Noé Midoux – Pierre Vardon | BTS SIO2

INSTALLATION D’UN ENVIRONNEMENT DE DEVELOPPEMENT

Plan de mission :

étape 1 : trouvé une version d’**Ubuntu** adapté à la situation
étape 2 : téléchargé **Ubuntu 24.04 LTS**
étape 3 : installée sur **VirtualBox**
étape 4 : lancé **Ubuntu** et faire le programme d’installation
étape 5 : faire les mises a jour de sécurité du système
étape 6 : créer un compte développeur et donné ces différents rôles et droits
étape 7 : installé les logiciels nécessaires aux utilisateurs
étape 8 : configurer le réseau (pare-feu, antivirus, SSH, …)
bonus : comparatif entre **GitHub** et **GitLab**

Etape 1 : trouvé une version d’Ubuntu adapté à la situation

Pour trouvé la version adapté nous devons nous poses 3 questions :

- Quelle version a le meilleur niveau de sécurité et corrige les failles les plus connues ?

- Quelle version est compatibilité avec **Visual Studio Code** et ses extensions (**GitLab Workflow** et **GitHub Pull Requests and Issues**) ?

- Quelle version bénéficie du support le plus long en termes de mises à jour ?

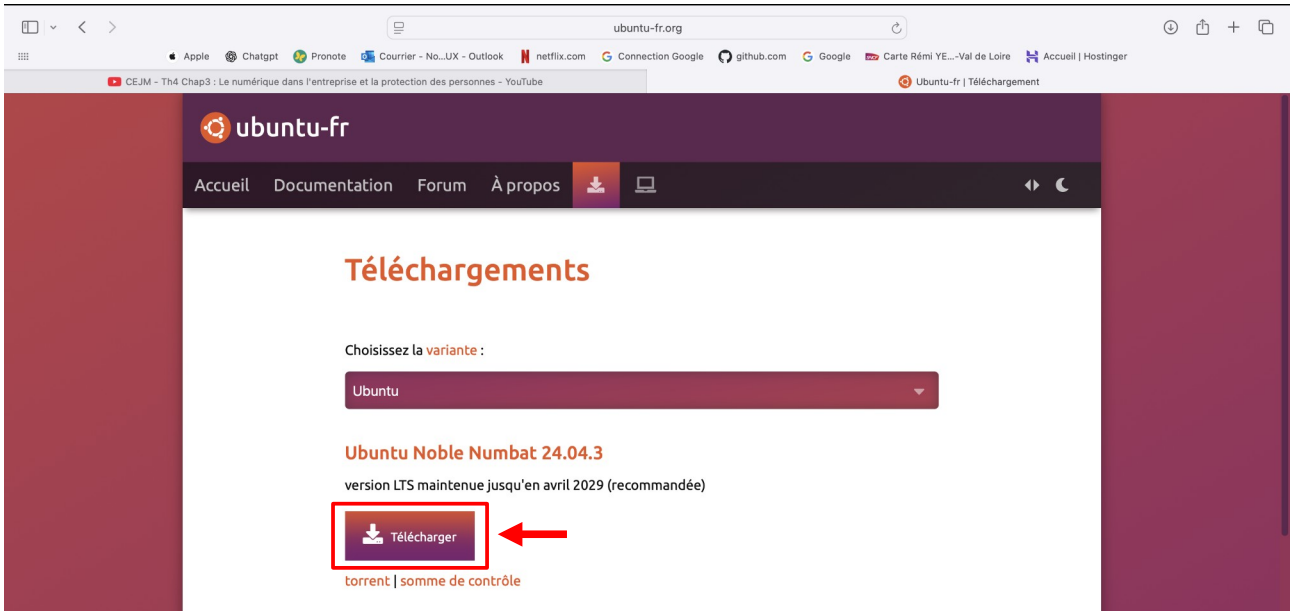
En posant ces trois critères, deux versions ressortent :
- **Ubuntu 22.04 LTS** ;
- **Ubuntu 24.04 LTS** ;

Ces deux versions sont dites **LTS** (**Long Term Support**), c’est-à-dire qu’elles bénéficient d’un support étendu de 5 ans. Elles sont suffisamment récentes (2022 et 2024) et permettent d’installer sans problème les logiciels nécessaires aux utilisateurs.

En tenant compte de tous ces éléments, la version la plus adaptée est **Ubuntu 24.04 LTS**, car elle est la plus récente, offre un support plus long dans le futur et intègre les dernières améliorations de sécurité.

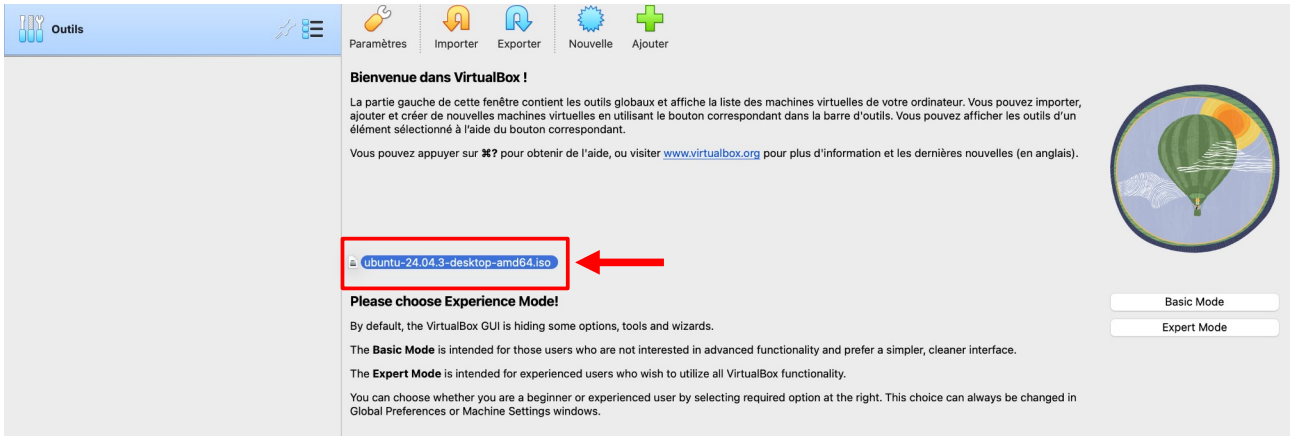
Etape 2 : téléchargé Ubuntu 24.04 LTS

Pour cela, aller sur ce lien `https://releases.ubuntu.com/noble/`, et cliqué sur ‘Desktop image – 64-bit PC (AMD64) desktop image’ pour l’installé.

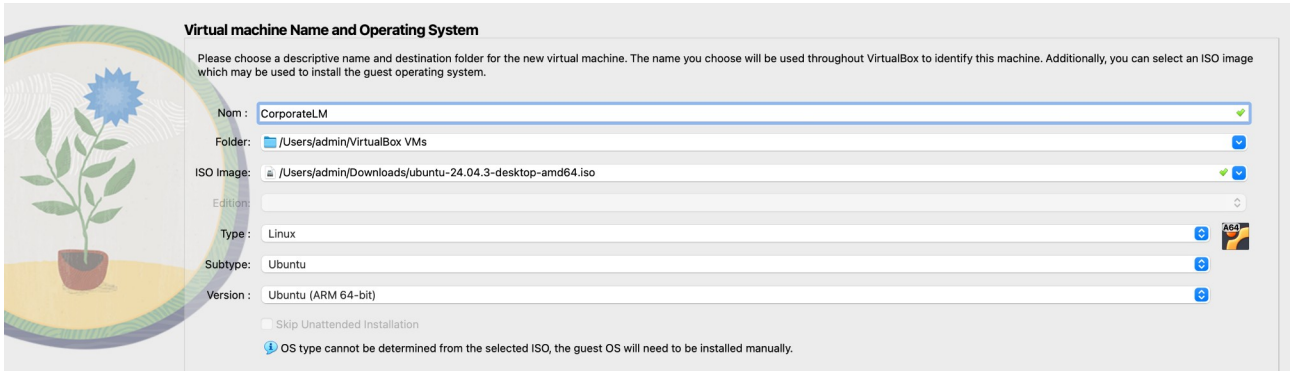


Etape 3 : installée sur VirtualBox

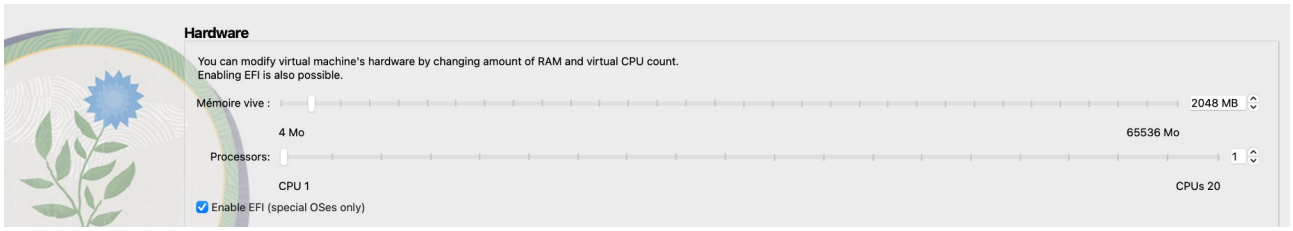
Après avoir téléchargé cette ISO, glisser déposer le fichier sur la fenêtre de VirtualBox.



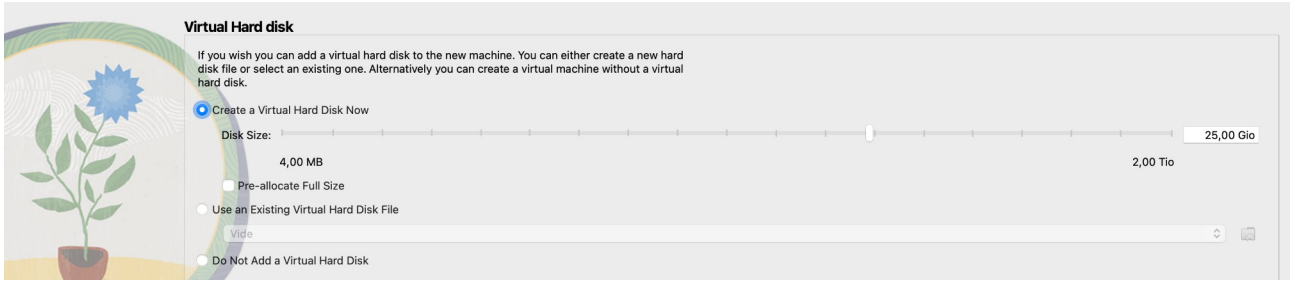
Cela ouvrira une fenêtre pour paramètre l’ISO, vous devrez entré le nom de la machine ainsi que le type d’OS si dessous. ATTENTION, vous devez coché la case `Skip Unattended Installation` (qui n’est pas dans la capture d’écran. Puis cliqué sur `suivant`.



Vous serez redirigé vers les paramètres hardware de la machine virtuelle, changé ces paramètres en fonction de vos besoins, mais comme ces postes seront destiné au développeur et qu'ils auront besoins de beaucoup de RAM, je conseillerai d'en mettre 8-16 pour le multitache.

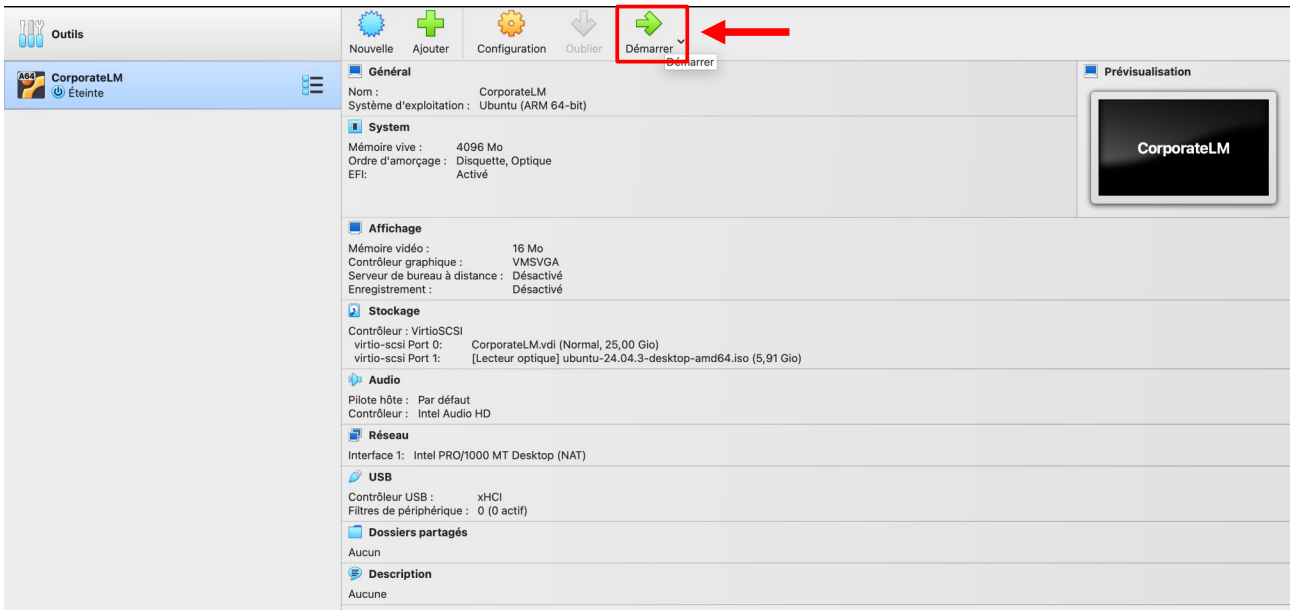


Enfin, vous serez redirigé vers l'espace de stockage nécéssaire pour cette machine, pour des développeur, pas plus de ce qui est conseiller par **VirtualBox** car la taille **Visual Studio Code** est de 400 Mo et le code ne sera pas stocké sur la machine mais dans le cloud grâce a **GitHub**.



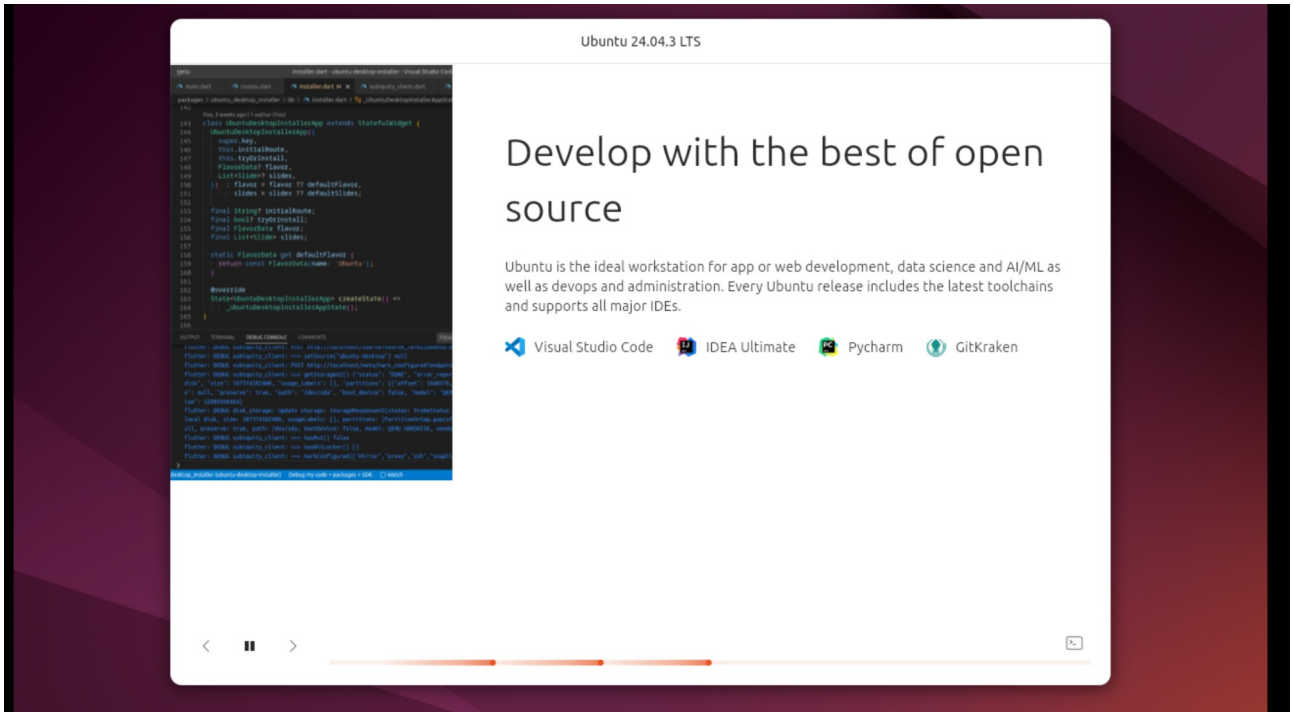
Etape 4 : lancé Ubuntu et faire le programme d’installation

Une fois **Ubuntu 24.04 LTS** installé sur `**Oracle VirtualBox**`, lancé le via le bouton `Démarrée`, cela devrait lancé une fenêtre de l’image d’**Ubuntu**.

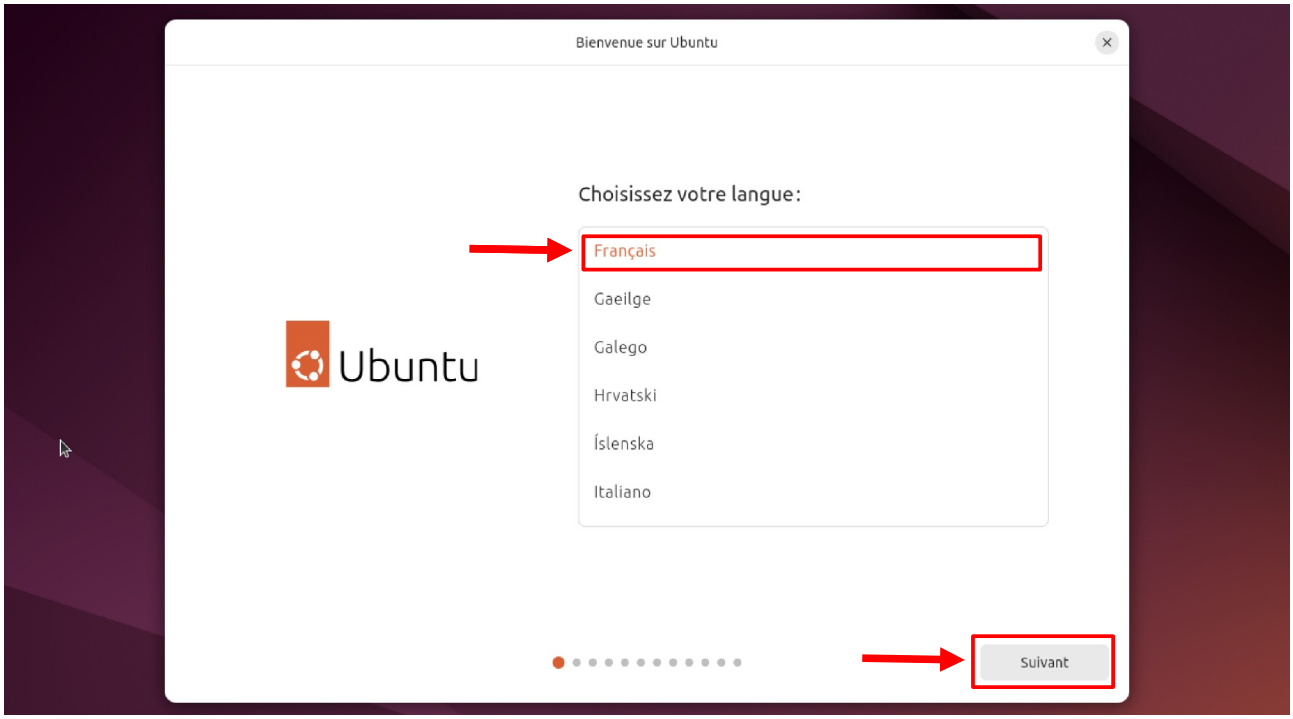


Laissé la machine virtuelle ce lancé toutes seul, puis, après plusieurs minutes, vous devriez voir cette écran :

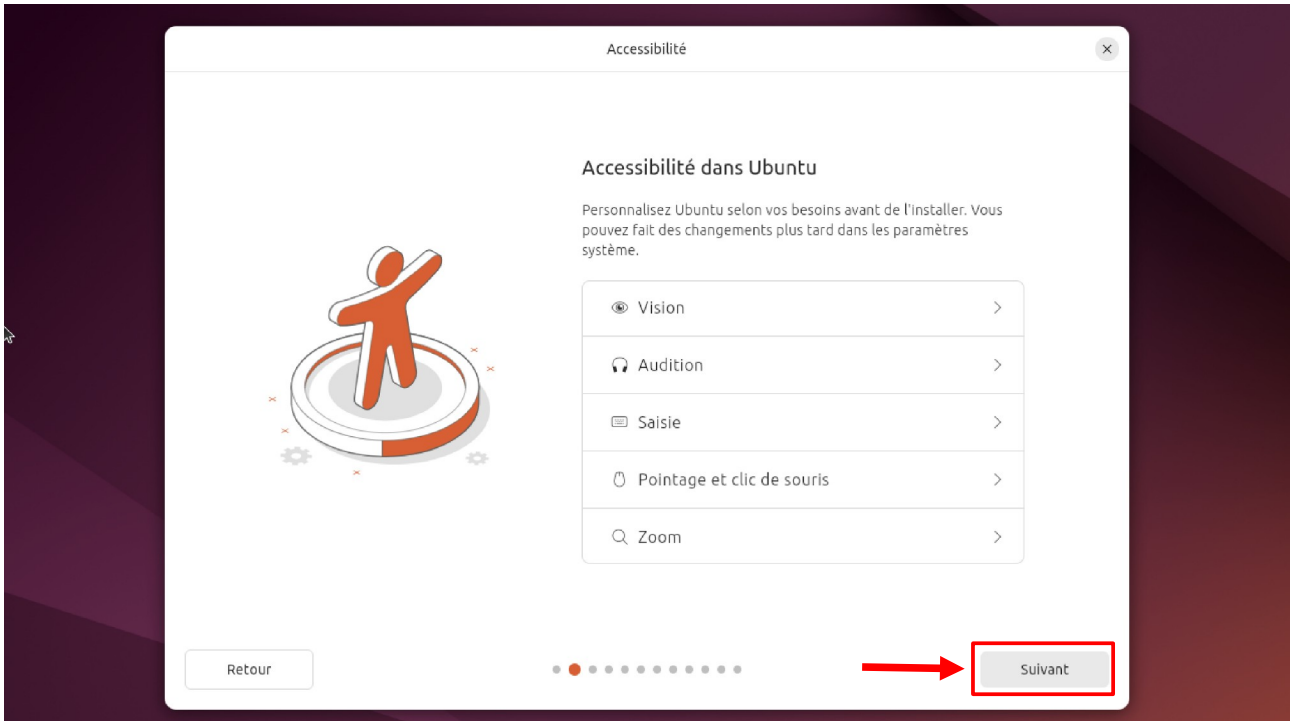
Cela signifie que Ubuntu est entrain de ce téléchargé pour que vous puissiez enfin arrivé a cette



fenêtre pour configurer Ubuntu, sélectionné le `Français` puis sur `Suivant`.



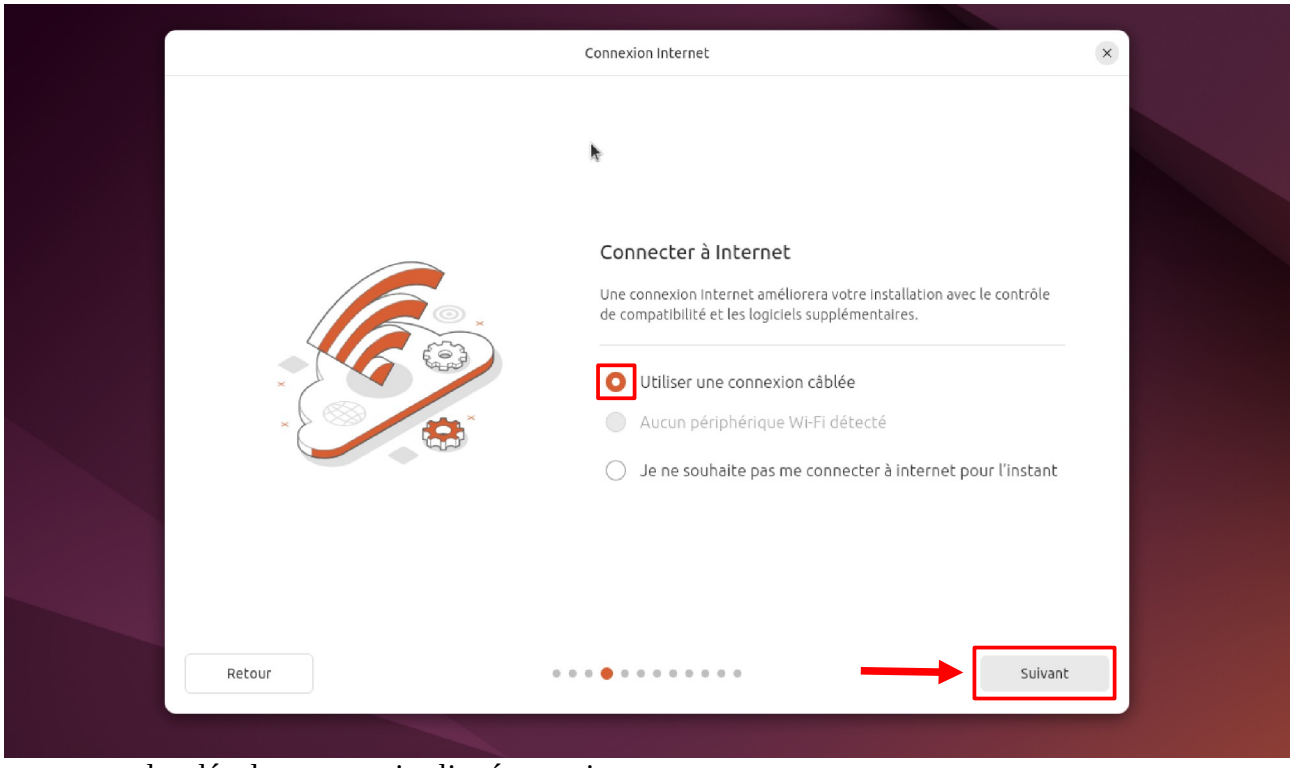
Puis, vous arrivez sur `Accessibilité dans Ubuntu`, cliqué sur `Suivant`.



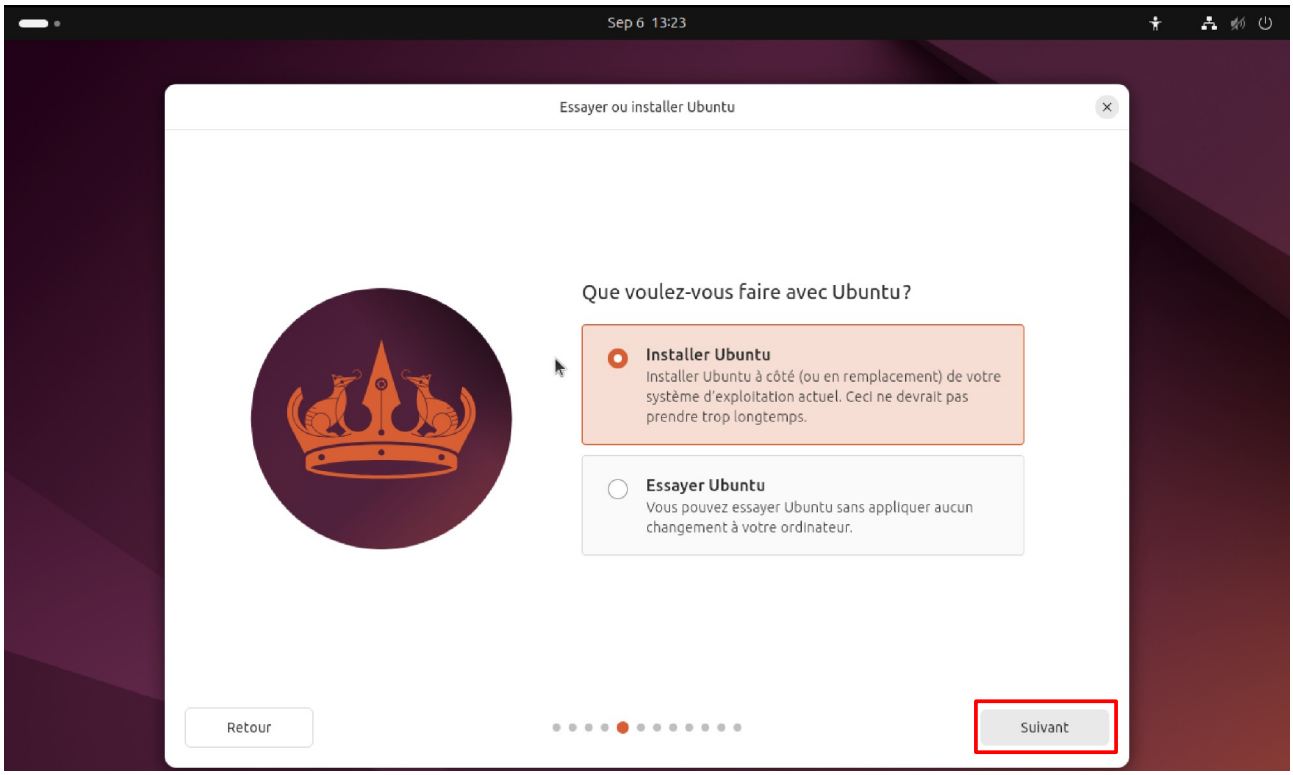
Dans `Sélectionnez la disposition de votre clavier`, sélectionné `Français`, puis `Suivant`.

Dans `Connecter à internet`, sélectionner `Utiliser une connexion cablé` pour installé les driver, logiciel, … etc. Puis, cliqué sur `Suivant`.

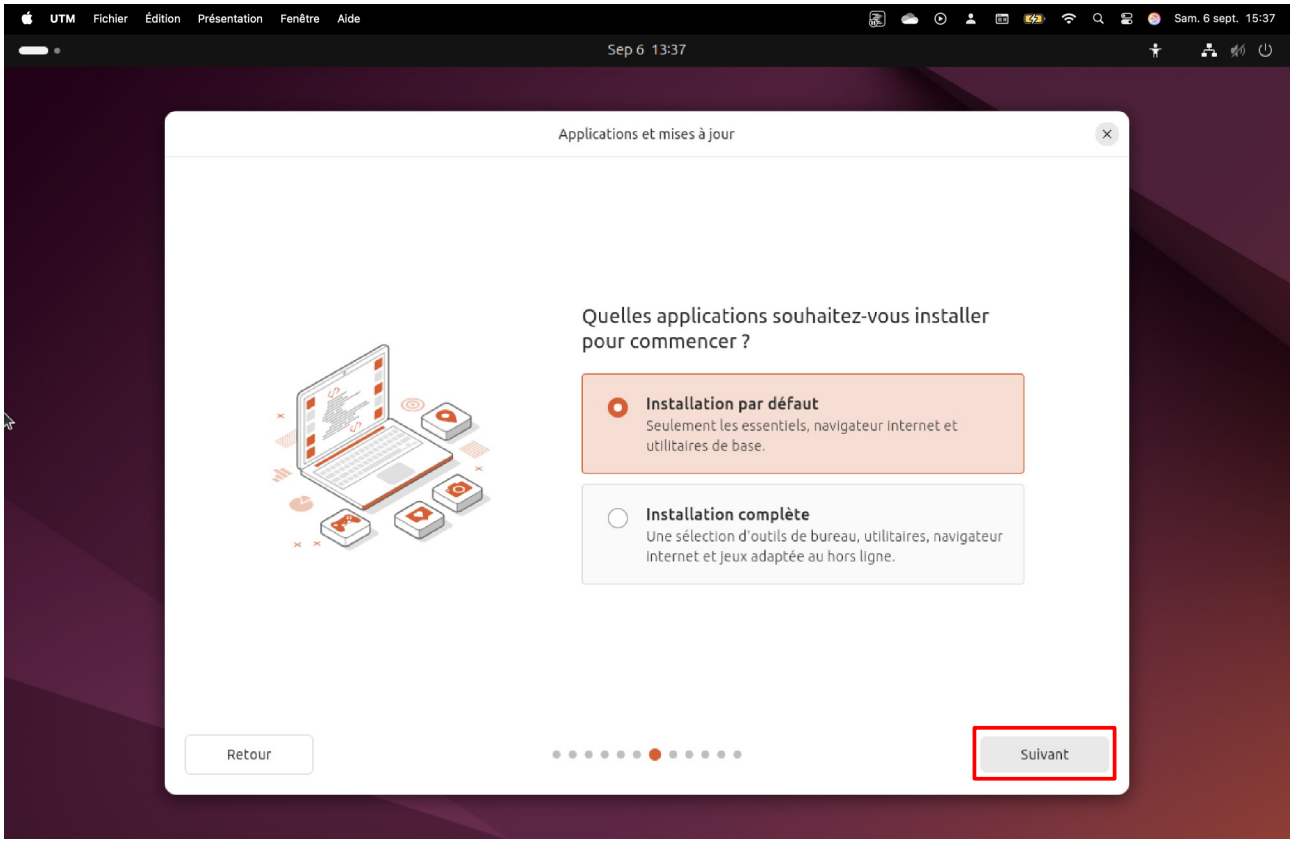
Dans `Que voulez-vous faire avec Ubuntu ?`, sélectionner `Installer Ubuntu` pour avoir un vrai



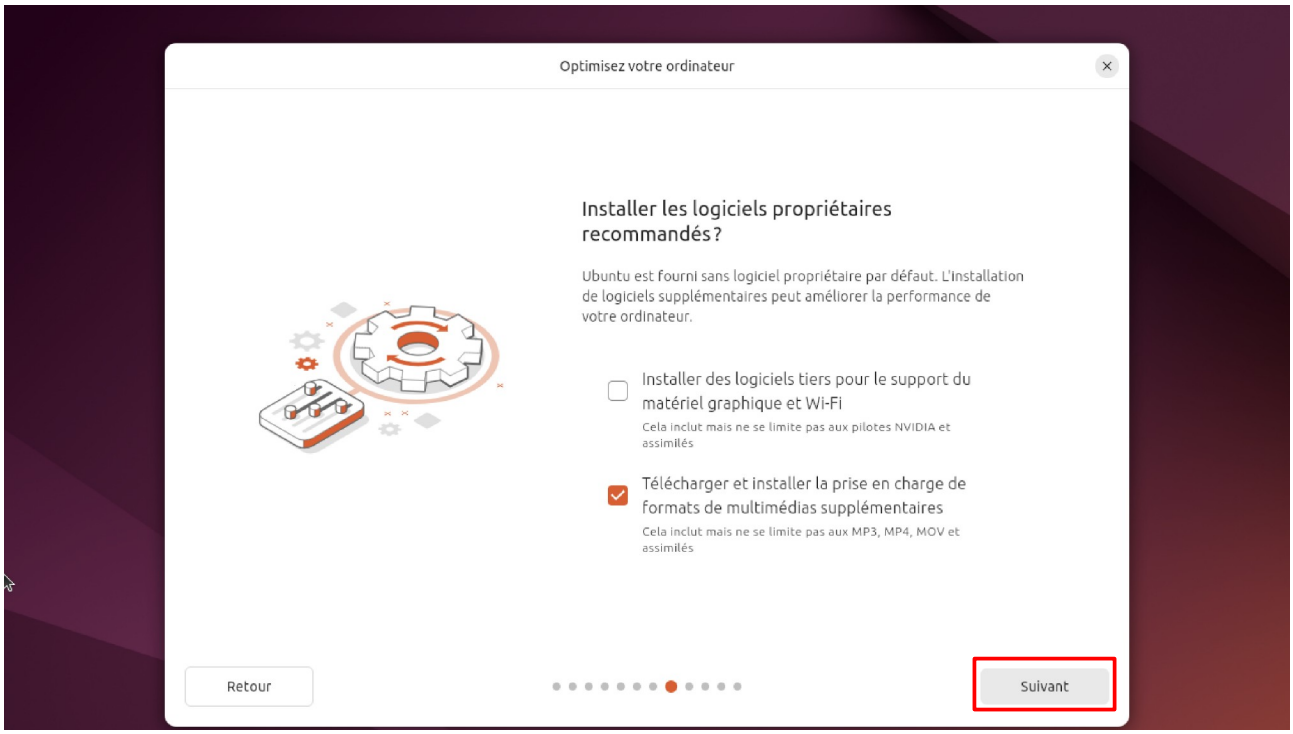
poste pour les développeur, puis cliqué sur suivant.



Dans `Quelles applications souhaitez-vous installer pour commencer ?`, sélectionner `Installation par défaut` pour installé a la main ce que les développeurs ont besoins comme logiciel, puis cliqué sur `Suivant`.

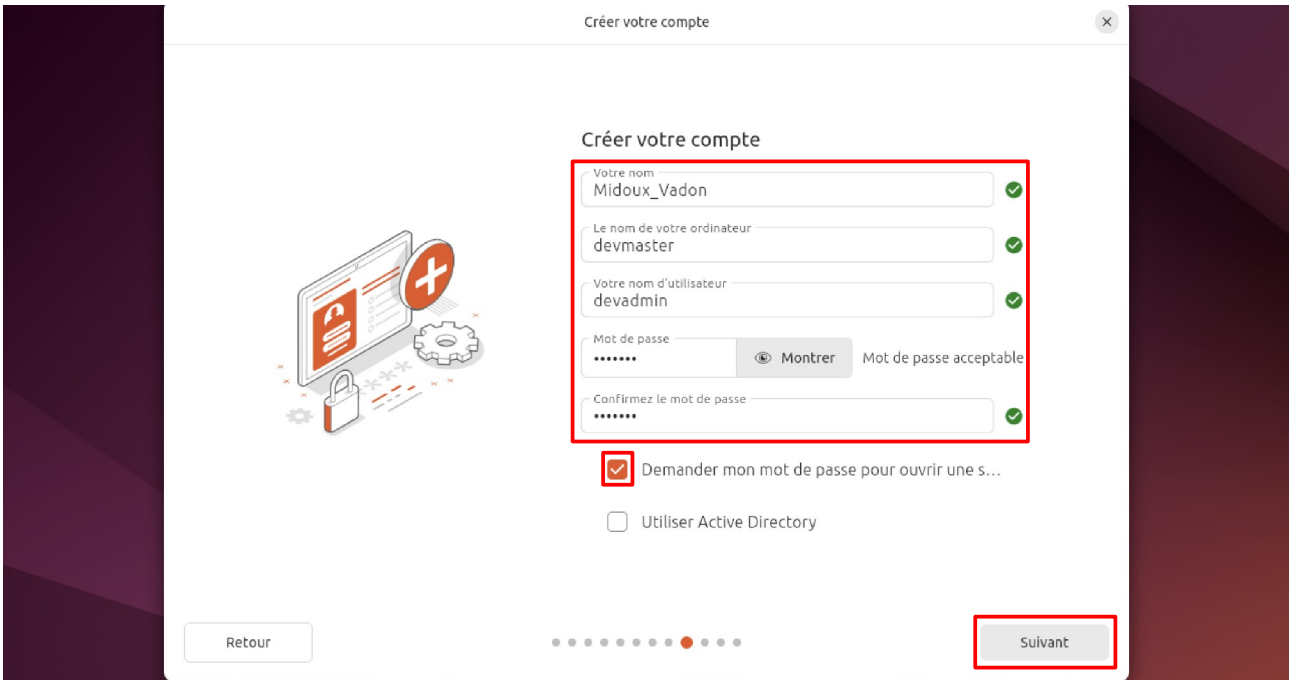


Dans `Installer les logiciels propriétaires recommandés ?`, sélectionner `Téléchargé et installer la prise en charge…` car nous n'auront pas besoins des pilotes NVIDIA vu que nous sommes sur VirtualBox pour l'instant, puis cliqué sur `Suivant`.

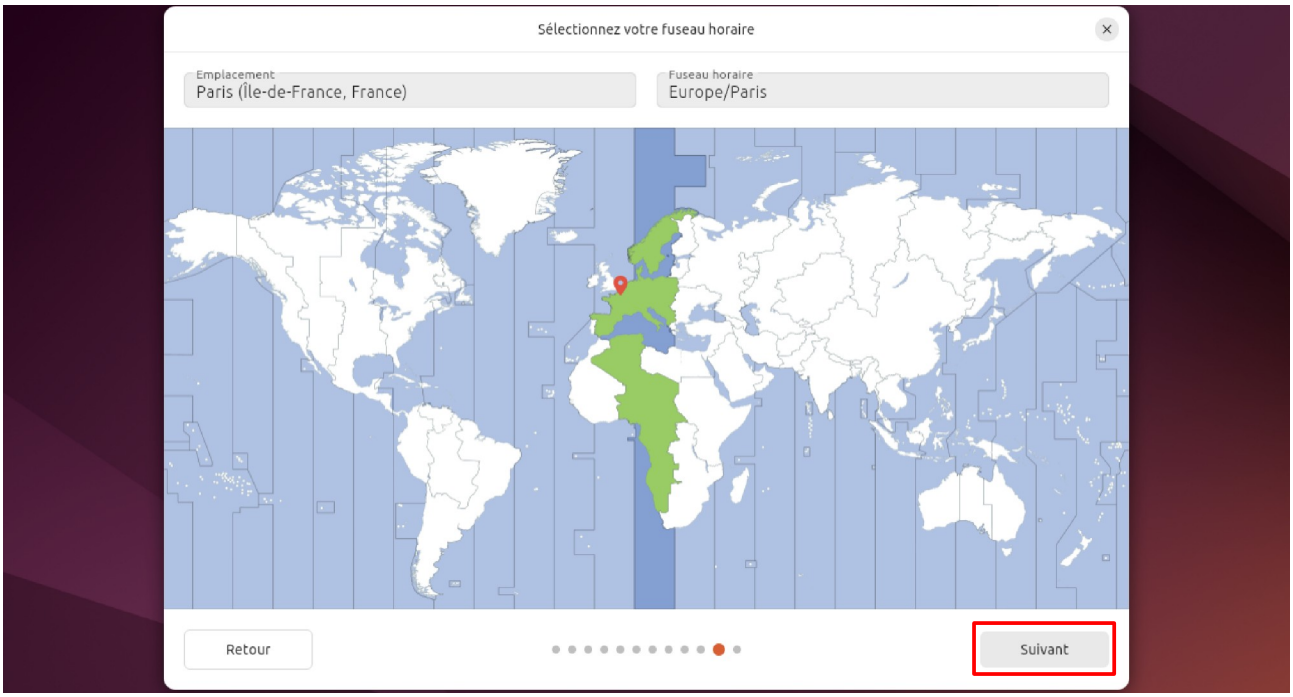


Dans `Créer votre compte`, entré les différents informations mis dans le cadre rouge ci-dessous. ATTENTION, il faut que vous choisissiez un mot de passe qui soit conforme a ces trois critères pour respecter la cybersécurité :
- longueur entre 12 et 16 caractères ;
- ne pas être utiliser autres part ;
- être complexe avec des lettres minuscules, lettres majuscules, chiffres et caractères spéciaux ;

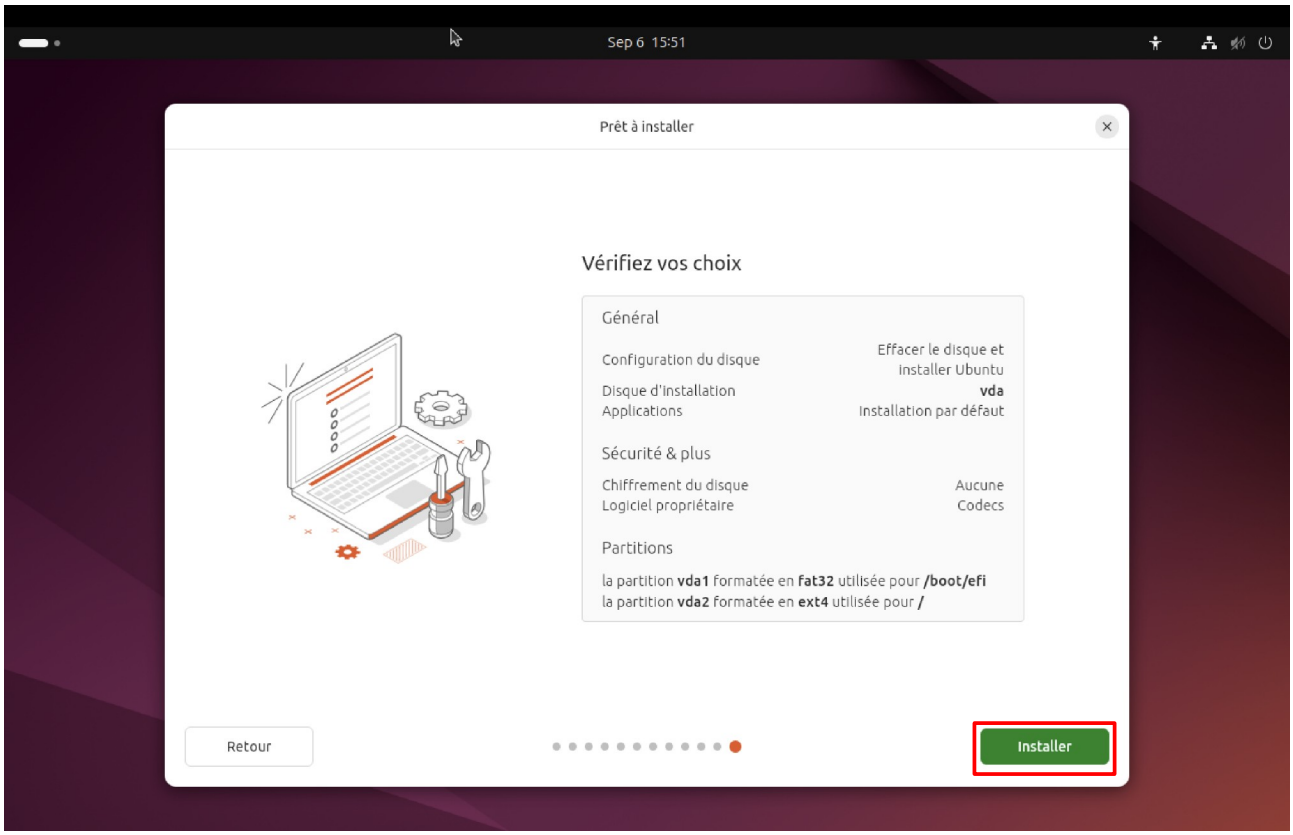
Après avoir tout compléter dans le premier carré rouge, coché `Demander mon mot de passe pour ouvrir …` mais ne coché pas `Utiliser Active Directory` car c'est un poste personnel et non un serveur. Puis cliqué sur `Suivant`.



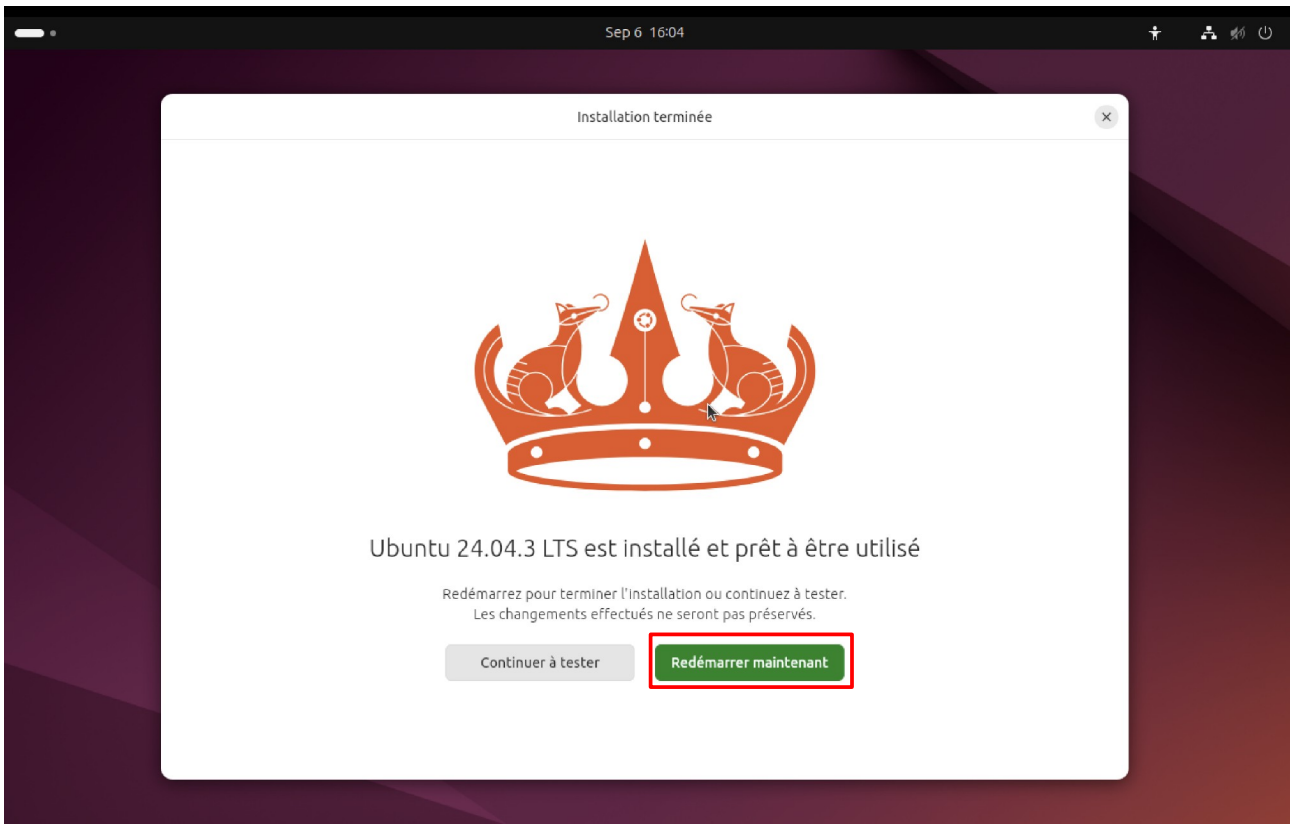
Dans `Sélectionnez votre fuseau horaire`, sélectionner le pays dans le quel vous êtes, puis appuyer sur `Suivant`.



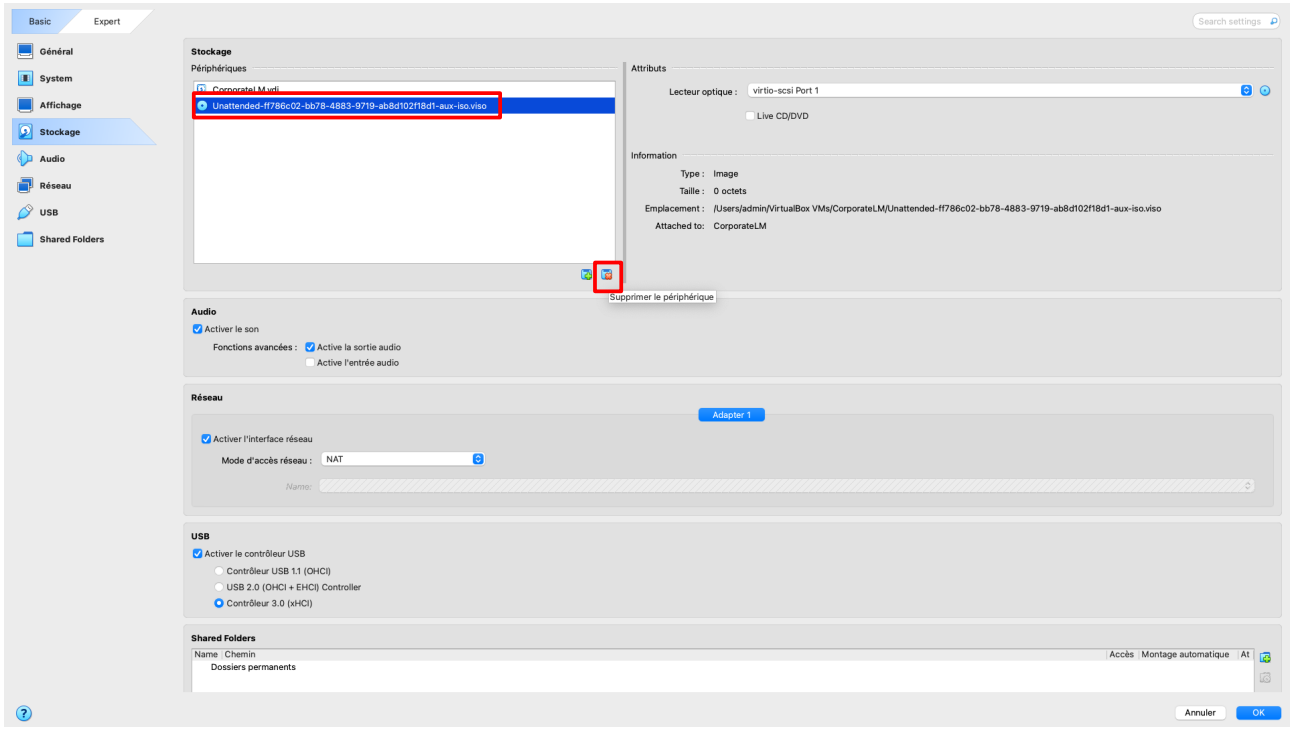
Enfin, quand vous êtes dans `Près à installer`, cliqué sur installé.



Après l'installation fini, **Ubuntu** vous demandera de Redémarré le poste, appuyer sur `Redémarrer maintenant`.



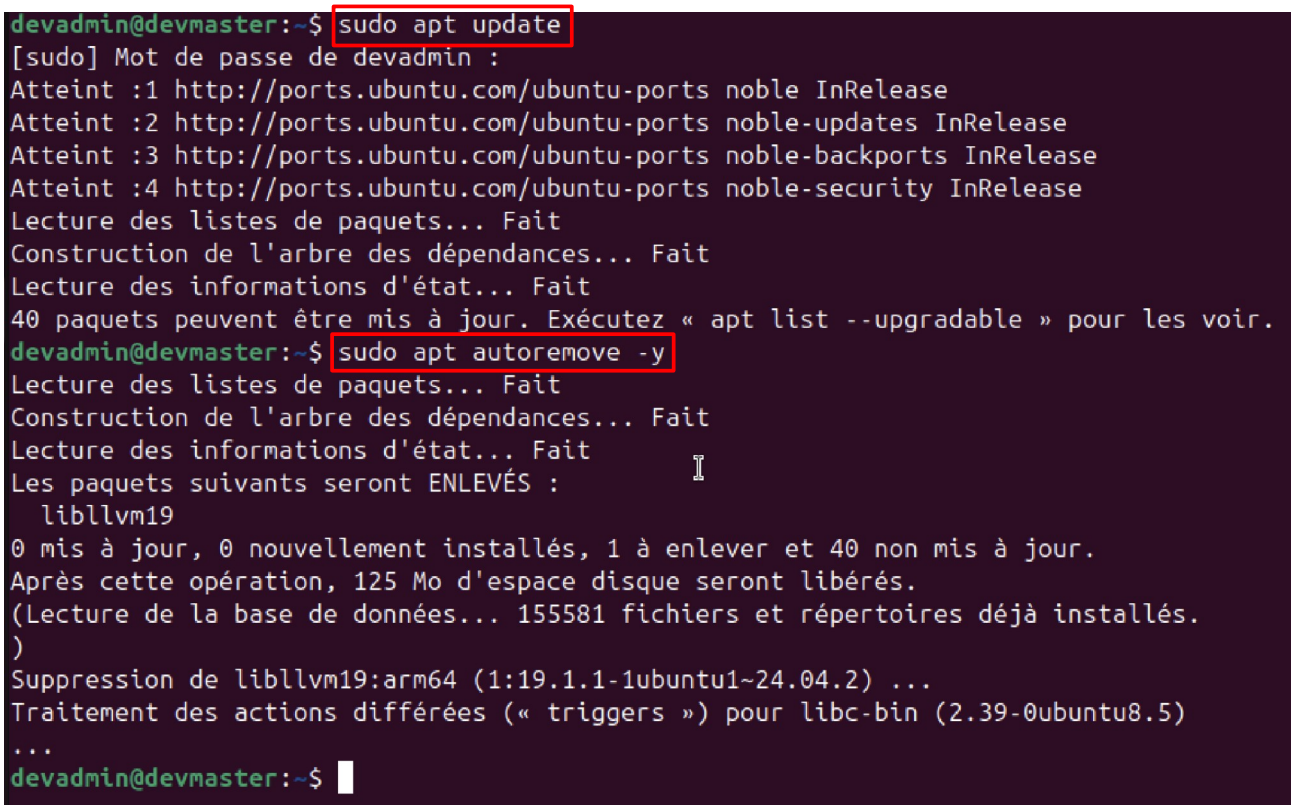
Une fois cela fait, vous devez éteindre la **VM** et aller dans les `paramètres > stockage` et vous devez supprimer l'ISO d'installation pour booter sur **Ubuntu**.



Puis, relancé votre machine pour vous connecter en temps qu'admin et vous voila maintenant sur Ubuntu.

Etape 5 : faire les mises a jour de sécurité du système

Pour ce faire, nous allons aller dans le `Terminal` d'**Ubuntu** pour exécuté plusieurs commande. Tout d'abord, exécuter cette commande `sudo apt update` tout en vous demandent votre mot de passe admin, cela va faire les différentes mise à jour. Puis, lancé cette commande `sudo apt autoremove -y` qui va supprime automatiquement les paquets devenus inutiles.



Etape 6 : créer un compte développeur et donné ces différents droits

Création d'un compte développeurs ainsi que le rôle

Le développeurs disposera donc des droits suivants :

- Coder et gérer leurs fichiers dans son répertoire personnel.

- Installer des paquets via `sudo apt install`, ce qui lui permet d'ajouter des dépendances nécessaires à ses projets sans solliciter en permanence l'administrateur.

- Lancer **Docker** ou des machines virtuelles, afin de tester son développement dans des environnements isolés et reproductibles.

Ce niveau de droits est un compromis entre autonomie et sécurité : les développeurs peuvent travailler efficacement, mais n'ont pas accès aux fichiers des autres utilisateurs ni aux paramètres critiques du système.

Voici les commande a executé : `sudo adduser DevCorporateLM` cela permet de créer un groupe de développement nommé `DevCorporateLM` a qui on va attribué ces droites et rôle .

Ensuite execute cette commande `sudo groupadd developers` qui va permettre de cérer un groupe de développement puis executer cette commande `sudo usermod -aG developers,sudo

```
devadmin@devmaster:~$ sudo adduser dev_corporate_lm
info: Ajout de l'utilisateur « dev_corporate_lm » ...
info: Choix d'un UID/GID dans la plage 1000 à 59999 ...
info: Ajout du nouveau groupe « dev_corporate_lm » (1001) ...
info: Ajout du nouvel utilisateur « dev_corporate_lm » (1001) avec le groupe « dev_corporate_lm » (1001) ...
info: Création du répertoire personnel « /home/dev_corporate_lm » ...
info: Copie des fichiers depuis « /etc/skel » ...
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd : mot de passe mis à jour avec succès
Modifier les informations associées à un utilisateur pour dev_corporate_lm
Entrer la nouvelle valeur, ou appuyer sur ENTER pour la valeur par défaut
        NOM []:
        Numéro de chambre []:
        Téléphone professionnel []:
        Téléphone personnel []:
        Autre []:
Ces informations sont-elles correctes ? [O/n] o
info: Ajout du nouvel utilisateur « dev_corporate_lm » aux groupes supplémentaires « users » ...
info: Ajout de l'utilisateur « dev_corporate_lm » au groupe « users » ...
```

dev_corporate_lm` pour mettre l'utilisateur dans le groupe.

```
devadmin@devmaster:~$ sudo groupadd developers
devadmin@devmaster:~$ sudo usermod -aG developers,sudo dev_corporate_lm
devadmin@devmaster:~$ groups dev_corporate_lm
dev_corporate_lm : dev_corporate_lm sudo users developers
```

Enfin `sudo usermod -aG developers,sudo dev_corporate_lm` qui permet de un, crée le compte avec un répertoire personnel, et de deux, il ajoute l'utilisateur aux groupes developers et au sudo qui donne des droits administratifs limités.

étape 7 : installé les logiciels nécessaires aux utilisateurs

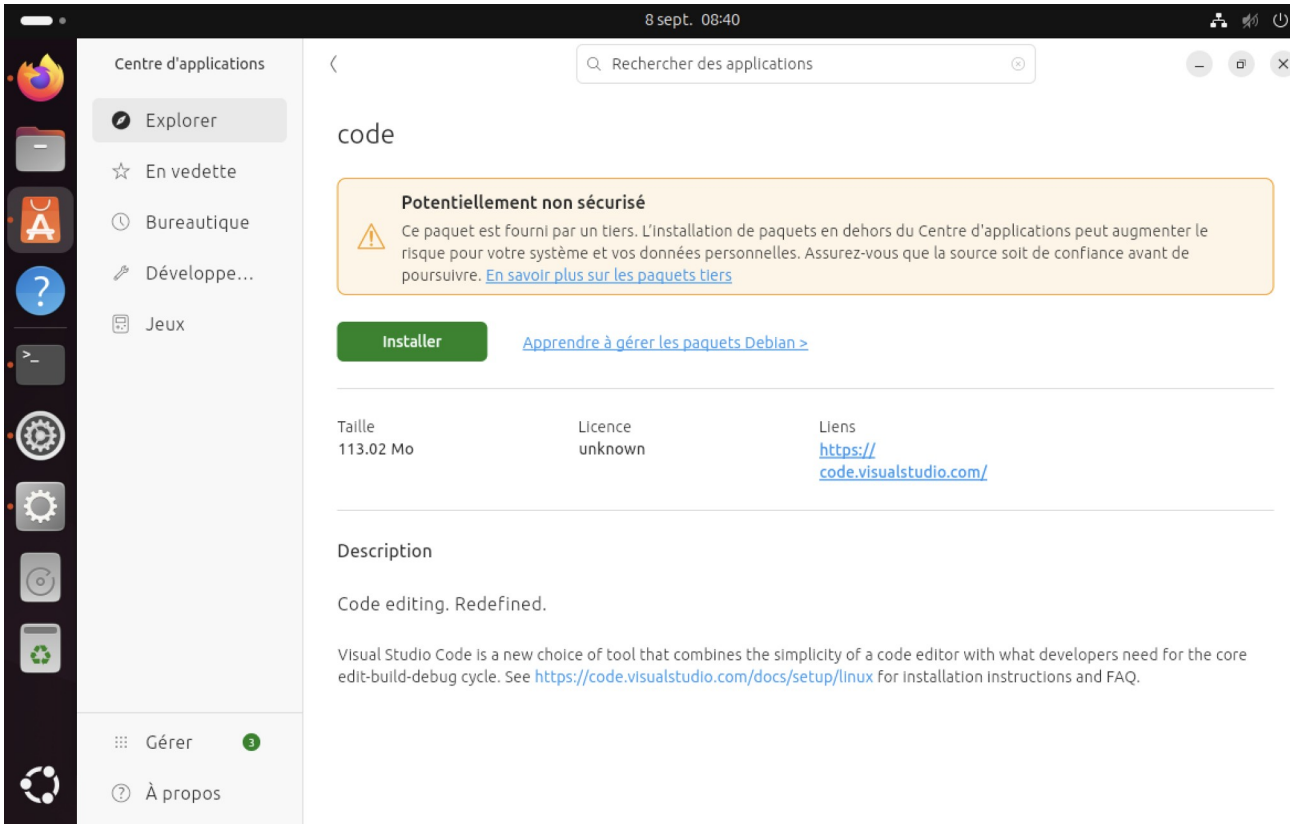
liste des outils a installé :

- Git ;
- VS Code ;
- GitHub Pull Requests ;
- GitLab Workflow ;
-
- Google Chrome ;
- SSH ;

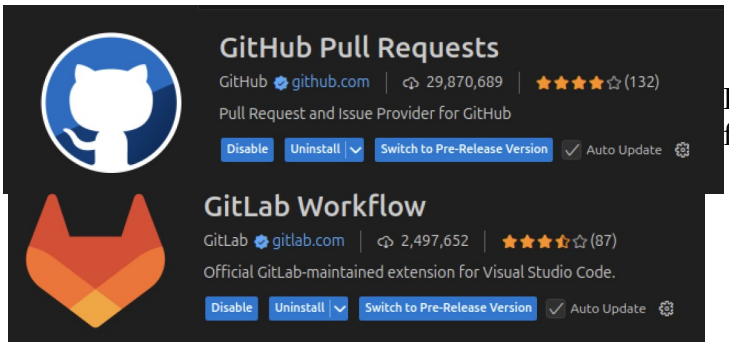
- Docker

- installation de **Git** depuis le terminal avec cette commande : `sudo apt install -y git`

- installé **Visual Studio Code** depuis le navigateur web (firefox) grâce a ce lien : `https://code.visualstudio.com/download` en sélectionnant l'option `.deb` pour la version **Ubuntu**. Puis, après avoir téléchargé le fichier, lancé le.

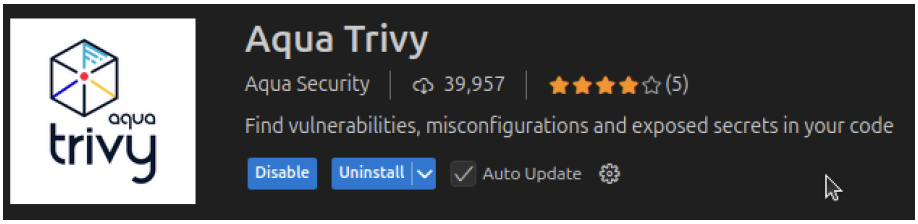


Une fois lancé, aller dans la partie `Extention` (logo constitué de 4 briques qui forme un carré) et tapé dans la barre de recherche **GitHub Pull Requests** et téléchargé l'extention.

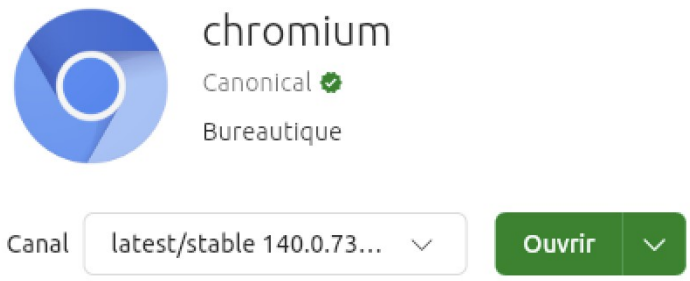


Ensuite, refaite la meme chose mais cette fois si pour **GitLab Workflow**.

Enfin, recherché **Aqua Trivy** et téléchargé le.



Une fois que toutes les extentions sont téléchargé, téléchargé **Chromium** via le centre d'appliction, cette application sera utile au les développeur pour tester sur les différents navigateurs web.



Pour terminé, téléchargé SSH via Terminal en éxécutant cette commande : `sudo apt install -y openssh-client`

```
devadmin@devmaster:~$ sudo apt install -y openssh-client
[sudo] Mot de passe de devadmin :
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
openssh-client est déjà la version la plus récente (1:9.6p1-3ubuntu13.13).
openssh-client passé en « installé manuellement ».
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
devadmin@devmaster:~$
```

étape 8 : configurer le réseau (pare-feu, antivirus, SSH, ...)

Après avoir installé les logiciels nécessaires, il est essentiel de sécuriser et configurer le poste de travail pour limiter les risques d'intrusion et garantir un environnement fiable. La sécurité passe par plusieurs points : pare-feu, gestion des accès SSH, mises à jour automatiques et protection contre les logiciels malveillants.

Pour ce faire, nous allons tout d'abord installé UFW qui permet de gérer facilement les règles réseau avec ces commande :

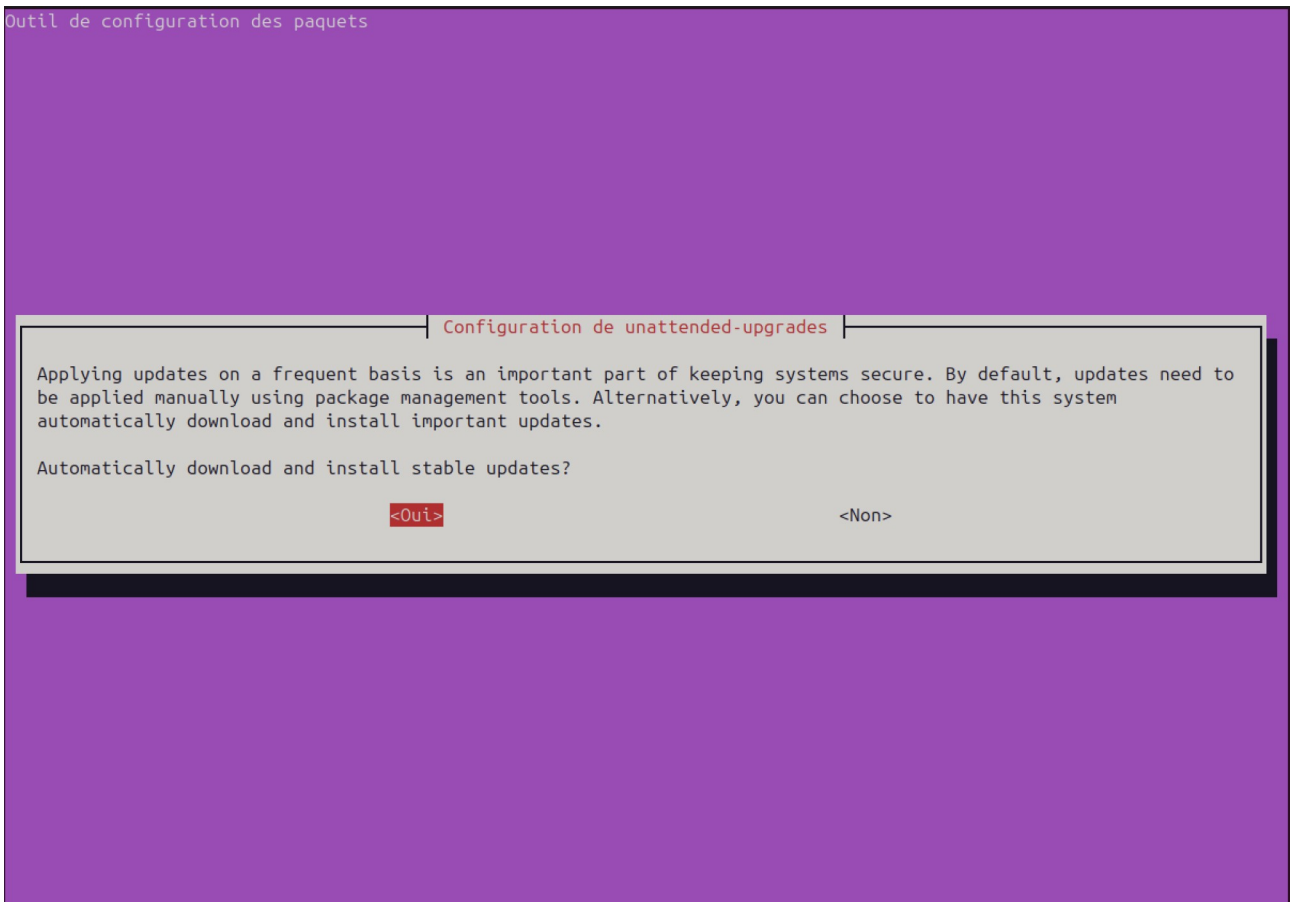
```
sudo apt install -y ufw
sudo ufw default allow outgoing
sudo ufw enable
```

```
devadmin@devmaster:~$ sudo apt install -y ufw
[sudo] Mot de passe de devadmin :
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
ufw est déjà la version la plus récente (0.36.2-6).
ufw passé en « installé manuellement ».
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
devadmin@devmaster:~$ sudo ufw default allow outgoing
La stratégie par défaut pour le sens « outgoing » a été remplacée par « allow »
(veillez à mettre à jour vos règles en conséquence)
devadmin@devmaster:~$ sudo ufw enable
Le pare-feu est actif et lancé au démarrage du système
devadmin@devmaster:~$
```

Puis, nous allons sécurisé l'installation du protocole SSH via cette commande :
`sudo apt install -y openssh-client`.

Ensuite, téléchargé l'antivirus ClamAV qui est recommander via cette commande :
```
sudo apt install -y clamav clamav-daemon
sudo systemctl enable clamav-freshclam
sudo systemctl start clamav-freshclam
```

Enfin, téléchargé les mises à jour automatiques de sécurité via cette commande :
```
sudo apt install -y unattended-upgrades
sudo dpkg-reconfigure unattended-upgrades
```



---

bonus : comparatif entre GitHub et GitLab

GitHub est particulièrement adapté si l'objectif est de travailler sur des projets open source, de profiter d'une large communauté et d'une intégration simplifiée avec Visual Studio Code.

GitLab est plus approprié pour les entreprises qui souhaitent un contrôle total sur leurs données, une intégration native de la chaîne DevOps et des fonctionnalités de sécurité avancées.

Critère	GitHub	GitLab
Hébergement	SaaS, Enterprise payant	SaaS, auto-hébergement (Community gratuite)
Collaboration	Excellente, orientée open source	Bonne, orientée entreprise
CI/CD	GitHub Actions (récent, efficace)	GitLab CI/CD (très complet)
Sécurité	Dependabot, code scanning	SAST, DAST, conteneurs, sécurité avancée
Intégration IDE	Native avec VS Code	Extension VS Code disponible
Communauté	Très large (open source)	Moins vaste, usage privé et entreprise

Même si GitHub et GitLab offrent des fonctionnalités avancées pour la gestion de versions et la collaboration, il est important de préciser que, dans leur version SaaS (cloud), les deux solutions sont hébergées aux États-Unis.

Cela signifie que les données des utilisateurs et des projets peuvent potentiellement être soumises aux lois américaines, telles que le Patriot Act ou le Cloud Act, qui autorisent les autorités américaines à accéder à certaines données stockées sur des serveurs situés aux États-Unis, même si les utilisateurs sont en Europe.

Dans le cadre du RGPD (Règlement Général sur la Protection des Données), cette situation pose un problème de conformité pour les entreprises européennes, en particulier lorsqu'il s'agit de données personnelles sensibles.
Ainsi, bien que GitHub et GitLab soient techniquement performants, leur utilisation en version cloud peut présenter des risques juridiques et de confidentialité pour une entreprise française comme Corporate LM.

Pour pallier ce problème, deux solutions existent :

- GitHub Enterprise Server : une version auto-hébergée de GitHub, installée dans l'infrastructure de l'entreprise.
- GitLab Community ou Enterprise Edition : qui peut être installée directement sur les serveurs internes de l'entreprise, garantissant ainsi que les données restent hébergées en Europe et sous le contrôle de Corporate LM.