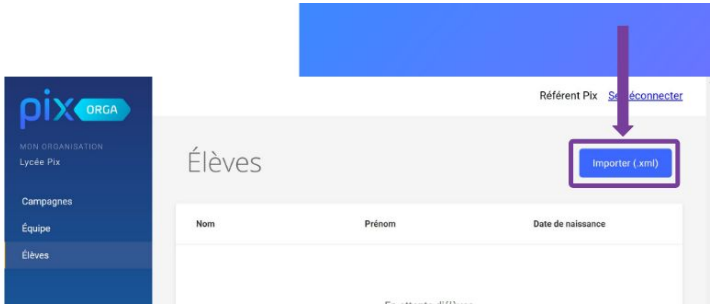
Sep 19 17:52:13.786	web-6	30.9min	/api/organizations/6938/sco-organizatio…
Sep 19 17:52:12.419	web-14	32.02min	/api/organizations/6938/sco-organizatio…
Sep 19 17:51:25.646	web-1	19.73min	/api/organizations/4577/sco-organizatio…

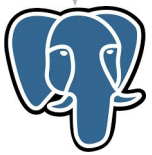
Réflexion

Pix s’adresse au grand public, et aussi aux élèves de l'éducation nationale. Dans ce cas, chaque classe suit un parcours particulier, choisi par son professeur. Pour cela, un responsable d'établissement importe dans Pix la liste des élèves, au format XML.

Nous aurions préféré que l’inscription des élèves se fasse par appel API unitaire, sur chaque élève. En effet, la taille du fichier XML dépend de la taille de l'établissement et peut atteindre plusieurs dizaines de milliers de lignes. L’année précédente, confronté au crash des conteneurs, nous avons implémenter du stream.

Réflexion





UPDATE organization_learners

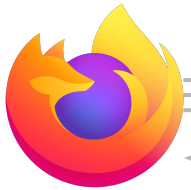
Réflexion

Cette année, il y a plus d'établissements, donc plus de trafic. Si au bout d'une minute, le fichier n'est pas intégré, alors un message d'erreur est affiché dans l'IHM.

L'utilisateur relance l'import.

Mais comme la précédente requête SQL n'est pas arrêtée, une nouvelle requête SQL s'empile, et la situation se détériore..

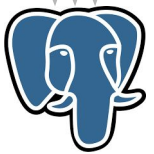
Réflexion

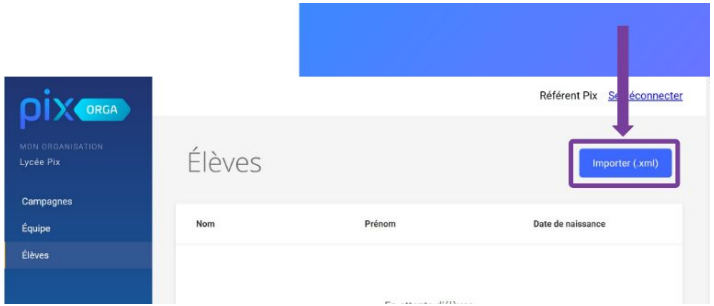


Pas de réponse avant 60 s :
Coupure par routeur API de
la connexion back + 504

Pas d'arrêt de
la requête REST

Pas d'arrêt de
la requête SQL





UPDATE organization_learners
UPDATE organization_learners
UPDATE organization_learners

Action !

Préventif :

- ajout d’un message IHM
- blocage de la route sur CDN Baleen en attendant la MEP

One Team

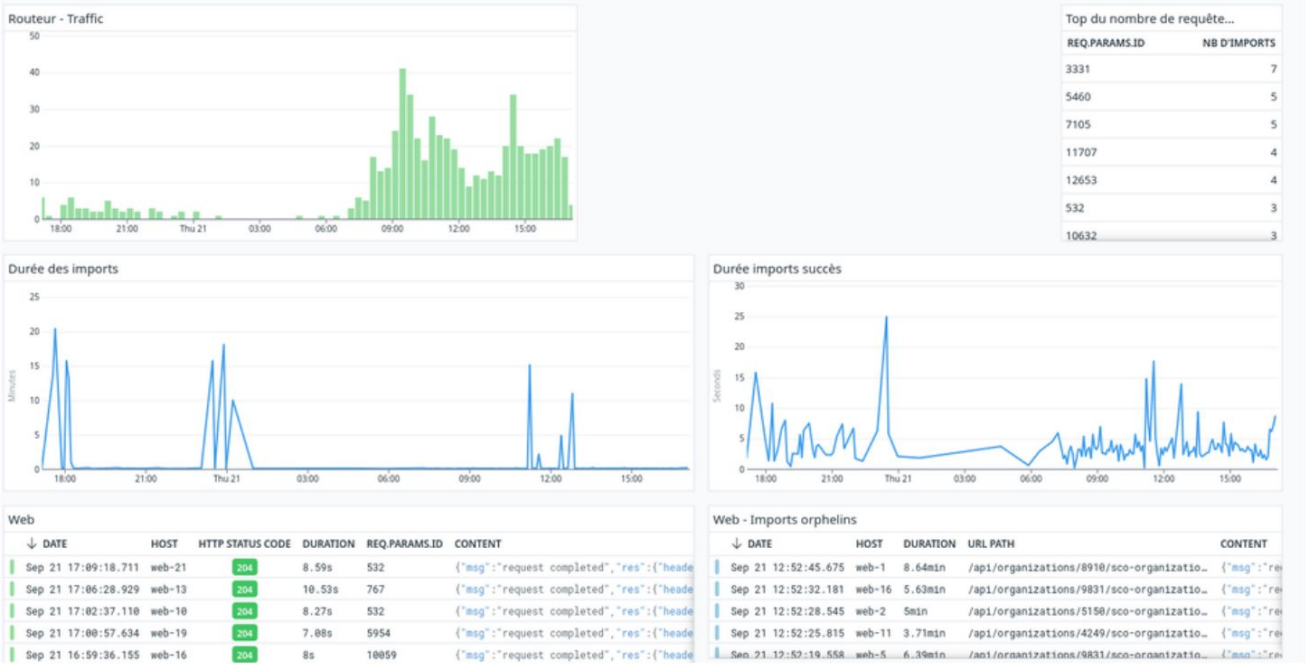
L’équipe SRE a passé la matinée avec la feature team en charge de l’import XML. Nous avons observé la production ensemble, réfléchi aux causes possibles, et compris que la combinaison infrastructure (timeout 60s) + IHM créait un cercle vicieux. Seuls, nous n’y aurions pas pensé.

La feature team a développé le correctif d’IHM en autonomie, et pouvait grâce à un dashboard, suivre la situation actuelle en production.

Nous avons aussi mitigé le risque collectivement : la solution d’ajouter du queuing (ex: pg_boss) sur ce endpoint n’était pas activable immédiatement.

Outillage

Dashboard Datadog pour suivre l’activité - et arrêter un import.



Jeudi 21 : VACUUM, a (necessary) evil

Constat

Un AUTOVACUUM a eu lieu, d'une durée conséquente. Ce ne sont pas les tables habituelles, mais la table **organization-learners,** celle alimentée par l'import de fichier XML. Comme la table est modifiée plus fréquemment sur cette période, alors les VACUUM doivent être plus fréquents.

Nous remarquons aussi une requête SQL longue, plus de 10 minutes, qui met à jour cette table.

FAITS : table organization-learners

- AUTOVACUUM long
- requête SQL longue en UPDATE

Outillage

Nous monitorons les VACUUM car il est arrivé qu'ils **impactent significativement les performances de la base de données** - lorsqu'ils passent sur les quelques tables volumineuses.

Comme Scalingo ne permet pas de suivre leur avancement en temps réel (table **pg_stat_progress_vacuum** inaccessible), nous obtenons via Metabase :

- leur avancement via la table **pg_stat_activity**
- la date de dernière exécution par table dans **pg_stat_user_tables.last_autovacuum**

Leur durée est tracée dans les logs avec l'option **log_autovacuum_min_duration** sur la table concernée. Les logs sont envoyés sur Datadog via un log drain.

Outillage

Nous monitorons les requêtes SQL longues en récupérant celles qui sont en cours depuis 10 minutes sur **pg_stat_activity** et en les envoyant à Datadog.

Cela est fait dans une application de type cron. Le repository pix-db-stats est open-source, configurable et avec tests automatisés.

Il permet aussi de récupérer :

- les métriques de BDD via l'API Scalingo;
- les statistiques agrégées des requêtes exécutées (**pg_stat_statements**).

Réflexion

```
UPDATE "organization-learners"
SET "isDisabled" = $1, "updatedAt" = CURRENT_TIMESTAMP
WHERE "organizationId" = $2 AND "isDisabled" = FALSE;
```

Cette requête est exécutée lors de l'import du fichier XML, et **désactive tous les élèves d'un établissement**. Est-il normal que son exécution prenne autant de temps ?

Pour en avoir le cœur net, nous exécutons la requête en production* sur le même établissement. Elle prend moins d'une seconde.

A-t-elle été longue parce que :

- les ressources (CPU, RAM, I/O) étaient accaparées ?
- elle attendait l'obtention d'un verrou sur la table ?

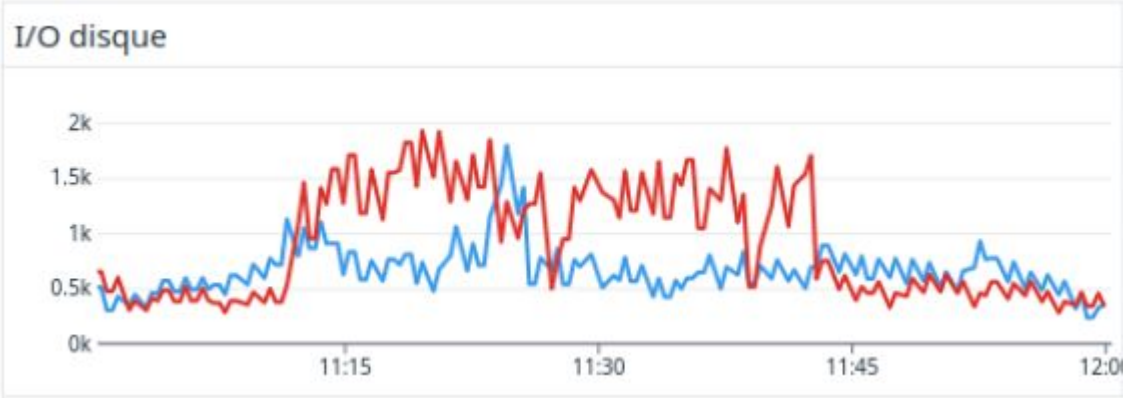
** avec transaction et rollback ^^*

Réflexion

A **11h15**, les I/O sont, comme avant la MEP de mardi, fois multipliés par 4.
On exécute à nouveau la requête, et elle prend cette fois plus d'une minute.

A **11h45**, un log d'AUTOVACUUM sur cette table apparaît.
Il semble bien que ce soit l'AUTOVACUUM qui ralentisse la requête.

Comment en être sûr ?

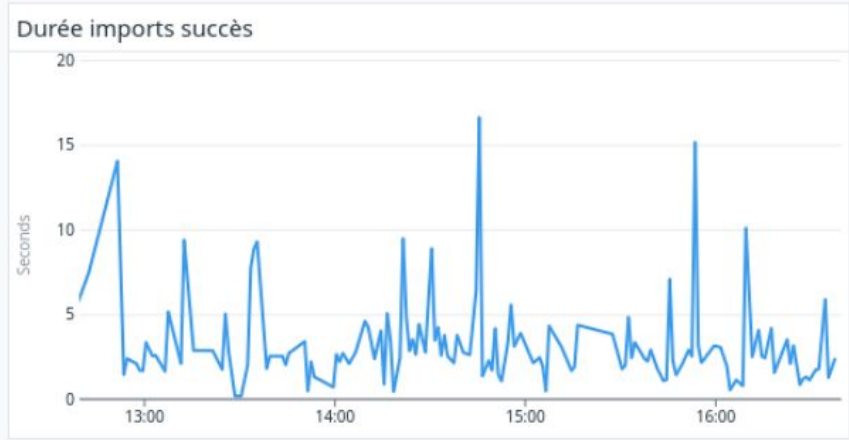


Action !

On désactive les AUTOVACUUM sur cette table pour l’après-midi.

L’effet est immédiat: la durée des imports passe de quelques minutes à quelques secondes.

On le réactive en partant le soir, pensifs.



FAITS :

- l’AUTOVACUUM impacte les performances de l’import XML

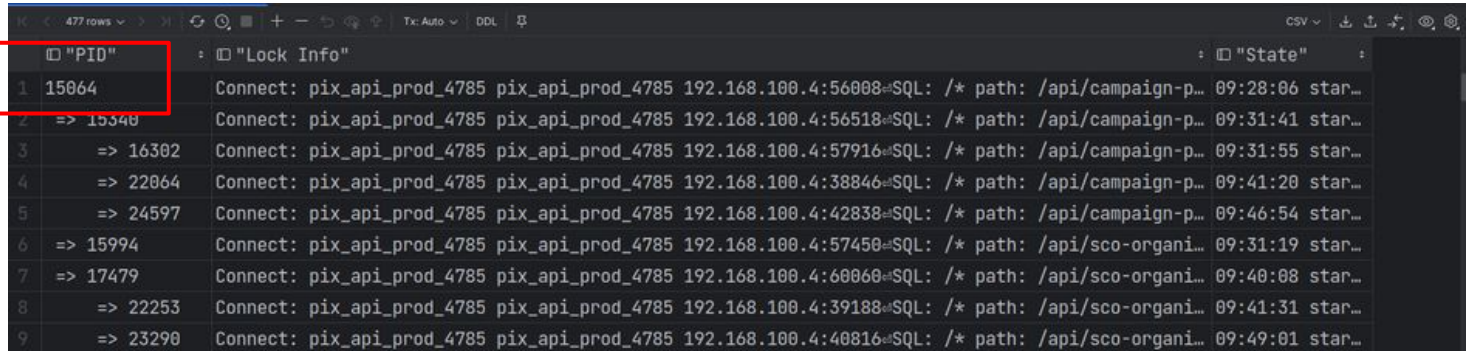
Vendredi 22: Eurêka !

Constat

FAITS : nombre de connexions élevé sur la BDD

A **11h52**, nouvelle alerte sur le nombre de connexions BDD. L'AUTOVACUUM est pourtant désactivé.

Nous consultons l'arbre des verrous : une requête bloque la quasi-totalité du trafic. Que fait-elle ?



	"PID"	"Lock Info"	"State"
1	15064	Connect: pix_api_prod_4785 pix_api_prod_4785 192.168.100.4:56008 SQL: /* path: /api/campaign-p…	09:28:06 star…
2	=> 15340	Connect: pix_api_prod_4785 pix_api_prod_4785 192.168.100.4:56518 SQL: /* path: /api/campaign-p…	09:31:41 star…
3	=> 16302	Connect: pix_api_prod_4785 pix_api_prod_4785 192.168.100.4:57916 SQL: /* path: /api/campaign-p…	09:31:55 star…
4	=> 22064	Connect: pix_api_prod_4785 pix_api_prod_4785 192.168.100.4:38846 SQL: /* path: /api/campaign-p…	09:41:20 star…
5	=> 24597	Connect: pix_api_prod_4785 pix_api_prod_4785 192.168.100.4:42838 SQL: /* path: /api/campaign-p…	09:46:54 star…
6	=> 15994	Connect: pix_api_prod_4785 pix_api_prod_4785 192.168.100.4:57450 SQL: /* path: /api/sco-organi…	09:31:19 star…
7	=> 17479	Connect: pix_api_prod_4785 pix_api_prod_4785 192.168.100.4:60060 SQL: /* path: /api/sco-organi…	09:40:08 star…
8	=> 22253	Connect: pix_api_prod_4785 pix_api_prod_4785 192.168.100.4:39188 SQL: /* path: /api/sco-organi…	09:41:31 star…
9	=> 23290	Connect: pix_api_prod_4785 pix_api_prod_4785 192.168.100.4:40816 SQL: /* path: /api/sco-organi…	09:49:01 star…

“ Cette attente est insoutenable. J'espère qu'elle va durer longtemps. ”

Oscar Wilde (L'Importance d'être Constant)

Constat

```
1  /* path: /api/campaign-participations */ update "organization-learners" set "isDisabled" = $1 returning "id"
```

Réflexion

```
1  /* path: /api/campaign-participations */ update "organization-learners" set "isDisabled" = $1 returning "id"
```

Silence sur le Meet où nous travaillons avec la feature team : une requête SQL est en train de réécrire toute la table (aucune clause WHERE) !

Le commentaire en début de ligne nous indique qu'il s'agit bien de l'API. Nous retrouvons le code, endpoint `POST api/campaign-participations`.

```
const [{ id }] = await queryBuilder('organization-learners').update({ isDisabled: false }).returning('id');
```

Cette PR a été mergée il y a trois mois. Lorsqu'un élève désactivé rejoint une campagne, l'intégralité des élèves est réactivé !

Réflexion

Action !

Arrêt de la requête SQL.

Blocage de la route API concernée sur le CDN.

Action !

Vérification des données :

- la requête a–elle déjà été exécutée auparavant ? oui !
- tous les élèves ont-ils été réactivés ? non
- .. ouf !

Pas de reprise de données nécessaire.

↓ DATE	HOST	SERVICE	POSTGRES.QUERY	CONTENT
Sep 22 02:18:59.234	postgresql-2	pix-api-production	update "organization-learners" set "isD…	duration: 3444765.471 ms exe
Sep 20 17:38:57.659	postgresql-2	pix-api-production	update "organization-learners" set "isD…	duration: 2068426.121 ms exe
Sep 20 15:32:11.041	postgresql-2	pix-api-production	update "organization-learners" set "isD…	duration: 2826279.724 ms exe

Action !

Implémentation et déploiement du hotfix.

Réactivation de la route.

Réactivation de l’AUTOVACUUM.

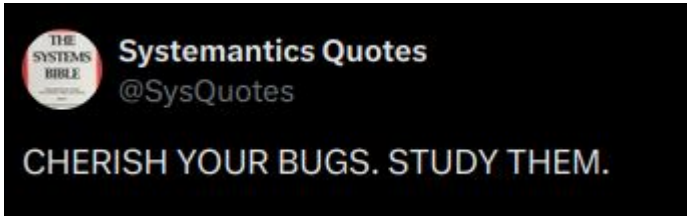
Outillage

Nous pouvons corréler l'appel API et les requêtes SQL grâce à une solution custom, avec 2 monkey-patching utilisant le async local storage de Node:

- Request dans Hapi et
- QueryBuiler.toSQL dans Knex.

La solution classique aurait été d'utiliser un APM, comme Jaeger.

La morale de cette histoire ?



One Team

La PR introduisant le bug avait passé les tests automatisés, la revue de code et les tests manuels. Après coup, le bug semble évident, mais nous sommes humains.

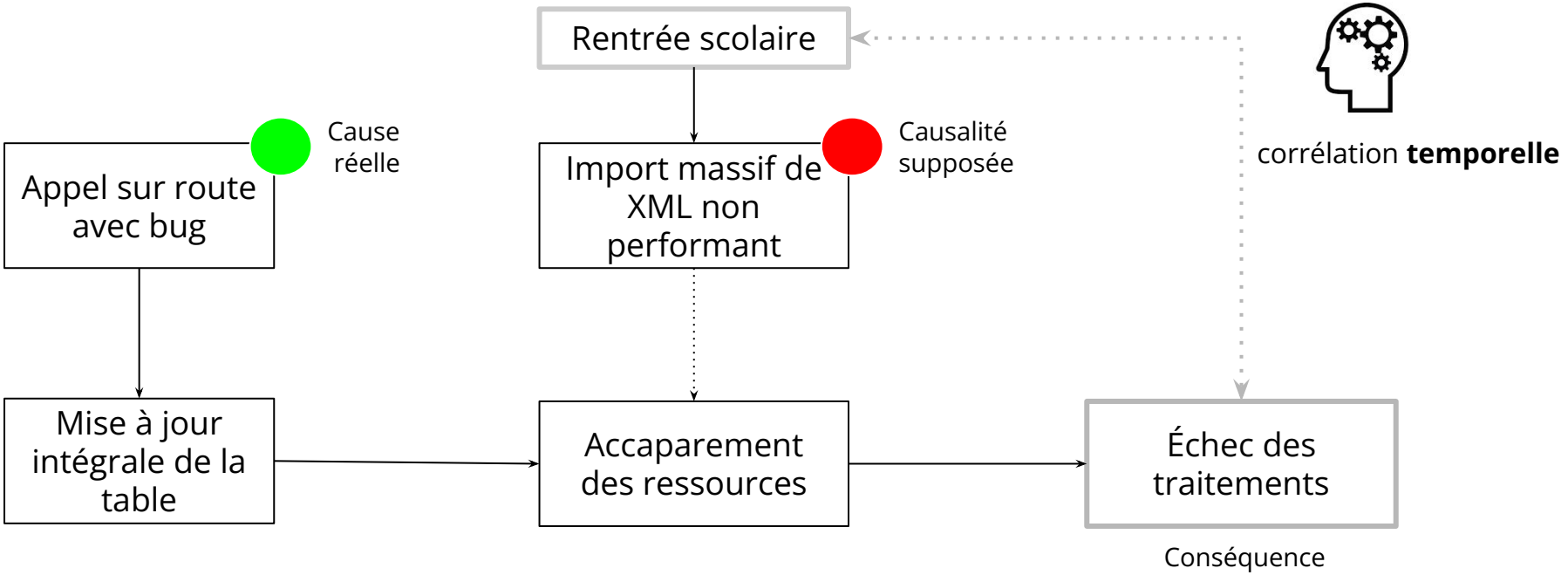
“Critique code instead of people – be kind to the coder, not to the code”

Egoless Programming, commandement n°10

L’équipe SRE et les autres feature teams ont travaillé ensemble, dans le même Meet, trois jours durant.

Corrélation n'implique pas causalité

"Si ça plante pendant la rentrée, c'est forcément l'import XML.."

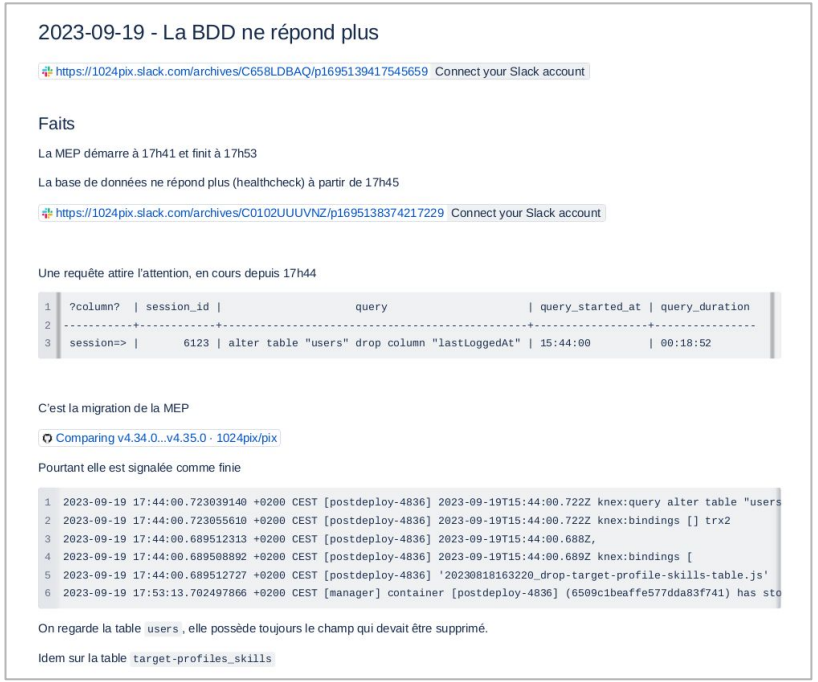


Se souvenir

- consigner dans des journaux de bord , comme pour la navigation en mer

- raconter l'histoire, en interne et en externe

- montrer : la codebase open source permet de recréer en live la situation !



2023-09-19 - La BDD ne répond plus

https://1024pix.slack.com/archives/C658LDBAQ/p1695139417545659 Connect your Slack account

Faits

La MEP démarre à 17h41 et finit à 17h53

La base de données ne répond plus (healthcheck) à partir de 17h45

https://1024pix.slack.com/archives/C0102UUUVNZ/p1695138374217229 Connect your Slack account

Une requête attire l'attention, en cours depuis 17h44

```
 ?column?  | session_id |                    query                    | query_started_at | query_duration
-----------+------------+---------------------------------------------+------------------+----------------
session=> |       6123 | alter table "users" drop column "lastLoggedAt" | 15:44:00         | 00:18:52
```

C'est la migration de la MEP

Comparing v4.34.0...v4.35.0 · 1024pix/pix

Pourtant elle est signalée comme finie

```
2023-09-19 17:44:00.723039140 +0200 CEST [postdeploy-4836] 2023-09-19T15:44:00.722Z knex:query alter table "users
2023-09-19 17:44:00.723055610 +0200 CEST [postdeploy-4836] 2023-09-19T15:44:00.722Z knex:bindings [] trx2
2023-09-19 17:44:00.689512313 +0200 CEST [postdeploy-4836] 2023-09-19T15:44:00.688Z,
2023-09-19 17:44:00.689508892 +0200 CEST [postdeploy-4836] 2023-09-19T15:44:00.689Z knex:bindings [
2023-09-19 17:44:00.689512727 +0200 CEST [postdeploy-4836] '20230818163220_drop-target-profile-skills-table.js'
2023-09-19 17:53:13.702497866 +0200 CEST [manager] container [postdeploy-4836] (6509c1beaffe577dda83f741) has sto
```

On regarde la table `users`, elle possède toujours le champ qui devait être supprimé.

Idem sur la table `target-profiles_skills`

Journal de bord PIX (Wiki)

Se méfier, surtout de soi-même..

“The greatest trick the Devil ever pulled was to convince the world he didn't exist.”

Merci !

La version longue est sur https://blog.octo.com/usual-suspects