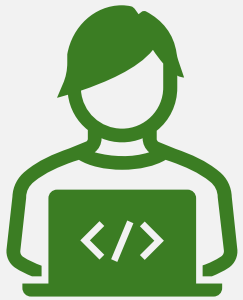
Projet Tutoré & DevOps

Hiba Najjar

S7, GInf4

UPF Université Privée de Fès
الجامعة الخاصة لفاس
Private University of Fez

DevOps, pourquoi?



Développement

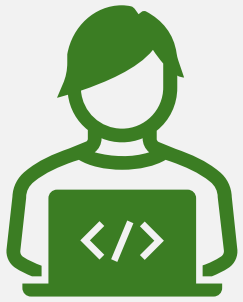
- Innover et livrer
- Ajout de nouvelles fonctionnalités
- Rythme rapide

Production

- Assurer la stabilité et disponibilité
- Réduction des incidents
- Rythme lent, contrôlé

Produit livré aux clients

DevOps, pourquoi?



Développement **Production**

DevOps: Automatisation des processus
(Provisionnement, Tests, Déploiement)

Produit mis en service aux clients

DevOps, est-ce pratique?

Etudes;

- Mise en place Pipeline CI/CD via les toolschains du groupe.
- Agile/Organiser car plusieurs projets à gérer en même temps.
- Gestion des espaces Devops avec les CI/CD, GI
  IBM CLOUD (PaaS/IaaS), Kubernetes (pods, dép
  CHART HELM, Docker, HVAULT (pour les gestio
  authentification).
- Notion databases en mode asservice (PostgreS

Phase de Mise en Production :

- en place la supervision et l'ordonnanceme
  on, Shell, powershell si possible
  cipation à la rédaction des livrables de mis
  installation/Dossier Exploitation);

Phase de support N3 / Maintenance

Support aux équipes APS en cas d'incident sur les applications.

- Maintenance des environnements Cloud liés à l'OPS Realestate. (Gest
  correction des vulnérabilités sur les images socles, VSI,etc...
- Expertise sur l'automatisation des tâches récurentes avec ANSIBLE.

Profil candidat:

Je recherche un **Ingénieur DevOps** en vue de l'
installation/migration/support/automatisation des applications au sein du Technology

- Exécuter les montées de version

Tu te reconnais dans ce descriptif ? Le poste est peut-

Petits plus : tu seras garant(e) de **l'amélioration contin**
de **conseiller/conseillère** auprès d'autres équipes IT q
pimenter ton quoti

Tu as 5 à 10 ans d'e
production (Grafan

Si tu es à l'aise avec
sacré avantage !

En bref : Si tu as en
équipe dont les ma
en pleine construct

Les missions sont les suivantes :

- Analyse des besoins et c
- Conception et développ
- Développement logiciel
  Embarqué.
- Rédaction protocole de
- Mise en production et m

Qualifications
Diplômé(e) d'une école d'ingéni
informatique.

Vous possédez une première ex
connaissez les langages C/C++.

Idéalement, vous avez des connaissances sur IDE Visual Studio Code, des architectures multiprocessus / threads, les communications TCP/IP, des bus SPI / I2C / série (UART).
Connaissances GIT, Jira appréciées
Anglais courant obligatoire.

Votre Terrain de Jeu

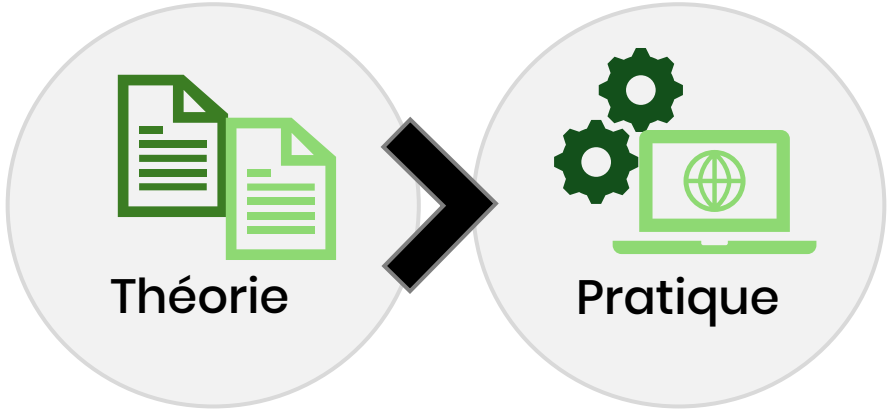
💻 Connaissances techniques :

• Parfaite maîtrise du langage Apex, la connaissance de Lightning ou Visualforce est un plus
• Certifications Salesforce fortement recommandées
• Une connaissance des environnements techniques microsoft serait un plus (C#, .Net Core, Azure DevOps, Azure Function/Web application/API, SQL Server/Azure SQL Database)
• Connaissances et expérience avérée sur les concepts CI/CD et testing
• Bonne connaissance des méthodes agiles
• Rigueur dans les projets et forte capacité d'analyse

Au sein du PPP, vous intégrez une équipe de développeurs juniors et confirmés, et apportez votre expertise technique. Vos missions principales seront les suivantes :

• Participer aux évolutions de la plateforme SalesForce (analyse d'impacts, recommandations, maquettes, POC, ...) en lien avec les streams projet
• Assurer la conception technique d'applications, au développement de la solution SFDC, à la gestion des étapes de tests, recettes et déploiement des applications.
• Réaliser de nouveaux développements en garantissant leur qualité et leur scalabilité.
• Réaliser la documentation de conception technique
• Etre force de proposition technique pour la mise en place de nouveaux modèles d'intégration et de nouvelles fonctionnalités personnalisées.
• Mettre en oeuvre des actions identifiées dans la roadmap technique (sonar, tests automatisés, déploiement auto, ...)
• Pratiquer de l'analyse de code et rétroengineering sur les développements spécifiques
• Identifier les évolutions des différentes release SF à mettre en oeuvre pour les besoins de

DevOps & Projet Tutoré

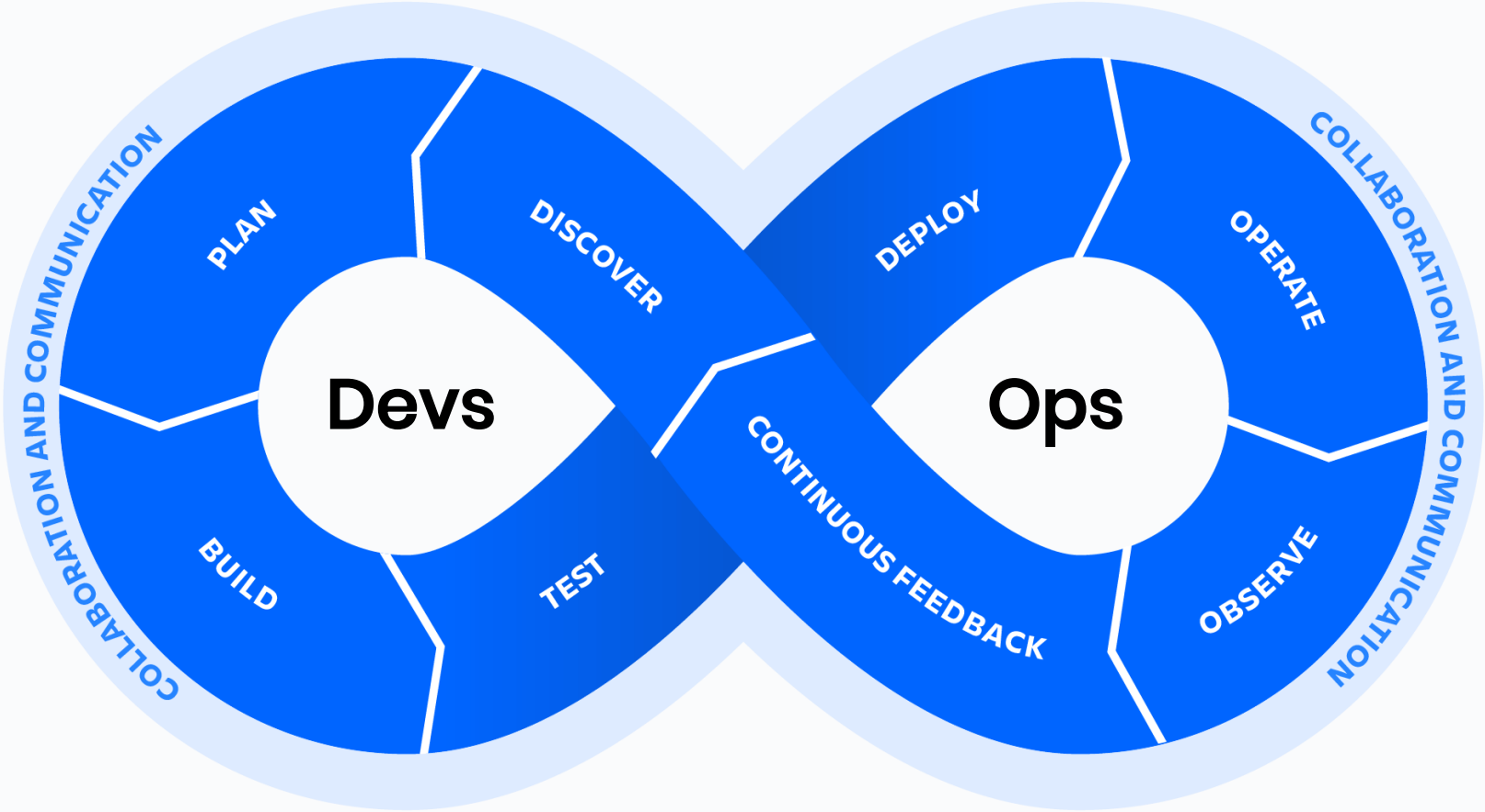


- Compétition durant la **Semaine d'Ingénieur**
- Travail approfondi durant le 2ème semestre
- Travail en groupe de 5/6
- Choix des sujets & Formation des groupes:
  - Choix final: 8 Décembre

Les rôles

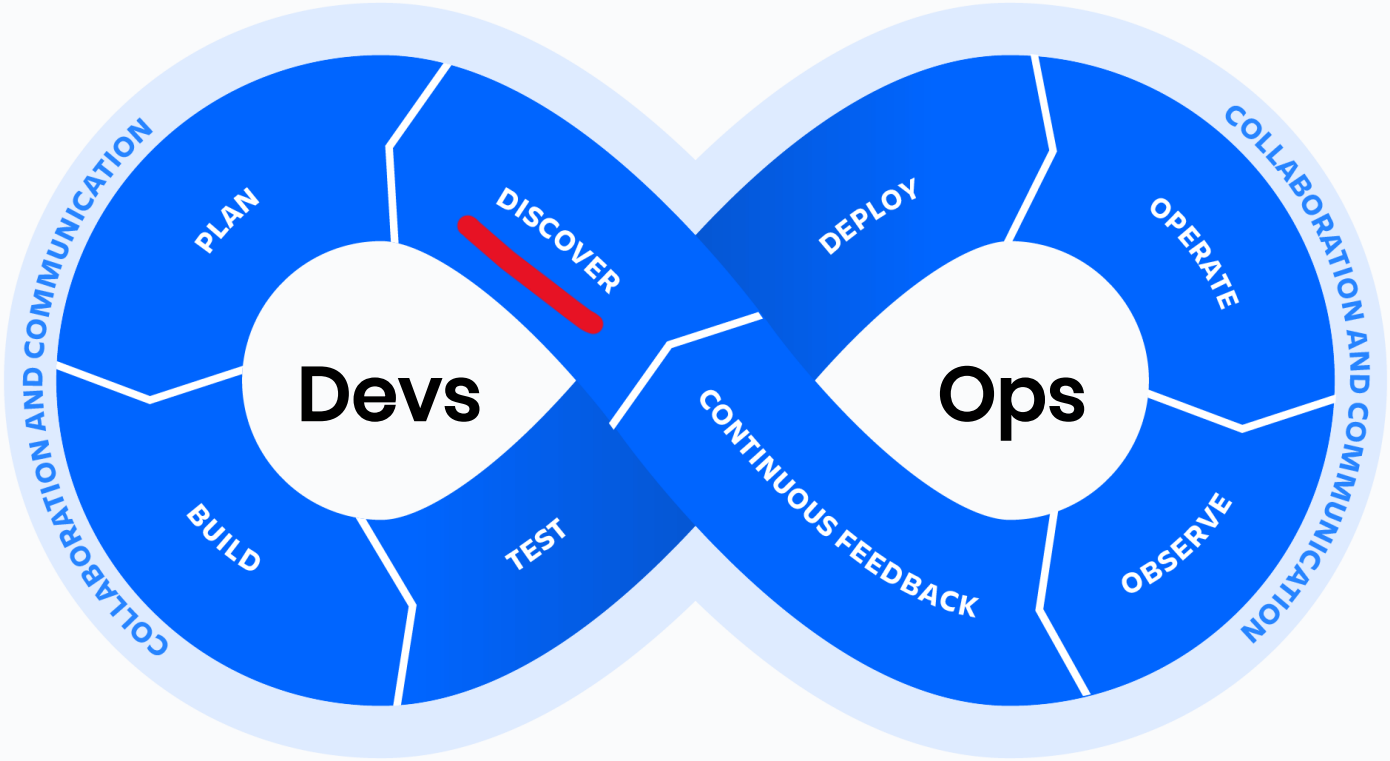
- Coordinateur
- Développeurs
- Automatisation, CI/CD
- Infrastructure
- Testeur
- Documentation et rapports
- Présentations

DevOps: concepts fondamentaux



Source: Atlassian

DevOps: concepts fondamentaux



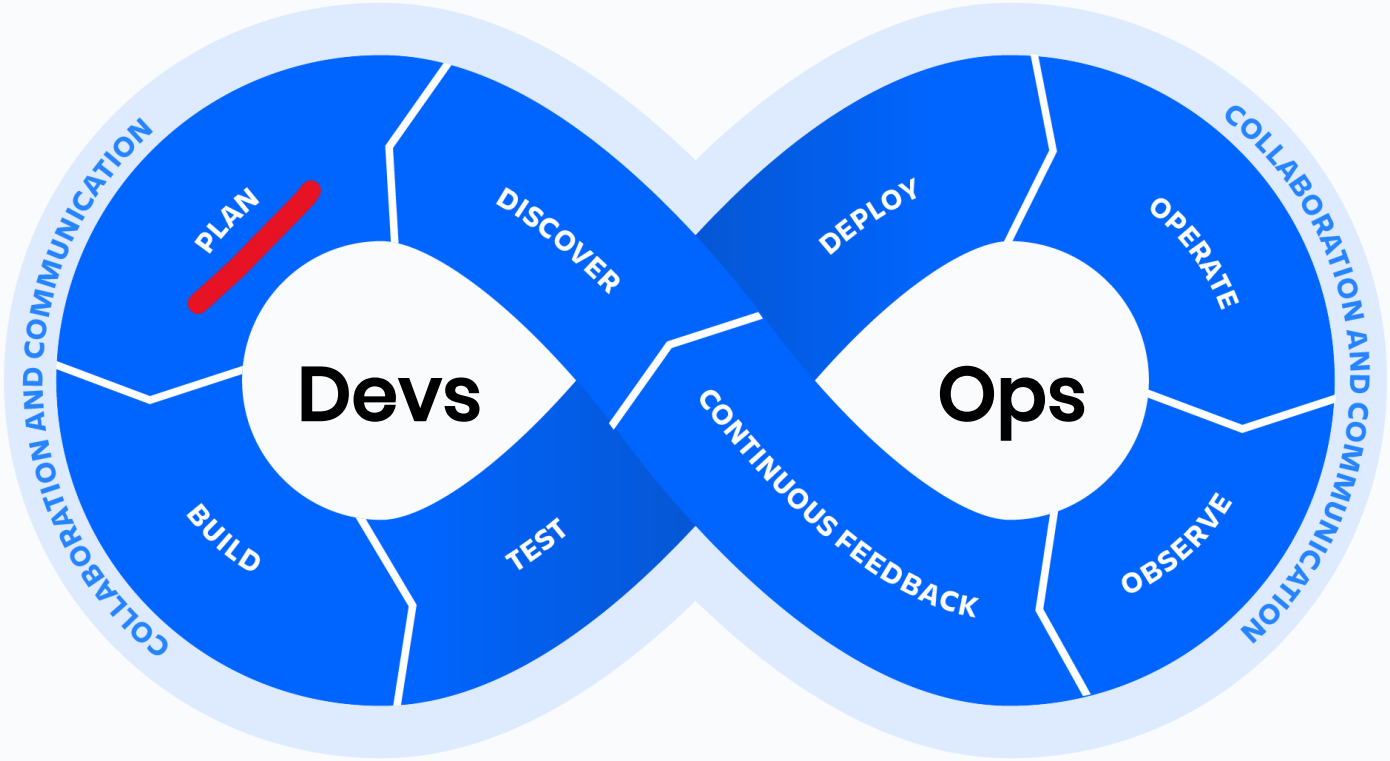
Découvrir

- En équipe,
- Explorer de nouvelles idées,
- Respecter le cadre des objectifs stratégiques préétablis,
- Organiser et hiérarchiser.

Agile

Source: Atlassian

DevOps: concepts fondamentaux



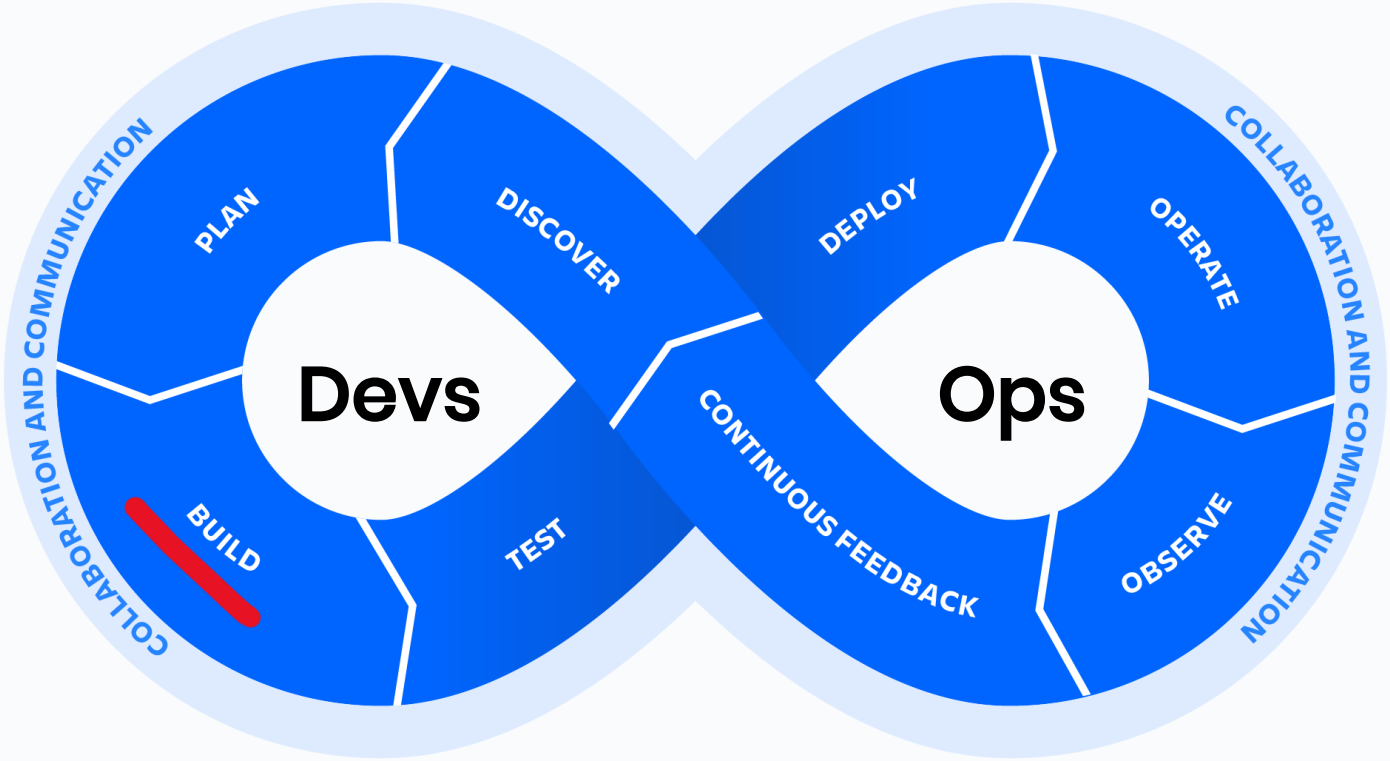
Plannifier

- Organiser les tâches,
- Le diviser en étapes incrémentales,
- Optimiser la vitesse & la qualité.

Agile

Source: Atlassian

DevOps: concepts fondamentaux



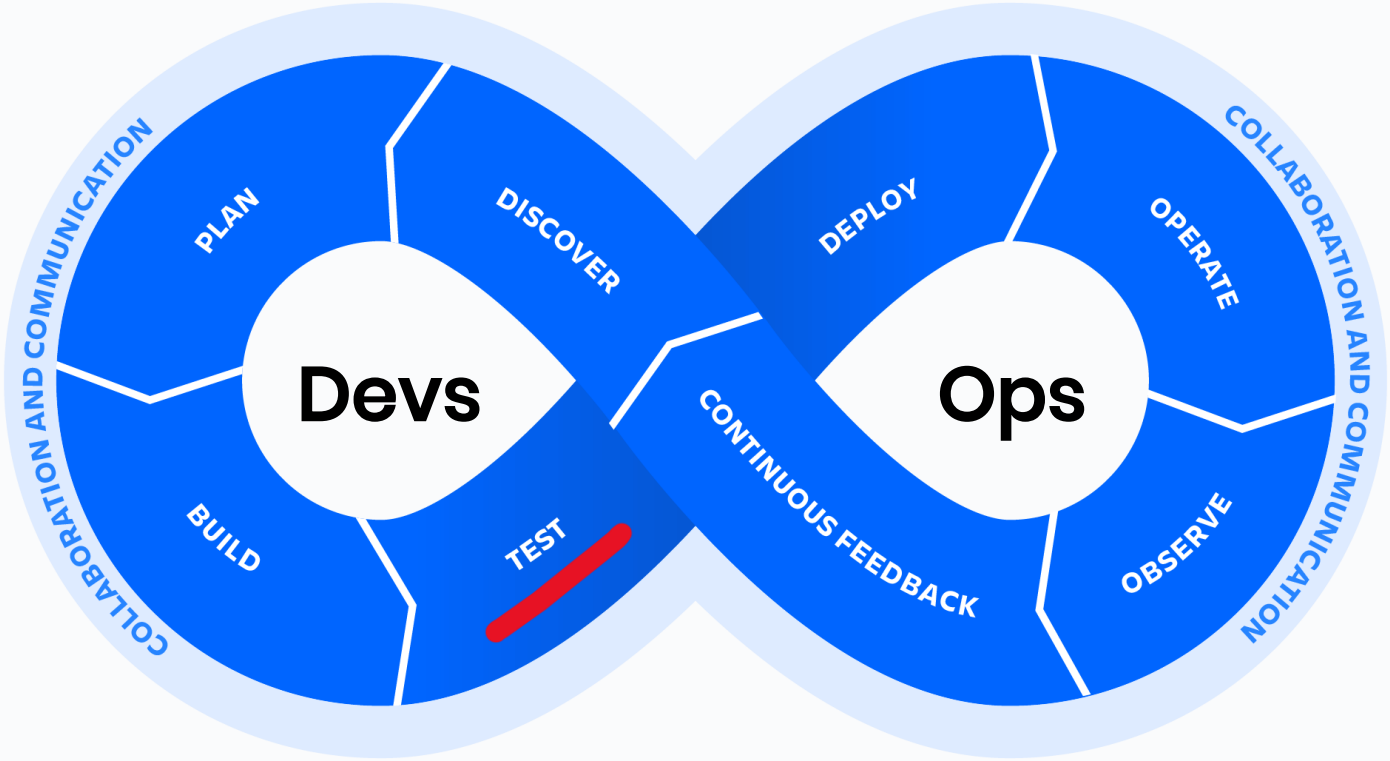
Build

- Optimiser le processus de développement,
- Contrôler les versions, les branches, et l'historique de toute activité.

Git

Source: Atlassian

DevOps: concepts fondamentaux



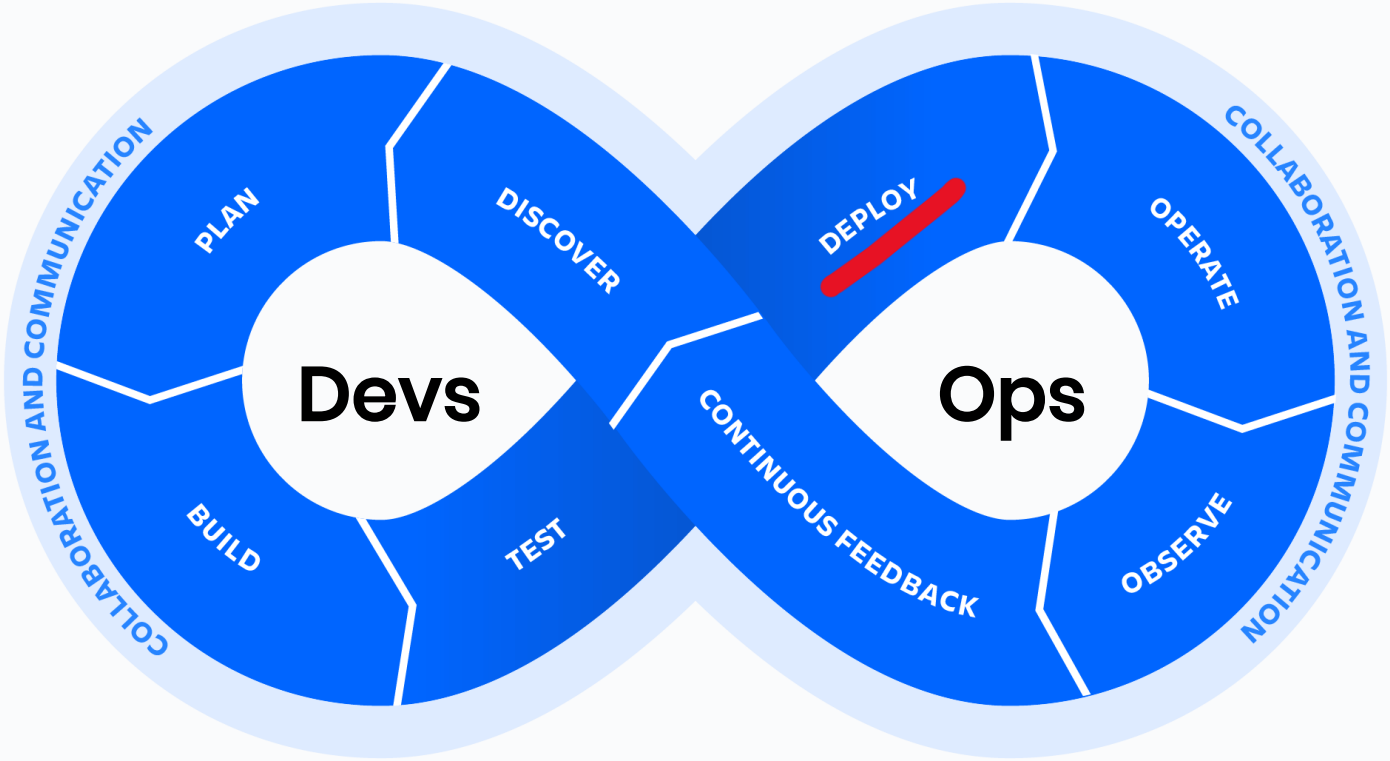
Tester

- Collaboration collective dans un dépôt partagé,
- Vérification et validation automatique des changements,
- Assurer la qualité du code en continu.

Intégration Continue (CI)

Source: Atlassian

DevOps: concepts fondamentaux



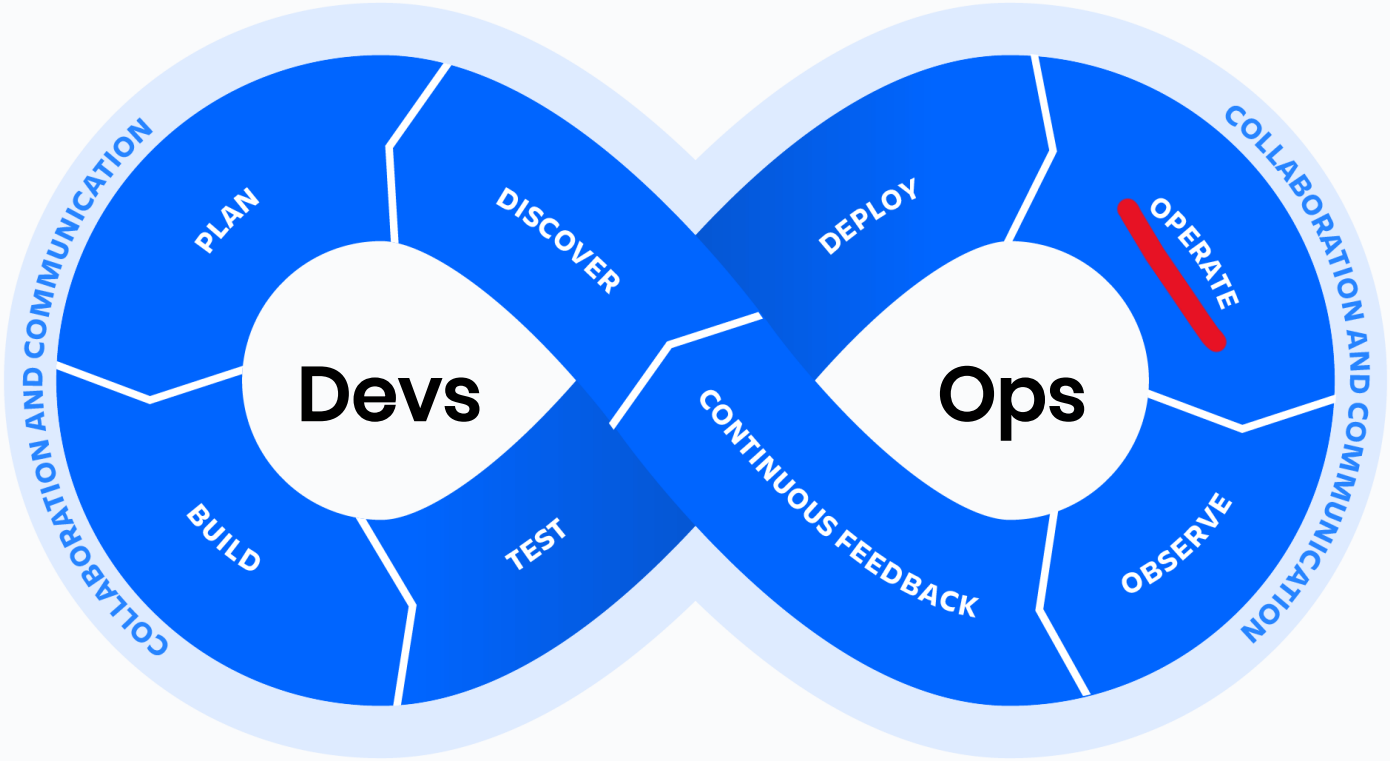
Déploiement

- Faciliter le déploiement des nouvelles fonctionnalité en production,
- Automatiser ce processus pour augmenter sa fréquence,

Déploiement Continue (CD)

Source: Atlassian

DevOps: concepts fondamentaux



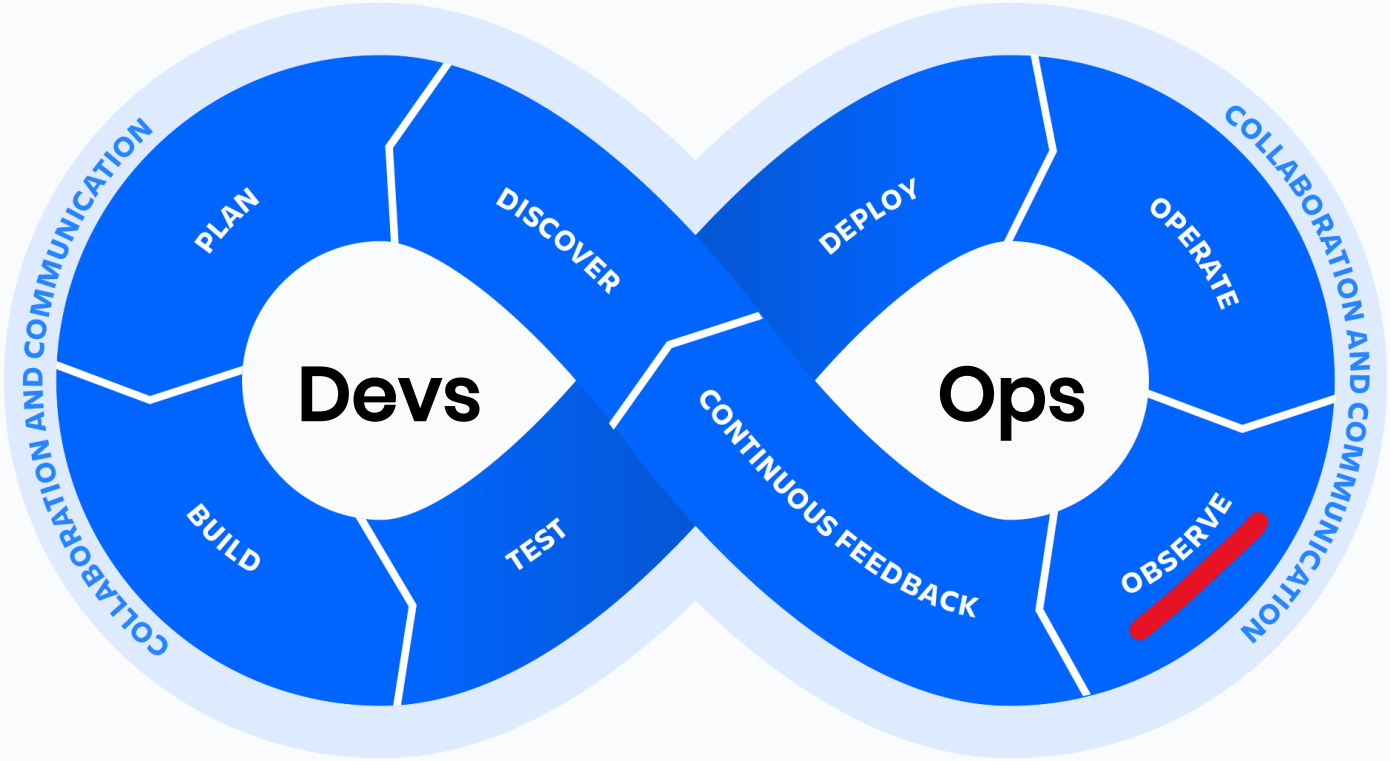
Opération

- Optimiser l'infrastructure informatique des services,
- Faciliter de bout en bout la livraison des services aux clients.

Infrastructure programmable (IaC)

Source: Atlassian

DevOps: concepts fondamentaux



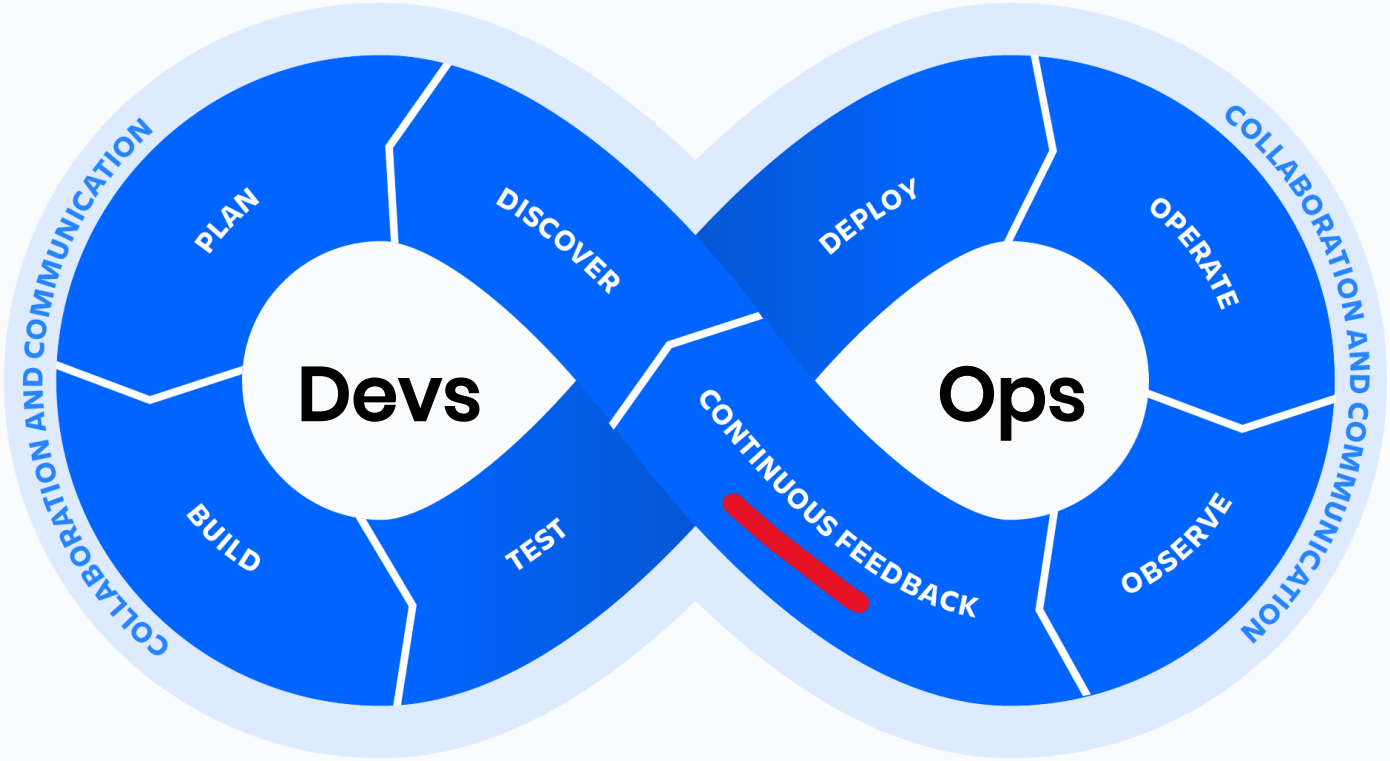
Observation

- Surveiller les applications et l'infrastructure (logs, métriques, alertes),
- Mesurer ce qui compte,
- détecter les problèmes en production

Monitoring / Observabilité

Source: Atlassian

DevOps: concepts fondamentaux



Feedback continu

- Collecter et intégrer le feedback des clients,
- Améliorer les livraisons futures,
- Planifier de nouvelles fonctionnalités.

Agile

Source: Atlassian

Gestion de versions avec Git

Git = système de contrôle de version open source gratuit.

À maitriser:

1. Concepts de base: repository, commit, branch, merge
2. Stratégies de branches: Gitflow/Trunk-based
3. Collaboration en équipe: pull-requests, code-review
4. Mise en place d'un dépôt partagé

Gestion de versions avec Git

› Concepts de base

Q1 — Qu’est-ce qu’un repository (dépôt) Git ?

A. Un fichier contenant le code
B. Un serveur distant
C. Un espace où l’historique et le code sont stockés
D. Un commit particulier

Q2 — Qu’est-ce qu’un commit ?

A. Une sauvegarde locale du code
B. Une copie complète du projet
C. Une modification enregistrée dans l’historique
D. Une branche à distance

Gestion de versions avec Git

› Concepts de base

Q3 — Quelle commande crée un dépôt Git ?

A. git add
B. git init
C. git commit
D. git clone

Q4 — À quoi sert une branche ?

A. À supprimer les commits
B. À travailler indépendamment du code principal
C. À renommer un repository
D. À corriger automatiquement les bugs

Gestion de versions avec Git

› Concepts de base

Q5 — La branche par défaut d'un dépôt Git moderne est :

A. main
B. master
C. default
D. origin

Q6 — Que fait la commande `git merge` ?

A. Crée une nouvelle branche
B. Combine l'historique de deux branches
C. Écrase tout le code existant
D. Supprime une branche

Gestion de versions avec Git

› Concepts de base

Q7 — Un conflit de merge apparaît lorsque :

A. Deux commits ont les mêmes messages

B. Deux branches modifient la même partie d'un fichier

C. Une branche est vide

D. Une branche est trop ancienne

Q8 — Quelle commande montre l'historique des commits ?

A. git history

B. git show

C. git log

D. git commit --list

Gestion de versions avec Git

› Concepts de base

Q9 — Que fait `git add` ?
A. Ajoute un nouveau commit
B. Ajoute des fichiers à l'index
C. Ajoute une nouvelle branche
D. Ajoute un nouveau dépôt

Q10 — Quand faut-il faire un commit ?
A. À chaque ligne de code
B. Une fois par mois
C. À chaque étape logique terminée
D. Jamais

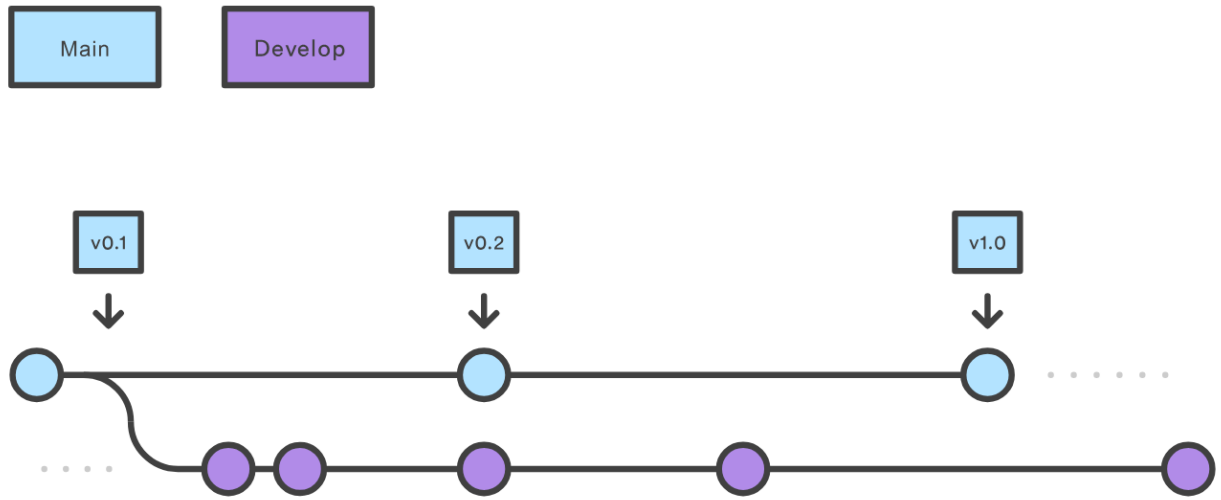
Gitflow Workflow

Gitflow est un modèle de branchement structuré, souvent utilisé dans des projets nécessitant:

- Structure claire pour les grandes équipes.
- Parfait pour des versions officielles (planifiées).
- Séparation forte entre les branches de développement, test et production.

Branches principales:

- Main, qui contient le code en production (versions stables).
- Develop, qui contient le code de la prochaine version. Cette branche sert à intégrer les fonctionnalités validées.

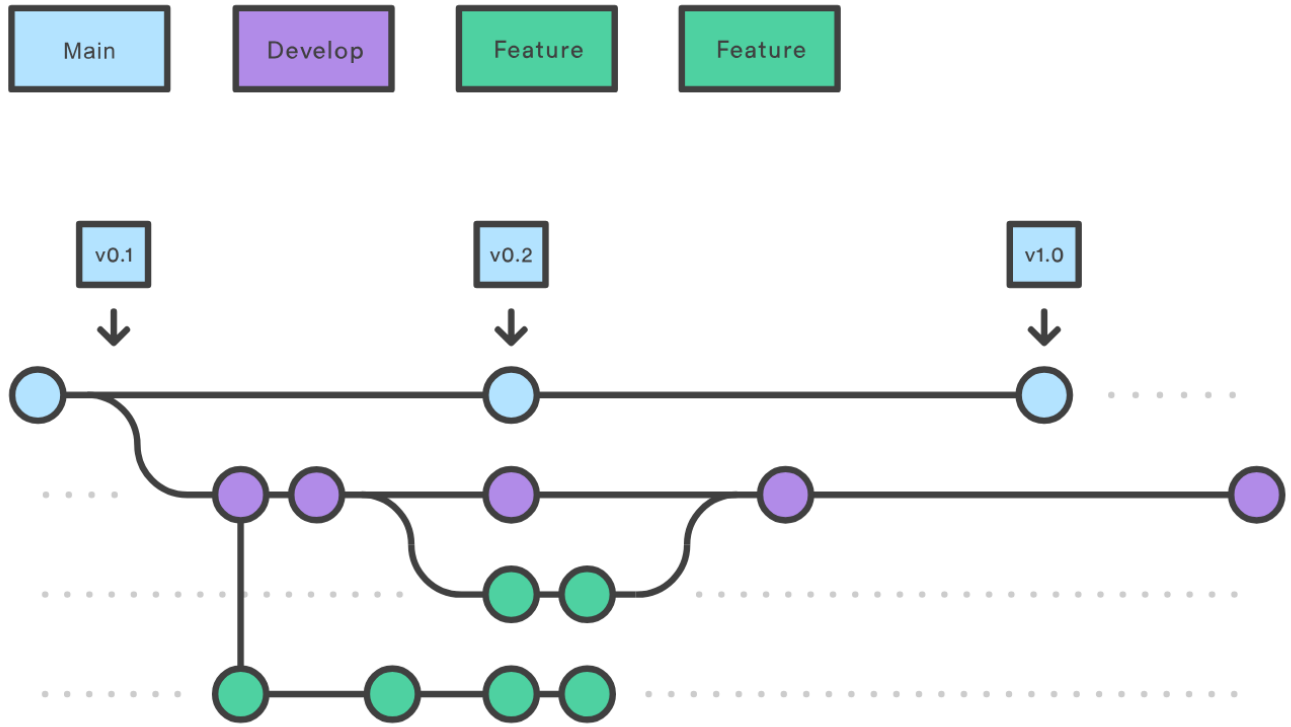


Source: Atlassian

Gitflow Workflow

Branches secondaires:

- **Feature**, contient une fonctionnalité par branche, part de develop et revient vers develop.

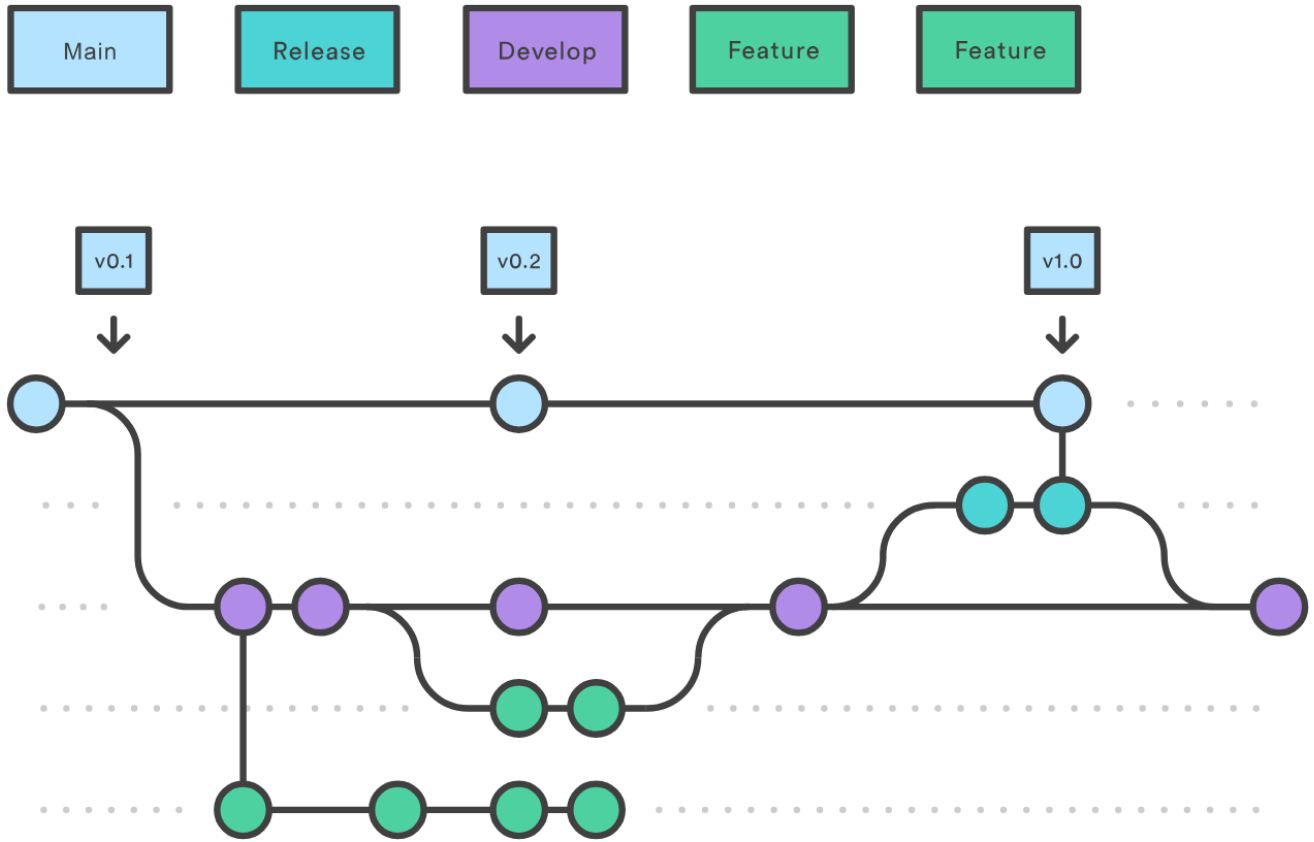


Source: Atlassian

Gitflow Workflow

Branches secondaires:

- **release**, est la branche de préparation à la publication d'une nouvelle version. La préparation inclut les tests et les corrections mineures. Elle part de develop et est ensuite fusionnée avec main et develop.

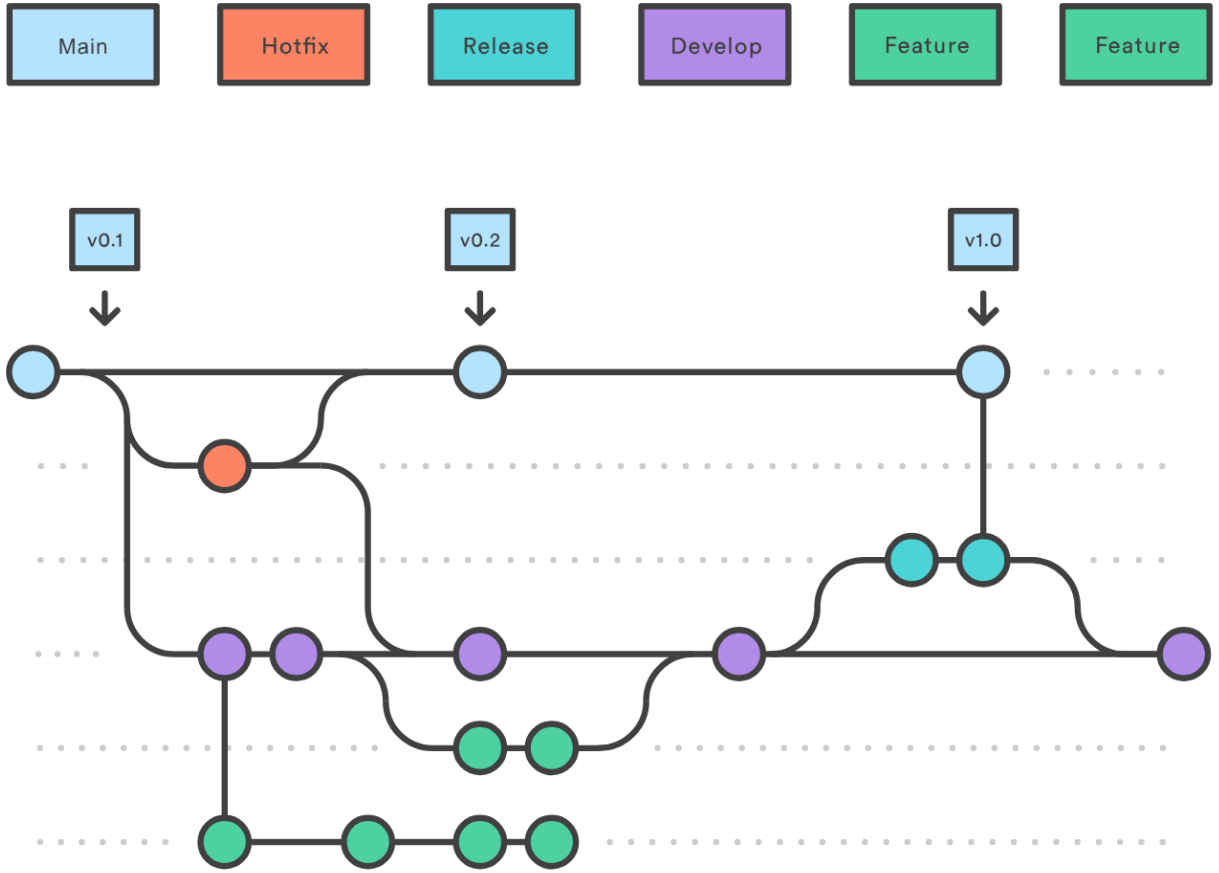


Source: Atlassian

Gitflow Workflow

Branches secondaires:

- **hotfix**, permet la correction en urgence d'une erreur dans la version de production, et part donc de main, et est ensuite fusionnée avec main et develop.



Source: Atlassian

Gitflow Workflow

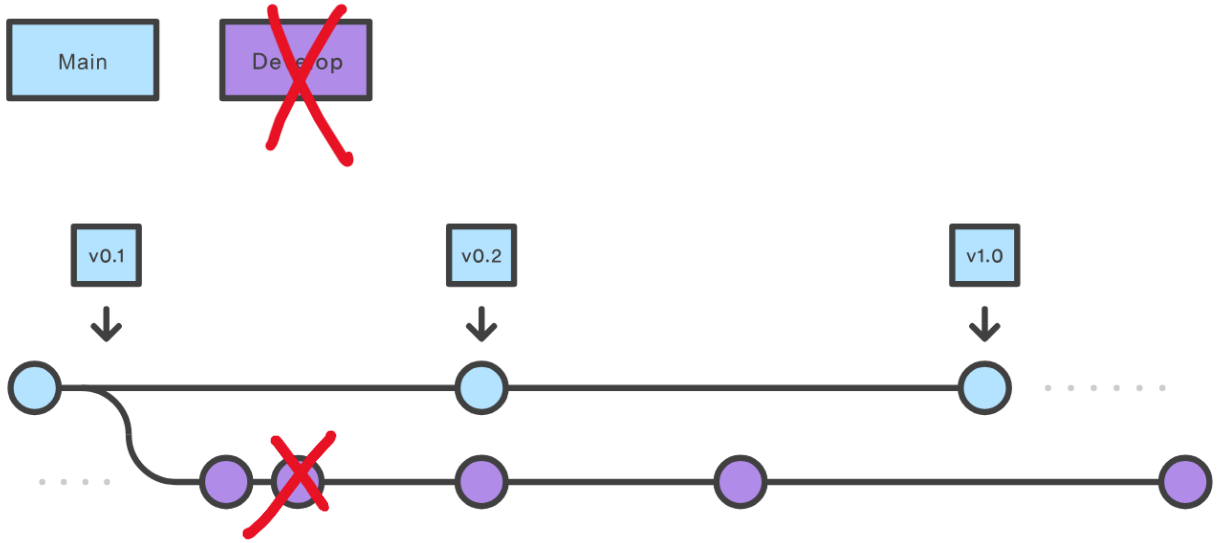
Le déroulement en pratique:

1. Une branche develop est créée à partir de la branche main.
2. Des branches de fonctionnalités feature sont créées à partir de la branche develop
3. Lorsqu'une fonctionnalité feature est terminée, elle est fusionnée dans la branche develop
4. Une branche de publication release est créée à partir de la branche develop.
5. Lorsque la branche release est terminée, elle est fusionnée dans les branches develop et main.
6. Si un problème est détecté dans la branche main et qu'il est rapide à résoudre, une branche hotfix est créée à partir de celle-ci.
7. Une fois le correctif dans hotfix terminé, il est fusionné dans les branches develop et main.

Trunk-based Workflow

Le développement **Trunk-based** est un modèle plus simple, souvent recommandé dans les pipelines DevOps, quand les nouvelles fonctionnalités doivent être publiées en continu.

- Le Trunk-based se forme d'une seule branche principale : main (appelée "trunk").
- Les développeurs créent de petites branches très courtes (quelques heures à quelques jours), rapidement fusionnées dans main. Dans ce cadre, les tests automatisés sont obligatoires.



Trunk-based Workflow

Les bonnes pratiques:

Développer en petits lots (small batches)

Effectuer de petits changements, faire des commits petits et fréquents, pour faciliter les révisions rapides.

Utiliser des Feature Flags

Un Feature Flag est un interrupteur logiciel qui permet d'activer ou désactiver une fonctionnalité sans changer de code.
Elle apparait sous la forme d'une condition dans le code, contrôlée par exemple par un fichier de configuration ou une variable d'environnement.

Trunk-based Workflow

Les bonnes pratiques:

Mettre en place des tests automatisés complets

Ils sont nécessaires pour assurer une intégration et déploiement continus (CI/CD). Ils sont exécutés lors de chaque commit/merge, permettant d'assurer la qualité du code et de rejeter automatiquement les commits qui cassent quelque chose.

Merger au trunk (main) au moins une fois par jour

Les branches prêtes sont fusionnées tous les jours, pour maintenir un rythme clair de livraison continue.

Gestion de versions avec Git

› Stratégies de branches : Gitflow/Trunk-based

Q11 — Gitflow utilise principalement quelles branches permanentes ?

A. main et bugfix
B. release et support
C. main et develop
D. hotfix et feature

Q12 — Quelle branche contient le code prêt pour la production dans Gitflow ?

A. develop
B. main
C. feature/*
D. hotfix/*

Gestion de versions avec Git

› Stratégies de branches: Gitflow/Trunk-based

Q13 — Dans Gitflow, où développe-t-on les nouvelles fonctionnalités ?

A. Directement dans `main`
B. Dans `develop`
C. Dans des branches `feature/*`
D. Dans `hotfix/*`

Q14 — Trunk-based development consiste à…

A. Ne pas utiliser de branches du tout
B. Faire des merges rapides et fréquents dans main
C. Ne jamais toucher à la branche principale
D. Avoir 10 branches permanentes

Gestion de versions avec Git

› Stratégies de branches: Gitflow/Trunk-based

Q15 — Dans Trunk-based, une branche feature dure généralement :
A. Plusieurs semaines
B. Plusieurs mois
C. Quelques heures ou jours
D. Elle n'existe pas

Q16 — Quel modèle est le plus adapté aux livraisons fréquentes ?
A. Gitflow
B. Trunk-based
C. Aucune idée

Gestion de versions avec Git

› Stratégies de branches: Gitflow/Trunk-based

Q17 — Gitflow est particulièrement adapté à :

A. Les projets à cycles de release lents
B. Les micro-services
C. Le travail sans version
D. Les équipes de 1 personne

Q18 — Une branche hotfix sert à :

A. Tester des fonctionnalités
B. Corriger un bug critique en production
C. Refactoriser le code
D. Créer une release

Gestion de versions avec Git

› Stratégies de branches: Gitflow/Trunk-based

Q19 — Dans trunk-based, comment éviter de casser la production ?

A. Ne jamais faire de commit

B. Feature flags

C. Faire des branches longues

D. Ne pas tester

Gestion de versions avec Git

› Collaboration : pull-requests, code-review

Q21 — À quoi sert une pull request ?

A. Demander l'avis des autres sur du code avant fusion (merge)
B. Sauvegarder son code localement
C. Ajouter un fichier
D. Supprimer une branche

Q22 — Une code-review permet :

A. D'assurer la qualité du code
B. D'augmenter la vitesse de merge sans vérification
C. De supprimer les tests
D. D'exécuter automatiquement le programme

Gestion de versions avec Git

› Collaboration : pull-requests, code-review

Q23 — Qui valide une pull request ?
A. L'auteur obligatoirement
B. Les membres du groupe (reviewers)
C. Personne
D. Le professeur seulement

Q24 — Pourquoi commenter du code dans une PR ?
A. Pour laisser un message au reviewer
B. Pour proposer des améliorations
C. Pour supprimer la PR
D. Pour créer une branche

Gestion de versions avec Git

› Collaboration : pull-requests, code-review

Q25 — Que signifie “approve” une PR ?
A. Refuser la PR
B. Accepter les changements
C. Ignorer la PR
D. Supprimer la branche source

Q26 — Quel outil est utilisé pour créer une PR ?
A. BitBucket
B. GitHub / GitLab
C. Jupyter Notebook
D. Jira / Confluence

Gestion de versions avec Git

› Collaboration : pull-requests, code-review

Q27 — Quel est l'avantage des PR ?
A. On fusionne (merge) sans vérifier
B. On collabore mieux et plus proprement
C. On ne laisse pas de trace (l'historique)
D. On supprime les bugs automatiquement

Q28 — Une bonne PR doit être :
A. Très large
B. Petite et ciblée
C. Remplie de fichier, utiles ou non
D. Quotidienne

Gestion de versions avec Git

› Collaboration : pull-requests, code-review

Q29 — Une PR se fusionne (merge) quand :
A. Quelqu'un l'a relue
B. Les reviewers ont approuvé
C. Elle contient du code non testé
D. Elle casse le pipeline

Q30 — Lors d'une review, il faut :
A. Être respectueux et constructif
B. Critiquer l'auteur
C. Refuser autant de PR que possible
D. Ne jamais tester le code

Gestion de versions avec Git

› Dépôt partagé

Q31 — Un dépôt partagé est :

A. Un dépôt utilisé par plusieurs personnes
B. Un dépôt privé inaccessible
C. Un dépôt distant
D. Un dépôt local

Q32 — Quelle commande envoie du code vers un dépôt partagé ?

A. git pull
B. git push
C. git add
D. git clone

Gestion de versions avec Git

› Dépôt partagé

Q33 — Que signifie “origin” dans Git ?

A. Le nom du PC

B. Le nom par défaut du dépôt distant

C. Le nom d’une branche

D. Le nom d’un tag

Q34 — Comment récupérer les dernières modifications du dépôt partagé ?

A. git push

B. git clone

C. git pull

D. git commit

Gestion de versions avec Git

› Dépôt partagé

Q35 — Pourquoi utiliser un dépôt partagé ?

A. Pour travailler seul
B. Pour collaborer efficacement
C. Pour supprimer l'historique
D. Pour empêcher le versionnage

Q36 — Quel outil héberge un dépôt partagé ?

A. GitHub
B. Git
C. VSCode
D. Excel

Gestion de versions avec Git
› Dépôt partagé

Q37 — Que faut-il faire avant de pousser du code ?
A. S'assurer d'être à jour (git pull)
B. Ne rien faire
C. Supprimer tous les nouveaux fichiers
D. Annuler les commits locaux

Q38 — Pour travailler à plusieurs, il faut :
A. Utiliser chacun son dépôt privé sans partage
B. Utiliser un dépôt partagé et des branches
C. Eviter de communiquer
D. Push directement sur la branche main sans PR

Gestion de versions avec Git

› Dépôt partagé

Q39 — Un dépôt partagé évite :
A. Les conflits potentiels
B. La perte de code
C. La collaboration
D. Le versionning

Q40 — Qui peut pousser sur un dépôt partagé ?
A. Uniquement le professeur
B. Les contributeurs autorisés
C. Tout le monde sur Internet
D. Personne

TP

https://tinyurl.com/3wetrdvu

1. Créez un repository vide sur Github/Gitlab
2. Invitez un camarade (à votre droite/gauche) et moi-même (hibanajjar998) (**CHACUN Crée son propre repository!**)
3. Clonez-le sur votre ordinateur et poursuivez le travail localement;
4. Sur la branche main, créez un fichier git_cours.txt où vous écrirez un commentaire sur ce que vous avez retenu de ce cours. + add + commit
5. Pousser le nouveau fichier vers le dépôt distant (sur origin/main)
6. Votre camarade créera son clone local de votre dépôt, puis une branche develop, sur laquelle il créera aussi une branche feature/complete_description,
7. Sur cette branche de fonctionnalité, il modifiera le fichier git_cours.txt en ajoutant un autre élément du cours;
8. Votre camarade poussera les deux branches dans le repository distant
9. Il ouvrira un PR dans Github, pour fusionner feature/complete_description dans develop
10. Révisez le PR, commentez, puis approuver/refuser.
11. Si vous approuvez, supprimez la branch feature/complete_description.

Projet tutoré: sujet

1. Proposition individuelle d'un sujet (réflexion 20min)
   - Pensez à l'impact (social, environnemental, financier)
   - Cherchez à résoudre un problème réel (société, université, etc)
   - Pensez au côté innovant (IA, VR)
   - Qu'est ce qui impressionnera le jury
   - Qu'est ce qui intéresserai le recruteur (expérience valorisée)
   - Projet faisable en 5 mois
   - À déployer tôt (version beta) pour collecter du feedback

2. Ajoutez votre nom + prénom+ un court descriptif (ne dépassant pas 4 phrases), sur le ppt partagé <lien supprimé>

Prochaine séance

1. Conteneurisation avec Docker
2. Ponctualité, à l'heure on est déjà en classe (passez le message à vos camarades)
3. Pensez à vos sujets projet (amélioration / changement d'idées)

Avant de quitter:

1. Ajustez vos identifiants Github
2. m'envoyer un email avec votre: Nom, prénom, identifiant github