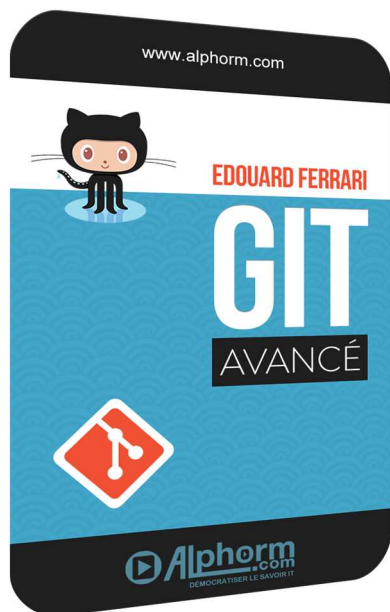




Alphorm.com

Formation Git avancé

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com

Formation Git Avancé

alphorm.com™©

Plan

- Présentation du formateur
- Plan de formation
- Objectifs de la formation
- Public concerné
- Les possibilités de Git
- Les connaissances requises



Formation Git Avancé

alphorm.com™©

Présentation du formateur

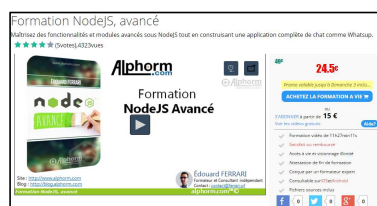
Edouard FERRARI

- contact@ferrari.wf
- Développeur full stack chez Summview
- Mission de conseil, d'architecture et de migration
- Mes références :
 - LinkedIn : <https://fr.linkedin.com/in/edouardferrari>
 - Alphorm : <http://www.alphorm.com/formateur/edouard-ferrari>
 - Github : <https://github.com/didouard>



Mes formations

- Coursus complet sur NodeJS



- Formation, les fondamentaux sur Git





Plan de la formation

- Git sur le serveur
 - Protocoles
 - Installation de Git sur un serveur
 - Mise en place du serveur
 - Git daemon et Git HTTP
 - GitWeb
 - GitLab
- Git distribué
 - Développements distribués
 - Guide pour validation
 - Projet d'une petite équipe
 - Équipe privée importante
 - Projet public dupliqué
 - Projet public via email
 - Appliquer des patches



Objectifs de la formation

- Étudier Git pour pouvoir l'utiliser pour des projets personnels ou professionnels.
- Git surclasse les autres outils **SCM** (*Source Code Management*) par sa performance, la taille des dépôts et ses fonctionnalités uniques.
- Cette formation est la seconde du cursus GIT, nous allons étudier précisément comment travailler avec GIT en entreprise et avec un projet important.
- À l'issue de la formation, vous aurez appris à configurer et utiliser GIT dans un contexte de gestion quotidienne des sources d'un projet professionnel.



Public concerné

- À qui s'adresse cette formation :
 - Aux étudiants
 - Aux développeurs
 - Aux chefs de projet
 - Aux amoureux des nouvelles technologies
 - Aux personnes souhaitant mieux s'organiser
- À tout le monde !

Les possibilités de Git

- Git permet :
 - De versionner ses fichiers (code, images, documents, ...)
 - De mettre en attente une version et de travailler sur une autre
 - De pouvoir fusionner un même fichier sur lequel plusieurs personnes ont travaillé
 - Organiser son travail par version
 - Publier en production son code à partir d'une version donnée
 - Avoir un historique précis de son projet
 - De pouvoir blâmer quelqu'un ! ;-)
 - ...



Les connaissances requises

- Formation Git, le système de contrôle de version :
<http://www.alphorm.com/tutoriel/formation-en-ligne-git-le-systeme-de-contrôle-de-version>

Formation Git, le système de contrôle de version

Maîtrisez Git, le leader de contrôle de version et apprenez à versionner votre code comme jamais auparavant.

★★★★★ (5votes), 12514vues



Formation Git Avancé

alphorm.com™©

Liens et ressources

- <https://git-scm.com/> : Site officiel de Git
- [Http://github.com](http://github.com) : Plateforme en ligne de Git
- <http://gitref.org/> : Pour ne jamais oublier les commandes
- <https://try.github.io/> : Pour s'entraîner

Formation Git Avancé

alphorm.com™©



Formation Git Avancé

alphorm.com™©



Alphorm.com

Git serveur

Les protocoles

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com
alphorm.com™©

Plan

- Protocole local
- Protocole HTTP
- Protocole SSH
- Protocole Git



Protocole local

- Pour cloner un répertoire local
 - `$> git clone /path/to/project.git`
- Pour ajouter un projet en local
 - `$> git remote add local_proj /path/to/project.git`
- **Avantage** 😊
 - Facile à mettre en place
- **Inconvénients** ☹️
 - Il est généralement difficile de rendre disponible un partage réseau et d'y accéder en dehors d'un réseau local.



Protocole HTTP

- Deux manières de communiquer en HTTP
 1. HTTP intelligent (comprends une couche d'authentification)
 2. HTTP idiot :
 - Les fichiers sont simplement placer sur un serveur web.
 - Nécessite l'ajout un post-hook.
- **Avantage 😊** (http intelligent)
 - Une seule URL
 - HTTP / HTTPS, la plupart des firewalls sont ouverts sur le port 80/443
- **Inconvénients ☹**
 - Plus compliqué à mettre en place.



Protocole SSH

- Pour accéder à un GIT
 - `$> git clone ssh://utilisateur@serveur/projet.git`
- **Avantage 😊**
 - Basé sur le protocole SSH (Connexion cryptée + authentification)
- **Inconvénients ☹**
 - Impossible de proposer un accès anonyme

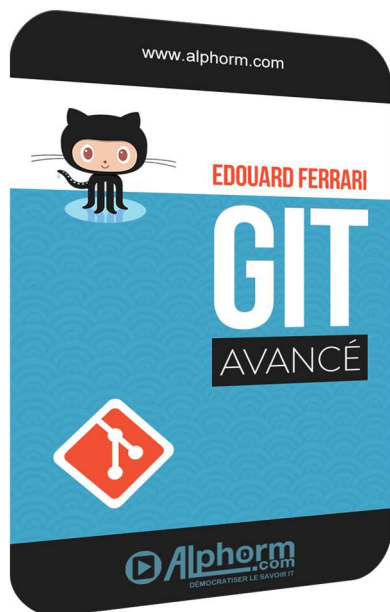
Protocole Git

- Le protocole Git. Celui-ci est géré par un daemon spécial livré avec Git.
- **Avantage** 😊
 - Protocole plus rapide
- **Inconvénients** ☹️
 - Pas d'authentification, tout le monde peut pousser

Ce qu'on a couvert

- Les différents protocoles disponibles :
 - Protocole local
 - Protocole HTTP
 - Protocole SSH
 - Protocole Git
- Prochaine vidéo :
 - Installation de Git sur un serveur





Alphorm.com

Git serveur

Copie d'un projet git sur un serveur

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com

Formation Git Avancé

alphorm.com™©

Plan

- Copie du projet
- Copie du dépôt nu sur un serveur
- Petites installations
- Génération des clés publiques SSH
- Ajout des clefs SSH au serveur



Formation Git Avancé

alphorm.com™©

Copie du projet

- Pour réaliser l'installation initiale d'un serveur Git, il faut exporter un dépôt existant dans un nouveau dépôt nu
 - `$> git clone --bare mon_project mon_projet.git`
- Equivalent à :
 - `$ cp -Rf mon_projet/.git mon_projet.git`

Copie du dépôt nu sur un serveur

- A présent on peut copier notre projet sur notre serveur :
 - `$> scp -r mon_projet.git git@git.exemple.com:/opt/git`
- Et le cloner :
 - `$ git clone git@git.exemple.com:/opt/git/mon\_projet.git`
- Git ajoutera automatiquement les droits de groupe en écriture à un dépôt si vous lancez la commande `git init` avec l'option `--shared`.
 - `$ ssh git@git.exemple.com`
 - `$ cd /opt/git/mon_projet.git`
 - `$ git init --bare --shared`



Petit installation

- A partir du protocole SSH, on peut :
 - Soit créer plusieurs utilisateurs et partager le groupe entre ces utilisateurs.
 - Soit créer un utilisateur et partager la clef privée.
 - Soit ajouter plusieurs clef ssh publique et les ajouter au compte git.



Génération des clés publiques SSH

- Sur les postes utilisateurs :
 - `$> ssh-keygen`
- Le résultat :
 - `$> cat ~/.ssh/id_rsa.pub`

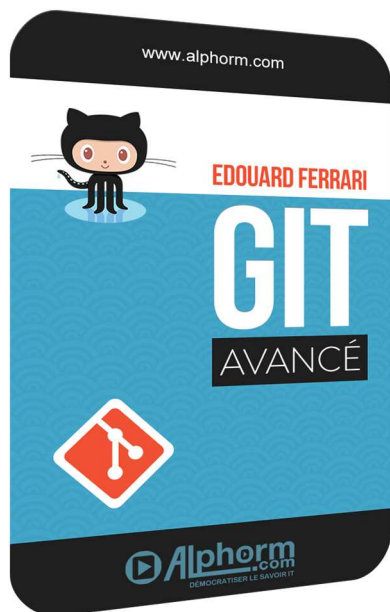
Ajout des clefs SSH au serveur

- Sur le serveur :
 - `$> sudo adduser git`
 - `$> su git`
 - `$> cd`
 - `$> mkdir .ssh && chmod 700 .ssh`
 - `$> touch .ssh/authorized_keys && chmod 600 .ssh/authorized_keys`
- Puis ajouter les clefs publiques :
 - `$ cat /tmp/id_rsa.john.pub >> ~/.ssh/authorized_keys`
 - `$ cat /tmp/id_rsa.josie.pub >> ~/.ssh/authorized_keys`

Ce qu'on a couvert

- Nous avons vu comment
 - Comment copier un repertoire pour l'exporter sur un serveur
 - Les 3 methodes pour partager le serveur avec le protocole SSH
 - Comment créer les clefs SSH
- Prochaine vidéo
 - Mise en place du serveur





Alphorm.com

Git Serveur

Création d'un projet git sur le serveur

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com

Formation Git Avancé

alphorm.com™©

Plan

- Création du répertoire GIT
- Première configuration
- Configuration des PC utilisateurs
- Restriction du compte 'git' sur le serveur



Formation Git Avancé

alphorm.com™©

Création du répertoire GIT

- Maintenant, vous pouvez créer un dépôt vide nu en lançant la commande *git init* avec l'option *--bare*, ce qui initialise un dépôt sans répertoire de travail :
 - \$> cd /opt/git
 - \$> mkdir project.git
 - \$> cd project.git
 - \$> git init --bare

Première configuration

- Sur le premier PC :
 - \$> cd monproject
 - \$> git init
 - \$> git add .
 - \$> git commit -m 'première validation'
 - \$> git remote add origin git@gitserveur:/opt/git/projet.git
 - \$> git push origin master

Configuration des PC utilisateurs

- Sur les autres PC des utilisateurs :
 - `$> git clone git@gitserveur:repository/projet.git`
 - `$> cd projet`
 - `$> emacs LISEZMOI`
 - `$> git commit -am 'correction du fichier LISEZMOI'`
 - `$> git push origin master`

Restriction du compte 'git' sur le serveur

- Vous pouvez simplement restreindre l'utilisateur **git** à des actions Git avec un shell limité appelé **git-shell** qui est fourni avec Git.
 - `$ cat /etc/shells # voir si 'git-shell' est déjà déclaré. Sinon...`
 - `$ which git-shell # s'assurer que git-shell est installé sur le système`
 - `$ sudo vim /etc/shells # et ajouter le chemin complet vers git-shell`
 - `$ sudo chsh git # saisir le chemin vers git-shell, souvent : /usr/bin/git-shell`
- À présent, l'utilisateur **git** ne peut plus utiliser la connexion SSH que pour pousser et tirer sur des dépôts Git
 - `$ ssh git@gitserveur # retourne une erreur`

Ce qu'on a couvert

- Nous avons vu comment mettre en place un serveur collaboratif
- Prochaine vidéo
 - Démon (Daemon) Git



Alphorm.com

Git Serveur

Git daemon
et Git HTTP

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com

Plan

- Git daemon
- Script de démarrage automatique
- Git HTTP



Git Daemon

- Pour lancer le démon :
 - `$> git daemon --reuseaddr --base-path=/opt/git/ /opt/git/`
- Puis dans chaque projet il suffit de créer un fichier :
 - `$> touch git-daemon-export-ok`

Script de démarrage automatique

- Avec Ubuntu, il est possible de créer un script de démarrage automatique :

- `$> emacs /etc/event.d/local-git-daemon`

```
start on startup
stop on shutdown
exec /usr/bin/git daemon \
  --user=git --group=git \
  --reuseaddr \
  --base-path=/opt/git/ \
  /opt/git/
respawn
```

- On inscrit le daemon
 - `$> initctl start local-git-daemon`

Git HTTP

- Nous avons à présent un accès authentifié par SSH et un accès non authentifié par `git://`, mais il existe aussi un protocole qui peut faire les deux à la fois.
- La configuration d'un HTTP intelligent revient simplement à activer sur le serveur un script CGI livré avec Git qui s'appelle **git-http-backend**.
- Nous utiliserons Apache2 comme serveur HTTP/CGI

Git HTTP

- Sur Ubuntu / Debian, installons le apache à partir des paquets :
 - \$ sudo apt-get install apache2 apache2-utils
 - \$ a2enmod cgi alias env
- Ensuite, nous devons ajouter quelques lignes à la configuration d'Apache pour qu'il lance *git-http-backend*
 - SetEnv GIT_PROJECT_ROOT /opt/git
 - SetEnv GIT_HTTP_EXPORT_ALL
 - ScriptAlias /git/ /usr/libexec/git-core/git-http-backend/

Git HTTP

- Puis nous allons indiquer à Apache les repertoires de travail :

```
<Directory "/usr/lib/git-core*">
  Options ExecCGI Indexes
  Order allow,deny
  Allow from all
  Require all granted
</Directory>

<LocationMatch "^/git/.*/git-receive-pack$">
  AuthType Basic
  AuthName "Git Access"
  AuthUserFile /opt/git/.htpasswd
  Require valid-user
</LocationMatch>
```

- Enfin, nous n'avons plus qu'à créer le fichier de mot de passe :
 - \$> htdigest -c /opt/git/.htpasswd "Git Access" eferrari

Ce qu'on a couvert

- Nous avons vu comment créer deux nouveaux types de serveur GIT
- Prochaine vidéo:
 - GitWeb



Alphorm.com

Git sur le serveur GitWeb

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com

Plan

- GitWeb
- Démarrage
- Hébergement permanent



GitWeb

- Après avoir réglé les accès de base en lecture/écriture et en lecture seule pour vos projets, vous souhaiterez peut-être mettre en place une interface web simple de visualisation.
- Git fournit un script CGI appelé GitWeb qui est souvent utilisé à cette fin.



projects / .git / summary			git
summary shortlog log commit commitdiff tree			commit 1 2 search: re
description Unnamed repository; edit this file 'description' to name the repository.			
owner Ben Straub			
last change Wed, 11 Jun 2014 12:20:23 -0700 (21:20 +0200)			
shortlog			
2014-06-11	Carlos Martin...	remote: update documentation	commit commitdiff tree branches
2014-06-11	Vicent Marti	Merge pull request #2817 from libgit2/cmn/rewalk-array-fix	commit commitdiff tree branches
2014-06-10	Carlos Martin...	rewalk: more sensible array handling	commit commitdiff tree branches
2014-06-10	Vicent Marti	Merge pull request #2416 from libgit2/cmn/treebuilder...	commit commitdiff tree branches
2014-06-10	Carlos Martin...	paths: use C guards in header	commit commitdiff tree branches
2014-06-09	Carlos Martin...	treebuilder: insert sorted	commit commitdiff tree branches
2014-06-09	Carlos Martin...	remote: fix rename docs	commit commitdiff tree branches
2014-06-08	Carlos Martin...	Merge branch 'cmn/soversion' into development	commit commitdiff tree branches
2014-06-08	Carlos Martin...	Bump version to 0.21.0	commit commitdiff tree branches
2014-06-08	Carlos Martin...	Change SOVERSION at API breaks	commit commitdiff tree branches
2014-06-08	Vicent Marti	Merge pull request #2407 from libgit2/cmn/remote-rename...	commit commitdiff tree branches
2014-06-08	Vicent Marti	Merge pull request #2409 from phkelly/win32_thread_fixes	commit commitdiff tree branches
2014-06-07	Philip Kelley	React to review feedback	commit commitdiff tree branches
2014-06-07	Philip Kelley	Win32: Fix object::cache::threadmania test on x64	commit commitdiff tree branches
2014-06-07	Philip Kelley	Merge pull request #2406 from phkelly/win32_test_fixes	commit commitdiff tree branches
2014-06-07	Philip Kelley	Win32: Fix diff::workdir::submodules test #2361	commit commitdiff tree branches
tags			
3 weeks ago	v0.21.0-rc1	commit commitdiff tree	
7 months ago	v0.20.0	commit commitdiff tree	
12 months ago	v0.19.0	commit commitdiff tree	
14 months ago	v0.18.0	commit commitdiff tree	
2 years ago	v0.17.0	commit commitdiff tree	
2 years ago	v0.16.0	libgit2 v0.16.0	commit commitdiff tree
2 years ago	v0.15.0	commit commitdiff tree	
2 years ago	v0.14.0	commit commitdiff tree	
3 years ago	v0.13.0	commit commitdiff tree	
3 years ago	v0.12.0	commit commitdiff tree	
3 years ago	v0.11.0	commit commitdiff tree	

git Démarrage

- Pour démarrer
 - `$> git instaweb --httpd=webrick`
- Pour arrêter
 - `$> git instaweb --httpd=webrick --stop`

Hébergement permanent

- Si vous souhaitez fournir l'interface web en permanence sur le serveur pour votre équipe ou pour un projet open source que vous hébergez, il sera nécessaire d'installer le script CGI pour qu'il soit appelé par votre serveur web.
 - \$ git clone git://git.kernel.org/pub/scm/git/git.git
 - \$ cd git/
 - \$ make GITWEB_PROJECTROOT="/opt/git" prefix=/usr gitweb
 - [...]
 - \$ sudo cp -Rf gitweb /var/www/

Hébergement permanent

- Puis il suffit de configurer apache :

```
<VirtualHost *:80>
  ServerName gitserver
  DocumentRoot /var/www/gitweb
  <Directory /var/www/gitweb>
    Options ExecCGI +FollowSymLinks +SymLinksIfOwnerMatch
    AllowOverride All
    order allow,deny
    Allow from all
    AddHandler cgi-script cgi
    DirectoryIndex gitweb.cgi
  </Directory>
</VirtualHost>
```

Ce qu'on a couvert

- Nous avons vu comment créer un serveur web pour avoir une overview sur un projet Git.
- Prochaine vidéo:
 - GitLab



Alphorm.com

Git sur le serveur GitLab

Site : <http://www.alphorm.com>

Blog : <http://blog.alphorm.com>



Édouard FERRARI

Formateur et Consultant indépendant

Contact : edouard.ferrari@gmail.com

Plan

- GitLab
- Mise en place
- Configuration des utilisateurs
- Configuration des groupes



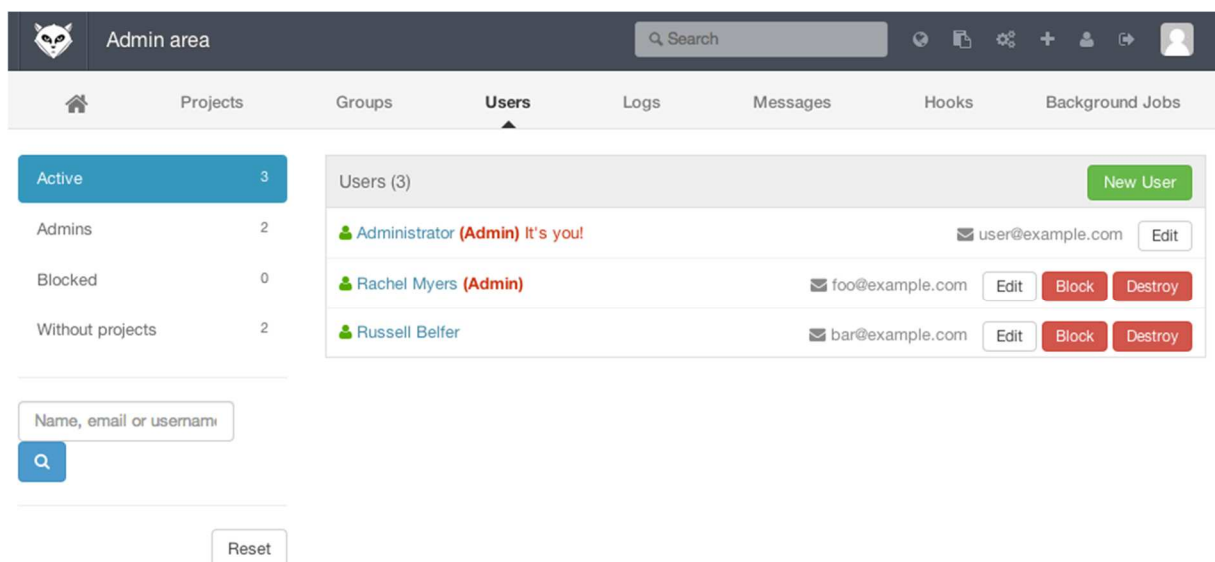
GitLab

- GitWeb reste tout de même simpliste.
- Si vous cherchez un serveur Git plus moderne et complet, il existe quelques solutions libres pertinentes.
- Comme GitLab est un des plus populaires, nous allons prendre son installation et son utilisation comme exemple.
- Cette solution est plus complexe que l'option GitWeb et demandera indubitablement plus de maintenance, mais elle est aussi plus complète.

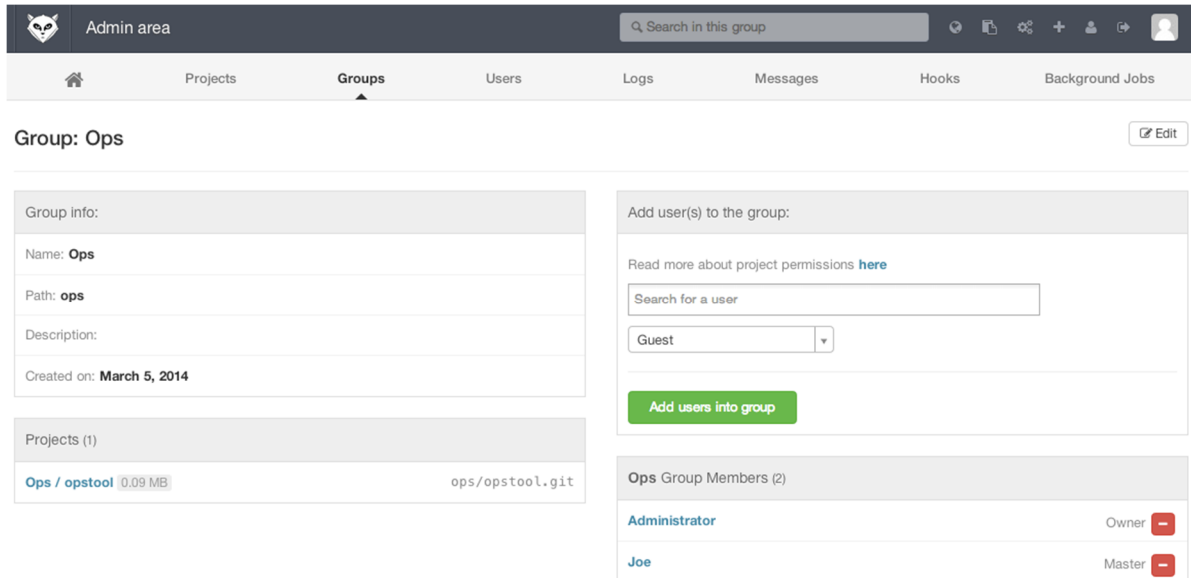
Mise en place

- GitLab est une application web reposant sur une base de données.
- L'installation est très bien documentée et supportée.
- Nous pouvons installer GitLab de plusieurs façons :
 - Installation manuelle à partir des sources ou de packages.
 - Installation avec une image virtuelle
 - Installation « dans le cloud »

Configuration des utilisateurs



Configuration des groupes



The screenshot shows the GitLab Admin interface for a group named 'Ops'. The top navigation bar includes 'Admin area', a search bar, and links for 'Projects', 'Groups', 'Users', 'Logs', 'Messages', 'Hooks', and 'Background Jobs'. The 'Groups' tab is active.

Group: Ops [Edit]

Group info:

- Name: **Ops**
- Path: **ops**
- Description:
- Created on: **March 5, 2014**

Projects (1)

Project	Size	Path
Ops / opstool	0.09 MB	ops/opstool.git

Add user(s) to the group:

Read more about project permissions [here](#)

Search for a user:

Guest

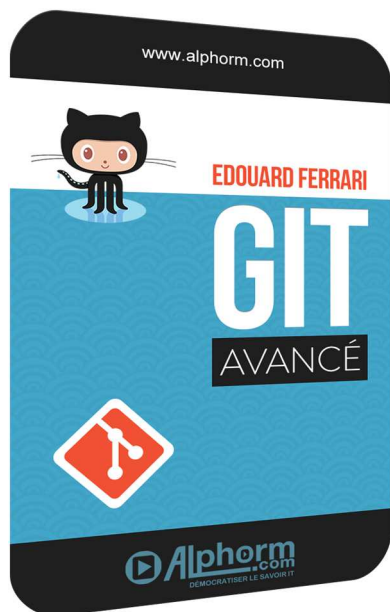
Ops Group Members (2)

User	Role
Administrator	Owner
Joe	Master

Ce qu'on a couvert

- Nous avons vu comment utiliser GitLab
- Prochain Chapitre :
 - Git distribué





Alphorm.com

Git distribué Développements distribués

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com

Formation Git Avancé

alphorm.com™©

Plan

- Développements distribués
- Gestion centralisée
- Mode du gestionnaire d'intégration
- Mode dictateur et ses lieutenants



Formation Git Avancé

alphorm.com™©



Introduction

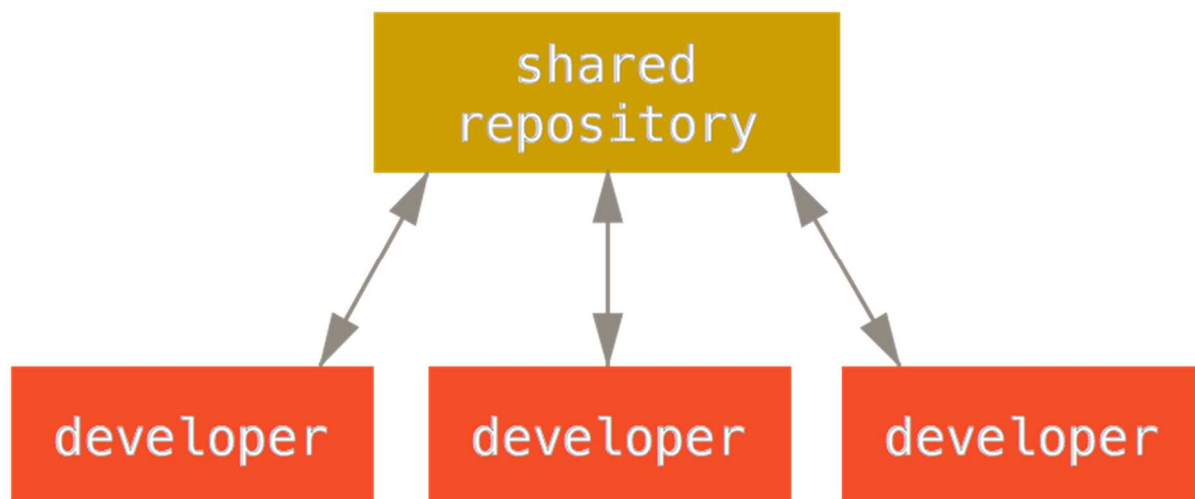
- Dans ce chapitre, nous allons découvrir comment travailler dans un environnement distribué avec Git en tant que contributeur ou comme intégrateur.
- Nous allons voir une manière de contribuer efficacement à un projet et de rendre la vie plus facile au mainteneur.



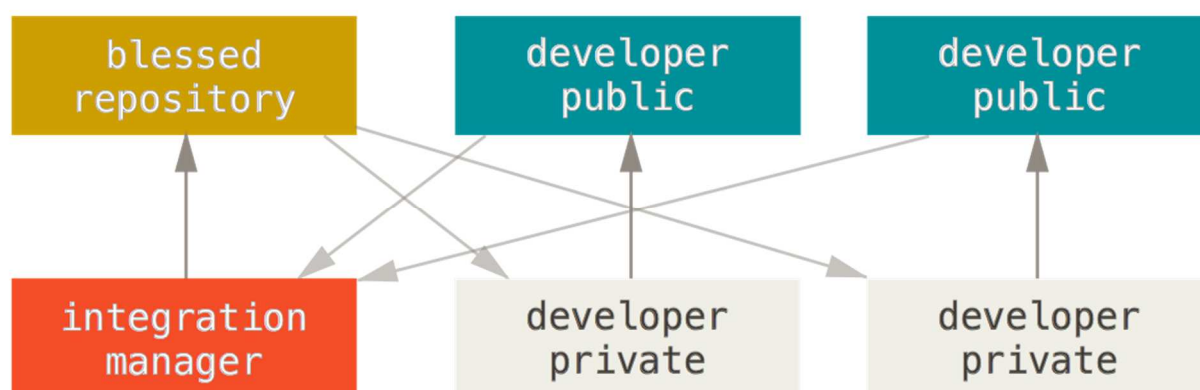
Développements distribués

- La nature distribuée de Git permet une bien plus grande flexibilité dans la manière dont les développeurs collaborent sur un projet.
- Dans Git, tout développeur est potentiellement un nœud et un concentrateur
- Chaque développeur peut à la fois contribuer du code vers les autres dépôts et maintenir un dépôt public sur lequel d'autres vont baser leur travail et auquel ils vont contribuer.

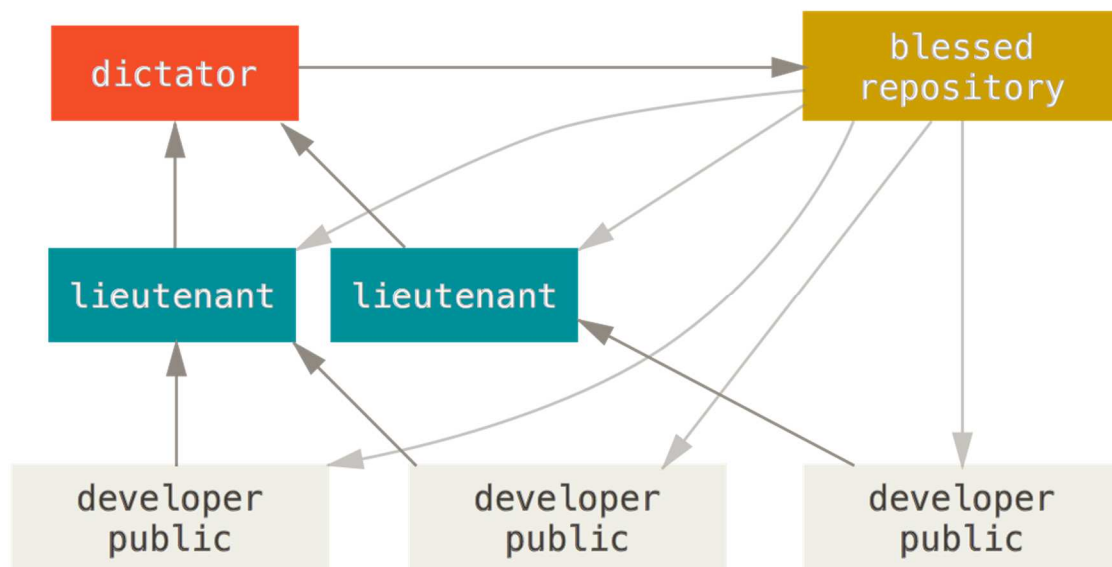
Gestion centralisée



Mode du gestionnaire d'intégration



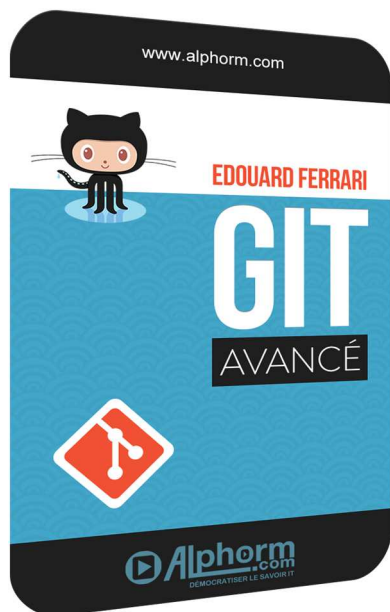
Mode dictateur et ses lieutenants



Ce qu'on a couvert

- Nous avons vu dans cette vidéo trois méthodes de travail en entreprise ou en communauté :
 - La gestion centralisée
 - Le mode du gestionnaire d'intégration
 - Le mode dictateur et ses lieutenants
- Prochaine vidéo :
 - Guide pour validation





Alphorm.com

Git distribué

Guide pour validation

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com

Formation Git Avancé

alphorm.com™©

Plan

- Guides pour une validation
- Les erreurs d'espace
- Un sujet = une validation
- Le message de validation



Formation Git Avancé

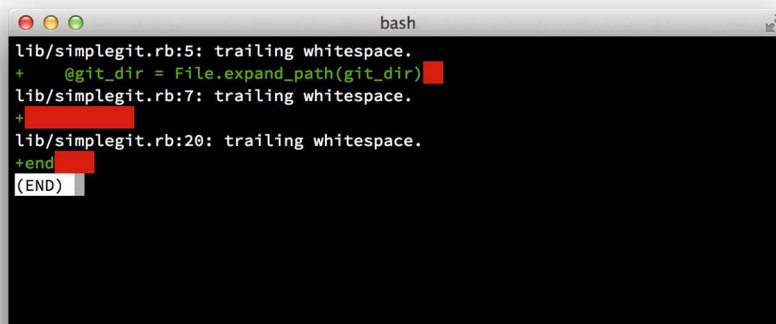
alphorm.com™©

Guide pour validation

- Nous allons voir un guide de bonne conduite sur les validations.
- Git fournit un guide de bonnes pratiques qui est disponible dans les sources de Git '*Documentation/SubmittingPatches*'.
- Il ne faut pas soumettre de patchs comportant des erreurs d'espace.

Les erreurs d'espace

- Il ne faut pas soumettre de patchs comportant des erreurs d'espace.
- Git fournit un moyen simple de le vérifier — avant de valider, lancez la commande
 - `$> git diff --check`



```
bash
lib/simplegit.rb:5: trailing whitespace.
+ @git_dir = File.expand_path(git_dir)
lib/simplegit.rb:7: trailing whitespace.
+ 
lib/simplegit.rb:20: trailing whitespace.
+end
(END)
```

Un sujet = une validation

- Ne développez pas pendant un week-end entier en traitant de 5 problèmes différents.
- Assurez-vous de faire de chaque validation une modification logiquement atomique.
- Faites une branche, avec 5 validations différentes.
- Cette approche simplifie aussi le retrait ou l'inversion ultérieure d'une modification en cas de besoin.

Le message de validation

- S'habituer à écrire des messages de validation de qualité facilite grandement l'emploi et la collaboration avec Git.
- En règle générale, les messages doivent débiter par une ligne unique d'au plus 50 caractères décrivant concisément la modification, suivie d'une ligne vide, suivie d'une explication plus détaillée.
- Le projet Git exige que l'explication détaillée inclue la motivation de la modification en contrastant le nouveau comportement par rapport à l'ancien.
- Une bonne règle consiste aussi à utiliser le présent de l'impératif ou des verbes substantivés dans le message:
 - « Ajoute des tests pour » ou « Ajout de tests pour ».

Le message de validation

- Voici ci-dessous un modèle écrit par Tim Pope :

```
Court résumé des modifications (50 caractères ou moins)

Explication plus détaillée, si nécessaire. Retour à la ligne vers 72
caractères. Dans certains contextes, la première ligne est traitée
comme le sujet d'un courriel et le reste comme le corps. La ligne
vide qui sépare le titre du corps est importante (à moins d'omettre
totalement le corps). Des outils tels que rebase peuvent être gênés
si vous les laissez collés.

Paragraphes supplémentaires après des lignes vides.

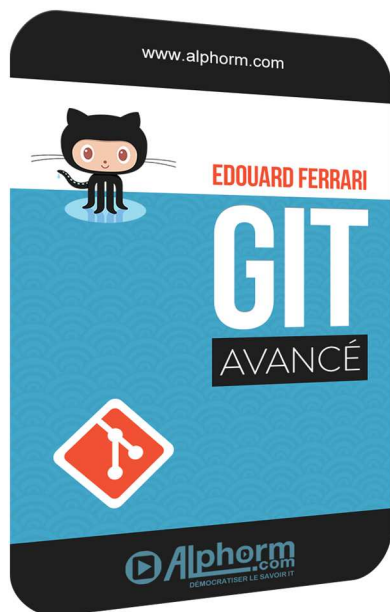
- Les listes à puce sont aussi acceptées

- Typiquement, un tiret ou un astérisque précédés d'un espace unique
séparés par des lignes vides mais les conventions peuvent varier
```

Ce qu'on a couvert

- Quelles sont les « best practice » pour faire des validations
- Prochaine vidéo :
 - Contribution à un projet





Alphorm.com

Git distribué

Le cas d'une petite équipe

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com

Formation Git Avancé

alphorm.com™©

Plan

- Introduction
- Le travail à deux



Formation Git Avancé

alphorm.com™©



Introduction

- Chaque projet github est différent, il est difficile d'indiqué une méthode globale de contribution à un projet.
- On peut décrire 3 grandes variables :
 - La taille du corps des contributeurs
 - Le mode de gestion utilisé pour le projet
 - La gestion des accès en écriture

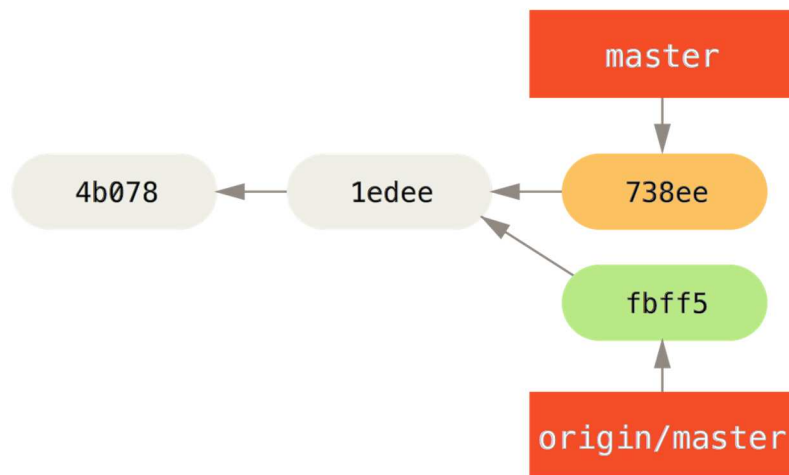


Le travail à deux

- Imaginons deux collaborateurs :
 - John
 - Jessica

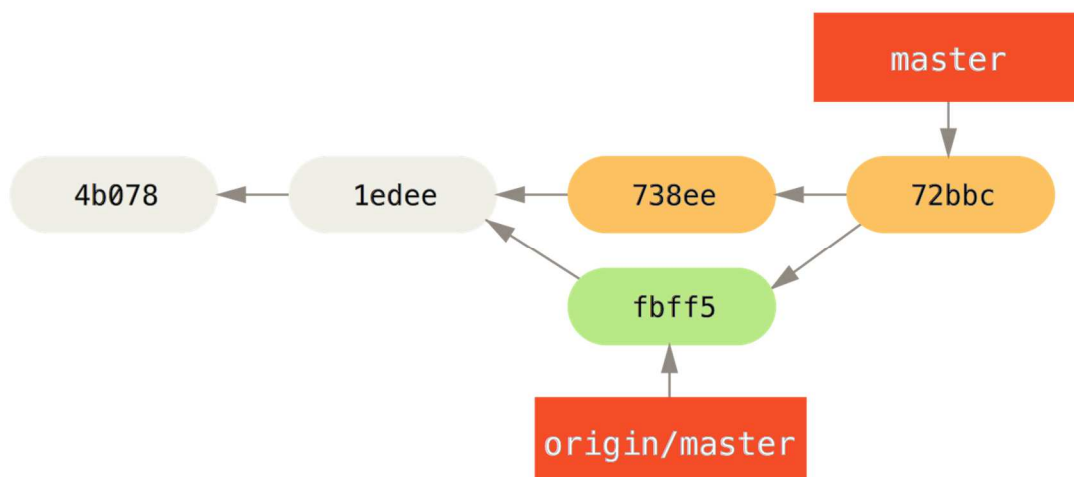
Le travail à deux

- Historique divergent de John



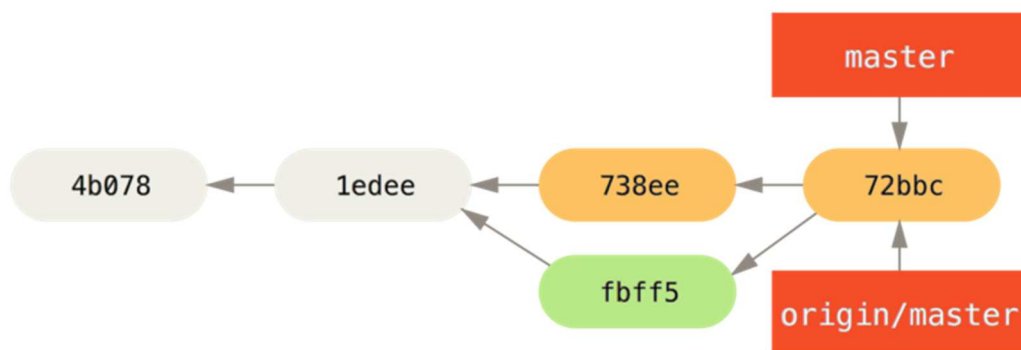
Le travail à deux

- Le dépôt de John après la fusion d'origine/master.



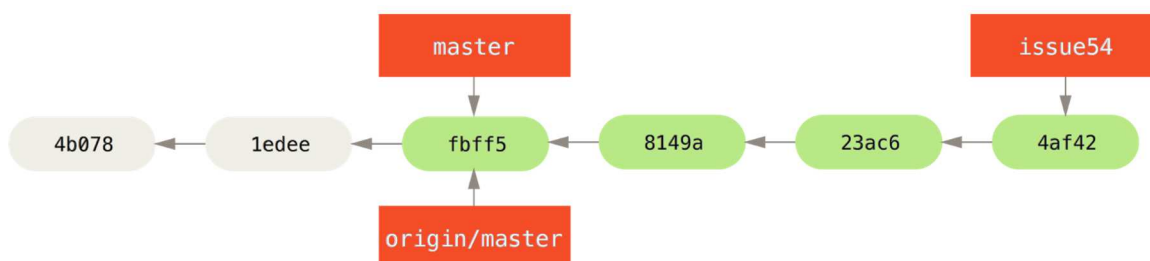
Le travail à deux

- Le dépôt de John après la fusion d'origine/master.



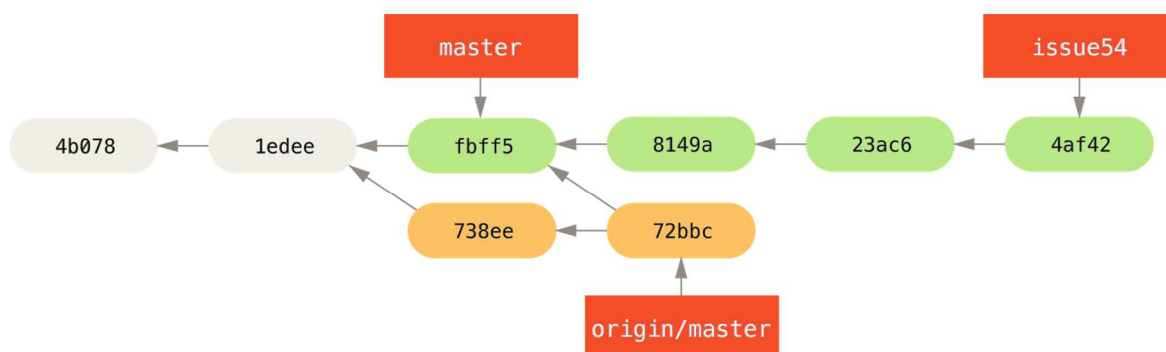
Le travail à deux

- La branche thématique de Jessica



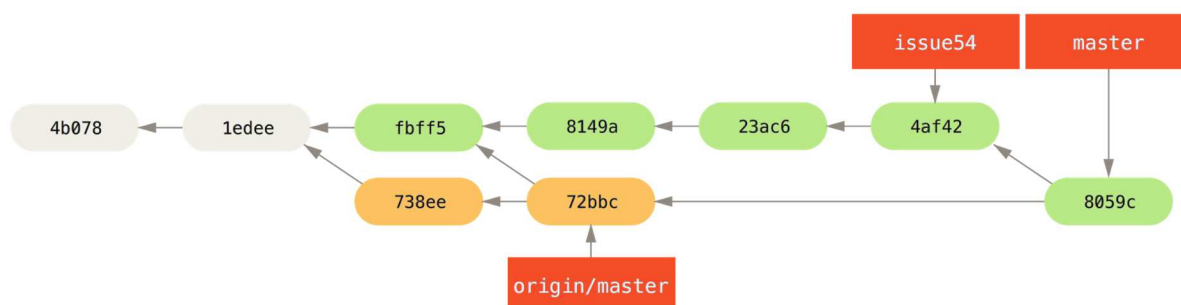
Le travail à deux

- L'historique de Jessica après avoir récupéré les modifications de John.

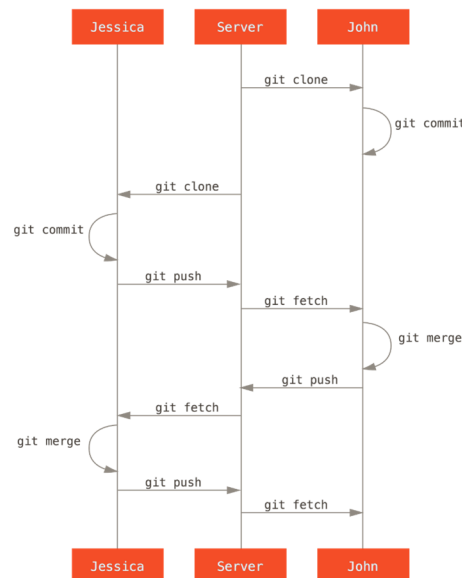


Le travail à deux

- L'historique de Jessica après avoir fusionné les modifications de John.

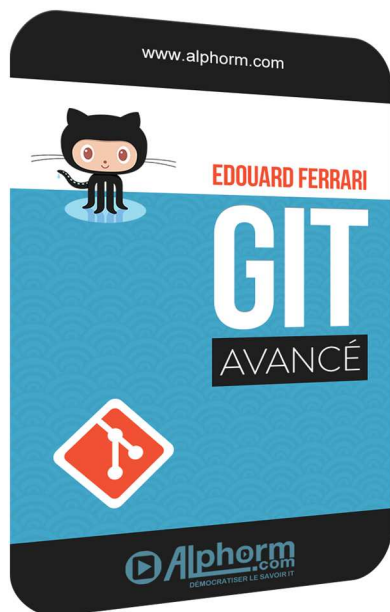


- En résumé



- Nous avons vu ensemble un des schémas les plus simples.
- Prochaines vidéo :
 - Contribution à un projet : Équipe privée importante





Alphorm.com

Git distribué

Équipe privée importante

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com

Formation Git Avancé

alphorm.com™©

Plan

- Équipe privée importante



Formation Git Avancé

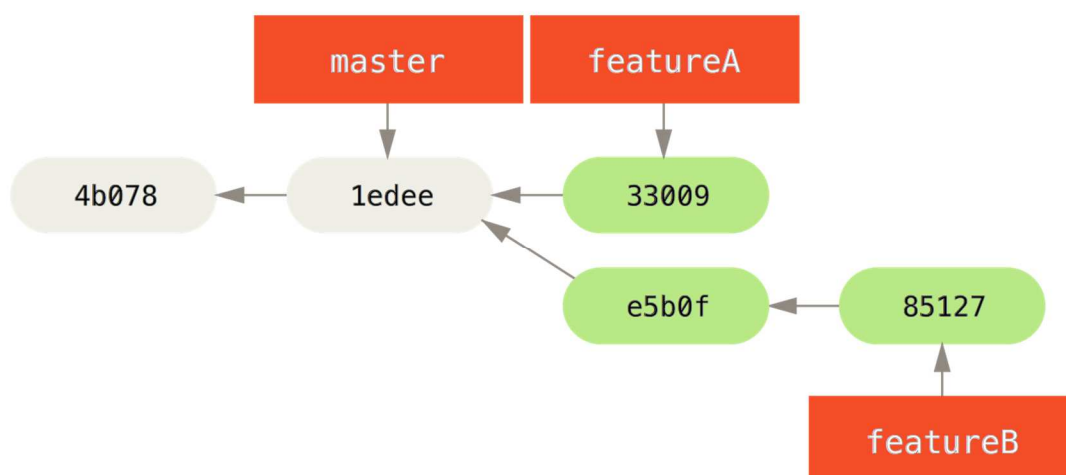
alphorm.com™©

Le travail à deux

- Imaginons que **John** et **Jessica** travaillent ensemble sur une première fonctionnalité.
- Tandis que **Jessica** et **Josie** travaillent sur une autre.
- Dans ce cas, l'entreprise utilise un mode d'opération de type « **gestionnaire d'intégration** » où le travail des groupes est intégré par certains ingénieurs, et la branche master du dépôt principal ne peut être mise à jour que par ces ingénieurs.
- Dans ce scénario, tout le travail est validé dans des branches orientées équipe, et tiré plus tard par les intégrateurs.

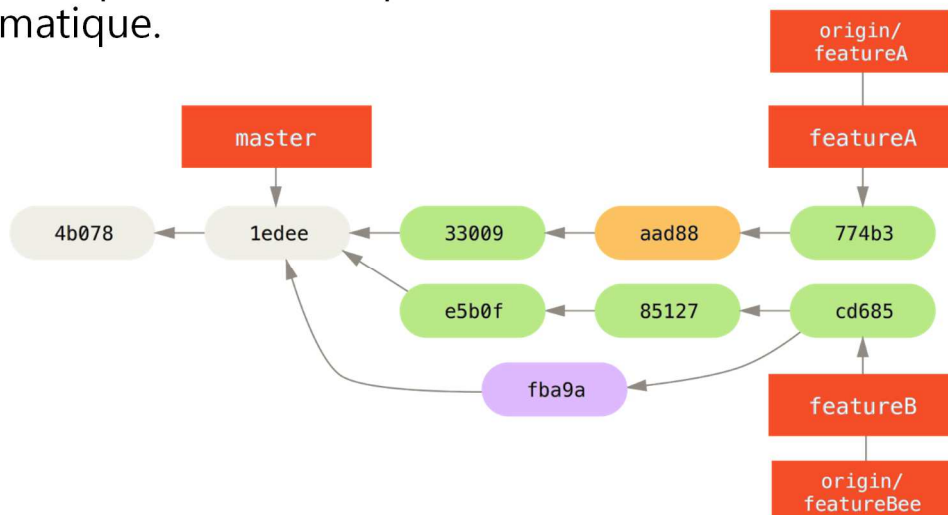
Le travail à deux

- Historique initial de Jessica



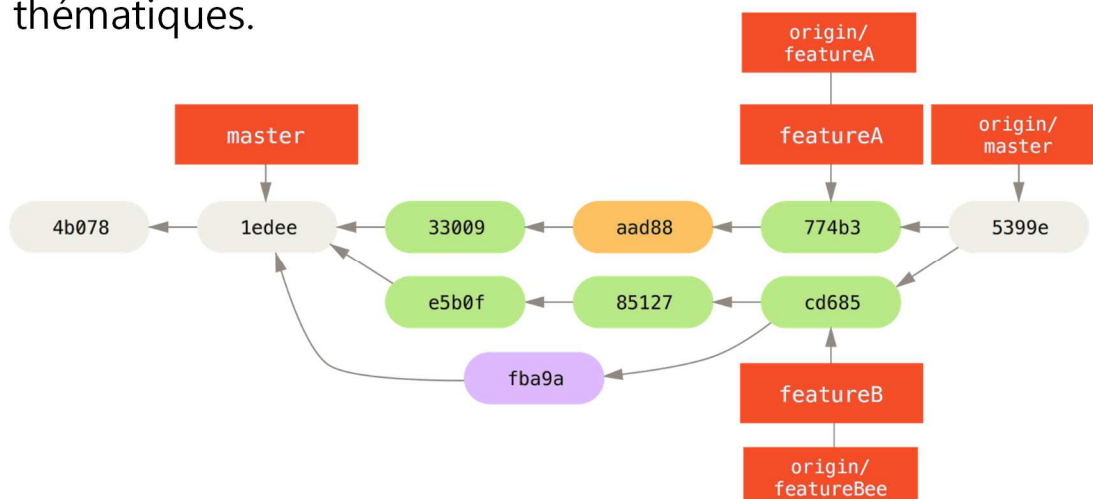
git Le travail à deux

- L'historique de Jessica après la validation dans la branche thématique.



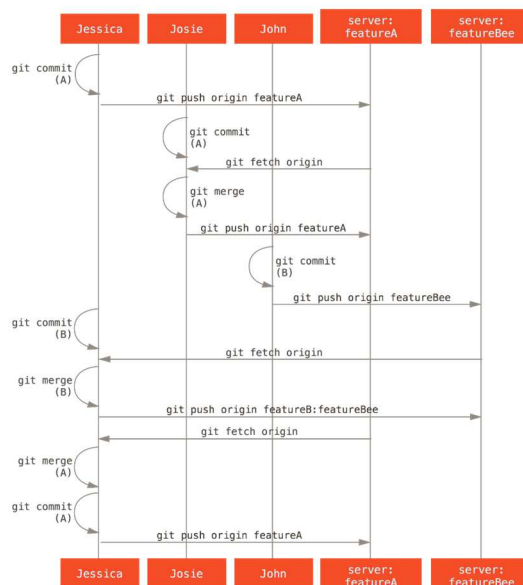
git Le travail à deux

- L'historique de Jessica après la fusion de ses deux branches thématiques.



Le travail à deux

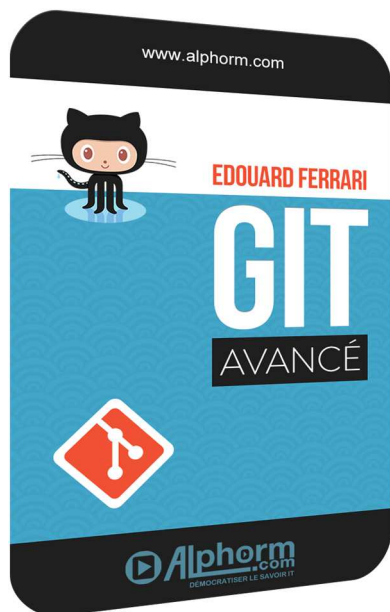
- En résumé :



Ce qu'on a couvert

- Nous avons vu ensemble un des schémas les plus utilisé en entreprise.
- De nombreux groupes basculent vers Git du fait de cette capacité à gérer plusieurs équipes travaillant en parallèle.
- Prochaine vidéo :
 - Contribution à un projet : Projet public via courriel





Alphorm.com

Git distribué

Projet public dupliqué

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com

Formation Git Avancé

alphorm.com™©

Plan

- Introduction
- Projet public dupliqué



Formation Git Avancé

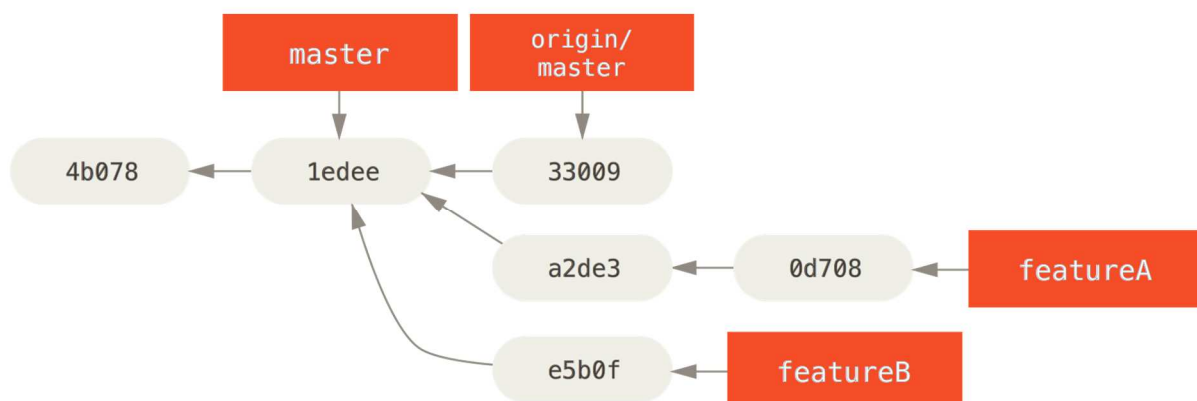
alphorm.com™©

Introduction

- Contribuer à un projet public est assez différent. Il faut présenter le travail au mainteneur.
- Vu que le projet est public, vous n'avez pas la possibilité de mettre à jour directement des branches du projet.
- De nombreux sites proposent cette méthode (dont *GitHub*, *BitBucket*, *Google Code*, *repo.or.cz*).

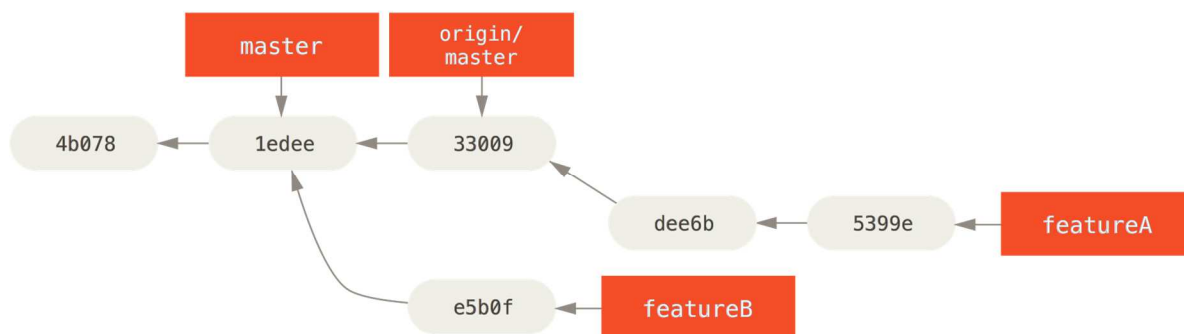
Le travail à deux

- Historique initial des commits avec les modifications de fonctionB.



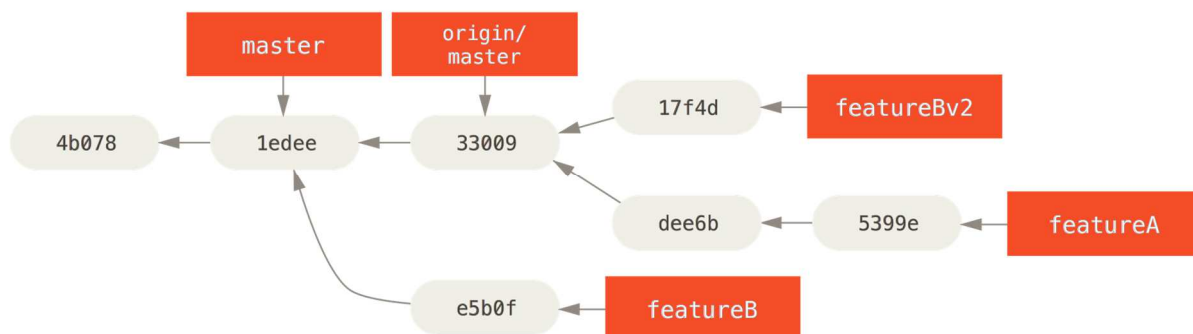
Le travail à deux

- Historique des validations après le travail sur fonctionA.



Le travail à deux

- Historique des validations après le travail sur fonctionBv2.



Ce qu'on a couvert

- Nous avons vu comment travailler sur un projet public.
- Prochaine vidéo :
 - Contribution à un projet : Projet public via email



Alphorm.com

Git distribué
Projet public
via email

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com

Plan

- Création d'un patch
- Envoyer un patch via Email



Création d'un patch

- Il est possible de créer un export d'une validation grâce au patch :
 - `$> git format-patch -M origin/master`

Envoyer un patch via email

- Vous pouvez configurer gmail pour envoyer ces patches, dans `~/.gitconfig` :

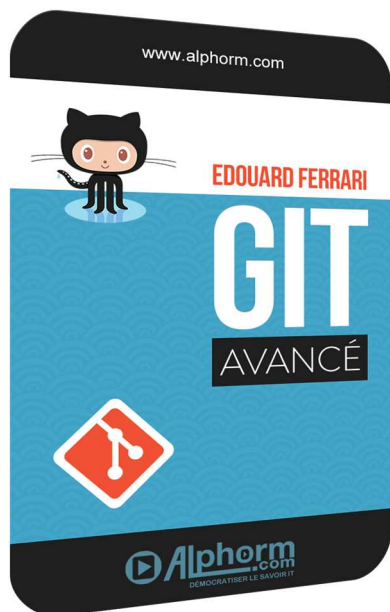
```
[imap]
  folder = "[Gmail]/Drafts"
  host = imaps://imap.gmail.com
  user = utilisateur@gmail.com
  pass = m0td3p4ss3
  port = 993
  sslverify = false
```

- Puis vous pouvez envoyer les patches dans votre répertoire 'Drafts' :
 - `cat *.patch | git imap-send`
- Vous retrouverez vos patches dans votre Gmail, dans le répertoire 'Drafts'. Il vous suffira d'ajouter le destinataire et d'envoyer l'email.

Ce qu'on a couvert

- Nous avons vu comment créer des patches et les envoyer par email.
- Prochaine vidéo :
 - Contribution à un projet : Appliquer des patches





Alphorm.com

Git distribué

Appliquer des patches

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com

Formation Git Avancé

alphorm.com™©

Plan

- Travail dans des branches thématiques
- Appliquer des patches
- Application d'un patch avec *apply*
- Application d'un patch avec *am*
- Vérification des branches distantes



Formation Git Avancé

alphorm.com™©

Travail dans des branches thématiques

- Quand vous vous apprêtez à intégrer des contributions, une bonne idée consiste à les essayer d'abord dans une branche thématique.
- Par exemple vous pouvez créer une branche thématique avec les initiaux du développeur suivi du nom de son travail :
 - `$> git checkout -b jp/js_client`

Appliquer des patches

- Il existe deux moyens d'appliquer un patch reçu par mail :
 - `$> git apply`
 - `$> git am`

Application d'un patch avec *apply*

- Si vous avez reçu le patch de quelqu'un qui l'a généré avec la commande **git diff** ou **diff** Unix, vous pouvez l'appliquer avec la commande **git apply**.
 - `$> git apply patch-js-client.patch`
- Les fichiers dans votre copie de travail sont modifiés.
- C'est quasiment identique à la commande **patch -p1** qui applique directement les patches.
- Vous pouvez aussi utiliser *git apply* pour voir si un patch s'applique proprement avant de réellement l'appliquer :
 - `$> git apply --check patch-js-client.patch`

Application d'un patch avec *am*

- Si le contributeur est un utilisateur de Git qui a été assez gentil d'utiliser la commande **format-patch** pour générer ses patches, le patch contient alors déjà l'information d'auteur et le message de validation :
 - `$> git am patch-js-client.patch`
- Il est possible que le patch ne s'applique pas :

```
$ git am 0001-seeing-if-this-helps-the-gem.patch
Application : seeing if this helps the gem
error: patch failed: ticgit.gemspec:1
error: ticgit.gemspec: patch does not apply
Le patch a échoué à 0001.
Lorsque vous aurez résolu ce problème, lancez "git am --continue".
Si vous préférez sauter ce patch, lancez "git am --skip" à la place.
Pour restaurer la branche d'origine et stopper le patchage, lancez
"git am --abort".
```

- Comme avec *git rebase*, on modifie le fichier, on *git add* et *git am --continue*



Vérification des branches distantes

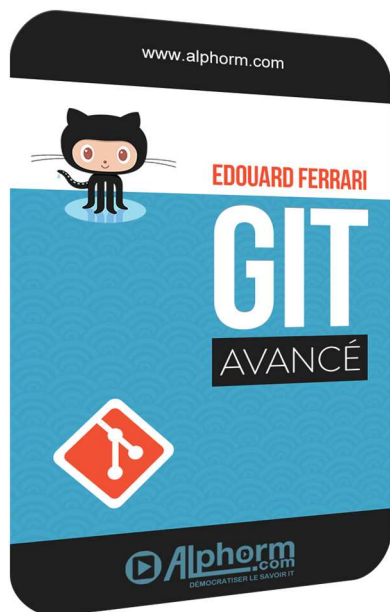
- Si la contribution à été fournie par un utilisateur de qui a mis en place son propre dépôt public, vous pouvez ajouter en tant que dépôt distant et réaliser les fusions localement.
 - `$ git remote add jessica git://github.com/jessica/monproject.git`
 - `$ git fetch jessica`
 - `$ git checkout -b jsclient jessica/js-client`



Ce qu'on a couvert

- Nous avons vu dans cette vidéo :
 - Comment appliquer des patches
 - Comment ajouter des branches distantes et ajouter leur modification
- Prochaine vidéo :
 - La conclusion





Alphorm.com

Conclusion

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : edouard.ferrari@gmail.com

Formation Git Avancé

alphorm.com™©

Ce qu'on a couvert

- Nous avons vu comment mettre en place un serveur GIT :
 - Par accès direct
 - Par SSH
 - Par protocole git
 - Par HTTP
- Différent serveur HTTP
 - GitWeb
 - GitLab

Formation Git Avancé

alphorm.com™©

Ce qu'on a couvert

- Nous avons différentes méthodes de travail avec Git
 - Pour des petits groupes de travail
 - Pour des entreprises
 - Pour des grands groupes de travail
- Nous avons aussi vu comment contribuer et faire partager son travail dans GIT

Avez-vous des Questions / Remarques / Commentaires ?





Keep in touch !

E-mail : edouard@ferrari.wf
Linkedin : <https://fr.linkedin.com/in/edouardferrari>
Twitter : https://twitter.com/edouard_ferrari
Alphorm : <http://www.alphorm.com/formateur/edouard-ferrari>

