



Cégep de Saint-Hyacinthe
Département d'informatique

Programmation orientée objet

420-2DP-HY

(3-3-3)

Initiation à un gestionnaire de versions (GIT) #1

(Version 1.2)

3 heures

Préparé par

Martin Lalancette

Comprendre les éléments suivants:

- Qu'est-ce qu'un gestionnaire de versions?
- Comment lier VS à DevOps
- Pousser/Tirer des modifications (branche *master*)

Table des matières

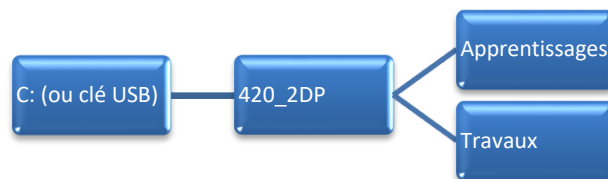
Introduction.....	3
Préparation de base	3
Qu'est-ce qu'un gestionnaire de versions?	3
Pourquoi utiliser un gestionnaire de version	4
Types de gestionnaire de versions	4
Terminologie à comprendre.....	5
Définir les paramètres GIT.....	6
Démarrer un projet DevOps avec VS (développeur maitre)	6
Projet DevOps (AVANT) et solution/projet Visual Studio (APRÈS)	6
Exercices en équipe de deux	14
Bibliographie.....	16

Introduction

Cette séquence a pour but de vous initier aux notions de base nécessaires à la gestion de versions en utilisant DevOps (Azure Repos – GIT). Nous commencerons par énoncer les éléments théoriques appuyés d'exemples simples et faciles à reproduire. Afin d'axer l'attention sur la compréhension de ces notions, il y aura des exercices à faire tout au long de cette séquence. **Il faut préalablement avoir fait la séquence « Configuration d'Azure DevOps »** avant de faire cette séquence.

Préparation de base

Pour bien suivre les instructions qui vont être mentionnées tout au long des séquences d'apprentissage, une préparation de base s'impose. Il est important de créer un répertoire de travail (sur votre C: ou clé USB). Voici une suggestion d'arborescence:



Préparation : S'assurer d'avoir créé l'arborescence ici haut mentionnée sur votre C ou votre clé USB. Copier ce document dans le répertoire en rouge ici haut mentionné.

Qu'est-ce qu'un gestionnaire de versions?

Un gestionnaire de versions est un logiciel qui ***permet d'emmagasiner les fichiers sources d'un projet, de conserver la chronologie de toutes les modifications qui ont été effectuées et de favoriser la collaboration entre les développeurs.*** Il existe plusieurs logiciels qui accomplissent ce rôle : Team-Foundation, Subversion, Git, etc. Dans ce cours, nous utiliserons **Git** fourni gratuitement par **Azure DevOps**.

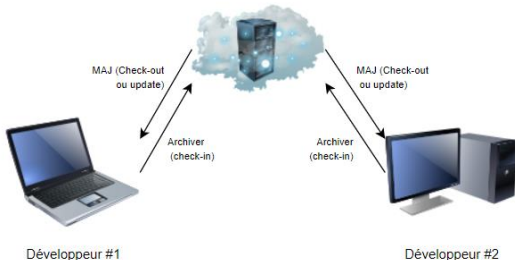
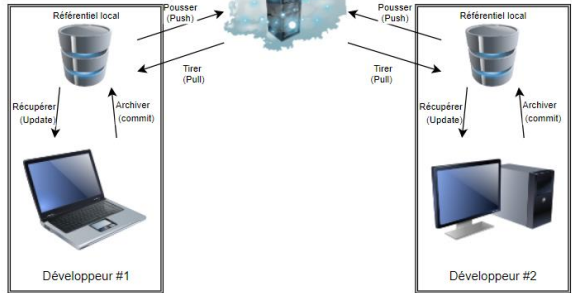
Pourquoi utiliser un gestionnaire de version

Voici quelques raisons de mettre en place un gestionnaire de versions dans son entreprise et même pour des projets personnels à la maison:

- Permet de maintenir plusieurs versions du code
- Permet de revenir en tout temps à une version antérieure (*rollback*)
- Plusieurs développeurs peuvent travailler sur un même projet en parallèle
- Traçabilité des modifications effectuées (Quand?, Ou?, Quoi?, Par qui?) - Historique
- Permet de synchroniser le code entre les développeurs
- Permet de copier, de fusionner ou d'annuler des changements (code)
- Permet de comparer les différences entre les versions
- Permet de conserver des copies de sécurité

Types de gestionnaire de versions

Il existe 2 deux types de gestionnaire de versions : Centralisé et distribué.

Gestion de versions <u>Centralisée</u>	Gestion de versions <u>Distribuée</u>
<p style="text-align: center;">Centralisé Référentiel central (Repository)</p>  <p style="text-align: center;">(diagrams.net/draw.io, 2022)</p>	<p style="text-align: center;">Décentralisé Référentiel central (Repository)</p>  <p style="text-align: center;">(diagrams.net/draw.io, 2022)</p>
<p>Logiciels :</p> <ul style="list-style-type: none"> • TeamFoundation (DevOps) • Subversion • CVS • Etc. 	<p>Logiciels :</p> <ul style="list-style-type: none"> • Git ou GitHub (DevOps) • Veracity • Plastic SCM • Etc.

Avantages : <ul style="list-style-type: none"> • Simple à utiliser (facile à apprendre) • Facile à configurer 	Avantages : <ul style="list-style-type: none"> • Plus flexible • Permet de travailler hors connexion (<i>Commit, history, etc.</i>) • Plus rapide car moins de connexion • Gestion des branches plus facile
--	--

Terminologie à comprendre

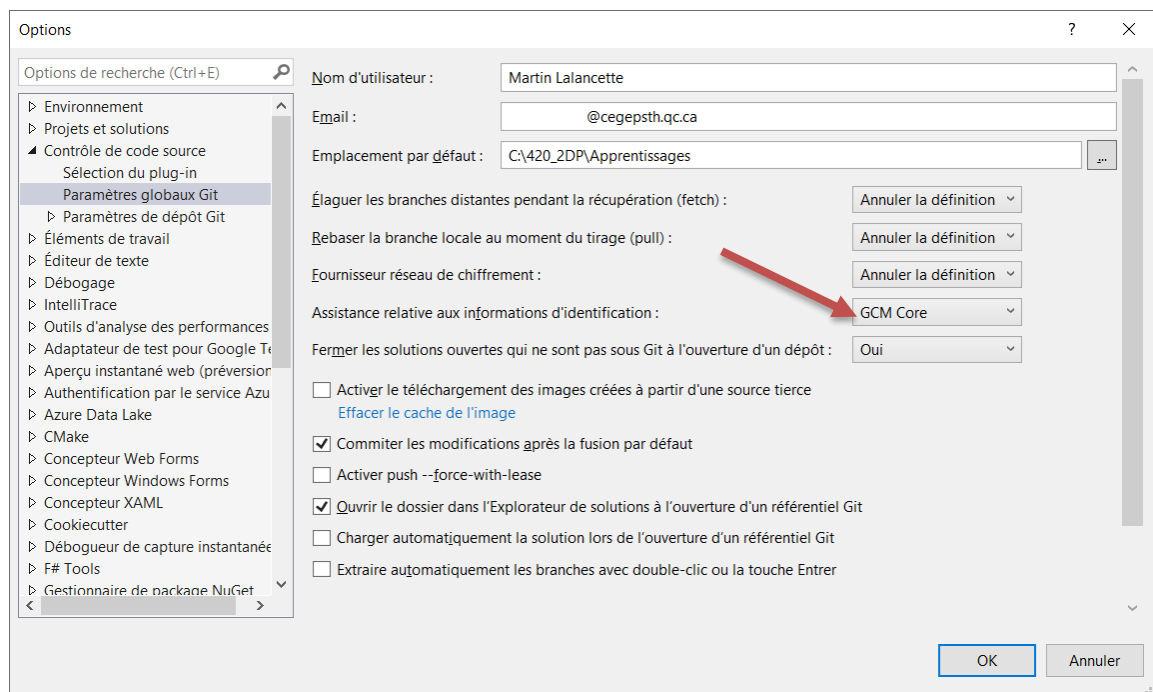
Voici quelques termes de base à maîtriser pour débiter. D'autres s'ajouteront avec le temps.

Mot	Description
<i>Repository</i> Référentiel ou dépôt	Un référentiel est l'endroit où sont emmagasinés les fichiers faisant partie d'un ou plusieurs projets. Local : Signifie que le référentiel se trouve sur le poste du développeur. Central : Signifie que le référentiel se trouve sur un serveur. Un référentiel peut contenir plusieurs projets.
<i>Branch</i> Branche	Les branches permettent de gérer plusieurs versions de fichiers associées à un projet. La branche principale est appelée master ou main . Cette branche est considérée comme déployable. D'autres branches peuvent être créées à partir de cette branche afin de développer une nouvelle version du produit, effectuer des tests, essayer de nouvelles idées, etc.
<i>Check-out</i>	Associée aux gestionnaires de versions centralisés , cette opération est nécessaire la première fois qu'un développeur souhaite participer. Elle consiste à copier localement (dans un dossier) le contenu d'une branche à partir du référentiel central.
<i>Clone</i> Cloner	Associée aux gestionnaires de versions distribués (comme GIT), cette opération consiste à effectuer une copie du référentiel central vers un dossier local. Ceci évite d'être toujours connecté au référentiel central pour effectuer les opérations (Ex. : <i>Commit, Revert</i>).
<i>Commit ou Check-in</i> Archiver ou Valider	Permet de déposer une ou plusieurs modifications dans le référentiel. Centralisé : Les modifications sont déposées directement sur le serveur (les autres développeurs pourront voir vos modifications et les prendre). Distribué (GIT): Les modifications sont déposées dans votre référentiel local (les autres ne les voient pas encore). Ici, une autre étape sera nécessaire → Pousser.
<i>Update</i> Mettre à jour ou Synchroniser	Permet de récupérer les modifications faites par les autres développeurs à partir du référentiel central.
<i>Push</i> Pousser	Distribué (GIT): Permet de déposer une ou plusieurs modifications dans le référentiel central . Ici, les autres développeurs pourront voir vos modifications et les prendre.
<i>Pull</i> Tirer	Distribué (GIT): Permet de récupérer une ou plusieurs modifications faites par les autres développeurs à partir du référentiel central .

Lien sur l'utilisation de [DevOps GIT avec VS2019](#).

Définir les paramètres GIT

Afin de vous assurer d'un bon fonctionnement avec GIT, voici les paramètres à définir via le menu **GIT/Paramètres**:



Vous pouvez si vous le souhaitez changer l'emplacement par défaut pour le dossier **Apprentissages** de ce cours.

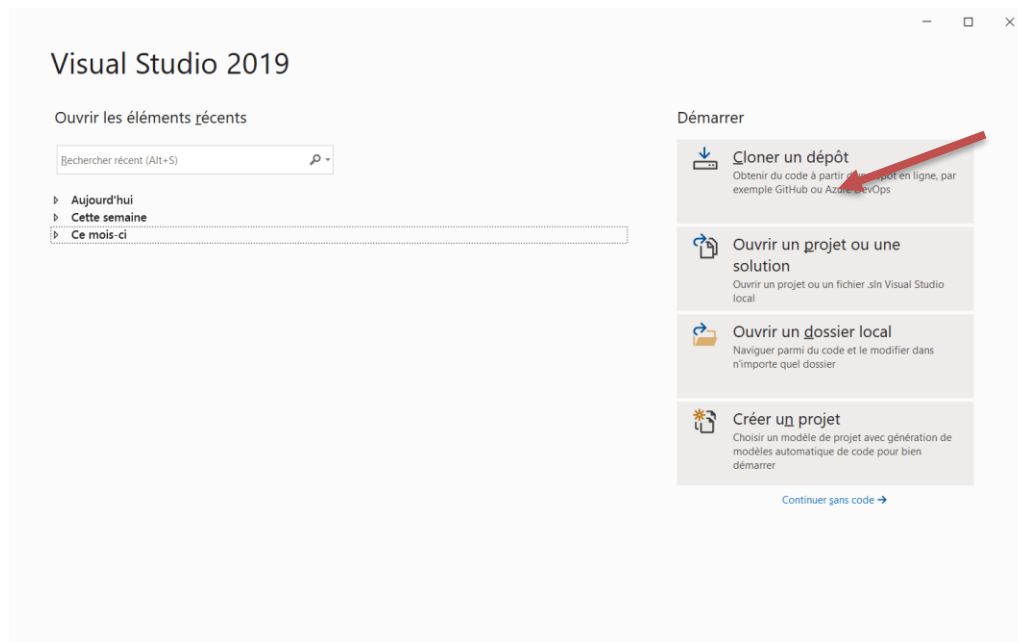
Démarrer un projet DevOps avec VS (développeur maitre)

Projet DevOps (AVANT) et solution/projet Visual Studio (APRÈS)

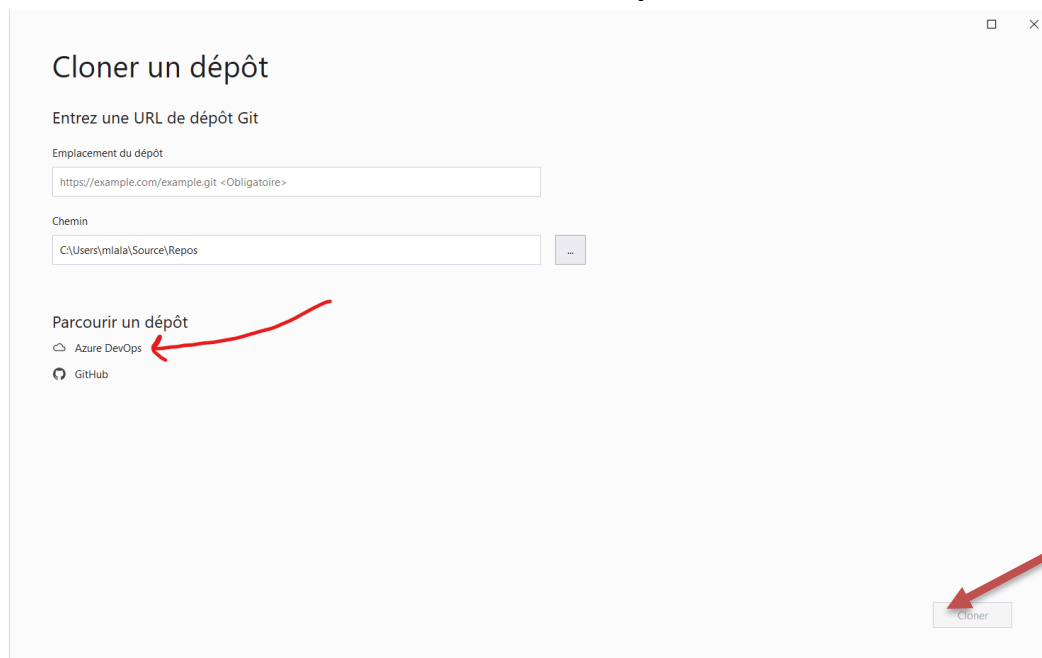
Après avoir créé un projet dans DevOps, voici la procédure pour y des fichiers sources de départ à la branche *master* :

Voici maintenant la procédure à suivre pour accéder à DevOps dans Visual Studio.

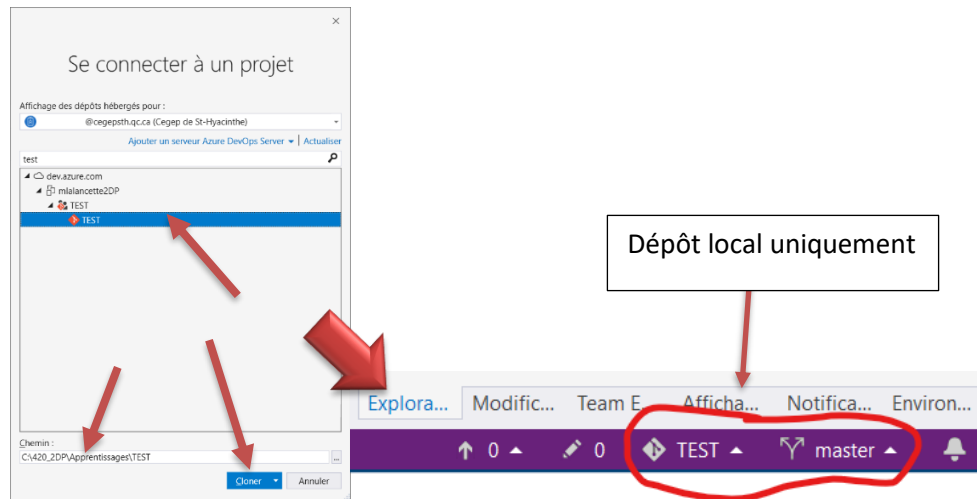
1. Démarrer Visual Studio et choisir « Cloner un dépôt »,



2. Dans l'écran suivant, choisir "Azure DevOps",

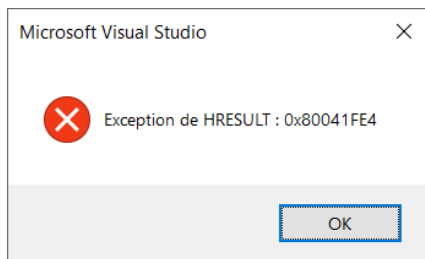


3. Se connecter au projet se trouvant dans DevOps. Exemple:

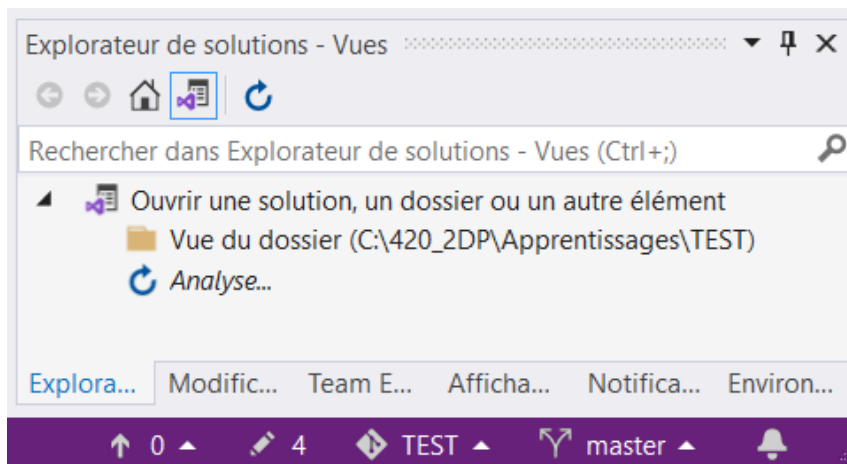


Cliquer sur les éléments spécifiés par les flèches.

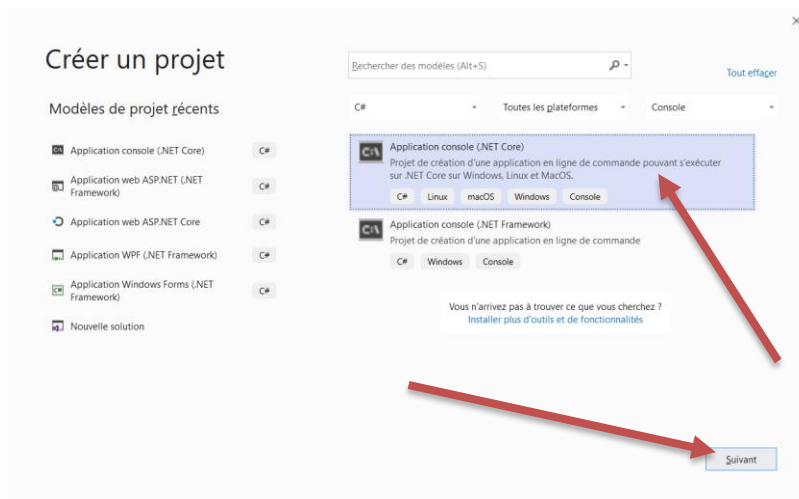
4. **Fermer Visual Studio pour éviter l'erreur suivante** lors de la création d'un nouveau projet:



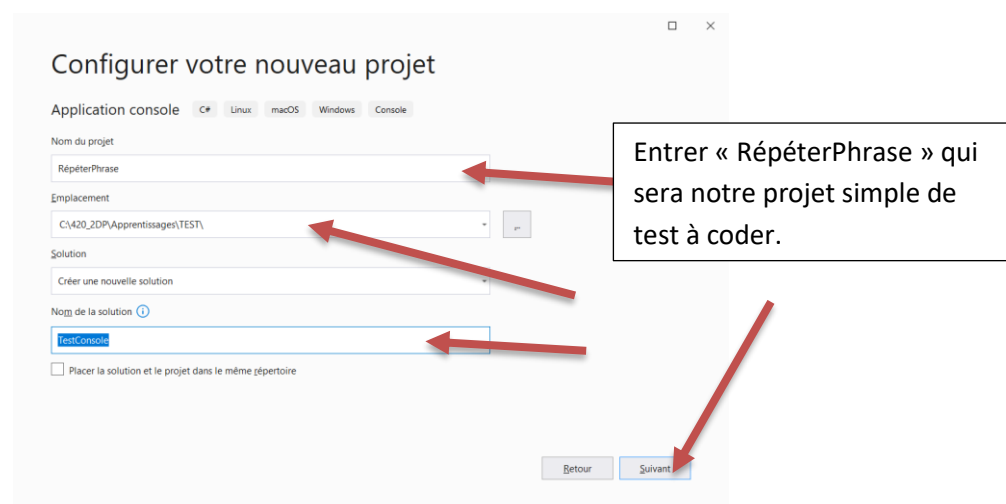
5. **Démarrer Visual Studio, choisir "Continuer sans code".**
6. Dans menu **GIT**, choisir "**Dépôts locaux...**" et choisir le projet TEST.



7. Pour ajouter une nouvelle solution et un nouveau projet: Dans le menu **Fichier/Nouveau**, choisir **Projet...**:

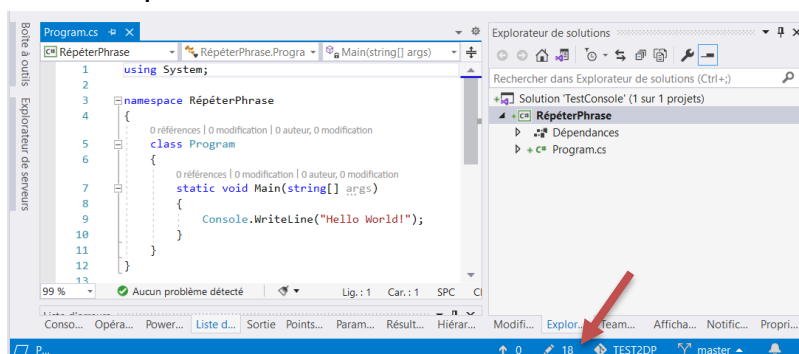


Sélectionner le type de projet et cliquer sur le bouton **Suivant**.

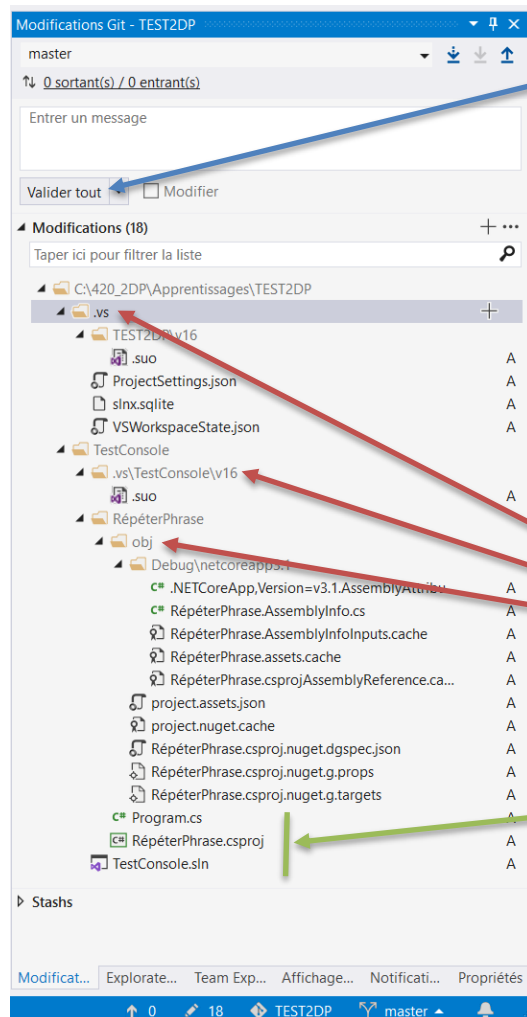


Cliquer sur le bouton **Suivant** et **Créer**.

8. Voici ce que vous devriez obtenir :



9. Sélectionner les éléments à soumettre dans le référentiel. Suivre les directives mentionnées par l'enseignant:



Valider Tout: Déposer les changements dans le **référentiel local**.

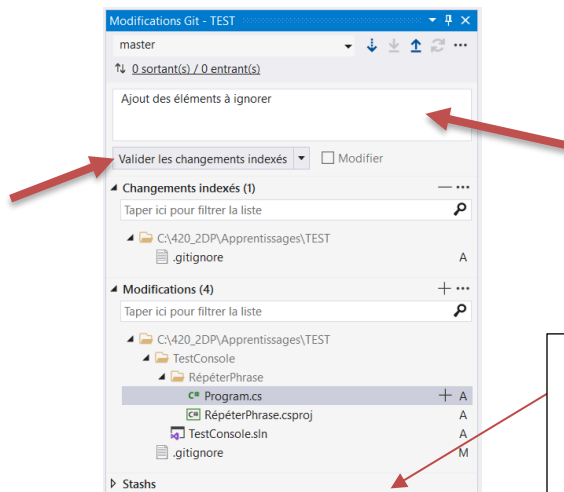
Valider tout et pousser: Déposer les changements dans le référentiel **local** et les envoyer dans le **central**.

Valider tout et synchroniser: Déposer les changements dans le référentiel **local**, récupérer (tirer) les modifications du **central** (peut avoir des conflits), et pousser les changements dans le **central**.

Ignorer ces éléments locaux: cliquer-droit sur l'élément.

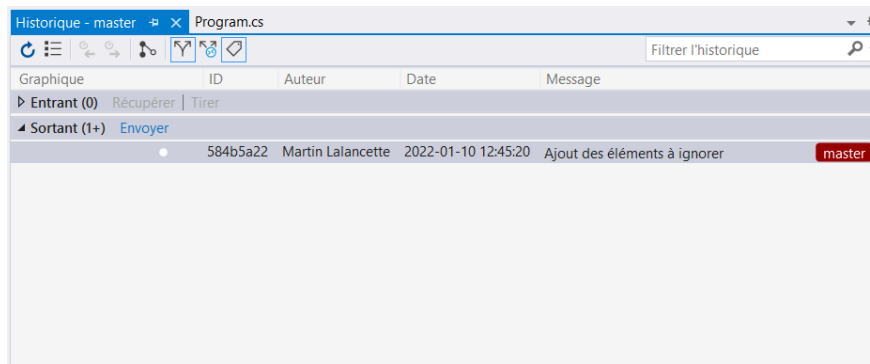
À soumettre dans le référentiel: Peu déjà l'être (lettre A). Choisir Indexer pour ajouter un élément qui ne l'est pas.

10. En ce moment, les fichiers ne sont pas encore ajoutés au **référentiel local** et au **référentiel central** à la branche *master*. Pour se faire, choisir **GIT/Commit ou stash**, entrer un commentaire et cliquer sur **Valider les changements indexés**.

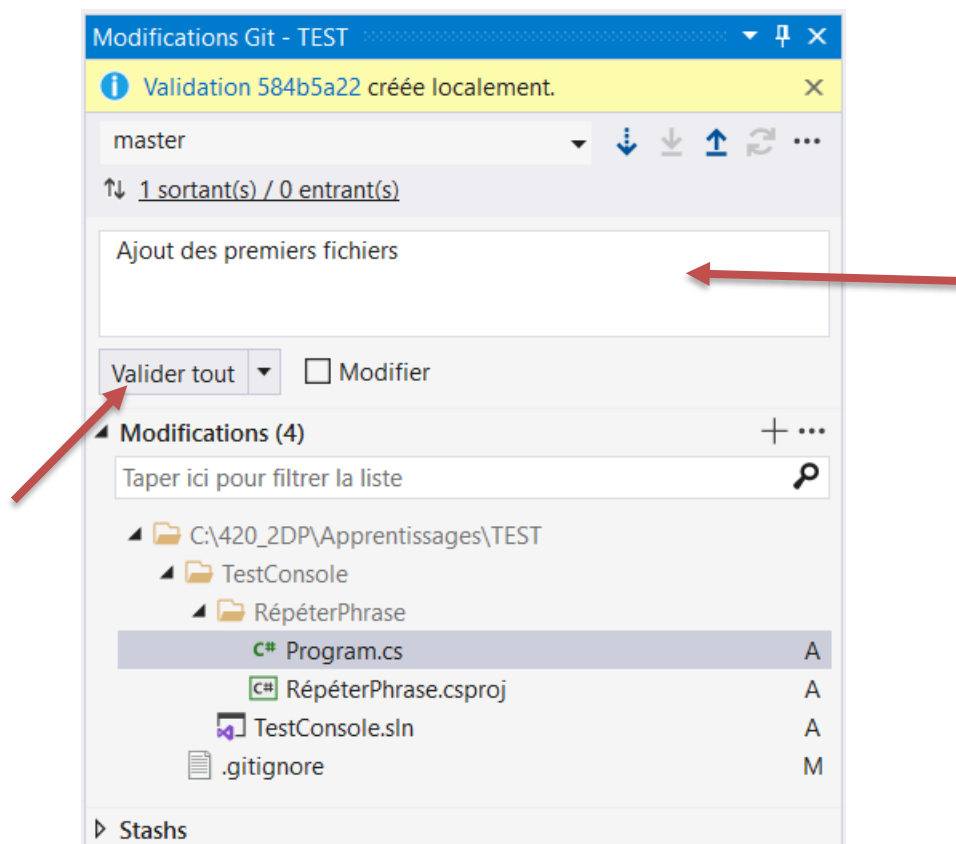


Stashes: Permet de conserver vos modifications que vous ne souhaitez pas soumettre tout de suite.

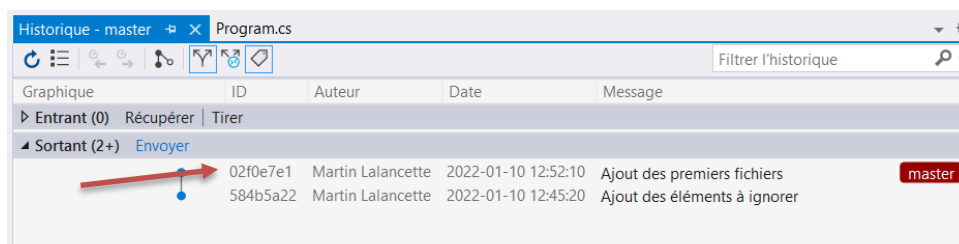
Ici, les fichiers à ignorer sont maintenant ajoutés au **référentiel local**.



11. Pour ajouter les fichiers du projet au référentiel local:



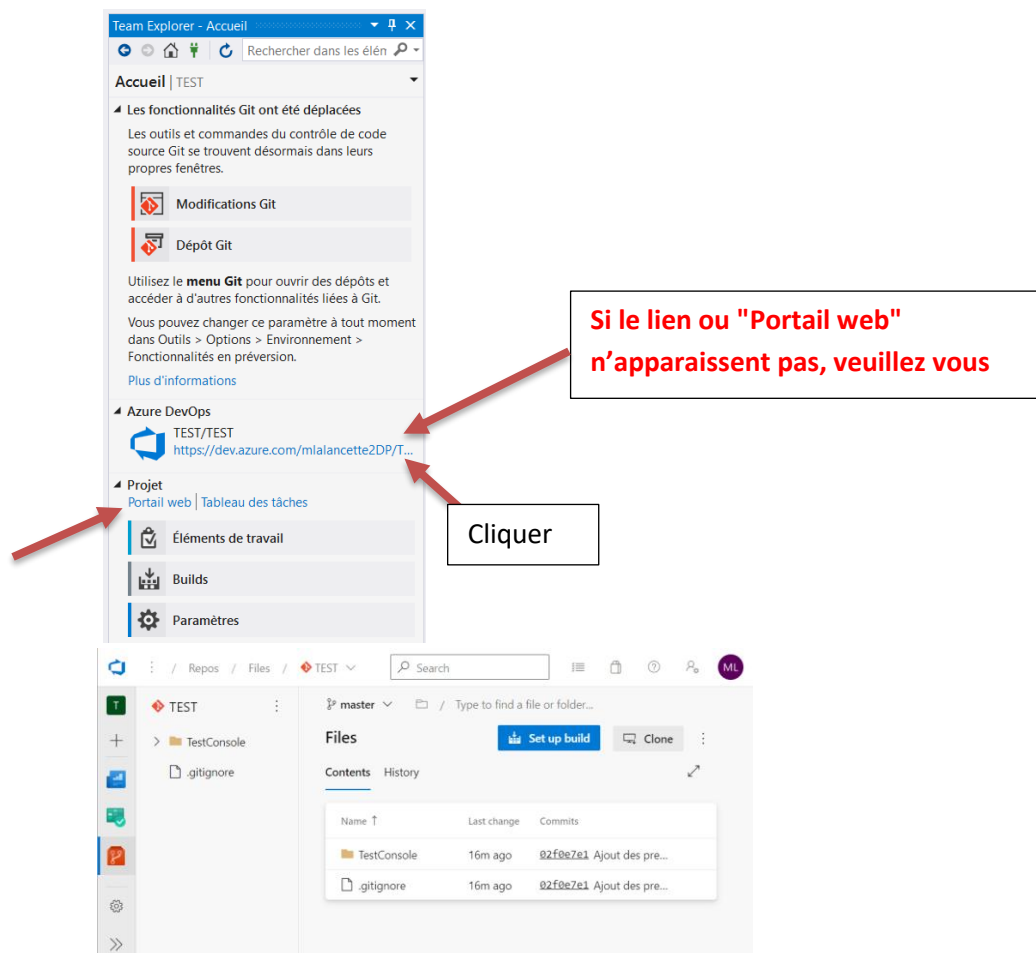
Ajouter un commentaire et cliquer sur le bouton **Valider Tout**.



12. Pour synchroniser avec le serveur, cliquer sur la flèche vers le haut :



13. Pour valider sur le serveur :



En principe, le développeur devrait valider (*commit*) ces modifications localement. Cela lui permet de revenir en arrière si des problèmes persistent. Une fois que le tout fonctionne adéquatement et il est prêt à offrir ces modifications aux autres, il peut alors les pousser (*push*). Il peut en profiter pour récupérer les modifications des autres en les tirant (*pull*) vers le référentiel local.

Les prochains exercices doivent être faits en séquence et en respectant exactement les étapes demandées et se feront avec la branche *master*.

Exercice 1. : Le développeur principal doit faire les actions suivantes:

- Modifier **Program.cs** pour ajouter une commande qui affiche « Entrer une phrase : » uniquement. Déposer le tout dans le référentiel **local** et **central** en entrant un commentaire. Valider que les modifications retrouvent sur DevOps Azure.

Exercice 2. : Le développeur principal doit faire les actions suivantes:

- Modifier **Program.cs** pour ajouter la **lecture de la réponse** de l'utilisateur après « Entrer une phrase ». Déposer le tout dans le référentiel **local seulement**.
- Modifier **Program.cs** pour modifier le texte suivant « Entrer une phrase **courte** : ». Mettre de côté cette modification.
- Synchroniser avec le référentiel central.

Exercice 3. : Le développeur principal doit faire les actions suivantes:

- Modifier le fichier Program.cs pour faire afficher 5 fois la phrase qui a été saisie. Déposer votre modification dans le référentiel **local** ainsi que **central**.
- Synchroniser avec le référentiel central.

Exercices en équipe de deux

Exercice 4. : En équipe de deux **obligatoire**, désigner le développeur **principal** et le développeur **secondaire**. Actions :

- Le développeur principal doit ajouter le développeur secondaire (utiliser le DA) comme membre de son projet TEST.
- Le développeur secondaire doit récupérer le projet en le clonant localement.

Exercice 5. : Le développeur principal doit faire les actions suivantes:

- Modifier **Program.cs** pour ajouter une commande qui affiche le titre dans la console suivant: "Répéter une phrase...". Déposer le tout dans le référentiel **local seulement** en entrant un commentaire.
- Demander au développeur secondaire de mettre à jour son référentiel local. Qu'arrive-t-il?

Exercice 6. : Le développeur principal doit maintenant déposer ses modifications dans le référentiel **central**. Faire les actions nécessaires.

- Demander au développeur secondaire de mettre à jour son référentiel local et valider que les modifications ont bel et bien été reçues.

Exercice 7. : Faire les prochaines actions en parallèle :

Principal : Modifier le fichier Program.cs pour faire afficher **10 fois** la phrase qui a été saisie. Déposer votre modification dans le référentiel local ainsi que central.

Secondaire : Modifier le fichier Program.cs pour **demandeur un nombre et faire afficher la phrase le nombre de fois spécifié par ce nombre**. Déposer votre modification dans le référentiel local ainsi que central.

Qu'arrive-t-il? → Conflit! → Voyons comment le résoudre.

Bibliographie

Azure DevOps. (2022, 01 10). *Référentiels GIT*. Récupéré sur Azure Repos la Documentation Git:
<https://docs.microsoft.com/fr-ca/azure/devops/repos/git/?view=azure-devops>

diagrams.net/draw.io. (2022, 01 05). *Usage terms for diagrams created in diagrams.net*.
Récupéré sur diagrams.net: <https://www.diagrams.net/doc/faq/usage-terms>