

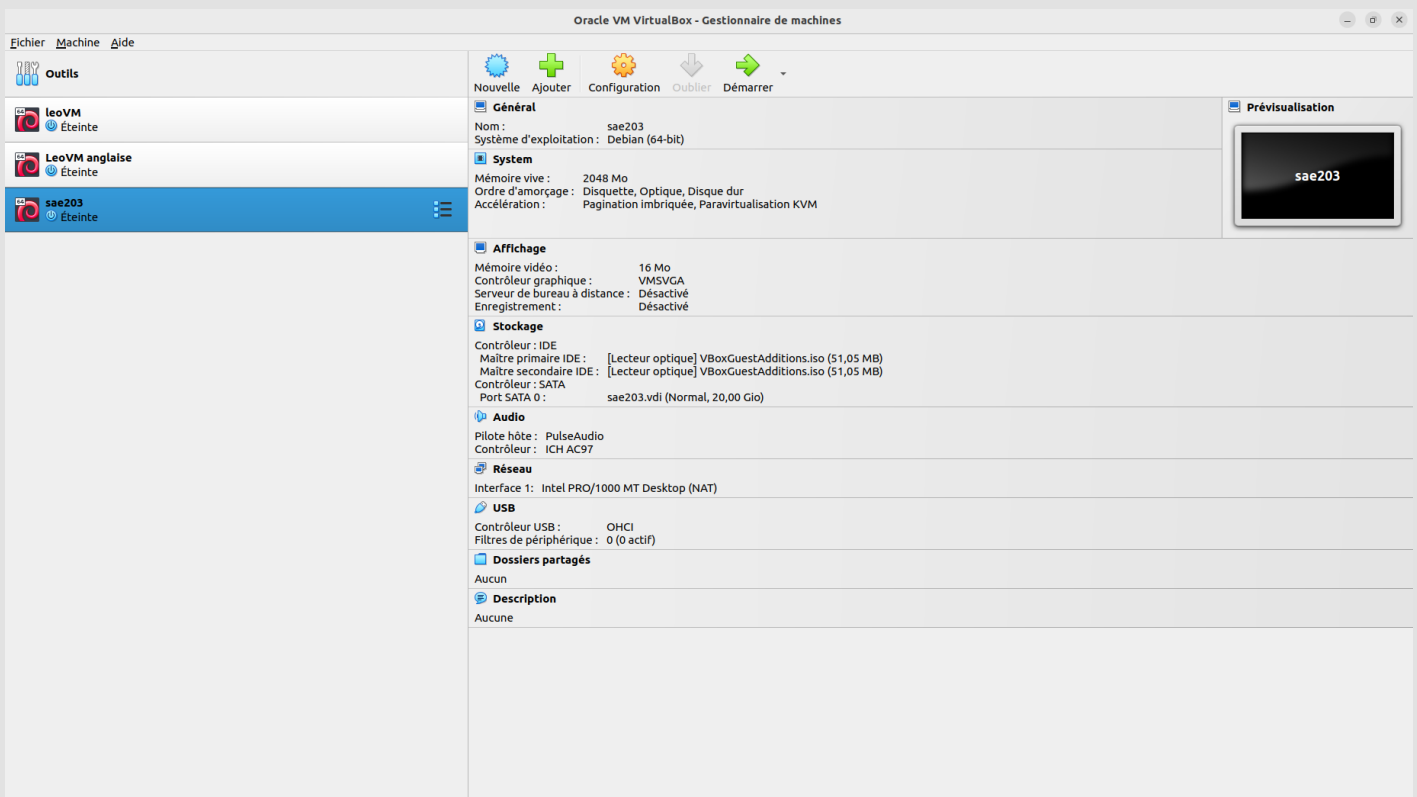
SAÉ 2.03 — Rapport

Table of Contents

1. Installation d'une machine virtuelle automatique	2
2. Apprentissage balisage de base	9
3. Utilisation de Git	11
4. Installation de Gitea	16
5. Balisage léger avancé	23
Index	25

Présentation du projet

*Savoir installer correctement une machine (virtuelle ou non) ou un réseau peut s'avérer **nécessaire** lorsque l'on travaille dans le domaine de l'informatique. C'est pourquoi nous avons appris à installer convenablement une **machine virtuelle**, mais nous avons également eu l'occasion d'apprendre à mettre en place un **service réseau** interne lors de plusieurs semaines de travail.*



1. Installation d'une machine virtuelle automatique

Création de la machine

Pour commencer, histoire de se remettre dans le bain, nous avons créé une machine virtuelle nommée « sae203 » de type « Linux », possédant ainsi une version « Debian 64-bit », une mémoire vive (RAM) de 2048 Mo ainsi qu'un disque dur de 20Go.

Lors de la création de la machine, nous avons eu l'occasion de nous poser certaines questions, auxquelles nous avons trouvé réponse :

- **Que signifie "64-bit" dans "Debian 64-bit" ?**

- « 64-bit » dans « Debian 64-bit » signifie que la machine sera installée avec un processeur de 64 bits. Ainsi, un processeur 64 bits est un microprocesseur dans lequel la taille d'un mot machine est de 64 bits (la machine peut ainsi traiter des données par paquets de 64 bits à la fois, ce qui permet une mémoire plus grande.). Une condition indispensable pour les applications fortement consommatrices de données et de mémoire (Comme les systèmes de gestion de bases de données et les serveurs hauts performances).

- **Quelle est la configuration réseau utilisé par défaut ?**

- La configuration réseau utilisé par défaut est la configuration réseau « NAT ». Ce mode d'accès au réseau permet aux machines virtuelles de communiquer à l'extérieur du réseau virtuel, à travers l'interface réseau de votre ordinateur physique hôte. Il s'agit là d'un autre Switch virtuel qui assure la connexion des machines virtuelles car l'adresse réseau IP est différente.

- **Quel est le nom du fichier XML contenant la configuration de votre machine ?**

- Le nom du fichier XML contenant la configuration de la machine est « sae203.vbox-prev ». L'extension « .vbox-prev » ici signifie que le fichier est une sauvegarde de la configuration de la machine virtuelle. Chaque modification de la machine crée une copie portant cette extension, il suffit ainsi de remplacer « .vbox-prev » par « .vbox » pour utiliser cette sauvegarde en cas d'erreur sur la machine principale.

- **Sauriez-vous le modifier directement ce fichier de configuration pour mettre 2 processeurs à votre machine ?**

- Afin de mettre 2 processeurs à la machine, il faut se rendre dans le fichier XML dans la balise nommée « CPU » et y ajouter la ligne `<topology sockets="2" cores="2" threads="1"/>`. Ensuite, il faut redémarrer VirtualBox et la machine virtuelle, puis cela devraient fonctionner. Il est également possible de réaliser tout cela dans l'onglet « Système » des paramètres de la machine, puis cliquer sur « processeur » et enfin déterminer le nombre de processeurs souhaités.

Jusque-là, nous n'avons rencontré aucune difficulté concernant la mise en place de la machine avant installation. Il s'agit là d'une étape déjà exercée dans le passé lors d'un précédent projet.

iso bootable

Nous avons ensuite attribué à la machine un **iso bootable** de **Debian 12** afin d'installer Debian sur la machine virtuelle. L'installation pouvait ainsi démarrer.

Nous avons décidé d'installer une machine nommée « **serveur** », **sans domaine**, avec une **langue** réglée sur « **français** » comportant le **miroir** français « <http://debian.polytech-lille.fr> », **sans proxy**, avec un **compte administrateur** nommé « **root** » (comportant un mot de passe « **root** ») et un **utilisateur** nommé « **User** » avec comme **mot de passe** « **user** ». pour finir, notre **disque** comporte une **seule partition**, et nous avons également ajouté quelque **logiciels de démarrage** utiles tels que l'environnement de bureau **Debian**, **MATE** (sans **Gnome**), un **serveur web** et **ssh** ainsi que **l'utilitaire usuels du système**.

Suite à cela, plusieurs interrogations nous sont venues en tête :

1. Qu'est-ce qu'un fichier iso bootable ?

- a. Un fichier **iso bootable** est un fichier image créé à partir d'un **CD** ou d'un **DVD**. Ainsi, une **clé USB** peut être transformée afin d'imiter le comportement d'un **CD** ou d'un **DVD**. Ainsi, en y instaurant uniquement [l'https://fr.wikipedia.org/wiki/Image_disque\[iso\]](https://fr.wikipedia.org/wiki/Image_disque[iso]) du système d'exploitation souhaité, il est possible d'installer ce dernier sur la machine. (Source : <https://www.lesnumeriques.com/telecharger/iso-to-usb-20012>).

2. Qu'est-ce que MATE ? GNOME ?

- a. **MATE** est un **Fork** de **Gnome** (c'est-à-dire un logiciel créé à partir du **code source** de **Gnome**). Il s'agit là d'un **environnement de bureau libre** utilisant une **boîte à outils** « **GTK+ 3.x** » et destiné aux **systèmes d'exploitation** apparentés à **UNIX**. (source : <https://fr.wikipedia.org/wiki/MATE>).
- b. **Gnome** quant à lui est également un **environnement de bureau libre** dont l'objectif est de rendre accessible l'utilisation du **système d'exploitation** **GNU** au plus grand nombre. (Source : <https://fr.wikipedia.org/wiki/GNOME>).

3. Qu'est-ce qu'un serveur web ?

- a. Un **serveur web** peut soit être un **logiciel de service de ressources web** (**serveur HTTP**), soit un **serveur informatique** (ordinateur) qui répond à des **requêtes du World Wide Web** sur un **réseau public** (**Internet**) ou privé (**intranet**), en utilisant principalement le **protocole HTTP**. (Source : https://fr.wikipedia.org/wiki/Serveur_web).

4. Qu'est-ce qu'un serveur ssh ?

- a. **ssh** est un « **shell** » (Un « **terminal** », permettant la **gestion des serveurs Linux** via la possibilité de **dialoguer** avec une **machine** ou un **serveur** via l'exécution de différentes **commandes** qui retourneront des informations) ou **protocole réseau** qui permet aux administrateurs d'accéder à distance à un ordinateur, en toute sécurité. (Source : <https://www.it-connect.fr/chapitres/quest-ce-que-ssh/>) (Source 2 : <https://www.cloudflare.com/fr-fr/learning/access-management/what-is-ssh/>).

5. Qu'est-ce qu'un serveur mandataire ?

- a. Un **serveur mandataire** (souvent appelé aussi « **proxy** ») est un **serveur informatique** qui a pour fonction de relayer des **requêtes** entre un **poste client** et un **serveur**. Les **serveurs mandataires** sont notamment utilisés pour assurer les fonctions suivantes :
 - i. Mémoire cache ;

- ii. La journalisation des requêtes (" logging ") ;
- iii. La sécurité du réseau local ;
- iv. Le filtrage et l'anonymat.

Attribution droits sudo

Nous avons par la suite attribuer le groupe « sudo » à l'utilisateur « user » via la commande suivante :

```
sudo adduser user sudo
```

NOTE

Pour savoir à quels groupes appartient l'utilisateur « user », il faut écrire la commande `groups user` dans le terminal de commande.

De même que pour les étapes précédentes, aucune difficulté n'a été rencontrée, car nous connaissions déjà le processus grâce à l'un de nos projets passés.

Installation des suppléments invités

Ensuite, il a fallu installer des **suppléments invités**. Pour cela, nous pouvons cliquer sur « Périphériques » en haut de l'écran de la machine virtuelle lancée, puis « Insérer l'image CD des Additions invité... », puis monter le CD à l'aide de la commande `sudo mount /dev/cdrom/mnt puis sudo /mnt/VboxLinuxAdditions.run` (ou faire un `cd /media/cdrom0 puis sudo sh VboxLinuxAdditions.run`). Cela permet par exemple d'avoir une page dimensionnée aux bonnes dimensions.

À partir de cette étape, il s'agit d'une situation **nouvelle** à laquelle nous n'avons jamais dû faire face. Nous avons ainsi dû nous renseigner (notamment sur Internet) sur ce qu'étaient les suppléments invités ainsi que la manière de les installer et leurs spécificités, suite à quoi nous avons pu nous poser les questions suivantes :

- **Quel est la version du noyau Linux utilisé par votre VM ?**
 - Afin de connaître la **version** du **noyau Linux** utilisé par notre machine, nous pouvons entrer la commande `sudo uname -r`. Dans notre cas, cela nous renvoie « 6.1.0-17-amd64 ». Cela signifie que notre **noyau Linux** utilise la version « 6.1.0-17 » d'[https://fr.wikipedia.org/wiki/AMD64\[adm64\]](https://fr.wikipedia.org/wiki/AMD64[adm64]) (**ADM64** étant le nom de l'architecture des premiers **microprocesseurs 64 bits** de la société « Advanced Micro Devices »). (Source : <https://fr.wikipedia.org/wiki/AMD64>).
- **À quoi servent les suppléments invités ?**
 - Les **additions invités** sont une **collection** de **pilotes** de **périphériques** et d'**applications système** pour **VirtualBox** qui améliorent les performances du système d'exploitation invité et permettent une meilleure interaction entre la machine hôte et la machine invitée. Cela permet par exemple d'y intégrer le **pointeur de la souris**, les **dossiers partagés**, de **meilleures performances**

graphiques, la **synchronisation de l'heure*** entre l'hôte et l'invité et bien plus encore. (Source : <https://lecrabeinfo.net/virtualbox-installer-les-additions-invite-guest-additions.html>).

- **À quoi sert la commande `mount` (dans notre cas de figure et dans le cas général) ?**

- Dans le cas général, la commande `mount` permet de demander au **système d'exploitation** de rendre un **système de fichiers accessible**, à un emplacement spécifié. Ainsi, elle nous permet de monter un **système de fichiers** indiqué comme répertoire à l'aide du paramètre « `Noeud:Répertoire` », sur le répertoire spécifié par le paramètre `Répertoire`. Dans notre cas, cela nous a permis de rendre le fichier « `cdrom` » accessible afin de pouvoir exécuter le fichier « `VBoxLinuxAdditions.run` ».

- **Qu'est-ce que le Projet Debian ? D'où vient le nom Debian ?**

- Le projet **Debian** est un groupe de personnes **volontaires** d'envergure mondiale qui s'efforcent de produire un **système d'exploitation** qui soit composé exclusivement de **logiciels libres**. Ce projet a pour principal produit la distribution **Debian GNU/Linux**, qui inclut le **noyau Linux** ainsi que des milliers d'applications pré empaquetées. Le nom vient des prénoms du créateur de Debian (*Ian Murdock*) et son épouse (*Debra*). (<https://www.debian.org/doc/manuals/project-history/intro.fr.html>).

- **Il existe 3 durées de prise en charge (support) de ces versions : la durée minimale, la durée en support long terme (LTS) et la durée en support long terme étendue (ELTS). Quelle sont les durées de ces prises en charge ?**

- La durée minimale des prises en charge des versions de **Debian** est :
 - Minimale : 2 ans.
 - Durée en support long terme (LTS) : minimum 5 ans. (pour une prise en charge à long terme)
 - Durée en support long terme étendue (ELTS) : minimum 1 an. (<https://www.debian-fr.org/t/debian-lts-quelle-duree-de-prise-en-charge-a-long-terme/80566/3>).

- **Pendant combien de temps les mises à jour de sécurité seront-elles fournies ?**

- Tous les deux mois, les responsables de la version **stable** vérifient la liste des **paquets** de « `proposed-updates` » et décident si un paquet est approprié ou non pour stable. Ces modifications sont **rassemblées** dans une **nouvelle version (révision)** de « `stable` » (par exemple « `2.2r3` » ou « `2.2r4` »). (<https://www.debian.org/security/faq.fr.html>).

- **Combien de version au minimum sont activement maintenues par Debian ?**

- Debian a toujours au moins **trois versions** activement entretenues : la version « `stable` » qui contient la **dernière distribution** officiellement sortie de **Debian**, la version « `testing` » qui contient les **paquets** qui n'ont pas encore été acceptés dans la distribution « `stable` » mais qui sont en attente de l'être, et la version « `unstable` » dont les **activités de développement** se déroulent. Généralement, cette distribution est utilisée par les développeurs et par ceux qui aiment **vivre sur le fil**. (<https://www.debian.org/releases/index.fr.html>).

- **Chaque distribution majeur possède un nom de code différent. Par exemple, la version majeur actuelle (Debian 12) se nomme bookworm. D'où viennent les noms de code données aux distributions ?**

- Quand une distribution **Debian** est en cours de développement, elle n'a **aucun numéro de version**, mais un **nom de code**. Le but de ces noms de code est de faciliter la **copie sur les miroirs** des distributions **Debian** (si un véritable répertoire comme « `unstable` » est soudainement renommé en « `stable` », beaucoup de choses devraient être inutilement téléchargées). Jusqu'ici, les noms

de code proviennent des personnages des films « Toy Story » par Pixar (ex : buzz, rex, bo, hamm, jessie, etc...). (<https://www.debian.org/doc/manuals/debian-faq/ftparchives.fr.html>).

- **L'un des atouts de Debian fut le nombre d'architecture (≈ processeurs) officiellement prises en charge. Combien et lesquelles sont prises en charge par la version Bullseye ?**

- Il y a **9 architectures** gérées par la version « Bullseye » :
 - PC 64 bits (amd64)
 - ARM 64 bits (AArch64)
 - EABI ARM (armel)
 - ARM avec unité de calcul flottant (armhf)
 - PC 32 bits (i386)
 - MIPS (petit-boutiste)
 - MIPS 64 bits (petit-boutiste)
 - PowerPC 64 bits (petit-boutiste)
 - System z (<https://www.debian.org/releases/bullseye/index.fr.html>).

- **Quelle a était le premier nom de code utilisé ? Quand a-t-il été annoncé ? Quelle était le numéro de version de cette distribution ?**

- Le **premier nom de code** utilisé par Debian est « Buzz » (il s'agit là de la version Debian 1.1). (<https://www.debian.org/distrib/archive>).
- Cette **première version** fut annoncée et publiée le **17 juin 1996** sous le nom de Debian GNU/Linux **1.1**. (<https://wiki.debian.org/fr/DebianBuzz>).
- Comme cité précédemment, le **numéro de version** de la distribution « Buzz » est « Debian 1.1 ». (<https://wiki.debian.org/fr/DebianBuzz>).

- **Quel est le dernier nom de code annoncée à ce jour ? Quand a-t-il été annoncé ? Quelle est la version de cette distribution ?**

- Actuellement, le **dernier nom de code** annoncé à ce jour est #« Bookworm »". (<https://www.debian.org/releases/index.fr.html>).
- La **version initiale** fut publiée le **10 juin 2023**, mais la **dernière en date** fut publiée le **10 février 2024**. (<https://www.debian.org/releases/bookworm/>).
- La **version** de cette distribution est « Debian 12.0 » (pour la première), et « Debian 12.4 » (pour la dernière en date). (<https://www.debian.org/releases/bookworm/>).

GNU GRUB version 2.06-13+deb12u1

```
*Debian GNU/Linux
Advanced options for Debian GNU/Linux
```

Utilisez les touches ↑ et ↓ pour sélectionner une entrée.
Appuyez sur Entrée pour démarrer le système sélectionné, « e »
pour éditer les commandes avant de démarrer ou « c » pour
obtenir une invite de commandes.
L'entrée sélectionnée sera exécutée automatiquement dans 1 s.

© debian 12

Concernant cette étape, la seule problématique rencontrée est le processus d'installation qui peut s'avérer très long et répétitif. C'est pourquoi dans la prochaine étape, nous verrons comment nous avons réussi à installer de manière automatique notre machine virtuelle à l'aide d'une pré-configuration.

Pré-configuration de la machine virtuelle

Pour finir, pour utiliser une pré-configuration afin de configurer automatiquement notre machine virtuelle, nous avons entré la commande suivante dans le terminal à l'emplacement exact de notre machine virtuelle :

```
sed -i -E 's/(--iprt-iso-maker-file-marker-bourne-sh).*$/\1=$(cat  
/proc/sys/kernel/random/uuid)/ S203-Debian12.viso'[1]
```

Nous avons ensuite **inséré le fichier « S203_Debian12.viso »** dans le lecteur optique (*cd/dvd*) de notre machine virtuelle, puis nous avons **démarré la machine virtuelle** et **laissé faire l'installation** se dérouler jusqu'au reboot. Pour finir, nous avons changé la pré-configuration en **ajoutant les paquets MATE, sudo, git, sqlite3, curl, bash-completion et neofetch** ainsi que les **droits sudo** à l'utilisateur **user**.

NOTE

Pour installer les **paquets** `MATE`, `sudo`, `git`, `sqlite3`, `curl`, `bash-completion` et `neofetch`, il faut entrer la commande `d-i pkgsel/include string mate-desktop-environment-core sudo git sqlite3 curl bash-completion neofetch` dans la catégorie « Packages, Mirrors, Image » du **fichier de préconfiguration** « `preseed.cfg` ». Pour **attribuer les droits sudo** à l'utilisateur **user**, il suffit d'ajouter la ligne de commande `d-i usermod -aG sudo user` dans la catégorie « Ajout des comptes root et user ».



Charger des composants depuis le support d'installation

Chargement de composants supplémentaires

Récupération de partman-target

Cette installation nous a posé beaucoup de problèmes dû au **temps d'installation** extrêmement **long** (parfois plus d'une **dizaine*** de minutes) ainsi que les difficultés rencontrées lors de la mise en place de l'installation automatique des paquets `sudo`, `git`, `sqlite3`, `curl`, `bash-completion`, `neofetch` et surtout **MATE**. Pour remédier à cela, nous avons **rechercher** sur de nombreux sites la solution à ces problèmes, suite à quoi nous avons réussi à les résoudre à l'aide des étapes citées précédemment.

2. Apprentissage balisage de base

Recherches

Dans un second temps, nous avons fait des recherches poussées de nous former à l'écriture d'un rapport grâce à Markdown / Pandoc ou AsciiDoctor (systèmes de balisages permettant la simplification d'écriture d'un rapport). Ainsi, voici les différentes balises possibles :

Balises	Markdown	AsciiDoctor
Italique	<code>_texte en italique_</code>	<code>__texte en italique__</code>
Gras	<code>**texte en gras**</code>	<code>**texte en gras**</code>
Texte barré	<code>~~texte barré~~</code>	
Titre	<code># Titre (plus il y'a de "#", plus le titre est petit)</code>	<code>= Titre (plus il y'a de "=", plus le titre est petit)</code>
Paragraphe	<code>saut de ligne</code>	<code>saut de ligne</code>
Citation	<code>>Citation</code>	
Commentaire	<code><!-- Commentaire -></code>	<code>// Commentaire</code>
Liste simple	<code>- Liste</code>	<code>* Liste</code>
Sous liste simple		<code>** Liste</code>
Liste numérotée	<code>1. Liste</code>	<code>1. Liste</code>
Liste à cocher	<code>[] A ou [X] B</code>	<code>[] A ou [x] B</code>
Code	<code>`code`</code>	<code>`code`</code>
Hyperliens	<code>[Lien](https://example.com/ "titre de lien optionnel")</code>	<code>Lien[(((titre de lien optionnel)))]</code>
Image	<code>![Ceci est un exemple d'image](https://example.com/bild.jpg)</code>	<code>image::image.png[Titre]</code>
Tableau	<code> cellule 1 cellule 2 </code>	<code> === cellule1 cellule2 ===</code>
Commentaire	<code><!-- Commentaire -></code>	<code>// Commentaire</code>
Note de bas de page	<code>[^1]: Vous trouverez ici le texte de la note de bas de page</code>	
Annuler utilisation d'une balise (exemple avec la balise <code>gras</code>)	<code>*exemple avec des astérisques*</code>	<code>*exemple avec des astérisques*</code>

Installations

Afin d'installer Pandoc, il faut écrire les commandes suivante dans le terminal

de commande :

```
sudo apt update  
sudo apt install pandoc
```

Afin d'installer **Asciidoctor**, il faut écrire les commandes suivante dans le terminal de commande :

```
sudo apt update  
sudo apt install ruby-full  
sudo gem install asciidoctor
```

Afin d'installer **Asciidoctor-pdf**, il faut écrire les commandes suivante dans le terminal de commande :

```
sudo apt update  
sudo apt install ruby-full  
sudo gem install asciidoctor-pdf
```

Utilisations

Pour transformer un format « Markdown » en « html », il faut entrer la commande suivant dans le terminal de commande:

```
pandoc -o nom_fichier.html nom_fichier
```

Pour **transformer** un format « adoc » en « html », il faut entrer la commande suivant dans le terminal de commande:

```
asciidoctor nom_fichier.adoc
```

Pour **transformer** un format « adoc » en « pdf », il faut entrer la commande suivant dans le terminal de commande:

```
asciidoctor-pdf nom_fichier.adoc
```

NOTE

Afin de rédiger un rapport en **Markdown** ou **AsciiDoctor**, il est possible d'utiliser n'importe quel éditeur de texte.

IMPORTANT

Ne pas oublier d'ajouter l'extension **".adoc"** au fichier contenant le code

L'apprentissage du **balisage léger** n'est pas une étape compliquée en soi. Le plus compliqué est surtout la **manière** de l'utiliser. Un grand nombre de nos difficultés fut dans la **mise en place** de notre rapport à l'aide des différents types de balisage disponible par **AsciiDoctor**. Cependant, une fois la prise en main effectuée (*bien que cela nous est pris quelques heures*), il est vrai que l'utilisation nous est déformée plus **simple**, nous permettant ainsi de rédiger ce rapport de manière simple et naturelle.

3. Utilisation de Git

Présentation de Git

Parmi les nombreux outils disponibles permettant de travailler de manière efficace en équipe (notamment dans le domaine du développement informatique dans notre cas, mais pas seulement), nous retrouvons notamment le logiciel Git.

Git est un **logiciel de gestion de versions décentralisé**. Il s'agit là d'un logiciel **libre** et **gratuit** créé en 2005 par **Linus Torvalds**, auteur du **noyau Linux**.

Voici les nombreuses options disponibles avec **Git**:

Commande	Description
<code>git init</code>	crée un nouveau dépôt
<code>git clone</code>	clone un dépôt
<code>git add</code>	ajoute de nouveaux objets blobs dans la base des objets pour chaque fichier modifié depuis le dernier commit. Les objets précédents restent inchangés
<code>git commit</code>	intègre la somme de contrôle SHA-1 d'un objet <i>tree</i> et les sommes de contrôle des objets <i>commits</i> parents pour créer un nouveau objet <i>commit</i>
<code>git branch</code>	pour la gestion des branches
<code>git merge</code>	fusionne une branche dans une autre
<code>git rebase</code>	déplace les commits de la branche courante devant les nouveaux commits d'une autre branche
<code>git log</code>	affiche la liste des commits effectués sur une branche
<code>git push</code>	publie les nouvelles révisions sur le <i>remote</i> (La commande prend différents paramètres)

<code>git pull</code>	récupère les dernières modifications distantes du projet (depuis le <i>Remote</i>) et les fusionne dans la branche courante
<code>git stash</code>	stocke de côté un état non commité afin d'effectuer d'autres tâches
<code>git checkout</code>	annule les modifications effectuées, déplacement sur une autre référence (branche, hash)
<code>git switch</code>	changement de branche
<code>git revert</code>	défait les modifications d'un commit précédent
<code>git remote</code>	gestion des <i>remotes</i>

Configuration de Git

Ainsi, après avoir installer **Git** à l'aide de la commande `sudo apt install git`, nous l'avons configuré (afin de pouvoir l'utiliser convenablement) à l'aide des commandes suivantes:

```
git config --global user.name "Prénom Nom"
```

NOTE

Cette commande définit dans les options globales le nom de l'utilisateur. **"Prénom Nom"** est à remplacer par le nom d'utilisateur souhaité.

```
git config --global user.mail "adresse mail"
```

NOTE

Cette commande définit dans les options globales l'adresse mail de l'utilisateur. **"adresse mail"** est à remplacer par l'adresse mail souhaité.

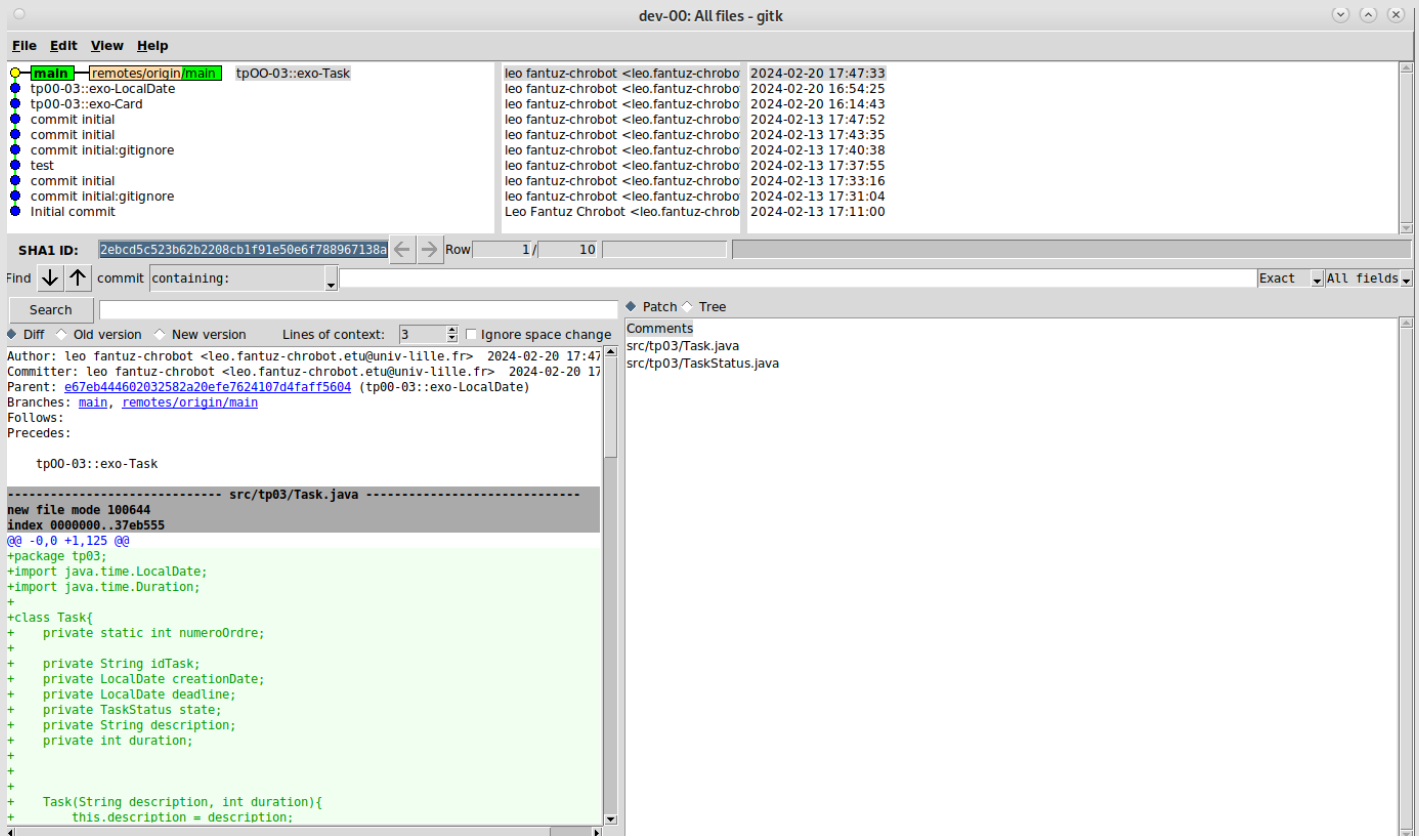
```
git config --global init.defaultBranch « master »
```

NOTE

Cette commande définit dans les options globales permet d'éviter le warning concernant la création d'une branche par défaut.

Installation de Gitk

Par la suite, nous avons installé **Gitk** grâce à la commande `sudo apt install gitk`.



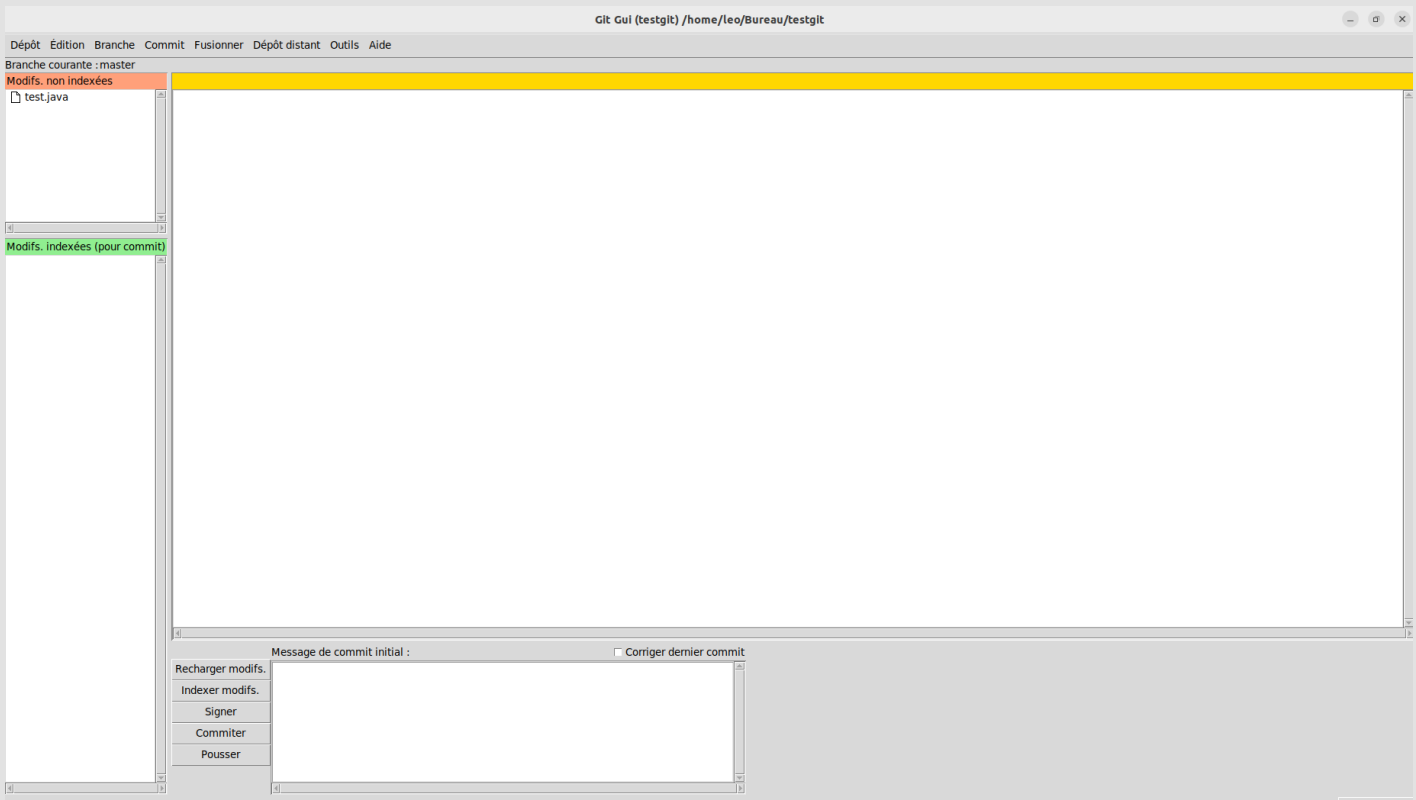
NOTE

Gitk est un **navigateur de dépôt graphique**, le **premier** de son genre. Il peut être considéré comme un **encapsuleur graphique** pour git log. Il permet notamment **d'explorer** et de **visualiser** l'historique d'un dépôt. Pour le lancer, il suffit de noter la commande suivante dans le terminal de commande, à l'emplacement du répertoire Git:

```
gitk.
```

Installation de Git-Gui

Puis, nous avons également installé **git-gui** à l'aide de la commande `sudo apt install git-gui`.

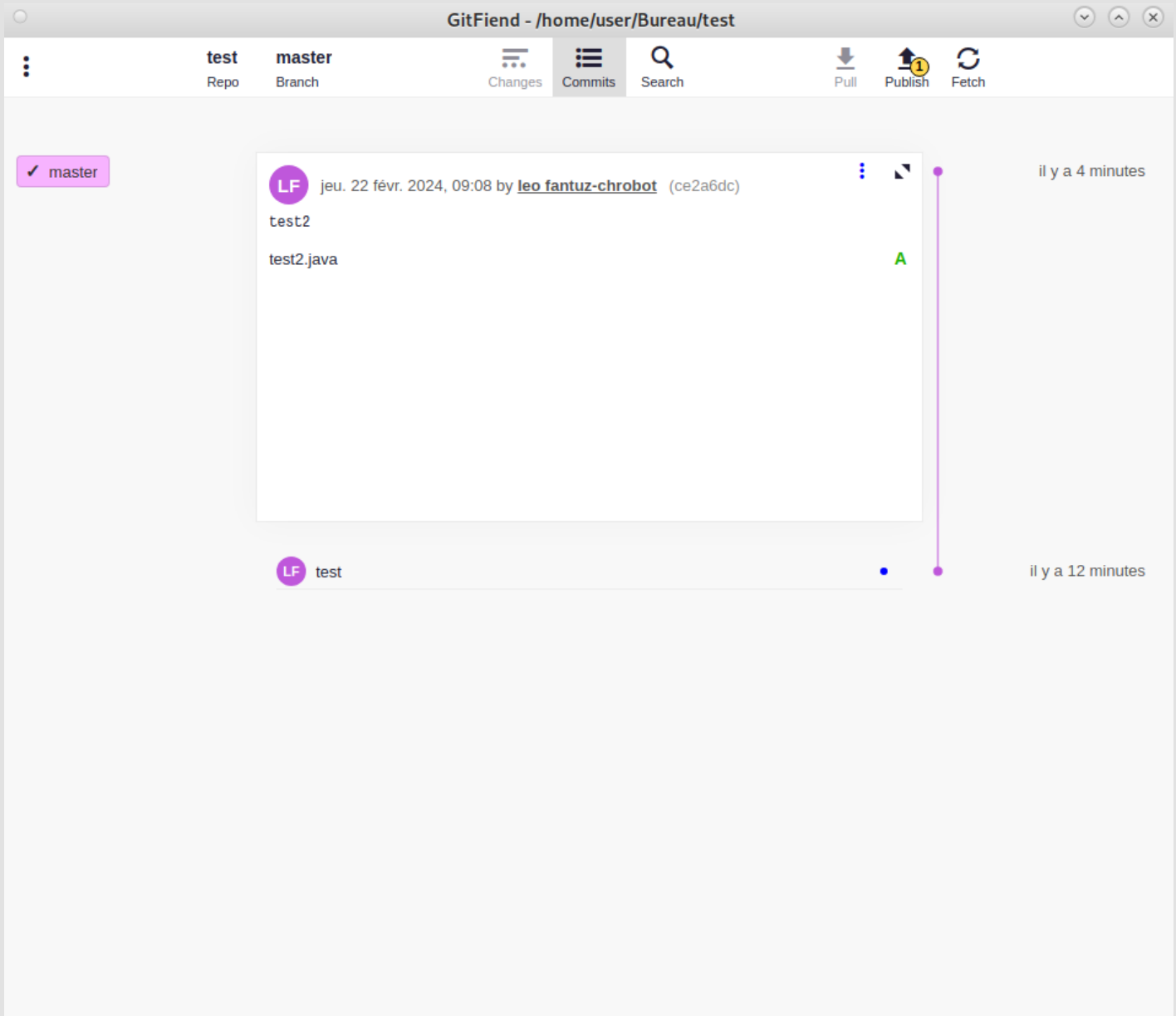


NOTE

git-gui est un des outils de bases fourni avec **Git** lors de son installation. **Git Gui** est une interface graphique permet de voir la différence des modifications en cours dans notre espace de travail ou encore de faire des commits et des pushes (et bien plus encore). En terme de fonctionnalité, **Git Gui** est très similaire à **Gitk**. Pour le lancer, il suffit de noter la commande suivante dans le terminal de commande, à l'emplacement du répertoire Git: `git gui` ^[2].

Installation de GitFiend

*Finally, we have installed a graphical interface named **GitFiend**. It is a graphical interface **free** and **free** that we found via the site git-scm.com, allowing the management of a Git project.*



Ce logiciel est **facilement utilisable** par son **interface** bien **organisée** et très **intuitive**. De plus, la possibilité de **créer** et **ouvrir** un **dépôt** directement via l'interface permet un usage plus **simple** de **Git**.

Grâce au site cité précédemment, nous avons pu **installer** un fichier **".deb"** (**deb** étant le format de fichier des **paquets logiciels** de la distribution **Debian GNU/Linux**) permettant l'installation via la ligne de commande suivante :

SU -

```
sudo apt-get install /home/user/Bureau/GitFiend_0.44.3_amd64.deb
```

NOTE

Nous avons **déplacé** le fichier dans le **Bureau** de l'utilisateur **user**. Le **".deb"** représente la **dernière version** de **GitFiend**. Il est aussi bon de spécifier que le fichier **".deb"** peut également être **téléchargé** via le site officiel de **GitFiend**.

Si l'on compare **GitFiend** avec **Gitk** et **git-gui**, nous pouvons remarquer que toutes ces interfaces permettent de **créer** de nouveaux fichiers et de **faire des commits, push et pull**. Cependant,

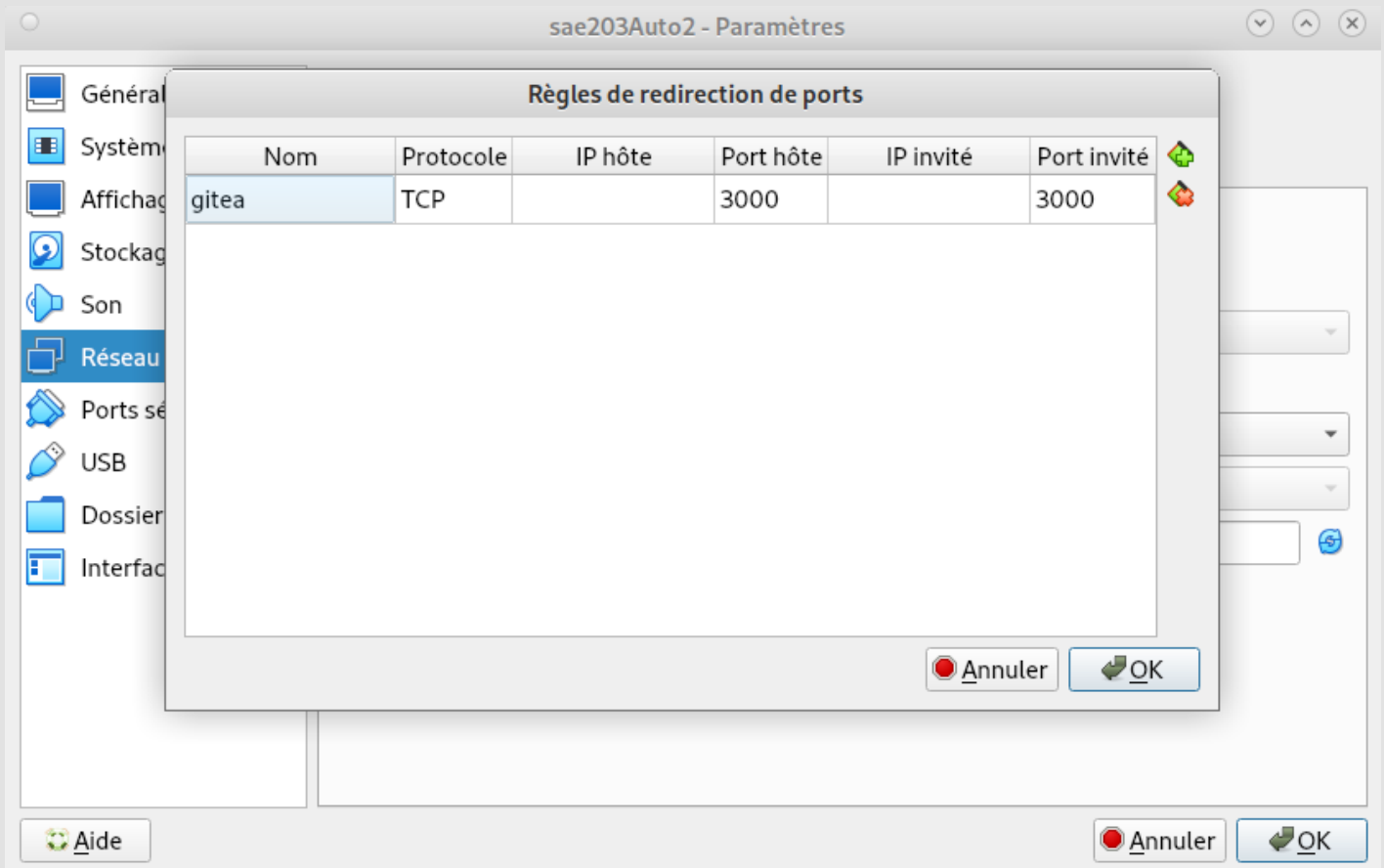
L'utilisation de **GitFiend** est plus **intuitive**, ce qui peut grandement **faciliter** la gestion d'un projet Git car l'interface peut paraître plus **"moderne"** pour certains. Néanmoins, il est vrai que **l'ajout de nouvelles options** n'est pas possible via l'interface **GitFiend**. Ce manque de possibilité d'ajout d'option peut parfois rendre le logiciel **obsolète** lorsque les fonctionnalités souhaitées deviennent **précises** et **spécifiques**.

Cette étape nous a posé problème lors de la **recherche d'un logiciel** autre que **gitk** et **git-gui** (*nous ayant ainsi amenées à l'installation de **GitFiend***) car bon nombre de logiciels d'interface **git** ne sont pas totalement **gratuit** et **libre**. Cependant, après quelque recherche, nous avons réussi à trouver GitFiend, que nous avons testé et rapidement réussi à prendre en main par son utilisation **simple** et **intuitive**. Il s'agit là d'un logiciel que nous pourrions évidemment réutiliser lors de nos prochains projets afin de simplifier la gestion de ces derniers.

4. Installation de Gitea

Mise en place redirection des ports

*Par la suite, nous nous sommes intéressé à **gitea** et ses **spécificités**. Ainsi, nous avons commencé par réaliser la **redirection du port 3000** de notre machine hôte vers le **port 3000** de notre machine virtuelle en se rendant dans les paramètres de notre machine virtuelle, puis dans la catégorie **« Réseau »**, puis nous avons cliquer sur le bouton situé en bas de la page intitulé **« Redirection de ports »**, suite à quoi nous avons créé une nouvelle redirection comportant les caractéristiques présents sur l'image ci-dessous :*



Présentation de Gitea

*Avant de commencer à installer **Gitea**, nous pouvons nous demander qu'est ce que Gitea ? Eh bien Gitea est une **forge logicielle** libre en **Go** sous la licence **MIT** permettant **l'hébergement de développement logiciel**, basé sur le logiciel de gestion de versions **Git** concernant la gestion du code source, comportant ainsi un système de suivi des bugs, un wiki, ainsi que des outils pour la relecture du code. Gitea est également disponible 24 heures sur 24, totalement libre et gratuit et permet également **l'import et l'export de données** ainsi que la **gestion des médias** et bien plus encore.*

Ce logiciel peut notamment être comparé à d'autres logiciels bien connus dans ce domaine, comme **GitLab** (offrant ainsi une **gestion du référentiel git**, les **révisions de code** et bien plus encore), ou bien même **GitHub** (étant ainsi le meilleur endroit pour **partager du code** avec des amis, des collègues, des camarades de classe ou bien même de parfait inconnus). Nous pouvons également retrouver des logiciels un peu moins connus tels que **Gogs**, **Phabricator** ou bien même **Bitbucket**.

Ainsi, nous avons commencer l'installation de **Gitea** sur notre machine virtuelle :

1ère étape

*Pour installer **Gitea**, nous somme passer par **Wget** (programme en ligne de commande non interactif de téléchargement de fichiers depuis le Web), à l'aide des*

commandes suivantes :

```
wget -O gitea https://dl.gitea.com/gitea/1.21.7/gitea-1.21.7-linux-amd64
chmod +x gitea
```

NOTE

(si wget n'est pas installé, il faut l'installer avant d'exécuter les commandes précédentes à l'aide de la commande `sudo apt install wget`).

Ensuite, nous avons vérifié les **signature GPS** à l'aide des commandes suivantes :

```
gpg --keyserver keys.openpgp.org --recv 7C9E68152594688862D62AF62D9AE806EC1592E2
gpg --verify gitea-1.21.7-linux-amd64.asc gitea-1.21.7-linux-amd64
```

NOTE

(dans cette dernière commande, **gitea-1.21.7-linux-amd64.asc** est à remplacer par le **.asc** souhaité et **gitea-1.21.7-linux-amd64** est à remplacer par la **version de gitea** souhaité, ici, il s'agit des fichiers installés par nos soins).

Nous avons ensuite vérifié que Git était installé sur notre machine à l'aide de la commande `git --version`.

Puis nous avons créé un utilisateur afin de lancer **Gitea** sur ce dernier à l'aide de la commande suivante :

```
adduser \ --system \ --shell /bin/bash \ --gecos 'Git Version Control' \ --group \ --disabled \
--password \ --home /home/git \ git
```

Par la suite, la création d'une **structure de dossiers** fût nécessaire afin d'installer **Gitea** à l'aide des commandes suivantes :

```
mkdir -p /var/lib/gitea/{custom,data,log} chown -R git:git /var/lib/gitea/ chmod -R 750
/var/lib/gitea/ mkdir /etc/gitea chown root:git /etc/gitea chmod 770 /etc/gitea
```

Ensuite, nous avons **configuré** le dossier **Gitea** à l'aide de la commande `export GITEA_WORK_DIR=/var/lib/gitea/`, puis nous avons **copié** le binaire **Gitea** vers une *localisation globale* à l'aide de la commande `cp gitea /usr/local/bin/gitea`.

2ème étape

afin de lancer Gitea, nous avons procédé ainsi :

Pour commencer, nous avons **créé** un fichier **« gitea.service »** dans le répertoire **« /etc/systemd/system/gitea.service »**, puis nous y avons **copié** quelques lignes nécessaires à son bon fonctionnement que nous ne mentionnerons pas ici car **beaucoup trop de lignes**.

Ensuite, nous avons **activé et lancé Gitea** en tant que boot avec les commandes suivantes :

```
sudo systemctl enable gitea  
sudo systemctl start gitea
```

NOTE

Il est également possible **d'activer** et **d'activer** et de **lancer Gitea** à l'aide de la commande suivante dans le cas où notre **systemd** se trouve en **version 220** ou plus :

```
sudo systemctl enable gitea --now
```

Par la suite, nous avons installer **supervisor** à l'aide de la commande suivante :

```
sudo apt install supervisor
```

NOTE

(Il s'agit là d'un **système client/serveur** qui permet à ses utilisateurs de **surveiller** et de **contrôler** un certain nombre de **processus** sur des **systèmes d'exploitation** de type **UNIX**).

La création d'un **fichier supervisor** fut ainsi nécessaire à l'utilisation de **supervisor**. Cela à été réalisé par nos soins à l'aide de la commande suivante :

```
mkdir /home/git/gitea/log/supervisor
```

NOTE

(Cette commande s'exécute en supposant que **Gitea** est installé dans le fichier « **/home/git/gitea** »).

L'une des dernières étapes fut de **configurer** notre **supervisor** en inscrivant dans le fichier « **/etc/supervisor/supervisord.conf** » les lignes suivantes :

```
directory=/home/git/go/src/github.com/go-gitea/gitea/      command=/home/git/go/src/github.com/go-  
gitea/gitea/gitea      web      autostart=true      autorestart=true      startsecs=10  
stdout_logfile=/var/log/gitea/stdout.log  stdout_logfile_maxbytes=1MB  stdout_logfile_backups=10  
stdout_capture_maxbytes=1MB  stderr_logfile=/var/log/gitea/stderr.log  stderr_logfile_maxbytes=1MB  
stderr_logfile_backups=10  stderr_capture_maxbytes=1MB  user      =      git  environment      =  
HOME="/home/git", USER="git"
```

Pour finir, nous avons **activé** et **lancé supervisor** en boot à l'aide des commandes suivantes :

```
sudo systemctl enable supervisor  
sudo systemctl start supervisor
```

NOTE (ou à l'aide de la commande `sudo systemctl enable supervisor --now`).

Les permissions du dossier **Gitea** ont également été **modifiées** une fois l'installation finie à l'aide des commandes suivantes :

```
chmod 750 /etc/gitea
chmod 640 /etc/gitea/app.ini
```

Points à savoir

Pour connaître la version de Gitea que nous avons installée, il faut exécuter la commande suivante :

```
gitea --version
```

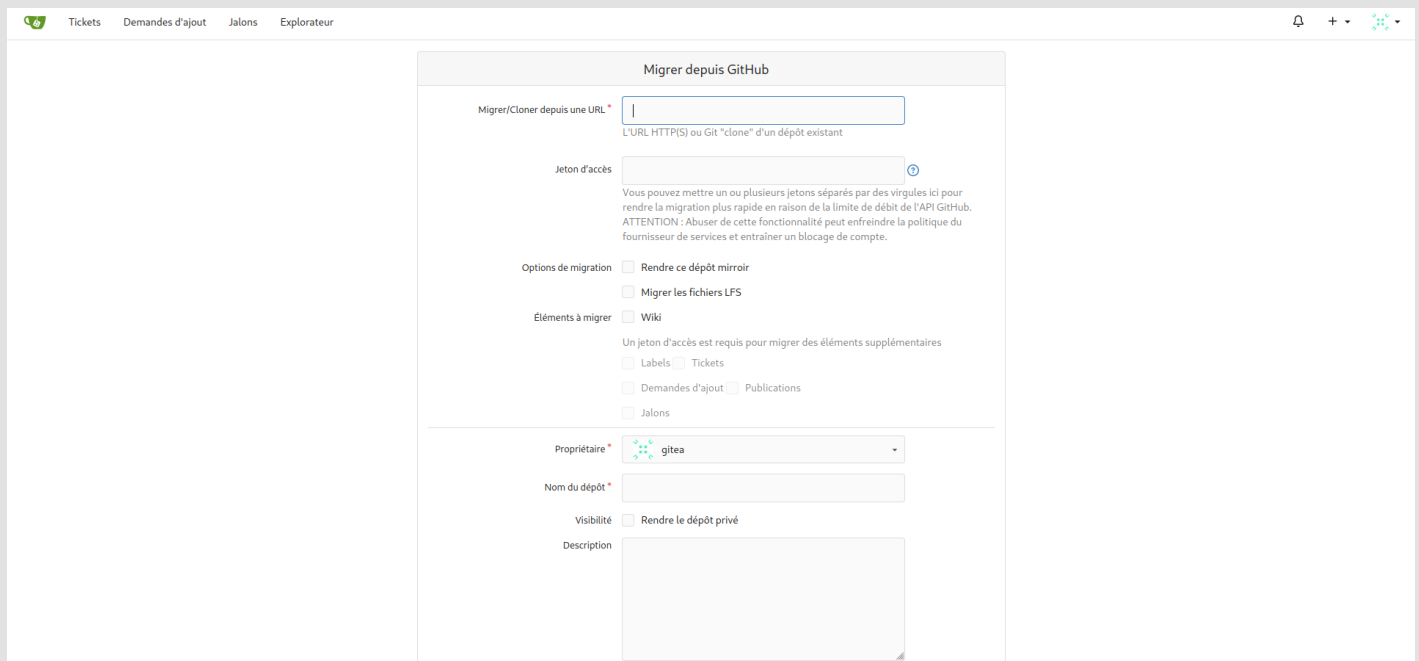
Dans notre cas, nous avons installé la **version 1.21.7** car il s'agit de la **dernière version** disponible de **Gitea**.

Pour **mettre à jour** le **binaire** de notre service sans devoir tout reconfigurer, nous devons **stopper Gitea** à l'aide de la commande `sudo systemctl stop gitea`, puis nous devons **remplacer** le **binaire** actuellement installé par le **binaire** souhaité à l'emplacement `« /usr/local/bin/gitea »`, puis nous devons **relancer Gitea**.

Utilisation de Gitea

Ainsi, nous pouvons dès à présent **utiliser Gitea** afin de réaliser plusieurs **tâches** plutôt efficaces pour la gestion de projet :

Nous pouvons commencer par **migrer** nos fichiers. Cela se présente de la manière suivante (ici, nous prendrons l'exemple de Github) :




Nous pouvons également **créer** un nouveau **dépôt**. Cela se présente de la manière suivante :

Tickets Demandes d'ajout Jalons Explorateur

Nouveau dépôt

Un dépôt contient tous les fichiers d'un projet, ainsi que l'historique de leurs modifications. Vous avez déjà ça ailleurs ? [Migrez-le ici.](#)

Propriétaire *  gitea

Certaines organisations peuvent ne pas apparaître dans la liste déroulante en raison d'une limite maximale du nombre de dépôts.

Nom du dépôt *

Idéalement, le nom d'un dépôt devrait être court, mémorisable et unique.

Visibilité ☐ **Rendre le dépôt privé**

Seuls le propriétaire ou les membres de l'organisation, s'ils ont des droits, seront en mesure de le voir.

Description

Décrive brièvement votre dépôt

Modèle

Répliquer un modèle

Jeu de labels pour les tickets

Sélectionner un jeu de label.

.gitignore

Sélectionner quelques .gitignore prédéfinies



De nombreux outils et compilateurs génèrent des fichiers résiduels qui n'ont pas besoin d'être supervisés par git. Composez un .gitignore à l'aide de cette liste des langages de programmation courants.

Licence

Sélectionner une licence

Une fois le dépôt créé, vous devriez atterrir sur cette page :

Tickets Demandes d'ajout Jalons Explorateur


 [gitea / rapport](#) 

Ne plus suivre 1 Ajouter aux favoris 0

<> Code Tickets Paquets Projets Wiki Paramètres

Introduction rapide

Cloner ce dépôt Besoin d'aide pour dupliquer ? Visitez [l'aide.](#)

Nouveau fichier Téléverser un fichier **HTTP** <http://localhost:3000/gitea/rapport.git> 

Création d'un nouveau dépôt en ligne de commande

```
touch README.md
git init
git checkout -b main
git add README.md
git commit -m "first commit"
git remote add origin http://localhost:3000/gitea/rapport.git
git push -u origin main
```

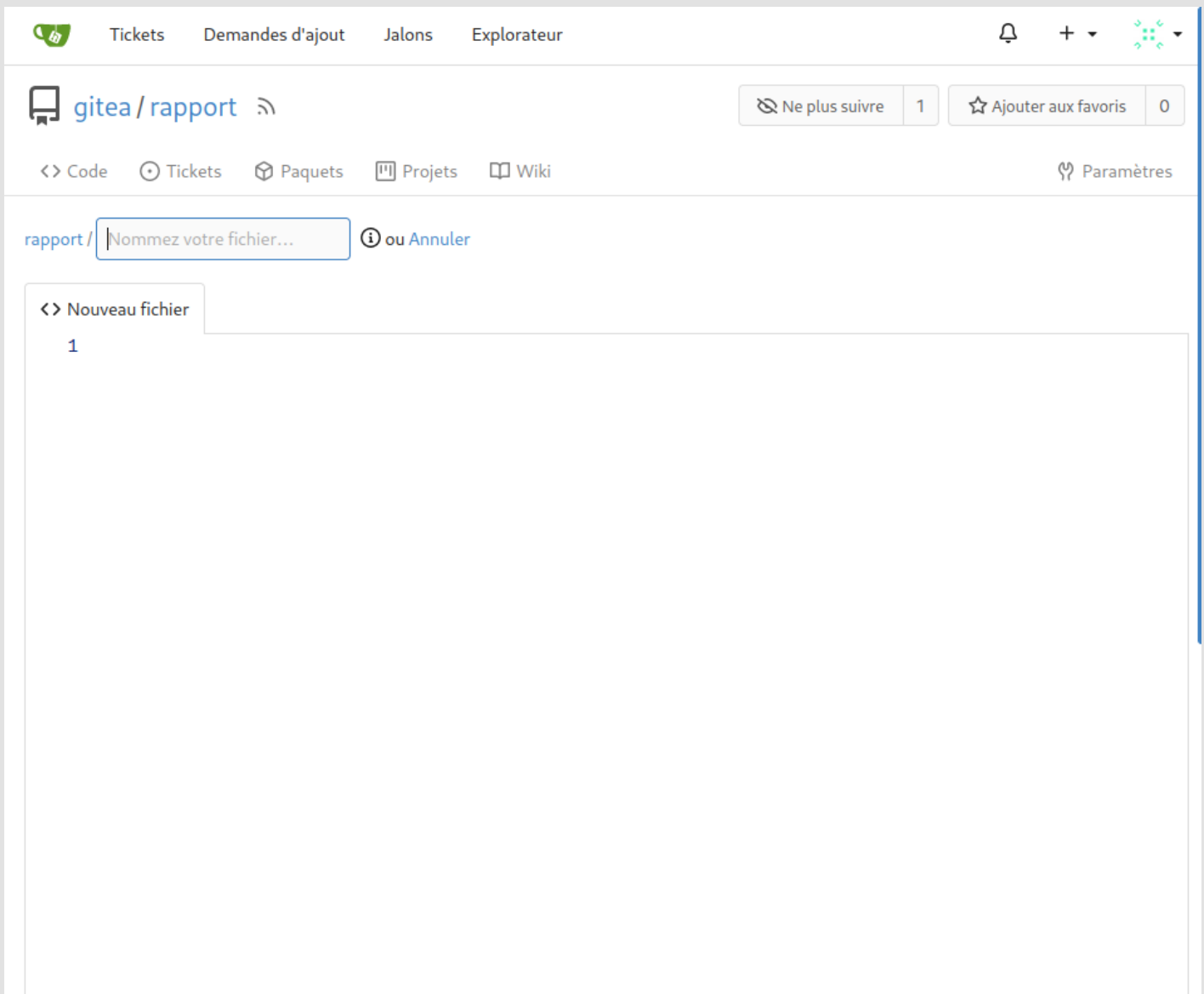
Soumission d'un dépôt existant par ligne de commande

```
git remote add origin http://localhost:3000/gitea/rapport.git
git push -u origin main
```

NOTE

(Cette page contient les éléments sauvegarder dans nos dépôts ainsi que la manière de cloner ce dépôt).

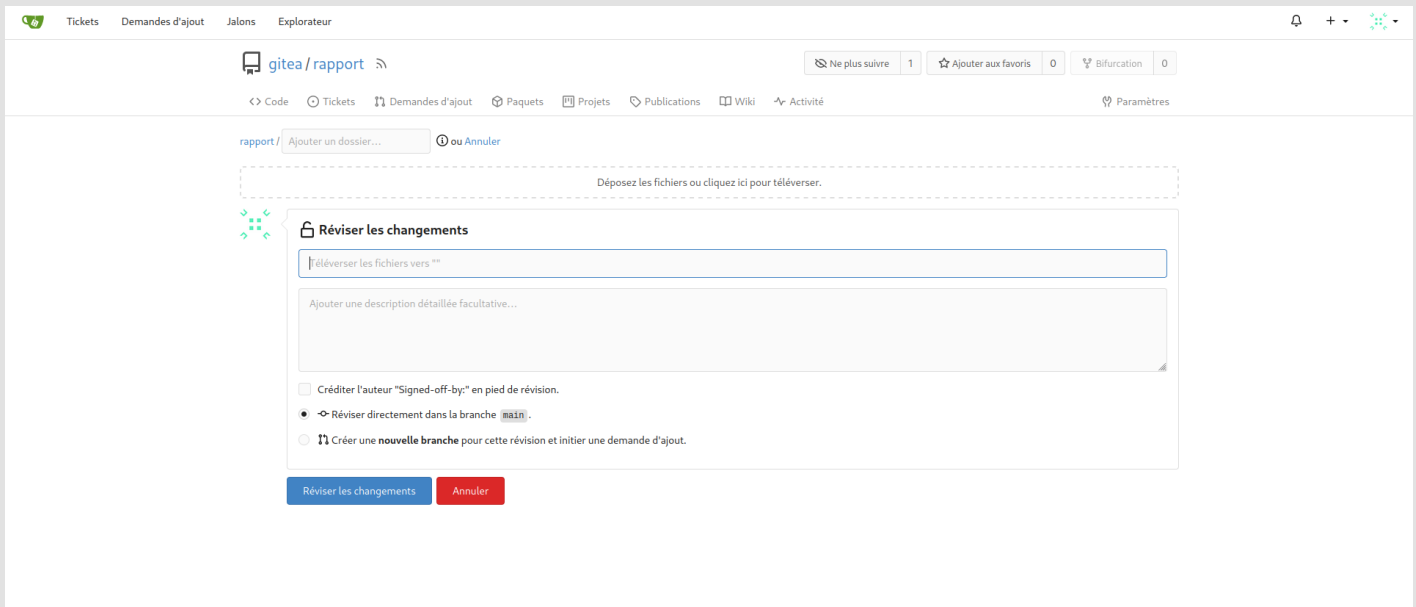
Nous pouvons également **créer** directement un **nouveau fichier** dans notre **dépôt** en cliquant sur le bouton **« Nouveau fichier »**, cela nous amène sur cette page :



NOTE

Dans notre cas, nous avons **créé** un **fichier** nommé **« test »** comportant un texte simple afin de nous permettre de **tester** cette fonctionnalité.

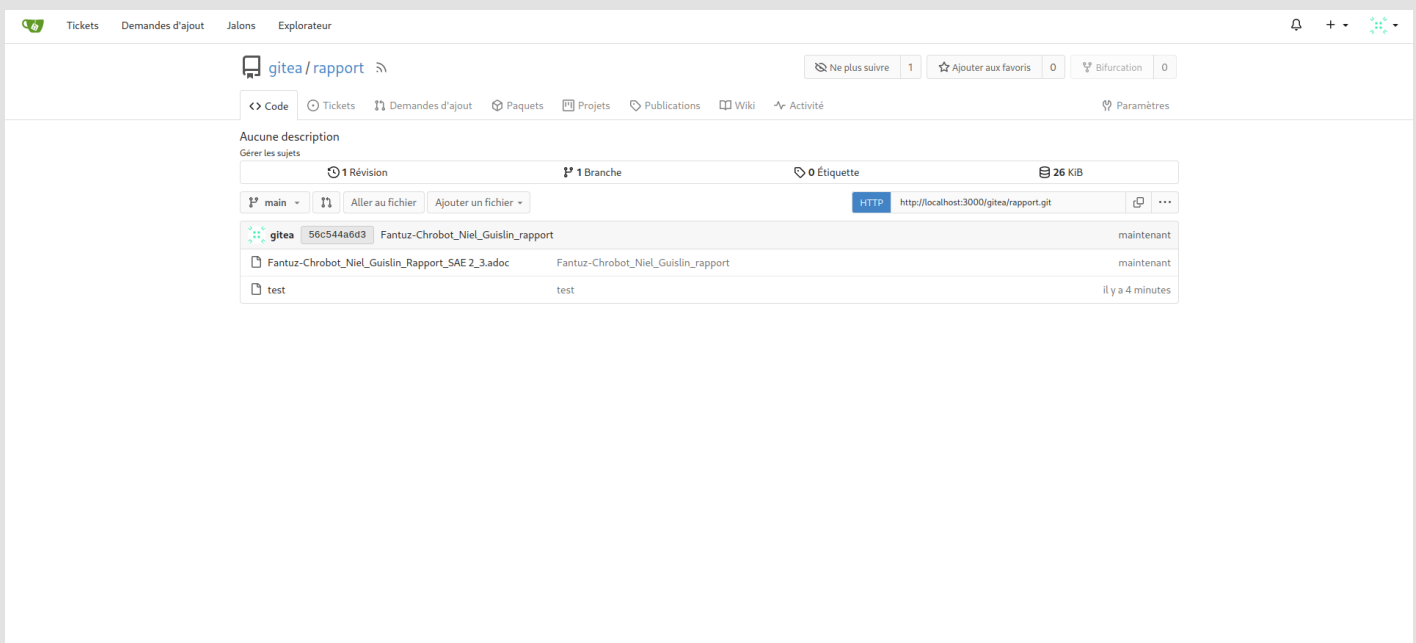
Nous pouvons également **ajouter/téléverser** un **fichier** **déjà existant** à l'aide du bouton **« Téléverser fichier »**, cela nous amène sur la page suivante :



NOTE

Dans notre cas, nous avons **téléversé** notre **rapport** dans le fichier « **rapport** », et nous l'avons intitulé « **Fantuz-Chrobot_Niel_Guislin_rapport** ».

Après avoir fait tout cela, voici le **résultat final** :



5. Balisage léger avancé

Mise en place d'un template personnalisé

Nous avons, pour cette dernière partie de notre projet, **amélioré** l'aspect esthétique de notre rapport via l'ajout d'un fichier CSS (CSS étant un langage informatique permettant de styliser une page Internet).

Nous avons d'abord commencé par **recupérer** le CSS déjà présent lors de la génération de la page **HTML**

(langage de structuration d'une page Internet) de notre rapport, puis nous y avons fait quelques modifications (à l'aide de quelque commande CSS) :

Parties modifiées	modification	couleur
Pages	Body: nouvelle image d'arrière plan (de couleur grise).	
	Bar de navigation: Bar de navigation disposée au début de la page afin d'éviter un surdosage de la page et changement de la couleur des liens de cette dernière passant de bleu à rouge lorsque la souris survole le texte.	
	Centrage des textes.	
Images	ajout d'une bordure.	Blanc
Textes en gras	changement de la couleur des textes en gras .	Violet
Textes surlignés	changement de la couleur des textes surlignés .	Bleu clair
Lien	changement de la couleur des liens.	Rouge
Sous-titres	Sous-titres disposés à gauche de la page et changement de leur couleur	Rouge clair
Codes sources	modification couleur du texte des codes sources ainsi que leur font	font: orange clair, couleur : rouge
Blocs de textes composés de codes sources (content)	changement de la couleur des blocs de textes composés de codes sources	orange clair

Pour finir, dans cette dernière partie de notre projet, nous avons également appris à **faire** un **tableau complexe** (comme présenté ci-dessus) de la manière suivante :

Fusion de colonnes

Pour avoir une **plage de cellules de colonnes consécutives**, il faut entrer le facteur de plage de colonne et l'opérateur de l'intervalle de la manière suivante : **<n>+** ("**<n>**" correspond au nombre de colonne dont la cellule sera composée). Cela doit être **mentionner** juste avant le début de la cellule souhaitée. **N'insérez pas d'espace** entre le **"+"**, les **opérateurs d'alignement ou de style** (en cas de présence) et le **séparateur de la cellule "****|****"**.

Fusion de lignes

Pour fusionner des lignes, le processus est le même, mais avec un **."** disposé juste avant le facteur de plage. Voici comment cela s'écrit: **."<n>+**.

Fusion de colonnes et lignes

Une seule cellule peut couvrir un bloc de colonnes et de lignes adjacentes. Inscrire le facteur de plage de la colonne **"<n>**", suivis du facteur d'intervalle de ligne **."<n>**", puis l'opérateur **"+"**.

Concernant le PDF, nous avons modifier les couleurs de ce dernier (pas tout à fait équivalent à celles de la page HTML histoire de faire varier les plaisirs), et nous avons également modifier l'emplacement de la bar de navigation (ainsi déplacée en haut de la page), et nous avons également centré les sous-titres de nos différentes parties.

NOTE

Il est bon de mentionner que pour modifier l'apparence d'un PDF généré avec asciidoctor, il ne faut pas utiliser un fichier CSS mais un fichier YML, en ajoutant l'extension `".yml"` au fichier comportant les modifications du style du PDF.

Index

A

additions invitées, 4
ADM64, 4
adm64, 4
AsciiDoctor, 9, 10, 11, 11

B

bases de données, 2

D

deb, 15
Debian, 3, 3, 5, 5, 5, 5, 5, 6
Debian 12, 3
Debian GNU/Linux, 15
données, 2

F

Fork, 3

G

Git, 11, 11, 11, 12
git-gui, 13, 14, 15
git-scm.com, 14
Gitea, 17, 17, 17, 17, 17, 17
GitFiend, 14, 15, 15, 16, 16
Gitk, 12, 13, 14, 15
Gnome, 3, 3, 3, 3

I

iso, 3, 3, 3

L

Linus Torvalds, 11
l'adresse réseau IP, 2

M

Markdown, 9, 10
MATE, 3, 3
mémoire, 2

N

noyau Linux, 4, 4, 5

P

Pandoc, 9
proxy, 3

S

serveur informatique, 3
serveur mandataire, 3
serveur web, 3, 3
serveurs mandataires, 3
SHA-1, 11
ssh, 3, 3
suppléments invités, 4
système de fichiers, 5, 5
système d'exploitation, 5

V

VirtualBox, 2, 4

W

Wget, 17

X

XML, 2

[1] Dans cette commande, "s02-Debian12.viso" est a modifier par le nom du ".viso" que l'on souhaite utilisé

[2] Selon les sources trouvées sur internet, la commande indiquée afin de lancer git gui peut parfois être `git-gui`, cependant, il se pourrait bien que cette commande ne fonctionne pas toujours.