Contrôle du code source

Guy Tremblay
Professeur

Département d'informatique
UQAM
http://www.labunix.uqam.ca/~tremblay_gu

INF600A
2 octobre 2018

Contenu

Les premières choses à faire, quand on veut développer du code de façon professionnelle. . .

« *You need to get the development infrastructure environment in order. That means adopting (or improving) the fundamental Starter Kit practices :*

- *?*
- *?*
- *?*

? *needs to come before anything else. It's the first bit of infrastructure we set up on any project.* »

« *Practices of an Agile Developer—Working in the Real World* »,
Subramaniam & Hunt, 2006.

Les premières choses à faire, quand on veut développer du code de façon professionnelle...

« *You need to get the development infrastructure environment in order. That means adopting (or improving) the fundamental Starter Kit practices :*

- *Version control*
- *Unit Testing*
- *Build automation*

Version control needs to come before anything else. It's the first bit of infrastructure we set up on any project. »

« *Practices of an Agile Developer—Working in the Real World* »,
Subramaniam & Hunt, 2006.

1. Qu’est-ce qu’un système de contrôle du code source ?

Qu’est-ce qu’un système de contrôle du code source ?

?

Qu’est-ce qu’un système de contrôle du code source ?

Qu’est-ce qu’un système de contrôle du code source ?

A source code control system [is like] a giant **UNDO** *key—a project-wide time machine that can return you to those alcyon days of last week, when the code actually compiled and ran.*

«The Pragmatic Programmer», Hunt & Thomas, 2000.

Qu’est-ce qu’un système de contrôle du code source ?

A source code control system [is like] a giant **UNDO** *key—a project-wide time machine that can return you to those alcyon days of last week, when the code actually compiled and ran.*

«The Pragmatic Programmer», Hunt & Thomas, 2000.

Alcyon

1. *Calm and peaceful ; tranquil.*
2. *Prosperous ; golden : halcyon years.*

Ce que permet de faire un système de contrôle du code source

- Conserver tout le code source

Ce que permet de faire un système de contrôle du code source

- Conserver tout le code source
- Prendre note de tous les changements effectués au code source et à sa documentation

Ce que permet de faire un système de contrôle du code source

- Conserver tout le code source
- Prendre note de tous les changements effectués au code source et à sa documentation $\Rightarrow$ permet de retourner à une version antérieure (qui, elle, fonctionnait ! !)

Ce que permet de faire un système de contrôle du code source

- Conserver tout le code source
- Prendre note de tous les changements effectués au code source et à sa documentation $\Rightarrow$ permet de retourner à une version antérieure (qui, elle, fonctionnait ! !)

- Identifier quels fichiers ont été modifiés
- Déterminer qui a modifié un bout de code
- Comparer des versions
- Fusionner des versions développées de façon concurrente

Ce que permet de faire un système de contrôle du code source

- Conserver tout le code source
- Prendre note de tous les changements effectués au code source et à sa documentation $\Rightarrow$ permet de retourner à une version antérieure (qui, elle, fonctionnait ! !)

- Identifier quels fichiers ont été modifiés
- Déterminer qui a modifié un bout de code
- Comparer des versions
- Fusionner des versions développées de façon concurrente

- Identifier et gérer les *releases*, les versions, les branches de développement

Les différents SCCS

Qui a déjà utilisé ou utilise couramment. . . ?

CVS
Subversion (svn)

Git
Mercurial

?

Les différents SCCS

Qu'est-ce qui différencie… ?

CVS
Subversion (svn)

Git
Mercurial

?

Les différents SCCS

Qu'est-ce qui différencie… ?

Centralisé	Distribué
CVS	Git
Subversion (svn)	Mercurial

!

SCCS Centralisé vs. SCCS Distribué

Centralisé

Tout l'historique des changements est conservé sur un serveur central (distant), duquel n'importe qui peut obtenir la version la plus récente ou envoyer les changements les plus récents.

SCCS Centralisé vs. SCCS Distribué

Centralisé

Tout l'historique des changements est conservé sur un serveur central (distant), duquel n'importe qui peut obtenir la version la plus récente ou envoyer les changements les plus récents.

Distribué

Chaque usager a une copie **locale** de tout l'historique des changements. Il n'est donc pas nécessaire d'être connecté à un réseau/serveur pour sauvegarder des changements. En outre, n'importe quel usager peut se synchroniser avec n'importe quel autre.

SCCS Centralisé vs. SCCS Distribué

Centralisé

Centralized VCS systems are designed with the intent that there is One True Source that is Blessed, and therefore Good. All developers work (checkout) from that source, and then add (commit) their changes, which then become similarly Blessed.

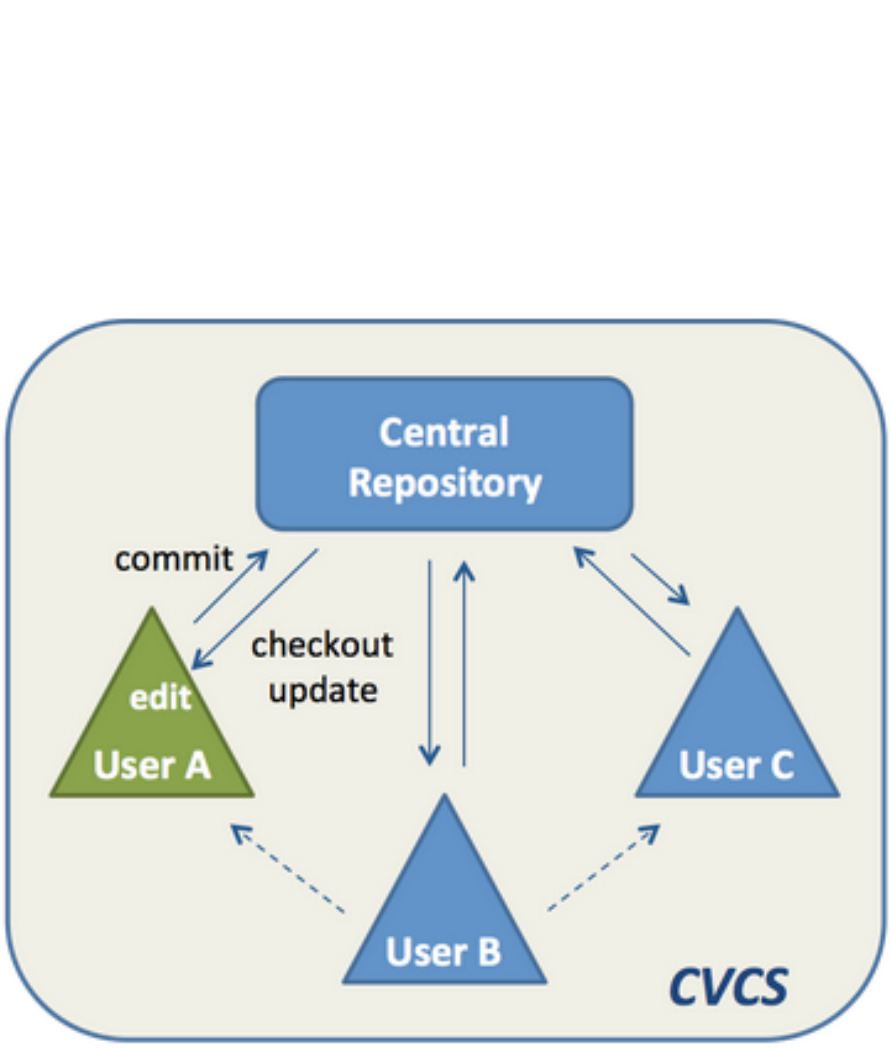
Distribué

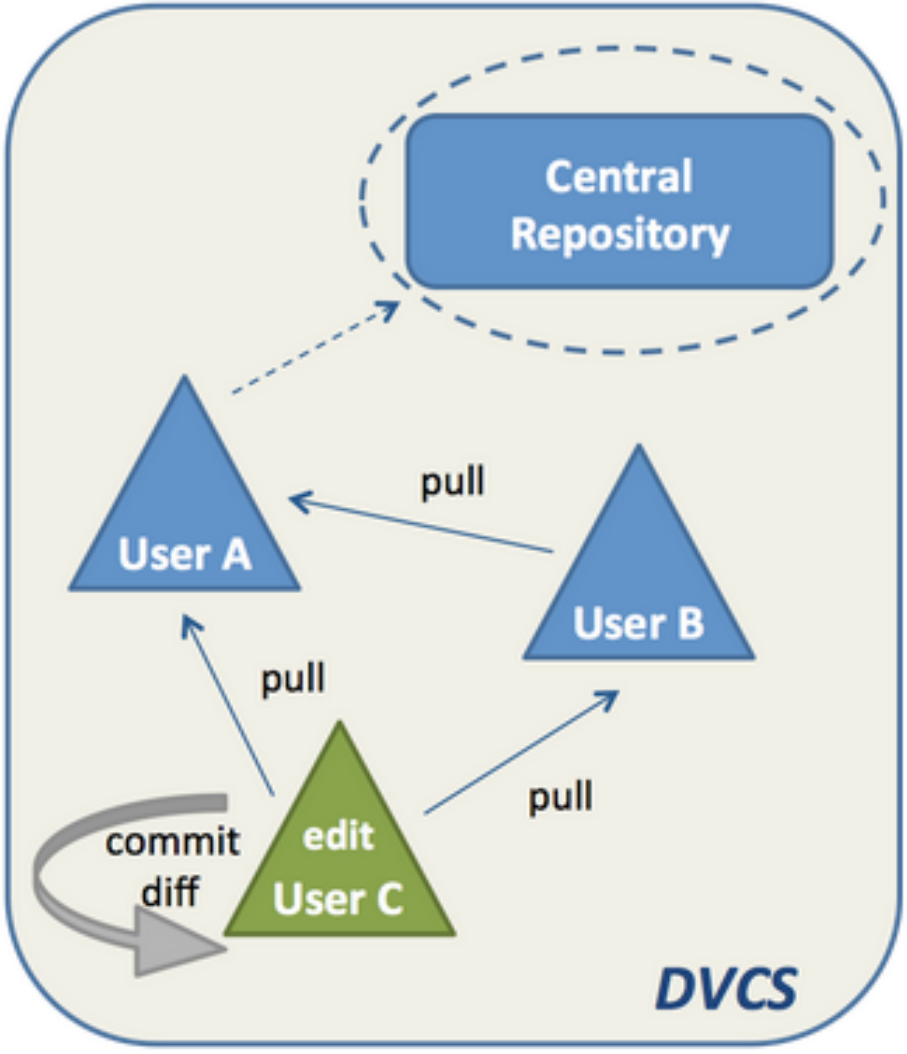
Distributed VCS systems are designed with the intent that one repository is as good as any other, and that merges from one repository to another are just another form of communication.

Source: `http://stackoverflow.com/questions/111031/comparison-between-centralized-and-distributed-version-control-systems`

SCCS Centralisé vs. SCCS Distribué

Source : https://www.appfusions.com/display/StashSCMImporter/CVCS+vs.+DVCS+In+a+Nutshell





Avantages/désavantages des SCCS distribués

Avantages

- Chaque développeur a son espace privé — son *sandbox*
- On peut travailler — et faire des *commits* ! — hors ligne
- Plusieurs opérations sont très rapides — exécution locale
- La création et fusion de branches est efficace

Avantages/désavantages des SCCS distribués

Avantages

- Chaque développeur a son espace privé — son *sandbox*
- On peut travailler — et faire des *commits* ! — hors ligne
- Plusieurs opérations sont très rapides — exécution locale
- La création et fusion de branches est efficace

Désavantages

- Quand même préférable d'avoir un dépôt central (sauvegarde)
- Pas nécessairement «une» version «la plus récente»

Source:

`https://www.appfusions.com/display/StashSCMImporter/CVCS+vs.+DVCS+In+a+Nutshell`

2. Utilisation de `git` dans le cours

Dans le cadre des devoirs, vous devrez utiliser `git`

Dans le cadre des devoirs, vous devrez utiliser `git`

Pour gérer la progression de votre solution et assurer sa non-régression

Dans le cadre des devoirs, vous devrez utiliser `git`

Pour gérer la progression de votre solution et assurer sa non-régression

Même si vous travaillez seul !

Dans le cadre des devoirs, vous devrez utiliser `git`

Pour gérer la progression de votre solution et assurer sa non-régression

Même si vous travaillez seul !

Et «j'évaluerai» votre utilisation de `git` !

Dans le cadre des devoirs, vous devrez utiliser `git`

Pour gérer la progression de votre solution et assurer sa non-régression

Même si vous travaillez seul !

Et «j'évaluerai» votre utilisation de `git` !
$\Rightarrow$ Pas juste un unique gros `commit` le jour de la remise !

Dans le cadre des devoirs, vous devrez utiliser `git`

Pour gérer la progression de votre solution et assurer sa non-régression

Même si vous travaillez seul !

Et «j'évaluerai» votre utilisation de `git` !
$\Rightarrow$ Pas juste un unique gros `commit` le jour de la remise !

Note : Nb. de *commits* pour ma solution au devoir #1 (avec les tests) = 105

Différentes possibilités quant à la façon dont vous pouvez utiliser `git`

- Vous travaillez **seul** et toujours sur `java.labunix.uqam.ca` :

  Faites des *commits* dans le dépôt `GestionVins` que vous avez cloné, sans jamais faire de `push`

Différentes possibilités quant à la façon dont vous pouvez utiliser `git`

- Vous travaillez seul et toujours sur `java.labunix.uqam.ca` :

  Faites des *commits* dans le dépôt `GestionVins` que vous avez cloné, sans jamais faire de `push`

- Vous travaillez seul mais sur différentes machines ou vous travaillez en équipe et vous voulez partager le dépôt :
  1. Dépôt partagé sur `java.labunix.uqam.ca`
  2. Dépôt partagé sur `BitBucket`
  3. Dépôt partagé sur `GitHub`

1. Pour créer un dépôt sur `java.labunix`

Pour copie distante ou partage du code entre coéquipiers-ères

Connectez vous à votre compte sur `java.labunix.uqam.ca`.
Exemple : `tremblay_gu`

```
$ cd ~
$ mkdir DepotsGit
$ cd DepotsGit
$ git init --bare GestionVins.git
```

Toujours sur `java.labunix.uqam.ca` :

```
$ cd ~/GestionVins
$ git remote rm origin
$ git remote add origin\
ssh://tremblay_gu@java.labunix.uqam.ca/home/tremblay_gu/\
DepotsGit/GestionVins.git
$ git push -u origin master
```

Note : Les «\» en fin de ligne ne sont présents que parce la commande indiquée est plus longue que la largeur de la diapositive ⇒ écrire collé !

2. Pour créer un dépôt BitBucket

Pour copie distante ou partage du code entre coéquipiers-ères

Connectez vous à votre compte BitBucket.
Exemple : tremblay_g

- Dans *Repositories*, *Create Repository* ⇒ gestion-vins [a]
- Dépôt *Private* (possible, par défaut !)

---

a. Le nom ne doit pas contenir de majuscules ☹

Sur java.labunix.uqam.ca :

```
$ cd GestionVins

$ git remote rm origin

$ git remote add origin\
https://tremblay_g@bitbucket.org/tremblay_g/\
gestion-vins.git

$ git push -u origin master
```

3. Pour créer un dépôt GitHub

Pour copie distante ou partage du code entre coéquipiers-ères

Connectez vous à votre compte GitHub — par exemple, dans mon cas = tremblay-guy.

- *Create a new repository* ⇒ GestionVins
- Sans README ou rien !

Sur java.labunix.uqam.ca :

```
$ cd GestionVins

$ git remote rm origin

$ git remote add origin\
https://github.com/tremblay-guy/GestionVins.git

$ git push -u origin master
```

Sauf que, par défaut. . . les dépôts GitHub sont publics ☹

Si vous ne connaissez pas `git`...

- Notes de cours sur `git` :

  `http://www.labunix.uqam.ca/~tremblay_gu/INF600A/Materiel/git.pdf`

- Divers liens sur `git` :

  `http://www.labunix.uqam.ca/~tremblay_gu/INF600A/Liens/`

3. Quelques recommandations d’utilisation

Tip 23

Always Use Source Code Control

Always. Even if you are a single-person team on a one-week project. Even if it's a "throw-away" prototype. Even if the stuff you're working on isn't source code. Make sure that everything is under source code control !

Tip 23

Always Use Source Code Control

Always. Even if you are a single-person team on a one-week project. Even if it's a "throw-away" prototype. Even if the stuff you're working on isn't source code. Make sure that everything is under source code control !

Tip 3

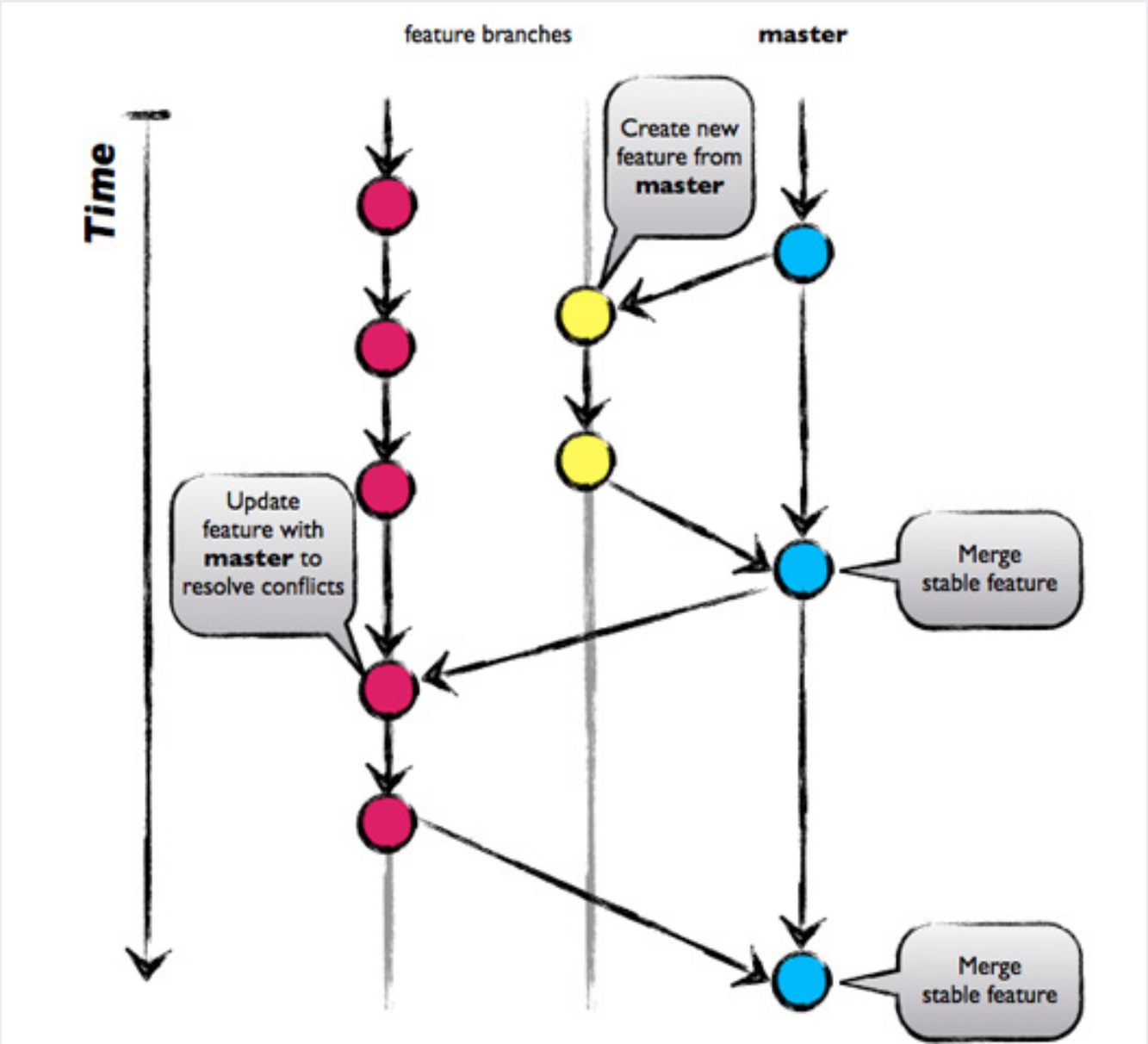
If you need it, check it in

If your development shop goes up in flames, you should be able to recover with just a backup of your repository. You should have everything you need to build the entire project ; if you don't, then perhaps you aren't using the tool properly.
[. . .]
The only exception to the «Keep everything you need to build your product in the SCM» top are files that you can generate.

Note : Par exemple, si on travaille en Java, on exclut les fichiers `.class` !

Le modèle GitHub flow (simplifié) illustré

Source : http://www.nicoespeon.com/fr/2013/08/quel-git-workflow-pour-mon-projet/



Le modèle GitHub flow (simplifié)

Source : `http://www.nicoespeon.com/fr/2013/08/quel-git-workflow-pour-mon-projet/`

1. Tout ce qui est dans `master` peut être déployé en production
2. Créer des branches explicites depuis `master` (*features*)
3. Pousser sur `origin` régulièrement
4. Ouvrir une *pull-request* à tout moment
5. Fusionner seulement après une *pull-request review*
6. Déployer immédiatement après `merge` dans `master`

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even days —between code check-ins,
  you aren't using the system actively.

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even heures—between code check-ins,
  you aren't using the system actively.

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even heures—between code check-ins, you aren't using the system actively.

- *If the hard drive on your workstation crashed right now, how much work yould you lose ?*

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even heures—between code check-ins, you aren't using the system actively.
- *If the hard drive on your workstation crashed right now, how much work yould you lose ?*
  If it's more than a day or two , consider changing the way you work.

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even heures—between code check-ins, you aren't using the system actively.
- *If the hard drive on your workstation crashed right now, how much work yould you lose ?*
  If it's more than une heure ou deux, consider changing the way you work.

Am I Doing This Right ?

Extraits de «Ship It !»

- *[A]re you actively using the system ?*
  If you go weeks–even heures—between code check-ins, you aren't using the system actively.
- *If the hard drive on your workstation crashed right now, how much work yould you lose ?*
  If it's more than une heure ou deux, consider changing the way you work.

Ma pratique personnelle : Dès que j'ai complété une fonctionnalité, je fais un *commit* :

- Permet de bien cerner ce que fait le *commit*
- Si la prochaine étape ne fonctionne pas, permet de revenir à un endroit. . . où «ça fonctionnait» !

Références

A. Hunt and D. Thomas.

The Pragmatic Programmer—From Journeyman to Master.

Addison-Wesley, 2000.

J. Richardson and W.A. Gwaltney.

Ship it ! A Practical Guide to Successful Software Projects.

The Pragmatic Bookshelf, 2005.

V. Subramaniam and A. Hunt.

Practices of an Agile Developer—Working in the Real World.

The Pragmatic Bookshelf, 2006.