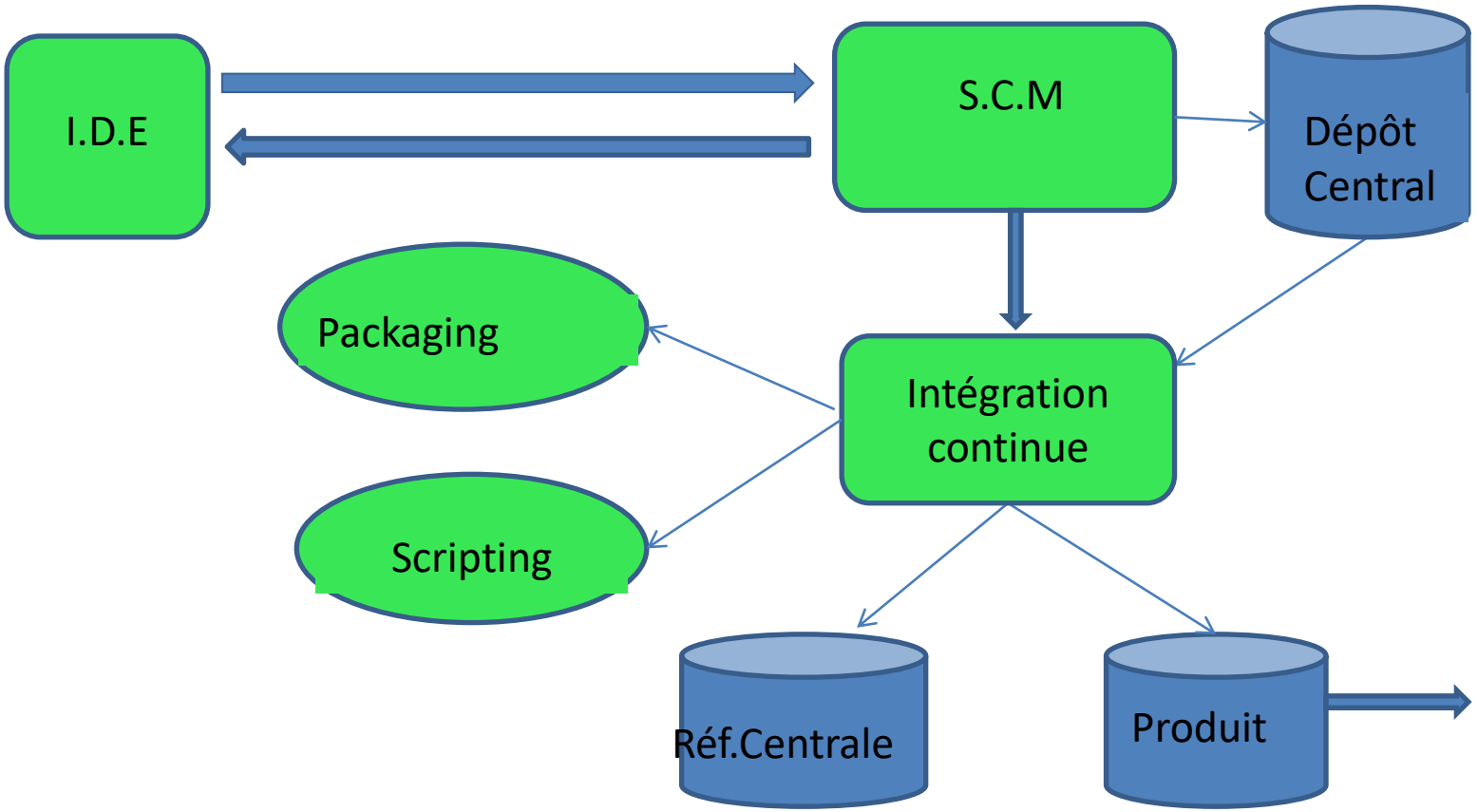
Usine Logicielle

Tarik NACEF

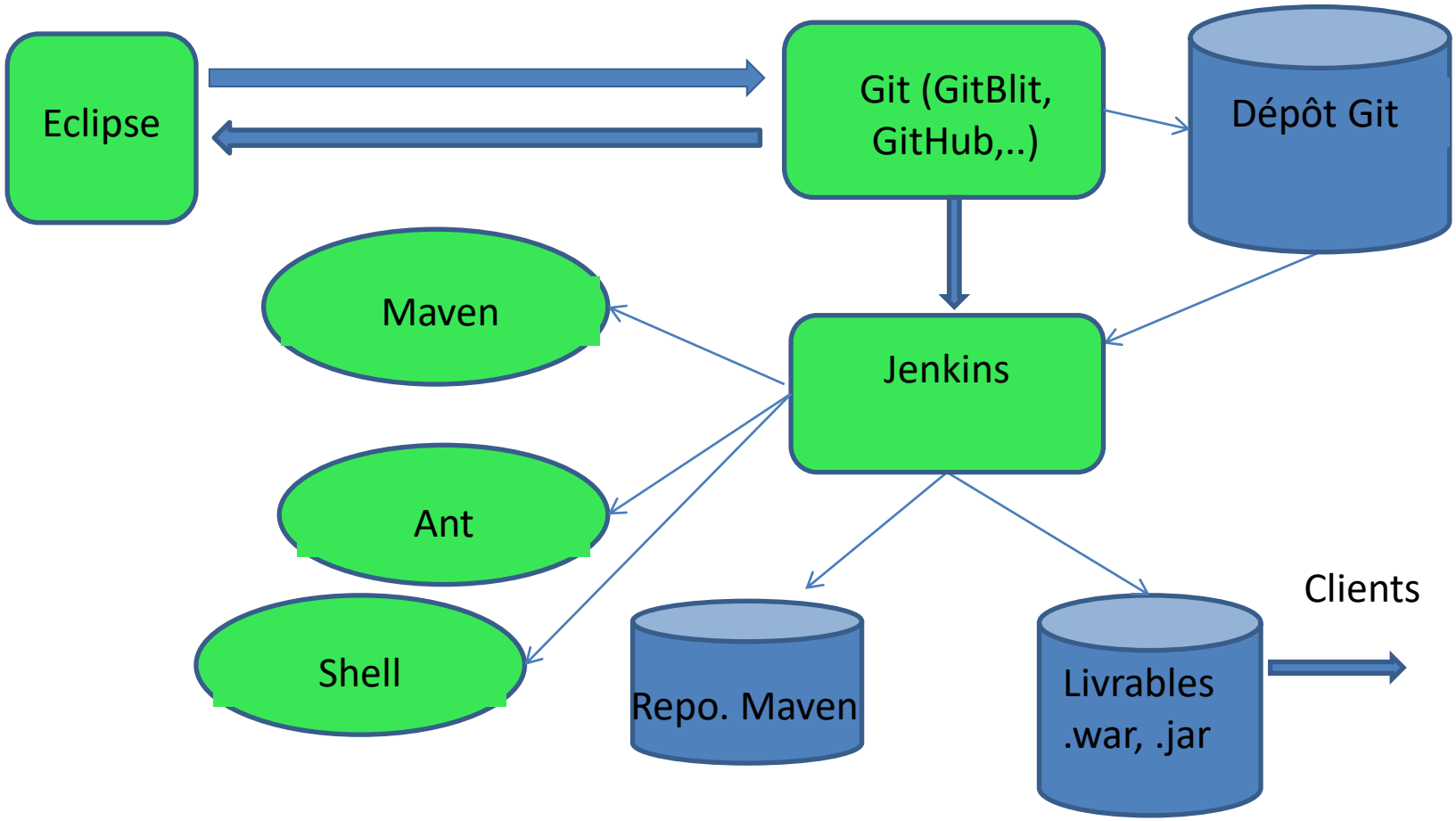
Java – Usine logicielle

Architecture type Usine Logicielle



Java – Usine logicielle

Architecture type Usine Logicielle : Etude & Outils



Java – Usine logicielle

Source Code Management (SCM)

Le bien le plus précieux d'un éditeur de logiciels est le code source des programmes constituant les produits et les applications qu'il publie.

La gestion du code source est donc une étape primordiale du système de développement pour toutes les organisations du secteur logiciel.

Maitriser les versions, savoir qui a modifié quel programme, quand et pourquoi est indispensable pour les responsables de développement et une aide précieuse pour les développeurs eux-mêmes.

Les produits de gestion du code source ont évolués ces dernières années et on distingue maintenant 3 grands type de produits d'aide à la gestion :

Gestionnaire local : RCS

Gestionnaire centralisé : CVS ou Subversion)

Gestionnaire décentralisé : Mercurial ou Git)

Java – Usine logicielle

Source Code Management (SCM)

v · m	Logiciel de gestion de versions	[masqu
Libres	Gestion locale	GNU RCS (1982) · GNU CSSC
	Client-serveur	CVS (1990) · CVSNT (1992) · SVN (2000)
	Décentralisé	GNU arch (2001) · Darcs (2002) · DCVS (2002) · SVK (2003) · Monotone (2003) · Codeville (2005) · Git (2005) · Mercurial (2005) · Bazaar (2005) · Fossil (2007) · Veracity (2011)
Propriétaires	Gestion locale	SCCS (1972) · **PVCS** (1985)
	Client-serveur	Rational ClearCase (1992) · CCC/Harvest (années 70) · CMVC (1994) · Visual SourceSafe (1994) · Perforce (1995) · AccuRev SCM (2002) · Sourceanywhere (2003) · Team Foundation Server (2005) · Rational Synergy (2006)
	Décentralisé	BitKeeper (1998) · Plastic SCM (2007)
Concepts	Branche · Changelog · Commit · Codage différentiel · Comparaison de fichiers · Changeset · Dépôt · Fork · Merge · Tag (en) · Trunk	

Java - Usine logicielle

Source Code Management (Git, GitBlit)

GitBlit peut être installé sur un serveur Web disposant d'un conteneur de servlet, comme Tomcat par exemple.

Il faudra déposer le fichier **war** d'installation sur le répertoire /webapps de tomcat pour obtenir l'application Web.

Compte admin, admin par défaut.

Ref. : http://gitblit.com/

Java - Usine logicielle

Git - (Git, GitBlit, Egit)

Dispositif d'édition des sources :

Créer un dépôt local (Egit) : Une fois

Associer les fichiers du workspace Eclipse au dépôt local (Egit)

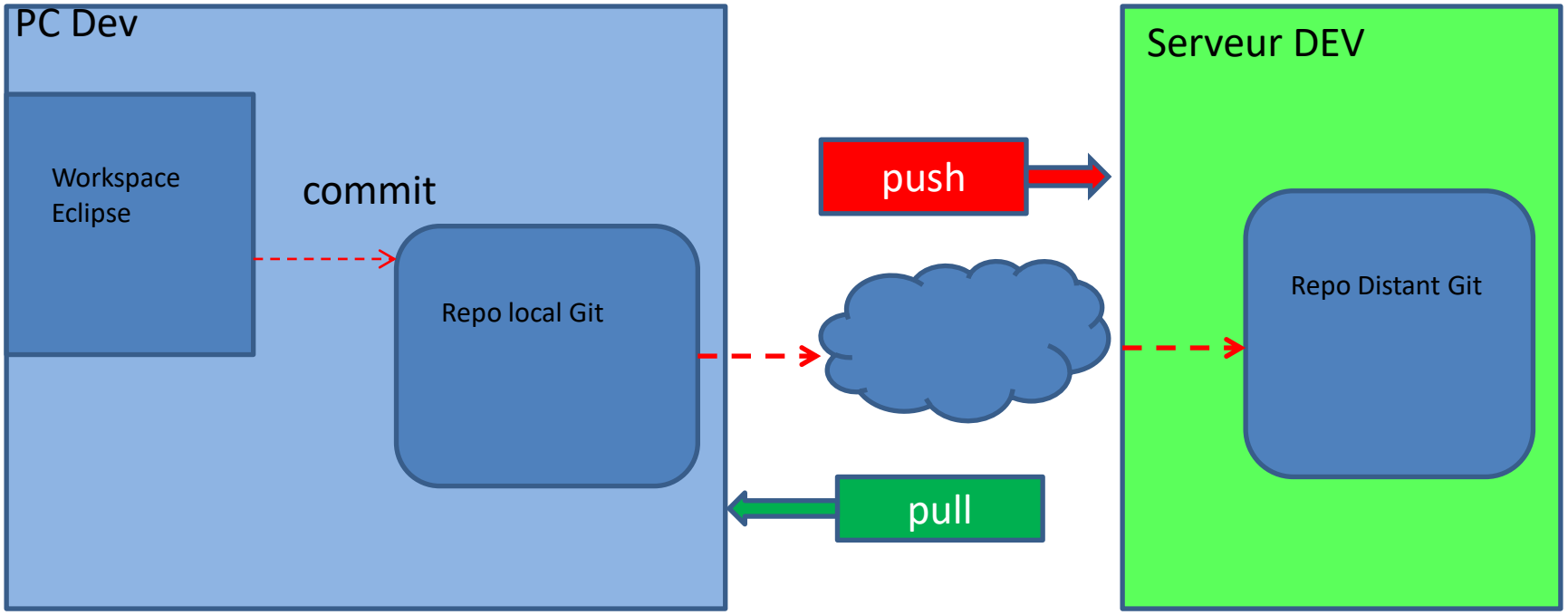
- Créer un nouveau projet sous Eclipse
- Team->Share Project

Déclarer un dépôt distant (GitBlit) : Pour chaque projet

Java - Usine logicielle

Source Code Management (Git, GitBlit, Egit)

Architecture type de développement avec Git :



Java - Usine logicielle

Source Code Management (Git, GitBlit, Egit)

Lors de la création d'un projet sous Eclipse les fichiers restent sous le contrôle exclusif d'Eclipse et dans le Workspace.

Dès que le développeur, au début ou en cours de projet exécute la commande « **Share Project** » tous les fichiers du projet passent sous le contrôle de ***Git***.

Le répertoire du projet devient le **repo local**, après avoir effectué l'opération **Team->Share Project**

Java - Usine logicielle

Git – Operations

- **Pull** : Récupérer les modifications effectuées sur un dépôt distant sur le dépôt local
- **Push** : envoyer les modifications effectuées sur un repo local vers le dépôt distant
- **Commit** : Enregistrer les modifications effectuées sur le dépôt local (id, commentaire, tag, …) (enregistrement local git)
- Git status : donne l'état du dépôt local par rapport au central
- Git checkout : Récupère une copie depuis le dépôt central

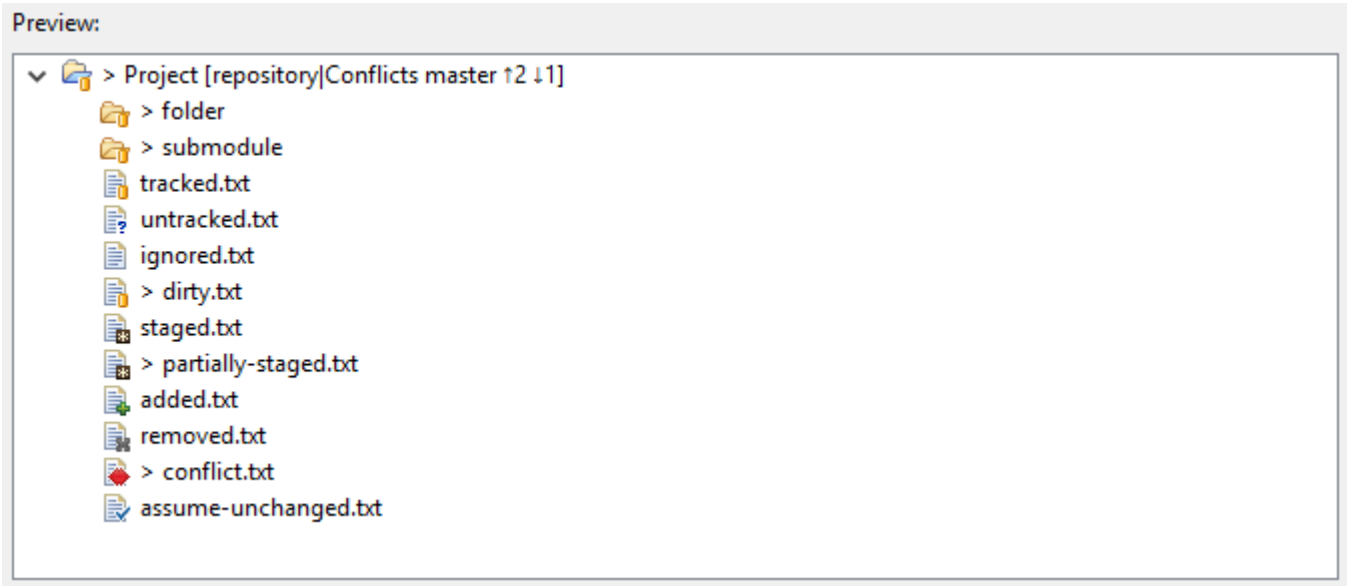
Java - Usine logicielle

Source Code Management – Operations

Légende : customizable sous Eclipse

↑ : Demande **Push :** Envoie les derniers commits du repo local vers le repo central

↓ : Demande **Pull :** Récupère les dernières modifications du repo central vers le repo local



Java - Usine logicielle

Git – Operations

Création d'un projet géré par Git (via GitLib, GitHub)

Créer le projet sur le repo Central via GitLib

- Onglet Dépôt -> Créer nouveau dépôt
- Initier le premier commit (Evite une commande Locale)
- Autoriser équipes ou développeurs

Cloner le Projet : (Central -> Local)

- Avec les menu contextuel Explorer
- clone

Importer le projet sous Eclipse avec File->Import->from Git Repo

Java - Usine logicielle

Source Code Management – Operations

Les outils associés à Git, et GitLib (Repo distant Web)

Learn Git

If you are unsure how to use this information, consider reviewing the Git Community Book for a better understanding on how to use Git.

Open Source Git Clients

Git	the official, command-line Git
TortoiseGit	Windows file explorer integration (requires official, command-line Git)
Eclipse/EGit	Git for the Eclipse IDE (based on JGit, like Gitblit)
Git Extensions	C# frontend for Git that features Windows Explorer and Visual Studio integration
GitX-dev	a Mac OS X Git client

Java - Usine logicielle

Source Code Management – Operations

Création du repo local en ligne de commande (Git)

Un projet Eclipse existe déjà et on veut le faire passer sous le contrôle de Git.

cd /chemin_du_projet (Ou si des repos existent déjà aller sur le repertoire racine et créer un nouveau répertoire au nom du projet)

git init // Init le repertoire et crée le repertoire **.git**

git add . // Associe tous les fichiers au repo local

git commit // Enregistre les modifications dans le repo local

Java - Usine logicielle

Source Code Management – Operations

La phase délicate est l'initialisation correcte du repo local avec Eclipse pour Git.

Le répertoire sous le contrôle de Git nécessite un paramétrage spécifique.

Ce même répertoire exploité par Eclipse nécessite aussi un paramétrage spécifique afin que l'IDE fonctionne correctement.

Invariants :

Il faut créer le projet Eclipse afin qu'il soit correctement paramètre.

Il faut créer le projet Git (Central) dans l'interface dédié à cet effet.

Il faut faire passer le projet Eclipse sous le contrôle de Git dans le repo local

Il faut associer le repo local au repo distant.

Java - Usine logicielle

Source Code Management – Operations

Suivant le contexte on procédera de différentes manières.

1- la création du repo sur le repo distant (central)

Cette étape est toujours à effectuer, quelque soit le contexte.

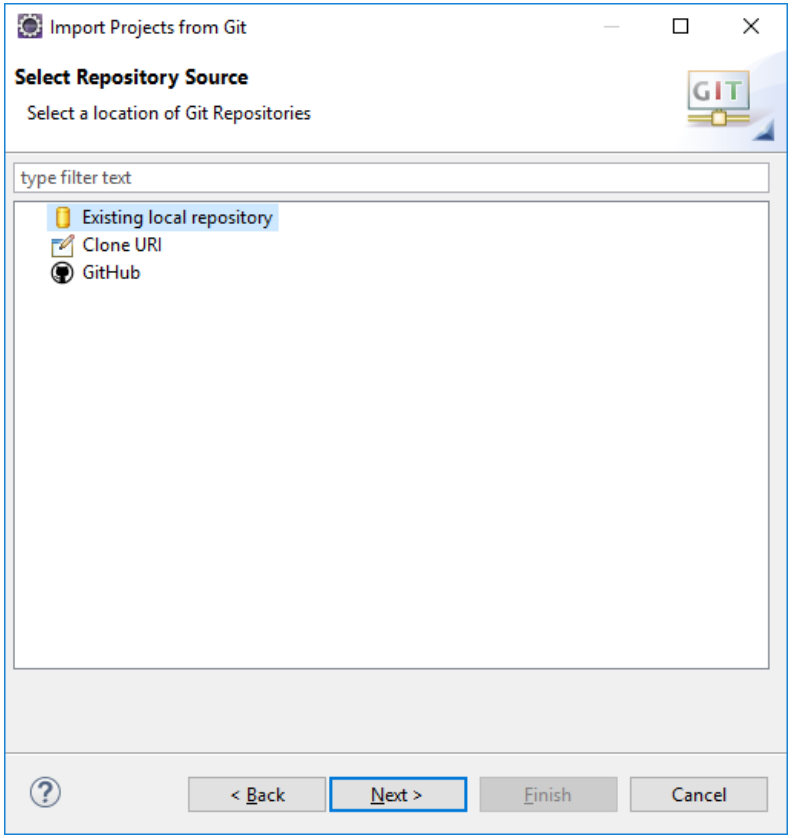
Faire bien attention au nom du dépôt qui doit être identique au nom du projet (Eclipse).

2- Si le projet existe et comporte déjà des objets :

- 2.1 – Dans Eclipse -> Share Project (Bugée Version: Neon.3 Release (4.6.3))
- 2.2 – Créer le projet en ligne de commande comme indiqué dans la slide précédente

Java - Usine logicielle

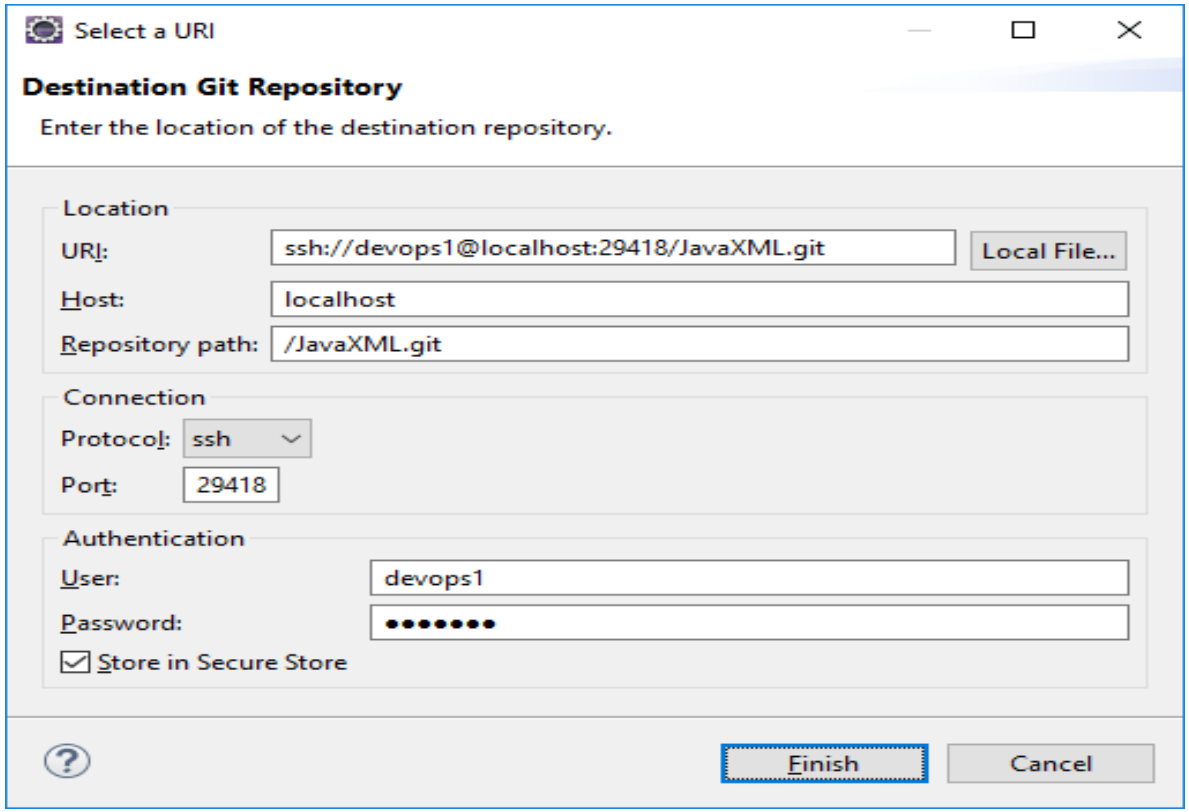
Source Code Management – Operations



Java - Usine logicielle

Source Code Management – Operations

Configuration, réglages Push (opérations vers le repo distant)



Java - Usine logicielle

Source Code Management – Operations

Opérations courantes : dès que > 2 développeurs sur un projet

Pull : met à jour le repo local avec les dernières modifications du repo distant

Modifications, développement

Commit : Enregistre les modifications dans le repo local

Push : Envoie l'ensemble des ***commits*** effectués sur le repo local vers le repo distant (Central)

Ces commandes sont les plus utilisés lors des phases de développements notamment dans les équipes de plus de 1 développeur.

Java - Usine logicielle

Source Code Management – Operations

Cas du **merge** : Cas classique

Le merge est une opération nécessaire lorsqu'un Pull ou un Push de l'utilisateur récupère une copie du repo qui a varié depuis qu'il lui-même effectué des modification sur sa version.

Supposons un projet P qui contient 3 classes Java A,B et C

Jour 1

Dev 1 : Prend une image du repo (pull)

Dev 2: Prend une image du repo (pull)

Jour 2

Dev1 : Modifie A et commit

Dev2 : modifie C et commit

Dev1 : Push

Jour 3

Dev2 : Est informé qu'il existe des modifications sur le repo, il doit faire un **Pull**, et son commit sera **mergé** automatiquement avec les modifications de la classe A et pourra faire un commit.

Java - Usine logicielle

Source Code Management – Operations

Cas du **merge** : Cas compliqué

Dans ce cas, le plus simple est d'utiliser l'interface graphique proposée par l'IDE.

Exemple Eclipse : ⇔ conflict

Java - Usine logicielle

Source Code Management – Operations

MAKE CHANGES

Review edits and craft a commit transaction

`$ git status`
Lists all new or modified files to be committed
`$ git diff`
Shows file differences not yet staged
`$ git add [file]`
Snapshots the file in preparation for versioning
`$ git diff --staged`
Shows file differences between staging and the last file version
`$ git reset [file]`
Unstages the file, but preserve its contents
`$ git commit -m "[descriptive message]"`
Records file snapshots permanently in version history

Java - Usine logicielle

Source Code Management – Operations

GROUP CHANGES

Name a series of commits and combine completed efforts

`$ git branch`
Lists all local branches in the current repository
`$ git branch [branch-name]`
Creates a new branch
`$ git checkout [branch-name]`
Switches to the specified branch and updates the working directory
`$ git merge [branch]`
Combines the specified branch's history into the current branch
`$ git branch -d [branch-name]`
Deletes the specified branch

Java - Usine logicielle

Git – Operations

Création de branches :

Sous Eclipse :

Team->Switch to -> New Branch [branch-name]

Ligne de Commande :

git checkout [branch-name]

Java - Usine logicielle

Source Code Management – Operations

Les branches sont utilisées afin de paralléliser des développement et/ou différer la livraison d'un développement dans le temps pour une version ultérieure.

Typiquement on peut imaginer une branche de développement et une branche de production.

La branche de développement sera utilisée pour faire évoluer un projet, celle de production correspondra à la dernière version livrée en production par exemple.

Lors de la création de la nouvelle version, la branche de développement sera mergée sur la branche de production qui elle sera figée jusqu'à la prochaine livraison du produit.

Il existe de nombreux scénarios d'utilisation des branches, les scénarios les plus simples sont conseillés.