GUIDE D'UTILISATION D'ATOM

July 12, 2021

Amin Mekacher
Centre des Littératures en Suisse Romande

1 Guide d'utilisation du terminal

Les sections suivantes nécessitant l'utilisation du terminal pour installer la machine virtuelle et manipuler des fichiers, cette section vise à offrir un tutoriel vis-à-vis des commandes les plus fréquemment utilisées dans le terminal, avec des exemples d'utilisation. A noter que ceci n'est valable que pour les machines tournant sur un système Unix, à savoir MacOS ou Linux.

Le terminal est une application permettant de naviguer dans les dossiers présents dans la machine et d'effectuer de nombreuses manipulations, que ce soit de déplacer des fichiers, modifier les droits d'accès ou encore de télécharger des applications. En lançant le terminal, l'interface graphique ressemble à ceci :

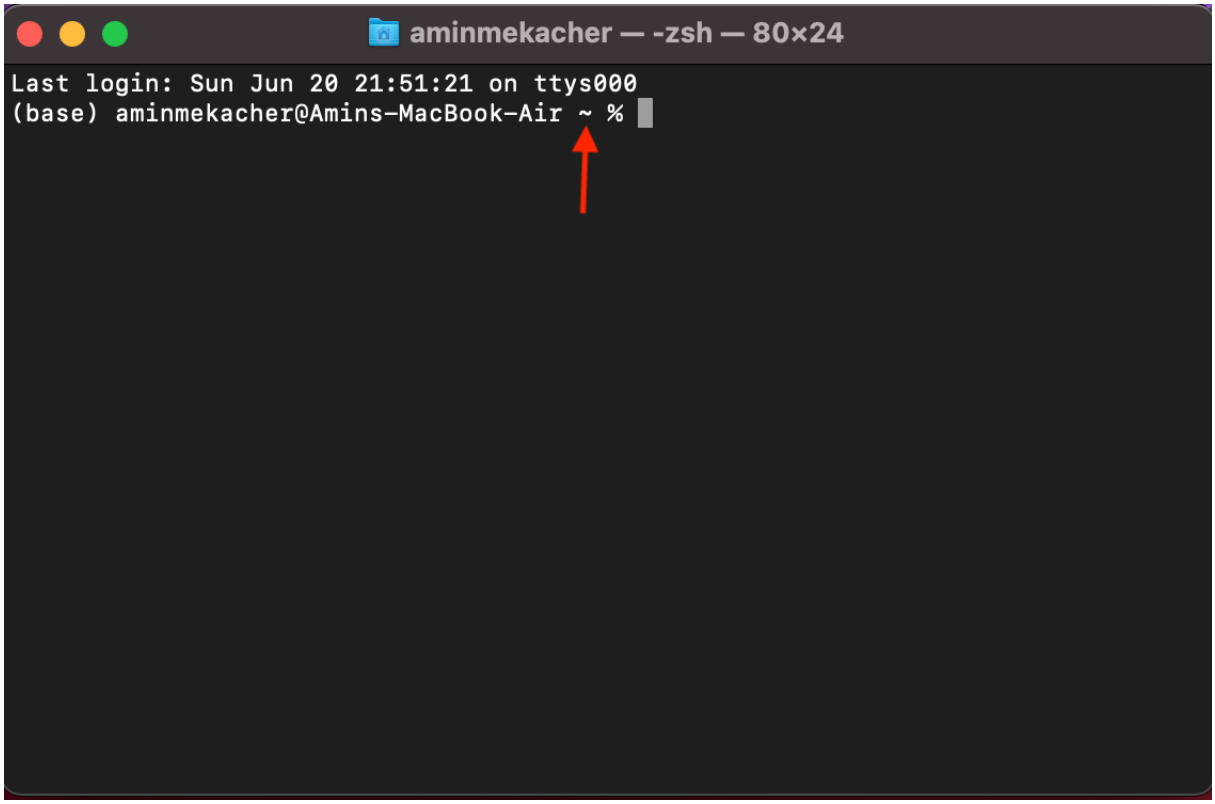


Figure 1: Interface graphique du terminal

A tout moment, le terminal indique le dossier dans lequel il est couramment situé. Cette information est indiquée à la fin de la ligne, avec le symbole s̃ur la capture d'écran ci-dessus, ce qui indique le dossier "Home" pour les systèmes d'exploitation Mac et Linux. Le symbole %, en fin de ligne, est appelé l'invite de commande, et indique à l'utilisateur que le terminal est prêt à recevoir des instructions. Les principales commandes sont listées ci-dessous, avec à chaque fois un exemple d'utilisation.

1.1 Terminologie

Certains termes sont couramment utilisés

- **Dossier parent :** Il est parfois nécessaire de pouvoir indiquer le dossier parent dans une

commande, que ce soit pour y accéder ou pour y déplacer un fichier. La syntaxe employée par le terminal est ...

- **Chemin relatif v/s absolu :** Un chemin est utilisé pour indiquer la localisation d'un fichier ou dossier. Un chemin est dit absolu s'il indique sa position relativement à la source du disque dur. Par exemple, `/home/Bob/Documents/Projet 1/` est un chemin absolu. A l'inverse, un chemin relatif indique le cheminement nécessaire pour rejoindre un dossier ou fichier depuis le dossier courant du terminal. Si nous nous trouvons par exemple dans le dossier `Documents/Projet 2`, alors le chemin relatif pour arriver au dossier Projet 1 serait `../Projet 1`.

ls. Cette commande, qui est une abbréviation de "list files", permet d'afficher une liste de tous les fichiers et dossiers compris dans le dossier où se trouve actuellement le terminal. Cette commande est très pratique pour se repérer dans l'arborescence et pour s'assurer de l'existence d'un fichier ou dossier que l'on souhaite modifier à l'aide d'une autre commande.

Les captures d'écran suivantes montrent le contenu du dossier *Home*, tel que l'on peut le voir dans le navigateur Finder, ainsi que le résultat de la commande `ls` lancée dans le même dossier.

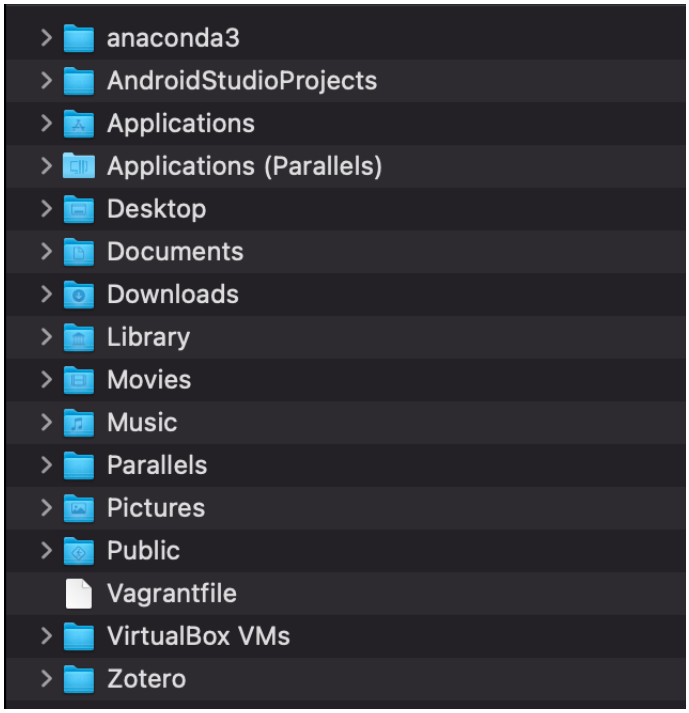


Figure 2: Contenu du dossier *Home*, indiqué par Finder

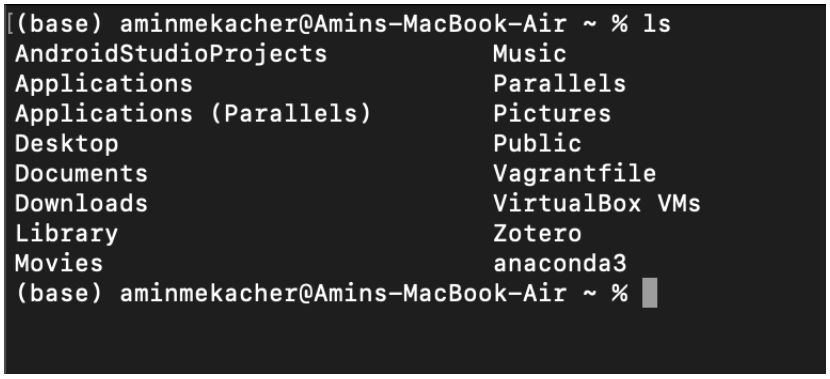


Figure 3: Contenu du dossier *Home*, indiqué par la commande `ls`

cd. Abbréviation pour *change directory*, la commande `cd` permet de se déplacer dans un des dossier compris dans le dossier courant, ou de remonter dans l'arborescence. La syntaxe pour chacune de ces options est la suivante :

- **Pour accéder à un dossier enfant :** Il suffit d'indiquer le nom du dossier suite à la commande `cd`, comme illustré dans la capture ci-dessous :

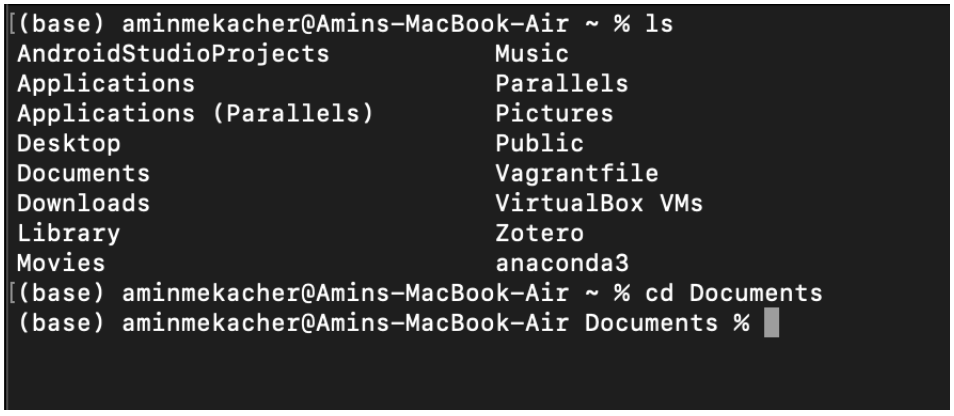


Figure 4: Premier exemple d'exécution de la commande `cd`

- **Pour accéder au dossier parent :** Le dossier parent est représenté par la syntaxe "..", qui peut ainsi être indiquée à la suite de la commande `cd` pour remonter dans la hiérarchie.

  Comme le montre la capture suivante, il est aussi possible d'enchaîner les sauts de dossier dans une même commande, par exemple pour passer de `~/Documents` à `~/Downloads` :

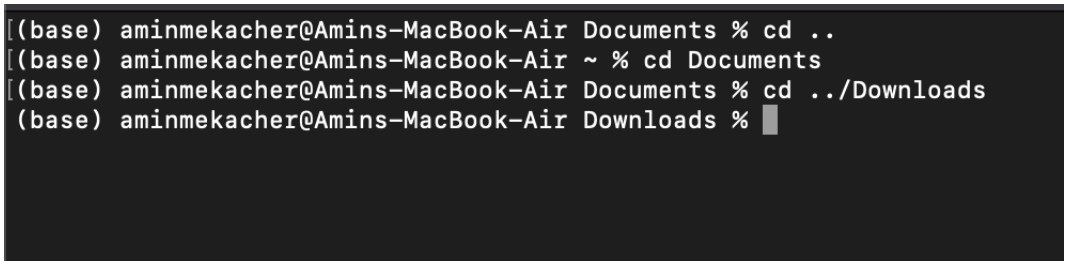


Figure 5: Deuxième exemple d'exécution de la commande `cd`

mv. La commande `move` (abbréviation de *move*) permet de déplacer des fichiers ou dossier d'un dossier à un autre. Pour cela, il est nécessaire d'indiquer en premier argument le chemin relatif du fichier / dossier à déplacer, et en deuxième argument le chemin relatif du dossier dans lequel il doit être déplacé. La syntaxe de la commande est détaillée dans la capture d'écran suivante :

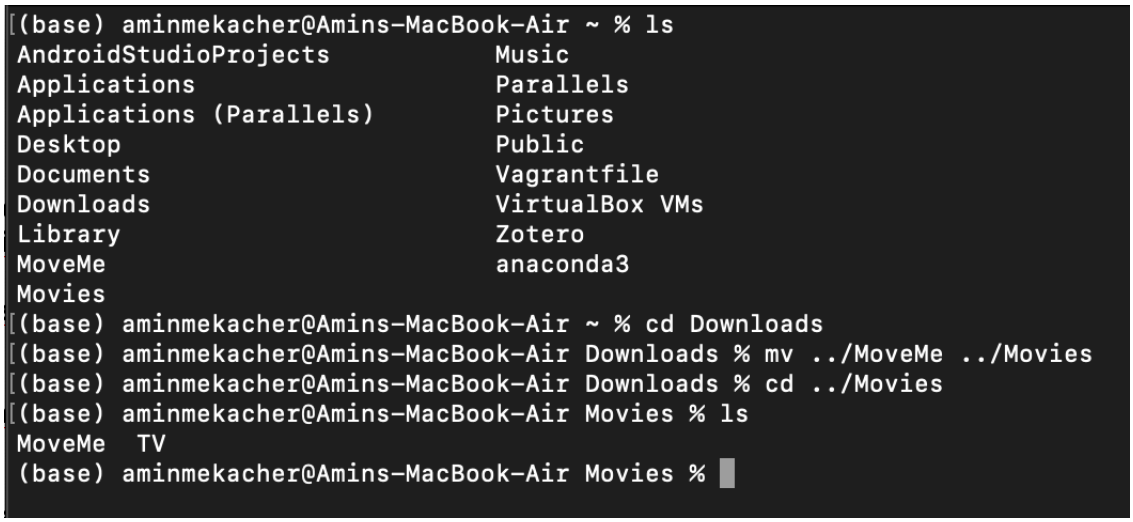


Figure 6: Exemple d'exécution de la commande `mv`

2 Configuration locale d'Atom

Afin de pouvoir tester localement des modifications visuelles d'Atom sans toucher à la version accessible publiquement, il est possible d'installer une machine virtuelle à l'aide de Vagrant et d'utiliser le repository GitHub de la faculté des Lettres pour obtenir la dernière version du code.

Cette section va décrire les étapes suivantes :

1. Installation d'une machine virtuelle avec Atom, configuration du dossier synchronisé.
2. Déplacement de fichiers présents dans la machine virtuelle au dossier synchronisé, à des fins de modifications.

2.1 Installation de Vagrant

Pour pouvoir lancer Vagrant depuis le terminal, il est d'abord nécessaire de le télécharger sur votre machine. Le site officiel donne toutes les indications nécessaires pour l'installation.

2.2 Installation d'Atom

Atom fournit une machine virtuelle prête à l'emploi, qui peut être installée via Vagrant. Pour ce faire, les étapes suivantes doivent être exécutées via le Terminal :

1. Créer un dossier pour y installer la machine virtuelle et y accéder via le terminal. Dans le cadre de ce tutoriel, la dossier a été crée dans le dossier de téléchargement, le chemin pour y accéder étant alors ∼/Downloads/atom-vagrant.
2. Lancer le terminal. Rejoindre le dossier crée à l'étape précédente en copiant la commande suivante :

   ```
   cd Downloads/atom-vagrant/
   ```

   La commande ci-dessus doit être adaptée en fonction du chemin pour accéder au dossier dédié à la machine virtuelle.
3. Exécutez la commande suivante dans le terminal, une fois avoir rejoint le dossier comme décrit dans l'étape précédente :

   ```
   vagrant init artefactual/atom
   ```

   Cette commande va créer un fichier Vagrantfile dans le dossier, qui contient toutes les instructions nécessaires à la création de la machine virtuelle.
4. Pour installer la machine virtuelle, lancez la commande suivante :

   ```
   vagrant up
   ```

   Cette commande va télécharger tous les fichiers nécessaires pour initialiser la machine virtuelle, ce qui peut prendre du temps en fonction de votre débit de téléchargement. Les dernières lignes dans le terminal indiquent le succès de l'opération et le chemin du dossier partagé dans la machine hôte.

```
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
==> default: Configuring and enabling network interfaces...
==> default: Mounting shared folders...
    default: /vagrant => /Users/username/Downloads/atom-vagrant
```

5. Afin de pouvoir accéder à la machine virtuelle depuis le terminal, lancer la commande suivante :

```
vagrant ssh
```

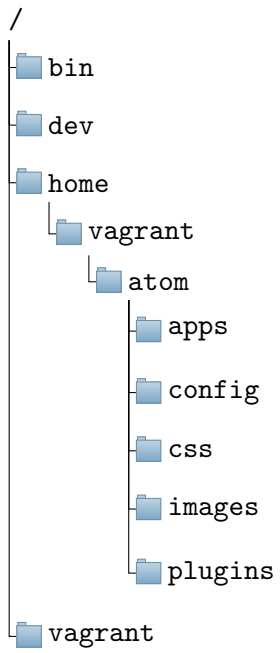
   La fenêtre du terminal sera alors redirigée dans la machine virtuelle. Vous pouvez confirmer cela en vérifiant que le dossier courant indiqué dans l'invite de commande est bien lié à vagrant :

```
[~]
vagrant$
```

Pour pouvoir relancer Atom après l'avoir installé, il suffit de se déplacer via le terminal jusqu'au dossier d'installation et de lancer les deux dernières commandes, à savoir `vagrant up` et `vagrant ssh`.

2.3 Accès au dossier synchronisé

Par défaut, le dossier crée au tout début de ce tutoriel et dans lequel a été installé le Vagrantfile est le dossier synchronisé du côté de la machine hôte. Ce dernier se retrouve dans la machine virtuelle, comme enfant direct du dossier *root* ('/'). Le schéma suivant détaille une partie de l'arborescence présente dans la machine virtuelle :



Les dossier les plus importants pour Atom sont les suivants :

- **atom** ([/]/home/vagrant/atom) : Il s'agit du dossier dans lequel les fichiers relatifs à l'installation d'Atom se trouvent. Il y a notamment le dossier *css*, contenant les fichiers d'apparence initialement inclus dans Atom, et le dossier *plugins*, dans lequel sera cloné le repository GitHub de la version de la Faculté des Lettres.
- **vagrant** ([/]/vagrant) : Ce dossier représente le dossier synchronisé du côté de la machine virtuelle. Contrairement aux autres dossiers, ce dernier peut être accédé depuis n'importe quelle position dans la hiérarchie, mais en sortir ramène automatiquement au dossier *root*, comme le montrent les commandes suivantes :

```
[~/atom] (qa/2.x)
vagrant$ cd /vagrant/
[/vagrant]
vagrant$ cd ..
[/]
vagrant$
```

2.4 Clonage du Github repository

Le GitHub repository de la Faculté des Lettres donne accès à la dernière version de l'identité visuelle du site d'Atom. Il est disponible en ligne à l'adresse suivante, et cette section va détailler les étapes

à suivre pour le cloner dans la machine virtuelle. Cloner un repository offre de nombreux avantages dans le cadre d'un projet d'équipe : chaque membre peut facilement obtenir la dernière version du projet et modifier les fichiers dans la version commune, à la manière d'un Google Docs.

Ce dossier doit être cloné dans le dossier *plugins*. Pour y accéder depuis le dossier *root*, et cloner le repository, entrez les commandes suivantes :

```
vagrant$ cd home/vagrant/atom/plugins/
[~/atom/plugins] (qa/2.x)
vagrant$ git clone https://github.com/unil-lettres/arLettresPlugin
```

Si le clonage se déroule sans problème, vous devriez voir des messages similaires à ceux ci-dessous s'afficher dans le terminal :

```
Cloning into 'arLettresPlugin'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 24 (delta 6), reused 23 (delta 5), pack-reused 0
Receiving objects: 100% (24/24), done.
Resolving deltas: 100% (6/6), done.
[~/atom/plugins] (qa/2.x)
vagrant$
```

La commande suivante permet alors d'entrer dans le repository :

```
[~/atom/plugins] (qa/2.x)
vagrant$ cd arLettresPlugin/
[~/atom/plugins/arLettresPlugin] (development)
```

L'indication *(development)* dans la ligne de commande spécifie que nous sommes bel et bien dans un GitHub repository et que toutes les modifications pourront directement être transférées à tous les autres membres du projet. Le chapitre suivant offre une introduction aux commandes permettant d'ajouter des modifications aux fichiers présents dans le repository.

2.5 Modification de fichiers

La machine virtuelle ne permet pas d'ouvrir de fichiers à l'aide d'éditeurs graphiques tels que Sublime Text, et n'est ainsi pas adaptée pour apporter des modifications aux fichiers de configuration d'Atom. Ainsi, la meilleure solution est de les déplacer temporairement dans le dossier synchronisé, de les modifier depuis la machine hôte, et de les redéplacer ensuite dans la machine virtuelle.

A l'aide du terminal, utilisez les commandes suivantes depuis le dossier *root* pour rejoindre le dossier *plugins*, et d'ensuite déplacer le repository *arLettresPlugin* dans le dossier synchronisé :

```
[/]
vagrant$ cd home/vagrant/atom/plugins/
[~/atom/plugins] (qa/2.x)
vagrant$ mv arLettresPlugin/ /vagrant/
[~/atom/plugins] (qa/2.x)
vagrant$
```

A partir de ce moment, vous pouvez retrouver le dossier *arLettresPlugin* dans le même dossier que le fichier Vagrantfile de la machine hôte, à savoir *Downloads/atom-vagrant* dans l'exemple donné

auparavant. Après avoir apporté les modifications nécessaires, la commande suivante permet de migrer le dossier à nouveau dans la machine virtuelle :

```
[~/atom/plugins] (qa/2.x)
vagrant$ mv /vagrant/arLettresPlugin/ .
[~/atom/plugins] (qa/2.x)
vagrant$
```

Afin de pouvoir appliquer la nouvelle apparence visuelle, il est nécessaire de compiler le fichier .less de sorte à obtenir un fichier de format CSS. En considérant un fichier .less nommé *main.less*, la commande suivante permet de créer le fichier *min.css* nécessaire pour Atom, en positionnant le terminal dans le dossier où se trouve le ficheir *main.less* :

```
lessc --compress --relative-urls main.less > min.css
```

3 Guide d'utilisation de GitHub

GitHub est une plateforme permettant à tout utilisateur d'héberger du code source et de le partager de manière dynamique avec des collaborateurs. GitHub permet de créer des *repositories*, qui correspondent à un dossier contenant tous les fichiers inclus dans un projet, et que chaque membre de l'équipe peut ensuite cloner localement sur sa machine.

GitHub offre une palette de commandes qui permettent d'interagir entre le repository et la machine utilisateur, que ce soit pour cloner les fichiers, obtenir les dernières modifications apportées par d'autres membres de l'équipe, ou ajouter ses propres modifications. Le fonctionnement suit un cycle, qui est implémenté par les commandes suivantes.

git clone. Cette commande permet de créer une copie locale d'un repository. Cette copie restera directement liée au repository GitHub, et permettra ainsi à l'utilisateur de modifier les fichiers et de pouvoir ultérieurement apporter ces modifications à la version du repository sauvegardée sur GitHub.

La capture d'écran suivante montre un exemple de clonage d'un repository. La commande `git status` invoquée après s'être déplacé dans la copie locale permet de vérifier l'état du repository :

```
(base) aminmekacher@Amins-MacBook-Air Documents % git clone https://github.com/unil-lettres/arLettresPlugin
Cloning into 'arLettresPlugin'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 24 (delta 6), reused 23 (delta 5), pack-reused 0
Receiving objects: 100% (24/24), done.
Resolving deltas: 100% (6/6), done.
(base) aminmekacher@Amins-MacBook-Air Documents % cd arLettresPlugin
(base) aminmekacher@Amins-MacBook-Air arLettresPlugin % git status
On branch development
Your branch is up to date with 'origin/development'.

nothing to commit, working tree clean
(base) aminmekacher@Amins-MacBook-Air arLettresPlugin %
```

Figure 7: Exemple d'exécution de la commande `git clone`

git status. Après avoir cloné un repository, les fichiers qui y sont présents sont directement accessibles sur la machine locale et peuvent être modifiés comme des fichiers classiques. La commande `git status` permet, à tout moment, de savoir quels fichiers ont été modifiés ou supprimés depuis la dernière copie, et ainsi de garder une trace des changements qui doivent être appliqués au repository GitHub.

Après avoir modifié les fichiers `css/main.less` et `README.md`, la commande `git status` retourne le message suivant :

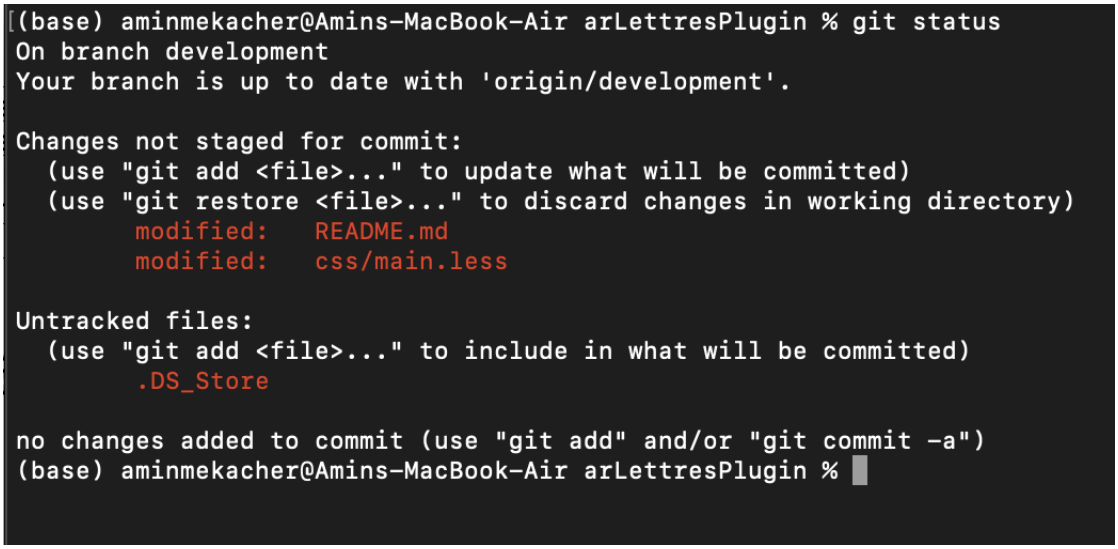


Figure 8: Exemple d'exécution de la commande `git status`

Tout fichier qui a été modifié mais qui n'a pas encore été ajouté pour la pro

git add. Cette commande permet d'ajouter des fichiers qui ont été modifiés localement pour la prochaine mise à jour du repository GitHub. Comme le montre le message affiché ci-dessus en réponse à la commande `git status`, un fichier qui n'est pas inclus dans la mise à jour, appelée *commit* dans la terminologie GitHub, est dit *unstaged*, tandis qu'il devient *staged* une fois qu'il a été ajouté via la commande `git add`, comme le montre la capture d'écran ci-dessous :

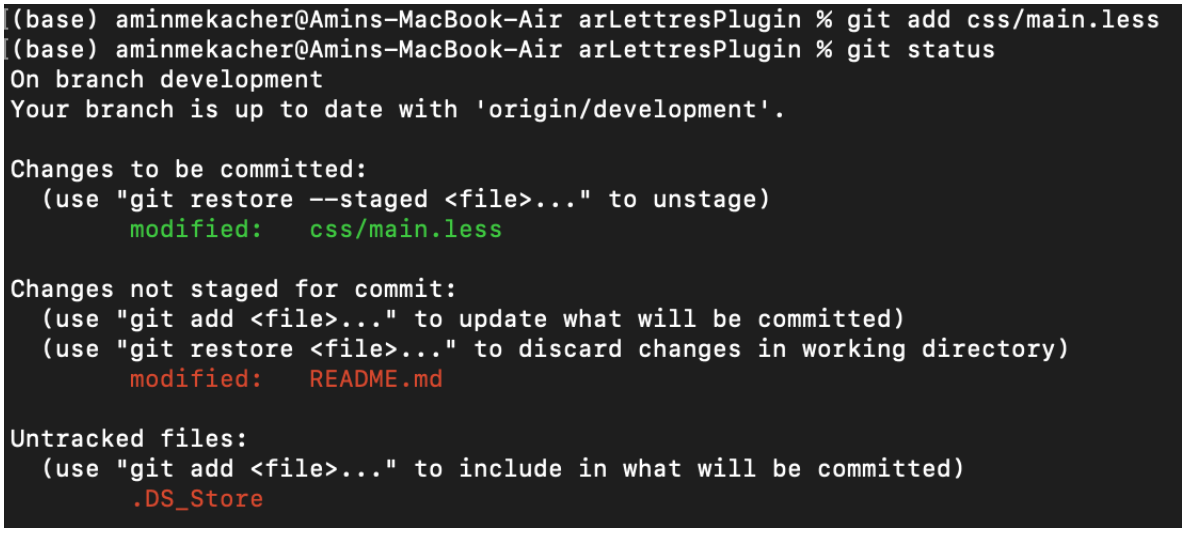


Figure 9: Exemple d'exécution de la commande `git add`

Dans le cas où un utilisateur souhaite ajouter tous les fichiers actuellement unstaged, un raccourci consiste à entrer la commande `git add .` , qui va automatiquement inclure tous les fichiers sans avoir à les énumérer.

git commit. Cette commande permet de créer un commit contenant tous les fichiers actuellement staged via la commande `git add`. La syntaxe employée pour cette commande est généralement `git commit -m "Message"`, où le message (qui doit être indiqué entre guillemets) permet de donner des détails sur les modifications qui ont été apportées par ce commit. La capture d'écran suivante montre un exemple de commit :

```
(base) aminmekacher@Amins-MacBook-Air arLettresPlugin % git commit -m "Stage test"
[development 0a02e1e] Stage test
 1 file changed, 1 insertion(+), 1 deletion(-)
(base) aminmekacher@Amins-MacBook-Air arLettresPlugin % git status
On branch development
Your branch is ahead of 'origin/development' by 1 commit.
  (use "git push" to publish your local commits)

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:   README.md

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

no changes added to commit (use "git add" and/or "git commit -a")
(base) aminmekacher@Amins-MacBook-Air arLettresPlugin %
```

Figure 10: Exemple d'exécution de la commande `git commit`

Une fois qu'un commit a été crée, la commande `git status` indique que la branche locale est en avance sur le repository GitHub et quels sont les fichiers qui n'ont pas été inclus dans le commit (le fichier `README.md` dans l'exemple ci-dessus).

git push. Une fois qu'un commit a été crée localement, la commande `git push` permet de l'envoyer sur le repository GitHub, afin que tous les autres utilisateurs du repository puissent avoir accès aux dernières modifications.

git log. La commande `git log` permet d'avoir un aperçu des derniers commits qui ont été uploadés sur le Github repository. Invoquer cette commande juste après avoir employé `git push` permet de s'assurer que le nouveau commit apparaît bel et bien dans le log.

Dans le cas du repository *arLettresPlugin*, on obtient actuellement la sortie suivante :

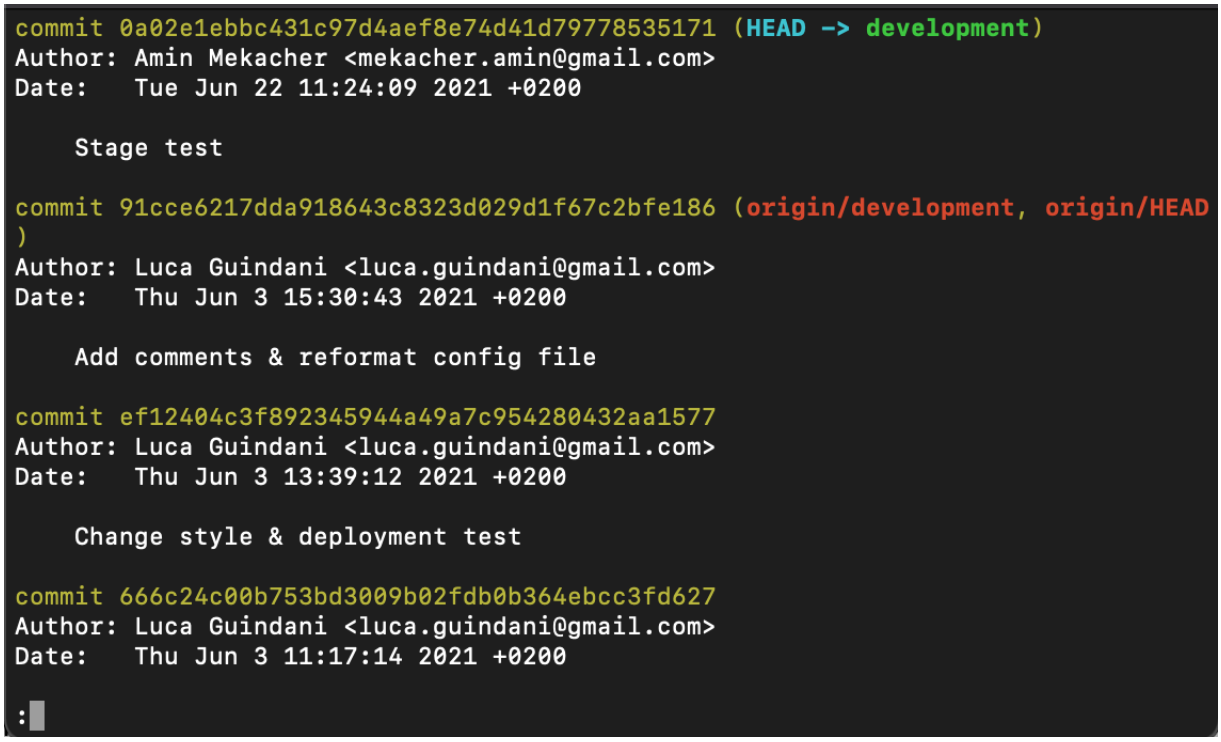


Figure 11: Exemple d'exécution de la commande `git log`

git pull. Cette commande permet à un utilisateur de mettre à jour la version du repository présente localement dans sa machine, en envoyant une requête pour recevoir les dernières modifications qui ont été ajoutées sur le repository GitHub par d'autres utilisateurs. Si la version présente localement est d'ores et déjà à jour, aucune modification ne sera apportée.

Le schéma ci-dessous synthétise les étapes décrites dans les paragraphes précédents et résume le cycle d'actions permettant d'employer de manière dynamique les opportunités de collaboration offertes par GitHub:

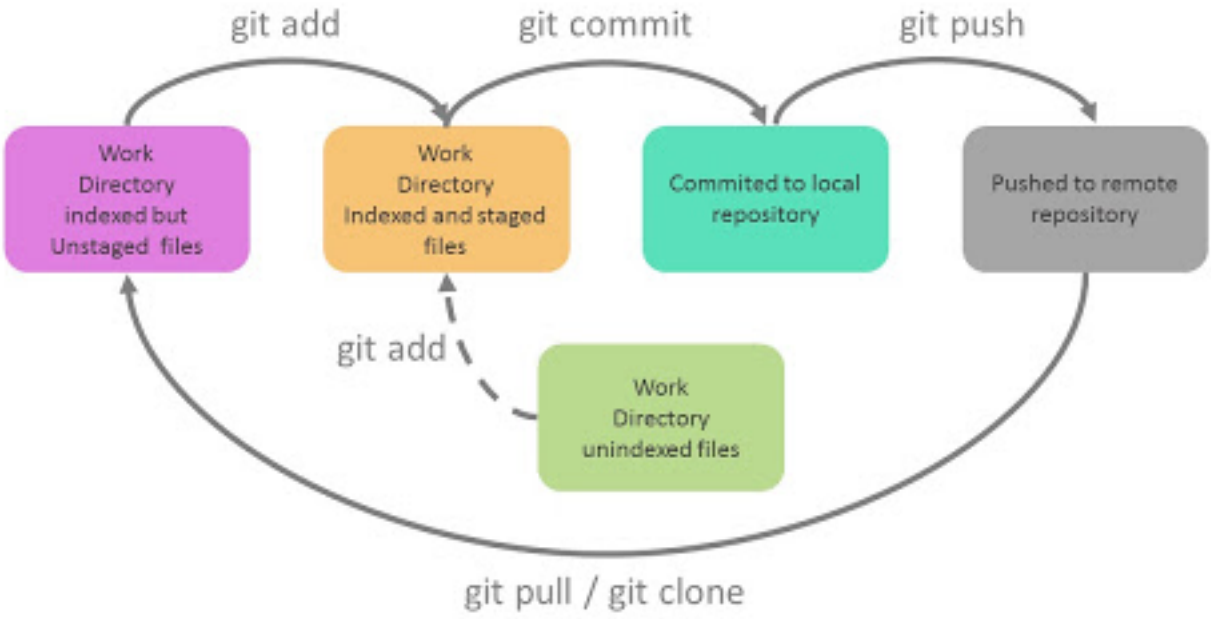


Figure 12: Contenu du dossier *Home*, indiqué par la commande `ls`

4 Modification de l'apparence d'Atom

4.1 Logo

Le logo du site se trouve dans la machine virtuelle, à l'emplacement *atom/images/logo.png*. Pour le remplacer, il suffit de modifier cette image avec une autre, en prenant soin de renommer le nouveau fichier *logo.png* et de respecter les dimensions maximales, qui sont de 50x50 px.