

# Formation Git

## IX - Dépôt distant

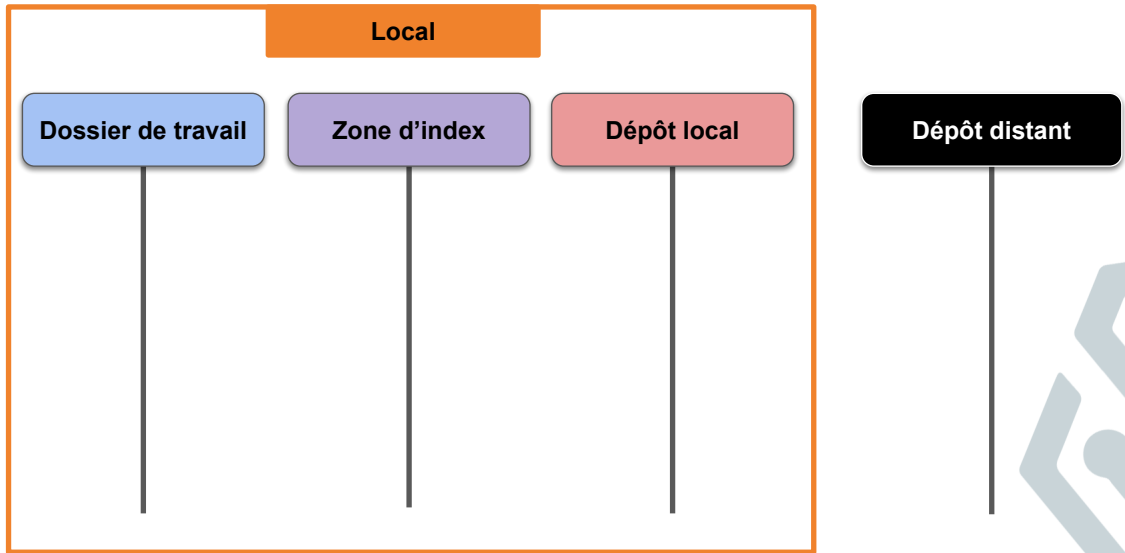


**Arnaud MERCIER**  
arnaud.mercier@hexotech.fr



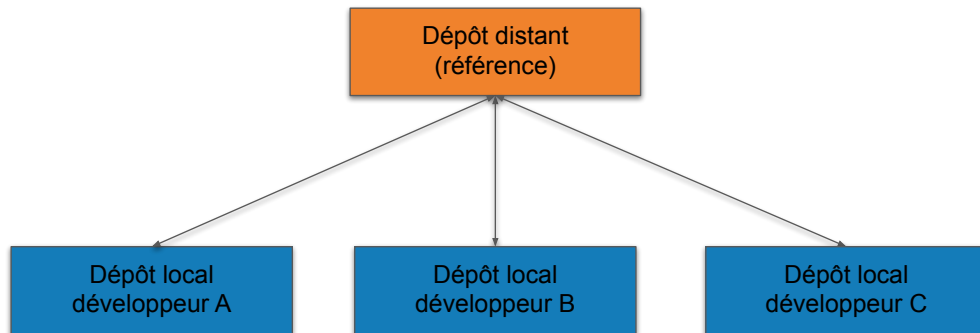
Nous allons voir dans ce chapitre comment travailler avec un serveur distant

## Dépôt distant



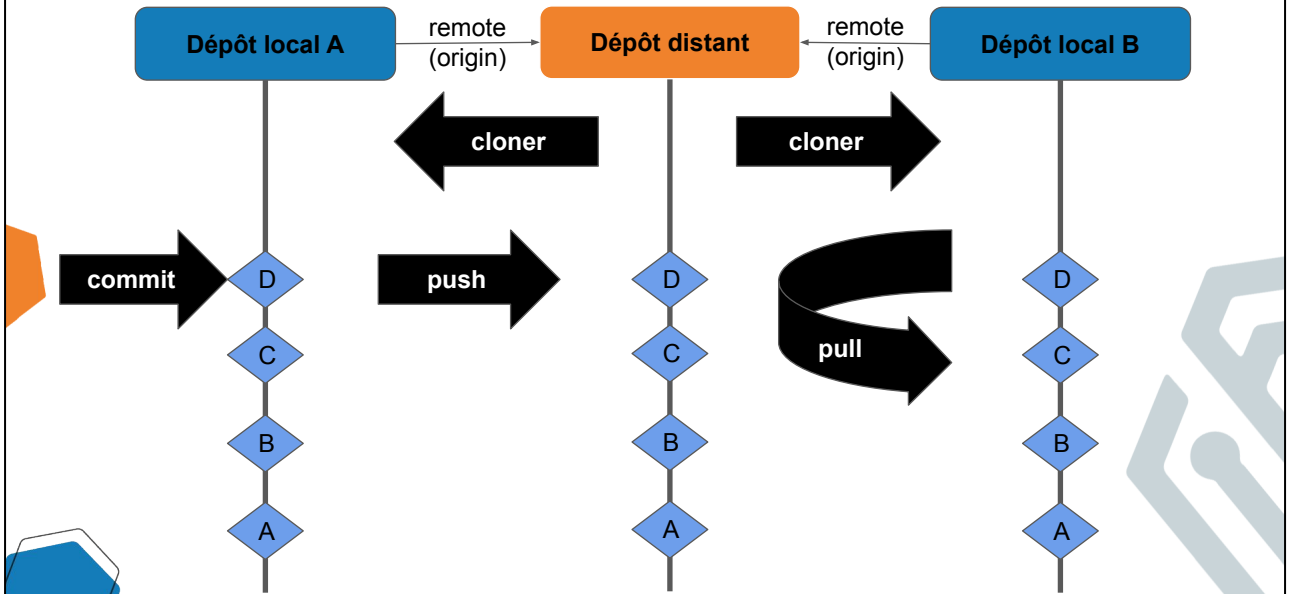
Nous avons pour le moment vu uniquement l'utilisation de git en local, nous allons à présent voir comment travailler avec un serveur distant

## Modèle distribué



Avec git, chaque développeur possède en local une copie complète du dépôt distant. Cela permet alors de travailler de manière indépendante même en cas de coupure réseau.

# Modèle distribué

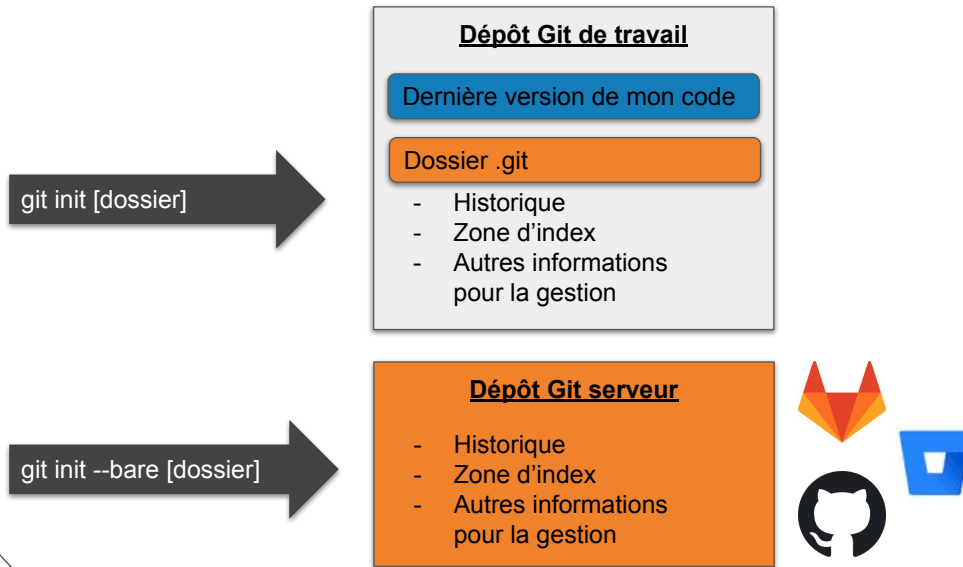


Voici un avance rapide sur le déroulement du travail de deux personnes sur un même dépôt via justement le dépôt distant et les copies en local:

- 1- les développeurs clones le dépôt en local
- 2- dev A fait un commit en local, puis pousse le commit sur le serveur
- 3- dev B se synchronise avec le serveur et récupère le commit de dev A.

INFO: le remote correspond à une référence vers un dépôt git distant (souvent l'URL du dépôt sur le serveur)

# Création d'un dépôt



La plupart des serveurs git, offrent une interface web qui rend les actions transparentes pour l'utilisateur. C'est le cas de github ou de gitlab par exemple. Mais au final, il faut bien comprendre que c'est un simple git qui est installé et utilisé sur ces serveurs.

Lorsque vous demandez au serveur la création d'un dépôt Git, celui-ci utilise la commande "git init" mais avec l'option "--bare", ce qui permet de créer un dépôt sans avoir de workspace (car personne ne vas développer sur le serveur). Il faut donc différencier 2 types de dépôts:

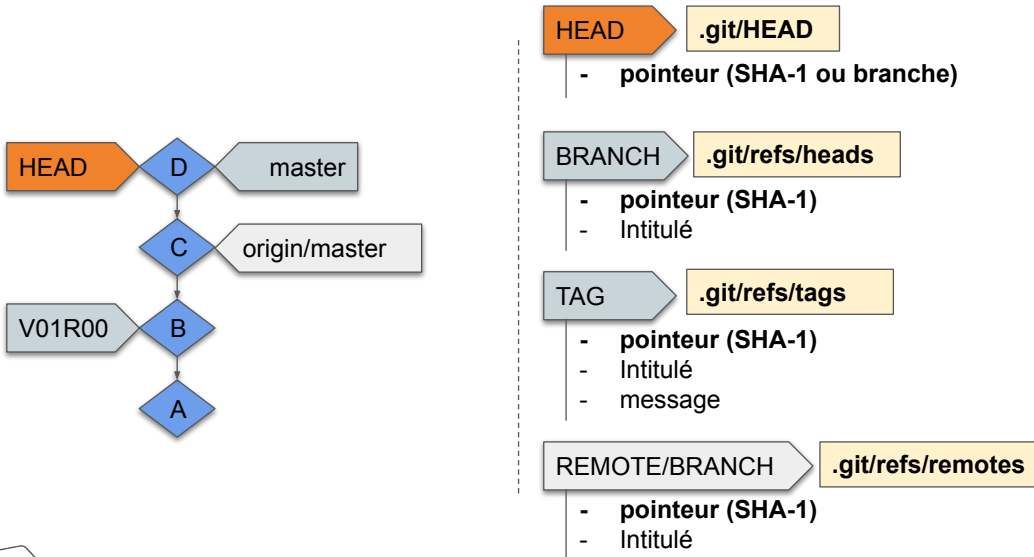
- Dépôt de travail (avec espace de travail)
- Dépôt de serveur (sans espace de travail)

INFO: le fichier HEAD d'un dépôt serveur, contient la référence de la branche principale.

## EXEMPLE:

- `git init --bare MON_SERVEUR`
- `cd MON_SERVEUR`
- `ls -la`
- `git status`

# Références et remotes



Les références de remote représentent la position des branches sur le remote en question. Par exemple, `origin/master` représente la position de `master` sur le remote `origin`.

Attention: cette référence de remote n'est pas une branche, nous ne pouvons pas l'activer. En revanche, elle reste une référence comme une autre.

# Cloner un dépôt distant



Pour cloner un dépôt, il faut utiliser la commande "git clone" en indiquant à la suite le chemins vers le dépôt visé puis le chemin en local ou vous souhaitez le cloner. le dernier paramètre est optionnel. En son absence, le dépôt sera cloné dans un dossier du même nom.

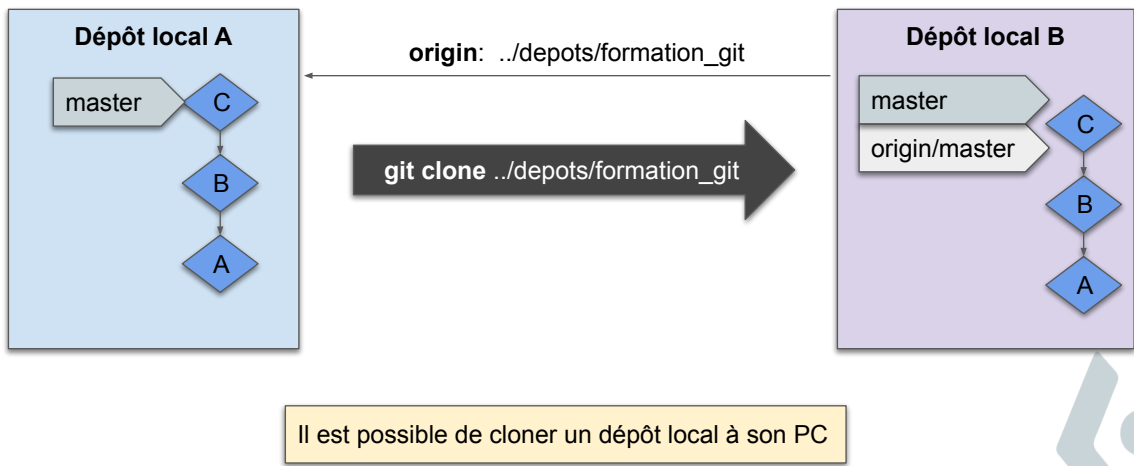
git clone permet de créer un dépôt de travail (équivalent à git init) et fait le lien avec le serveur à l'origine de l'historique (remote origin)

INFO: lors du clone, git crée en local la branche par défaut défini sur le remote (HEAD du remote)

## EXEMPLE:

- `git clone https://github.com/amgitformation/formation\_git`
- `cd formation_git`
- `git log --oneline`

# Cloner un dépôt local



Il est possible de cloner un dépôt local à son PC, qu'il soit "bare" ou non.

Attention: si vous clonez un dépôt de travail, il ne sera pas possible de pousser la branche qui est active sur sur remote.

## EXEMPLE:

- **git clone ./MON\_SERVEUR clone\_serveur**
- **cd clone\_serveur**
- **git status**



# Afficher les remotes



```
git remote -v  
origin https://github.com/formation_git (fetch)  
origin https://github.com/formation_git (push)
```

Les remotes se trouvent dans **.git/refs/remotes**

Les remotes sont des références vers d'autres dépôts. Cela permet alors de lier votre dépôt avec un autre.

Lors du clone d'un dépôt, un remote du nom de origin est automatiquement créé.

il est possible d'avoir plusieurs remotes.

la commande "git remote -v" affiche la liste des remotes de votre dépôt

**EXEMPLE (formation\_git):**

- **git remote -v**

# Inspecter un remote

## **git remote show origin**

```
Fetch URL: https://github.com/arnaudiko/mon_site_web.git
Push URL: https://github.com/arnaudiko/mon_site_web.git
HEAD branch: master
Remote branches:
  BRANCH_A_MERGE tracked
  master                tracked
Local branches configured for 'git pull':
  BRANCH_A_MERGE merges with remote BRANCH_A_MERGE
  master           merges with remote master
Local refs configured for 'git push':
  BRANCH_A_MERGE pushes to BRANCH_A_MERGE (local out of date)
  master          pushes to master      (fast-forwardable)
```

```
git remote show [nom_remote]
```

Pour afficher plus d'informations sur un remote, il est possible d'utiliser la commande "git remote show" suivi du nom du remote

# Ajouter un remote



```
git remote add [remote_name] [remote_url]
git fetch [remote_name]
```

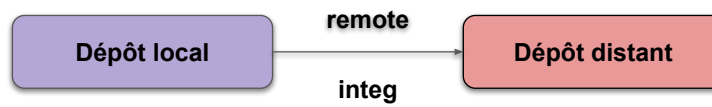
Pour ajouter un nouveau remote, il faut utiliser la commande “git remote add” suivi du nom que vous souhaitez donner au remote puis le chemin de celui-ci

Attention: faire un “git fetch” sur le remote fraîchement ajouté pour récupérer les informations le concernant.

## EXEMPLE (formation\_git):

- **git remote add TEST ../MON\_SERVEUR**

# Renommer un remote



```
git remote rename [prev_name] [new_name]
```

Il est également possible de renommer un remote en utilisant la commande "git remote rename" suivi du nom actuel du remote visé puis son nouveau nom

**EXEMPLE (formation\_git):**

- **git remote rename TEST SERV**

# Supprimer un remote



```
git remote rm [remote_name]
```

pour supprimer un remote, il faut utiliser la commande “git remote rm” suivi du nom du remote

**EXEMPLE (formation\_git):**

- **git remote rm SERV**

# Changer l'url d'un remote



```
git remote set-url [remote] [new_url]
```

Il est enfin possible de modifier l'url d'un remote en utilisant la commande "git remote set-url" suivi du nom actuel du remote visé puis sa nouvelle URL.

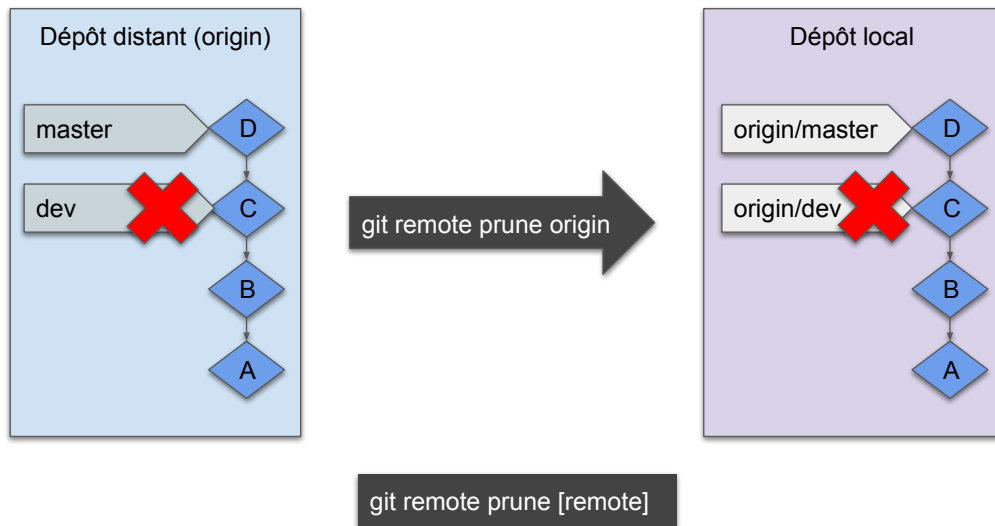
INFO: Cette commande peut être utilisé pour:

- Changer le mode de communication avec le serveur (https ou ssh)
- Appliquer la migration d'un serveur

**EXEMPLE (formation\_git):**

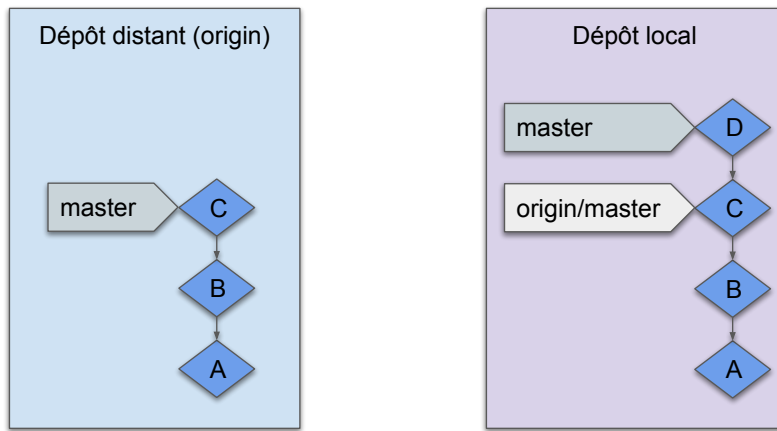
- **git remote set-url origin ../MON\_SERVEUR**
- **git fetch**

# Remotes prune



la commande permet de nettoyer les références de remote qui n'existent plus sur le serveur

# Modifications locales



Attention: les commits sont toujours fait en local. c'est la force même de Git, nous travaillons uniquement en local, puis une fois que tout est carré, on se synchronise avec le serveur.

Dans le cas ou nous ajoutons des commits en local, nous avons une avance sur l'historique du remote.

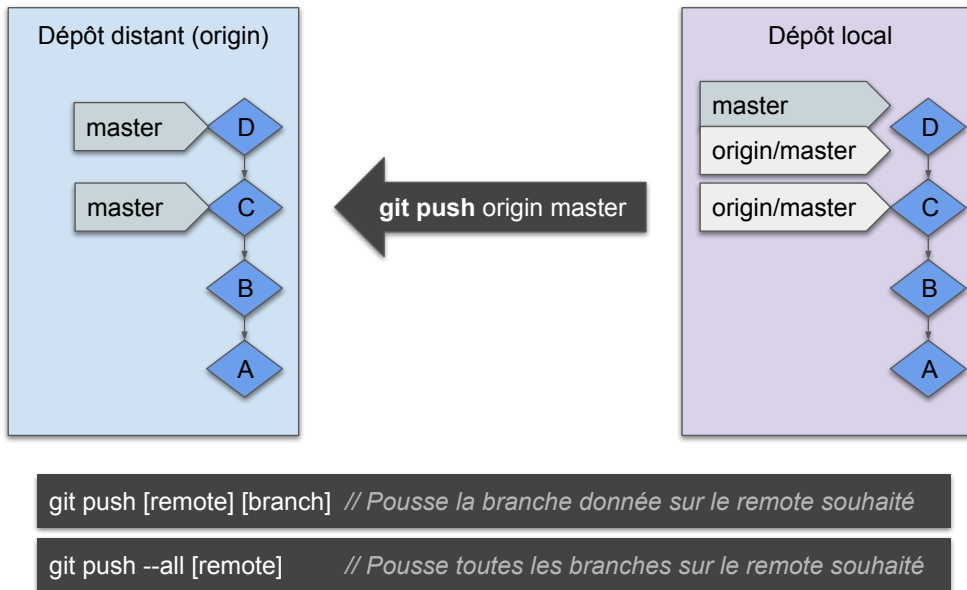
Cette avance est visible via le git status

## EXEMPLE (formation\_git):

- **modification dans hello.html**
- **git add .**
- **git commit**
- **git status**



# Push de branche



le push, est l'action de synchroniser de notre dépôt local avec le dépôt distant: envoyer nos nouvelles modifications (commit, tags, ...).

pour faire un push, il faut utiliser la commande "git push" suivi du nom du remote cible et enfin du nom de la référence a push.

ex: `git push origin master` (cela correspond à la demande "pousse les modifications de la branche local master vers le remote master")

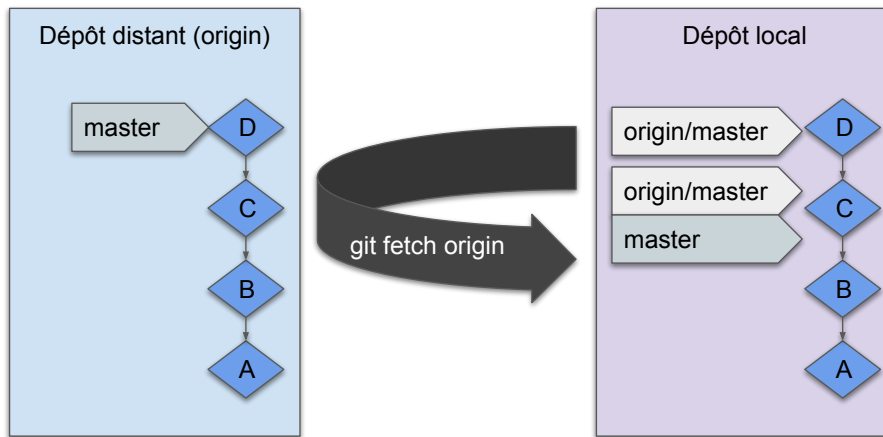
Attention: si le remote utilisé ne provient pas d'un "git clone" vous aurez peut être un message d'erreur qui vous indique qu'il ne sait pas vers quelle branche il doit faire le push. ajouter alors l'option "-u remote/branche"

ATTENTION: Il faut être en FastForward par rapport au serveur pour pouvoir push, sinon le remote refusera la commande

## EXEMPLE (formation\_git):

- **git push origin master**
- **git status**
- **git log --oneline**

# Fetch de remote



`git fetch [remote]` // Met à jour toutes les références distantes (branches et tags)

La commande fetch permet de récupérer les modifications présente sur le serveur que vous n'avez pas encore sur votre copie en local.

Attention, cette commande ne vas pas appliquer dans votre workspace les modifications récupérés. vous allez rester sur le commit courant.

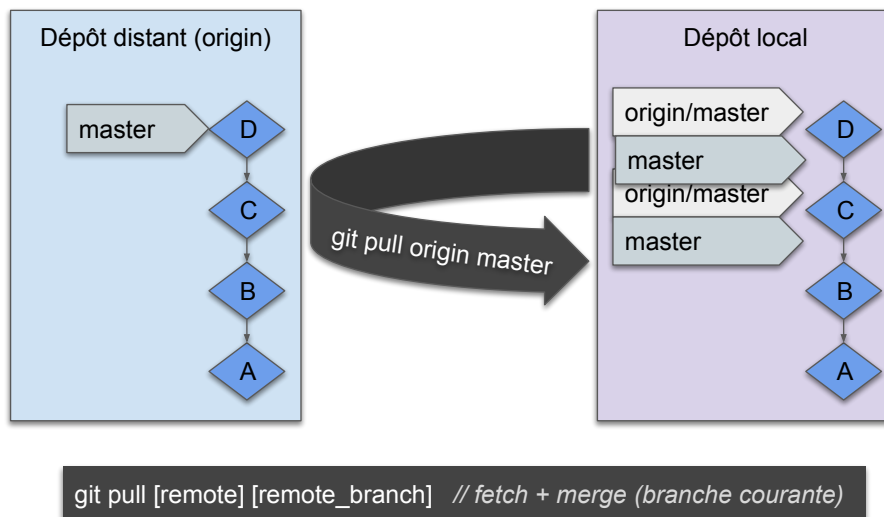
Pour appliquer les modifications sur votre espace de travail, il faut ramener le master en local sur le même commit que le master distant

exemple de la branche master et de la remote origin: "git merge origin/master

## EXEMPLE (clone\_serveur):

- `git fetch`
- `git status`
- `git log --oneline origin/master`

# Pull de branche



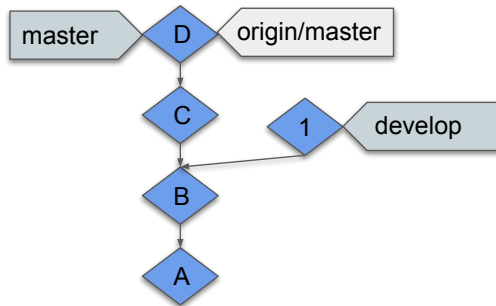
La commande pull est légèrement différente de la commande fetch. en fait le pull réalise les actions suivantes:

- 1- il fait un fetch
- 2- il fait le merge de la branche courante avec son image distante

## EXEMPLE (clone\_serveur):

- **git pull**
- **git status**

# Lister les branches locales et remotes



```
git branch -a
* master
develop
remotes/origin/master
```

```
git branch -vv
* master c57cc1c [origin/master] D
develop f65214b 1
```

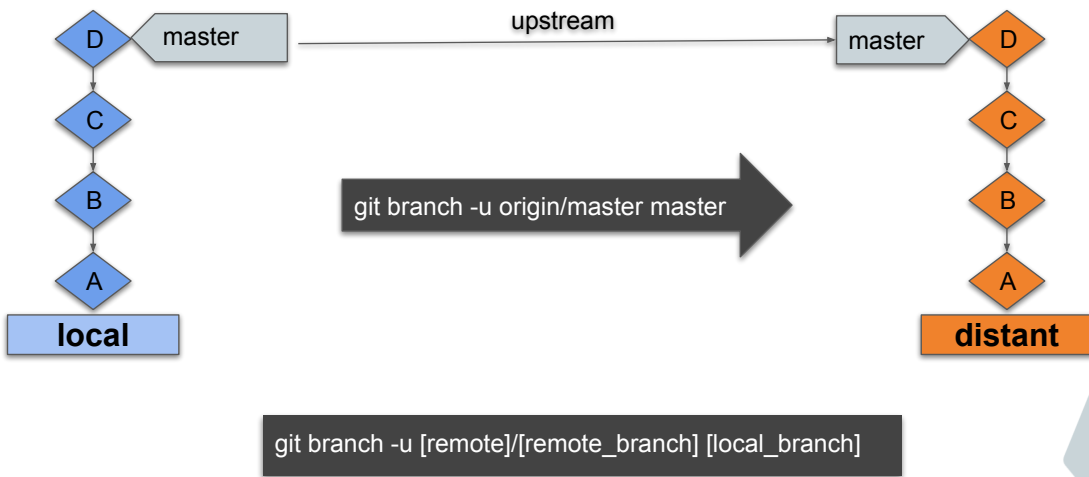
```
git branch -a // affiche les branches locales et distante
git branch -vv // affiche les informations de liens avec le remote
```

Pour lister les branches du dépôt local, il faut utiliser la commande “git branch”  
pour lister en plus les branches des remotes, il faut ajouter l’option -a à la commande  
“git branch”  
l’option -vv permet d’avoir plus de détail, notamment si la branche locale a une image  
distante (ex: master et origin/master)

## EXEMPLE (clone\_serveur):

- **git branch -a**
- **git branch -vv**

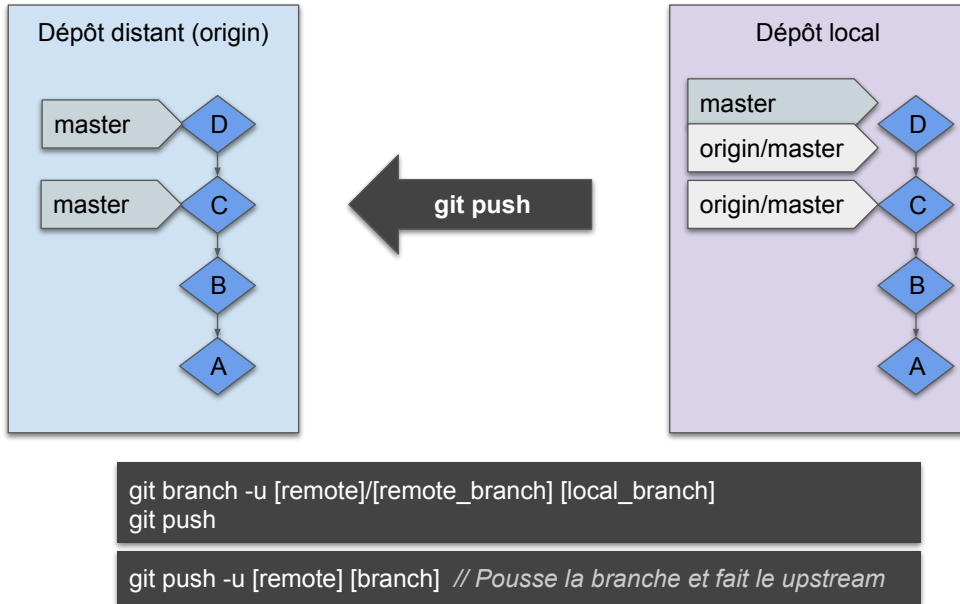
## Remote et branches soeurs



Le fait qu'une branche local soit en lien avec la branche du remote du même nom n'est pas obligatoire. Il faut réaliser ce lien de manière explicite via la commande "git branch -u"

INFO: lors du clone, git crée une branche de travail qui correspond à la branche par défaut du remote et fait automatiquement le lien entre les deux.

# Remote et branche

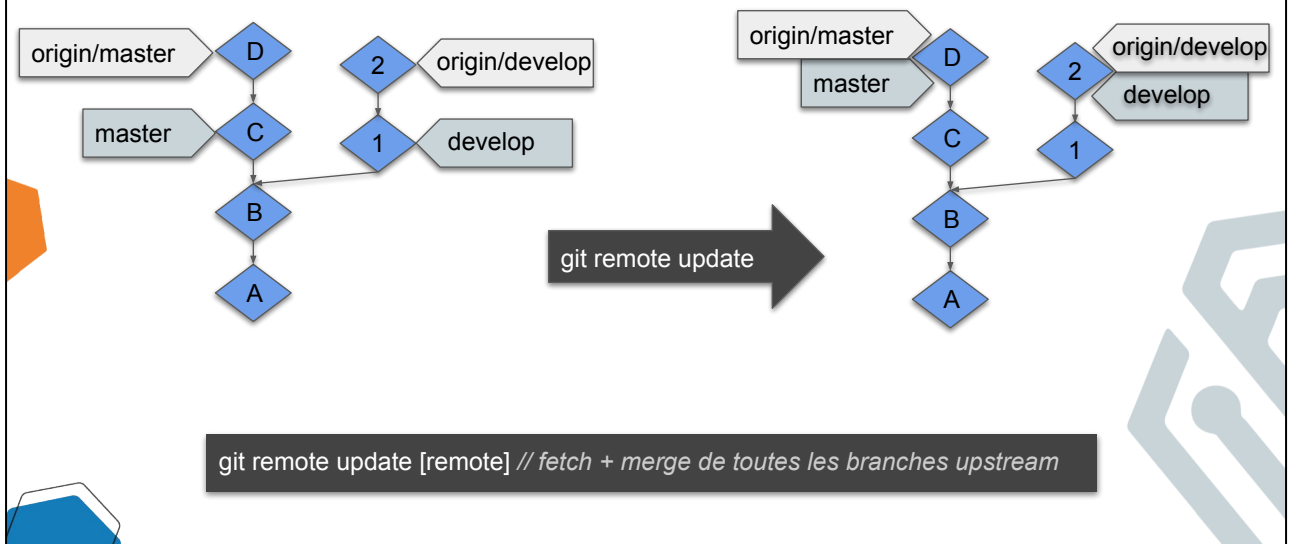


Lorsque vous avez établi le lien entre une branche locale et distante, vous n'avez plus besoin d'indiquer le remote et le nom de la branche distante lors d'un push. Git utilise alors l'upstream pour savoir où pousser votre branche courante

## EXEMPLE (clone\_serveur):

- faire une modification dans `hello.html`
- `git commit -am "test"`
- `git push`

# Remotes update

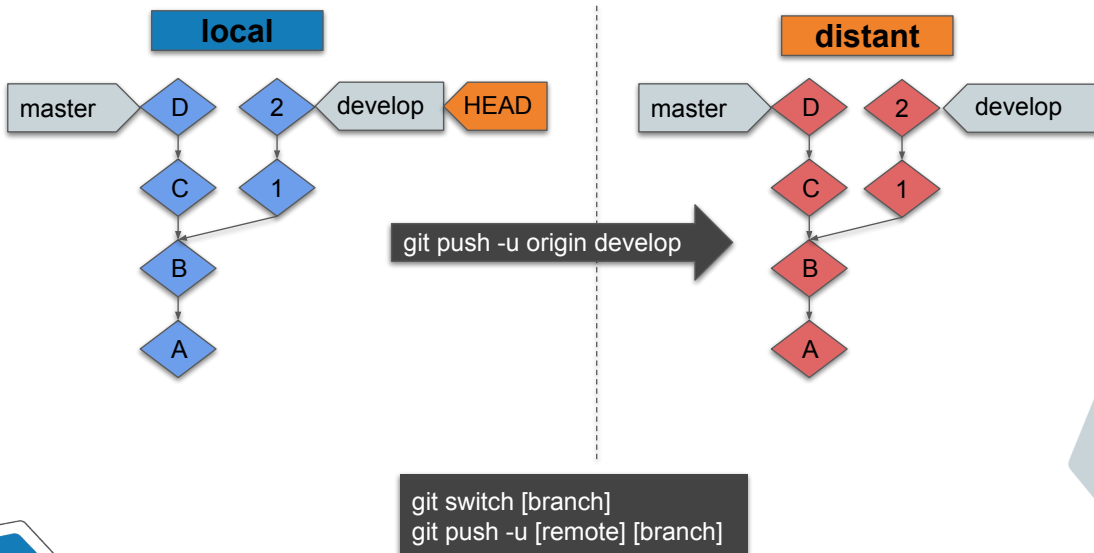


La commande “git remote update” permet de faire un fetch + un merge des branches qui ont un upstream

## EXEMPLE (formation\_git):

- **git fetch**
- **git status**
- **git remote update origin**

# Pousser une nouvelle branche



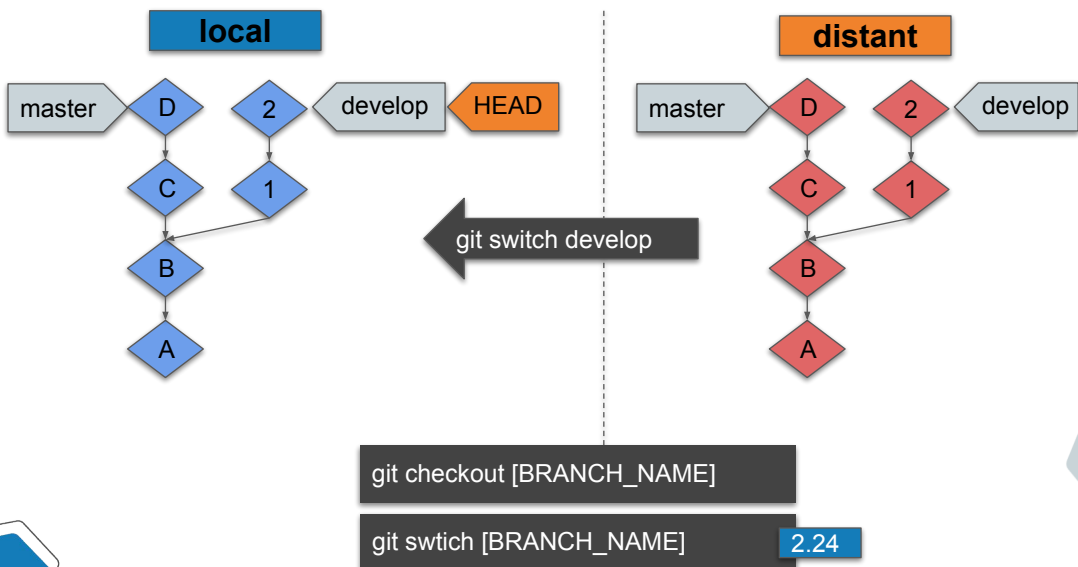
Pour pousser une branche locale qui n'existe pas encore sur le serveur (pas de upstream), il faut faire un push avec l'option -u pour faire le lien. Dans le cas contraire il faut indiquer au push le nom du remote et de la branche a chaque fois.

## EXEMPLE (formation\_git):

- `git switch -c develop`
- `faire un commit`
- `git push`
- `git push -u origin develop`
- `git branch -vv`
- `git log --oneline`



# Récupérer une branche distante

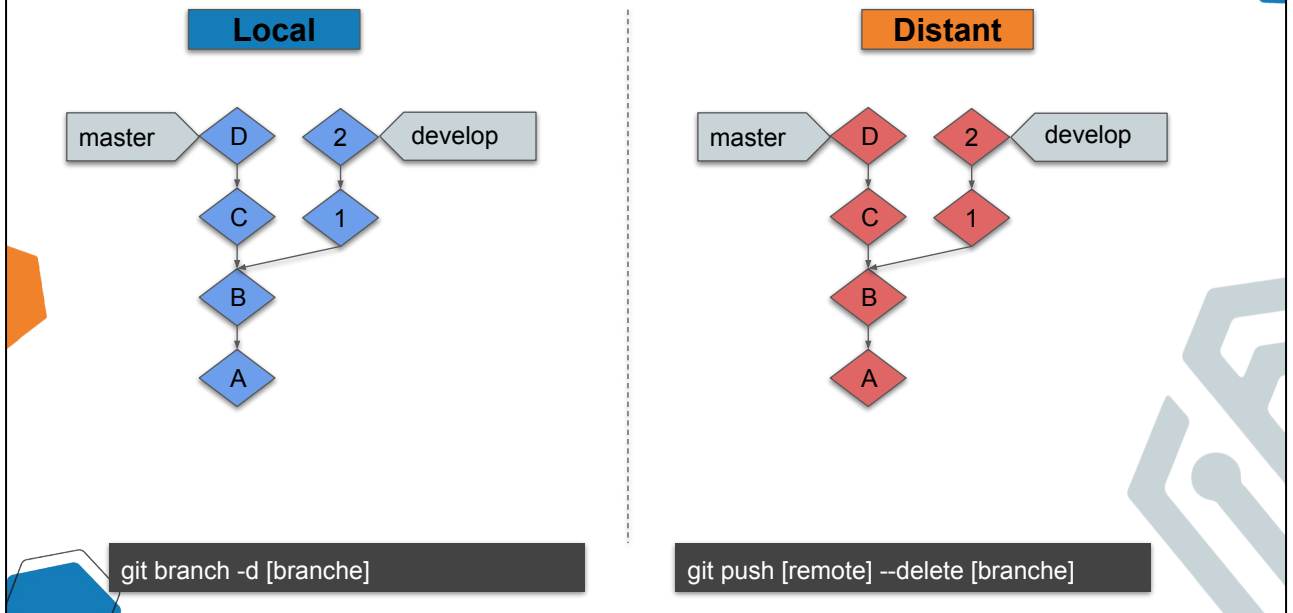


Pour créer une branche de travail locale en lien avec une branche distante, vous pouvez simplement faire un `git switch` de la branche. Si elle n'existe pas en local, git va la créer et faire le lien automatiquement

## EXEMPLE (clone\_serveur):

- `git fetch`
- `gitk master origin/develop`
- `git switch develop`
- `git branch -vv`

# Supprimer une branche distante



Il est possible de supprimer une branche en local via la commande “git branch -d” suivi du nom de la branche.

Pour les branches distante, il faut utiliser la commande “git push [remote] -d” suivi du nom de la branche

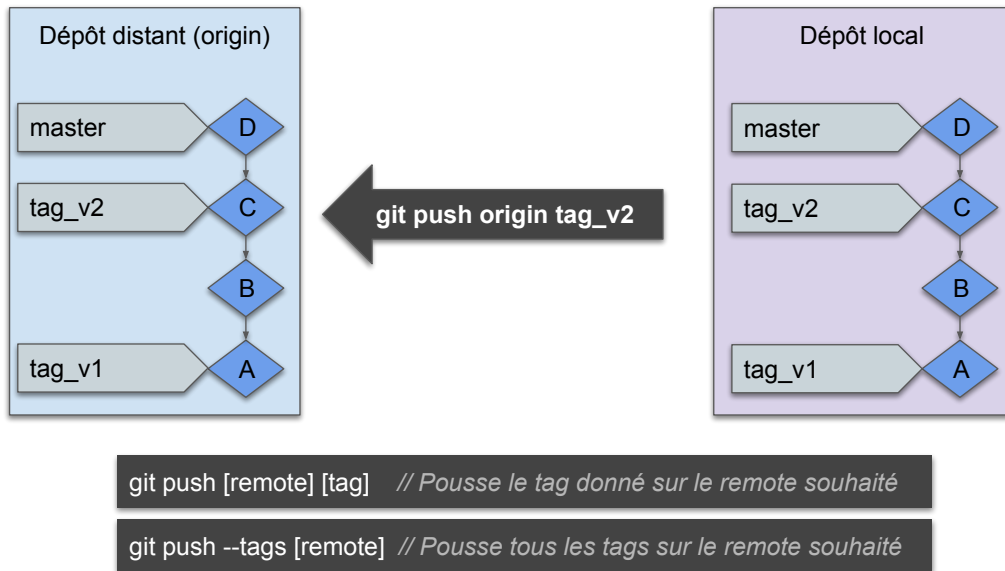
## EXEMPLE (clone\_serveur):

- **git fetch**
- **git push -d develop**
- **git branch -a**

## EXEMPLE (formation\_git):

- **git fetch**
- **git switch master**
- **git branch -D develop**
- **git remote prune origin**
- **git branch -a**

# Pousser un tag

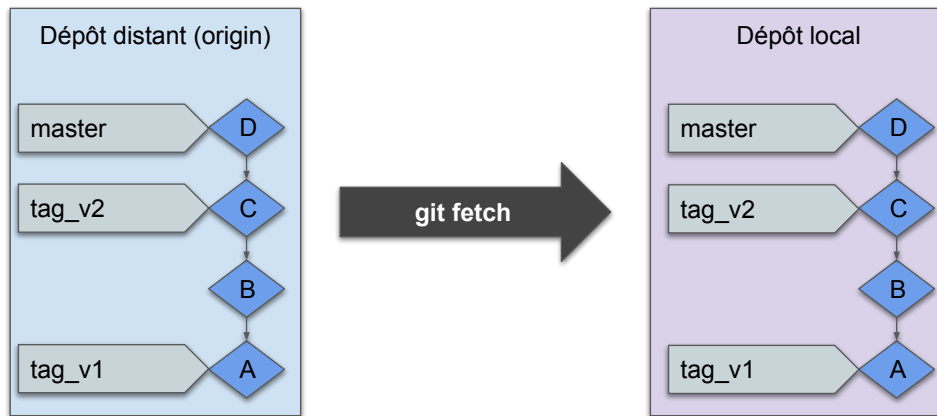


Si vous ajoutez un tag en local et que vous réalisez des push, vous allez voir que les tags n'apparaissent pas sur le dépôt distant. Il faut en effet pousser les tags via une autre commande: `git push [remote] [tag_name]`.

## EXEMPLE (formation\_git):

- **git tag TEST**
- **git push origin TEST**

# Récupérer les tags depuis le serveur

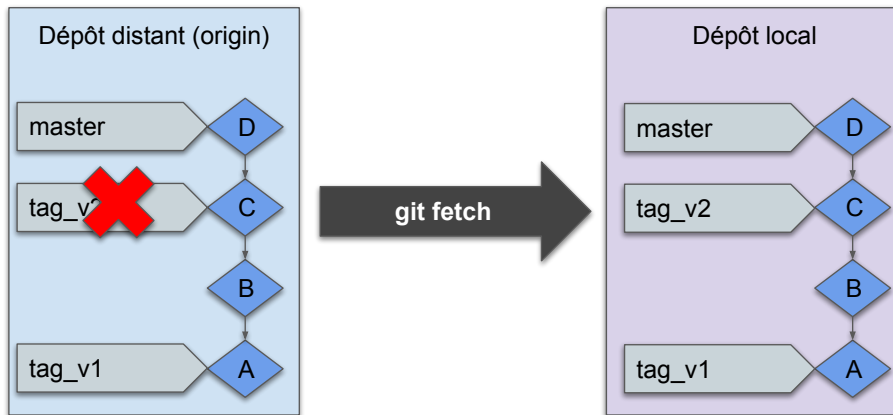


Lors qu'un `git fetch`, les références de tag du serveur sont récupérées en local

## EXEMPLE (`clone_serveur`):

- `git fetch`
- `git tag`

## Récupérer les tags depuis le serveur



Git ne supprime pas les références lors d'un fetch

Lors qu'un git fetch, si un tag n'existe pas sur le serveur mais existe en local, git ne le supprimera pas localement

# Supprimer un tag sur le serveur



```
git tag -d/D [tag_name] // Supprime le tag local
git push --delete [remote_name] [tag_name] // Supprime le tag sur le remote
```

Pour supprimer un tag sur le serveur distant, il faut utiliser la commande “git push --d [remote] [tag]”

Attention: si vous souhaitez supprimer le tag en local et distant, il faut alors utiliser les deux commandes associées

## EXEMPLE (clone\_serveur):

- **git push -d origin TEST**
- **git tag**
- **git tag -d TEST**

## EXEMPLE (formation\_git):

- **git fetch**
- **git tags**
- **git tag -d TEST**

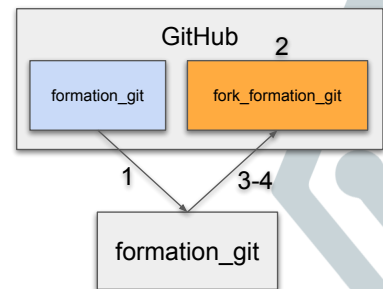
# TP 9.1 (GitHub)

## Scénario

Nous allons faire un fork d'un dépôt pour avoir les accès en lecture/écriture sur notre nouveau dépôt mais avec l'historique déjà présent

## Objectifs

- 1- Cloner, en local, le dépôt [https://github.com/amgitformation/formation\\_git](https://github.com/amgitformation/formation_git)
- 2- Créez un dépôt vide (sans readme !) sur GitHub du nom de "fork\_formation\_git"
- 3- Modifier le remote origin pour qu'il pointe sur votre dépôt "fork\_formation\_git"
- 4- Pousser l'historique et les tags sur ce nouveau remote



## SOLUTION:

- 1- on clone en local le dépôt:
  - git clone [https://github.com/amgitformation/formation\\_git](https://github.com/amgitformation/formation_git)
  - cd formation\_git
- 2- Allez sur <https://github.com> pour créer un dépôt vide
- 3- On modifie origin pour pointer sur notre dépôt serveur
  - git remote set-url origin [url\_depot]
  - git fetch
- 3- On envoie la branche et les tags sur notre serveur:
  - git push --all
  - git push --tags

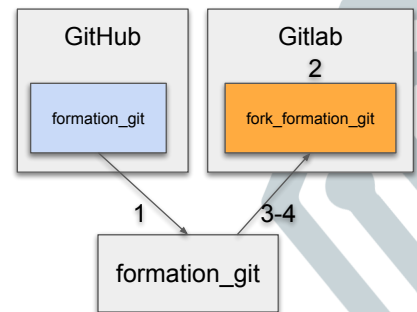
# TP 9.1 (Gitlab)

## Scénario

Nous allons faire un fork d'un dépôt pour avoir les accès en lecture/écriture sur notre nouveau dépôt mais avec l'historique déjà présent

## Objectifs

- 1- Cloner, en local, le dépôt [https://github.com/amgitformation/formation\\_git](https://github.com/amgitformation/formation_git)
- 2- Créez un dépôt vide (sans readme !) sur Gitlab du nom de "fork\_formation\_git"
- 3- Modifier le remote origin pour qu'il pointe sur votre dépôt "fork\_formation\_git"
- 4- Pousser l'historique et les tags sur ce nouveau remote



## SOLUTION:

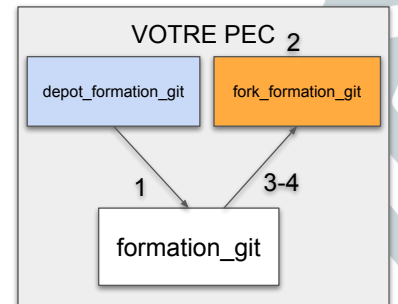
- 1- on clone en local le dépôt:
  - git clone [https://github.com/amgitformation/formation\\_git](https://github.com/amgitformation/formation_git)
  - cd formation\_git
- 2- Allez sur <https://gitlab.com> pour créer un dépôt vide
- 3- On modifie origin pour pointer sur notre dépôt serveur
  - git remote set-url origin [url\_depot]
  - git fetch
- 3- On envoie la branche et les tags sur notre serveur:
  - git push --all
  - git push --tags



# TP 9.1 (Local)

## Objectifs

- 1- Créer un dossier SERVER et y placer le dossier depot\_formation\_git qui se trouve dans depot\_formation\_git.zip (pensez à le décompresser)
- 2- Toujours dans le dossier SERVER, créer un dépôt "bare" du nom de "fork\_formation\_git" (il sera vide)
- 3- Cloner, en local, le depot\_formation\_git qui se trouve dans depot\_formation\_git.zip (pensez à le décompresser)
- 4- Modifier le remote origin pour qu'il pointe sur votre dépôt "fork\_formation\_git"
- 5- Pousser l'historique et les tags sur ce nouveau remote



## **SOLUTION:**

- 1- Créer le dossier et y placer le dépôt bare depot\_formation\_git
- 2- Créer un dépôt –bare:
  - git init –bare fork\_formation\_git
  - lancer gitk dans ce repo pour voir l'évolution
- 3- On clone le dépôt –bare fournis dans le TP:
  - cd ..
  - git clone ../SERVER/depot\_formation\_git formation\_git
  - cd formation\_git
- 4- On modifie origin pour pointer sur notre dépôt serveur
  - git remote set-url origin ../SERVER/fork\_formation\_git
  - git fetch
- 5- On envoie la branche et les tags sur notre serveur:
  - git push –all
  - git push --tags

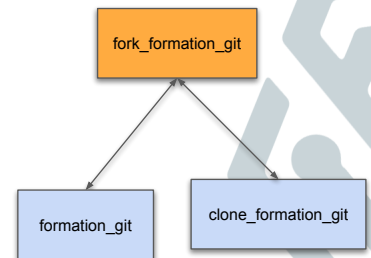
## TP 9.2

### Scénario

Simuler une deuxième personne dans votre équipe et réaliser une modification que vous allez récupérer par la suite

### Objectifs

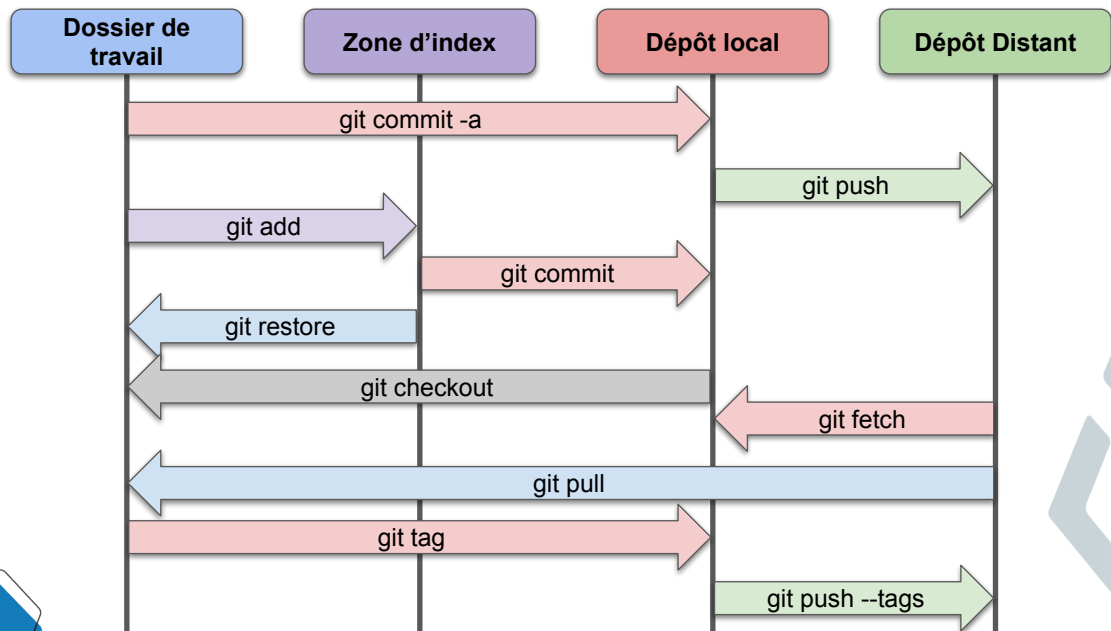
- 1- Cloner une 2ème fois le dépôt "fork\_formation\_git". En plus de la copie local du nom de "formation\_git", vous allez avoir un dépôt de travail du nom de "clone\_formation\_git".
- 2- Dans le dépôt "clone\_formation\_git", changer localement le nom et l'email de l'utilisateur. L'objectif est d'avoir des identifiants différentes de ceux globaux.
- 3- Dans le dépôt "formation\_git", faire un commit et faire le push
- 4- Ajouter le tag LOT1 et faire son push
- 5- Dans le dépôt "clone\_formation\_git", récupérez les modifications réalisées précédemment. Vérifiez que les modifications sont bien présentes (commit et tag)



### **SOLUTION:**

- 1- git clone [url\_depot] clone\_formation\_git
- 2- git config user.name et user.email
- 3- git commit -am "message"; git push
- 4- git tag et git push
- 5- git pull

# Bilan



Nous avons vu dans ce chapitre comment travailler avec un serveur distant.

On commence par cloner le dépôt pour avoir une version de travail en local

Puis on réalise des commits et des push de ces derniers

Enfin on récupère les modifications présentes sur le serveur