

Méthodes agiles

- processus et outils -

Outils pour le développement logiciel

Comment travailler?

- Sur des projets complexes
 - ▶ Gestion de versions
 - ▶ Outils de build
 - ▶ Intégration continue

Intégration continue

- Ensemble de bonnes pratiques en développement logiciel :
 - ▶ Dépôt versionné
 - ▶ Compilation automatique
 - ▶ Automatisation des tests
 - ▶ Déploiement automatisé
- La version du dépôt est toujours utilisable (compilable, exécutable, testable)

Utiliser

- des systèmes de gestion de versions
 - Subversion / GIT / ...
- des outils de build
 - Make / Ant / Maven / ...
- des plate-formes d'intégration continue
 - (Hudson) / Jenkins / ...

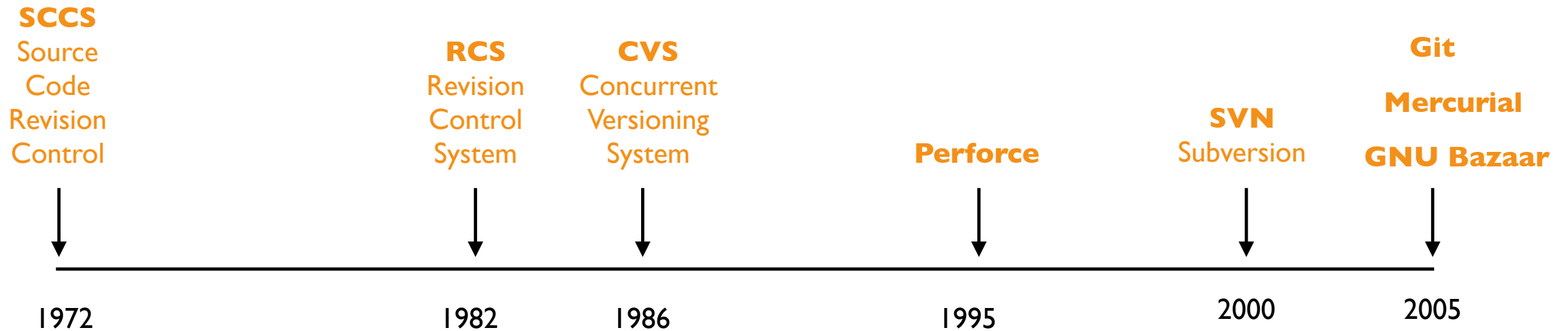
Systemes de gestion de versions

*Version Control Systems
(VCS)*

Problématiques

- Sécurité (sauvegarde avec historique)
 - ▶ Sauvegarder ses données
 - ▶ Revenir à une version antérieure d'un fichier/projet
- Parallélisme (multi-machines, multi-développeurs)
 - ▶ Travailler sur plusieurs machines
 - ▶ Travailler à plusieurs sur un même fichier/projet

Histoire

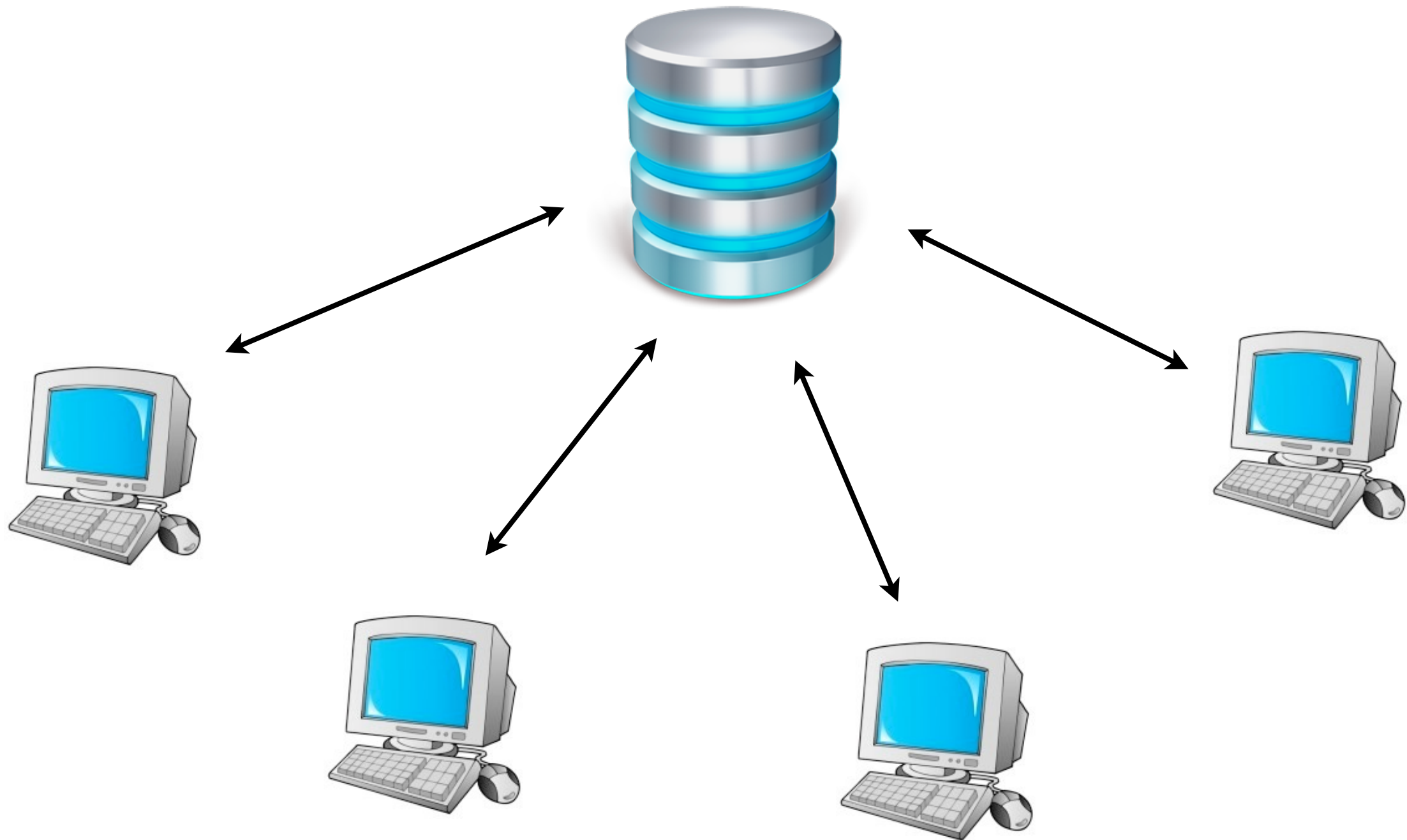


Generation	Repository	Software
First	Local	RCS, SCCS
Second	Centralized	CVS, Perforce, SVN,
Third	Distributed	Bazaar, Git, Mercurial

Familles

- **Centralisé** Subversion, CVS, ...
 - ▶ Chaque collaborateur travaille sur une copie locale
 - ▶ Synchronisation des copies via un dépôt centralisé
- **Décentralisé** Git, Mercurial, Darcs, ...
 - ▶ Travail sur une copie locale
 - ▶ Synchronisation sur le dépôt local
 - ▶ Synchronisation sur des dépôts distants

Subversion

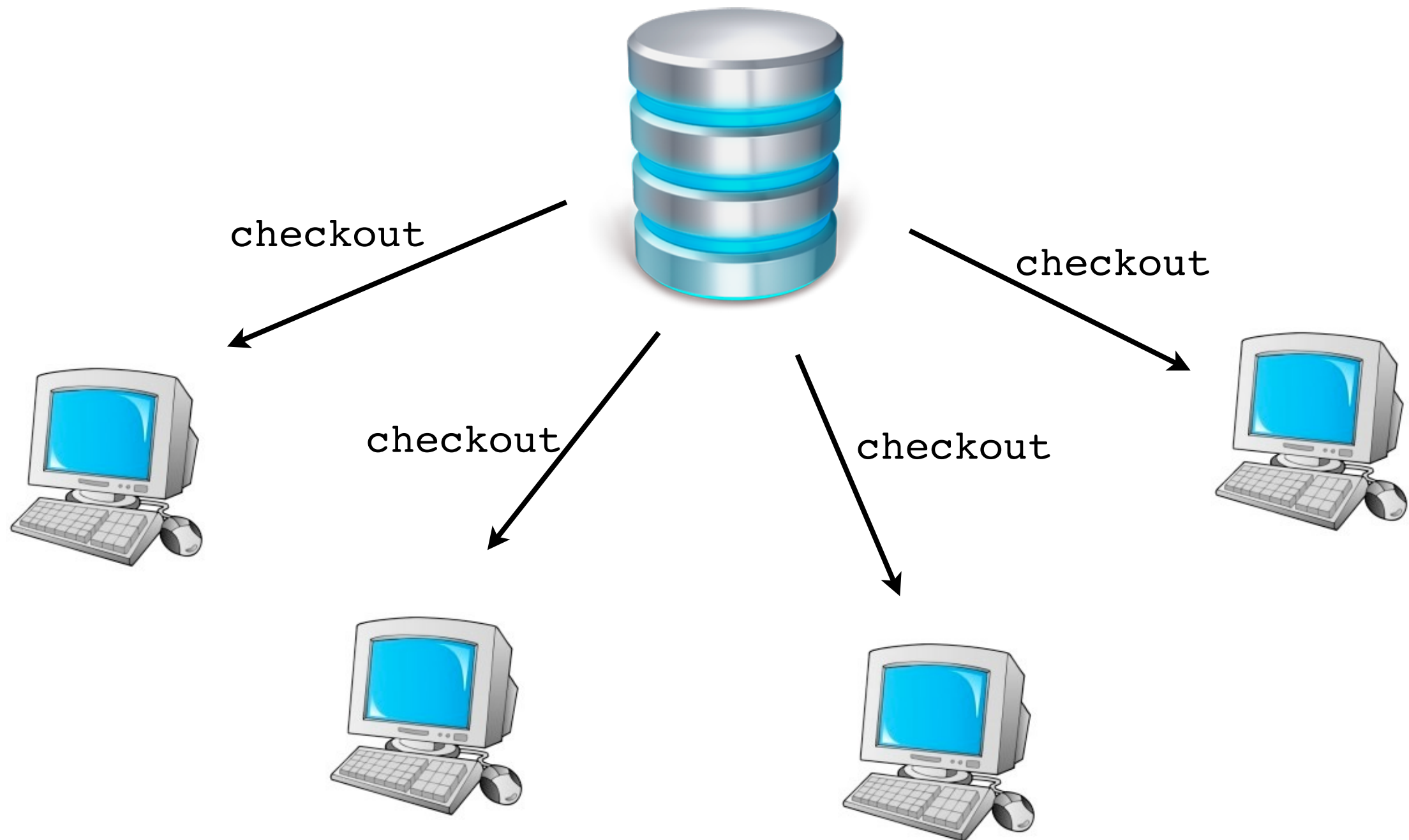


Subversion

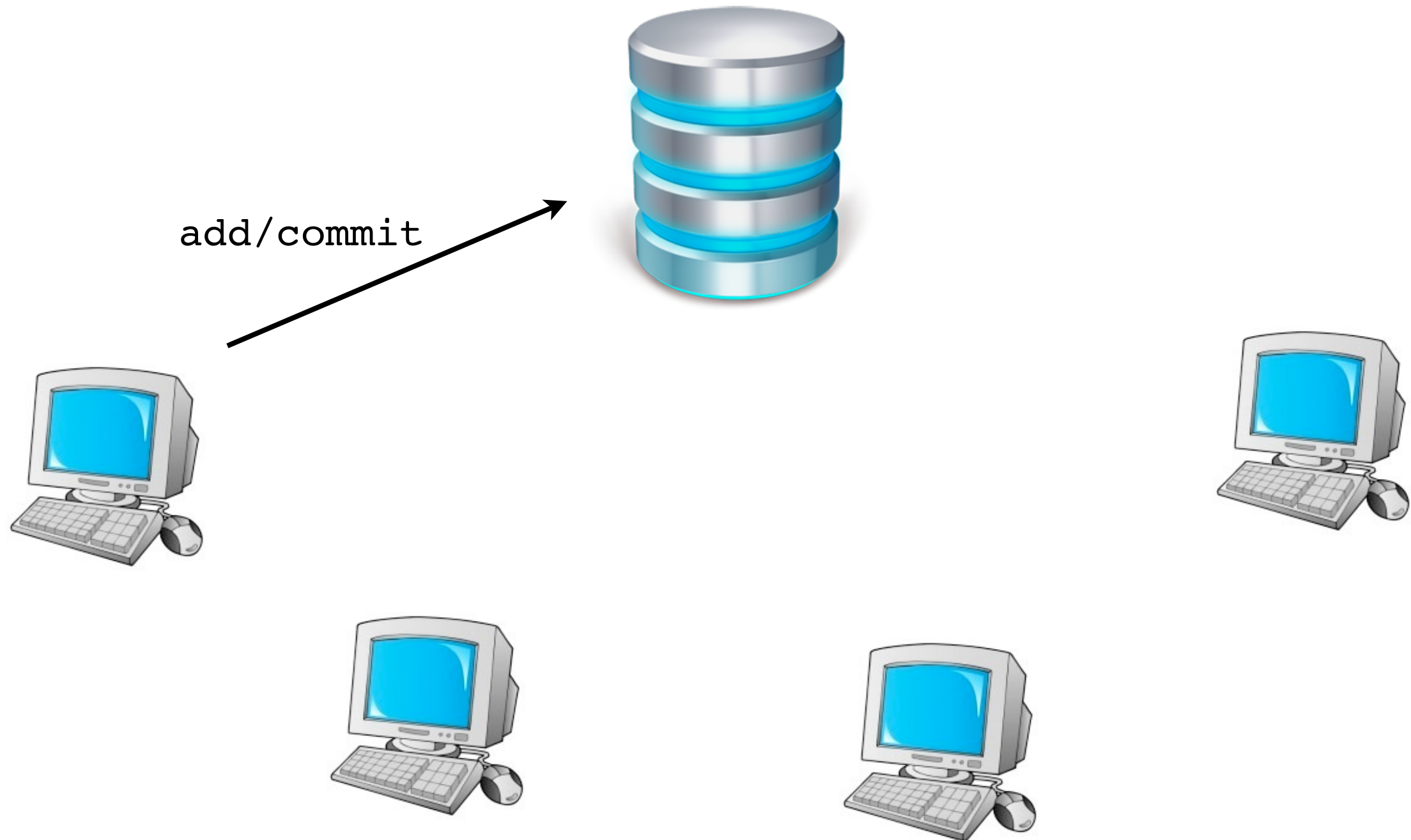
Commande **svn**

- **checkout**: récupère localement la version du dépôt
- **commit**: valide les changements faits localement et les ajoute au dépôt
- **update**: met à jour la copie locale
- **add** : ajoute un fichier dans le dépôt
- **status**: vérifie l'état local par rapport à l'état du dépôt distant

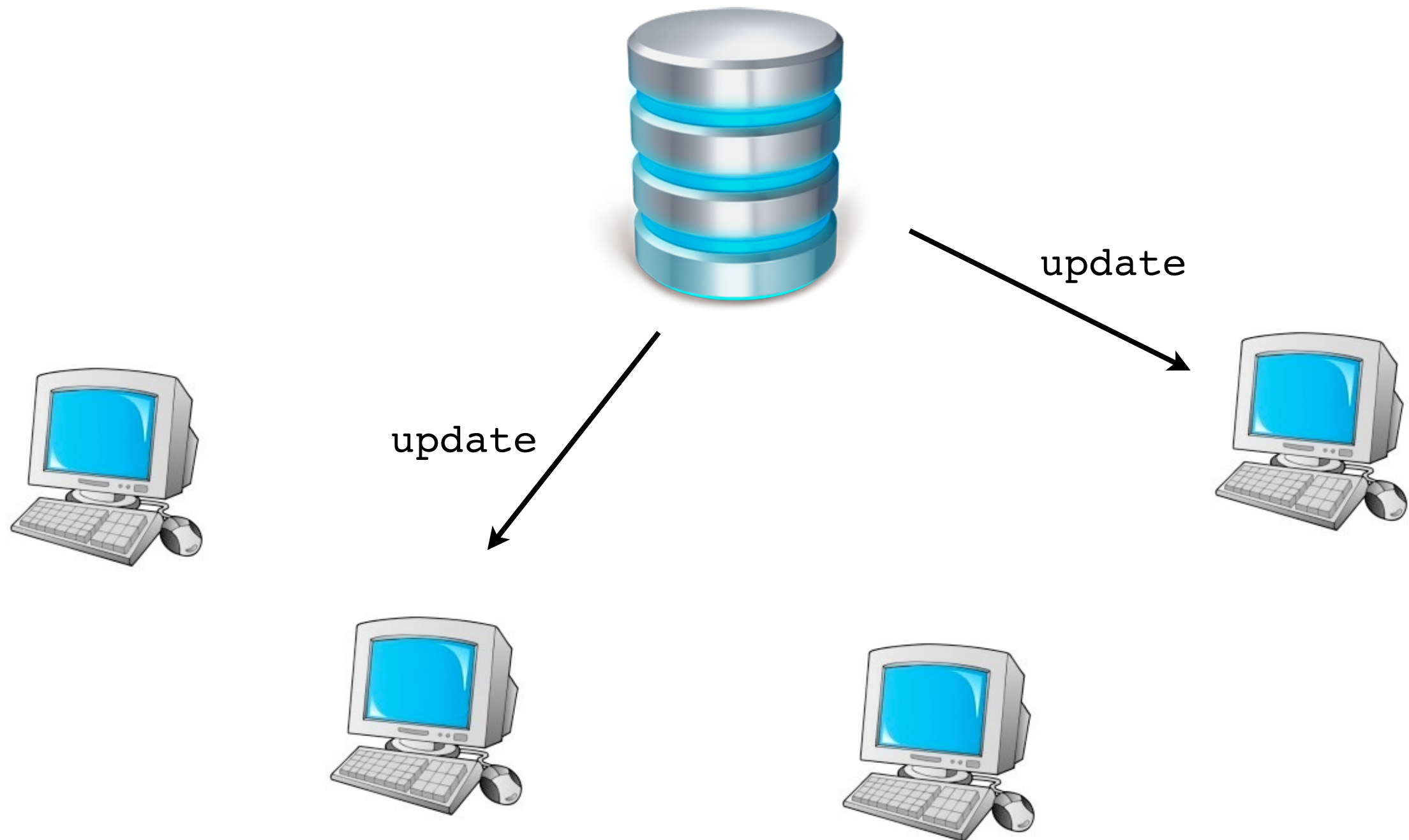
Récupérer dépôt



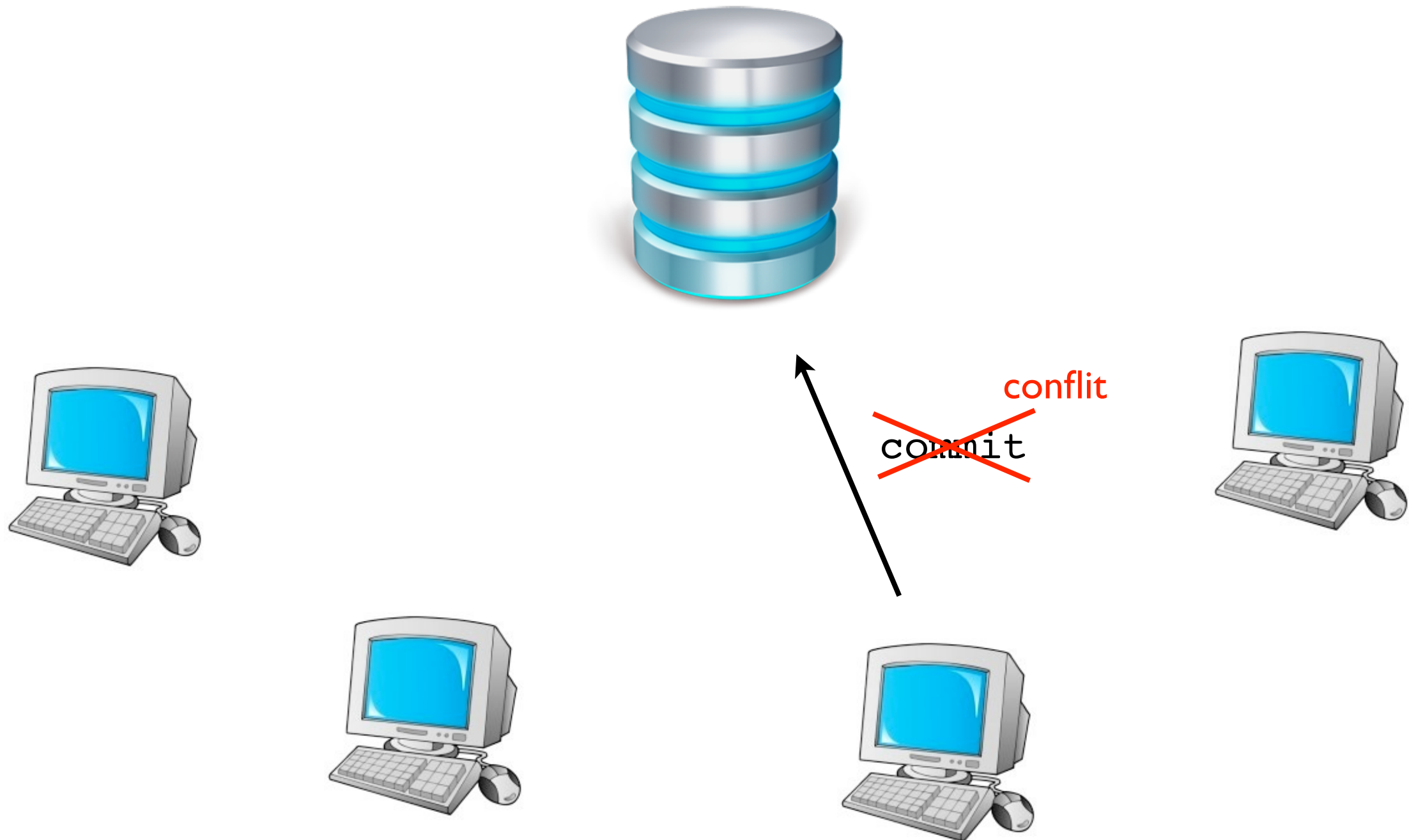
Ajouter/modifier fichier(s)



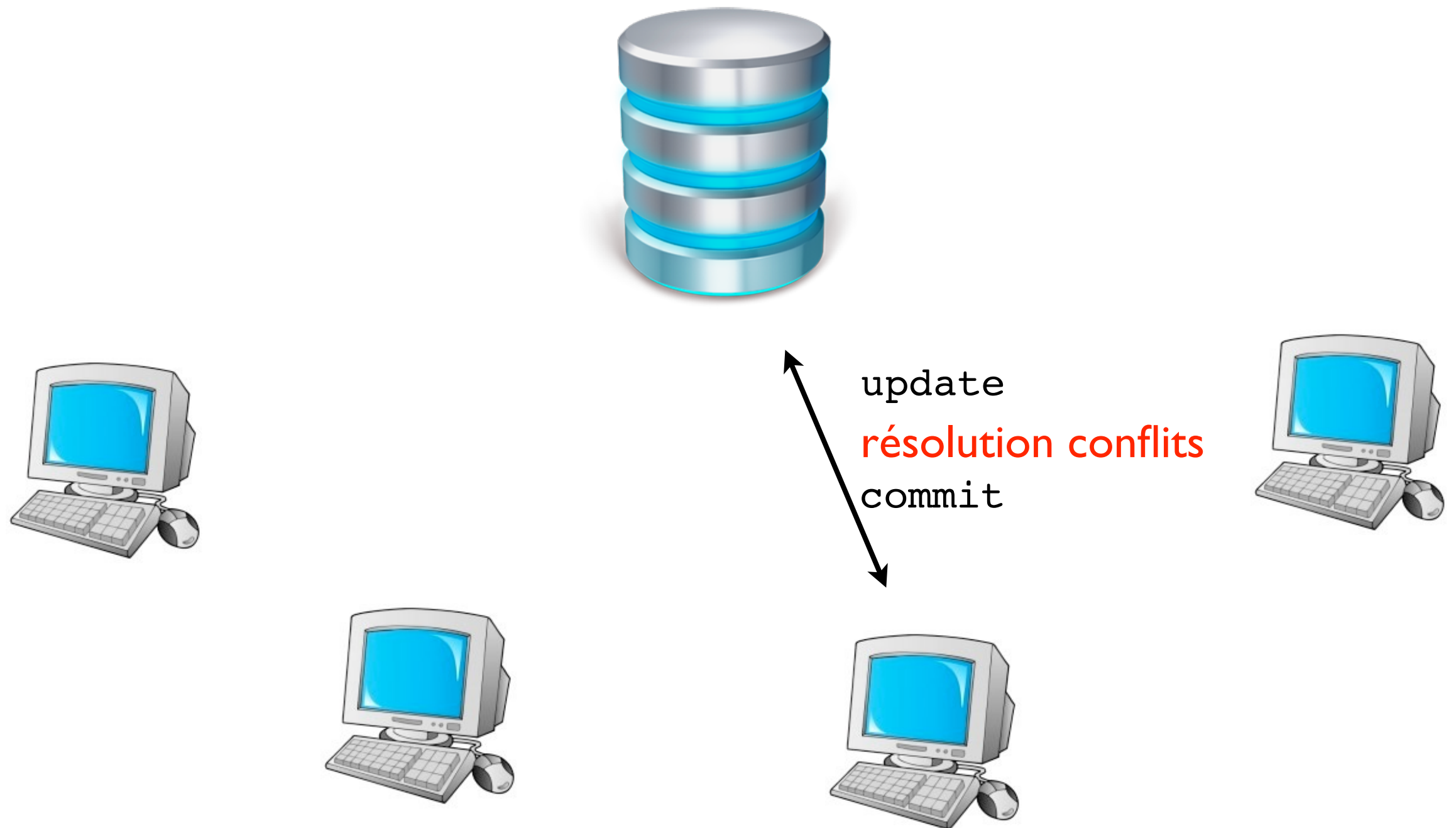
Mis à jour copie locale



(Résolution de) conflits

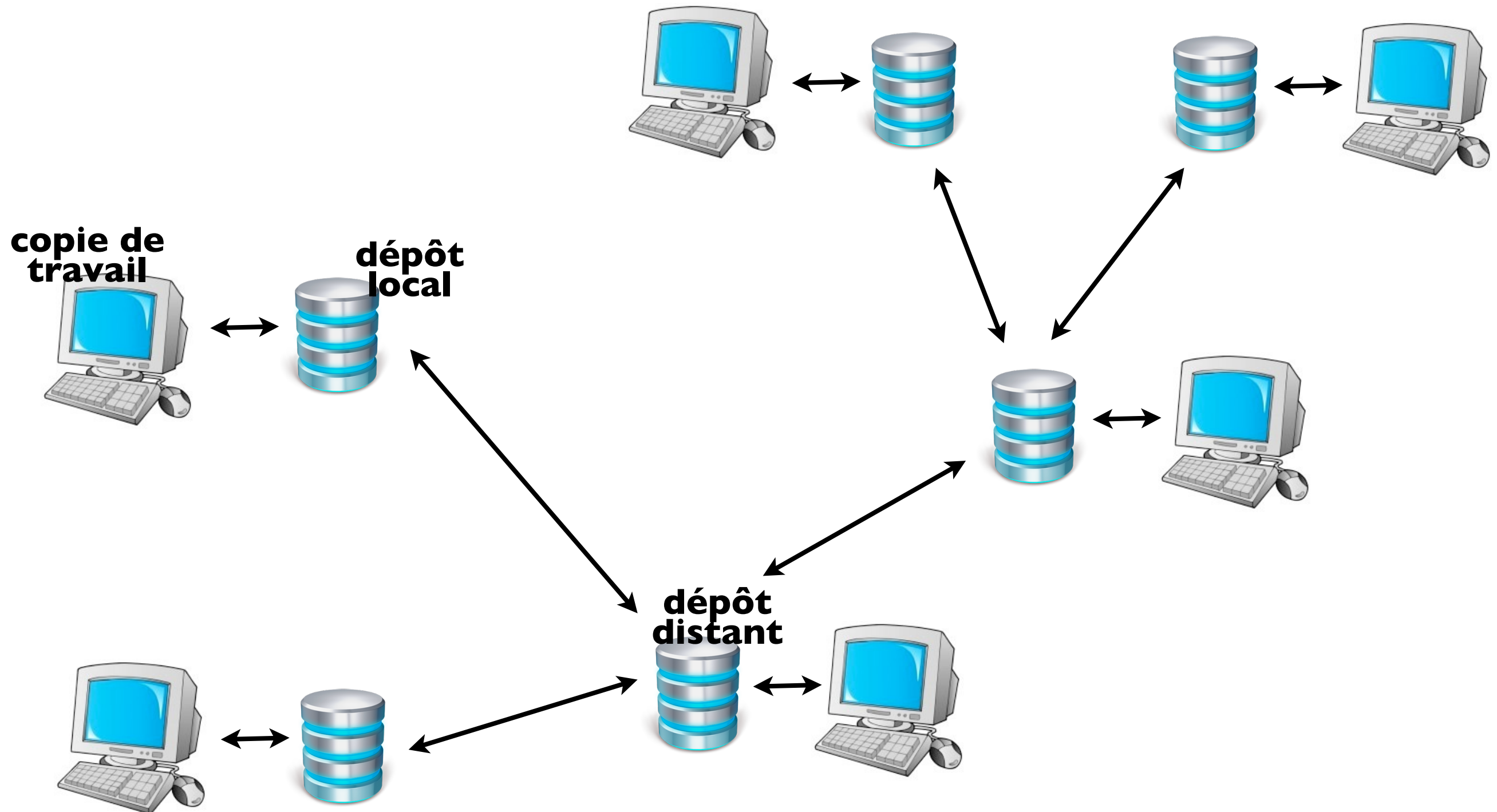


(Résolution de) conflits



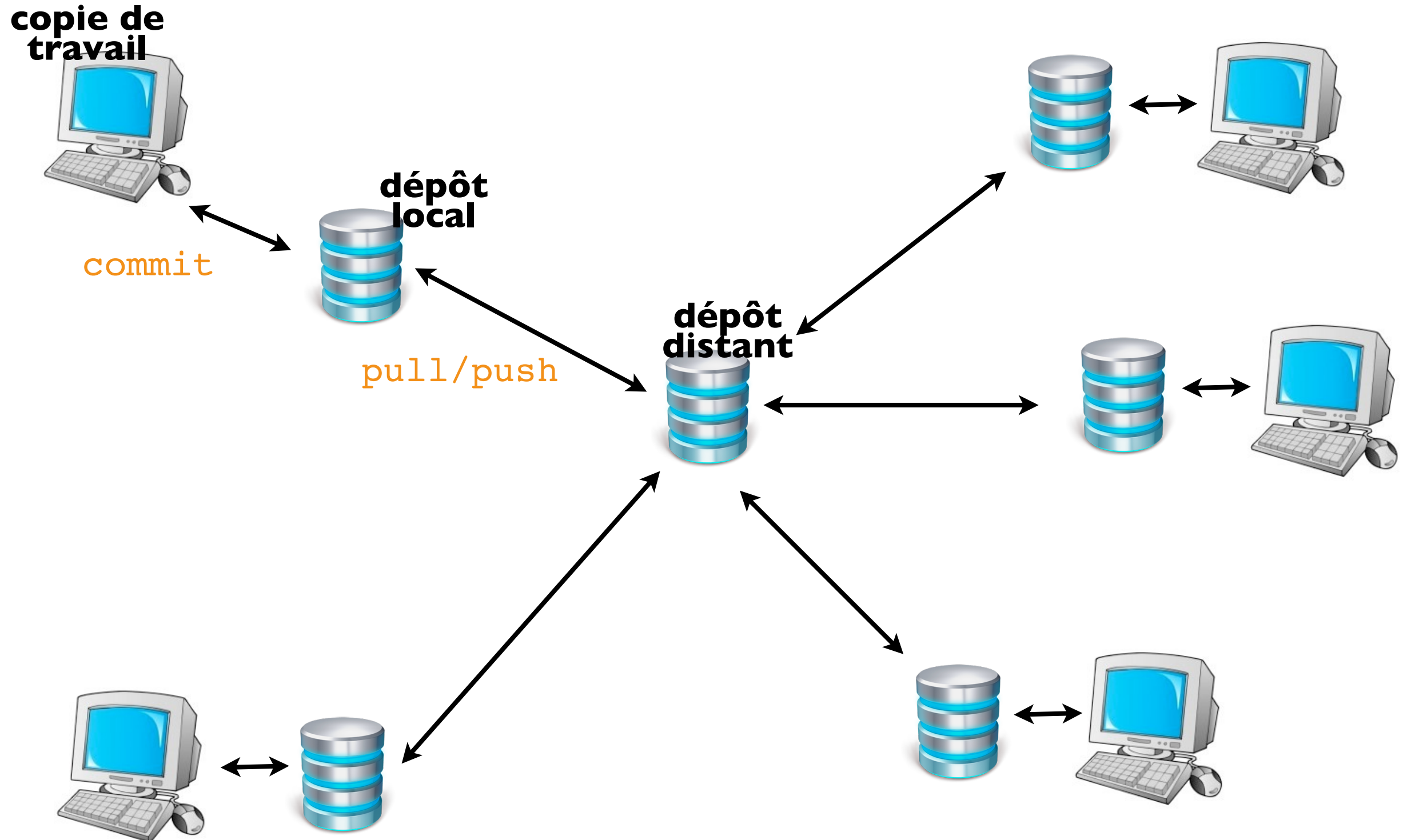
Git

<http://git-scm.com/>

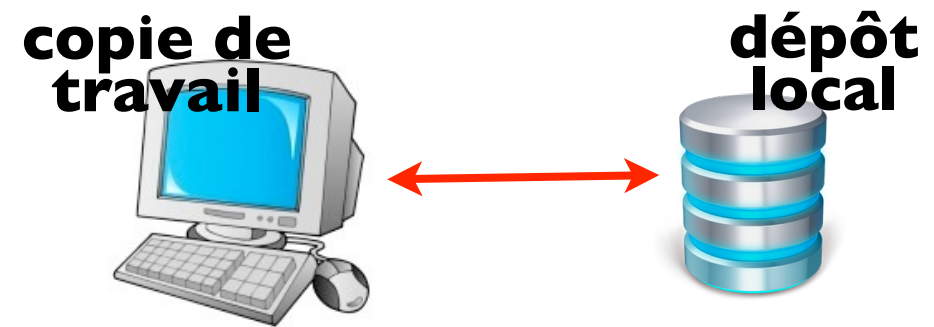


Git

<http://git-scm.com/>



Git



Commande **git**

- Commandes principales pour le dépôt local :
 - **add** : ajoute un fichier au suivi pour le dépôt local
 - **commit** : valide les changements faits localement et les intègre au dépôt local
 - **checkout** : récupère la version du dépôt local (et écrase les modifications non validées)

Git



Commande **git**

- Commandes principales pour synchroniser avec le(s) dépôt(s) distant(s) :
 - **clone**: récupère localement un dépôt distant
 - **pull**: récupère les mises à jour sur les dépôts distants
 - **push**: envoie les mises à jour locales vers les dépôts distants

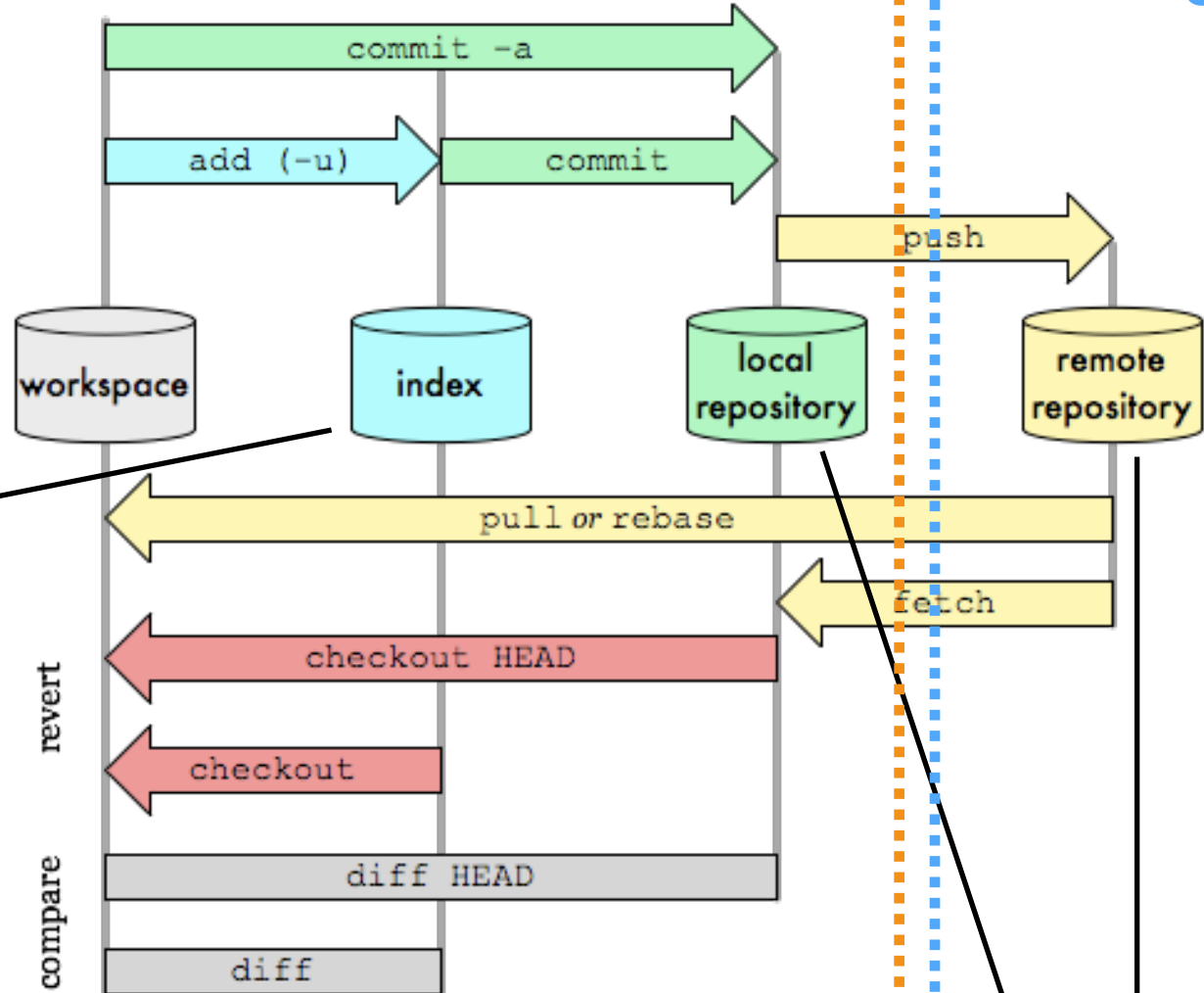
Git Data Transport Commands

<http://osteele.com>

local distant

copie locale
des sources

espace temporaire qui permet de
spécifier le sous-ensemble de
modification du workspace à
répercuter dans le dépôt local
lors d'un commit



dépôt(s) contenant toutes les
versions commitées ainsi des
méta-informations (logs, tags, etc.)

Création et ajout

- Création

- ▶ dépôt local vide : `git init`
- ▶ à partir d'un dépôt existant : `git clone`

- Ajout

- ▶ au suivi : `git add`
- ▶ au dépôt local : `git commit`

Informations

- ▶ infos répertoire travail et index : `git status`
- ▶ historique des commits : `git log`
- ▶ détails d'un commit : `git show`
- ▶ différences
 - ◆ working-index : `git diff`
 - ◆ index-repository : `git diff --cached`
 - ◆ working-repository : `git diff HEAD`

Init local repository

```
> git init
```

Check configuration

```
> git config -l
...
user.email=Horatiu.Cirstea@loria.fr
user.name=hcirstea
...
```

Create file and add it to the index

```
> ...
> git add file.md
```

Check what is ready to be committed

```
> git diff --cached
diff --git a/file.md b/file.md
new file mode 100644
index 0000000..e69de29
>
```

```
> git diff
>
```

Commit changes (to local repository)

```
> git commit -m"create new file"
```

Modify file.md

```
> git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   file.md

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

```
> git diff --cached
>
```

```
> git diff
diff --git a/file.md b/file.md
index e69de29..94e5e27 100644
--- a/file.md
+++ b/file.md
@@ -0,0 +1 @@
+# FILE
```

```
> git add file.md
```

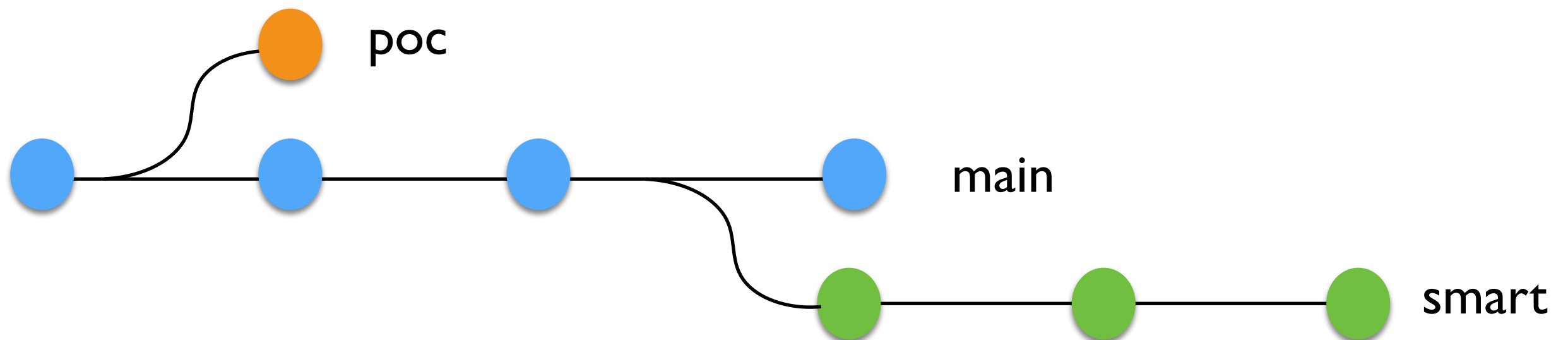
```
> git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   file.md
```

```
> git diff --cached
diff --git a/file.md b/file.md
index e69de29..94e5e27 100644
--- a/file.md
+++ b/file.md
@@ -0,0 +1 @@
+# FILE
```

```
> git diff
>
```

Branches

- ▶ `git branch`
 - ▶ liste les branches
- ▶ `git branch <branch>`
 - ▶ création de `branch`
- ▶ `git switch <branch>`
 - ▶ selection de `branch`
- ▶ `git switch -c <branch>`
 - ▶ création et sélection de `branch`



Create and switch to branch

```
> git branch
* main
> git switch -c poc
Switched to a new branch 'poc'
> git branch
  main
* poc
```

Work on new branch

```
...
> git commit -a -m"add a new line"
...
> git commit -a -m"add more lines"
```

Work on initial branch

```
> git switch main
..
> git commit -a -m"start with title"
...
> git commit -a -m"underline"
```

Check differences

```
> git diff poc
diff --git a/file.md b/file.md
index b33ead9..48de934 100644
--- a/file.md
+++ b/file.md
@@ -1,7 +1,2 @@
-# FILE ...
```

Branches

- ▶ `git merge`
 - ▶ intégrer les modifications d'une autre branche
- ▶ `git branch -d <branch>`
 - ▶ suppression (safe) de `branch`
- ▶ `git branch -D <branch>`
 - ▶ suppression (forcée) de `branch`

Merge branch

```
> git merge poc
Auto-merging file.md
CONFLICT (content): Merge conflict in file.md
Automatic merge failed; fix conflicts and then commit the result.
```

Fix conflicts

```
<<<<<< HEAD
## FILE
-----
=====
# FILE

experiment with new lines

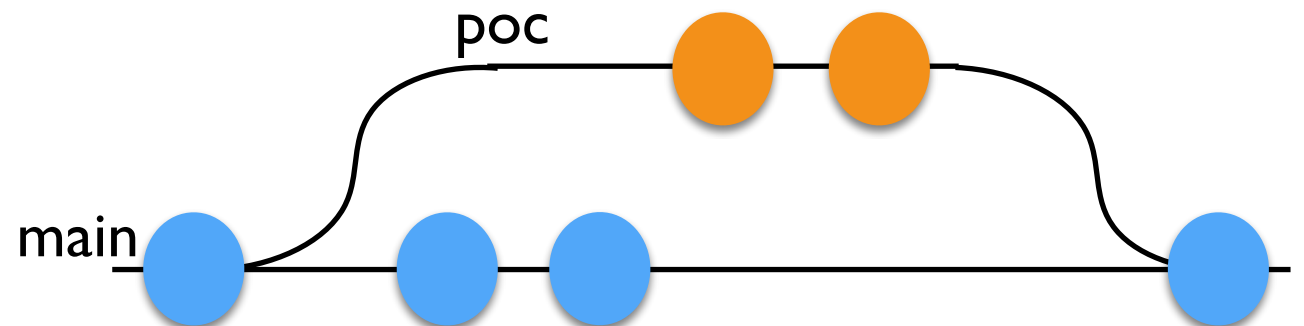
one more line

and another one
>>>>>> poc
```

Commit the fix

```
> git commit -a -m"fix conflicts"
```

```
> git log --graph --oneline main
* 5bbf98a (HEAD -> main) fix conflicts
|
| * b8881d9 (poc) add more lines
| * 303a2c2 add a new line
| * 4e08a78 underline
| * e8d1413 start with title
|/
* 9604ca0 create new file
```



Collaboration

- ▶ `git clone`
 - ▶ cloner un dépôt distant
- ▶ `git remote add`
 - ▶ ajouter un dépôt distant
- ▶ `git fetch`
 - ▶ récupérer changements dépôt distant - dépôt local
- ▶ `git pull`
 - ▶ récupérer changements dépôt distant - working
- ▶ `git push`
 - ▶ propager changements vers dépôt distant

Set remote repository

```
> git remote add origin https://gitlab.univ-lorraine.fr/cirstea6/acl4demo.git
```

Check configuration

```
> git config -l
...
user.email=Horatiu.Cirstea@loria.fr
user.name=hcirstea
...
remote.origin.url=https://gitlab.univ-lorraine.fr/cirstea6/acl4demo.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
```

Fetch remote repository

```
> git fetch origin main
```

Check branches

```
> git branch -a
* main
  poc
  remotes/origin/main
```

Check differences (w.r.t. index)

```
> git diff origin/main
```

Check what's new remotely

```
> git log -p HEAD..FETCH_HEAD
```

Check the log, local and remote

```
> git log --graph --oneline main
* 5bbf98a (HEAD -> main) fix conflicts
| \
| * b8881d9 (poc) add more lines
| * 303a2c2 add a new line
* | 4e08a78 underline
* | e8d1413 start with title
| /
* 9604ca0 create new file
* 97207ec more files
* 4e14839 create file cl
```

```
> git log --graph --oneline origin/main
* 3a220b3 (origin/main) Minimalist README.md
* 2762c2b Only GS and AYF in README.md
* cd73558 Initial commit
```

Fetch and merge remote repository

```
> git pull origin main --allow-unrelated-histories --rebase=false
```

- * --allow-unrelated-histories # merge unrelated histories
- * --rebase=false # merge, not rebase

```
> git log --graph --oneline main
* 6ae8b60 (HEAD -> main) Merge branch 'main' of https://gitlab.univ-lorraine.fr/cirstea6/acl4demo
| \
| * 3a220b3 (origin/main) Minimalist README.md
| * 2762c2b Only Getting started and Add your files in README.md
| * cd73558 Initial commit
* 5bbf98a fix conflicts
| \
| * b8881d9 (poc) add more lines
| * 303a2c2 add a new line
* | 4e08a78 underline
* | e8d1413 start with title
| /
* 9604ca0 create new file
* 97207ec more files
* 4e14839 create file cl
```

Track local branch remotely

```
> git branch --set-upstream-to=origin/main main
```

Push to remote repository (implicit origin)

```
> git push
```

Modify local and remote repository

```
> git log --graph --oneline main origin/main
* d3387a0 (HEAD -> main) add a last line
| * a629572 (origin/main) add scenarios section in README.md
|/
* 6ae8b60 Merge branch 'main' of https://gitlab.univ-lorraine.fr/cirstea6/acl4demo
| \
| * 3a220b3 Minimalist README.md
| * 2762c2b Only Getting started and Add your files in README.md
| * cd73558 Initial commit
* 5bbf98a fix conflicts
| \
| * b8881d9 (poc) add more lines
| * 303a2c2 add a new line
* | 4e08a78 underline
* | e8d1413 start with title
|/
* 9604ca0 create new file
* 97207ec more files
* 4e14839 create file cl
```

Fetch and merge remote repository

```
> git pull --rebase=true
```

Modify local and remote repository

```
> git log --graph --oneline main origin/main
* d3387a0 (HEAD -> main) add a last line
* a629572 (origin/main) add scenarios section in README.md
* 6ae8b60 Merge branch 'main' of https://gitlab.univ-lorraine.fr/cirstea6/acl4demo
| \
| * 3a220b3 Minimalist README.md
| * 2762c2b Only Getting started and Add your files in README.md
| * cd73558 Initial commit
* 5bbf98a fix conflicts
| \
| * b8881d9 (poc) add more lines
| * 303a2c2 add a new line
* | 4e08a78 underline
* | e8d1413 start with title
|/
* 9604ca0 create new file
* 97207ec more files
* 4e14839 create file cl
```

Specify workflow

```
> git pull
...
hint: git config pull.rebase false # merge
hint: git config pull.rebase true # rebase
hint: git config pull.ff only # fast-forward only
...
```

Opérations

- ▶ associer une balise : `git tag`
- ▶ déplacer fichier(s) : `git mv`
- ▶ supprimer un fichier : `git rm`

Associate a name to (current) commit

```
> git tag v1.0
> git show v1.0 --oneline
d3387a0 (HEAD -> main, tag: v1.0) add a last line
...
> git tag v0.9 6ae8b60
> git tag
v0.9
```

Push the tags remotely

```
> git push origin tag v1.0
```

Fetch the tags

```
> git fetch --tags
> git tag
v0.1
v0.9
v1.0
```

Use tags as identifier for operations

```
> git log v0.9.. --oneline
d3387a0 (HEAD -> main, tag: v1.0, origin/main) add a last line
16559d9 add scenario
a629572 add scenarios section in README.md
...
```

Pull requests

- ▶ Créer une branche localement
 - ▶ `git switch -c greatfeature`
- ▶ Commiter les fonctionnalités dans cette branche
- ▶ Propager la branche sur le dépôt distant
 - ▶ `git push -u origin greatfeature`
- ▶ Request

Create branch

```
> git switch -c greatfeature
> git branch
* greatfeature
  main
...
> git commit -a -m"new feature started"
...
> git commit -a -m"add scenario for new feature"
> git log --graph --oneline
* 194e702 (HEAD -> greatfeature, origin/greatfeature) add scenario for new feature
* 1f12417 new feature started
* d3387a0 (tag: v1.0, main) add a last line
* 16559d9 add scenario
* a629572 add scenarios section in README.md
...
```

Push branch remotely

```
> git push -u origin greatfeature
> git branch -r
origin/HEAD -> origin/main
origin/greatfeature
origin/main
```

Initiate merge request

🏠

+

📄

🔗

📧

🔍 Search or go to...

Project

A

ACL4demo

📌

Pinned

🔍

Issues0

🔗

Merge requests0

CIRSTEA Horatiu / ACL4demo / Merge requests / New

New merge request

Source branch

cirstea6/acl4demo

greatfeature

add scenario for new feature
hcirstea authored May 29, 2024

194e702b

Compare branches and continue

Target branch

cirstea6/acl4demo

main

add a last line
hcirstea authored May 27, 2024

d3387a0e

Validate merge request

🏠

+

📄

🔗1

📧1

🔍 Search or go to...

Project

A

ACL4demo

📌

Pinned

🔍

Issues0

🔗

Merge requests1

👤

Manage>

📅

Plan>

📄

Code>

🔗

Merge requests1

📄

Repository

CIRSTEA Horatiu / ACL4demo / Merge requests / !1

Greatfeature

Open

CIRSTEA Horatiu requested to merge greatfeature into main just now

Overview0

Commits-

Pipelines0

Changes-

00

8

Revoke approval

Approved by you

Ready to merge!

☒ Delete source branch

☐ Squash commits

☐ Edit commit message

2 commits and 1 merge commit will be added to main.

Merge

Check merge

```
> git fetch origin main
> git log --graph --oneline origin/main
*   a68269c (origin/main, origin/HEAD) Merge branch 'greatfeature' into 'main'
| \
| * 194e702 (HEAD -> greatfeature, origin/greatfeature) add scenario for new feature
| * 1f12417 new feature started
| /
* d3387a0 (tag: v1.0, main) add a last line
...
```

Merge to main and delete branch (locally)

```
> git checkout main
Your branch is behind 'origin/main' by 3 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)
> git pull
Fast-forward
 README.md | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
> git log --graph --oneline
*   a68269c (origin/main, origin/HEAD) Merge branch 'greatfeature' into 'main'
| \
| * 194e702 (HEAD -> greatfeature, origin/greatfeature) add scenario for new feature
| * 1f12417 new feature started
| /
* d3387a0 (tag: v1.0, main) add a last line
...
> git branch -d greatfeature
Deleted branch greatfeature (was 194e702).
> git branch
* main
```

Numérotation versions

Major.Minor.Patch

- **Major** : *Backwards incompatible* breaking changes
- **Minor** : New features *backwards compatible*
- **Patch** : *Backwards compatible* bug fixes only

Pre-release versions :

- M.m.p-**alpha** : version interne (bugs potentiels)
- M.m.p-**beta** : version publique (tests clients)

Bonnes pratiques

- mettre à jour (update/pull) avant de tester
- ne pas casser le build : tester avant de committer
- committer souvent, par petits incréments
- mettre des commentaires utiles pour chaque commit (commit -m “mon commentaire”)

Quizz



1

Allez sur wooclap.com

2

Entrez le code d'événement dans le bandeau supérieur

Code d'événement
QCBYGD

 Activer les réponses par SMS

Outils de build

Outils de build

- Simplifier/automatiser le processus de compilation/test/construction/déploiement d'un projet
- Exemples d'outils:
 - ▶ Make
 - ▶ Ant
 - ▶ Maven

make

- Ancêtre de Ant
- Très utilisé dans le monde C/C++
- Configuration par un fichier **Makefile** qui décrit des règles:

cible : dépendance1, dépendance2,...

commande

simple : hello.c hellolib.c

gcc -o hello hello.c hellolib.c -I.

cible : dépendances

commande

all : hello.c hellolib.c

gcc -o hello hello.c hellolib.c -I.

hello.o : hello.c

gcc -c -o hello.o hello.c -I.

hellolib.o : hellolib.c

gcc -c -o hellolib.o hellolib.c -I.

variables, motifs, macros

```
CC = gcc
```

```
CFLAGS = -I.
```

```
DEPS = hellolib.h
```

```
OBJ = hello.o hellolib.o
```

```
hello : $(OBJ)
```

```
    gcc -o $@ $^ $(CFLAGS)
```

```
clean :
```

```
    rm -rf *.o
```

```
    rm hello
```

```
> ls
Makefile
hello.c
hellolib.c
hellolib.h

> make hello
gcc -c -o hello.o hello.c -I.
gcc -c -o hellolib.o hellolib.c -I.
gcc -o hello hello.o hellolib.o -I.

> ls
Makefile
hello
hello.c
hello.o
hellolib.c
hellolib.h
hellolib.o
```

make

- prend comme paramètre une cible :

`make hello`

- si pas de paramètre, make applique la première cible du fichier `Makefile`
 - ne compile que ce qui doit l'être
- ⇒ Difficile à utiliser pour un gros projet sans l'aide d'outils annexes de génération

Apache Ant

- Séquenceur de tâches
- Equivalent de make pour le monde Java
- Configuration des tâches (<target>) par un fichier XML (build.xml)
- <http://ant.apache.org>

build.xml

- fichier XML conforme à la grammaire définie par Ant
- racine `<project>` contenant
 - ▶ des `<target>` = des tâches à accomplir
 - ▶ des `<property>` = des variables
 - ▶ ...

Example

```
<project default="hello">  
  <target name="hello">  
    <echo message="Hello World" />  
  </target>  
</project>
```

⇒ attributes `project`

- ▶ `default`
- ▶ `name`
- ▶ `basedir`

```
> ant  
Buildfile: build.xml  
  
hello:  
    [echo] Hello World  
  
BUILD SUCCESSFUL  
Total time: 1 seconds
```

Example

```
<project default="hello">  
  <target name="hello">  
    <echo message="Hello World" />  
  </target>  
  <target name="goodbye">  
    <echo message="Goodbye World" />  
  </target>  
</project>
```

```
> ant  
Buildfile: build.xml
```

```
hello:  
    [echo] Hello World
```

```
BUILD SUCCESSFUL  
Total time: 1 seconds
```

```
> ant goodbye  
Buildfile: build.xml
```

```
hello:  
    [echo] Goodbye World
```

```
BUILD SUCCESSFUL  
Total time: 1 seconds
```

Example

```
<project default="compile">  
  <target name="compile">  
    <javac srcdir="." />  
  </target>  
</project>
```

```
> ant
```

```
Buildfile: build.xml
```

```
compile:
```

```
    [javac] Compiling 1 source file
```

```
BUILD SUCCESSFUL
```

```
Total time: 1 seconds
```

```
> ls
```

```
Hello.class  Hello.java  build.xml
```

```
<project default="hello">
  <target name="hello">
    <echo message="Hello World" />
  </target>
  <target name="goodbye">
    <echo message="Goodbye World" />
  </target>
  <target name="all" depends="hello,goodbye">
    <echo message="Done" />
  </target>
</project>
```

Attention aux cycles !

```
> ant all
Buildfile: build.xml
hello:
    [echo] Hello World
goodbye:
    [echo] Goodbye World
all:
    [echo] Done
BUILD SUCCESSFUL
Total time: 1 seconds
```

Propriétés

- définition de variables utilisables dans le fichier

- attributs

- ▶ **name** : nom de la variable
- ▶ **value** : valeur de la variable

```
<property name="src.dir" value="src" />
```

- utilisation

```
<javac srcdir="${src.dir}" ...>
```

Tâches

- la plus petite unité de travail
- peuvent opérer sur plusieurs fichiers avec la même opération appliquée à chaque fichier
 - ▶ **echo** : afficher un message (attribut : message)
 - ▶ **delete** : supprimer un dossier/fichier (dir/file)
 - ▶ **mkdir** : créer un dossier (dir)
 - ▶ **javac** : compiler (srcdir et destdir)
 - ▶ **jar** : créer une archive (destfile, basedir)
 - ▶ ...

Exemple projet Java

basedir

|----- src

|----main

|----acl

|----- Hello.java

|----- java

```
<project name="HelloApp" basedir="." default="main">
```

```
<property name="src.dir" value="src/main"/>
```

```
<property name="build.dir" value="build"/>
```

```
<property name="class.dir" value="${build.dir}/classes"/>
```

```
<property name="jar.dir" value="${build.dir}/jar"/>
```

|----acl

```
<property name="main-class" value="acl.HelloApp"/>
```

|----- Hello.class

|----- .class

```
<target name="compile">
```

```
<mkdir dir="${class.jar.dir}"/>
```

```
<javac srcdir="${src.dir}" destdir="${class.dir}"/>
```

```
</target>
```

propriétés
génériques

propriété
spécifique

classes

HelloApp.jar

Exemple projet Java

```
<project name="HelloApp" basedir="." default="main">
...
<target name="jar" depends="compile">
  <mkdir dir="${jar.dir}" />
  <jar destfile="${jar.dir}/${ant.project.name}.jar"
    basedir="${class.dir}">
    <manifest>
      <attribute name="Main-Class" value="${main-class}" />
    </manifest>
  </jar>
</target>

<target name="run" depends="jar">
  <java jar="${jar.dir}/${ant.project.name}.jar" fork="true" />
</target>
```

`java -jar build/jar/HelloApp.jar`

attribut du tag `project`

Exemple projet Java

```
<project name="HelloApp" basedir="." default="main">
  ...
  <target name="jar" depends="compile">
    <mkdir dir="${jar.dir}" />
    <jar destfile="${jar.dir}/${ant.project.name}.jar"
        basedir="${class.dir}">
    </jar>
  </target>

  <target name="run" depends="jar">
    <java classname="${main-class}" fork="true">
      <classpath path="${jar.dir}/${ant.project.name}.jar" />
    </java>
  </target>
```



```
java -cp build/jar/HelloApp.jar acl.Hello
```

Exemple projet Java

```
> ls
```

```
build.xml  src
```

```
> ant jar
```

```
Buildfile: <bd>/build.xml
```

```
compile:
```

```
    [mkdir] Created dir: <bd>/build/classes
```

```
    [javac] Compiling 1 source file to <bd>/build/classes
```

```
jar:
```

```
    [mkdir] Created dir: <bd>/build/jar
```

```
    [jar] Building jar: <bd>/build/jar/HelloApp.jar
```

```
Total time: 1 second
```

```
> ls
```

```
build  build.xml  src
```

```
> ls build
```

```
classes jar
```

Exemple projet Java

```
<project name="HelloApp" basedir="." default="main">
  ...
  <target name="run" depends="jar">
    <java jar="${jar.dir}/${ant.project.name}.jar" fork="true"/>
  </target>

  <target name="clean">
    <delete dir="${build.dir}"/>
  </target>

  <target name="clean-jar">
    <delete file="${jar.dir}/${ant.project.name}.jar"/>
  </target>

  <target name="clean-compile">
    <delete>
      <fileset dir="${class.dir}" includes="**/*.class"/>
    </delete>
  </target>

  <target name="clean-build" depends="clean,jar"/>

  <target name="main" depends="clean,run"/>
</project>
```

Tests

```
<project name="HelloApp" basedir="." default="main">
  ...
  <property name="test.dir" value="src/test"/>
  <property name="test.report.dir" value="${build.dir}/testrep"/>

  <target name="compile">
    <mkdir dir="${class.dir}"/>
    <javac srcdir="${src.dir}" destdir="${class.dir}"/>
    <javac srcdir="${test.dir}" destdir="${class.dir}"/>
  </target>
  ...
```

** On suppose que les jar de JUnit sont dans le classpath*

Tests

...

```
<target name="junit" depends="compile">
  <mkdir dir="${test.report.dir}" />
  <junit printsummary="yes" fork="yes" haltonfailure="off">
    <classpath>
      <pathelement location="${class.dir}" />
    </classpath>

    <formatter type="plain" />

    <!-- Test by test -->
    <test name="acl.HelloTest" todir="${test.report.dir}" />

    <!-- Same with batch testing -->
    <batchtest fork="yes" todir="${test.report.dir}">
      <fileset dir="${class.dir}">
        <include name="acl/*Test.class" />
      </fileset>
    </batchtest>
  </junit>
</target>
```

Apache Ant

- outil puissant souvent intégré dans les IDE
description de tâches (<target>) dans un fichier XML
- chaque <target> correspond à une étape du processus de construction
- extensible : il est possible d'ajouter d'autres tâches et éléments.

Maven

- Outil pour faciliter la gestion de projets Java
 - ▶ utilise une structure ***standard*** de projet
 - ▶ suit un cycle de vie de projet ***standard***
 - ▶ utilise un modèle abstrait de projet (POM)

Maven

- **Automatisation**

- ▶ compilation
- ▶ récupération de dépendances
- ▶ déploiement (génération jar, war, ...)
- ▶ test unitaires
- ▶ génération de la documentation
- ▶ production site web du projet (membres, état d'avancement, rapports sur les tests, ...)

Structure projet standard

`/src` : sources du projet

`/src/main` : fichiers source principaux

`/src/main/java` : code source Java

`/src/main/resources` : fichiers de ressources

...

`/src/test` : fichiers de test

`/src/test/java` : code source de test

...

`/target` : fichiers résultat, binaires, packages
générés, résultats des tests

➤ Convention plutôt que configuration

Phases standard

validate : vérification projet correct et info dispo

compile : compilation du code source

test : exécution tests unitaires

package : assemblage binaires dans un jar, war, ...

integration-test : test du code dans un clone
de l'environnement de déploiement

verify : vérification intégrité/qualité de l'archive

install : ajouter package sur le dépôt local

deploy : ajouter package sur le dépôt public

➤ Phases mappées à des *goals*

Project Object Model

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>fr.ul.acl</groupId>
  <artifactId>acl-maven-app</artifactId>
  <packaging>jar</packaging>
  <version>0.1.0</version>
  <name>acl-maven-app</name>
  <url>http://maven.apache.org</url>
</project>
```

version de l'object model

identifiant organisation à l'origine du projet

identifiant de l'artefact généré par le projet

type packaging

version de l'artefact généré

nom du projet utilisé pour l'affichage

adresse du site du projet

acl-maven-app

```
|
|---src/main/java/fr/ul/acl/...
|---src/test/java/fr/ul/acl/...
...
|---target/acl-maven-app-0.1.0.jar
...
```

Créer la structure

```
> mkdir -p src/main/java/fr/ul/acl/greet  
> mkdir -p src/main/java/fr/ul/acl/start
```

Créer les classes

```
package fr.ul.acl.greet;  
  
public class Greeter {  
    public String greet(String name) {  
        return "Hello, " + name + "!";  
    }  
}
```

```
package fr.ul.acl.start;  
  
import fr.ul.acl.greet.Greeter;  
  
public class Main {  
    public static void main(String[] args) {  
        Greeter greet = new Greeter();  
        System.out.println(greet.greet("JB"));  
    }  
}
```

Compiler

```
> mvn compile  
> ls  
pom.xml    src    target  
> ls target  
classes    generated-sources    maven-status
```

Packager

```
> mvn package  
> ls target  
acl-maven-app-0.1.0.jar    classes    generated-sources    maven-status
```

POM - dépendances

```
<project xmlns="..." xmlns:xsi="..." ...>
  ...
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Créer la structure

```
> mkdir -p src/test/java/fr/ul/acl/greet
```

Créer les tests

```
package fr.ul.acl.greet;

import static org.junit.Assert.*;
import org.junit.Test;

public class GreeterTest {
    @Test
    public void testGreet() throws Exception {
        Greeter greeter = new Greeter();
        assertEquals("Hello, World!", greeter.greet("World"));
    }
}
```

Tester

```
> mvn test
```

```
...
[INFO] -----
[INFO]  T E S T S
[INFO] -----
[INFO] Running fr.ul.acl.greet.GreeterTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.03 s -- in fr.ul.acl.greet.GreeterTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```

POM - exec

```
<project xmlns="..." xmlns:xsi="..." ...>
...
<build>
  <plugins>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>exec-maven-plugin</artifactId>
      <version>1.4.0</version>
      <configuration>
        <mainClass>fr.ul.acl.start.Main</mainClass>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

```
> mvn exec:java
```

Maven

- Comprendre un projet Maven permet de comprendre tous les projets Maven
- Quand ces conventions sont respectées, tout est quasi-automatique

Quizz



1

Allez sur wooclap.com

2

Entrez le code d'événement dans le bandeau supérieur

Code d'événement

IJBDEL



Activer les réponses par SMS

Framework d'intégration continue

- Planification, automatisation et intégration des tâches de compilation, de tests
- Gestion des dépendances entre tâches (ex. : les tests unitaires sont lancés lorsque la compilation s'est bien passée)
- Notifications
- Utilise des outils de build, de versionning, etc.
- Exemples : Hudson/Jenkins, Tinderbox, etc.

Aperçu CI/CD (gitlab)

```
# .gitlab-ci.yml: configuration file.
variables:
  # MAVEN_OPTS: local repository path.
  MAVEN_OPTS: -Dmaven.repo.local=.m2/repository
# image: Docker image to be used in pipeline.
image: maven:latest
# stages: pipeline stages.
stages:
  - build
  - test
  - package
  - release

# cache: to speed up the pipeline.
# path Maven copies files like the .jar to
# when it runs the cmds in build life cycle
cache:
  paths:
    - .m2/repository
    - target

# first job: compile the project
# (7th step in the Maven build life cycle).
build_job:
  stage: build
  tags:
    - ondocker
  script:
    - echo "Maven compile started"
    - "mvn compile"
```

```
# second job: run unit tests (15th).
test_job:
  stage: test
  tags:
    - ondocker
  script:
    - echo "Maven test started"
    - "mvn test"

# third job: create jar package (17th).
package_job:
  stage: package
  tags:
    - ondocker
  script:
    - echo "Maven packaging started"
    - "mvn package"

# fourth job: release the project
release_job:
  stage: release
  image: registry.gitlab.com/gitlab-org/release-cli:latest
  rules:
    - if: $CI_COMMIT_TAG # when tag created
  script:
    - echo "Releasing $CI_COMMIT_TAG"
  release:
    tag_name: '$CI_COMMIT_TAG'
    description: '$CI_COMMIT_TAG'
```