Architecture Développement-Déploiement ML

Agenda

- Dans cette présentation, on passe en revue les points suivants
  - Principes de mise en production en continue
  - Problèmes

Qu'est-ce que l'intégration continue?

- L'intégration continue (CI) est une pratique de développement logiciel dans laquelle de petits changements sont continuellement intégrés dans la base de code du projet. Chaque modification est automatiquement testée pour garantir que le projet fonctionnera comme prévu pour les utilisateurs finaux dans un environnement de production.
- Dans un projet de science de données le pipeline de données comme une série de tâches ordonnées
- Chaque bloc bleu représente une tâche de pipeline

workflow ML-CI

- Un scientifique des données pousse les changements de code (par exemple, modifie l'une des tâches du pipeline)
- La poussée déclenche le service CI pour exécuter le pipeline de bout en bout et tester chaque artefact généré
- Si les tests réussissent, une révision du code suit
- Si les modifications sont approuvées par le réviseur, le code est fusionné.
- Chaque matin, le pipeline «production» (dernier commit dans la branche principale) s'exécute de bout en bout et envoie le rapport aux analystes métiers.

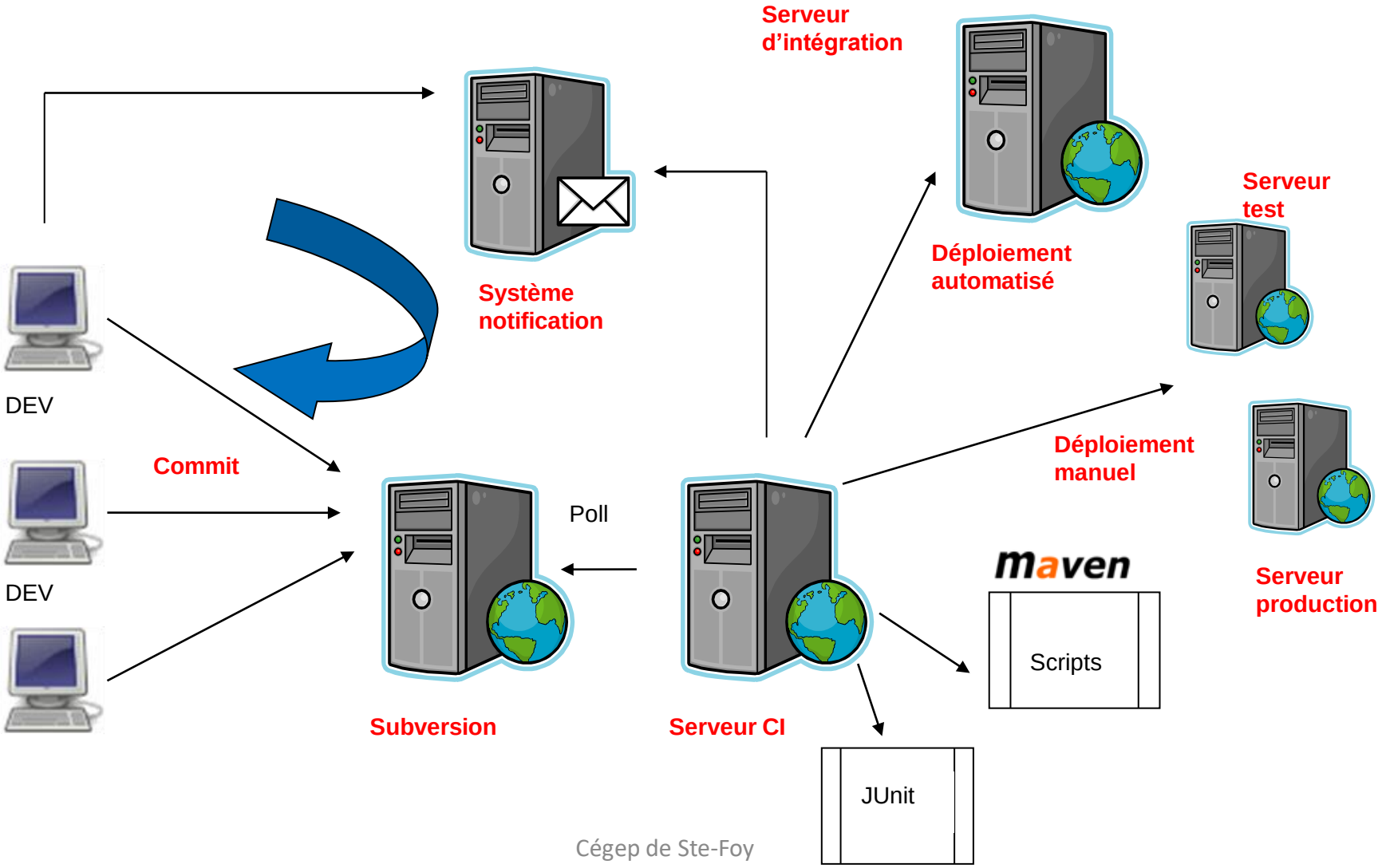
workflow ML-CI- avantages

- Les bogues sont détectés dans la phase de développement, au lieu de la production
- Toujours prêt pour la production: comme nous avions besoin de changements de code pour réussir tous les tests avant de les intégrer à la branche principale, nous nous assurons de pouvoir déployer notre dernière fonctionnalité stable en continu en déployant simplement le dernier commit dans la branche principale

Intégration continue: problèmes

- **Cycles de développement traditionnels**
  - L'intégration est assez longue et difficile
  - Progrès du projet est difficile en terme de visibilité
  - Incidents ou bugs sont difficiles à cerner et régler

Intégration continue: les composantes



Intégration continue: bénéfices

- Processus d'intégration souple
- Test de régression automatique
- Release du produit régulière
- Test fonctionnel assez tôt dans le cycle
- Résolution des bugs rapide
- Visibilité du projet

Pour démarrer en IC

- Processus de build automatique (**Maven** ou **Ant**)
- Tests automatiques (**Junit**, **Selenium**)
- Référentiel pour le code (**Subversion**, **CVS**)
- Serveur d'intégration continue (**Cruise control**, **Jenkins**, **continuum**)

Intégration continue en quelques mots

- Suivi des problèmes d'intégration
- Publication automatique des artifacts (Maven)
- Suivi du processus de build
- Suivi et rapport sur la qualité du code

Cycle de développement-déploiement ML

- Collecte-Acquisition
- Exploration
- Pré-traitement
- Modele
- Mise en production
- Utilisation

Comparaison avec cycle du logiciel

- Les modèles qu'on doit déployer doivent être:
  - Reproductible
  - Testable
  - Peuvent passer un audit
  - Surtout: capable d'etre continuellement amélioré

Problèmes avec cette approche ML

- Il n'y a pas de solutions standard
  - Chaque projet a ses propres contraintes
- Processus est difficile à tester
- Chaque étape demande une certaine expertise
  - Collecte: Data engineer
  - Exploration: Stats, Viz
  - Etc.
- Amélioration d'une solution est encore plus difficile

Sources de changement

- Données
  - Modèle de données
  - Sources de données peuvent varier dans le temps
  - Volume/Vitesse
  - etc

Sources de changement | suite

- Modèle
  - État de l’art change très vite
  - Beaucoup de recherche dans le domaine
  - Performance bonne aujourd’hui n’est pas garantie pour demain!

Sources de changement | suite

- Code et application pour consommer le modele
  - Nouvelle demande de clients usagers
  - Bugs
  - Dépendances qui peuvent changer dans le temps
  - Etc.

Mise en production continue

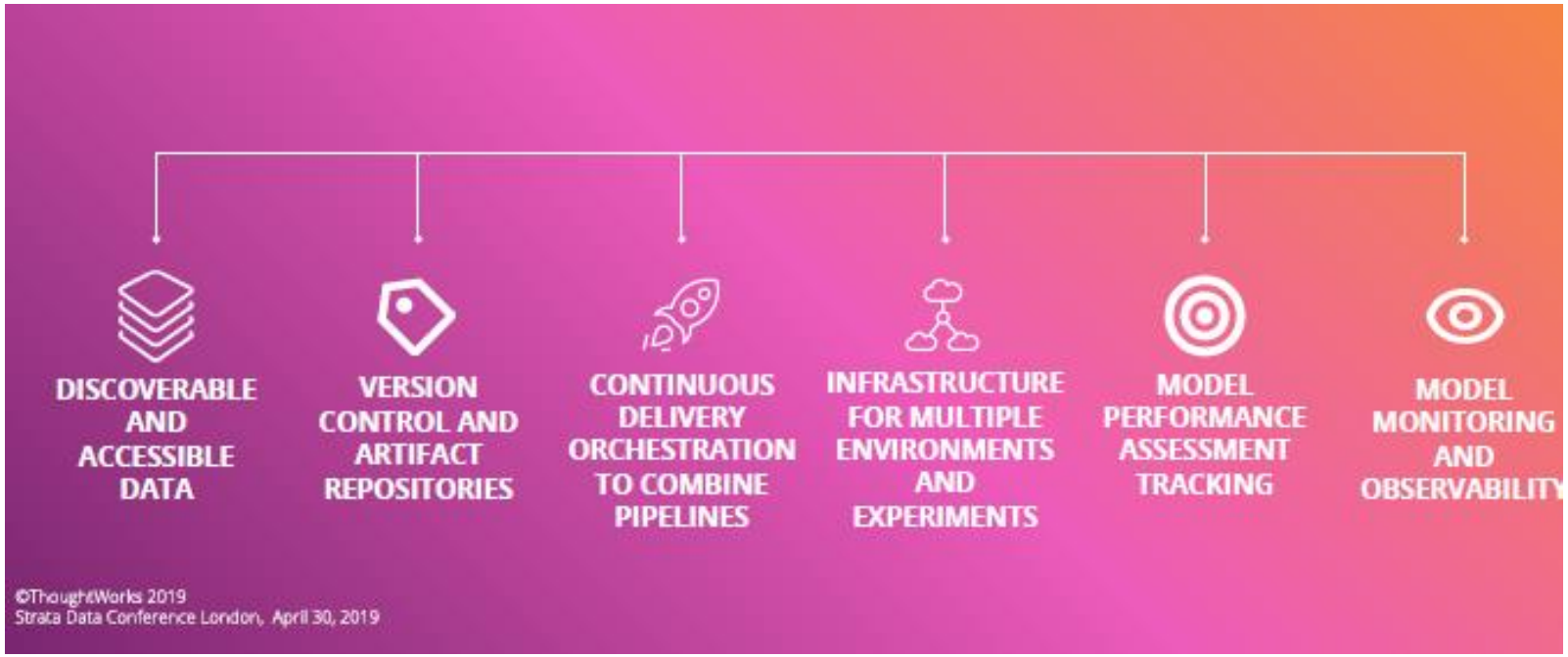
- La mise en production en continue (continuous delivery) permet:
  - De prendre les changements tels que les nouvelles fonctionnalités
  - Bugs,
  - Nouveau data
  - Etc.
- Et les met en production
  - De façon efficace
  - Rapide

Mise en production continue

Principe

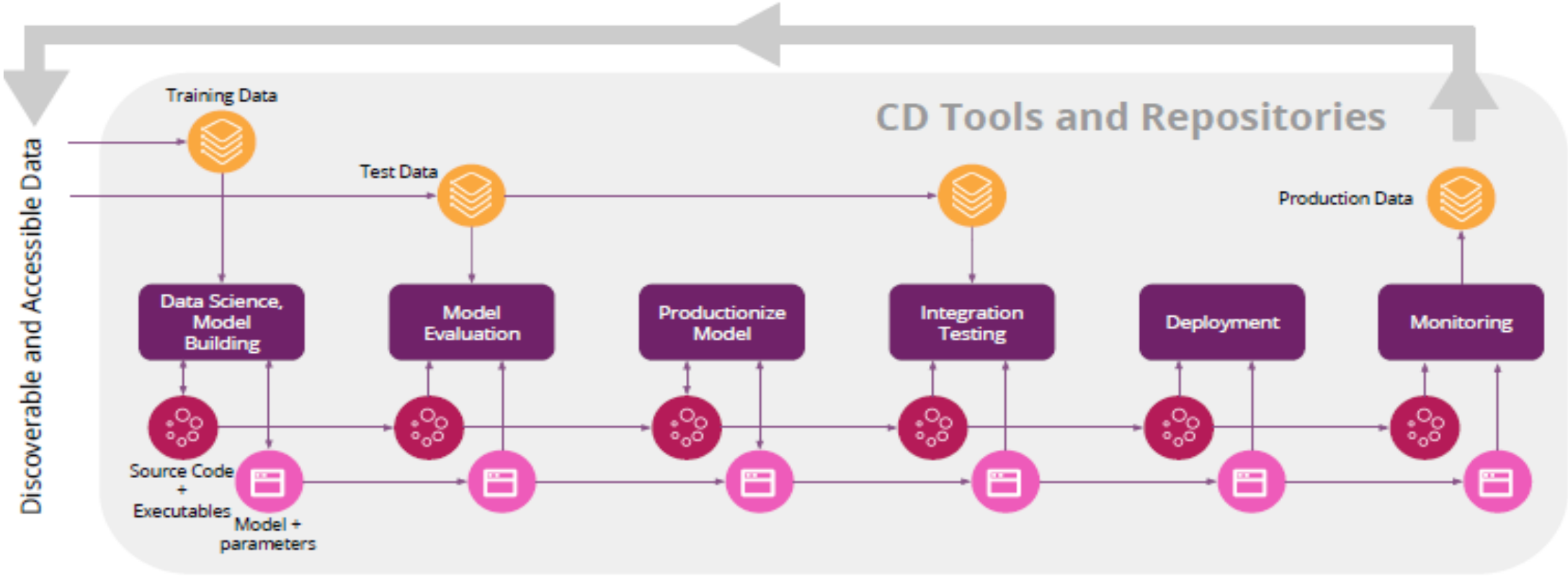
- Définir et créer un processus répétable et fiable pour la mise en production du modèle
- Permettre d'automatiser la presque totalité du processus
- Permettre de faire un suivi de changement (change control)
- Permettre de garder tous les artefacts dans un gestionnaire d'artefacts
- Suivre et gérer le cycle du modèle de manière continue

Stack



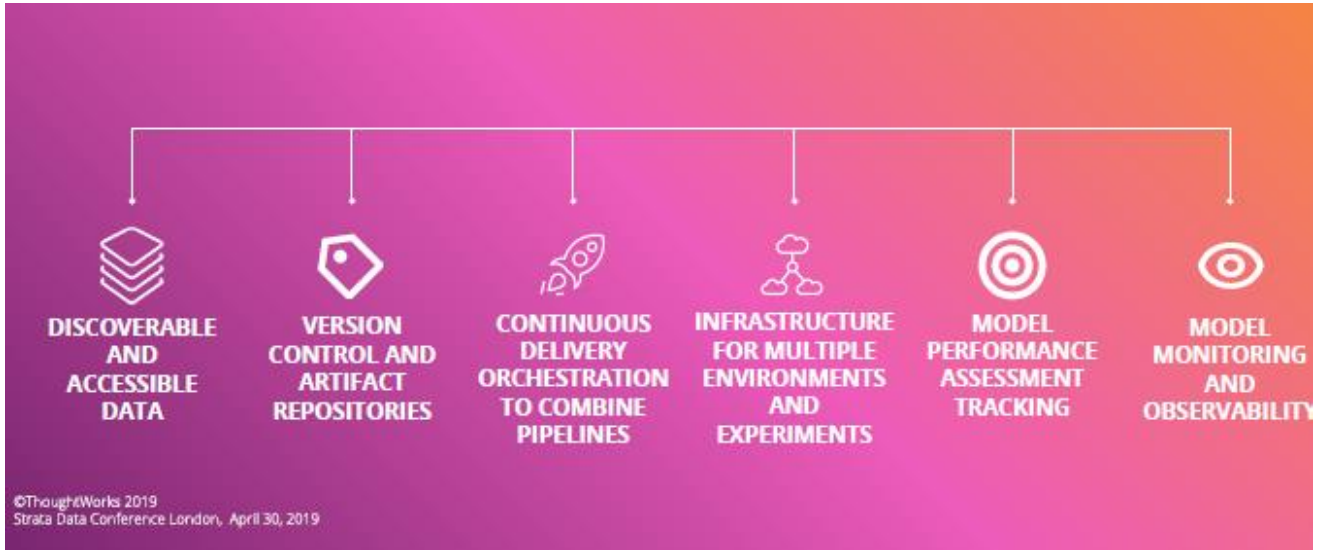
Réf: Thoughtworks 2019 Strata data conference London 2019

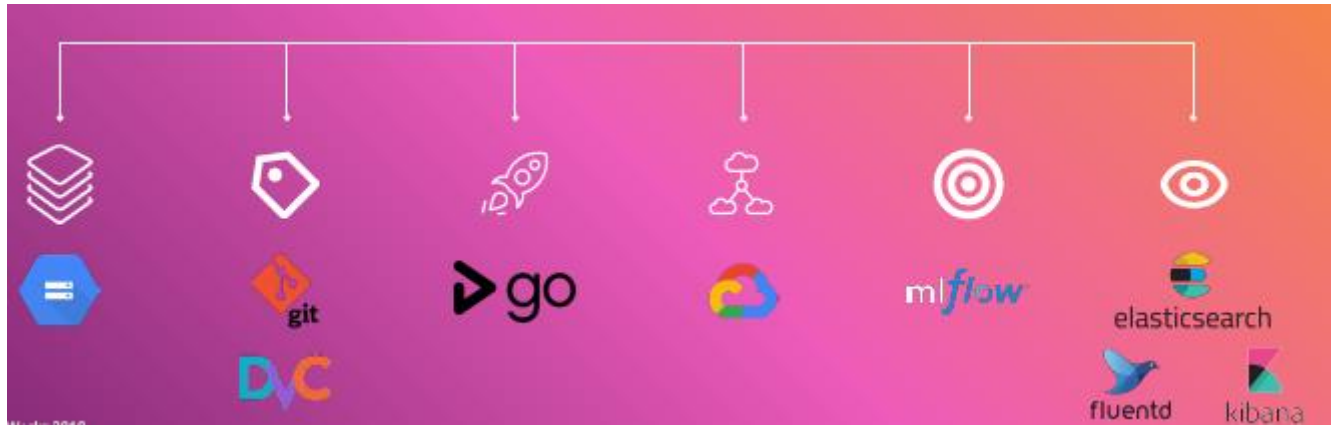
Pipeline



- Réf: Thoughtworks 2019 Strata data conference London 2019

Outils (2019)

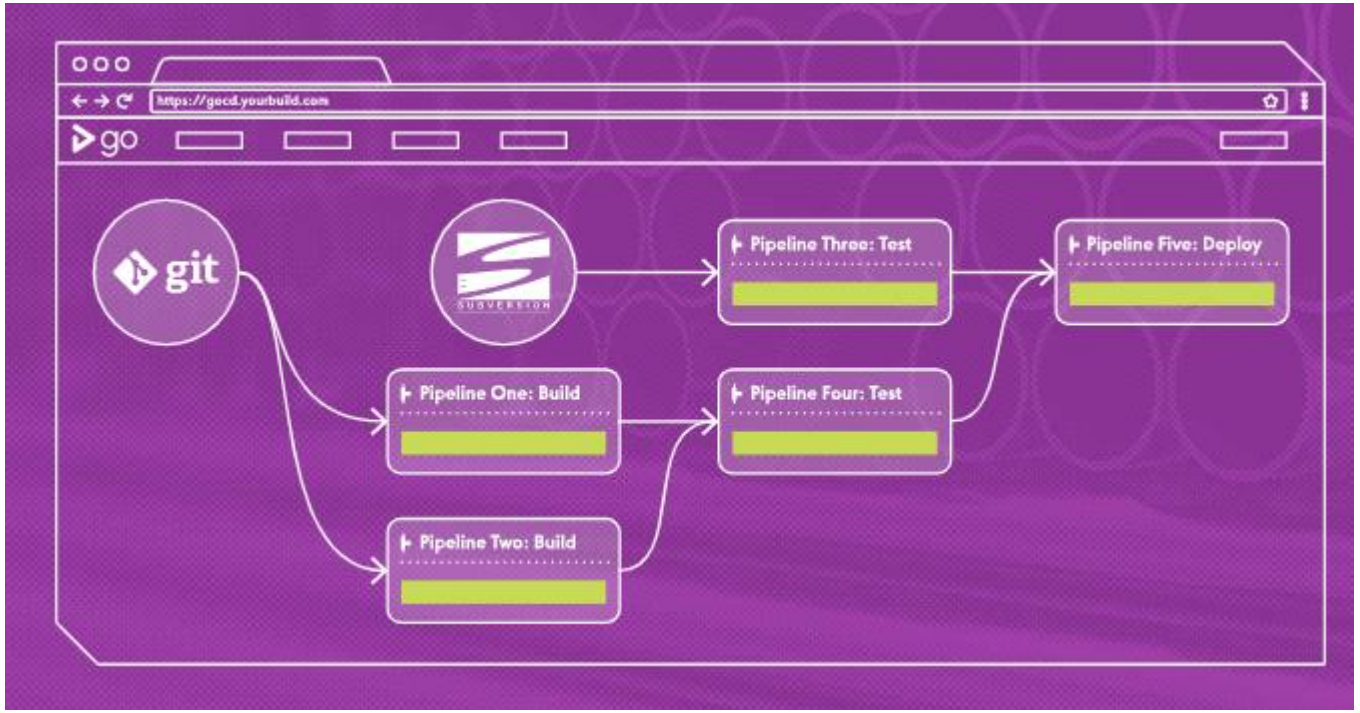




- Réf: Thoughtworks 2019 Strata data conference London 2019

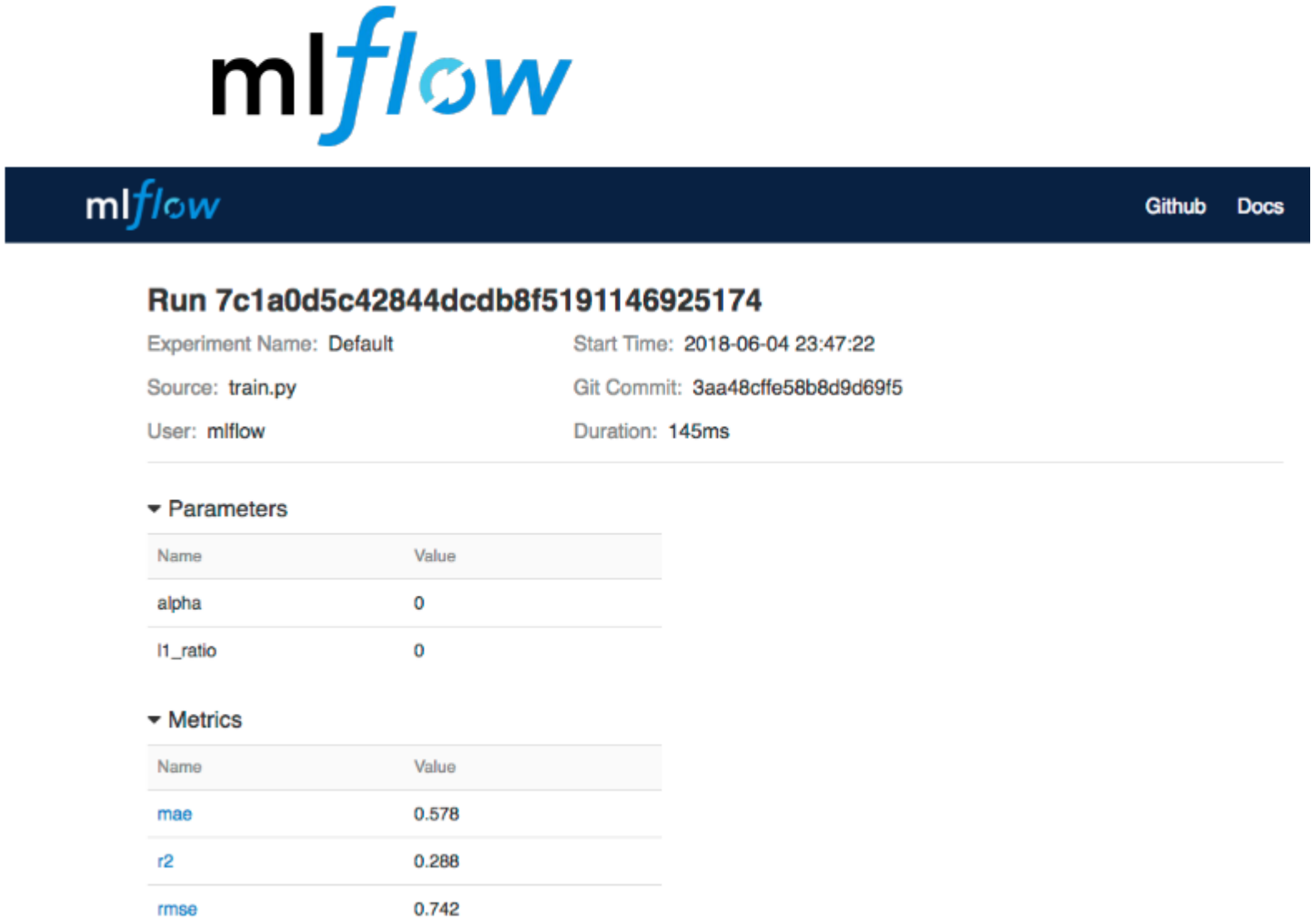
Outils | go

- *An Open Source Continuous Delivery server to model and visualise complex workflows*



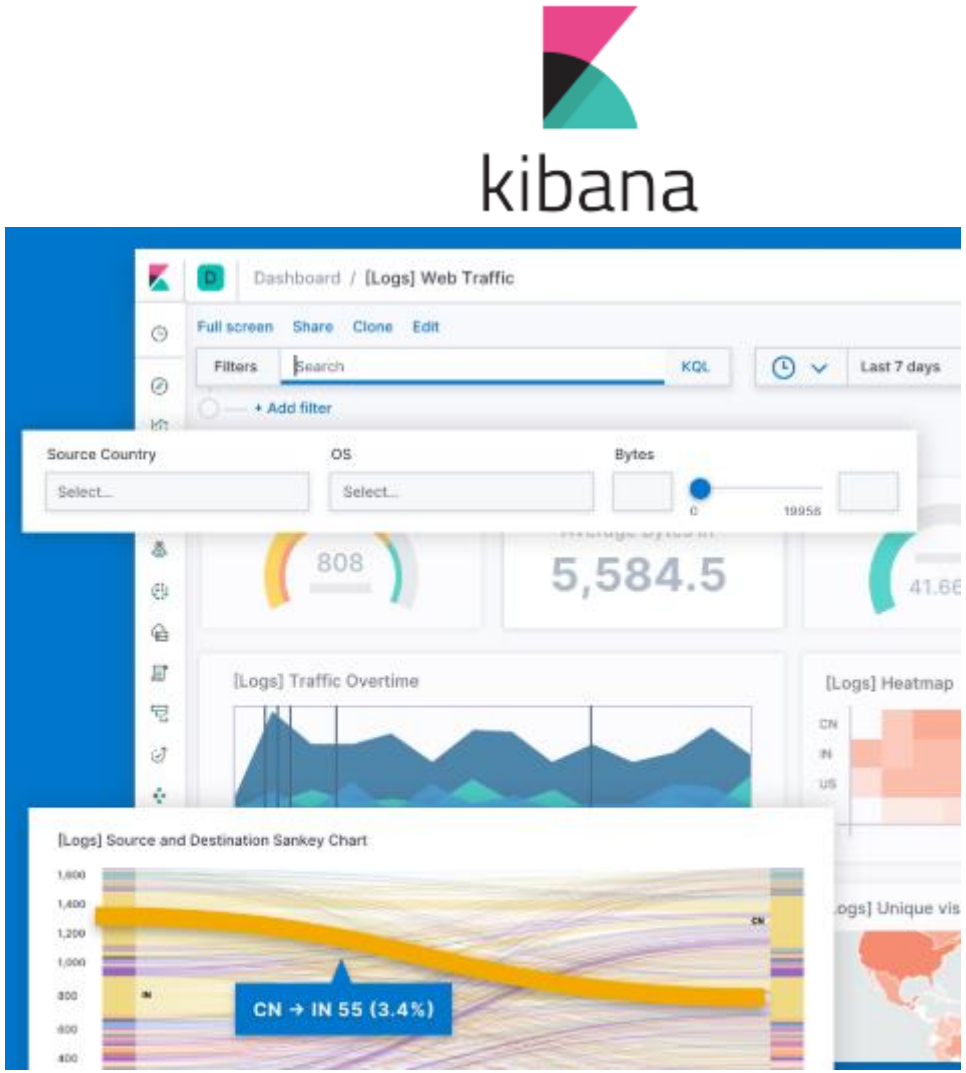
Outil | mlflow

- *An Open Source platform for managing end-to-end machine learning lifecycle*
- *Tutoriel:* https://mlflow.org/docs/latest/tutorial.html



Outil | Kibana

- Open Source web UI to **explore and visualise data sur Elasticsearch**



Challenges

Mise en production | Après !

- Comment procéder à un nouveau apprentissage le plus fréquemment possible?
- Comment éviter des problèmes de clash de version de déploiement?
- Comment redéployer le modèle mis à jour?
- Comment être sur que le processus de développement est toujours d'actualité?
- Comment mesure la performance de notre modèle?

Suivi du développement

- Il ne suffit pas de déployer, il faut aussi un suivi du processus de développement
- Quelles sont les hypotheses qui sont en train d'etre explorées?
- Quelles techniques de pré-traitement a t-on essayées?
- Combien de temps prend chaque processus pour s'executer?
- Quels sont les parametres et hyperparametres utilisées?
- Quels sont les métriques utilisés?
- Etc.

Amélioration continue

- Une fois en production, on doit capturer des métriques de production afin d'améliorer nos modèles
  - **Suivre comment le modèle est utilisé**
  - **Voir quel type de données on utilise sur le modèle**
  - **Évaluer la sortie du modèle pour voir son écart par rapport à la normale**
  - **Essayer de détecter s'il y'a un overfit**
  - **Essayer de voir si le modèle n'a pas de biais**
  - **Etc.**

Références

- https://stacktoheap.com/blog/2018/11/19/mlflow-model-repository-ci-cd/
- https://cdn.oreillystatic.com/en/assets/1/event/292/Continuous%20intelligence_%20Moving%20machine%20learning%20into%20production%20reliably%20Presentation.pdf
- http://blog.innodatalabs.com/automating-ml-training-with-jenkins-pipelines/
- https://thegurus.tech/posts/2019/06/mlflow-production-setup/