

**AP**

Eclipse-Git-4 : Opérations courantes lors d'un partage de projet avec Eclipse / Egit / GitHub

- 1- Le scénario global et les problèmes à gérer
- 2- Le chef de projet ajoute un nouveau fichier (commit & push)
- 3- Le collaborateur récupère le nouveau fichier (pull)
- 4- Le collaborateur modifie le nouveau fichier (commit & push)
- 5- Le chef de projet récupère le fichier modifié (pull)
- 6- Et maintenant, ladies and gentlemen, le CONFLIT en modification
- 7- Le chef de projet supprime le fichier (commit & push)
- 8- Le collaborateur constate la suppression du fichier (pull)
- 9- Quelques conseils pratiques

En pré-requis, on suppose que :

- chaque membre du groupe a créé un compte **delasalle-sio-xxxx-x** (où **xxxx-x** est le nom du membre) sur le site de **GitHub**
- un des membres (nous l'appellerons "**chef de projet**") a créé un dépôt nommé **tracegps** sur **GitHub**, et a désigné le ou les autres membres comme collaborateurs autorisés à modifier ce dépôt.

Dans ce document, le chef de projet s'appelle **yves-zenels** et le collaborateur s'appelle **amedee-bogueur**, mais vous remplacerez bien sûr ces noms par vos propres comptes.

1- Le scénario global et les problèmes à gérer

1-1 L'équipe projet

L'équipe de développement est composée de 2 personnes :



Yves ZENELS (chef de projet)
email : **yves.zenels@gmail.com**
compte GitHub : **yves-zenels**



Amédée BOGUEUR (développeur)
email : **amedee.bogueur@gmail.com**
compte GitHub : **amedee-bogueur**

**TP : vous remplacerez ces 2 acteurs
par les membres de votre équipe**

1-2 Les outils utilisés

Les membres de l'équipe utilisent tous l'EDI (Environnement de Développement Intégré) **Eclipse** muni du **plugin EGit** pour travailler avec le logiciel de gestion de versions **Git**.

Le partage du projet se fait avec la forge logicielle **GitHub** et le système de gestion de versions **Git**.

1-3 Le scénario présenté dans la suite de ce document

1. Le chef de projet crée et ajoute un nouveau fichier **test.php** à son projet local, il le **commit** sur le dépôt local et le **push** sur le dépôt distant
2. Le collaborateur récupère le fichier **test.php (pull)** dans son projet local
3. Le collaborateur modifie le fichier **test.php**, il le **commit** sur son dépôt local et le **push** sur le dépôt distant
4. Le chef de projet récupère le fichier **test.php (pull)** dans son projet local

A ce stade d'avancement, le dépôt distant et les 2 dépôts locaux possèdent la même version du fichier **test.php**

Maintenant, les 2 membres vont modifier simultanément le même fichier **test.php**, ce qui va provoquer une situation de conflit :

5. Le chef de projet modifie son fichier **test.php**, il le **commit** sur son dépôt local mais il ne le **push** pas tout de suite sur le dépôt distant
6. Le collaborateur modifie son fichier **test.php**, il le **commit** sur le dépôt local et le **push** sur le dépôt distant ; pas de problème pour l'instant
7. Le chef de projet veut maintenant faire le **push** qu'il n'avait pas encore effectué, et il essuie un refus car il est parti d'une ancienne version du fichier et risque d'écraser les modifications réalisées par son collaborateur
8. Le chef de projet doit résoudre le problème en récupérant la dernière version du fichier (**pull**) qui va être fusionnée avec sa version locale ; puis il devra décider du contenu à conserver et effectuer ensuite un **commit** et un **push**

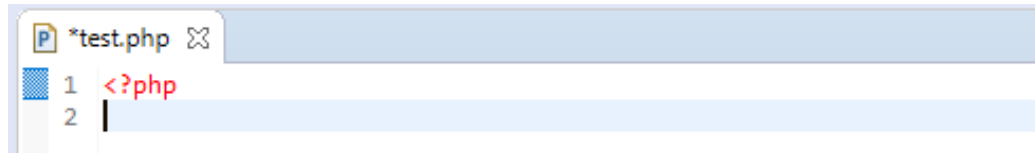
Maintenant, le chef de projet veut supprimer le fichier **test.php**

9. Le chef de projet supprime le fichier (**commit & push**)
10. Le collaborateur constate la suppression du fichier (**pull**)

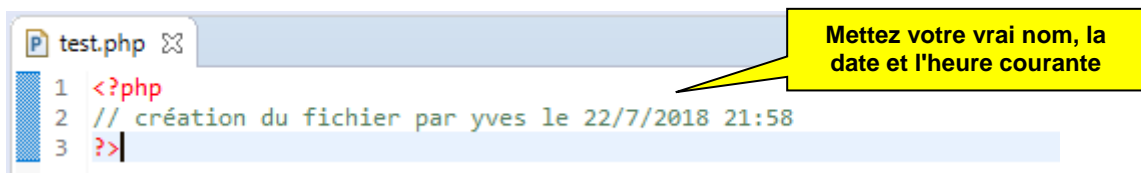
2- Le chef de projet ajoute un nouveau fichier (commit & push)

Vous êtes sur le poste de travail du chef de projet (**Yves Zenels**) et vous voulez ajouter un nouveau fichier **test.php** à la racine du projet **tracegps**.

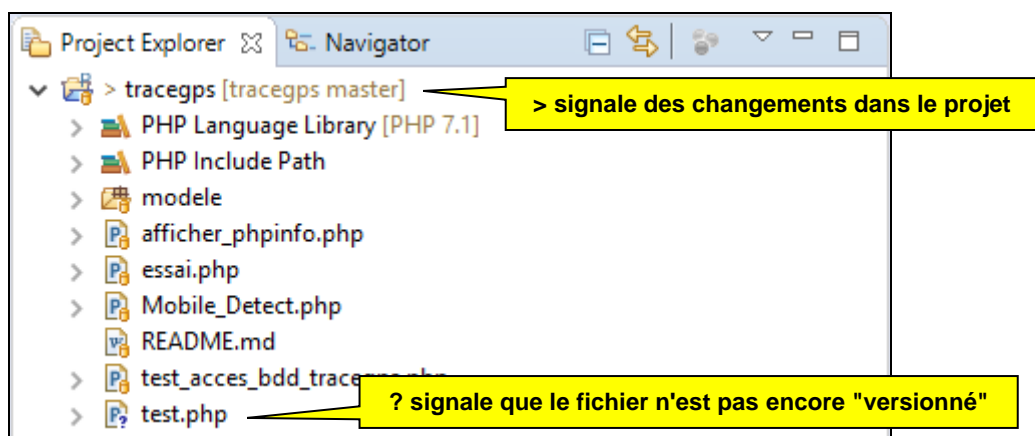
Faites un clic droit sur la racine du projet, choisissez la commande **New / PHP File** et nommez le fichier **test.php** ; après validation, vous devez obtenir :



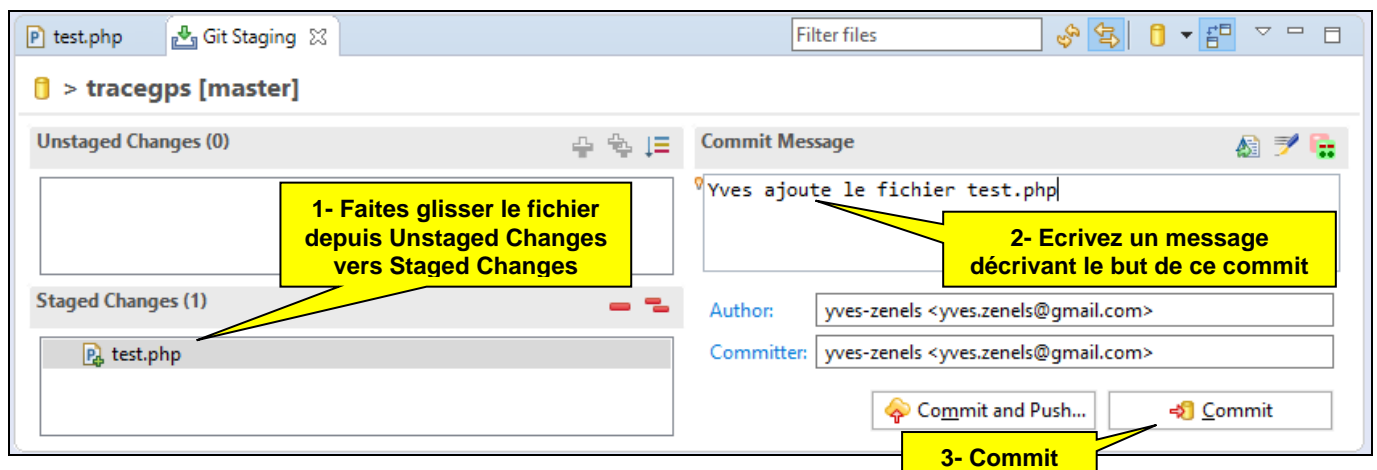
Complétez ce fichier par un simple commentaire, ce qui suffira pour nos expériences, et enregistrez-le :



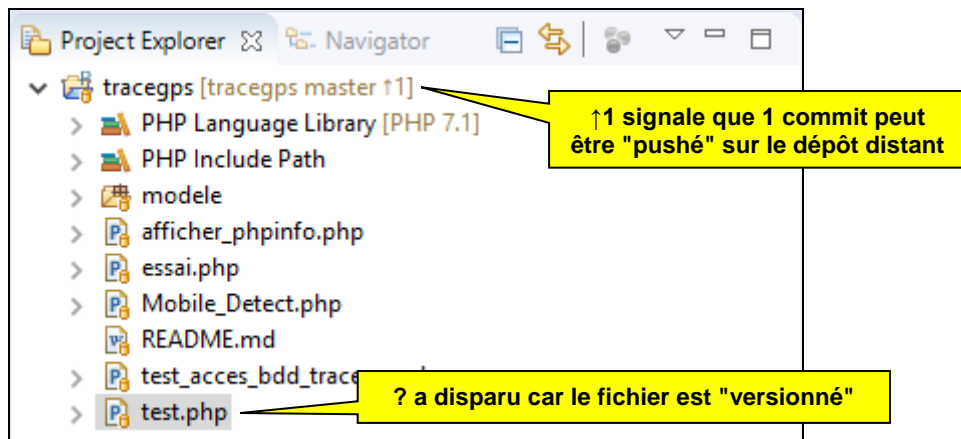
Observez la vue **Project Explorer** :



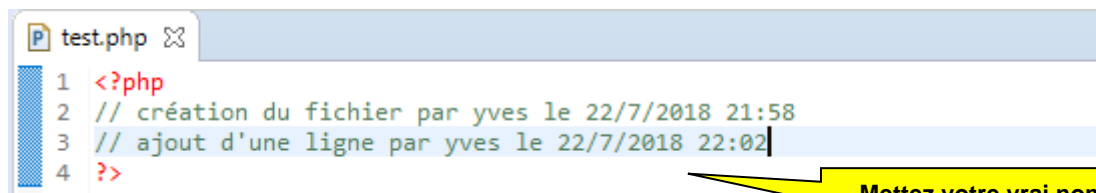
Vous estimez que ce fichier est suffisamment avancé (on est en expérimentation) pour effectuer un "**commit**" local qui permettra à Git de le "versionner" ; faites un clic droit sur le projet et choisissez la commande **Team / Commit...** :



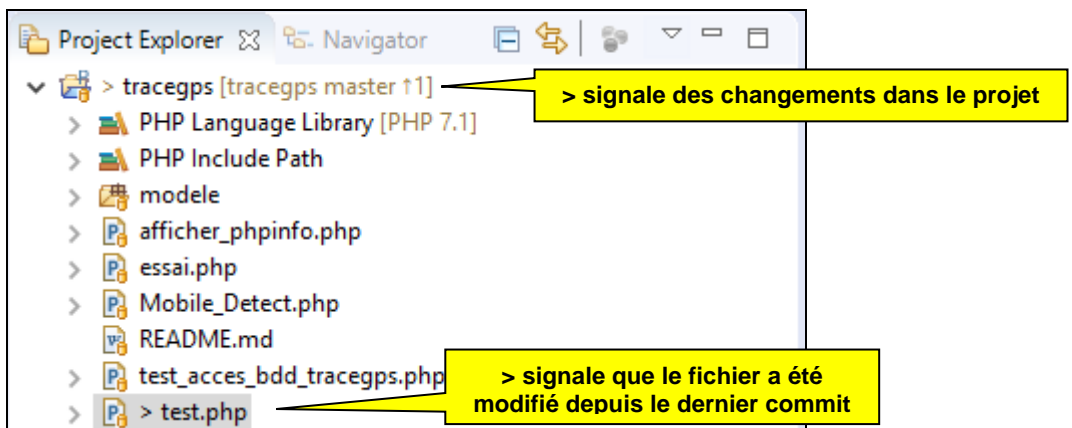
Observez la vue **Project Explorer** :



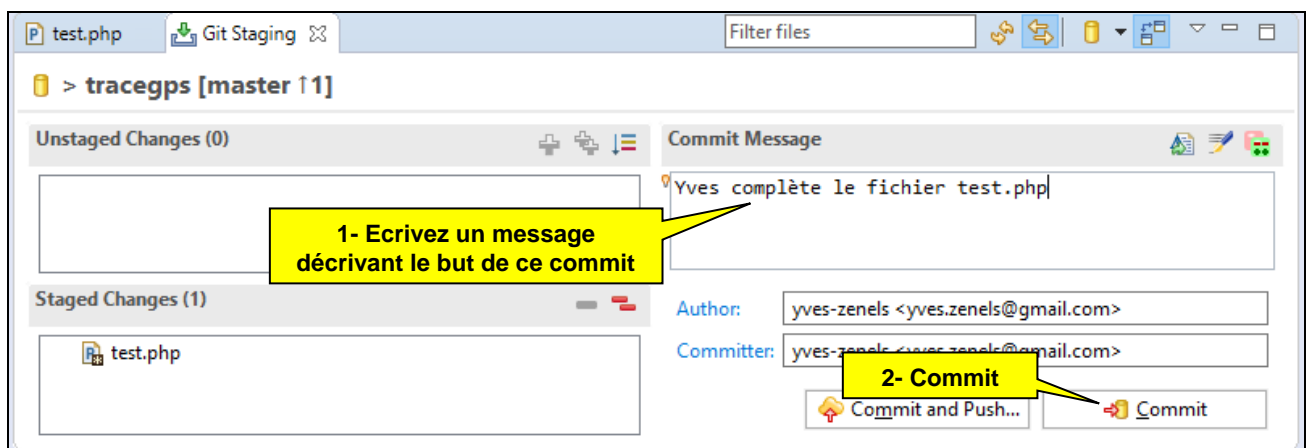
Complétez ce fichier par un autre commentaire (la ligne 3), et enregistrez-le :



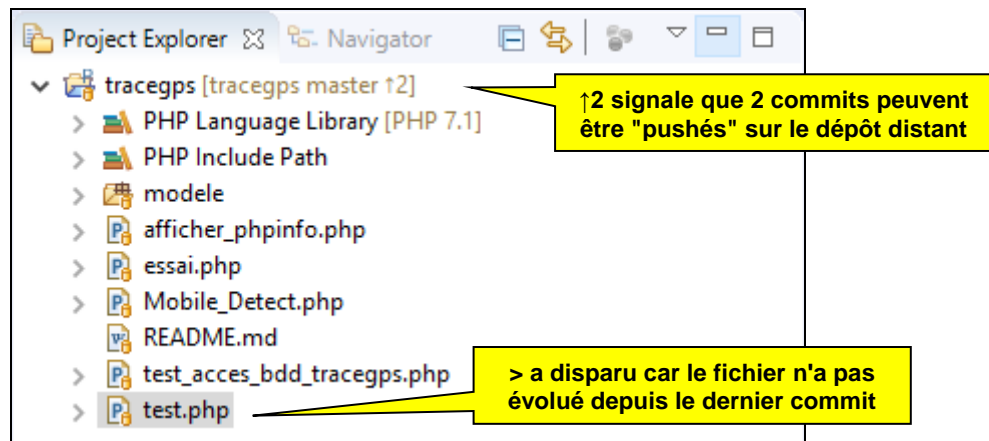
Observez la vue **Project Explorer** :



Vous estimez que ce fichier a bien avancé (on est toujours en expérimentation) pour effectuer un 2^{ème} "commit" local ; faites un clic droit sur le projet et choisissez la commande **Team / Commit...** :

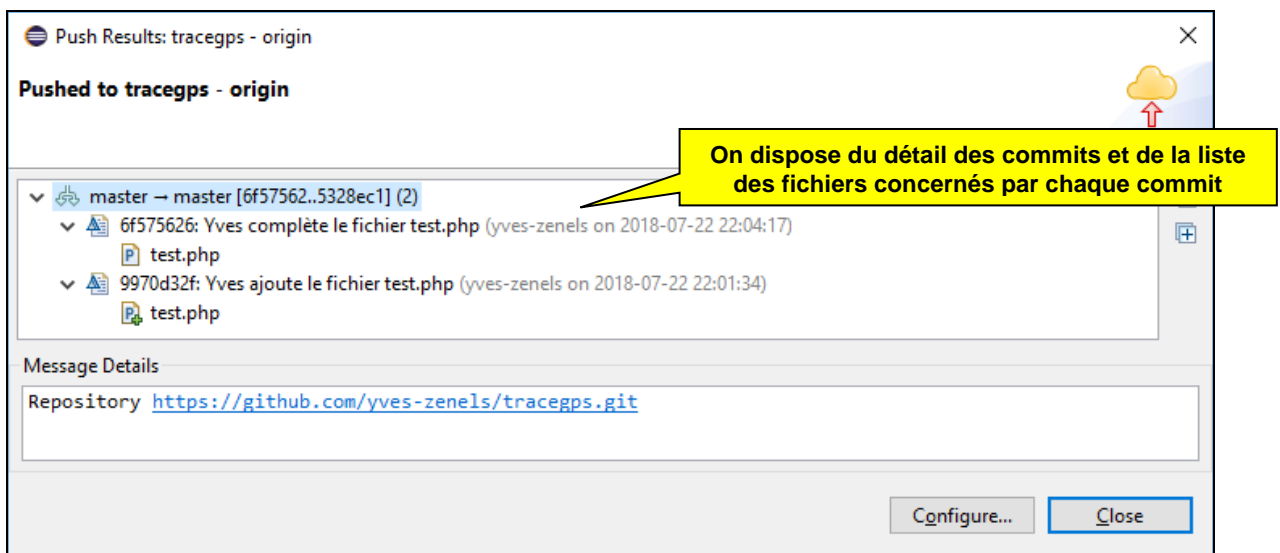


Observez la vue **Project Explorer** :

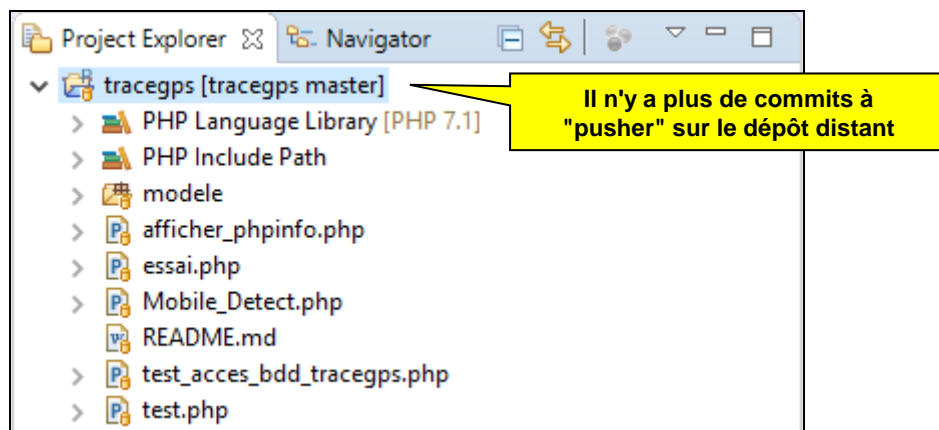


Après quelques commits en local (2 suffiront ici), vous estimez qu'il est temps de les copier sur le dépôt distant ; faites un clic droit sur le projet et choisissez la commande **Team / Push to Upstream**.

Si tout se passe bien, les mises à jour (donc le nouveau fichier) sont transmises sur le dépôt distant, et un compte-rendu apparaît :



Observez le **Project Explorer** :



Vous pouvez maintenant constater les changements sur la page de GitHub :

5 commits

Cliquez sur le nouveau fichier

Cliquez sur History

Vous pouvez consulter l'historique des modifications, et même obtenir les contenus de chaque version :

Ouvrez la dernière version

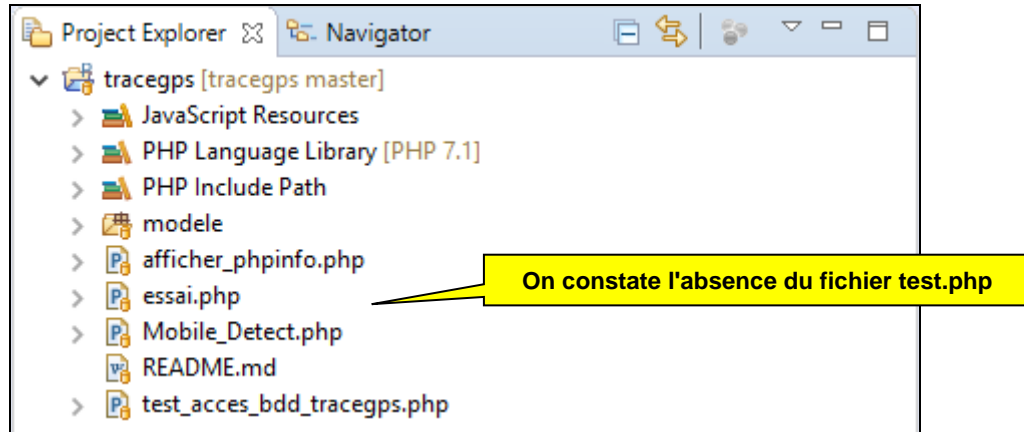
ou ici

+ : précède une ligne ajoutée
- : précède une ligne supprimée

3- Le collaborateur récupère le nouveau fichier (pull)

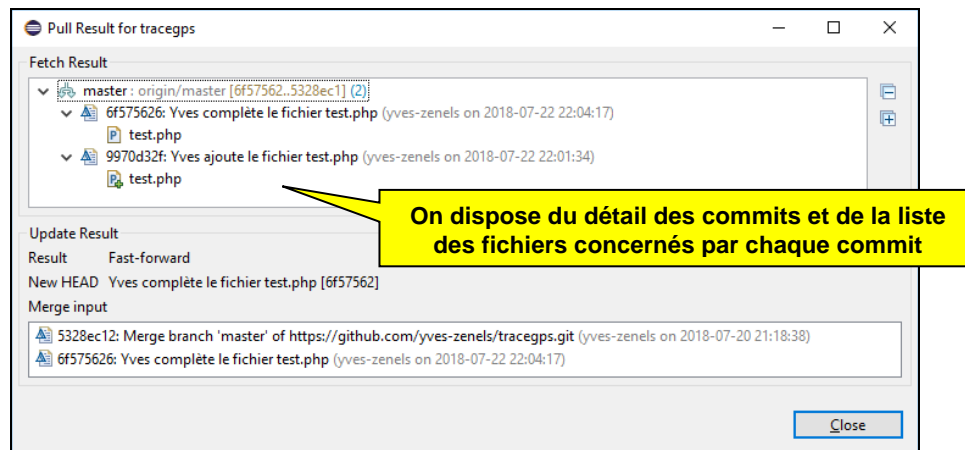
Vous êtes sur le poste de travail du collaborateur (**Amédée Bogueur**) et vous vous demandez si le projet a évolué sur le dépôt distant (en fait, il est recommandé de se poser souvent cette question).

La structure du projet est pour l'instant :

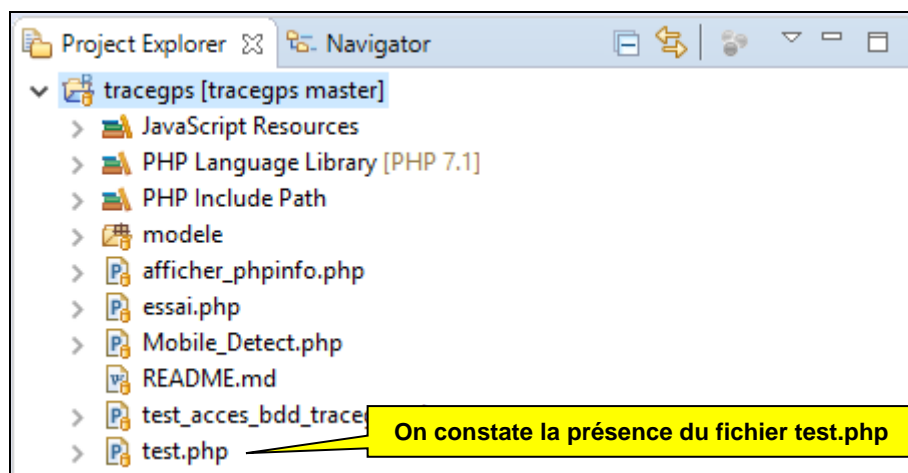


Faites un clic droit sur la racine du projet, choisissez la commande **Team / Pull** :

Si tout se passe bien, les mises à jour (donc le nouveau fichier) sont transmises depuis le dépôt distant vers le dépôt local, et un compte-rendu apparaît :

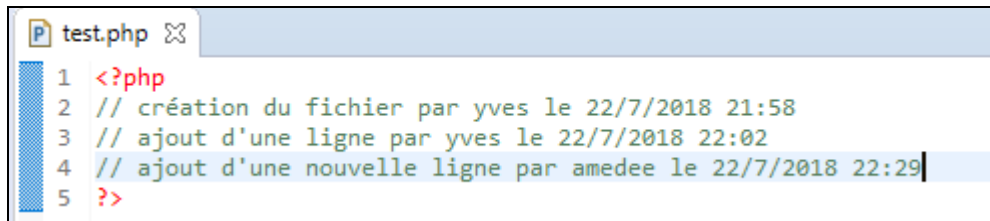


La structure du projet est devenue :



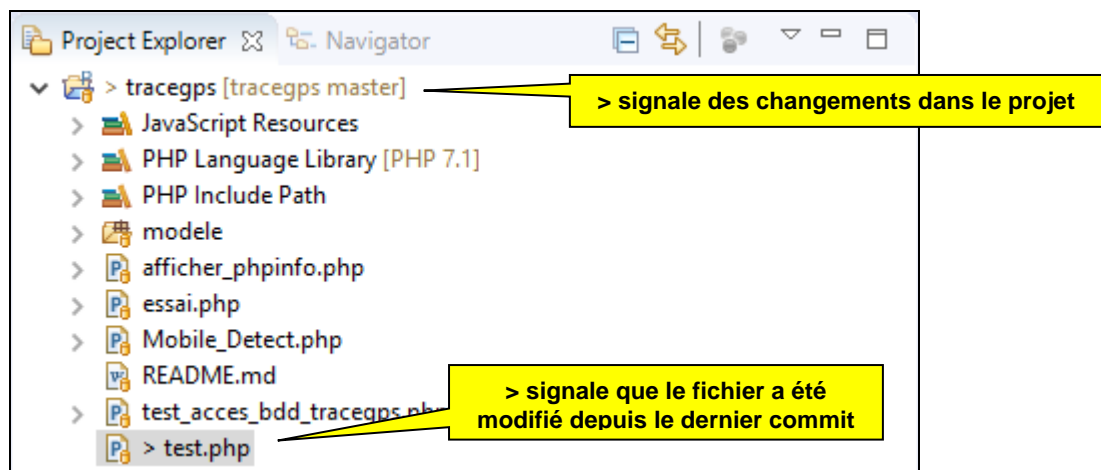
4- Le collaborateur modifie le nouveau fichier (commit & push)

Vous êtes toujours sur le poste de travail du collaborateur (**Amédée Bogueur**) et vous voulez compléter vous aussi le nouveau fichier **test.php** en lui rajoutant une ligne (la ligne 4) :

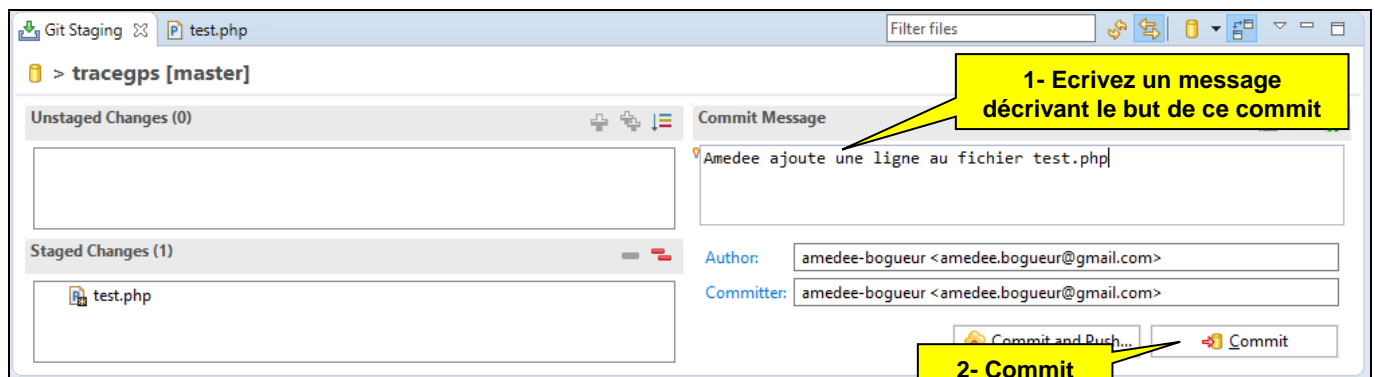


```
1 <?php
2 // création du fichier par yves le 22/7/2018 21:58
3 // ajout d'une ligne par yves le 22/7/2018 22:02
4 // ajout d'une nouvelle ligne par amedee le 22/7/2018 22:29
5 ?>
```

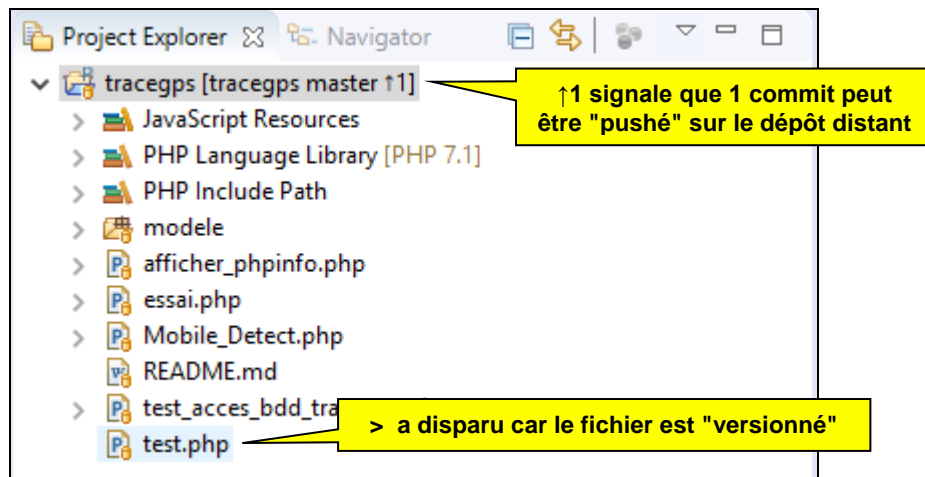
Enregistrez le contenu du fichier et observez la vue **Project Explorer** :



Vous voulez effectuer un "**commit**" local qui permettra à Git de le "versionner" ; faites un clic droit sur le projet et choisissez la commande **Team / Commit...** :

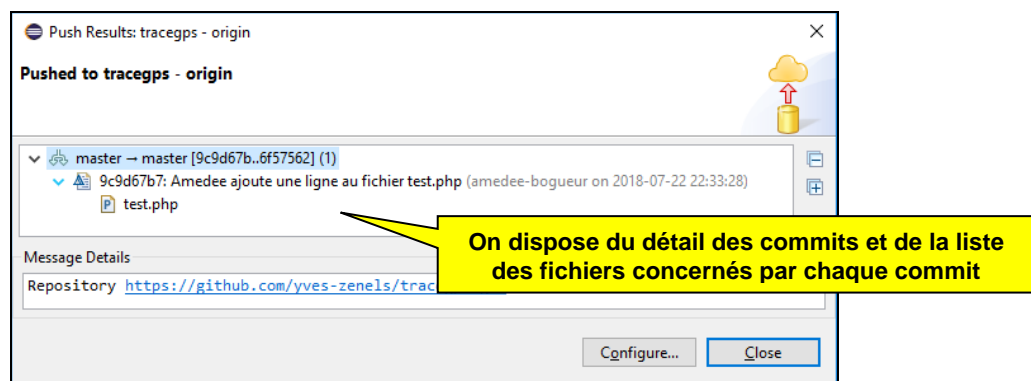


Observez la vue **Project Explorer** :

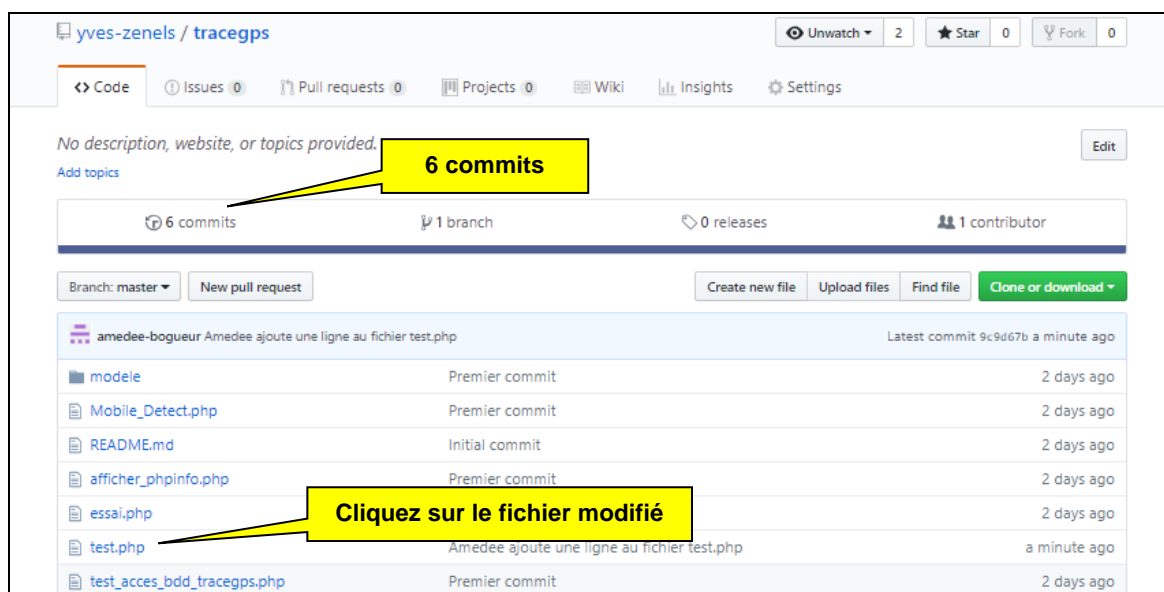


Après quelques commits en local (un seul ici), vous estimez qu'il est temps de les copier sur le dépôt distant ; faites un clic droit sur le projet et choisissez la commande **Team / Push to Upstream**.

Si tout se passe bien, les mises à jour (donc le nouveau fichier) sont transmises sur le dépôt distant, et un compte-rendu apparaît :



Vous pouvez maintenant constater les changements sur la page de GitHub :





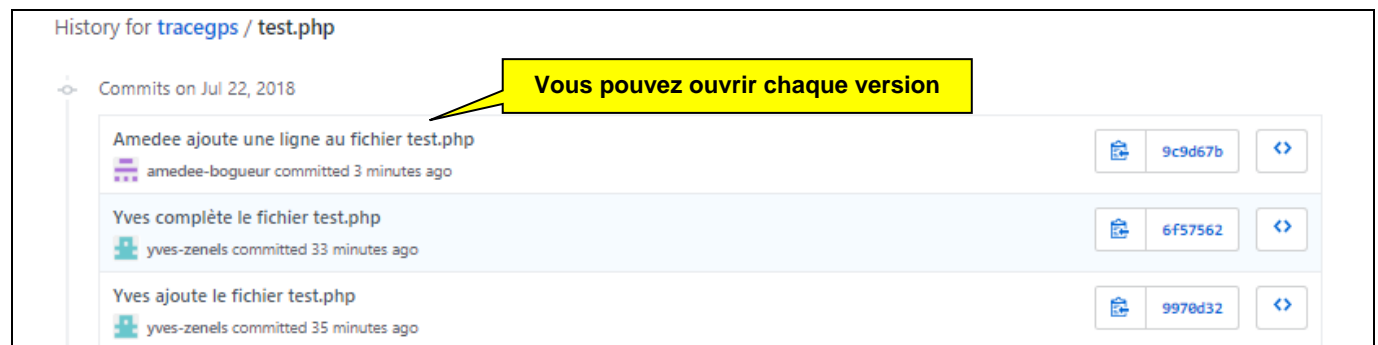
5 lines (5 sloc) | 173 Bytes

Raw Blame History

```
1 <?php
2 // création du fichier par yves le 22/7/2018 21:58
3 // ajout d'une ligne par yves le 22/7/2018 22:02
4 // ajout d'une nouvelle ligne par amedee le 22/7/2018 22:29
5 ?>
```

Cliquez sur History

Vous pouvez consulter l'historique des modifications, et même obtenir les contenus de chaque version :



History for [tracegps](#) / test.php

Commits on Jul 22, 2018

Amedee ajoute une ligne au fichier test.php
amedee-bogueur committed 3 minutes ago

Yves complète le fichier test.php
yves-zenels committed 33 minutes ago

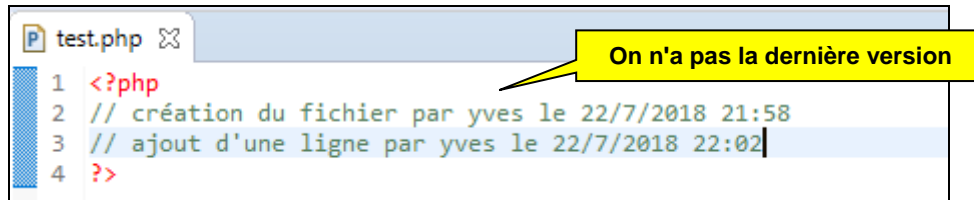
Yves ajoute le fichier test.php
yves-zenels committed 35 minutes ago

Vous pouvez ouvrir chaque version

5- Le chef de projet récupère le fichier modifié (pull)

Vous êtes maintenant sur le poste de travail du chef de projet (**Yves Zenels**) et vous vous demandez si le projet a évolué sur le dépôt distant (en fait, il est recommandé de se poser souvent cette question).

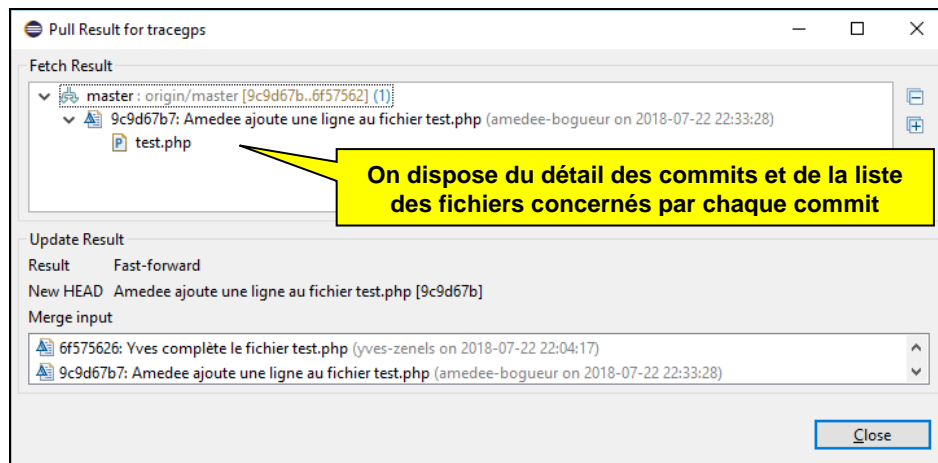
Le contenu du fichier **test.php** est pour l'instant :



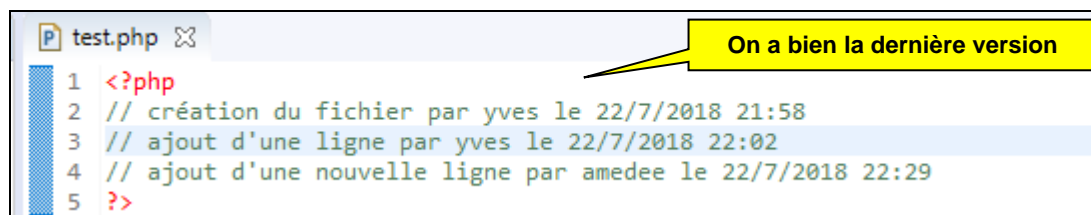
```
1 <?php
2 // création du fichier par yves le 22/7/2018 21:58
3 // ajout d'une ligne par yves le 22/7/2018 22:02
4 ?>
```

Faites un clic droit sur la racine du projet, choisissez la commande **Team / Pull** :

Si tout se passe bien, les mises à jour du fichier modifié sont transmises depuis le dépôt distant vers le dépôt local, et un compte-rendu apparaît :



Le contenu du fichier **test.php** est devenu :

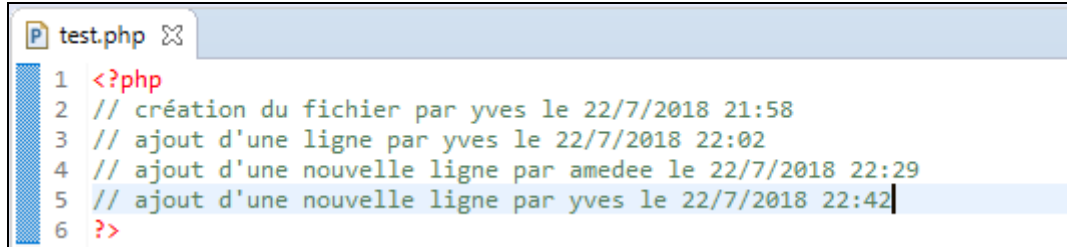


```
1 <?php
2 // création du fichier par yves le 22/7/2018 21:58
3 // ajout d'une ligne par yves le 22/7/2018 22:02
4 // ajout d'une nouvelle ligne par amedee le 22/7/2018 22:29
5 ?>
```

6- Et maintenant, ladies and gentlemen, le CONFLIT en modification

6-1 Le chef de projet modifie le fichier et effectue un commit sans push

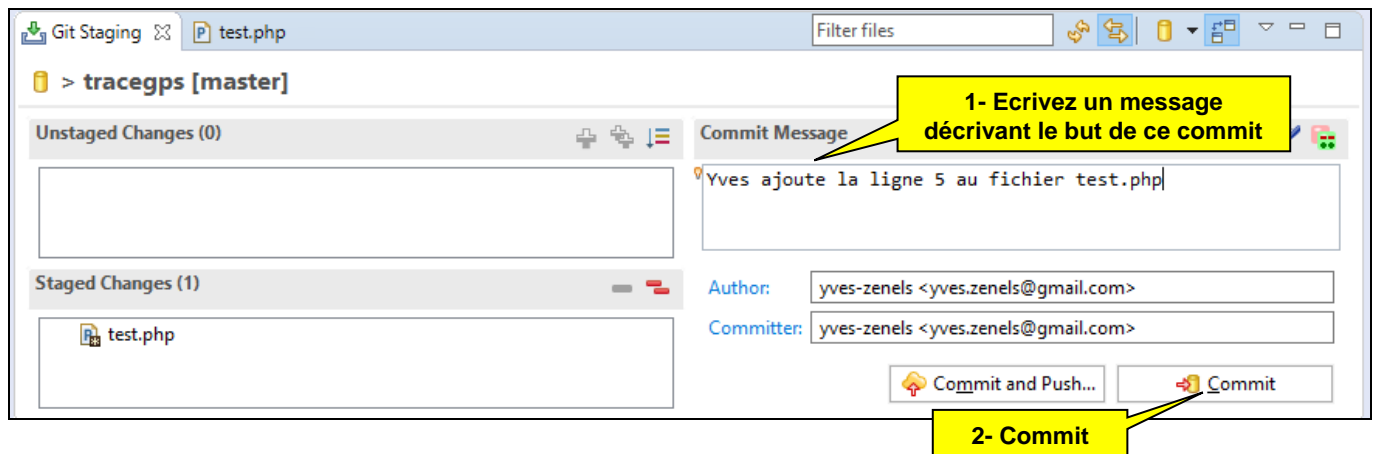
Vous êtes encore sur le poste de travail du chef de projet (**Yves Zenels**) et vous voulez compléter le fichier **test.php** en lui rajoutant une ligne (la ligne 5) :



```

1 <?php
2 // création du fichier par yves le 22/7/2018 21:58
3 // ajout d'une ligne par yves le 22/7/2018 22:02
4 // ajout d'une nouvelle ligne par amedee le 22/7/2018 22:29
5 // ajout d'une nouvelle ligne par yves le 22/7/2018 22:42
6 ?>
  
```

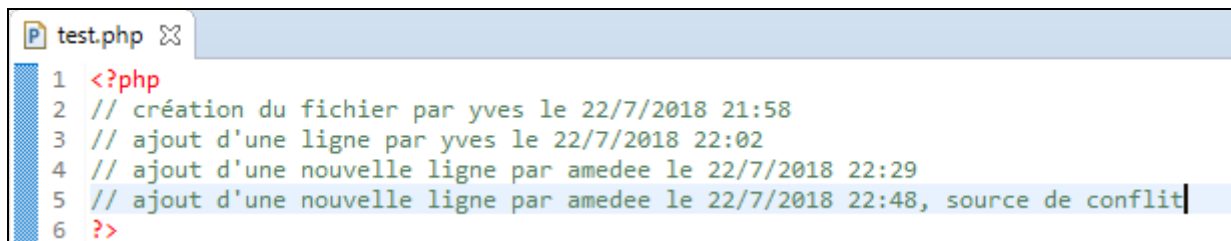
Vous estimez que ce fichier a bien avancé (on est toujours en expérimentation) pour effectuer un nouveau "**commit**" local ; faites un clic droit sur le projet et choisissez la commande **Team / Commit...** :



Et le commit ne va pas être "pushé" immédiatement !

6-2 Le collaborateur modifie aussi le fichier et effectue un commit puis un push

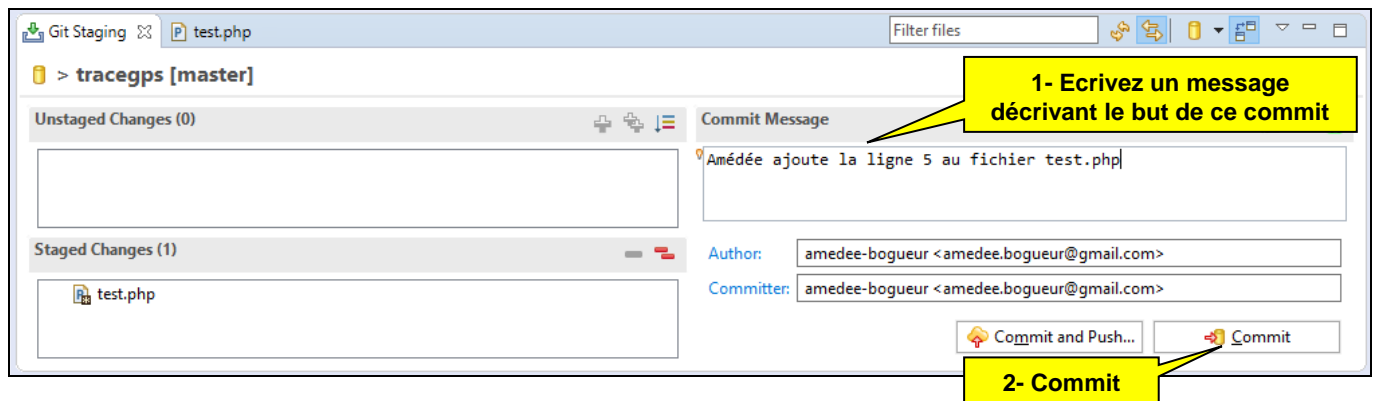
Vous êtes sur le poste de travail du collaborateur (**Amédée Bogueur**) et vous voulez compléter le fichier **test.php** en lui rajoutant également une ligne (la ligne 5) ; cet ajout va bien sûr être en conflit avec le dernier ajout effectué par votre chef de projet et dont vous n'avez pas encore pris connaissance :



```

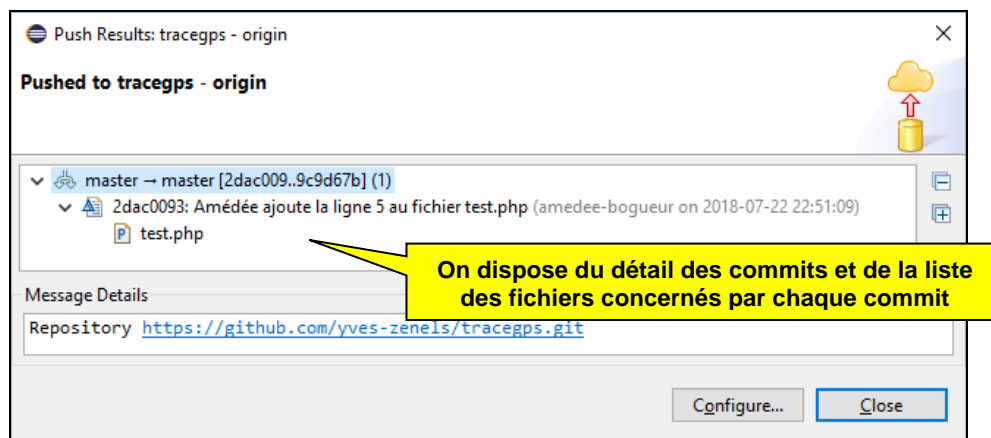
1 <?php
2 // création du fichier par yves le 22/7/2018 21:58
3 // ajout d'une ligne par yves le 22/7/2018 22:02
4 // ajout d'une nouvelle ligne par amedee le 22/7/2018 22:29
5 // ajout d'une nouvelle ligne par amedee le 22/7/2018 22:48, source de conflit
6 ?>
  
```

Vous estimez que ce fichier a bien avancé (on est toujours en expérimentation) pour effectuer un nouveau "**commit**" local ; faites un clic droit sur le projet et choisissez la commande **Team / Commit...** :

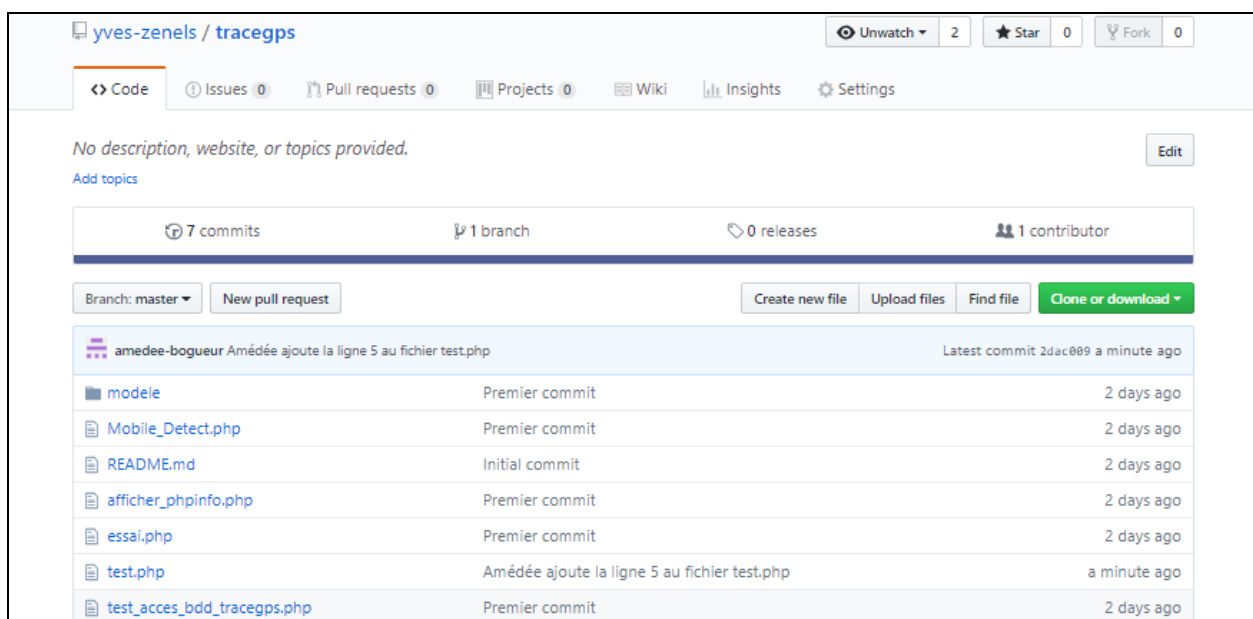


Après ce commit local, vous voulez copier le fichier sur le dépôt distant ; faites un clic droit sur le projet et choisissez la commande **Team / Push to Upstream**.

Si tout se passe bien, les mises à jour (donc le fichier modifié) sont transmises sur le dépôt distant, et un compte-rendu apparaît :



Vous pouvez maintenant constater les changements sur la page de GitHub :



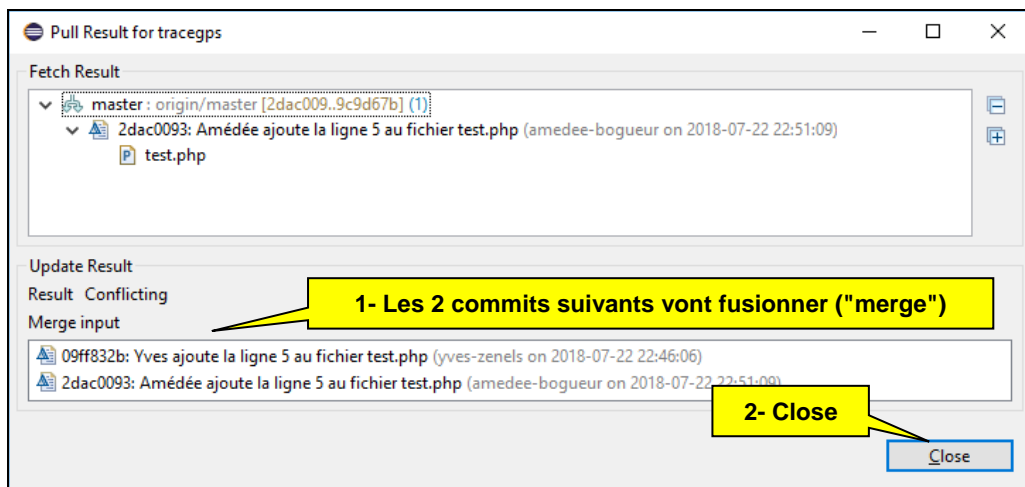
6-3 Le chef de projet effectue un push pour envoyer son dernier commit

Vous êtes maintenant sur le poste de travail du chef de projet (**Yves Zenels**) et vous voulez effectuer un **push** pour envoyer votre dernier commit dans lequel vous avez modifié le fichier **test.php** en lui rajoutant une ligne (la ligne 5).

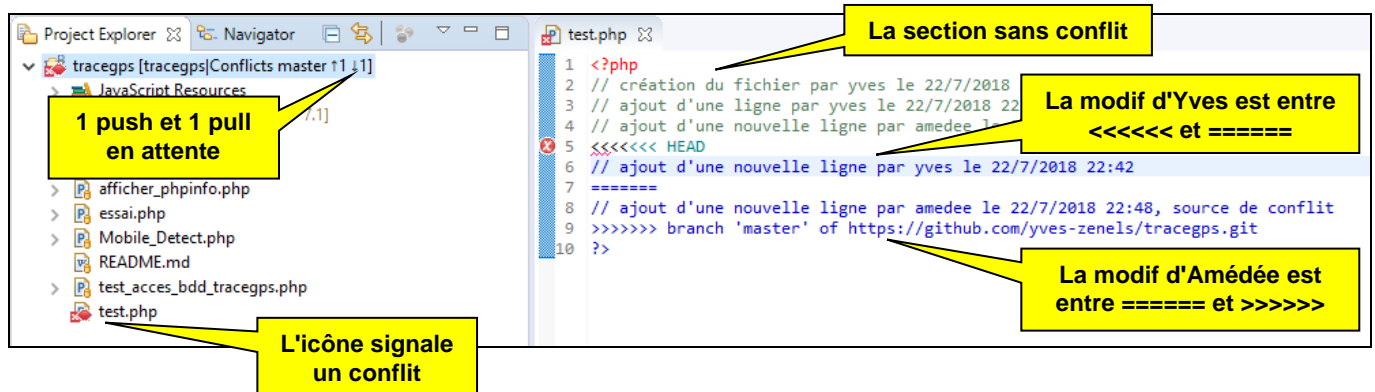
Faites un clic droit sur le projet et choisissez la commande **Team / Push to Upstream**. Cette fois, la commande est rejetée :



Puisqu'on ne peut pas faire de **push**, on va récupérer la version du fichier en conflit à partir du dépôt central, faites un clic droit sur le projet et choisissez la commande **Team / Pull** :



Les 2 versions du fichier en conflit sont maintenant fusionnées et comportent des zones avec 2 versions pour lesquelles il va falloir trancher :



Vous devez donc lever le conflit manuellement en réorganisant le contenu du fichier.

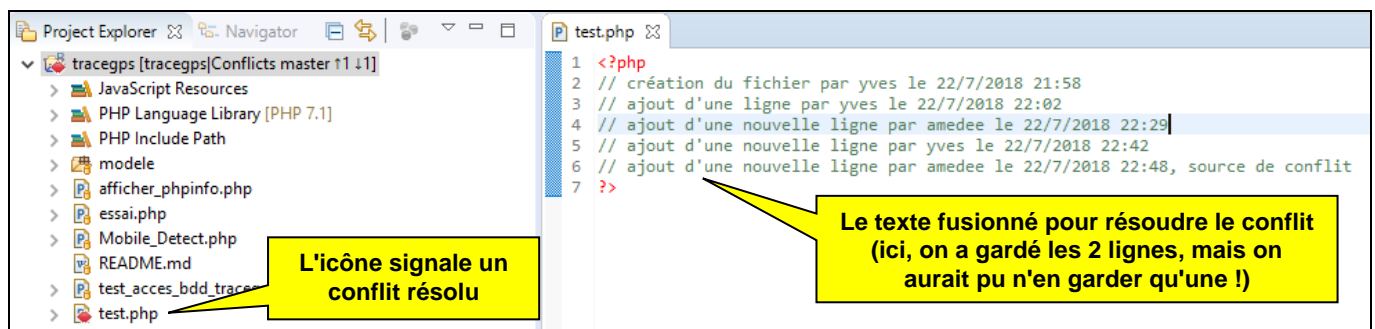
Si on préfère garder la version d'Yves, on supprime les lignes surlignées :

```
<?php
// création du fichier par yves le 22/7/2018 21:58
// ajout d'une ligne par yves le 22/7/2018 22:02
// ajout d'une nouvelle ligne par amedee le 22/7/2018 22:29
<<<<<<< HEAD
// ajout d'une nouvelle ligne par yves le 22/7/2018 22:42
=====
// ajout d'une nouvelle ligne par amedee le 22/7/2018 22:48, source de conflit
>>>>>>> branch 'master' of https://github.com/yves-zenels/tracegps.git
?>
```

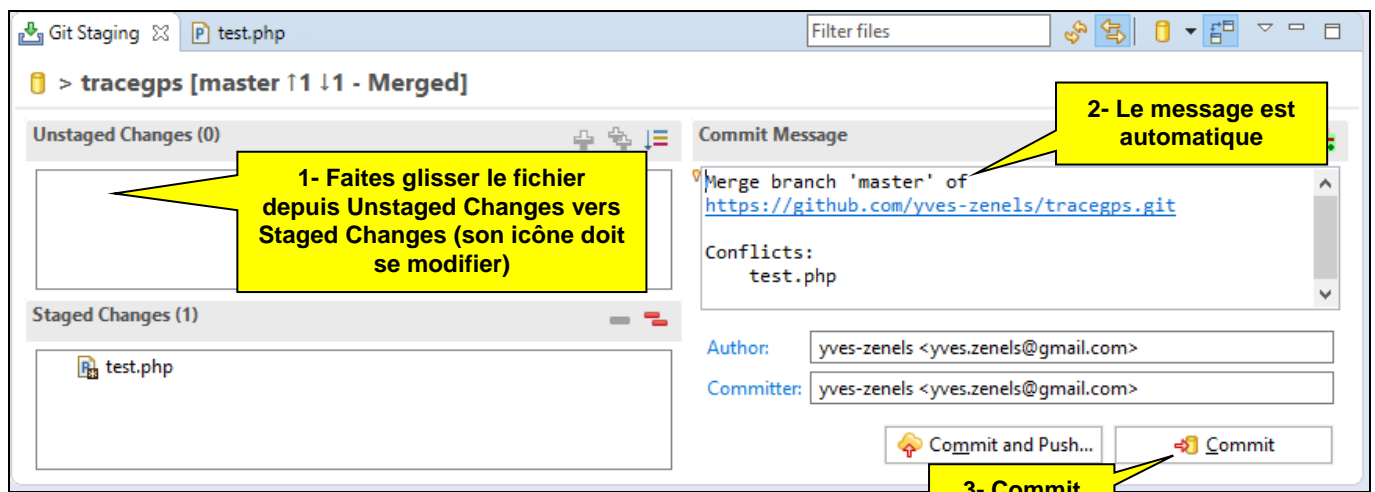
Si on préfère garder la version d'Amédée, on supprime les lignes surlignées:

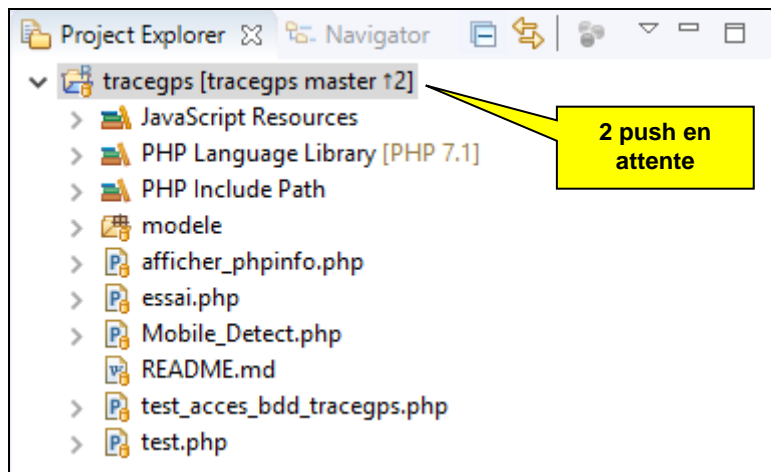
```
<?php
// création du fichier par yves le 22/7/2018 21:58
// ajout d'une ligne par yves le 22/7/2018 22:02
// ajout d'une nouvelle ligne par amedee le 22/7/2018 22:29
<<<<<<< HEAD
// ajout d'une nouvelle ligne par yves le 22/7/2018 22:42
=====
// ajout d'une nouvelle ligne par amedee le 22/7/2018 22:48, source de conflit
>>>>>>> branch 'master' of https://github.com/yves-zenels/tracegps.git
?>
```

Ici, on conserve les 2 propositions :

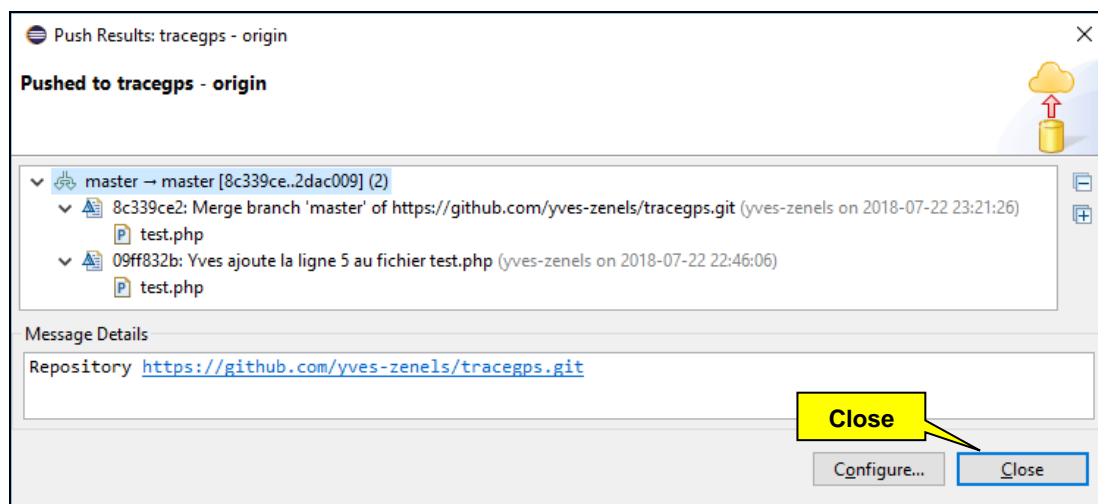


Enregistrez le fichier, et exécutez la commande **Team / Commit** :

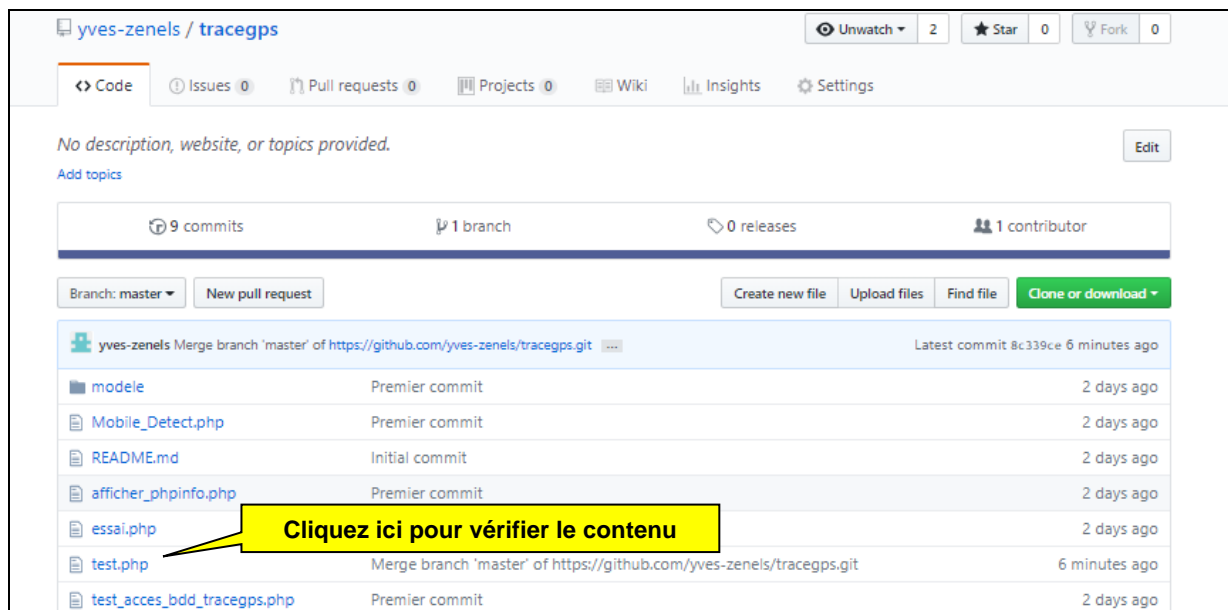




Afin d'envoyer le fichier vers le dépôt distant, exécutez la commande **Team / Push to Upstream** :



Et un petit tour sur **GitHub** pour vérifier :




```
7 lines (7 sloc) | 309 Bytes
1 <?php
2 // création du fichier par yves le 22/7/2018 21:58
3 // ajout d'une ligne par yves le 22/7/2018 22:02
4 // ajout d'une nouvelle ligne par amedee le 22/7/2018 22:29
5 // ajout d'une nouvelle ligne par yves le 22/7/2018 22:42
6 // ajout d'une nouvelle ligne par amedee le 22/7/2018 22:48, source de conflit
7 ?>
```

Il reste un petit problème : le collaborateur n'a pas la dernière version du fichier sur son dépôt local.

Il peut l'obtenir **s'il pense à faire un pull**.

6-4 Le collaborateur peut récupérer la dernière version avec un pull

Vous êtes sur le poste de travail du collaborateur (**Amédée Bogueur**) et vous vous demandez si le projet a évolué sur le dépôt distant (en fait, il est recommandé de se poser souvent cette question).

Faites un clic droit sur la racine du projet, choisissez la commande **Team / Pull**.

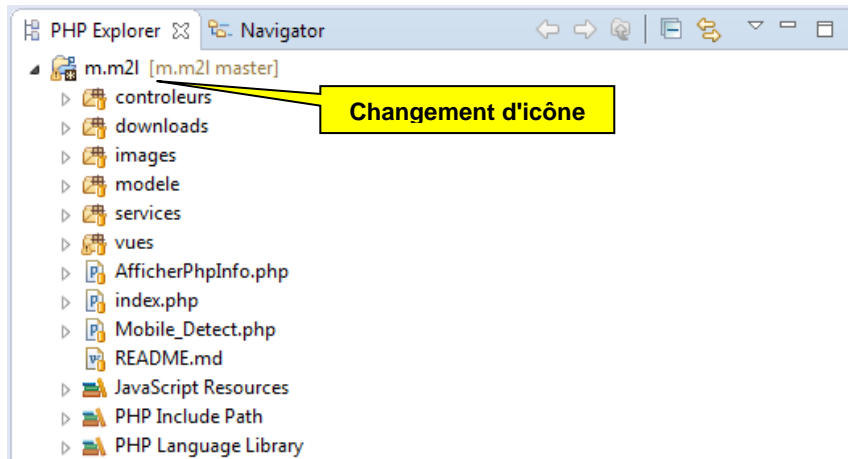


Si tout se passe bien, vous récupérez la dernière version du fichier **test.php**.

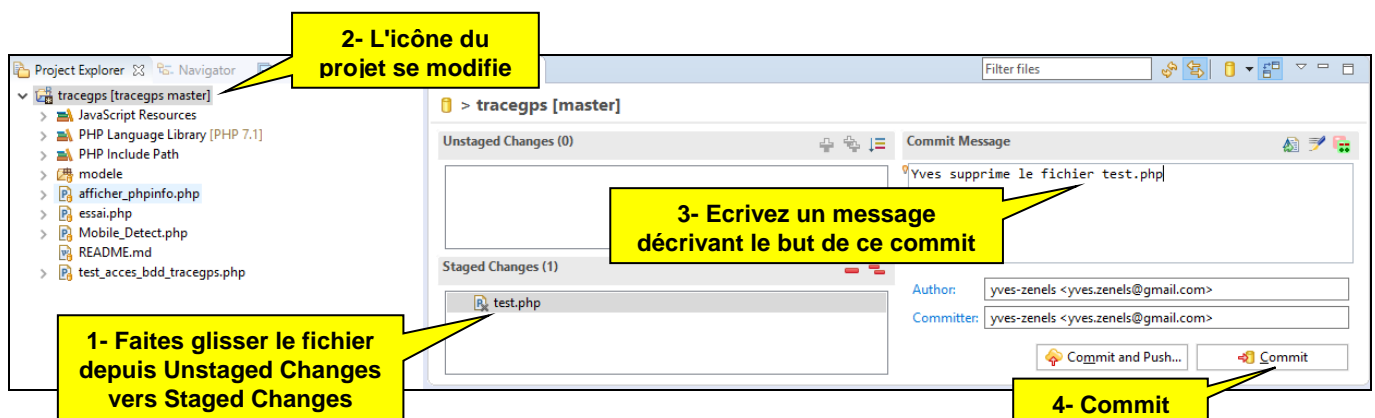
7- Le chef de projet supprime le fichier (commit & push)

Vous êtes sur le poste de travail du chef de projet (**Yves Zenels**) et vous voulez maintenant supprimer le fichier **test.php** ; pour cela, vous faites un clic droit sur le fichier et choisissez la commande **Delete** que vous confirmez.

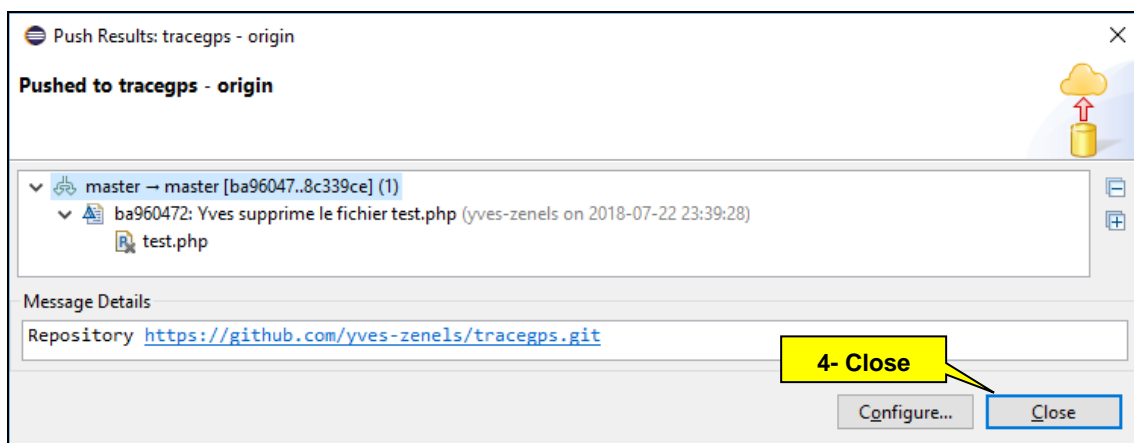
Le fichier est supprimé et l'icône du projet est alors modifiée :



Faites ensuite un commit avec la commande **Team / Commit** :



Faites ensuite un push avec la commande **Team / Push to Upstream** :



Et un petit tour sur **GitHub** pour vérifier la suppression du fichier **test.php** :

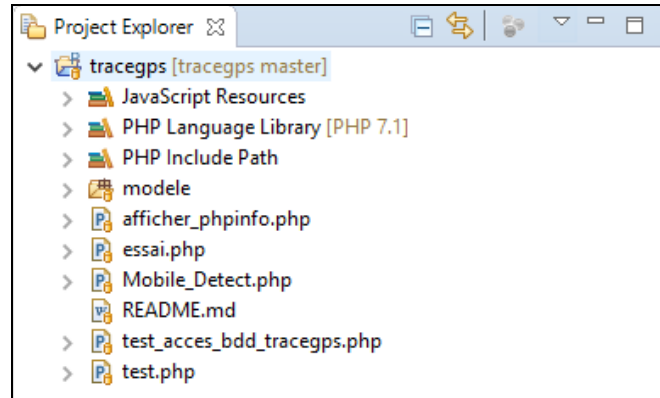
The screenshot shows the GitHub repository page for 'yves-zenels / tracegps'. The repository has 10 commits, 1 branch, 0 releases, and 1 contributor. The 'Code' tab is selected, showing a list of files and their commit history. The files listed are: 'modele', 'Mobile_Detect.php', 'README.md', 'afficher_phpinfo.php', 'essai.php', and 'test_accès_bdd_tracegps.php'. The commit history for each file shows the first commit, with the most recent commit being 'ba96047' from 'yves-zenels' just a minute ago, which is titled 'Yves supprime le fichier test.php'.

File	Commit	Time
modele	Premier commit	2 days ago
Mobile_Detect.php	Premier commit	2 days ago
README.md	Initial commit	2 days ago
afficher_phpinfo.php	Premier commit	2 days ago
essai.php	Premier commit	2 days ago
test_accès_bdd_tracegps.php	Premier commit	2 days ago

8- Le collaborateur constate la suppression du fichier (pull)

Vous êtes sur le poste de travail du collaborateur (**Amédée Bogueur**) et vous vous demandez si le projet a évolué sur le dépôt distant (en fait, il est recommandé de se poser souvent cette question).

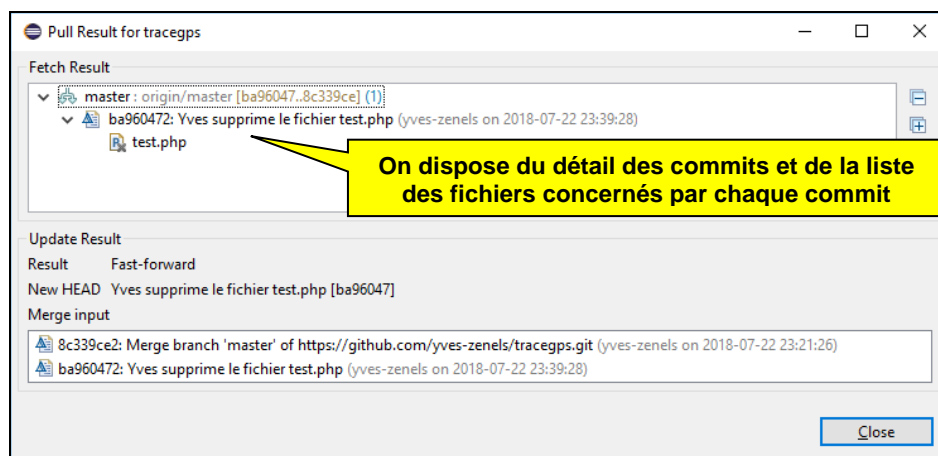
La structure du projet est pour l'instant :



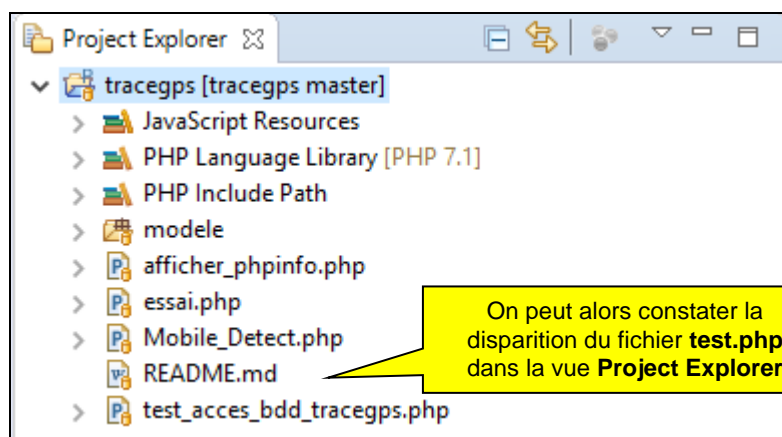
Le projet local contient encore le fichier **test.php**

Faites un clic droit sur la racine du projet, choisissez la commande **Team / Pull** :

Si tout se passe bien, les mises à jour (donc la suppression du fichier **test.php**) sont transmises depuis le dépôt distant vers le dépôt local, et un compte-rendu apparaît :



On dispose du détail des commits et de la liste des fichiers concernés par chaque commit



On peut alors constater la disparition du fichier **test.php** dans la vue **Project Explorer**

9- Quelques conseils pratiques

9-1 Faites des "pull" sans hésitation

N'hésitez pas à faire des **pull** sans crainte (chargement des dernières versions depuis le dépôt distant vers le dépôt local) ; si il n'y a rien à remonter depuis le dépôt distant, le **pull** ne fera rien :



Faites systématiquement un **pull** lorsque vous démarrez une session de travail avec Eclipse pour récupérer les dernières modifications du projet. Si des fichiers ont été modifiés par d'autres collaborateurs, ils seront fusionnés automatiquement avec vos versions locales.

Faites systématiquement un **pull** AVANT de faire un **push** ; le **pull** permettra de fusionner (**merge**) le fichier que vous avez modifié (et que vous voulez "pusher") avec la version distante si elle a été modifiée par un autre collaborateur. Faites ensuite un **commit** et un **push**.

Si vous procédez ainsi, vous diminuez considérablement le risque de conflit.

9-2 Fréquence des commit et des push

Faites un **commit** pour valider un **petit développement** qui a été **testé avec succès** :

- l'ajout d'une nouvelle fonction ou méthode
- la modification d'une fonction ou méthode
- la correction d'un dysfonctionnement

Faites un **push** si vous pensez que vos modifications sont attendues par d'autres collaborateurs, et au moins une fois dans une journée de développement.

9-3 Evitez de supprimer des lignes de code

Plutôt que de supprimer des lignes de code, il est préférable (au moins pendant quelques temps) de les désactiver en les transformant en commentaires :

- en plaçant des `//` au début de chaque ligne (avec Ctrl-7 ou Ctrl-/)
- ou en encadrant le bloc par des `/*` et `*/`

La suppression définitive des lignes pourra être faite plus tard, quand on sera sûr de leur inutilité.

9-4 Commentez chaque modification de code

A chaque fois que vous modifiez une section de code dans un fichier, encadrez la modification par un commentaire de début et un commentaire de fin, par exemple :

```
// début modif par dP le 19/6/20xx
.....
.....
// fin modif par dP le 19/6/20xx
```