Utilisation de Git

1. **Installation du SCM de Git**

Pour l'installation du SCM de git il faut ce rendre sur le site https://git-scm.com et de télécharger le logiciel pour windows. Git SCM est une console qui permet de gérer le versionning en local ou sur internet (GitHub...).

Suivre les consignes pendant l'installation, surtout il faut faire attention à changer l'éditeur par défaut si nécessaire (Ex : mettre visualStudio par défaut ou Atom etc).

> *Important : pensez à redémarrer votre système après l'installation*

Pour initialiser votre compte a git, il vous suffit d'inscrire ces commandes suivantes :

```
git config --global user.email "your_email_address@example.com"
git config --global user.name "your_name"
```

2. **Extension à ajouter pour l'utilisation de Git dans VisualStudio Code**

Sur VisualStudio Code l'implémentation de Git est de base dans le logiciel. Pour faciliter l'utilisation de Git, il est conseillé d'avoir l'extension GitHub Pull Request and Issue. Cette extension permet :

- Une authentification plus simple et rapide à Github
- Simplification du travail sur les Issues et les Requests
- Possiblité à partir de VS Code de mentionner une personne active du projet

3. **Utilisation de Git à partir du terminal de VS Code**

Pour l'utilisation de Git dans VS Code il suffit d'ouvrir un terminal en utilisant le commande :

```
CTRL + SHIFT + ù
```

Une fois dans le terminal il faut ce positionner dans le dossier qui doit accueillir le/les projet(s) de GitHub ou le dossier de l'application en cours de développement. Pour cela on utilise les commandes suivantes :

```
cd /leDossierRacine/MonProjet
```

Dès lors que vous êtes positionné dans le dossier cible, il faut inscrire la commande :

```
git clone "URL DU REPOSITORY"
```

> *Cette commande a permis au SCM de git d'ajouter des fichiers pour permettre le bon fonctionnement du versionning et de télécharger les fichiers du projet.*

Après avoir executer la commande précedente, il faut ce positionner dans le dossier qui vient d'être télécharger.

```
/leDossierRacine/MonProjet cd monRepository
```

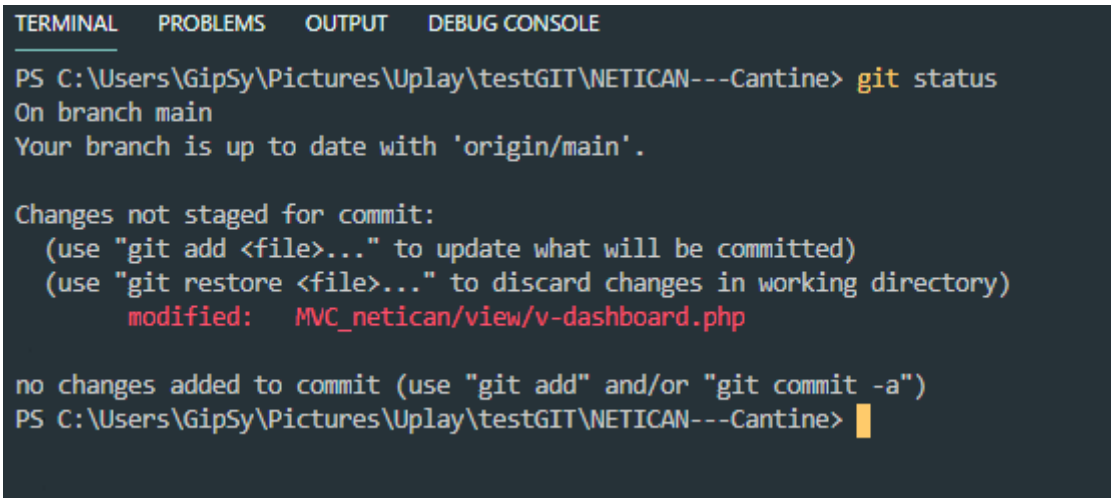
Voici le résultat :

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell https://aka.ms/pscore6

PS C:\Users\GipSy\Pictures\Uplay\testGIT> git clone https://github.com/BTS-SIO2019/NETICAN---Cantine.git
Cloning into 'NETICAN---Cantine'...
remote: Enumerating objects: 102, done.
remote: Counting objects: 100% (102/102), done.
remote: Compressing objects: 100% (55/55), done.
Receiving objects:  86% (88/102), 580.00 KiB | 1.11 MiB/s-reused 0
Receiving objects: 100% (102/102), 1.16 MiB | 1.27 MiB/s, done.
Resolving deltas: 100% (46/46), done.
PS C:\Users\GipSy\Pictures\Uplay\testGIT> cd .\NETICAN---Cantine\
PS C:\Users\GipSy\Pictures\Uplay\testGIT\NETICAN---Cantine>
```

Dès lors que le clonage du Repository est fait, vous pouvez librement changer les fichiers concernés par vos tâches.

Une fois les modifications effectuées il va falloir valider les changements faits. Pour cela on vérifie déjà si toute les évolutions ont été pris en compte avec la commande suivante :

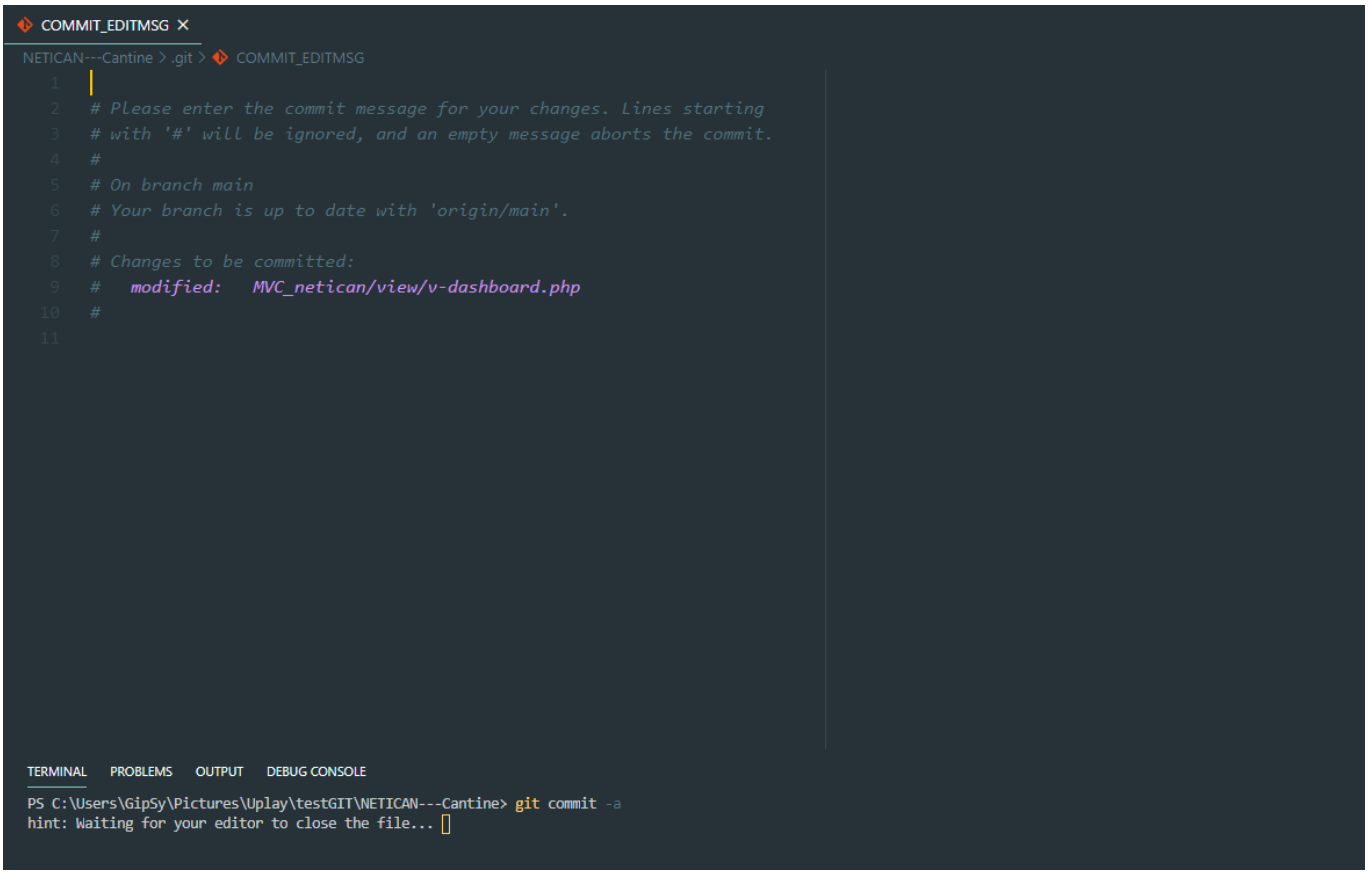
```
git status
```



> *Comme dans l'image ci-dessus, on peut s'apercevoir qu'il y a eu un changement sur le fichier MVC_netican/view/v-dashboard.php. Il faut bien faire attention au(x) fichier(s) modifié(s) que ce soit le(s) bon(s)*

Suite à la vérification de(s) fichier(s) il faut ensuite utliser la commande :

```
git commit -a
```



> *Enregistre des instantanés de fichiers de façon permanente dans l'historique des versions Le fichier ouvert automatiquement par la commande est utile si vous souhaitez faire un commentaire sur ce que vous avez fait(s). Pour effectuer le commit il suffit de fermer le fichier.*

La dernière étapes est de push notre commit sur GitHub, pour cela il nous suffit d'utiliser la commande :

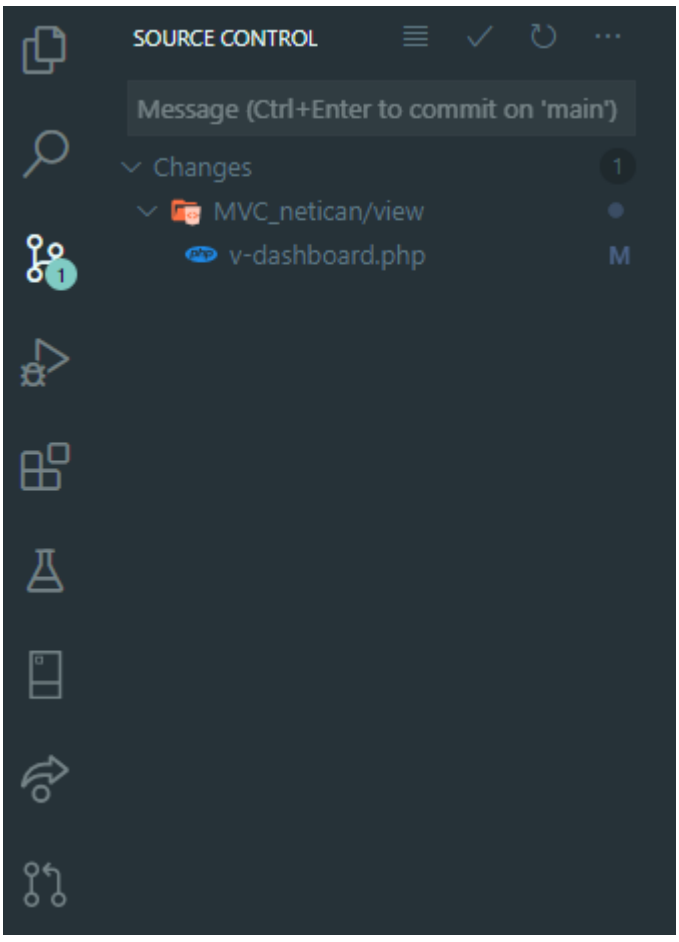
```
git push
```

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE
PS C:\Users\GipSy\Pictures\Uplay\testGIT\NETICAN---Cantine> git commit -a
[main bac8cb0] Test
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\GipSy\Pictures\Uplay\testGIT\NETICAN---Cantine> git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 427 bytes | 427.00 KiB/s, done.
Total 5 (delta 3), reused 3 (delta 2), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/BTS-SIO2019/NETICAN---Cantine.git
   7829285..bac8cb0  main -> main
PS C:\Users\GipSy\Pictures\Uplay\testGIT\NETICAN---Cantine>
```

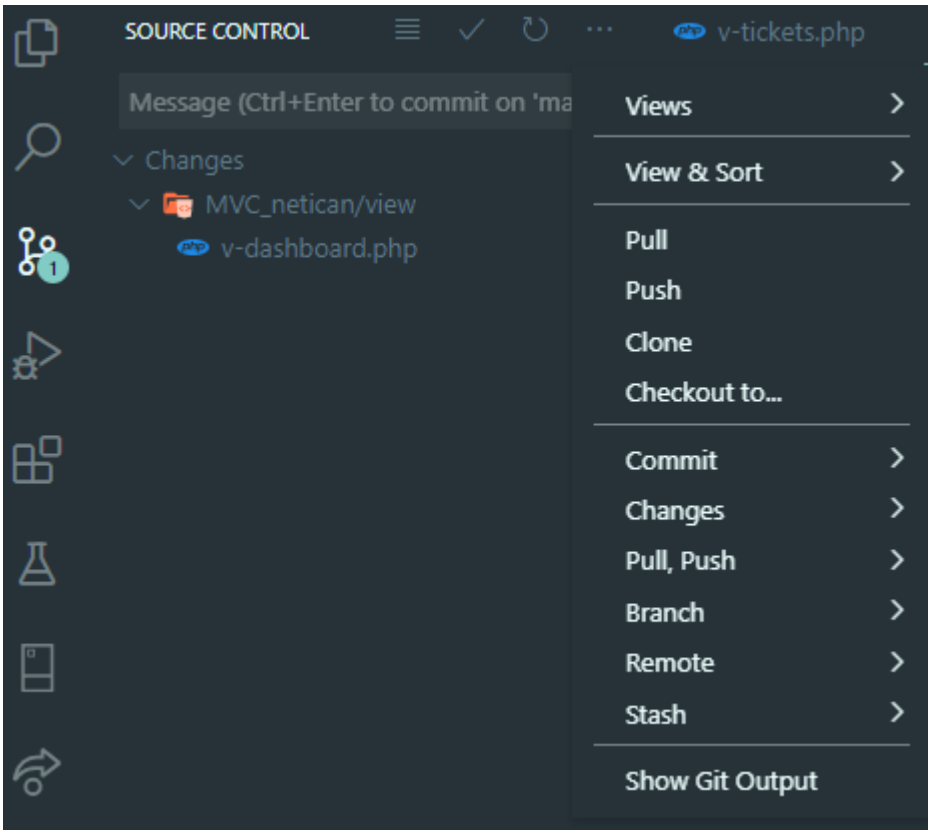
Informations complémentaire

> Voici quelque informations pratique pour la visualisation de git sur VSCode

1. **La visualisation des modifications sur un ou plusieurs fichier(s)**



2. **Utilisation des actions présente sur VSCode**



Nom	Description
Pull	Rapatrier et intégrer un autre dépôt ou une branche locale
Push	Met à jour les références distantes ainsi que les objets associés
Clone	Clone un dépôt dans un nouveau répertoire
Checkout	Bascule sur une autre branche ou restaure des fichiers de l'arbre de travail
Commit	Enregistre des instantanés de fichiers de façon permanente dans l'historique des versions
Change	Inconnu pour l'instant*
Pull, Push	Rapatrier et intégrer un autre dépôt ou une branche locale / Met à jour les références distantes ainsi que les objets associés
Branch	Liste, crée, ou supprime des branches
Remote	Gère un ensemble de dépôts (« distants ») dont vous suivez les branches.
Stash	Remettre en ordre tout les fichiers