
ARTEFACT
DATA IS ABOUT PEOPLE

ARTEFACT

DATA IS ABOUT PEOPLE

L'histoire cauchemardesque d'un SPOF de l'architecture data qui est tombé.

SRE France meetup - Benoît Goujon - 13 février 2023

Hey.

Benoît Goujon

Data engineer chez Artefact

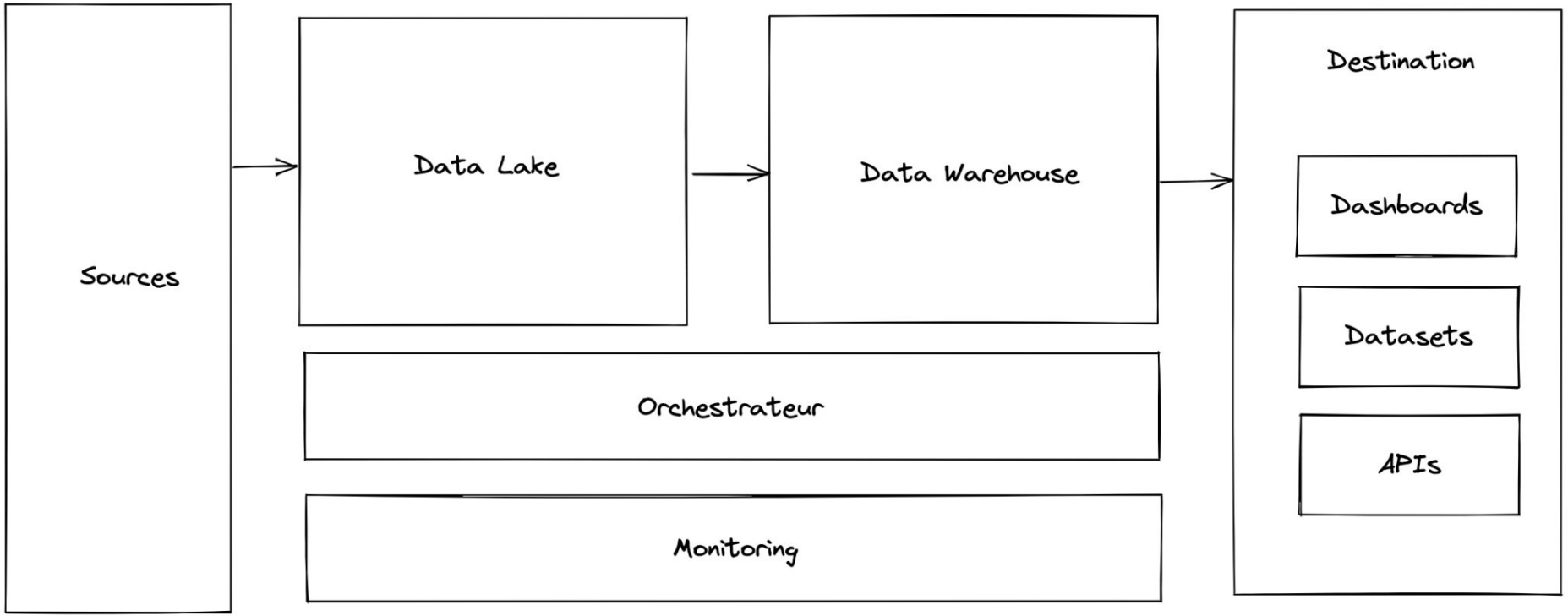
Chez Artefact

- Aide les entreprises à tirer parti de leur donnée à travers
  - La construction de data platforms
  - L'industrialisation de cas d'usage data
- Nouvelle mission en tant que SRE / DRE au sein de la data factory d'un grand groupe du luxe

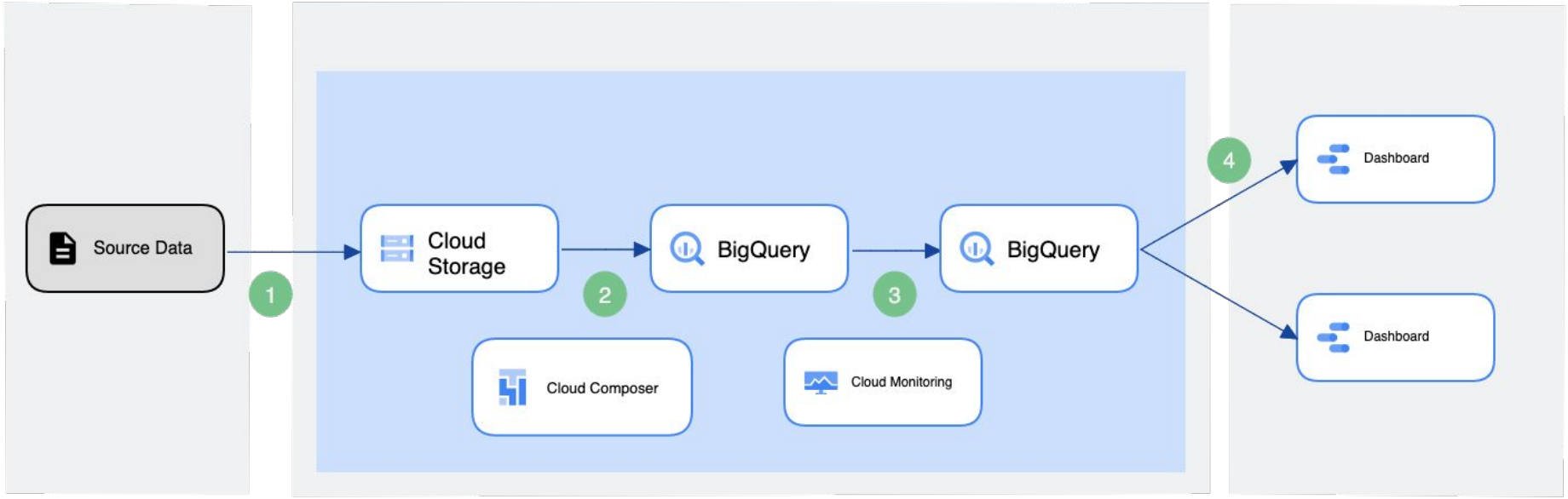
Table of content.

1. L'orchestrateur, le composant clé d'une stack data.
2. La déclaration de l'incident.
3. Les premières étapes de mitigation.
4. L'analyse de la root cause et la résolution technique.
5. L'incident comme opportunité de changement.
6. Les enseignements clés.

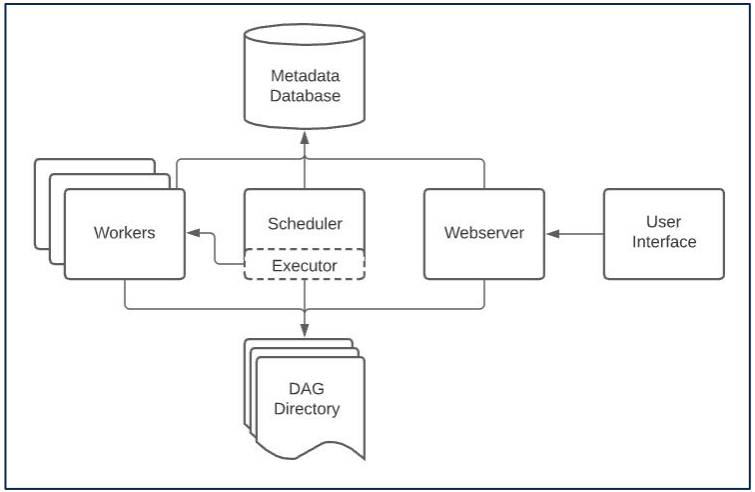
Une stack data regroupe souvent les mêmes composants.
L'orchestrateur est une brique centrale.



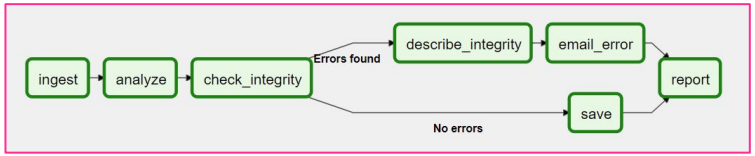
Sur Google Cloud, la stack data se décline avec Composer comme orchestrateur.



Cloud Composer repose sur le projet Apache Airflow exécuté sur Kubernetes. Le client l'utilise de manière intensive.



Architecture Airflow



Anatomie d'un DAG Airflow

En quelques chiffres clés

—

Configuration de l'environnement

- 1 scheduler
- 3 workers
- 18 tâches exécutables en parallèle

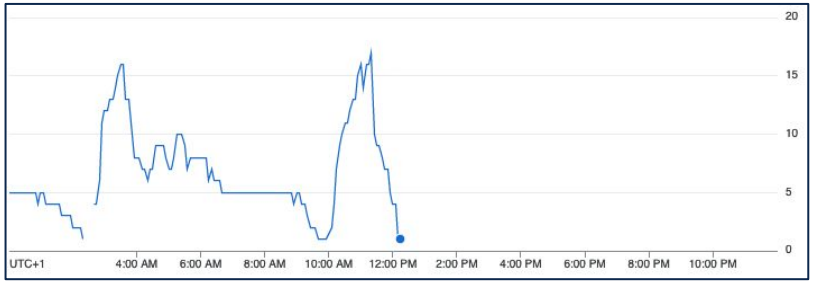
Charge

- 220 DAGs

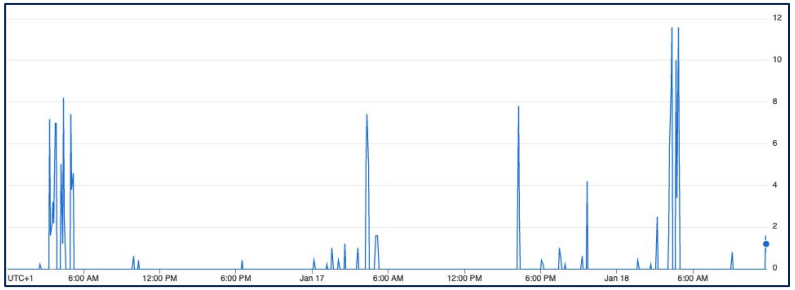
Table of content.

1. L'orchestrateur, le composant clé d'une stack data.
2. La déclaration de l'incident.
3. Les premières étapes de mitigation.
4. L'analyse de la root cause et la résolution technique.
5. L'incident comme opportunité de changement.
6. Les enseignements clés.

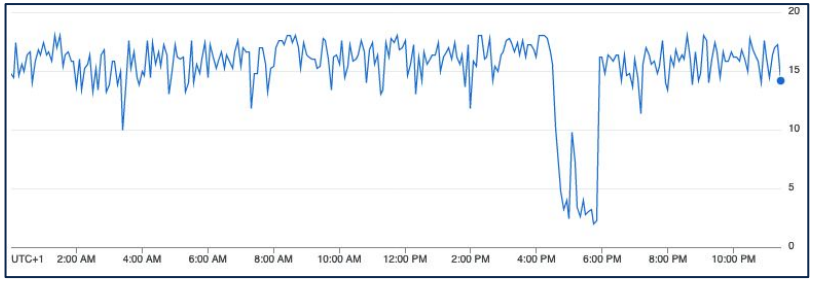
L'alerte est venue d'un utilisateur qui a remarqué que la donnée n'était pas à jour. Une rapide investigation a permis de comprendre que l'incident provenait de l'orchestrateur.



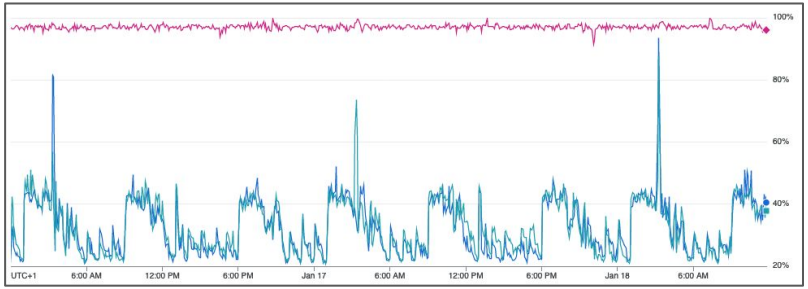
Nombre de DAGs en échec



Nombre de tâches dans la queue



Nombre de tâches en cours d'exécution



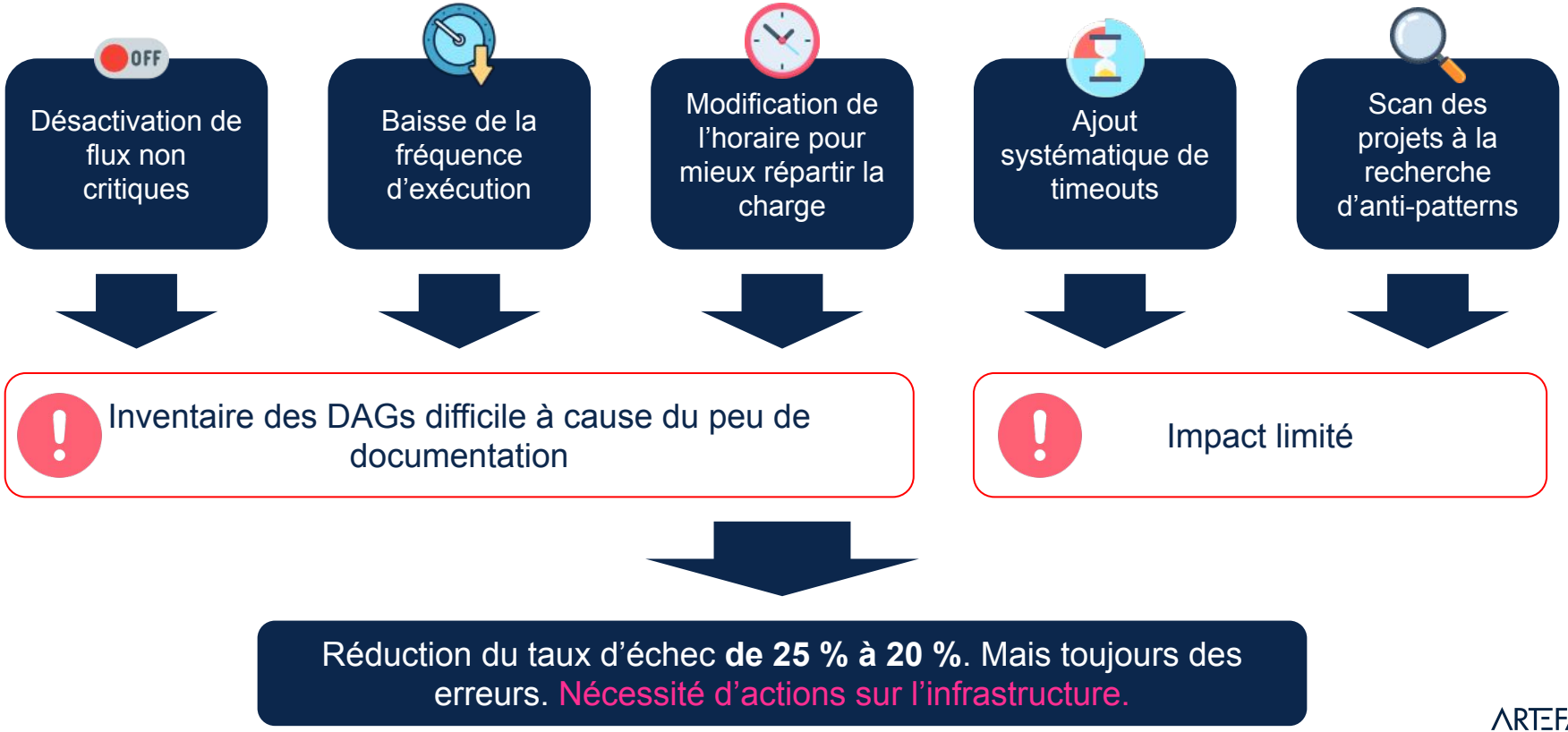
Utilisation CPU sur chaque noeud

ARTEFACT

Table of content.

1. L'orchestrateur, le composant clé d'une stack data.
2. La déclaration de l'incident.
3. Les premières étapes de mitigation.
4. L'analyse de la root cause et la résolution technique.
5. L'incident comme opportunité de changement.
6. Les enseignements clés.

Après avoir communiqué un premier statut, l'équipe a identifié des pistes de mitigation pour réduire la charge.



ARTEFACT

Table of content.

1. L'orchestrateur, le composant clé d'une stack data.
2. La déclaration de l'incident.
3. Les premières étapes de mitigation.
4. L'analyse de la root cause et la résolution technique.
5. L'incident comme opportunité de changement.
6. Les enseignements clés.

Après les pistes de mitigation, un travail sur l'infrastructure a été engagé. Mais cela a été plus difficile que prévu.

Equipe infra indépendante qui avait les accès au repo d'infra as code et les permissions pour faire des changements

Plus suffisamment d'IPs disponibles dans le sous réseau de la node pool. Risque trop élevé d'en créer une nouvelle.

Environnement provisionné en 2021. Version de Composer v1 plus supportée. Impossible de recréer un service avec la même version

Décision de créer un nouveau service Composer v2.

Table of content.

1. L'orchestrateur, le composant clé d'une stack data.
2. La déclaration de l'incident.
3. Les premières étapes de mitigation.
4. L'analyse de la root cause et la résolution technique.
5. L'incident comme opportunité de changement.
6. Les enseignements clés.

Plutôt que de faire une migration “lift and shift”, nous avons choisi de revoir en profondeur l’utilisation de Composer.

Gouvernance de l’environnement

- Revue du DAG avant passage en production
  - Revue du code
  - Revue de la documentation
- Audit des droits
- Policies
  - Owner obligatoire
  - Tags pour donner du contexte
  - Nom des DAGs standardisé

Introduction et utilisation de nouvelles fonctionnalités

- Utilisation des datasets
- Utilisation de nouveaux opérateurs
- Environnement de développement en local
- Nouvelle façon plus robuste de gérer les secrets
- Configuration orientée haute disponibilité
- AUTOSCALING 🤩

Cette migration nécessitait du temps. Nous avons donc accéléré cette migration grâce à plusieurs initiatives.

Documentation, documentation, documentation

- Documentation de référence
- How-to
- Troubleshooting guides
- Snippets de code prêts à l'emploi

Mise à dispo d'un environnement local

- Utilisation d'une image Docker avec les mêmes caractéristiques que la prod

Travail sur la Developer Experience

- Messages d'erreur actionnables sur les policies

Support

- Discussion Teams dédiée à l'incident
- Partage des erreurs rencontrées et solutions trouvées

Table of content.

1. L'orchestrateur, le composant clé d'une stack data.
2. La déclaration de l'incident.
3. Les premières étapes de mitigation.
4. L'analyse de la root cause et la résolution technique.
5. L'incident comme opportunité de changement.
6. Les enseignements clés.

Cet incident a été une formidable opportunité d'apprendre collectivement et repartir sur des bases saines.

Bilan

- Statut : en cours
- Impact : modéré. Pas de perte de CA directe. Perte en efficacité opérationnelle du staff.
- Délai de résolution : trop long

Enseignements clés

- Une équipe doit prendre la responsabilité de monitorer un service aussi critique pour éviter d'être pris par surprise
- L'infra, d'autant plus qu'elle est *as code,* doit pouvoir être changée sans crainte. La scalabilité doit être pensée dès la conception.
- Des mauvaises pratiques de développement combinées à un manque de documentation peuvent conduire à une hausse importante du temps de résolution
- Un incident doit être traité comme une opportunité d'apprendre collectivement et challenger l'existant

Merci.