
Rapport Saé2.03 : Installation de services réseaux

Par Maxence ANTOINE, Hugo DEBUYSER, Gaël
DIERYNCK

3-23-2024


Table des matières

1	Préparation d'une machine virtuelle Debian	2
1.1	Prérequis matériel de la machine	2
1.2	Installation de Debian 64 sur la machine virtuelle	3
1.3	Préparation du système	4
2	Quelques détails	5
2.1	La maintenance	5
2.2	Architectures prises en charge	5
3	Installation préconfigurée	7
3.1	Automatisation	7
3.1.1	Récupérer et préparer les fichiers nécessaires	7
3.1.2	Ajustement de la préconfiguration	7
4	Analyse préliminaire de git et des outils graphiques associés	10
4.1	Configuration globale de git	10
4.2	Les interfaces graphiques pour git	10
4.3	Installons autre chose et comparons	10
5	Installation de Gitea	12
5.0.1	À propos de Gitea	12
5.1	Installons Gitea !	12
5.1.1	Installation du binaire	12
5.1.2	Vérification des clés	12
5.1.3	Création de l'utilisateur	12
5.1.4	Création des fichiers de structure	13
5.1.5	Démarrage en tant que service automatique / Redémarrage	13
5.1.6	Installation Web	13
5.1.7	Changement des permissions de lecture :	15
5.1.8	Mise à jour	15
5.1.9	Tests d'utilisation	16
5.2	Transfert de dépôt	16
5.3	Création d'un dépôt	16

Ce compte-rendu présente l'installation d'une machine virtuelle Debian, que ce soit sa configuration matérielle, mais aussi logicielle. Puis nous en ferons une installation automatisée et nous présenterons comment utiliser git avec un gestionnaire de versions. Et enfin, nous présenterons et installerons gitea.

1 Préparation d'une machine virtuelle Debian

1.1 Prérequis matériel de la machine

Pour cette installation, nous utiliserons le système d'exploitation Debian et un environnement Mate où nous placerons 2 utilisateurs ainsi que les logiciels essentiels. Pour cela, nous utiliserons  **Debian 64-bit** comme la version de notre machine, cela signifie que nous utiliserons une version de Debian prévue pour un processeur 64 bits et gère les applications x64, x32 et x86. De plus il nous permet de gérer plus de 4Go de mémoire ce qui rend plus performant et de gérer des applications et des données plus importantes qu'un système 32 bits.

Caractéristiques	Valeurs
Nom de la machine	sae203
Dossier de la machine	/usr/local/virtual_machine/infoetu/login
Type	Linux
Version	Debian 64-bit
Mémoire vive (RAM)	2048 Mo
Disque dur	20 Go

Nous laisserons les autres paramètres par défaut, notamment celle du réseau qui est par défaut en NAT qui permet à la machine virtuelle d'accéder à Internet via la connexion de l'hôte.

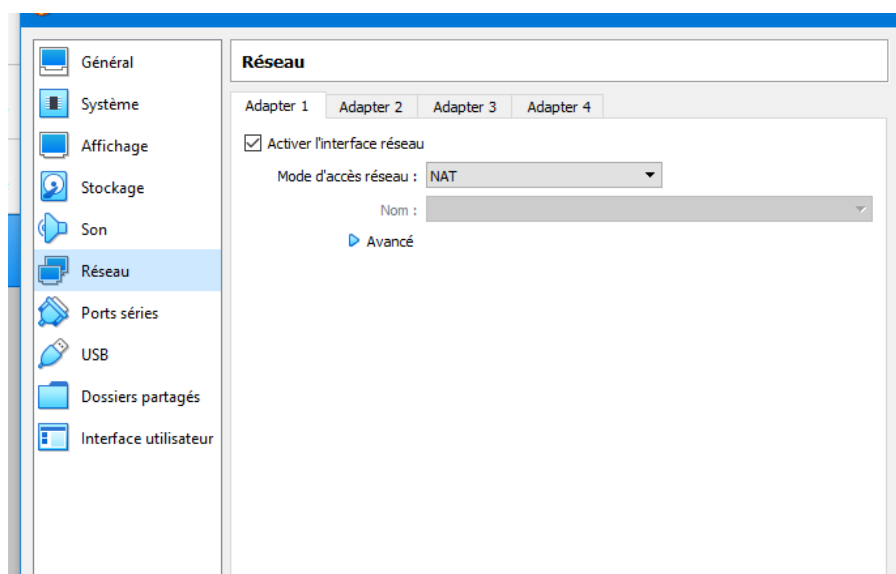


Figure 1: Configuration Réseau par défaut

Ces configurations sont sauvegardées dans le fichier `serveur.vbox.xml`, ainsi, il nous est possible de modifier directement la configuration de la machine depuis celui-ci. Par exemple, en ajoutant la ligne `<CPU count="1">` permet de modifier le nombre de processeurs de la machine à 2, ce fichier nous sera utile plus tard.

1.2 Installation de Debian 64 sur la machine virtuelle

Pour cette installation, nous utiliserons l'ISO bootable de Debian qui est une image disque capable de démarrer et d'installer un système d'exploitation, on la trouve sur le site officiel de Debian. Il permet de charger le système d'exploitation à partir de l'image disque ce qui permet à l'utilisateur d'installer le système d'exploitation sur le disque dur de l'ordinateur. Nous laisserons la plupart des paramètres par défauts, mais nous choisirons l'environnement MATE qui est un environnement de bureau qui affecte l'apparence visuelle et l'interaction de l'utilisateur avec le système. Nous pouvons également utiliser l'environnement GNOME qui est également populaire et utilisé par défaut, mais MATE offre une expérience intuitive, en plus d'être léger.

Caractéristiques	Valeurs
Nom de la machine	serveur
Domaine	Laisser vide
Pays/langue	France
Environnement	Mate
Miroir	http://debian.polytech-lille.fr
Proxy	pas de proxy
Compte administrateur	root / root
Compte utilisateur	User / user / user

Caractéristiques	Valeurs
Partition	1 seule partition de la taille du disque

Nous avons également choisi d'inclure un serveur web qui permet de host des pages web avec Apache ou Nginx, mais aussi partager des fichiers et tout types de documents. Nous avons aussi inclus un serveur SSH qui offre un accès sécurisé à distance pour la gestion du système, il est couramment utilisé pour gérer à distance les systèmes, l'administration de serveurs ou le transfert de fichiers [Serveur SSH](#). Il nous était également possible d'inclure un [serveur mandataire](#) qui est un serveur proxy jouant un rôle d'intermédiaire entre 2 hôtes, un client et un serveur afin d'accélérer ou de surveiller les échanges produits. Une fois l'installation finie, nous devons passer sur la configuration de celui-ci.

1.3 Préparation du système

Pour simplifier la gestion du système, nous avons accordé à l'utilisateur standard le droit d'utiliser sudo, cela facilite l'exécution de commandes avec les privilèges administratifs. Pour cela, nous sommes connectés sous root en mode console qui est un mode que l'on peut accéder en pressant **Ctrl + Alt + F1**. Ajoutez l'utilisateur au groupe sudo avec la commande `usermod -aG sudo user`. Afin de vérifier si l'opération a réussi, il nous suffit de faire la commande `groups user`, il ne nous reste plus qu'à redémarrer la machine pour nous connecter au compte `user` et utiliser la commande `sudo`.

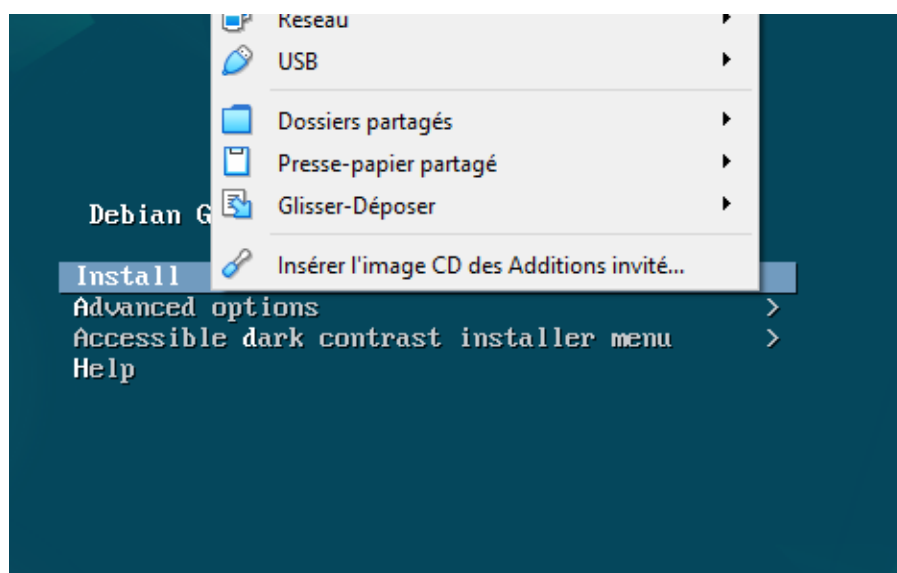



Figure 2: Insertion de l'image CD des additions invités

Les suppléments invités sont des outils ajoutés pour améliorer les performances et les fonctionnalités. Nous les avons installés en montant le CD des suppléments depuis Virtual Box : **Périphériques › Insérer l'image CD des additions invités...**


Il nous suffit ensuite de monter le CD avec la commande `sudo mount /dev/cdrom /mnt` et d'installer les suppléments avec `sudo /mnt/VBoxLinuxAdditions.run`

2 Quelques détails

En installant  **Debian**, on installe **Linux**, qui est un noyau. Le noyau est l'élément central d'un système d'exploitation. Il s'occupe d'interpréter le code des applications et de faire fonctionner les composants du système sans qu'on ait à s'en soucier. La version installée est la 6.1.0-17-**amd64**(vérifié avec neofetch), où **amd64** désigne bel et bien notre architecture de système **x64**.

La machine étant installée, nous pouvons commencer à installer des logiciels et composants essentiels.

Le premier de la liste quand on possède une machine virtuelle sont les Suppléments invités. Les suppléments servent à plusieurs raisons, l'une est la faculté de copier coller entre la machine et la VM. L'autre raison est une meilleure stabilité et des meilleures performances graphiques ainsi que le Drag & drop entre les machines et bien plus encore.. Pour installer les suppléments il suffit d'utiliser la commande **mount**, qui sert à monter le disque dans notre système de fichier pour qu'on puisse en voir le contenu, à la manière de rentrer un CD dans une machine et d'explorer ses fichiers

Pour les plus curieux, savoir ce qu'on installe Debian, déjà 31 ans de loyaux services. Créée en 1993 par Ian Murdock,  **Debian** se veut être composé de logiciels entièrement libres et donc, gratuits. Le nom se prononce 'Debyianne' et non 'debyan'. Pour la petite histoire, il vient de la fusion de 'Deb', les trois premières lettres du prénom de la femme de Ian(Debra) ainsi que des siennes, donnant Debian. En voilà un homme amoureux.

Les versions de Debian sont nommées après des personnages de Toy Story, voilà pourquoi la toute première vraie version: Debian 1.1 Buzz sortie le 17 Juin 1996 porte le nom de Buzz. Le dernier nom de code annoncé est Forky pour la version 14, annoncé le 13 octobre 2022.

2.1 La maintenance

Côté mise à jour, Debian possède un support de 3 ans de mises à jour, cumulé avec les 2 ans du support long terme LTS, on obtient donc 5 **ans de mises à jour**. Il reste le programme ELTS qui rajoute 5 ans de stabilité, mais **The Debian Project** n'est pas affilié avec les développeurs de **Freexian**(Programme ELTS).

Jusqu'à maintenant Debian 12 Bookworm, 11 Bullseye et 10 Buster sont encore maintenus à jour. Mais quand Debian 13 sortira, Buster ne sera plus pris en charge et sera déclaré comme OldStable

2.2 Architectures prises en charge

Voici la liste des architectures prises en charge :

Architecture	Pris en charge ?	Description
Armel	Oui	Architecture ARM à petit endian (little-endian). Principalement utilisée sur des appareils embarqués et IoT.
arm64	Oui	Architecture ARM 64 bits, également connue sous le nom de AArch64. Adaptée aux serveurs, stations de travail, etc.
i386	Oui	Architecture Intel x86 32 bits. Communément utilisée sur des systèmes plus anciens ou des applications spécifiques.
AMD64	Oui	Architecture Intel/AMD x86 64 bits, la plus répandue sur les ordinateurs de bureau et serveurs modernes.
ARMhf	Oui	Architecture ARM à petit endian (little-endian) avec matériel à virgule flottante (floating-point).
mipsel	Oui	Architecture MIPS à petit endian (little-endian). Souvent utilisée sur des routeurs et des équipements réseau.
mips64el	Oui	Architecture MIPS 64 bits à petit endian (little-endian). Utilisée sur des systèmes embarqués et des serveurs.
ppc64el	Oui	Architecture PowerPC 64 bits à petit endian (little-endian). Principalement utilisée sur des serveurs.
s390x	Oui	Architecture IBM zSeries 64 bits. Principalement utilisée sur des mainframes.

Le reste listé est supporté, mais pas officiel :

Alpha, hppa, ia64, m68k, mips(que sous 10 Buster), PowerPC, PPC64, riscv64, SH4, sparc64 ainsi que les processeurs 32bits.

À ce jour, **The Debian Project** compte 996 développeurs, le pic de devs était de 1075 en 2008.

3 Installation préconfigurée

3.1 Automatisation

Dans le cadre de cette installation automatisée, nous utiliserons les capacités de VirtualBox pour démarrer une installation avec des fichiers de configuration préexistants sur notre système de fichiers local. Cela nous permettra de préconfigurer le système de base ainsi que d'installer les suppléments invités de manière automatique, comme nous l'avons fait manuellement lors de la première semaine.

3.1.1 Récupérer et préparer les fichiers nécessaires

Nous avons commencé par récupérer l'archive `autoinstall_Debian.zip` sur Moodle, contenant les fichiers nécessaires à l'installation automatisée. Après avoir décompressé cette archive dans le répertoire de la machine virtuelle, nous avons effectué les étapes suivantes :

- Nous avons remplacé la chaîne `@@UUID@@` par un identifiant unique universel dans le fichier `S203-Debian12.viso`, en utilisant la commande suivante : `sed -i -E "s/(--iprt-iso-maker-file-marker-bourne-sh). *$/\1=$(cat /proc/sys/kernel/random/uuid)/" S203-Debian12.viso`
- Nous avons inséré le fichier `S203_Debian12.viso` dans le lecteur optique (cd/dvd) de la machine virtuelle.
- En démarrant la machine virtuelle, nous avons laissé l'installation se dérouler jusqu'au reboot.
- Après le reboot, nous avons testé les ajouts invités en nous connectant en utilisant les identifiants (`user/user` ou `root/root`), puis en modifiant la taille de la fenêtre pour vérifier leur fonctionnement.

Le premier de ces fichiers est `SAE203-Debian.viso`. Ce fichier est conçu pour être placé dans un lecteur optique d'une machine virtuelle et démarre comme une ISO bootable. Cependant, il permet également l'insertion automatique des suppléments invités et offre un accès simplifié à notre configuration personnalisée directement sur notre système hôte.

Le fichier de configuration automatisée pour une Debian (et ses dérivées) est le plus important. Il est souvent nommé `preseed.cfg`. Ce fichier répertorie l'ensemble des informations et actions à effectuer automatiquement lors de l'installation.

3.1.2 Ajustement de la préconfiguration

Pour ajuster la configuration préétablie, nous devons modifier le fichier `preseed.cfg`. Dans notre cas, nous devons ajouter les éléments suivants :

- Installer l'environnement MATE.


```

75 d-i apt-setup/services-select multiselect security, updates
76 #d-i apt-setup/restricted boolean true
77 #d-i apt-setup/universe boolean true
78
79 ## Installation meta-paquetages
80 # Tâches à installer (via des méta-paquetages)
81 # Lister les possibilités : tasksel --list-task (en ligne de commande)
82 # Utiliser au minimum "standard" est une bonne idée
83 tasksel tasksel/first multiselect standard ssh-server, mate-desktop
84
85
86 ### Suivi statistiques paquets installés
87 popularity-contest popularity-contest/participate boolean false
88
89
90 ### Grub
91 d-i grub-installer/grub2_instead_of_grub_legacy boolean true
92 d-i grub-installer/only_debian boolean true
93 d-i grub-installer/timeout string 2
94 # To install to the first device (assuming it is not a USB stick):
95 d-i grub-installer/bootdev string default
96

```

Figure 3: Installation de MATE

- Ajouter le droit d'utiliser sudo à l'utilisateur standard.
- Ajouter les paquets suivants :
 - `sudo` (nécessaire pour la gestion sudo)
 - `git`
 - `sqlite3`
 - `curl` (pour préparer l'installation de la semaine prochaine)
 - `bash-completion` (pour simplifier l'écriture des lignes de commande)
 - `neofetch` (un outil pour afficher des informations système)

```

# Packages needed for user.
#
# -----
#
# Installing packages for building kernel modules... | tee -a "${MY_LOGFILE}"
log_command_in_target apt-get -y install build-essential
log_command_in_target apt-get -y install linux-headers-$(uname -r)
#
#
#
#
#
#
# -----
#
# Installing VirtualBox Guest Additions... | tee -a "${MY_LOGFILE}"
MY_IGNORE_EXITCODE=2 # returned if modules already loaded and reboot required.
log_command_in_target /bin/bash "${MY_CHROOT_CDROM}/vboxadditions/VBoxLinuxAdditions.run" --nox11
log_command_in_target /bin/bash -c "udevadm control --reload-rules" # GAs doesn't yet do this.
log_command_in_target /bin/bash -c "udevadm trigger" # (ditto)
MY_IGNORE_EXITCODE=
log_command_in_target usermod -a -G vboxsf "user"
log_command_in_target usermod -a -G sudo "user"
log_command_in_target apt-get install sudo
log_command_in_target apt-get install sqlite3
log_command_in_target apt-get -y install curl
log_command_in_target apt-get -y install bash-completion
log_command_in_target apt-get -y install neofetch
log_command_in_target apt-get -y install git-all
#
#
# Test Execution Service.
#
#
# Run user command.
#

```

Figure 4: Installation des paquets

Une fois ces ajustements effectués, nous pouvons recommencer l'installation en utilisant les fichiers préconfigurés. Notre installation sera ainsi personnalisée et prête pour les tâches à venir.

```
root@debian:~# neofetch
root@debian
OS: Debian GNU/Linux 12 (bookworm) x86_64
Host: VirtualBox 1.2
Kernel: 6.1.0-13-amd64
Uptime: 1 min
Packages: 1326 (dpkg)
Shell: bash 5.2.15
Resolution: 800x600
DE: MATE 1.26.0
WM: Metacity (Marco)
Theme: Menta [GTK2/3]
Icons: menta [GTK2/3]
Terminal: mate-terminal
Terminal Font: Monospace 13
CPU: 11th Gen Intel i5-1135G7 (1) @ 2.419GHz
GPU: 00:02.0 VMware SVGA II Adapter
Memory: 357MiB / 1967MiB
```

Figure 5: Test de neofetch

4 Analyse préliminaire de git et des outils graphiques associés

4.1 Configuration globale de git

Le profile Git est établi par un nom, un mail et la branche dans laquelle nous souhaitons travailler. Pour mettre en place le profil git, il faut utiliser les commandes suivantes :

1. `git config --global user.name "[Prenom] [NOM]"`
2. `git config --global user.email "[prenom].[nom].etu@univ-lille.fr"`
3. `git config --global init.defaultBranch "master"`

4.2 Les interfaces graphiques pour git

Pour installer les deux interfaces graphiques de git sur Debian, il suffit d'utiliser les deux commandes suivantes :

1. `sudo apt install gitk`
2. `sudo apt install git-gui`

`git-gui` est une interface graphique pour créer des commits, des branches, etc. Pour l'utiliser, il faut exécuter la commande `git gui` depuis un terminal, et l'interface graphique permettra à l'utilisateur de créer un nouveau dépôt ou d'en ouvrir un déjà existant.

4.3 Installons autre chose et comparons

Pour l'utilisation de git, nous avons choisi `ungit` qui offre une interface graphique qui est facile pour des utilisateurs non-initiés à l'utilisation de git. Il permet de visualiser les différentes opérations comme des *commit* ou les *branches*, c'est une bonne alternative à l'utilisation de ligne de commande.

Pour l'installer, rien de plus simple ! Il suffit d'entrer cette commande dans un terminal Linux :

```
1 npm install -g ungit
```

Une fois installer, il suffit d'entrer la commande `ungit` dans le répertoire.

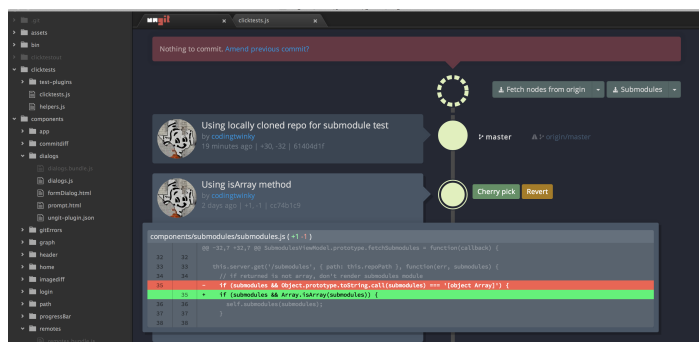


Figure 6: Aperçu de ungit

Comparaison :

- Ungit : Avec son interface graphique conviviale, Ungit simplifie la visualisation des opérations Git pour les débutants. Cependant, son utilisation nécessite Node.js.
- Ligne de commande Git : Avec son interface graphique conviviale, Ungit simplifie la visualisation des opérations Git pour les débutants. Cependant, son utilisation nécessite Node.js.
- Outils incluant Git : Ces outils offrent une variété de fonctionnalités pour gérer les dépôts Git et peuvent être intégrés dans des IDE. Ils offrent une flexibilité pour personnaliser les flux de travail.

5 Installation de Gitea

5.0.1 À propos de Gitea

Gitea est une plateforme de gestion de dépôts Git auto-hébergée et open source, offrant des fonctionnalités similaires à celles de GitHub ou GitLab. On peut le comparer à des solutions telles que GitHub et GitLab, qui sont également des plateformes de gestion de dépôts Git.

5.1 Installons Gitea !

5.1.1 Installation du binaire

Pour installer Gitea à partir des binaires, suivez ces étapes

Pour la première étape, nous utiliserons le service [wget](#) qui nous permet de télécharger le contenu nécessaire à l'installation. Dans un terminal, entrez

```
1 wget -O gitea https://dl.gitea.com/gitea/1.21.7/gitea-1.21.7-  
  linux-amd64
```

puis assurez vous d'avoir les droits d'exécution en entrant la commande suivante: `chmod +x gitea`.

5.1.2 Vérification des clés

Les fichiers de gitea sont signés par des clés [gpg](#) qui empêchent toutes modifications non voulues des fichiers binaires. Vérifiez les clés avec:

```
1 gpg --keyserver keys.openpgp.org --recv 7  
  C9E68152594688862D62AF62D9AE806EC1592E2
```

ainsi que

```
1 gpg --verify gitea-1.21.7-linux-amd64.asc gitea-1.21.7-linux-  
  amd64
```

Vous devriez avoir un message qui ressemble à **Good signature from “Teabot [teabot@gitea.io](#)”**.

5.1.3 Création de l'utilisateur

Naturellement, pour utiliser gitea il vous faut un compte utilisateur, voici la marche à suivre :

Sur Ubuntu/Debian:

```
1 sudo adduser \    # ajout de l'utilisateur  
2   --system \    # Signaler qu'on fait un utilisateur système  
3   --shell /bin/bash \    # On déclare le shell bash par défaut
```

```
4  --gecos 'Git Version Control' \ # Données pour un fichier
    situé dans le /etc/passwd
5  --group \ # Création d'un groupe git
6  --disabled-password \ # Pour ne pas avoir à renseigner de
    mots de passe
7  --home /home/git \ # Création d'un dossier home
8  git # Nom de l'utilisateur
```

5.1.4 Création des fichiers de structure

La structure nécessaire et importante pour gitea. Elle contiendra tous les fichiers de configuration et de maintenance, créez-là avec ces commandes :

```
1  mkdir -p /var/lib/gitea/{custom,data,log}
2  chown -R git:git /var/lib/gitea/
3  chmod -R 750 /var/lib/gitea/
4  mkdir /etc/gitea
5  chown root:git /etc/gitea
6  chmod 770 /etc/gitea
```

5.1.5 Démarrage en tant que service automatique / Redémarrage

Copiez le fichier `gitea.service` fourni par Gitea jusqu'à `/etc/systemd/system/gitea.service` et vérifiez la configuration du fichier de service avec la commande `nano gitea.service`

Entrez ensuite les commandes suivantes :

`sudo systemctl enable gitea`: Qui active le processus Gitea au démarrage

`sudo systemctl start gitea`: Qui démarre le processus Gitea

Pour redémarrer Gitea, la commande est la suivante : `systemctl restart gitea`.

5.1.6 Installation Web

Le service étant démarré, lancez un navigateur web sur votre machine hôte, puis connectez vous au serveur web à l'adresse: `localhost:3000`.

← → ↻

localhost:3000

☆

🔒

📁

☰

Configuration initiale

Si vous exécutez Gitea dans Docker, veuillez lire la [documentation](#) avant de modifier les paramètres.

Paramètres de la base de données

Gitea nécessite MySQL, PostgreSQL, MSSQL, SQLite3 ou TiDB (avec le protocole MySQL).

Type de base de données *

SQLite3

Emplacement *

/var/lib/gitea/data/gitea.db

Chemin d'accès pour la base de données SQLite3.
Entrer un chemin absolu si vous exécutez Gitea en tant que service.

Configuration générale

Titre du site *

Gitea: Git with a cup of tea

Entrez ici le nom de votre société.

Emplacement racine des dépôts *

/var/lib/gitea/data/gitea-repositories

Les dépôts Git distants seront stockés dans ce répertoire.

Répertoire racine Git LFS

/var/lib/gitea/data/lfs

Les fichiers suivis par Git LFS seront stockés dans ce dossier.
Laissez vide pour désactiver LFS.

Figure 7: Configuration Gitea

Exécuter avec le compte d'un autre utilisateur *

git

Le nom d'utilisateur du système d'exploitation sous lequel Gitea fonctionne. Notez que cet utilisateur doit avoir accès au dossier racine du dépôt.

Domaine du serveur *

localhost

Domaine ou adresse d'hôte pour le serveur.

Port du serveur SSH

22

Port d'écoute du serveur SSH. Laissez le vide pour le désactiver.

Port d'écoute HTTP de Gitea *

3000

Port sur lequel le serveur web Gitea attendra des requêtes.

URL de base de Gitea *

http://localhost:3000/

Adresse HTTP(S) de base pour les clones git et les notifications par courriel.

Chemin des journaux *

/var/lib/gitea/log

Les fichiers de journalisation seront écrits dans ce répertoire.

☐ Activer la vérification des mises-à-jour

Vérifie les mises à jour régulièrement en se connectant à gitea.io.

Figure 8: Configuration Gitea

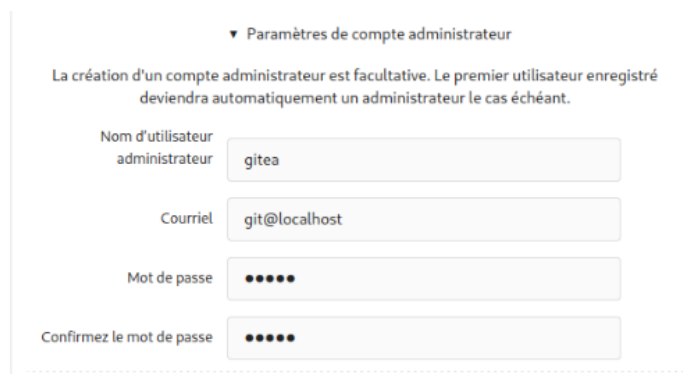
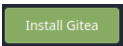


Figure 9: Configuration du compte administrateur Gitea

Une fois vos modifications apportées, il est temps de clôturer la partie configuration web en appuyant sur .

5.1.7 Changement des permissions de lecture :

Maintenant que Gitea est installé, nous pouvons changer les permissions de lecture en read-only des fichiers de configuration de Gitea dans `/etc/gitea`, que nous avons précédemment accordés en écriture au groupe `git`.

Dans le terminal, entrez :

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

5.1.8 Mise à jour

La version du binaire installée peut être obtenue en exécutant la commande suivante :

```
1 gitea --version
```

Et nous voyons que nous avons la version **1.21.7**.

Pour mettre à jour Gitea, vous devez stopper le `service gitea` avec la commande `sudo systemctl stop gitea` et remplacer le fichier binaire de Gitea localisé dans `/usr/local/bin/gitea` par le nouveau. **Attention, une sauvegarde avant le remplacement est toujours la bienvenue !**

Puis nous téléchargeons le fichier binaire nightly de Gitea :

```
1 wget -O gitea https://dl.gitea.com/gitea/main/gitea-main-nightly
  -linuxamd64
2 chmod +x gitea
```

Et en remplaçant le fichier binaire Gitea par le nouveau : `sudo mv gitea /usr/local/bin/gitea`. Il ne nous reste plus qu'à redémarrer Gitea avec la commande `sudo systemctl start gitea`. Il nous suffit de vérifier que la nouvelle version est installée avec la commande précédente.

5.1.9 Tests d'utilisation

5.2 Transfert de dépôt

Nous allons faire un transfert de dépôt de notre GitLab de développement objet vers notre Gitea. Pour se faire, nous allons d'abord cloner notre dépôt dans un dossier local :

```
1 git clone https://gitlab.univ-lille.fr/hugo.debuyser.etu/dev-oo
   dev-oo
```

Une fois fait, on supprime son lien avec le GitLab avec la commande `git remote remove origin`.

Nous créons un nouveau lien avec Gitea avec la commande `git remote add origin http://localhost:3000/gitea/dev-oo` et on sélectionne la branche `master` : `git branch -m main master`.

Puis une fois cela fait, on push nos modifications avec la commande `git push origin master`.

Ainsi, notre dépôt est maintenant sur Gitea et parfaitement utilisable.

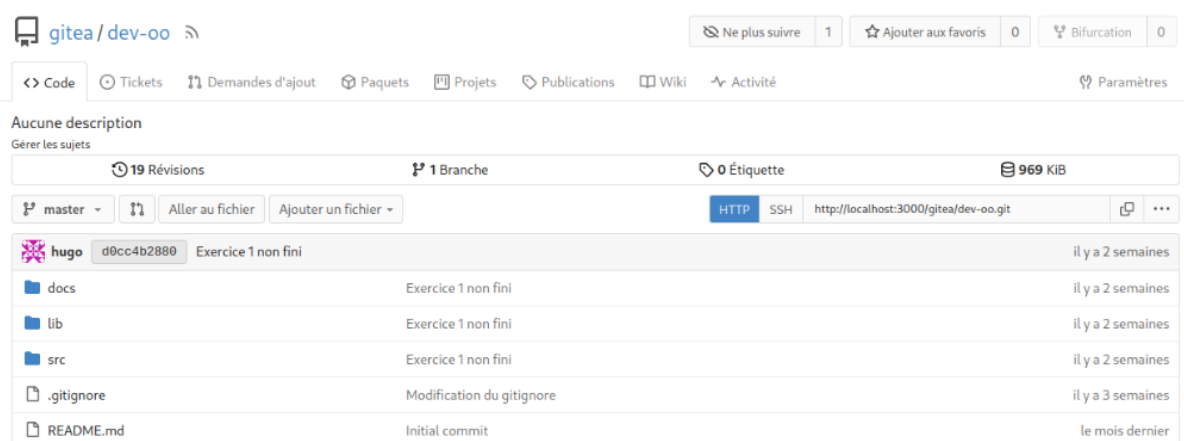


Figure 10: Dépôt de nos fichiers dev-oo sur Gitea

5.3 Création d'un dépôt

À présent intéressons nous à la création de dépôt sur Gitea, nous verrons cela avec notre Saé2.03, commençons d'abord par créer un dépôt :

```
1 mkdir sae2.03
```

Puis nous nous plaçons à l'intérieur avec la commande `cd sae2.03` et on initialise le dépôt avec la commande `git init`.

Ajoutons maintenant le lien à Gitea :

```
1 git remote add origin http://localhost:3000/gitea/sae2.03
```

Une fois cela fait il nous suffit d'ajouter les différents fichiers qui compose notre saé et de mettre à jour le dépôt. Pour ce faire nous ferons ces commandes :

- `git add .` afin d'ajouter les fichiers au dépôt de git.
- `git commit -m 'initial commit'` nous permet d'ajouter un commit, un message pour notre premier dépôt.
- `git push origin master` pour envoyer les ajouts à notre instance Gitea.

Nous avons maintenant créé un projet distant sur notre Gitea avec d'y mettre nos document de notre saé.

The screenshot shows the Gitea web interface for a repository named 'sae203'. At the top, there are navigation links: 'Code', 'Tickets', 'Demandes d'ajout', 'Paquets', 'Projets', 'Publications', 'Wiki', 'Activité', and 'Paramètres'. Below these, there's a section for 'Aucune description' and 'Gérer les sujets'. The main content area displays the repository details: '1 Révision', '1 Branche', '0 Étiquette', and '899 KiB'. Below this, there's a table of files and their commit history.

File	Commit	Status
Hugo Debuyser	116be2236f	initial commit
Rendu	initial commit	maintenant
Rendu Final	initial commit	maintenant
capture	initial commit	maintenant
README.md	initial commit	maintenant
VMSaé.zip	initial commit	maintenant

Figure 11: Dépôt de nos fichiers de la Saé2.03 sur Gitea