Université IBM i

7 novembre 2023

IBM Innovation Studio Paris

S17 – Les usages de Git sur IBM i

14:45 / 15:45

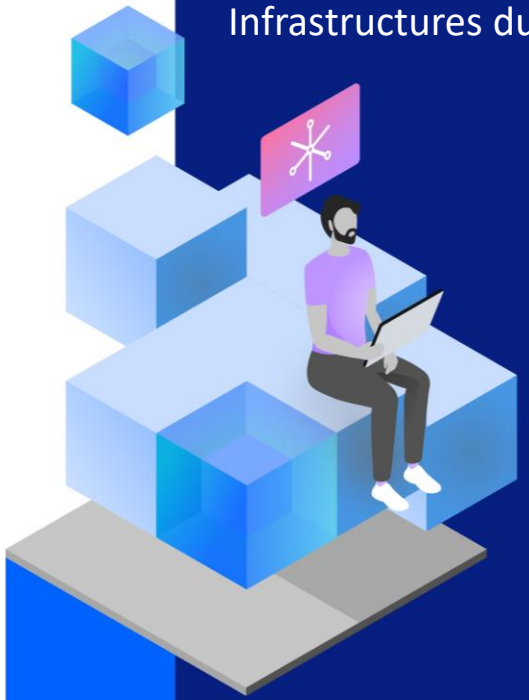
Gautier Dumas
Pierre Bec
CFD-Innovation
gdumas@cfd-innovation.fr
pbec@cfd-innovation.fr

infrasdufutur

#ibmi
#uii2023
#infrastructuredufuturIBM23

IBM

Infrastructures du futur



7 et 8 novembre 2023

Agenda

1. Présentation git
2. Git pour les développements Web & Open Source
3. Git pour les développements traditionnels
4. Conclusion

IBM

Infrastructures du futur

7 et 8 novembre 2023

Université IBM i

7 novembre 2023

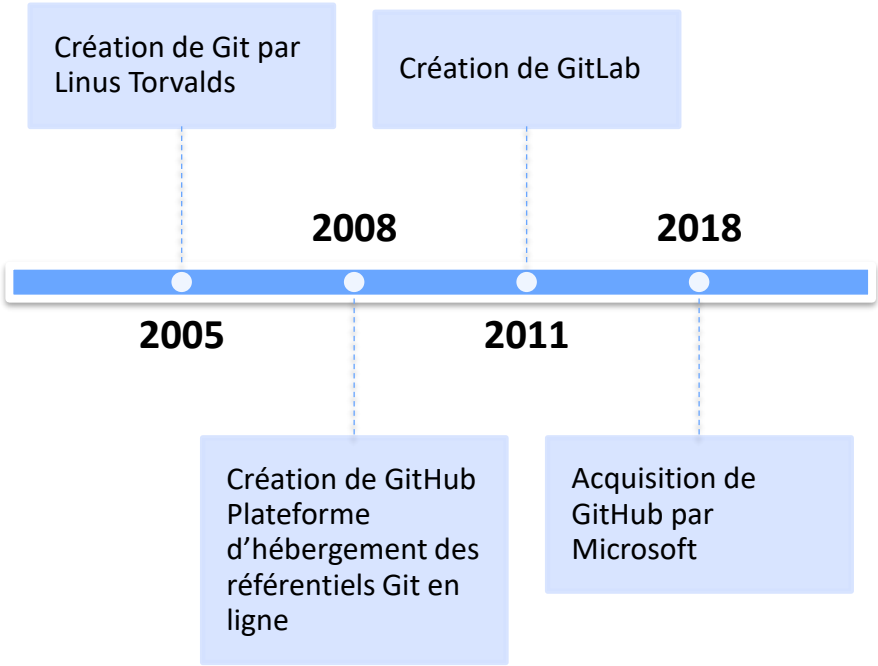
IBM i 35 YEARS

Let's Create

1. Présentation git

Git – Présentation

- Logiciel Open Source de gestion de versions
- Popularisé dans les années 2010
- Devenu un standard dans le versioning de codes sources (tout langage confondu)
- Repose sur un modèle distribué / Pair-à-pair

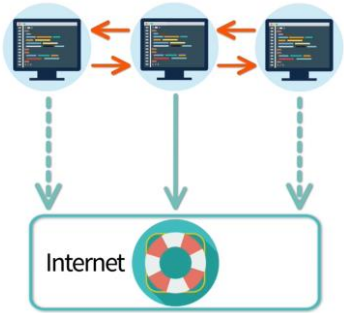


Pourquoi versionner son code ?

- Pour gérer les versions du code au fur et à mesure des modifications
- Pour faciliter les rollbacks
- Pour garder un historique des modifications
  - L'auteur
  - La date et l'heure
  - L'explication
  - Les fichiers modifiés et leurs modifications
- Pour faciliter le travail en équipe et paralléliser les développements
- Pour répondre à des exigences légales

Quelques concepts clés

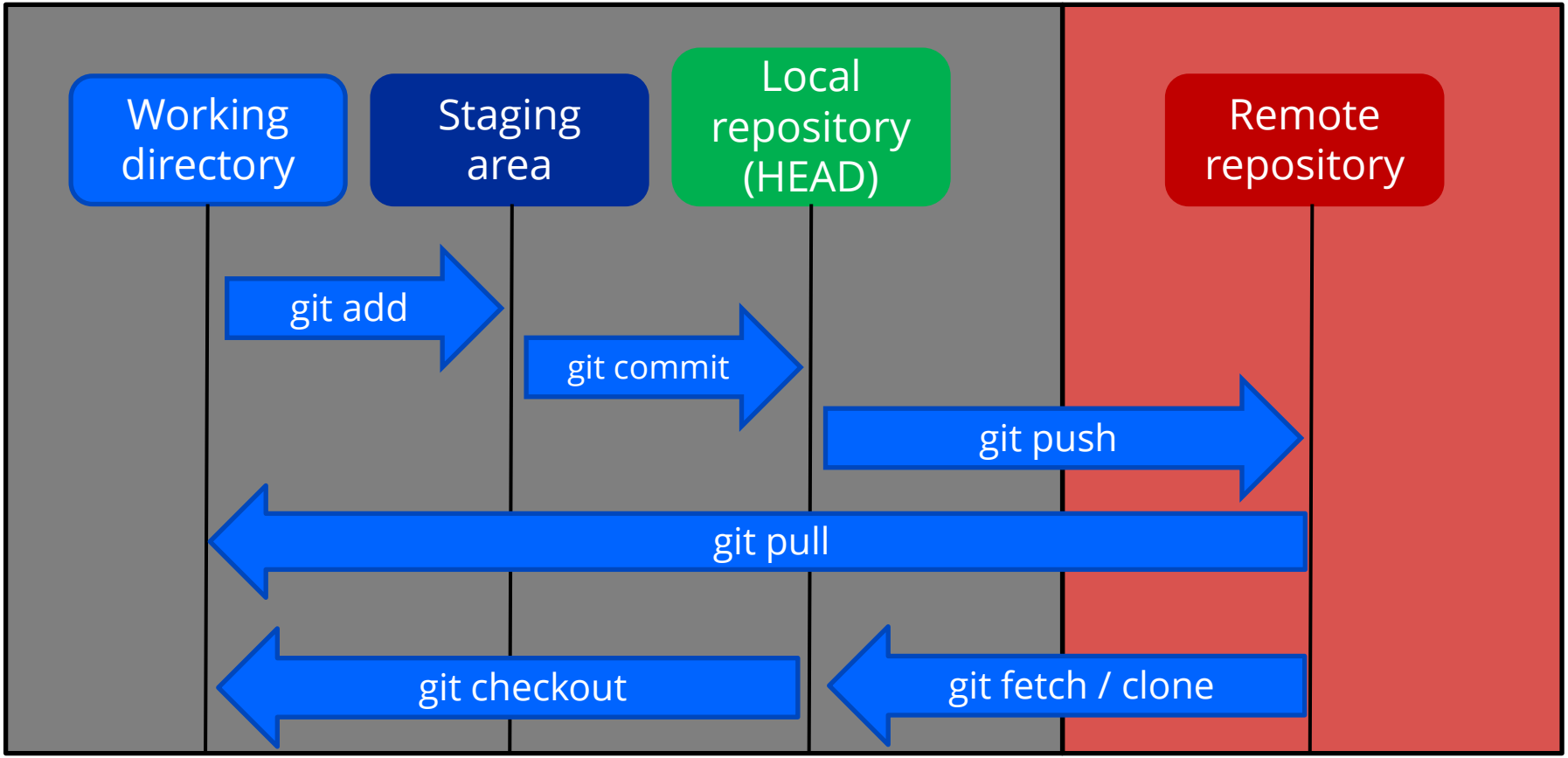
- **Un référentiel Git (repository)**
  - Contient la structure, l'historique et les fichiers d'un projet
- **les commits**
  - Modifications de fichiers (modifications/créations/suppression de lignes) = un commit
  - Un commit correspond donc à une version du projet à un instant T
  - Somme des commits = historique d'un projet
- **Remote Git** (GitHub / GitLab / BitBucket)
  - Centralisation des modifications de code d'un projet
  - Facilitation de la cohésion d'équipe (Chef de projet, Product Owner, Développeurs, …)



Remote Git

- **Github, Gitlab et BitBucket (Azure DevOps, GitBucket…)** sont des solutions de remote git, proposant :
  - des solutions SaaS dans le Cloud
  - hébergeant des dépôts GIT (privés et/ou publiques selon les solutions)
  - des interfaces web pour consulter les évolutions du projet
  - Des fonctionnalités de gestions de projets (Issues, CI/CD, Wiki, …)
- GitLab permet l'hébergement de la solution on-premise

Git Fonctionnement



Ce que Git n’est pas

- Git ne gère pas la compilation des sources et les déploiements
- Git ne comprend pas les dépendances IBM i
- Git ne comprend pas l’approche par librairie (environnement d’exécution)
- Git ne gère pas les objets

- D’autres outils sont dédiés à la gestion des déploiements
- Git est souvent la première brique dans l’adoption d’un CI/CD

Git sur IBM i

- Le futur du développement sur IBM i est un environnement multi-technologies où chaque développeur peut choisir la meilleure technologie pour répondre à un besoin
- Exemples :
  - Traitement et calcul => COBOL + RPG
  - Application web de gestion => PHP
  - Besoin d'une application temps réel => Node
  - Automatisation de tâches d'administration => Python
- L'utilisation d'un référentiel unique pour tous les codes sources permet
  - La coordination des changements
  - La centralisation de tous les changements de l'entreprise (IBM i ou autres)
- Protection du code source de l'IBM i

Git sur IBM i

- Pour les développements Web & Open Source
  - Typiquement pour les développements Java, PHP, Node, Python
  - Utilisation « classique » de git sur IBM i
- Pour les développements traditionnels (fichiers sources)
  - RPG, COBOL, CL …
  - Quelques étapes nécessaires

IBM

Infrastructures du futur

7 et 8 novembre 2023

Université IBM i

7 novembre 2023

IBM i 35 YEARS

Let's Create

2. Git pour les développements Web & Open Source

Git sur IBM i – Usage Web & Open Source

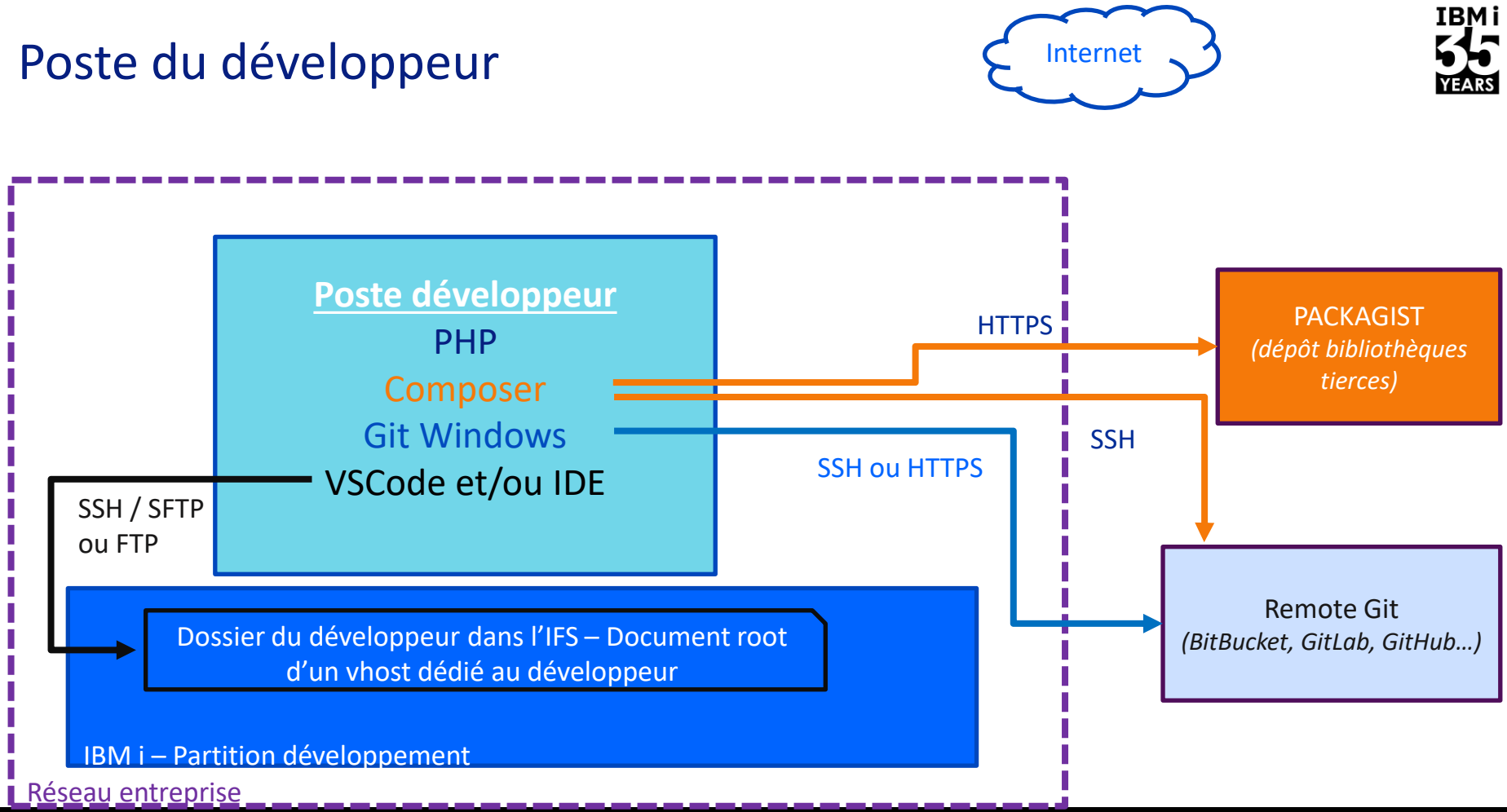
- Même philosophie et organisation de développement qu'avec un autre OS hôte comme Linux ou Windows
  - Respect des standards
  - Pas de spécifique IBM i qui est vu comme un nœud / une cible / un référentiel
  - Intégration de l'IBM i dans les processus CI/CD

- Git est disponible sur IBM i au format RPM
  - Installation facile et rapide avec YUM
    - yum install git
  - Dernières versions accessibles à partir de la V7R3
  - Exécutable disponible sous /QOpenSys/pkgs/bin/git
  - Interactions avec git au travers d'un shell PASE :
    - Connexion ssh (recommandée)
    - CALL QP2TERM

Etude de cas : Organisation développement PHP

- Prenons le cas d'une application web à base de technologies PHP/HTML/CSS/JS
- Deux partitions à disposition :
  - Partition développement
  - Partition production
- 4 développeurs web qui travaillent sur le même projet
  - Chaque développeur dispose
    - dans son environnement local Windows :
      - D'un IDE de développement (VSCode, PHPStorm …)
      - De git
      - De composer (gestionnaire de dépendances PHP)
    - sur l'IBM i de développement
      - D'un vhost Apache dédié

Poste du développeur



Partition développement

- La partition de développement accueille :
  - Un vhost et une version de l'application par développeur
  - Un vhost « develop », environnement permettant de tester les fusions des modifications des différents développeurs
- Aucun développement direct dans l'environnement « develop »
  - Déploiement de code via git sur IBM i (au travers d'un SSH par exemple)
    - Premier déploiement :
      - git clone
      - git checkout develop
      - composer install
    - Déploiements suivants :
      - git pull
      - composer install

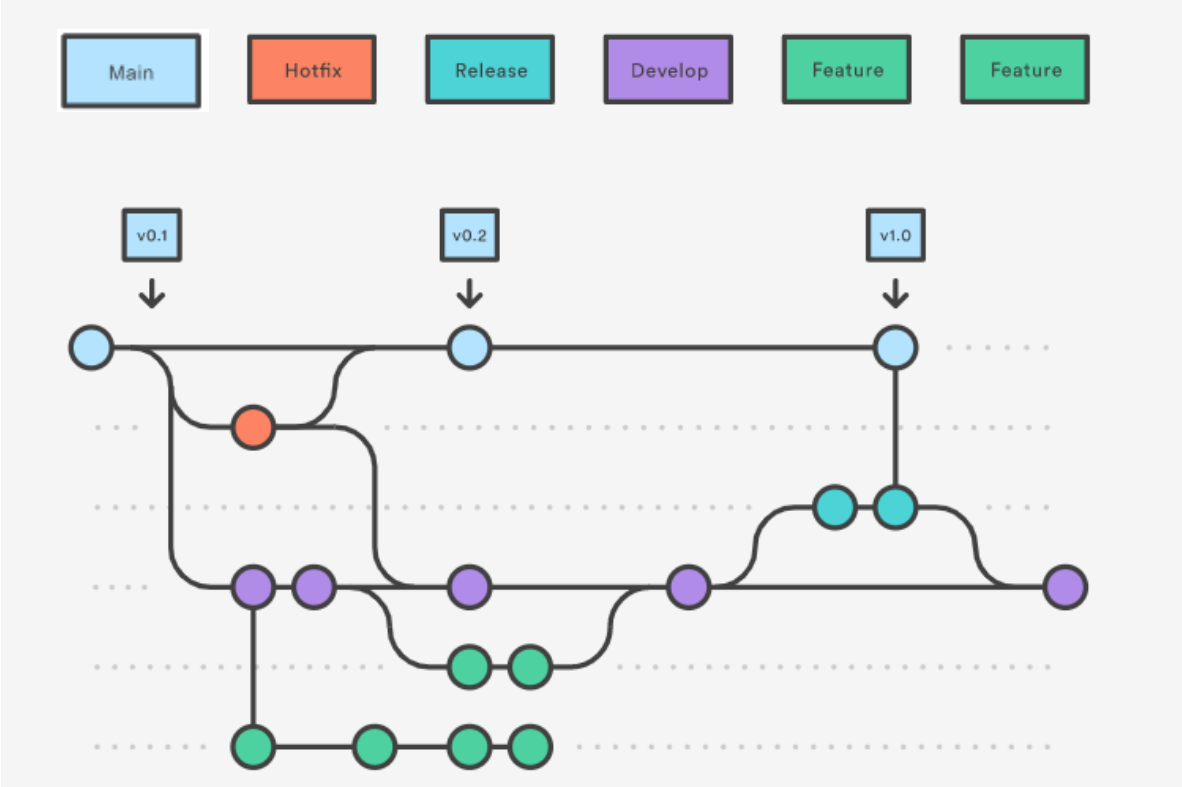
Partition production

- La partition de production accueille :
  - Un vhost « main », environnement correspondant aux dernières versions de l'application testées et livrées en production
- Aucun développement direct dans l'environnement « production »
  - Déploiement de code via git sur IBM i (au travers d'un SSH par exemple)
    - Premier déploiement :
      - git clone
      - composer install
    - Déploiements suivants :
      - git pull
      - composer install

Les branches git

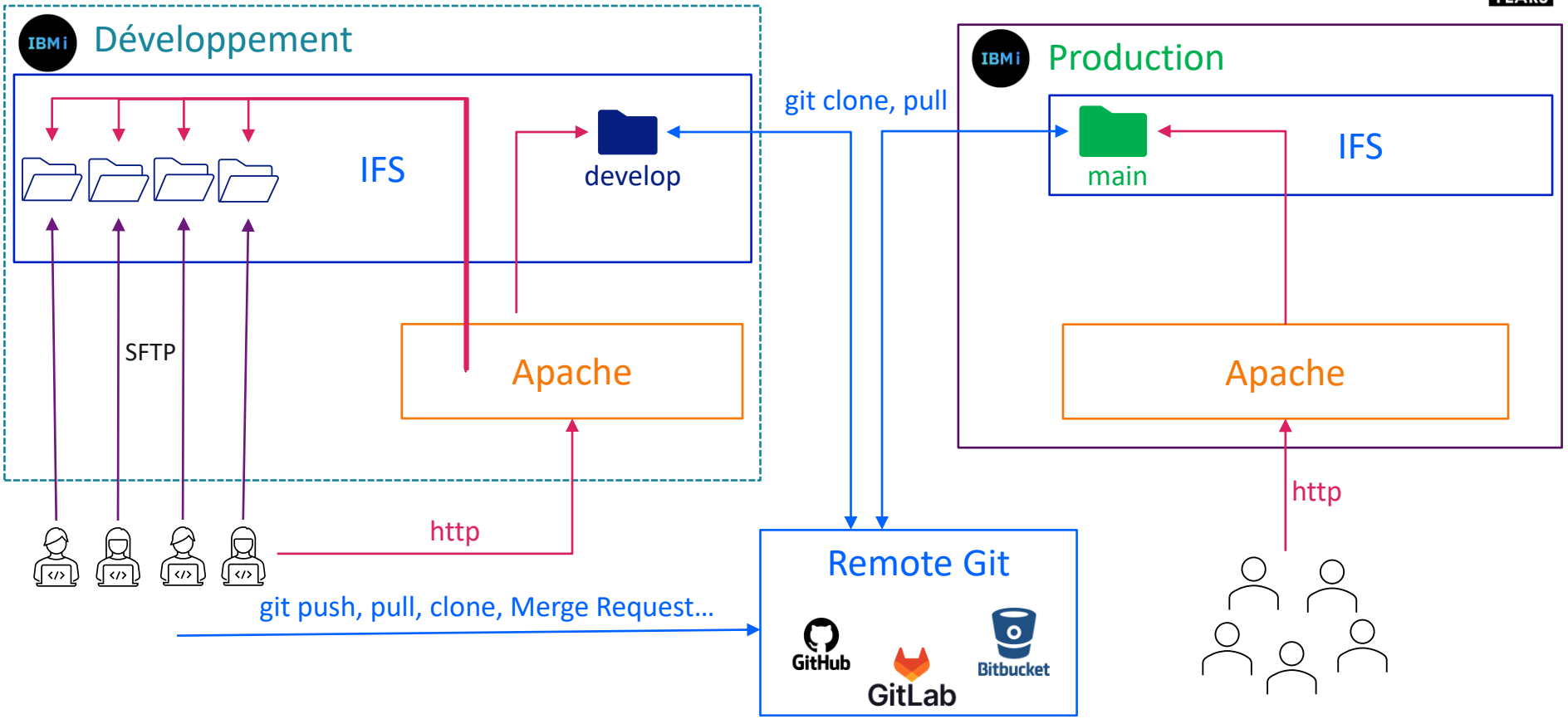
- Le principe des branches git est puissant
- Il existe beaucoup de workflows de développements possibles
- Dans notre étude de cas, nous utiliserons le workflow par branches features
- Il va permettre :
  - De représenter l'état du code de notre application en fonction des environnements stables définis :
    - develop
    - main
  - De permettre le travail en parallèle des différents développeurs sans risque d'écrasement (avec un principe de fusion du code)

Les branches git – workflow par branches features



Source : https://www.atlassian.com/fr/git/tutorials/comparing-workflows/gitflow-workflow

Récapitulatif étude de cas Web & Open Source



Pour aller plus loin

- L'architecture et l'organisation sont complètement personnalisables pour répondre aux besoins de l'entreprise, entre règles strictes et souplesse dans les process
- Exemples :
  - Ajout d'un environnement de recette et/ou de pre-production
  - Utiliser des branches releases vs utiliser des branches stables pour les environnements
  - Déploiements manuels / automatisés
  - Ajout de traitements spécifiques lors des livraisons (gestion des droits, modification de configurations, restart serveur web …)

IBM

Infrastructures du futur

7 et 8 novembre 2023

Université IBM i

7 novembre 2023

Démo git

Pour les développements Web & Open Source

IBM i 35 YEARS

Let's Create

git

GitLab

IBM

Infrastructures du futur

7 et 8 novembre 2023

Université IBM i

7 novembre 2023

IBM i 35 YEARS

Let's Create

3. Git pour les développements traditionnels

Waterfall to Agile (Parallélisme)

- Le passage à git induit le passage
  - d'un mode waterfall
    - Un développeur verrouille un objet/ un source pour commencer à travailler
  - à un mode Agile
    - Travail en parallèle possible sur un même source

- Git fait partie d'une culture du développement logiciel en mode Agile
- L'adoption d'un outil seul ne change pas la culture de l'entreprise

- On parle de développement
  - Pessimiste (Waterfall) -> On évite les conflits en bloquant
  - Optimiste (Agile) -> On règle les conflits quand ils arrivent

Organisation versioning git IBM i

- Schématiquement,
  - passage de 1 bibliothèque de sources avec fichiers et membres sources
  - à un stockage des sources par développeur dans un système de fichiers (IFS ou poste du développeur)
- Très souvent :
  - 1 bibliothèque => 1 dossier parent dans l'IFS
  - 1 fichier source => 1 sous-dossier dans l'IFS
    Exemples : QCLSRC ; QRPGLESRC ; …
  - 1 membre => 1 streamfile source .CLLE / .SQLRPGLE …

Quelques solutions

- Package git sur IBM i
  - Solution basique et Open Source directement sur IBM i pour versionner le code
  - Intégration de Git dans les process natif de développement IBM i
- VSCode et Code for IBM i
  - Solution de développement Open Source permettant de gérer le versioning et le code à partir du poste développeur
  - Possibilité d'intégrer le déploiement / compilation
- Plugin Rdi eGit
  - Solution intégrée à Rdi en mode projet
- Solutions commerciales
  - Gitit - Gaïa
  - iForGit – Richard Schoen
  - iGit - Eradani

Commencer à versionner

- Toutes les instructions git (add, commit, push, pull, clone …) sont utilisables sur toutes sources (PHP, Java, RPG …) contenues dans l'IFS

- Le point de départ est donc d'avoir les sources dans l'IFS

- La commande de base pour passer d'un membre source à un fichier dans l'IFS (attention au CCSID) :
  - **CPYTOSTMF FROMMBR('/QSYS.LIB/MYLIB.LIB/QCLSRC.FILE/QCSRC.MBR') TOSTMF('/home/GDUMAS/MYLIB/QCLSRC/qcsrc.c')**
  - https://www.ibm.com/docs/en/i/7.4?topic=ifs-copying-source-files-into

- Pour industrialiser les copies de fichiers sources vers l'IFS :
  - https://blog.faq400.com/en/system-administration-en/export-source-code-to-ifs/

Workflow possible

Au démarrage:

1. Convertir sources dans dossiers IFS
2. Initialiser un repo git dans le dossier contenant les sources

Lors des développements:

1. Ajouter / Modifier des sources
2. Versionner les modifications (branch, add, commit, push … selon le workflow de développement git adopté)
3. Compiler :
   1. Soit depuis l'IFS (nouveau référentiel de vos sources)
   2. Soit depuis le membre source (en faisant l'opération inverse : IFS -> MBR)

```
CPYFRMSTMF FROMSTMF('/ifs_path/to/source/myPgm.sqlrpgle')
TOMBR('/QSYS.lib/PROD.lib/QRPGLESRC.file/MYPGM.mbr') MBROPT(*REPLACE)
```

IBM

Infrastructures du futur

7 et 8 novembre 2023

Université IBM i

7 novembre 2023

IBM i 35 YEARS

Let's Create

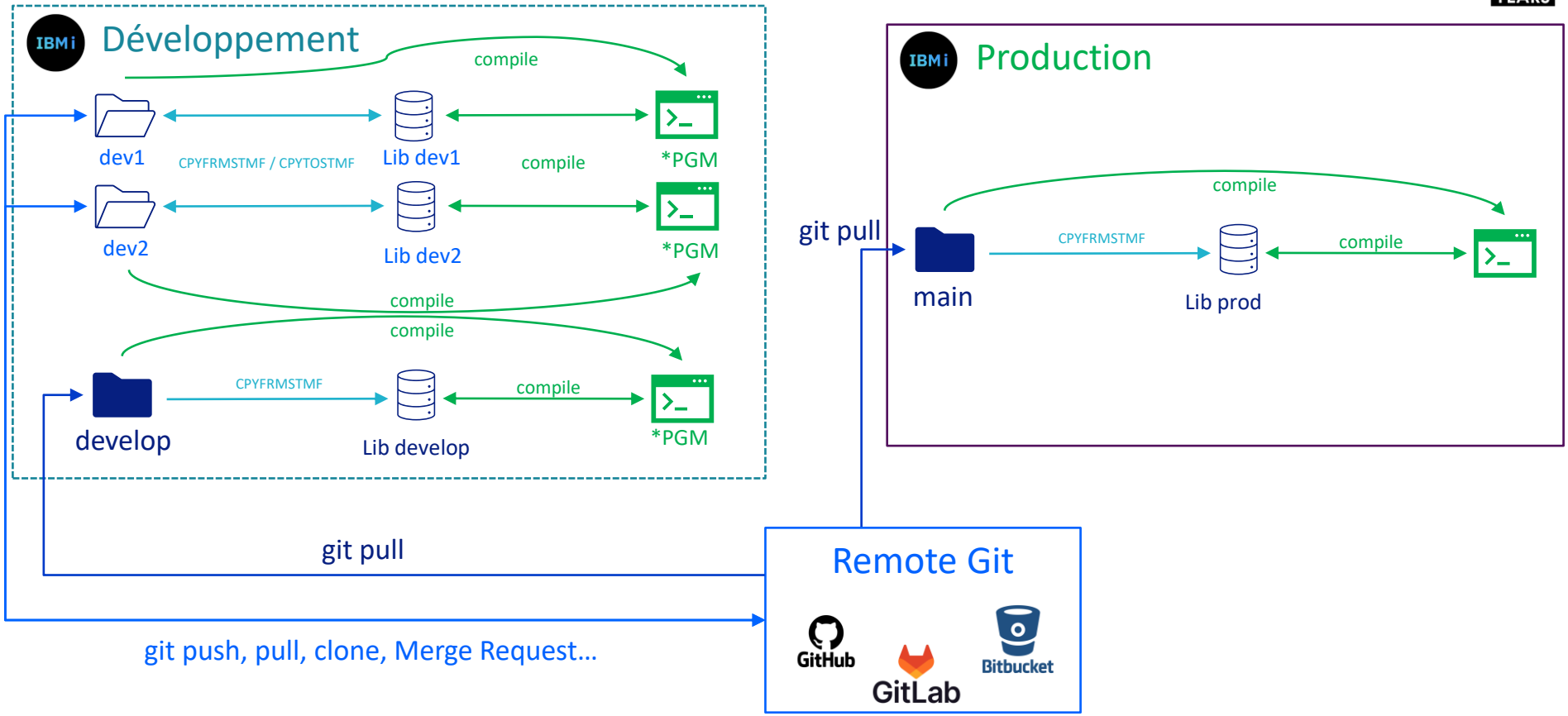
Démo git

Pour les développements traditionnels

Exemple Workflow avec branches

- Plusieurs organisations possibles (un peu comme avec les bibliothèques)
  - Avec branche
    - 1 dossier IFS par développeur (copie des sources) + 1 dossier par environnement stable
    - Versioning local (IFS) avec une branche par feature
    - Développement dans le dossier du développeur et dans la branche concernée
    - Compilation dans l'environnement du développeur pour premier niveau de test
    - Une fois le développement terminé et validé, MERGE REQUEST vers une branche stable de l'organisation (DEVELOP, RECETTE, TEST, PROD …)
    - Déploiement du code de la branche stable vers le dossier correspondant (git pull suivi des traitements de compilation / livraison)
  - Arrivée d'un nouveau développeur :
    - git clone dans son espace IFS
    - Début développement en suivant le workflow

Récapitulatif sources IBM i



IBM

Infrastructures du futur

7 et 8 novembre 2023

Université IBM i

7 novembre 2023

IBM i 35 YEARS

Let's Create

4. Conclusion

Conclusion

- Git est un outil puissant pour le versionning de source
  - Pas pour compiler / déployer !
  - Souvent une fondation nécessaire pour la mise en œuvre de CI/CD
- Un bon moyen pour commencer est de l'utiliser dans les développements Web & Open Source
  - Utilisation standard
  - S'approprier les concepts
  - Bien connus des développeurs Web & Open Source
  - Déjà bien intégré dans des processus CI/CD
- Git est intégré pour pouvoir versionner les sources « natives »
  - Une organisation à étudier et mettre en place selon vos processus de développement
  - Des outils existent pour industrialiser le versioning de l'existant et faciliter le quotidien des développeurs

35