



Cours 2 : Les bases.

Repository git

Un repository git c'est :

- Un répertoire de votre système de fichier
- Comprenant un repertoire **.git**

```
drwxr-xr-x 16 victor staff 544 Jul 23 18:43 .git
-rw-r--r-- 1 victor staff 44 Jul 16 14:23 .gitignore
-rw-r--r-- 1 victor staff 1083 Jul 16 14:23 LICENSE
-rw-r--r-- 1 victor staff 4214 Jul 22 12:35 README.md
drwxr-xr-x 4 victor staff 136 Jul 23 11:51 converter
drwxr-xr-x 3 victor staff 102 Jul 16 14:39 docs
drwxr-xr-x 12 victor staff 408 Jul 16 14:23 node_modules
-rw-r--r-- 1 victor staff 1303 Jul 21 17:55 npm-debug.log
-rw-r--r-- 1 victor staff 1225 Jul 16 14:23 package.json
drwxr-xr-x 5 victor staff 170 Jul 16 14:23 public
-rw-r--r-- 1 victor staff 418 Jul 16 14:23 server.js
drwxr-xr-x 3 victor staff 102 Jul 16 14:23 views
```

Initialiser un repository : **Git init**

Récupérer un projet existant

```
1 | git clone URL [DIR]
```



Remarquez qu'il y a trois types d'adresses :

SSH clone URL

git@github.com:sat



You can clone with [HTTPS](#), [SSH](#),
or [Subversion](#). [?](#)

1. Par le protocole https.
2. Depuis un serveur subversion ¹ : `git svn clone`.
3. Avec le protocole **git** : nécessite une clef ssh ²

¹ Pour plus d'informations. Je n'aborderai pas ici la question de subversion.

² Pour générer une paire de clef, suivez le tutoriel de github ici

Visualiser un historique

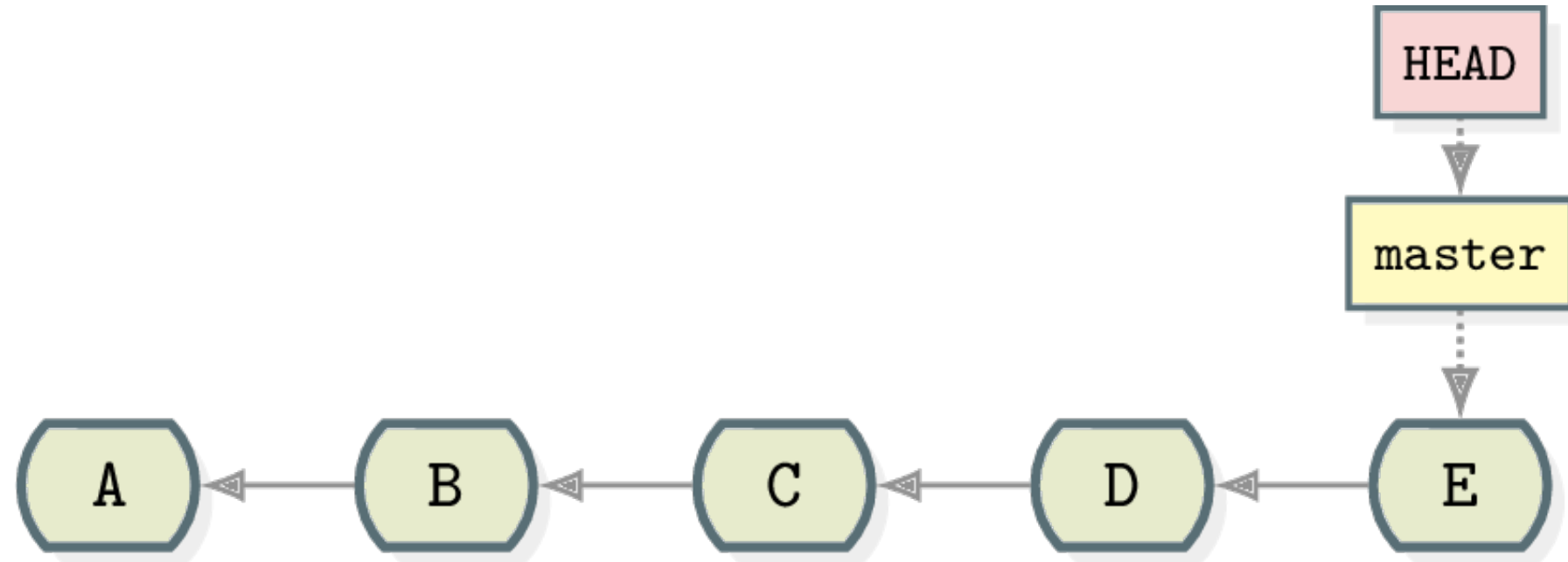


Figure 1: Un historique simple sous git

Pour afficher l'historique :

```
1 | git log [REF]
```

Pour visualiser un commit :

```
1 | git show [REF]
```

Naviguer dans un historique

```
1 | git checkout REF
```

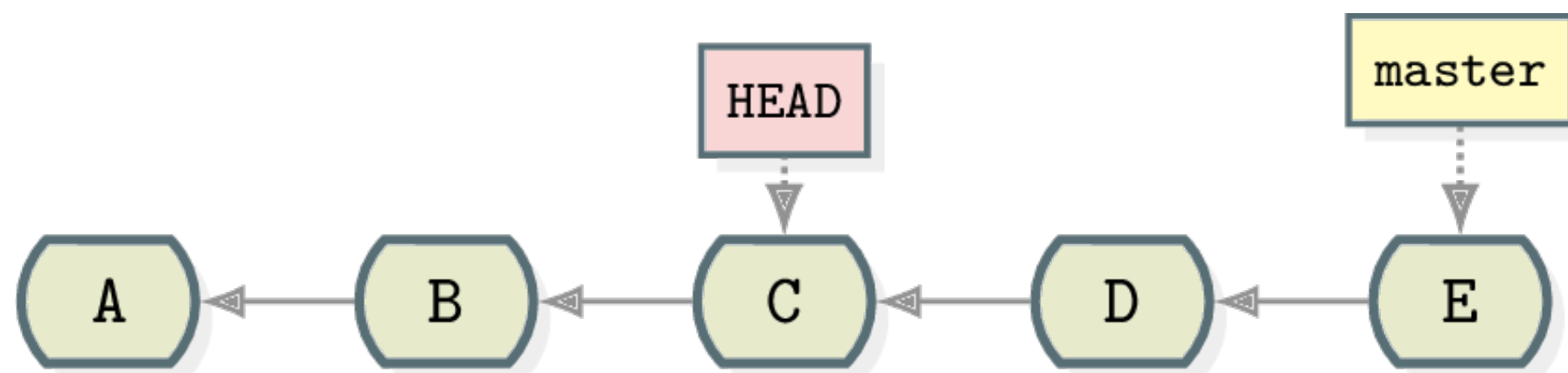
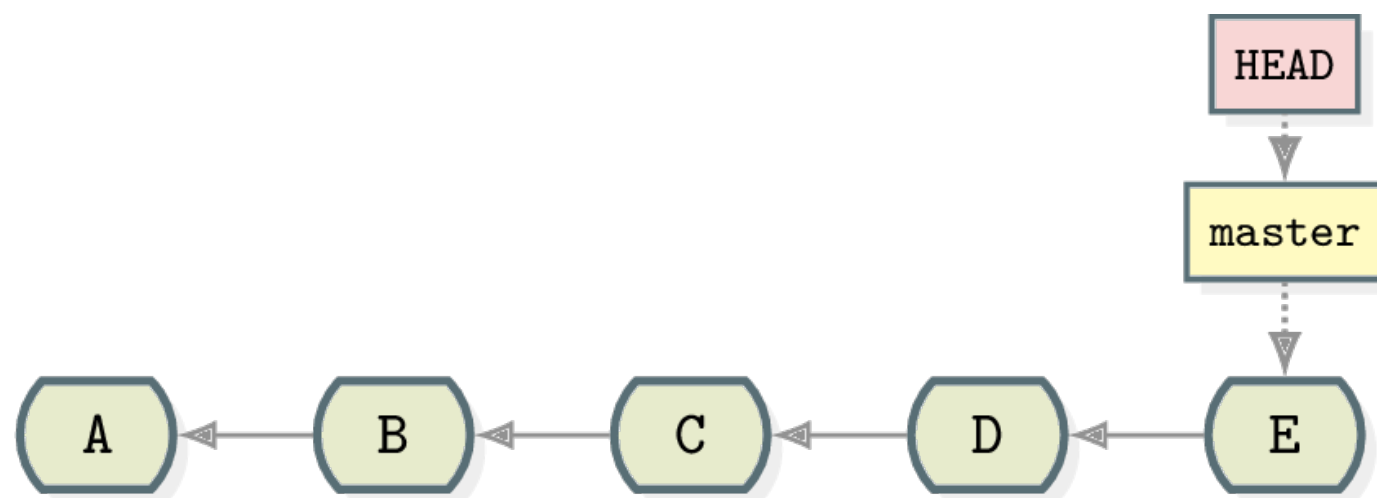


Figure 1: git checkout C

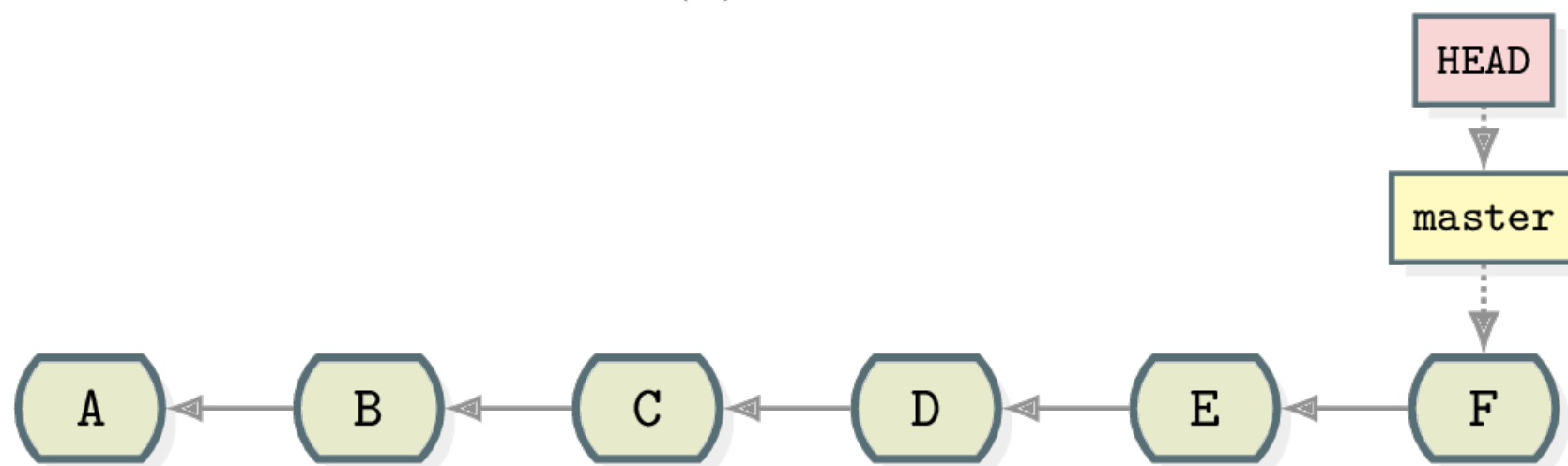
git show

Créer son premier changement

Notre objectif



(a) Avant



(b) Après

Étape 1 : Indexer le contenu de son changement.

Copie locale

Historique de
git

Copie locale

git commit

Historique de
git

Copie locale

Staging area

Historique de
git



Étape 2 : Commiter.



Les 4 états d'un fichier dans git

Non suivi

Modifié

Indexé

Commité

Comment connaître l'état d'un fichier :

```
1 | git status
```

Exemple

```
→ Realtime-Markdown-Viewer git:(master) ✗ git st
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   README.md
    modified:   package.json

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

    new.txt
```

Visualiser le contenu des changements

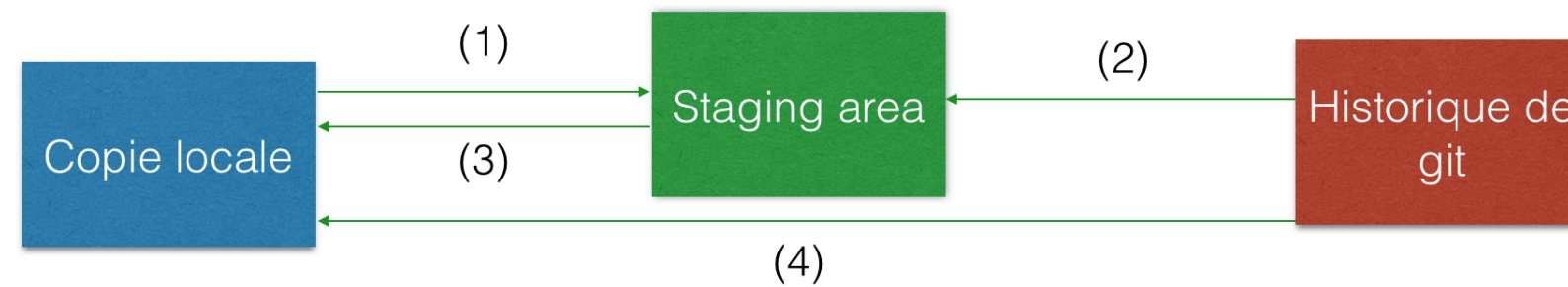
Connaître les changements dans ma copie locale :

```
1 | git diff
```

Connaître les changements dans la staging area :

```
1 | git diff --staged
```

Changer l'état d'un fichier



(1)

```
1 | git add
2 | git add -u ( fichiers connus par git )
3 | git rm
4 | git mv
```

(2)

```
1 | git reset HEAD <FILE>
```

(4) = (2) + (3)

```
1 | git checkout HEAD <FILE>
```

(3)

```
1 | git checkout -- <FILE>
```

git status est votre ami 😊

```
→ Realtime-Markdown-Viewer git:(master) X git st
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   README.md
    modified:   package.json

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

    new.txt
```

Enregistrer un changement ³



³ Le commit s'accompagne toujours d'un message, ajouter l'option -m. Ou sinon votre éditeur de texte favori s'ouvrira et vous pourrez rentrer votre message.

Git Gui (Realtime-Markdown-Viewer) /Users/victor/Elephorm/Realtime-Markdown-Viewer

Current Branch: master

Unstaged Changes

converter/markdown.js
converter/tests/features/horizontal_line.mc

copie locale

Staged Changes (Will Commit)

converter/tests/features/horizontal_line.ht

staging area

Modified, not staged File: converter/markdown.js

```
@@ -6,14 +6,24 @@  
    str = str.replace(stra[0], '<h' + count + '>' + stra[2] + '</h' + count + '>' + '\n');  
  }  
  return str;  
}  
  
+ var parseHorizontaleLine = function(str) {  
+   var horizontalRegExp = /^(?:([\*\-\_] ?)+)\1$/gm;  
+   var stra = [];  
+   while ((stra = horizontalRegExp.exec(str)) !== null) {  
+     str = str.replace(stra[0], '\n<hr/>\n');  
+   }  
+   return str;  
+ }  
  
var markdown = {  
  parse: function (str, strict) {  
    'use strict';  
    str = parseHeadline(str);  
+   str = parseHorizontaleLine(str);  
    return str;  
  }  
};  
  
module.exports = markdown;
```

git diff | git diff --staged

Commit Message:

☒ New Commit ☐ Amend Last Commit

Rescan
Stage Changed
Sign Off
Commit
Push

message de commit

Ready.