Travaux pratiques - Contrôle des versions logicielles avec Git

# Objectifs

Partie 1 : Lancer la DEVASC VM

Partie 2 : Initialisation de Git

Partie 3 : Mise en scène et validation d'un fichier dans le référentiel Git

Partie 4 : Gestion du fichier et suivi des modifications

Partie 5 : Succursales et fusion

Partie 6 : Gestion des conflits de fusion

Partie 7 : Intégration de Git avec GitHub

# Contexte/scénario

Dans ce TP, vous explorerez les principes fondamentaux du système de contrôle de version distribué Git, y compris la plupart des fonctionnalités que vous devez connaître pour collaborer à un projet logiciel. Vous allez également intégrer votre dépôt Git local avec le dépôt GitHub basé sur le cloud.

# Ressources requises

* 1 PC avec système d'exploitation de votre choix
* Boîte virtuelle ou VMWare
* Machine virtuelle DEVASC

# Instructions

## Lancer la machine virtuelle DEVASC

Si vous n'avez pas encore terminé **TP - Installer l'environnement de laboratoire de machines virtuelles**, faites-le maintenant. Si vous avez déjà terminé ce TP, lancez la machine virtuelle DEVASC maintenant.

## Initialisation en cours

Dans cette partie, vous allez initialiser un dépôt Git.

### Ouvrez un terminal dans le DEVASC-LABVM.

Double-cliquez sur l'icône de l'émulateur de terminal sur le bureau pour ouvrir une fenêtre de terminal.

### Initialiser un référentiel Git.

* + - 1. Utilisez la commande **ls** pour afficher une liste du répertoire en cours. Rappelez-vous que les commandes sont sensibles à la casse.

devasc@labvm:~$ **ls**

Desktop Downloads Music Public Templates

Documents labs Pictures snap Videos

devasc@labvm:~$

* + - 1. Ensuite, configurez les informations utilisateur à utiliser pour ce référentiel local. Cela associera vos informations au travail que vous contribuez à un référentiel local. Utilisez votre nom à la place de « Exemple d'utilisateur » pour le nom entre guillemets " ». Utilisez @example .com pour votre adresse e-mail.

**Remarque** : ces paramètres peuvent être tout ce que vous voulez à ce stade. Toutefois, lorsque vous réinitialisez ces valeurs globales dans la partie 7, vous utiliserez le nom d'utilisateur pour votre compte GitHub. Si vous le souhaitez, vous pouvez utiliser votre nom d'utilisateur GitHub maintenant.

devasc@labvm:~$ **git config --global user.name "SampleUser"**

devasc@labvm:~$ **git config --global user.email sample@example.com**

* + - 1. À tout moment, vous pouvez consulter ces paramètres à l'aide de la commande **git config —list**.

devasc@labvm:~$ **git config --list**

User.name=SampleUser

user.email=sample@example.com

devasc@labvm:~$

* + - 1. Utilisez la commande **cd** pour accéder au dossier **devnet-src** :

devasc@labvm:~$ **cd labs/devnet-src/**

devasc@labvm:~/labs/devnet-src$

* + - 1. Faites un répertoire **git-intro** et changez le répertoire dedans :

devasc@labvm:~/labs/devnet-src$ **mkdir git-intro**

devasc@labvm:~/labs/devnet-src$ **cd git-intro**

devasc@labvm:~/labs/devnet-src/git-intro$

* + - 1. Utilisez la commande **git init** pour initialiser le répertoire courant (git-intro) en tant que dépôt Git. Le message affiché indique que vous avez créé un référentiel local dans votre projet contenu dans le répertoire masqué **.git**. C'est là que se trouve l'historique de vos modifications. Vous pouvez le voir avec la commande **ls -a**.

devasc@labvm:~/labs/devnet-src/git-intro$ **git init**

Initialized empty Git repository in /home/devasc/labs/devnet-src/git-intro/.git/

devasc@labvm:~/labs/devnet-src/git-intro$ **ls -a**

. .. .git

devasc@labvm:~/labs/devnet-src/git-intro$

* + - 1. Lorsque vous travaillez sur votre projet, vous voudrez vérifier quels fichiers ont changé. Ceci est utile lorsque vous validez des fichiers dans le repo, et que vous ne voulez pas les valider tous. La commande **git status** affiche les fichiers modifiés dans le répertoire de travail qui sont mis en scène pour votre prochain commit.

Ce message vous indique :

* Que vous êtes sur le maître de branche. (Les branches sont discutées plus loin dans ce laboratoire)
* Le message de validation est la validation initiale.
* Il n'y a rien de changé à commettre.

Vous verrez que le statut de votre repo changera une fois que vous aurez ajouté des fichiers et que vous commencerez à apporter des modifications.

devasc@labvm:~/labs/devnet-src/git-intro$ **git status**

On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

devasc@labvm:~/labs/devnet-src/git-intro$

## Mise en place et validation d'un fichier dans le référentiel

Dans cette partie, vous allez créer un fichier, mettre en scène ce fichier et valider ce fichier dans le dépôt Git.

### Créez un fichier journal.

* + - 1. Le dépôt **git-intro** est créé mais vide. À l'aide de la commande **echo**, créez le fichier **DEVASC.txt** avec les informations contenues entre guillemets.

devasc@labvm:~/labs/devnet-src/git-intro$ **echo "I am on my way to passing the Cisco DEVASC exam" > DEVASC.txt**

devasc@labvm:~/labs/devnet-src/git-intro$

* + - 1. Utilisez la commande **ls -la** terify le fichier, ainsi que le répertoire .git, se trouvent dans le répertoire **git intro**. Ensuite, utilisez **cat** pour afficher le contenu de **DEVASC.txt**.

devasc@labvm:~/labs/devnet-src/git-intro$ **ls -la**

total 16

drwxrwxr-x 3 devasc devasc 4096 Apr 17 20:38 .

drwxrwxr-x 5 devasc devasc 4096 Apr 17 19:50 ..

-rw-rw-r-- 1 devasc devasc   48 Apr 17 20:38 DEVASC.txt

drwxrwxr-x 5 devasc devasc 4096 Apr 17 19:50 ..

evasc@labvm:~/src/git-intro$ **cat DEVASC.txt**

I am on my way to passing the Cisco DEVASC exam

devasc@labvm:~/labs/devnet-src/git-intro$

### Examinez l'état du référentiel.

Examinez l'état du référentiel en utilisant **git status**. Notez que Git a trouvé le nouveau fichier dans le répertoire, et sait qu'il n'est pas suivi.

devasc@labvm:~/labs/devnet-src/git-intro$ **git status**

On branch master

No commits yet

Untracked files:

  (use "git add <file>..." to include in what will be committed)

DEVASC.txt

nothing added to commit but untracked files present (use "git add" to track)

devasc@labvm:~/labs/devnet-src/git-intro$

### Mise en scène du fichier.

* + - 1. Ensuite, utilisez la commande **git add** pour « mettre en scène » le fichier **DEVASC.txt**. La mise en scène est une phase intermédiaire avant la validation d'un fichier dans le référentiel avec la commande **git commit**. Cette commande crée un instantané du contenu du fichier au moment de la saisie de cette commande. Toute modification du fichier nécessite une autre commande **git add** avant de valider le fichier.

devasc@labvm:~/labs/devnet-src/git-intro$ **git add DEVASC.txt**

* + - 1. En utilisant à nouveau la commande **git status**, notez les modifications mises en page affichées sous la forme « new file : DEVASC.txt ».

devasc@labvm:~/labs/devnet-src/git-intro$ **git status**

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file:   DEVASC.txt

devasc@labvm:~/labs/devnet-src/git-intro$

### Validation d'un fichier.

Maintenant que vous avez mis en scène vos modifications, vous devrez les valider afin de laisser Git savoir que vous voulez commencer à suivre ces modifications. Validez votre contenu en tant que nouvel instantané de validation à l'aide de la commande **git commit**. Le commutateur de **message*-m*** vous permet d'ajouter un message expliquant les modifications que vous avez apportées. Notez la combinaison de chiffres et de lettres en surbrillance dans la sortie. Ceci est l'ID de validation. Chaque commit est identifié par un hachage SHA1 unique. L'ID de validation est les 7 premiers caractères du hachage de validation complet. Votre ID de validation sera différent de celui affiché.

devasc@labvm:~/labs/devnet-src/git-intro$ **git commit -m "Committing DEVASC.txt to begin tracking changes"**

[master (root commit) b510f8e] Commettre DEVASC.txt pour commencer le suivi des modifications

 1 file changed, 1 insertion(+)

 create mode 100644 DEVASC.txt

devasc@labvm:~/labs/devnet-src/git-intro$

### Affichage de l'historique des validations.

Utilisez la commande **git log** pour afficher tous les validations dans l'historique de la branche courante. Par défaut, toutes les validations sont effectuées dans la branche maître. (Les branches seront discutées plus tard.) La première ligne est le hachage de validation avec l'ID de validation comme 7 premiers caractères. Le fichier est validé dans la branche maître. Ceci est suivi de votre nom et adresse e-mail, de la date du commit et du message que vous avez inclus avec le commit.

devasc@labvm:~/labs/devnet-src/git-intro$ **git log**

commit b510f8e5f9f63c97432d108a0413567552c07356 (HEAD -> master)

Author: Sample User <sample@example.com>

Date:   Sat Apr 18 18:03:28 2020 +0000

    Committing DEVASC.txt to begin tracking changes

devasc@labvm:~/labs/devnet-src/git-intro$

## Modification du fichier et suivi des modifications

Dans cette partie, vous allez modifier un fichier, mettre en scène le fichier, valider le fichier et vérifier les modifications dans le référentiel.

### Modifiez le fichier enregistré.

* + - 1. Effectuez une modification à DEVASC.txt à l'aide de la commande **echo**. Assurez-vous d'utiliser ">>" pour ajouter le fichier existant. Le ">" écrasera le fichier existant. Utilisez la commande **cat** pour afficher le fichier modifié.

devasc@labvm:~/labs/devnet-src/git-intro$ **echo "I am beginning to understand Git!"** DEVASC.txt

* + - 1. Utilisez la commande **cat** pour afficher le fichier modifié.

devasc@labvm:~/labs/devnet-src/git-intro$ **cat DEVASC.txt**

I am on my way to passing the Cisco DEVASC exam

I am beginning to understand Git!

devasc@labvm:~/labs/devnet-src/git-intro$

### Vérifiez la modification apportée au référentiel.

Vérifiez la modification dans le référentiel à l'aide de la commande **git status**.

devasc@labvm:~/labs/devnet-src/git-intro$ **git status**

On branch master

Changes not staged for commit:

  (use "git add <file>..." to include in what will be committed)

  (use "git restore <file>..." to discard changes in working directory)

modified:   DEVASC.txt

no changes added to commit (use "git add" and/or "git commit -a")

devasc@labvm:~/labs/devnet-src/git-intro$

### Déposez le fichier modifié.

Le fichier modifié devra être remis en scène avant de pouvoir être validé à l'aide de la commande **git add**.

devasc@labvm:~/labs/devnet-src/git-intro$ **git add DEVASC.txt**

### Valider le fichier de la mise en scène.

Valider le fichier intermédiaire à l'aide de la commande **git commit**. Remarquez le nouvel ID de validation.

devasc@labvm:~/labs/devnet-src/git-intro$ **git commit -m "Added additional line to file"**

[master 9f5c4c5] Added additional line to file

 1 file changed, 1 insertion(+)

devasc@labvm:~/labs/devnet-src/git-intro$

### Vérifiez les modifications apportées au référentiel.

* + - 1. Utilisez à nouveau la commande **git log** pour afficher toutes les validations. Notez que le journal contient l'entrée de validation d'origine ainsi que l'entrée de la validation que vous venez d'effectuer. Le dernier commit est affiché en premier. La sortie met en évidence l'ID de validation (7 premiers caractères du hachage SHA1), la date/heure de la validation et le message de la validation pour chaque entrée.

devasc@labvm:~/labs/devnet-src/git-intro$ **git log**

commit 9f5c4c5d630e88abe2a873fe48144e25ebe7bd6a (HEAD -> master)

Author: Sample User <sample@example.com>

Date:   Sat Apr 18 19:17:50 2020 +0000

    Added additional line to file

commit b510f8e5f9f63c97432d108a0413567552c07356

Author: Sample User <sample@example.com>

Date:   Sat Apr 18 18:03:28 2020 +0000

    Committing DEVASC.txt to begin tracking changes

devasc@labvm:~/labs/devnet-src/git-intro$

* + - 1. Lorsque vous avez plusieurs entrées dans le journal, vous pouvez comparer les deux validations en utilisant la commande **git diff** ajoutant d'abord l'ID de commit original et le dernier commit second : **git diff** <commit ID original> <commit ID latest>. Vous devrez utiliser vos identifiants de validation. Le signe "+" à la fin, suivi du texte, indique le contenu qui a été ajouté au fichier.

devasc@labvm:~/labs/devnet-src/git-intro$ **git diff b510f8e 9f5c4c5**

diff —git a/DEVASC.txt b/DEVASC.txt

index 93cd3fb.. 085273f 100644

— a/DEVASC.txt

+++ b/DEVASC.txt

@@ -1 +1,2 @@

 I am on my way to passing the Cisco DEVASC exam

+I am beginning to understand Git!

devasc@labvm:~/labs/devnet-src/git-intro$

## Succursales et fusion

Lorsqu'un référentiel est créé, les fichiers sont automatiquement placés dans une branche appelée **master**. Dans la mesure du possible, il est recommandé d'utiliser des branches plutôt que de mettre directement à jour la branche maître. La ramification (branchement) est utilisée pour que vous puissiez apporter des modifications dans une autre zone sans affecter la branche principale. Ceci est fait pour aider à prévenir les mises à jour accidentelles qui pourraient remplacer le code existant.

Dans cette partie, vous allez créer une nouvelle branche, extraire la branche, apporter des modifications dans la branche, mettre en scène et valider la branche, fusionner les modifications de branche dans la branche maître, puis supprimer la branche.

### Créer une nouvelle branche

Créez une nouvelle branche appelée **fonctionnalité** à l'aide de la <branch-name> commande **git branch**.

devasc@labvm:~/labs/devnet-src/git-intro$ **git branch feature**

### Vérifier la branche actuelle

Utilisez la commande **git branch** sans nom de branche pour afficher toutes les branches de ce référentiel. Le "\*" à côté de la branche principale indique qu'il s'agit de la branche actuelle — la branche actuellement « récupérée ».

devasc@labvm:~/labs/devnet-src/git-intro$ **git branch**

  feature

\* master

devasc@labvm:~/labs/devnet-src/git-intro$

### Commander la nouvelle succursale

Utilisez la <branch-name> commande **git checkout** pour basculer vers la branche de *feature*.

devasc@labvm:~/labs/devnet-src/git-intro$ **git checkout feature**

### Vérifier la branche actuelle

* + - 1. Vérifiez que vous avez basculé vers la branche *feature* à l'aide de la commande **git branch**. Notez le "\*" à côté de la branche de *feature*. C'est maintenant la *branche de travail*.

devasc@labvm:~/labs/devnet-src/git-intro$ **git branch**

\* feature

  master

devasc@labvm:~/labs/devnet-src/git-intro$

* + - 1. Ajoutez une nouvelle ligne de texte au fichier DEVASC.txt, à nouveau en utilisant la commande **echo** avec les signes ">>".

devasc@labvm:~/labs/devnet-src/git-intro$ **echo "This text was added originally while in the feature branch" >> DEVASC.txt**

* + - 1. Vérifiez que la ligne a été ajoutée au fichier à l'aide de la commande **cat**.

devasc@labvm:~/labs/devnet-src/git-intro$ **cat DEVASC.txt**

I am on my way to passing the Cisco DEVASC exam

I am beginning to understand Git!

This text was added originally while in the feature branch

devasc@labvm:~/labs/devnet-src/git-intro$

### Déposez le fichier modifié dans la branche de fonctionnalité.

* + - 1. Déposez le fichier mis à jour vers la branche *d'entité* active.

devasc@labvm:~/labs/devnet-src/git-intro$ **git add DEVASC.txt**

* + - 1. Utilisez la commande **git status** et notez que le fichier modifié **DEVASC.txt** est mis en scène dans la branche de *fonctionnalité*.

devasc@labvm:~/labs/devnet-src/git-intro$ **git status**

On branch feature

Changes to be committed:

  (use "git restore --staged <file>..." to unstage)

modified:   DEVASC.txt

devasc@labvm:~/labs/devnet-src/git-intro$

### Valider le fichier stagé dans la branche de fonctionnalité.

* + - 1. Valider le fichier intermédiaire à l'aide de la commande **git commit**. Notez le nouvel ID de validation et votre message.

devasc@labvm:~/labs/devnet-src/git-intro$ **git commit -m "Added a third line in feature branch"**

[feature cd828a7] Added a third line in feature branch

 1 file changed, 1 insertion(+)

devasc@labvm:~/labs/devnet-src/git-intro$

* + - 1. Utilisez la commande **git log** pour afficher toutes les validations, y compris le commit que vous venez de faire à la branche de *fonctionnalité*. La validation précédente a été effectuée dans la branche *master*.

devasc@labvm:~/labs/devnet-src/git-intro$ **git log**

commit cd828a73102cf308981d6290113c358cbd387620 (HEAD -> feature)

Author: Sample User <sample@example.com>

Date:   Sat Apr 18 22:59:48 2020 +0000

    Added a third line in feature branch

commit 9f5c4c5d630e88abe2a873fe48144e25ebe7bd6a (master)

Author: Sample User <sample@example.com>

Date:   Sat Apr 18 19:17:50 2020 +0000

    Added additional line to file

commit b510f8e5f9f63c97432d108a0413567552c07356

Author: Sample User <sample@example.com>

Date:   Sat Apr 18 18:03:28 2020 +0000

    Committing DEVASC.txt to begin tracking changes

devasc@labvm:~/labs/devnet-src/git-intro$

### Extraire la branche principale.

Passez à la branche master à l'aide de la commande **git checkout** *master* et vérifiez la branche de travail actuelle à l'aide de la commande **git branch** .

devasc@labvm:~/labs/devnet-src/git-intro$ **git checkout master**

Switched to branch 'master'

devasc@labvm:~/labs/devnet-src/git-intro$ **git branch**

  feature

\* master

devasc@labvm:~/labs/devnet-src/git-intro$

### Fusionner le contenu du fichier de la fonction vers la branche principale.

* + - 1. Les branches sont souvent utilisées lors de l'implémentation de nouvelles fonctionnalités ou correctifs. Ils peuvent être soumis pour examen par les membres de l'équipe, puis une fois vérifiés, ils peuvent être tirés dans la base de code principale — la branche maître.

Fusionnez le contenu (connu sous le nom d'historique) de la branche d'entité dans la branche maître à l'aide de la <branch-name> commande **git merge**. Le nom de branche est la branche à partir de laquelle les histoires sont extraites dans la branche actuelle. La sortie affiche qu'un fichier a été modifié avec une ligne insérée.

devasc@labvm:~/labs/devnet-src/git-intro$ **git merge feature**

Updating 9f5c4c5..cd828a7

fast-forward

 DEVASC.txt | 1 +

 1 file changed, 1 insertion(+)

devasc@labvm:~/labs/devnet-src/git-intro$

* + - 1. Vérifiez le contenu ajouté au fichier DEVASC.txt dans la branche maître à l'aide de la commande **cat**.

devasc@labvm:~/labs/devnet-src/git-intro$ **cat DEVASC.txt**

I am on my way to passing the Cisco DEVASC exam

I am beginning to understand Git!

This text was added originally while in the feature branch

devasc@labvm:~/labs/devnet-src/git-intro$

### Suppression d'une branche.

* + - 1. Vérifiez que la branche de **fonctionnalité** est toujours disponible à l'aide de la commande **git branch**.

devasc@labvm:~/labs/devnet-src/git-intro$ **git branch**

  feature

\* master

devasc@labvm:~/labs/devnet-src/git-intro$

* + - 1. Supprimez la branche de **feature** à l'aide de la <branch-name> commande **git branch -d**. .

devasc@labvm:~/labs/devnet-src/git-intro$ **git branch -d feature**

Deleted branch feature (was cd828a7).

devasc@labvm:~/labs/devnet-src/git-intro$

* + - 1. Vérifiez que la branche de fonctionnalité n'est plus disponible à l'aide de la commande **git branch**.

devasc@labvm:~/labs/devnet-src/git-intro$ **git branch**

\* master

devasc@labvm:~/labs/devnet-src/git-intro$

## Gestion des conflits de fusion

Parfois, vous pouvez rencontrer un conflit de fusion. C'est alors que vous avez peut-être apporté des modifications superposées à un fichier et que Git ne peut pas fusionner automatiquement les modifications.

Dans cette partie, vous allez créer une branche de test, modifier son contenu, mettre en scène et valider la branche test, basculer vers la branche maître, modifier à nouveau le contenu, mettre en scène et valider la branche maître, tenter de fusionner les branches, localiser et résoudre le conflit, mettre en scène et valider la branche maître à nouveau, et vérifier votre commit.

### Créez un nouveau test de branche.

Créez un nouveau **test**de branche.

devasc@labvm:~/labs/devnet-src/git-intro$ **git branch test**

### Vérification du test de branche.

* + - 1. Passer à la commande (passer à) le **test** de branche.

devasc@labvm:~/labs/devnet-src/git-intro$ **git checkout test**

Switched to branch 'test'

devasc@labvm:~/labs/devnet-src/git-intro$

* + - 1. Vérifiez que la branche de travail est la branche de **test**.

devasc@labvm:~/labs/devnet-src/git-intro$ **git branch**

  master

test

devasc@labvm:~/labs/devnet-src/git-intro$

### Vérifiez le contenu actuel de DEVASC.txt.

Vérifiez le contenu actuel du fichier **DEVASC.txt**. Notez que la première ligne comprend le mot "Cisco".

devasc@labvm:~/labs/devnet-src/git-intro$ **cat DEVASC.txt**

I am on my way to passing the Cisco DEVASC exam

I am beginning to understand Git!

This text was added originally while in the feature branch

devasc@labvm:~/labs/devnet-src/git-intro$

### Modifiez le contenu de DEVASC.txt dans la branche de test.

Utilisez la commande **sed** pour remplacer le mot "Cisco" par "NetAcad" dans le fichier **DEVASC.txt**.

devasc@labvm:~/labs/devnet-src/git-intro$ **sed -i 's/Cisco/NetAcad/' DEVASC.txt**

### Vérifiez le contenu du fichier DEVASC.txt modifié dans la branche de test.

Vérifiez la modification apportée au fichier **DEVASC.txt**.

devasc@labvm:~/labs/devnet-src/git-intro$ **cat DEVASC.txt**

I am on my way to passing the NetAcad DEVASC exam

I am beginning to understand Git!

This text was added originally while in the feature branch

devasc@labvm:~/labs/devnet-src/git-intro$

### Stage et validation de la branche de test.

Stage et validation du fichier avec une seule commande **git commit -a**. L'option **-a** n'affecte que les fichiers qui ont été modifiés et supprimés. Cela n'affecte pas les nouveaux fichiers.

devasc@labvm:~/labs/devnet-src/git-intro$ **git commit -a -m "Change Cisco to NetAcad"**

[test b6130a6] Change Cisco to NetAcad

 1 file changed, 1 insertion(+), 1 deletion(-)

  feature

### Checkout the master branch.

* + - 1. Checkout (switch to) the **master** branch.

devasc@labvm:~/labs/devnet-src/git-intro$ **git checkout master**

Switched to branch 'master'

devasc@labvm:~/labs/devnet-src/git-intro$

* + - 1. Vérifiez que la branche maître est votre branche de travail actuelle.

devasc@labvm:~/labs/devnet-src/git-intro$ **git branch**

\* master

test

devasc@labvm:~/labs/devnet-src/git-intro$

### Modifiez le contenu de DEVASC.txt dans la branche maître.

Utilisez la commande **sed** pour changer le mot "Cisco" en "DevNet" dans le fichier DEVASC.txt.

devasc@labvm:~/labs/devnet-src/git-intro$ **sed -i 's/Cisco/DevNet/' DEVASC.txt**

### Vérifiez le contenu du fichier DEVASC.txt modifié dans la branche maître.

Vérifiez la modification apportée au fichier.

devasc@labvm:~/labs/devnet-src/git-intro$ **cat DEVASC.txt**

I am on my way to passing the DevNet DEVASC exam

I am beginning to understand Git!

This text was added originally while in the feature branch

devasc@labvm:~/labs/devnet-src/git-intro$

### Stage et validation de la branche maître

Stage et validation du fichier à l'aide de la commande **git commit -a**.

devasc@labvm:~/labs/devnet-src/git-intro$ **git commit -a -m "Changed Cisco to DevNet"**

[master 72996c0] Changed Cisco to DevNet

 1 file changed, 1 insertion(+), 1 deletion(-)

devasc@labvm:~/labs/devnet-src/git-intro$

### Essayez de fusionner la branche de test dans la branche maître.

Essayez de fusionner l'historique de la branche de test dans la branche maître.

devasc@labvm:~/labs/devnet-src/git-intro$ **git merge test**

Auto-merging DEVASC.txt

CONFLICT (content): Merge conflict in DEVASC.txt

Automatic merge failed; fix conflicts and then commit the result.

devasc@labvm:~/labs/devnet-src/git-intro$

### Trouve le conflit.

* + - 1. Utilisez la commande **git log** pour afficher les validations. Notez que la version HEAD est la branche principale. Cela sera utile dans la prochaine étape.

devasc@labvm:~/labs/devnet-src/git-intro$ **git log**

commit 72996c09fa0ac5dd0b8ab9ec9f8530ae2c5c4eb6 (HEAD -> master)

Author: Sample User <sample@example.com>

Date:   Sun Apr 19 00:36:05 2020 +0000

Changement de Cisco en DevNet

<output omitted>

* + - 1. Utilisez la commande **cat** pour afficher le contenu du fichier DEVASC.txt. Le fichier contient maintenant des informations pour vous aider à trouver le conflit. La version HEAD (branche principale) contenant le mot "DevNet" est en conflit avec la version de la branche test et le mot "NetAcad".

devasc@labvm:~/labs/devnet-src/git-intro$ **cat DEVASC.txt**

<<<<<<< HEAD

I am on my way to passing the DevNet DEVASC exam

=======

I am on my way to passing the NetAcad DEVASC exam

>>>>>>> test

I am beginning to understand Git!

This text was added originally while in the feature branch

devasc@labvm:~/labs/devnet-src/git-intro$

### Modifiez manuellement le fichier DEVASC.txt pour supprimer le texte en conflit.

* + - 1. Utilisez la commande **vim** pour modifier le fichier.

devasc@labvm:~/labs/devnet-src/git-intro$ **vim DEVASC.txt**

* + - 1. Utilisez les flèches haut et bas pour sélectionner la ligne de texte appropriée. Appuyez sur **dd** (delete) sur les lignes suivantes qui sont mises en surbrillance. **dd** supprimera la ligne sur laquelle se trouve le curseur.

<<<<<<< HEAD

I am on my way to passing the DevNet DEVASC exam

=======

I am on my way to passing the NetAcad DEVASC exam

>>>>>>> test

I am beginning to understand Git!

This text was added originally while in the feature branch

* + - 1. Enregistrez vos modifications dans vim en appuyant sur **ESC** (la touche d'échappement), puis en tapant **:** (deux points) suivi de **wq** et en appuyant sur entrée.

**ESC**

**:**

**wq**

**<Enter or Return>**

### Vérifiez vos modifications de DEVASC.txt dans la branche maître.

Vérifiez que vous avez modifié à l'aide de la commande **cat**.

devasc@labvm:~/labs/devnet-src/git-intro$ **cat DEVASC.txt**

I am on my way to passing the DevNet DEVASC exam

I am beginning to understand Git!

This text was added originally while in the feature branch

devasc@labvm:~/labs/devnet-src/git-intro$

### Stage et validation de la branche maître

Stage et validation DEVASC.txt dans la branche maître à l'aide de la commande **git commit -a**.

devasc@labvm:~/labs/devnet-src/git-intro$ **git add DEVASC.txt**

devasc@labvm:~/labs/devnet-src/git-intro$ **git commit -a -m "Manually merged from test branch"**

[master 22d3da4] Manually merged from test branch

devasc@labvm:~/labs/devnet-src/git-intro$

### Vérifiez la validation.

Utilisez la commande **git log** pour vérifier la validation. Si nécessaire, vous pouvez utiliser **q** pour quitter l'affichage git log.

devasc@labvm:~/labs/devnet-src/git-intro$ **git log**

commit 22d3da41e00549ce69dc145a84884af6a1697734 (HEAD -> master)

Merge: 72996c0 b6130a6

Author: Sample User <sample@example.com>

Date:   Sun Apr 19 00:36:05 2020 +0000

**manually merged from branch test**

<output omitted>

## Intégration de Git avec GitHub

Jusqu'à présent, toutes les modifications que vous avez apportées à votre fichier ont été stockées sur votre machine locale. Git s'exécute localement et ne nécessite aucun serveur de fichiers central ou service d'hébergement basé sur le cloud. Git permet à un utilisateur de stocker et de gérer localement des fichiers.

Bien que Git soit utile pour un seul utilisateur, l'intégration du référentiel Git local avec un serveur basé sur le cloud comme GitHub est utile lorsque vous travaillez au sein d'une équipe. Chaque membre de l'équipe conserve une copie sur le référentiel sur sa machine locale et met à jour le référentiel central basé sur le cloud pour partager les modifications éventuelles.

Il y a pas mal de services Git populaires, y compris GitHub, Stash d'Atlassian et GitLab. Parce qu'il est facilement accessible, vous utiliserez GitHub dans ces exemples.

### Créez un compte GitHub.

Si vous ne l'avez pas déjà fait, rendez-vous sur github.com et créez un compte GitHub. Si vous avez un compte GitHub passez à l'étape 2.

### Connectez-vous à votre compte GitHub Créer un référentiel.

Connectez-vous à votre compte GitHub.

### Créer un référentiel

* + - 1. Sélectionnez le bouton "**New repository**" ou cliquez sur l'icône "**+**" dans le coin supérieur droit et sélectionnez "**New repository**".
      2. Créez un référentiel à l'aide des informations suivantes :

Nom du référentiel: **devasc-study-team**

Description: **Travailler ensemble pour réussir l'examen DEVASC**

Public/Private: **Private**

* + - 1. Sélectionner : **Créer un référentiel**

### Créer un nouveau répertoire devasc-study-team.

* + - 1. Si vous n'êtes pas déjà dans le répertoire **git-intro**, changez-y maintenant.

devasc@labvm:~$ **cd ~/labs/devnet-src/git-intro**

* + - 1. Créez un nouveau répertoire appelé devasc-study-team. Le répertoire n'a pas besoin de correspondre au nom en tant que référentiel.

devasc@labvm:~/labs/devnet-src/git-intro$ **mkdir devasc-study-team**

### Changer le répertoire en devasc-study-team.

Utilisez la commande **cd** pour changer les répertoires en **devasc-study-team**.

devasc@labvm:~/labs/devnet-src/git-intro$ **git log**

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$

### Copiez le fichier DEVASC.

* + - 1. Utilisez la commande **cp** pour copier le fichier **DEVASC.txt** du répertoire parent **git-intro** vers le sous-répertoire **devasc-study-team**. Les deux points et une barre oblique précédant le nom du fichier indiquent le répertoire parent. L'espace et la période qui suivent le nom du fichier indiquent de copier le fichier dans le répertoire actif avec le même nom de fichier.

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$ **cp ../DEVASC.txt .**

* + - 1. Vérifiez que le fichier a été copié avec la commande **ls** et le contenu du fichier avec la commande **cat**.

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$ **ls**

DEVASC.txt

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$ **cat DEVASC.txt**

I am on my way to passing the DevNet DEVASC exam

I am beginning to understand Git!

This text was added originally while in the feature branch

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$

### Initialisez un nouveau dépôt Git.

* + - 1. Utilisez la commande **git init** pour initialiser le répertoire courant (devasc-study-team) en tant que dépôt Git. Le message affiché indique que vous avez créé un référentiel local dans votre projet contenu dans le répertoire masqué **.git**. C'est là que se trouve l'historique de vos modifications.

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$ **git init**

Initialized empty Git repository in /home/devasc/src/git-intro/devasc-study-team/.git/

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$

* + - 1. Ensuite, vérifiez vos variables git globales avec la commande **git config —list**.

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$ **git config --list**

User.name=SampleUser

user.email=sample@example.com

core.repositoryformatversion=0

core.filemode=true

core.bare=false

core.logallrefupdates=true

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$

* + - 1. Si les variables user.name et user.email ne correspondent pas à vos informations d'identification GitHub, modifiez-les maintenant.

devasc@labvm:~$ **git config --global user.name "***GitHub username***"**

devasc@labvm:~$ **git config --global user.email** *GitHub-email-address*

### Dépôt Git Point vers le dépôt GitHub.

* + - 1. Utilisez la commande **git remote add** pour ajouter une URL Git comme alias distant. La valeur "origin" pointe vers le dépôt nouvellement créé sur GitHub. Utilisez votre nom d'utilisateur GitHub dans le chemin d'accès URL pour *github-username*.

**Remarque** : Votre nom d'utilisateur est sensible à la casse.

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$ **git remote add origin https://github.com/*github-username*/devasc-study-team.git**

* + - 1. Vérifiez que le **remote** est en cours d'exécution sur github.com.

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$ **git remote --verbose**

origin https://github.com/username/devasc-study-team.git (fetch)

origin https://github.com/username/devasc-study-team.git (push)

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$

* + - 1. Affichez le **git log**. L'erreur indique qu'il n'y a pas de commits.

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$ **git log**

fatal: your current branch 'master' does not have any commits yet

### Stage et validation du fichier DEVASC.txt.

* + - 1. Utilisez la commande **git add** pour mettre en scène le fichier DEVASC.txt.

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$ **git add DEVASC.txt**

* + - 1. Utilisez la commande **git commit** pour valider le fichier DEVASC.txt.

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$ **git commit -m "Add DEVASC.txt file to devasc-study-team"**

[master (root-commit) c60635f] Add DEVASC.txt file to devasc-study-team

 1 file changed, 3 insertions(+)

 create mode 100644 DEVASC.txt

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$

### Vérifiez la validation.

* + - 1. Utilisez la commande **git log** pour vérifier la validation.

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$ **git log**

commit c60635fe4a1f85667641afb9373e7f49a287bdd6 (HEAD -> master)

Author: username <user@example.com>

Date : lun Apr 20 02:48:21 2020 +0000

    Add DEVASC.txt file to devasc-study-team

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$

* + - 1. Utilisez la commande **git status** pour afficher les informations d'état. L'expression "working tree clean" signifie que Git a comparé votre liste de fichiers à ce que vous avez dit à Git, et c'est une ardoise propre avec rien de nouveau à signaler.

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$ **git status**

On branch master

nothing to commit, working tree clean

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$

### Send (push) the file from Git to GitHub.

Utilisez la commande **git push origin master** pour envoyer (pousser) le fichier à votre dépôt GitHub. Vous serez invité à entrer un nom d'utilisateur et un mot de passe, qui sera celui que vous avez utilisé pour créer votre compte GitHub.

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$ **git push origin master**

Username for 'https://github.com': ***username***

Mot de passe pour 'https://username@github.com' :***mot de passe***

Enumerating objects: 3, done.

Counting objects: 100% (9/9), done.

Delta compression using up to 2 threads

Compressing objects: 100% (6/6), done.

Writing objects: 100% (6/6), 748 bytes | 748.00 KiB/s, done.

Total 8 (delta 0), reused 0 (delta 0)

To https://github.com/username/devasc-study-team.git

\* [nouvelle branche] maître -> maître

devasc@labvm:~/labs/devnet-src/git-intro/devasc-study-team$

**Remarque** : Si, après avoir entré votre nom d'utilisateur et votre mot de passe, vous obtenez une erreur fatale indiquant que le dépôt est introuvable, vous avez probablement soumis une URL incorrecte. Vous devrez inverser votre commande **git add** avec la commande **git remote rm origin**.

### Vérifiez le fichier sur GitHub.

* + - 1. Accédez à votre compte GitHub et sous « Référentiels » sélectionnez le nom d' **utilisateur/devasc-study-team**.
      2. Vous devriez voir que le fichier DEVASC.txt a été ajouté à ce dépôt GitHub. Cliquez sur le fichier pour afficher le contenu.

Fin du document