

Installation d'un environnement de développement (client)

Sommaire :

1. Choix de la version de Ubuntu :.....	3
1. Support à long terme et stabilité	3
2. Sécurité renforcée	3
3. Compatibilité et écosystème riche	3
4. Facilité d'utilisation	4
5. Performances et optimisation	4
6. Usage polyvalent	4
7. Mises à jour de noyau et compatibilité matérielle	4
2. Installation de Ubuntu :	5
1. Présentation de GitHub :	10
1.1. Avantages de GitHub :	10
1.2. Inconvénients de GitHub :	10
2. Présentation de GitLab :	11
2.1. Avantages de GitLab :	11
2.2. Inconvénients de GitLab :	11
3. Choix de GitHub ou GitLab :	12
1. Plateforme DevOps complète :	11
2. Open-source et auto-hébergé :	11
3. Automatisation avec CI/CD intégrée :	11
4. Gestion des permissions et sécurité :	11
5. Planification de projet :	12
6. Collaboration améliorée :	12
7. Flexibilité et extensibilité :	12
8. Communauté active et support professionnel :	12
9. Scalabilité :	12

10. Intégrations natives avec des outils de cloud :	12
4.Installation de Visual Studio Code :.....	14
Fonctionnalités principales de Trivy :.....	24
5.Les bonnes pratiques de sécurité sur un poste de travail :	25

1. Choix de la version de Ubuntu :

Choisir Ubuntu 24.04 LTS (Long Term Support) présente plusieurs avantages qui peuvent répondre à divers besoins, que ce soit pour les utilisateurs individuels, les entreprises, ou les développeurs. Voici pourquoi Ubuntu 24.04 LTS est souvent recommandé :

1. Support à long terme et stabilité

- **Support prolongé :** La version LTS d'Ubuntu bénéficie d'un support de cinq ans, incluant des mises à jour de sécurité et des correctifs de bugs. Cela garantit une stabilité à long terme pour les environnements de production, sans avoir à effectuer de mises à niveau fréquentes.
- **Mises à jour stables :** Les versions LTS sont plus rigoureusement testées que les versions non LTS, ce qui réduit les risques de bugs majeurs.

2. Sécurité renforcée

- **Patches réguliers :** Ubuntu LTS reçoit des mises à jour de sécurité critiques pendant toute la durée de son support. Cela en fait un bon choix pour les serveurs ou les environnements sensibles.
- **Facilité de gestion des vulnérabilités :** Grâce à des outils comme Ubuntu Advantage et Landscape, il est facile de gérer la sécurité à grande échelle, en particulier dans des environnements professionnels.

3. Compatibilité et écosystème riche

- **Large compatibilité logicielle :** Ubuntu est l'une des distributions Linux les plus populaires, ce qui signifie qu'elle est largement prise en charge par les éditeurs de logiciels, les pilotes matériels, et les plateformes cloud.
- **Disponibilité des paquets :** Les versions LTS incluent un large éventail de logiciels à jour via des dépôts sécurisés, garantissant l'accès à des versions stables et fiables des logiciels les plus courants.

4. Facilité d'utilisation

- Interface utilisateur intuitive : Ubuntu offre une interface GNOME simple et agréable à utiliser, même pour les utilisateurs moins expérimentés.
- Communauté et documentation : En raison de la popularité d'Ubuntu, il existe une vaste communauté et une documentation riche, facilitant la résolution de problèmes et l'apprentissage pour les débutants et les experts.

5. Performances et optimisation

- Performance améliorée : Avec chaque nouvelle version, Canonical (l'entreprise derrière Ubuntu) continue d'optimiser les performances du système, ce qui inclut une meilleure gestion des ressources et des temps de démarrage plus rapides.
- Optimisation pour le cloud et les conteneurs : Ubuntu LTS est souvent le choix par défaut dans les environnements cloud, grâce à ses performances, sa fiabilité, et sa compatibilité avec les outils modernes comme Docker, Kubernetes et les environnements de virtualisation.

6. Usage polyvalent

- Pour les postes de travail et serveurs : Que ce soit pour un usage personnel, de développement ou pour déployer des serveurs en production, Ubuntu LTS s'adapte facilement à ces contextes. Son faible coût d'entretien le rend très prisé dans les environnements professionnels.

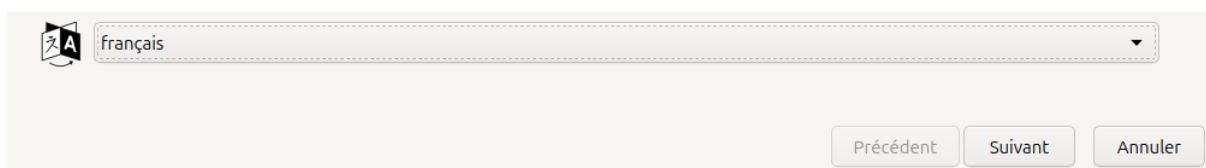
7. Mises à jour de noyau et compatibilité matérielle

- Ubuntu 24.04 LTS bénéficiera des dernières versions du noyau Linux, offrant des améliorations matérielles, de sécurité, et de performance.
- Améliorations matérielles : La version LTS garantit également une compatibilité accrue avec le matériel moderne, qu'il s'agisse de processeurs récents ou de nouvelles architectures.

2. Installation de Ubuntu :

Lors du lancement de la machine virtuelle, vous serez accueilli par l'écran de démarrage d'**Ubuntu 24.04 LTS**.

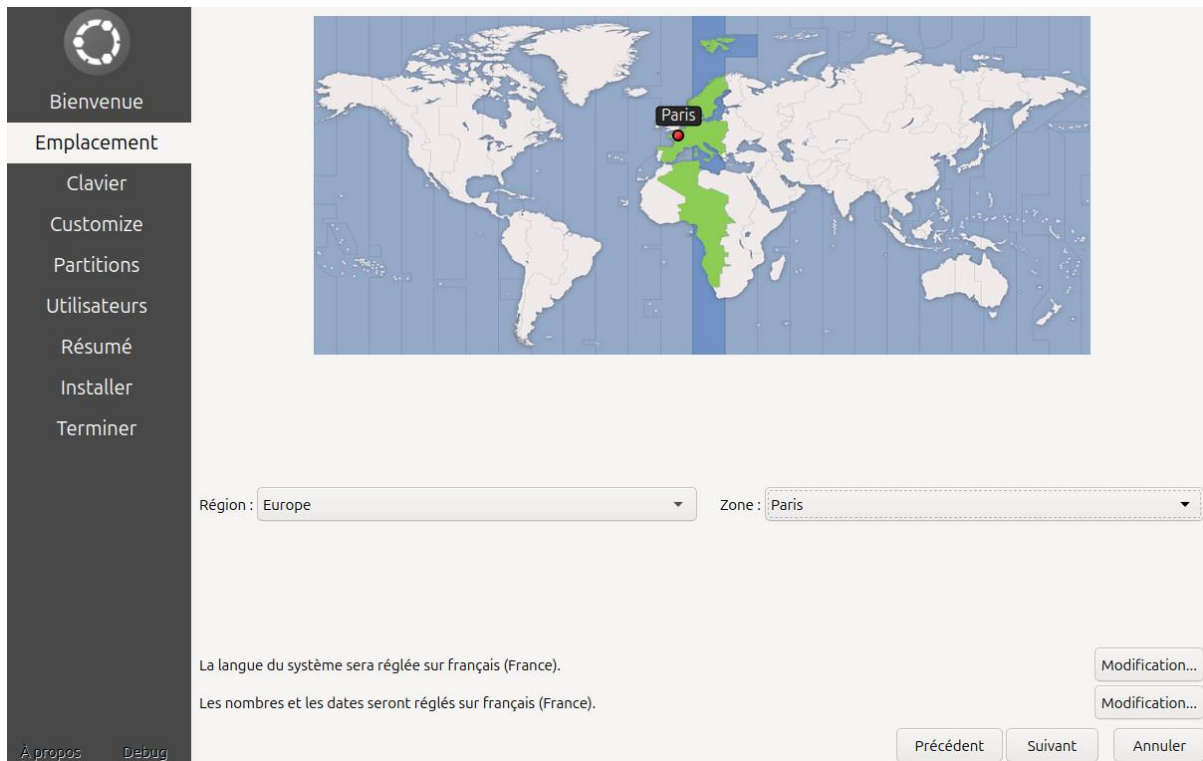
Cliquez sur l'icône « **Install Ubuntu 24.04 LTS** » située sur le bureau pour démarrer l'installation.



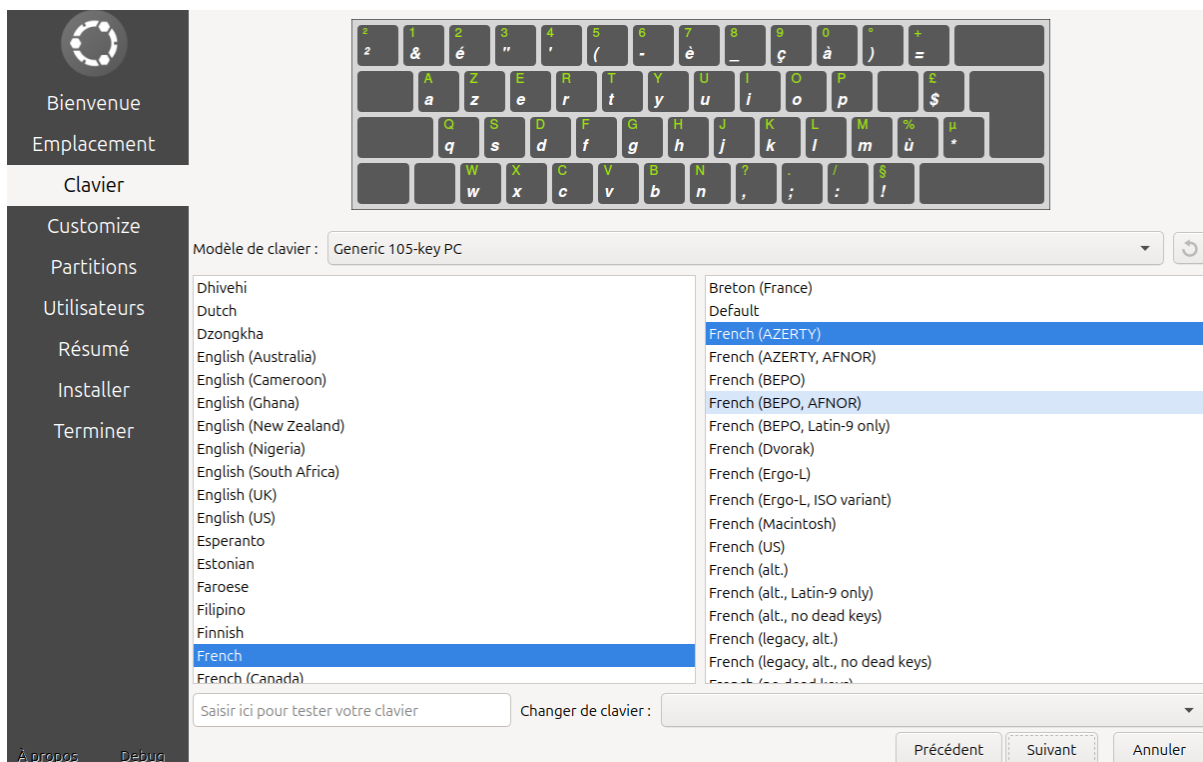
L'écran de bienvenue de l'installateur s'affichera avec l'option de choisir la langue d'installation.

Par défaut, la langue « **français** » est sélectionnée, mais vous pouvez choisir une autre langue dans la liste déroulante si nécessaire.

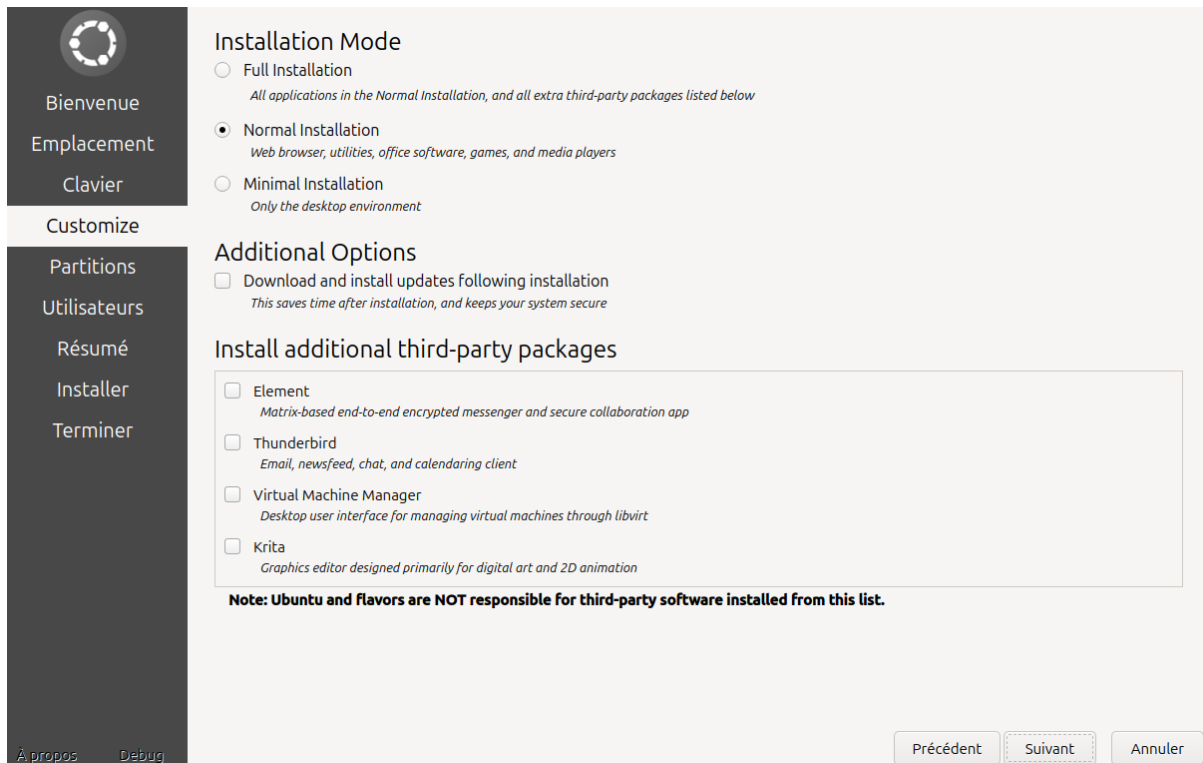
Cliquez sur « **Suivant** » pour continuer.



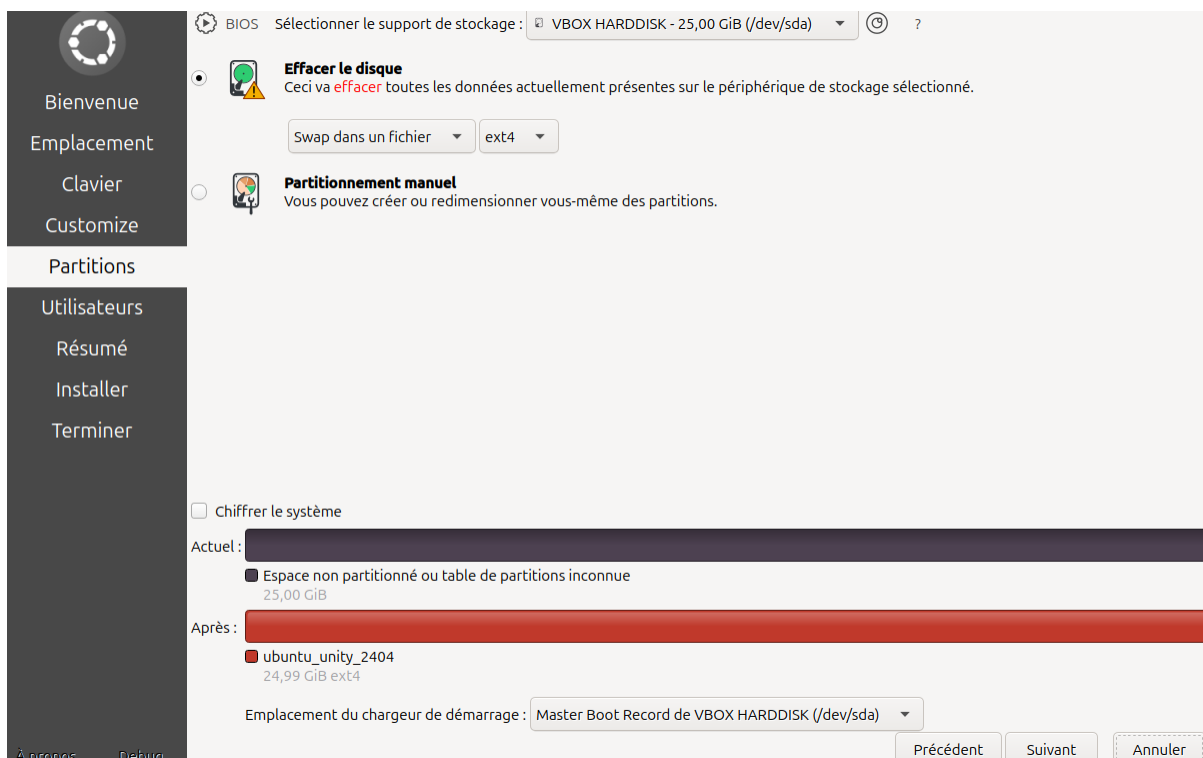
Emplacement : Sélectionnez votre fuseau horaire (Région : Europe, Zone : Paris).



Clavier : Choisissez votre configuration de clavier (AZERTY, QWERTY, etc.).



Cliquez sur **Normal Installation**.



Lors de l'étape de configuration des partitions, vous serez invité à choisir où et comment installer Ubuntu.

Il existe plusieurs options disponibles à ce stade, mais pour effacer entièrement le disque et faire une nouvelle installation, procédez comme suit :

- Choisissez l'option "**Effacer le disque et installer Ubuntu**".
- Cette option supprimera toutes les données existantes sur le disque et installera Ubuntu Unity à **partir de zéro**.
- Ensuite, sélectionnez le disque à utiliser si vous avez plusieurs disques connectés.

Vous recevrez un message de confirmation indiquant que toutes les données du disque seront effacées. Cliquez sur "**Suivant**" pour confirmer.

Continuez avec la création de l'utilisateur et le reste des configurations.

Quel est votre nom ?
olivier ✓

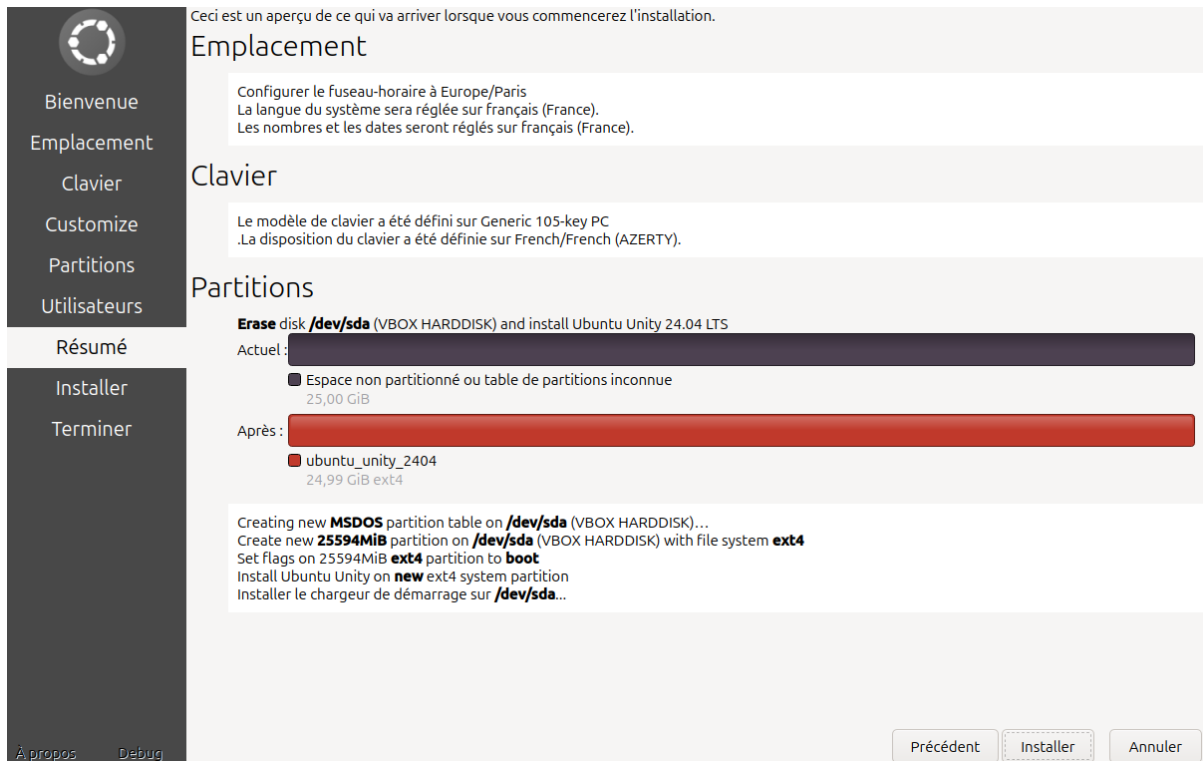
Quel nom souhaitez-vous utiliser pour la connexion ?
olivier ✓

Quel est le nom de votre ordinateur ?
olivier-virtualbox ✓

Veuillez saisir le mot de passe pour sécuriser votre compte.
..... ✓

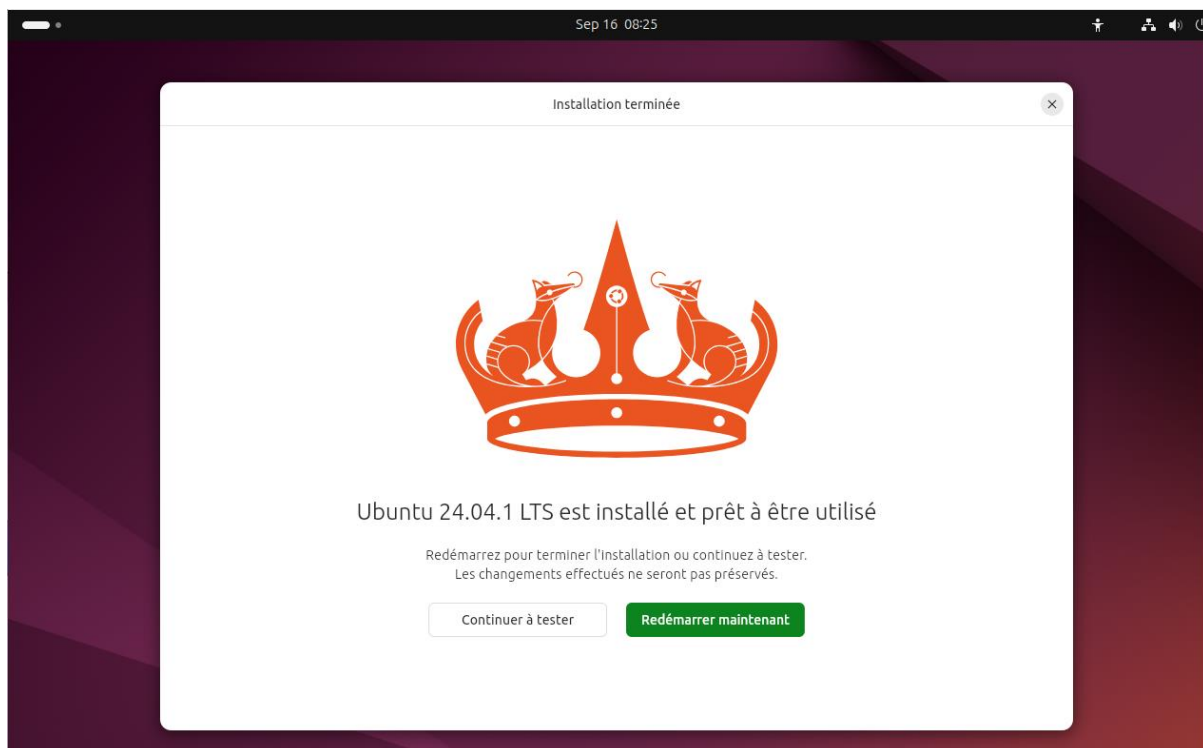
☐ Démarrer la session sans demander de mot de passe.

Précédent Suivant Annuler



Cliquez sur "**Installer**" pour commencer le processus d'installation. Cela va :

- Formater le disque.
- Installer Ubuntu sur l'ensemble du disque dur.



Une fois l'installation terminée, **redémarrez la machine** et commencez à utiliser Ubuntu.

1. Présentation de GitHub :



GitHub est une plateforme en ligne conçue pour l'hébergement, le développement et la gestion de projets de logiciels, particulièrement ceux utilisant le contrôle de version avec Git. Git est un système de contrôle de version distribué qui permet de suivre les modifications de fichiers et de collaborer facilement sur des projets, particulièrement pour les développeurs de logiciels.

GitHub fournit une interface web qui permet aux utilisateurs de naviguer, consulter le code source, commenter les lignes de code. Il propose également une intégration avec des services de CI/CD (Intégration Continue/Déploiement Continu).

1.1. Avantages de GitHub :

- **Plateforme open-source** : Plateforme majeure pour les projets open-source,
- **Pull requests et révisions de code** : Collaboration efficace grâce aux pull requests et aux commentaires sur le code,
- **GitHub Actions** : Automatisation des workflows de développement avec des pipelines intégrés,
- Les développeurs peuvent partager leurs projets de façon publique,
- Sur GitHub, toutes les modifications apportées à un projet sont sauvegardées dans un « changelog »,
- Les collaborateurs peuvent travailler ensemble sans se gêner mutuellement. Chacun peut voir et savoir ce que les autres font en temps réel,
- **Pages GitHub et Gists** : Hébergement gratuit de sites web statiques et partage facile d'extraits de code,
- **Documentation et collaboration** : Wikis et issues facilitent la gestion de la documentation et des discussions en équipe,
- Intégration avec des outils tiers.

1.2. Inconvénients de GitHub :

- Les dépôts privés sur GitHub nécessitent un abonnement payant,
- **Manque de personnalisation en auto-hébergement** : GitHub n'est pas conçu pour l'auto-hébergement, sauf avec GitHub Enterprise (coût élevé),
- **Fonctionnalités avancées payantes** : Les outils avancés (analyse, audit de sécurité) sont réservés aux plans payants,
- **Moins d'outils de gestion de projet** : Fonctionnalités de gestion de tâches limitées en comparaison avec GitLab.

2. Présentation de GitLab : GitLab

GitLab est une plateforme collaborative open source dédiée à la gestion de projets web. Git, conçu à l'origine par Linus Torvalds pour faciliter le développement du noyau Linux, permet de suivre l'historique des différentes versions d'un projet et simplifie grandement le travail en équipe. Grâce à GitLab, il devient possible d'héberger des projets web tout en gérant les versions des codes sources.

La plateforme se distingue par sa capacité à prendre en charge l'ensemble du processus de développement, du concept initial jusqu'à la production. Elle offre un environnement de travail qui facilite la collaboration entre les membres d'une équipe sur un même projet, en leur permettant de partager facilement leur travail et d'avancer ensemble.

En plus d'être open source et de promouvoir une démarche collaborative, GitLab propose une version gratuite particulièrement riche en fonctionnalités. Pour les entreprises, des solutions plus avancées et adaptées à des besoins spécifiques sont également disponibles.

2.1. Avantages de GitLab :

- **Gestion DevOps complète** : GitLab offre une plateforme intégrée pour tout le cycle de vie des applications, de la planification au déploiement,
- **Auto-hébergement** : GitLab propose une version open-source auto-hébergée, idéale pour les entreprises qui souhaitent un contrôle total sur leurs données,
- **Outils de sécurité et audit** : GitLab inclut des outils de sécurité intégrés (scans de vulnérabilités, audits de code) dans ses versions payantes,
- **Flexibilité des pipelines CI/CD** : GitLab offre une grande flexibilité pour personnaliser et automatiser les pipelines de déploiement,
- **Collaboration étendue** : Planification de projets, gestion des tâches avec des épics, feuilles de route et milestones directement intégrées,
- **Open-source** : GitLab Community Edition est open-source.

2.2. Inconvénients de GitLab :

- Interface moins intuitive que GitHub,
- **Plans payants coûteux** : Certaines fonctionnalités avancées (sécurité, monitoring, etc.) sont disponibles uniquement dans les plans premium,

- **Performance et latence** : Les utilisateurs de la version auto-hébergée peuvent rencontrer des problèmes de performance,
- **Moins d'intégrations tierces que GitHub** : Bien que GitLab s'intègre à de nombreux outils, il n'a pas autant de connecteurs ou d'intégrations que GitHub.

3. Choix de GitHub ou GitLab :

Choisir GitHub présente de nombreux avantages, notamment pour les développeurs et les équipes qui travaillent sur des projets logiciels. Voici plusieurs raisons pour lesquelles GitHub est une plateforme privilégiée :

1. Gestion de version avec Git

GitHub est basé sur **Git**, un système de contrôle de version distribué qui permet de suivre les modifications du code source au fil du temps. Grâce à Git, les développeurs peuvent :

- Suivre les versions des fichiers,
- Revenir à des versions précédentes,
- Comparer des versions,
- Travailler sur plusieurs branches sans risquer de casser le projet principal.

2. Collaboration facilitée

GitHub simplifie la collaboration entre développeurs en offrant des outils comme :

- **Pull requests** : Pour proposer des modifications, obtenir des retours et fusionner du code après validation.
- **Issues** : Pour signaler des bogues, discuter de nouvelles fonctionnalités, et gérer les tâches d'un projet.
- **Discussions et révisions de code** : Les développeurs peuvent échanger sur des parties spécifiques du code, améliorant ainsi la qualité via la révision collaborative.

3. Communauté vaste et ouverte

GitHub héberge des millions de projets open-source, ce qui en fait une plateforme centrale pour les contributions aux logiciels libres. Cela permet aussi :

- De découvrir du code existant pour apprendre ou s'inspirer,

- De contribuer à des projets bien établis,
- D'attirer des collaborateurs pour son propre projet.

4. Intégration avec des outils tiers

GitHub offre des intégrations avec divers outils pour améliorer les flux de travail :

- **CI/CD** : Intégration continue et déploiement continu avec des outils comme GitHub Actions pour automatiser les tests et déploiements.
- **Outils de gestion de projets** : Planification et gestion des tâches via GitHub Projects, qui permet d'organiser les issues et les pull requests en tableau Kanban.
- **Outils de code qualité** : Intégration avec des outils d'analyse statique pour garantir la qualité du code.

5. Hébergement de pages et projets

Avec **GitHub Pages**, les utilisateurs peuvent héberger des sites web directement depuis un dépôt GitHub, ce qui est pratique pour :

- Créer des pages de documentation pour les projets,
- Héberger des blogs ou des sites personnels,
- Montrer des projets de portfolio.

6. Visibilité et reconnaissance

Pour les développeurs, utiliser GitHub permet de :

- Partager leur travail avec le monde entier,
- Construire une visibilité en ligne grâce à leur profil GitHub,
- Recevoir des retours et des contributions de la communauté.

7. Sécurité et gestion des accès

GitHub propose des outils pour la sécurité des projets :

- Gestion des autorisations (public, privé),
- Sécurisation des accès via des clés SSH ou des authentifications à deux facteurs,
- Scans de vulnérabilités et alertes de sécurité pour les dépendances.

8. Interface utilisateur simple et intuitive

Même pour les nouveaux utilisateurs de Git, GitHub propose une interface graphique conviviale qui rend l'interaction avec les dépôts beaucoup plus simple. Les fonctionnalités comme la visualisation des commits, des branches, ou encore les historiques de fichiers sont claires et accessibles.

9. Gratuité et offres adaptées aux besoins

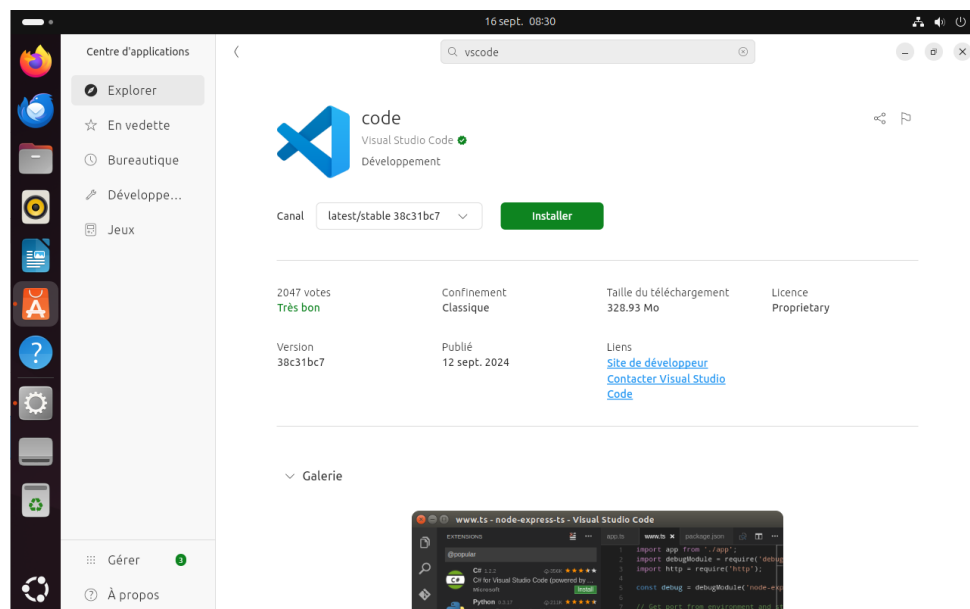
GitHub propose des offres gratuites pour les projets open-source, avec la possibilité de créer des dépôts privés. Il existe aussi des plans payants pour les entreprises et les équipes ayant des besoins plus spécifiques (plus d'espaces de stockage, support prioritaire, etc.).

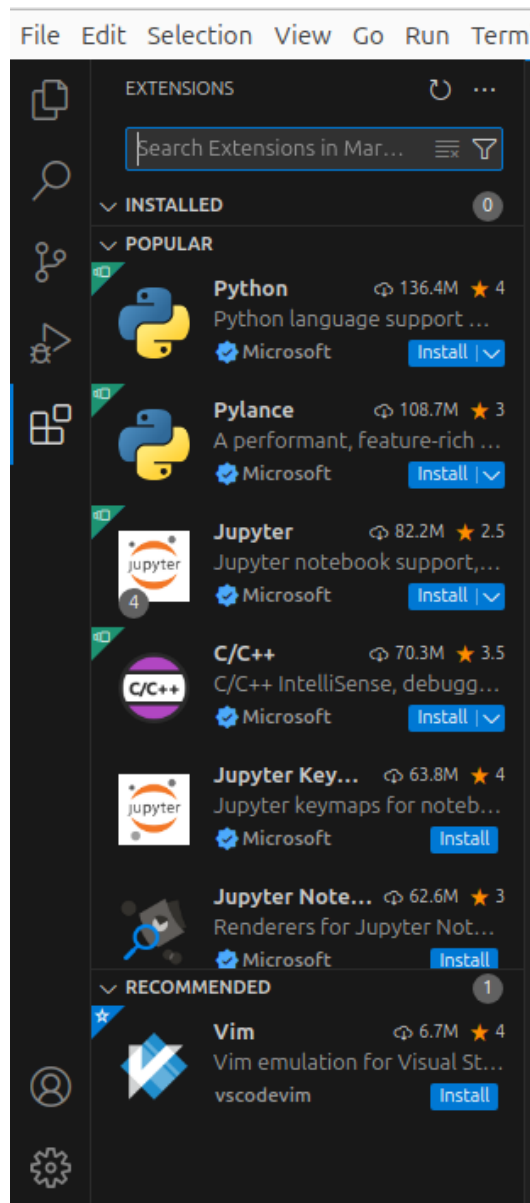
10. Support pour les entreprises et les équipes

Les fonctionnalités de GitHub comme **GitHub Enterprise** permettent aux entreprises de bénéficier d'une gestion centralisée du code, avec des outils de collaboration avancés, tout en assurant la conformité et la sécurité des projets à grande échelle.

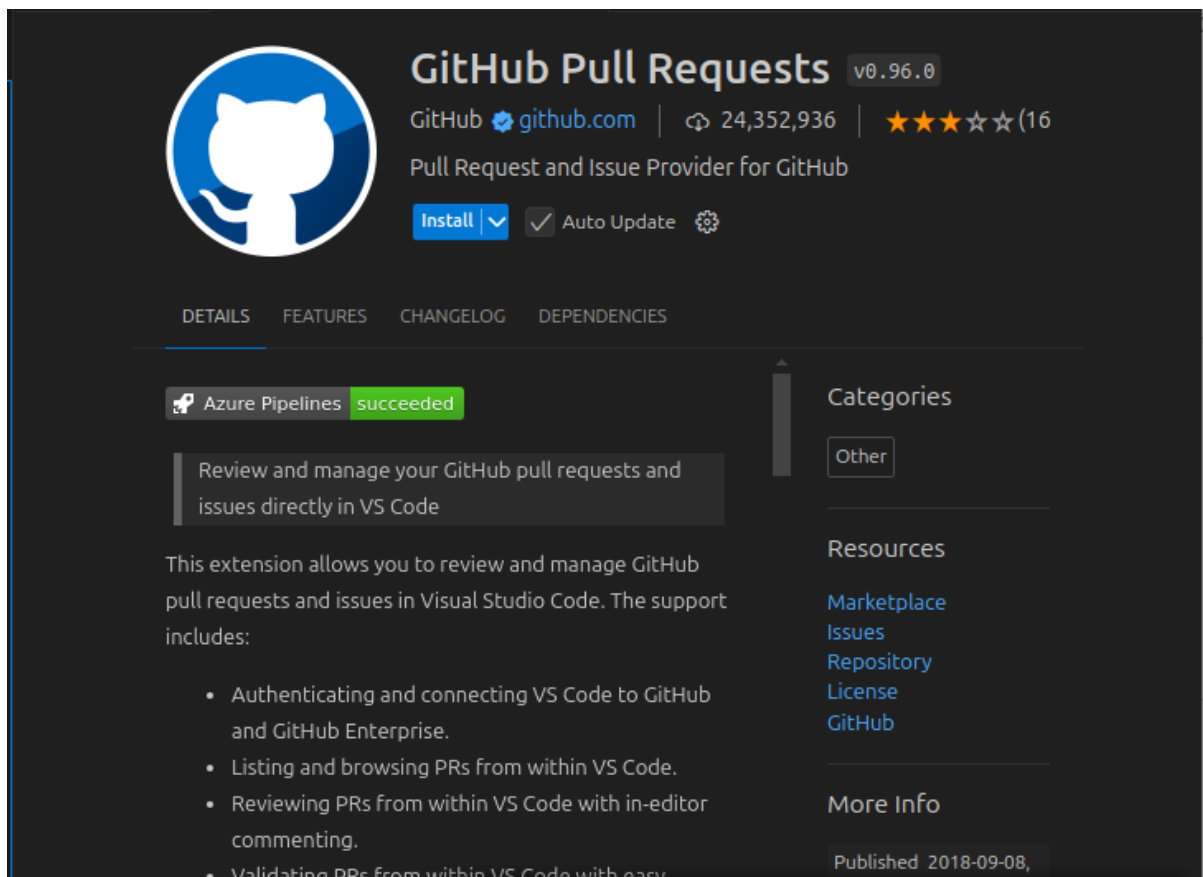
4.Installation de Visual Studio Code :

Prenez le premier lien pour installer Visual Studio Code.

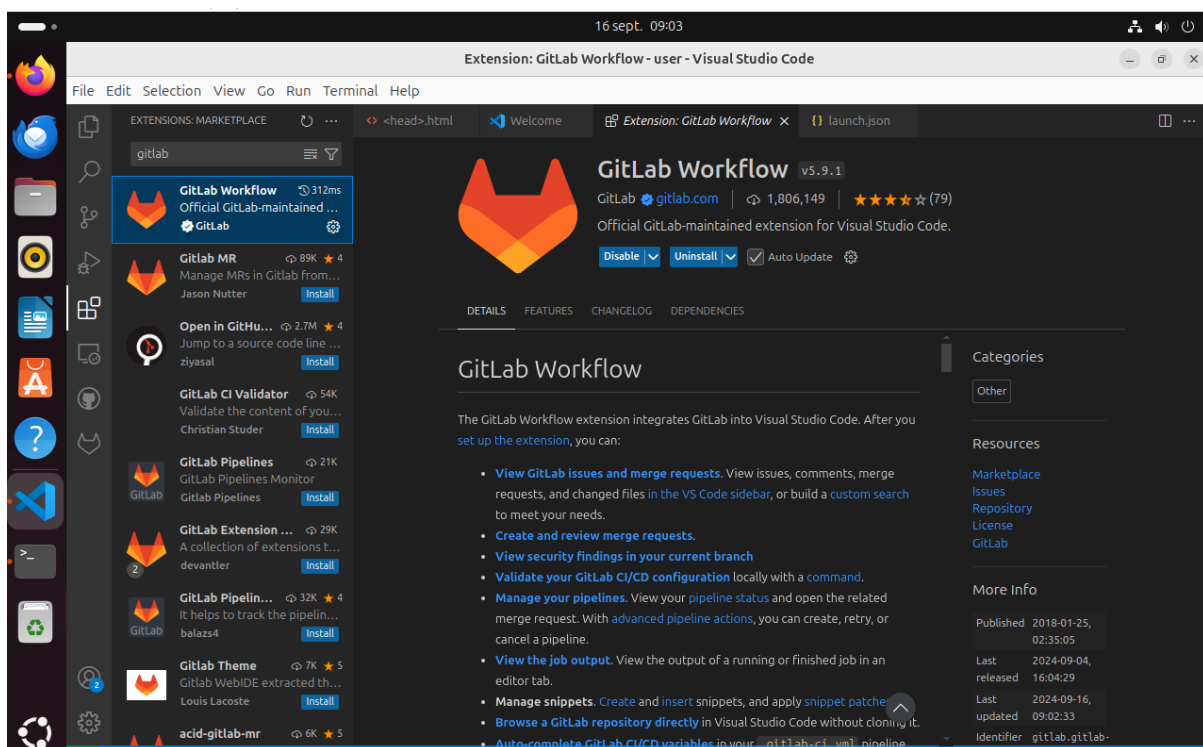




Allez dans **Extensions** dans la barre de tâche de Visual Studio Code.



Installer **GitLab** ou **GitHub** selon vos besoins.




```
user@user-VirtualBox:~$ sudo apt install git
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  git-man liberror-perl
Paquets suggérés :
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb
  git-cvs git-mediawiki git-svn
Les NOUVEAUX paquets suivants seront installés :
  git git-man liberror-perl
0 mis à jour, 3 nouvellement installés, 0 à enlever et 15 non mis à jour.
Il est nécessaire de prendre 4 804 ko dans les archives.
Après cette opération, 24,5 Mo d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n] O
Réception de :1 http://archive.ubuntu.com/ubuntu noble/main amd64 liberror-perl
all 0.17029-2 [25,6 kB]
Réception de :2 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 git-ma
n all 1:2.43.0-1ubuntu7.1 [1 100 kB]
Réception de :3 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 git am
d64 1:2.43.0-1ubuntu7.1 [3 679 kB]
4 804 ko réceptionnés en 1s (4 002 ko/s)
```

Installation de **Git** (nécessaire pour utiliser GitLab ou GitHub).

```
user@user-VirtualBox:~$ cd Documents
user@user-VirtualBox:~/Documents$ git config --global user.name "test test"
user@user-VirtualBox:~/Documents$ git config --global user.email mathisthedark@g
mail.com
user@user-VirtualBox:~/Documents$ S
```

Nous nous connectons avec notre **nom** et notre **adresse mail**.

Connectez-vous sur GitHub ensuite et faites un nouveau répertoire pour travailler.




 New repository

Appuyer.

Create a new repository



A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *  Mat45duv /

Great repository names are short and memorable. Need inspiration? How about [silver-system](#) ?

Description (optional)

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

- ☐ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

```
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Mini site MongoDB</title>
7 </head>
8 <body>
9   <h1>Bienvenue sur le mini site MongoDB</h1>
10
11
12   <ul>
13     <li><a href="ajouter_personne.html">Ajouter une personne</a></li>
14     <li><a href="afficher_personnes.php">Afficher les personnes</a></li>
15   </ul>
16 </body>
17 </html>
```

Petit programme HTML pour assurer le bon fonctionnement de **GitHub**.


Cliquer sur l'icône de **GitHub** à droite de VisualStudioCode:



Autoriser la connexion avec GitHub (une fenêtre flottante doit apparaitre en bas à droite de la page).

Git: Add Remote...

Chercher **git add remote** dans la barre de recherche.

 Add remote from GitHub Ensuite cliquer sur **add remote**.

Repository name (type to search)

 Mat45duv/mathisd <https://github.com/Mat45duv/mathisd.git>

Puis **saisissez l'url** de votre répertoire GitHub.

Remote name

Please provide a remote name (Press 'Enter' to confirm or 'Escape' to cancel)

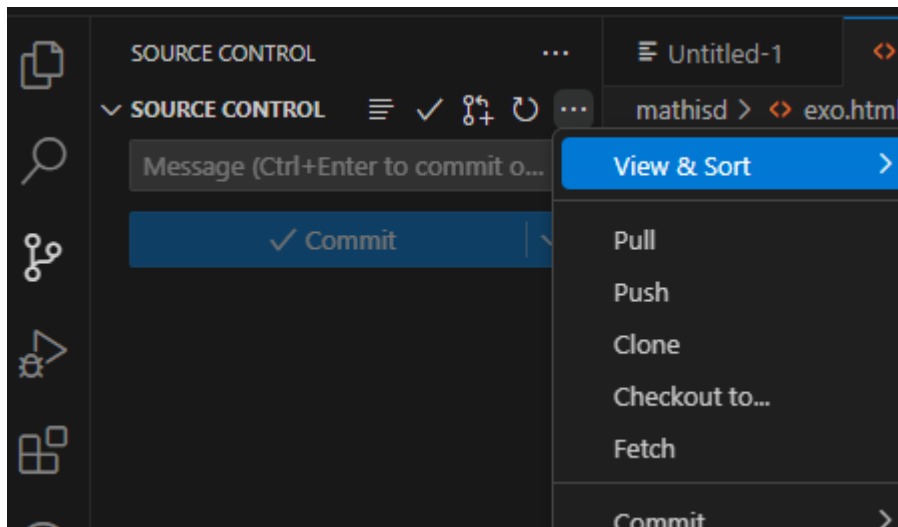
Donner un **nom**.

Pour utiliser les fonctionnalités de Git, vous pouvez ouvrir un dossier contenant un référentiel Git ou le cloner à partir d'une URL.

Open Folder

Clone Repository

Un message apparaitra.



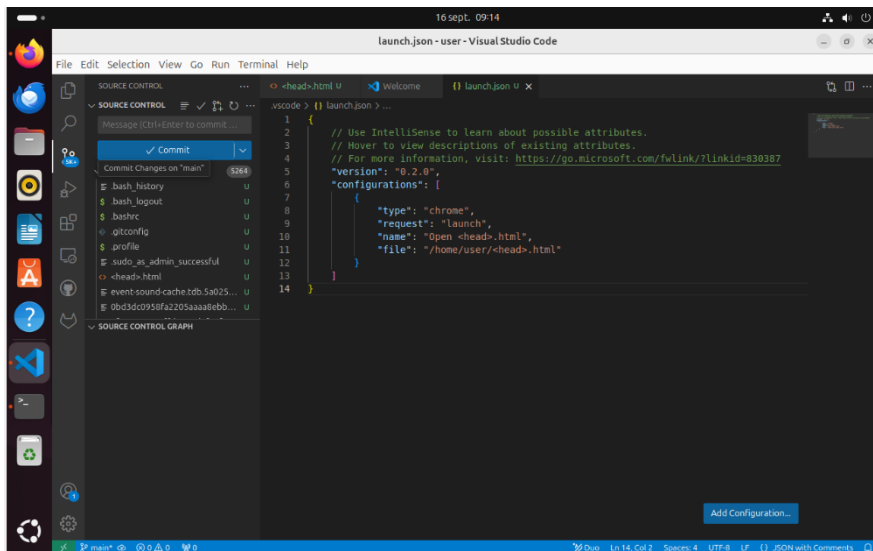
Appuyer ensuite sur

push, une fenêtre apparaîtra pour vous connecter à GitHub.

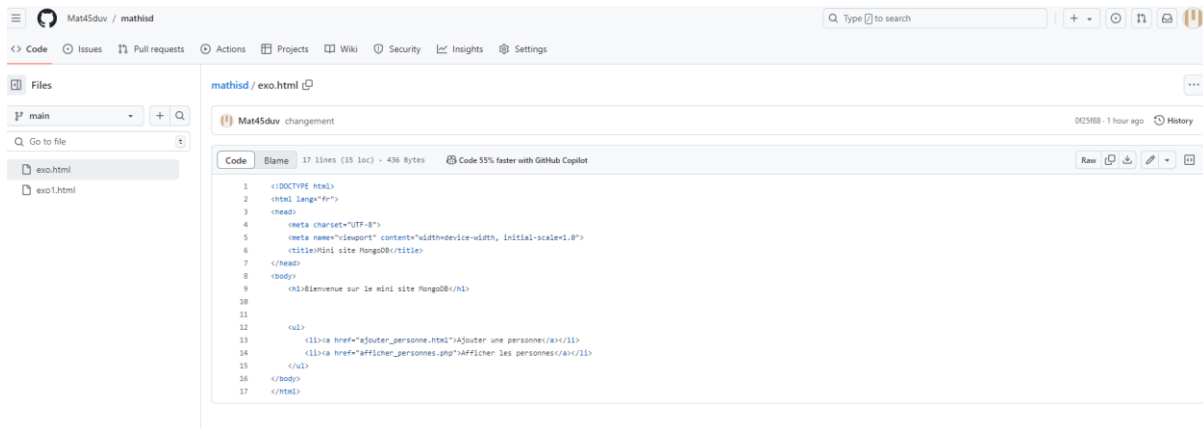


Connectez-vous.

Une fois connecter, on peut retrouver notre petit code grâce à un dossier où se trouve notre code où le cloner.



En se connectant à **GitHub** avec l'utilisateur **Mat_45** nous voyons que les changements sont **Commits** (enregistrés) sur notre petit programme.

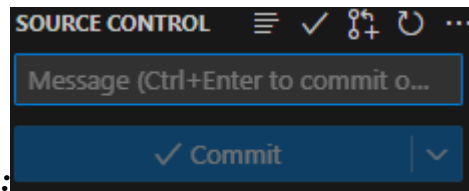


Où créer un répertoire grâce aux lignes de commande, créer votre fichier, ouvrir le **cmd** dans **Visual Studio Code** : **Ecrivez les lignes suivantes en changeant le nom du fichier et le lien.**

```
echo "# olivier-mathis" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Mat45duv/olivier-mathis.git
git push -u origin main
```

Votre code est maintenant relié à GitHub.

Appuyer sur **push** pour envoyer votre code sur GitHub. Appuyer sur **pull** si vous avez changé votre code directement sur GitHub pour le mettre à jour.

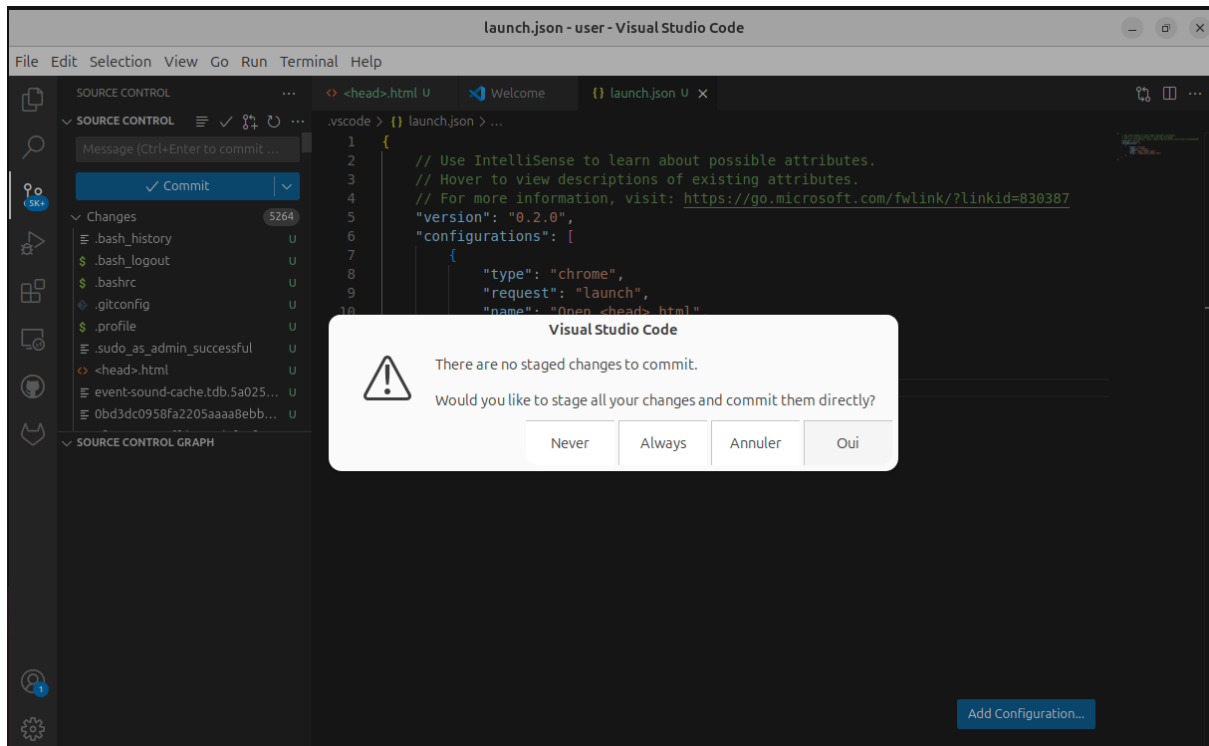


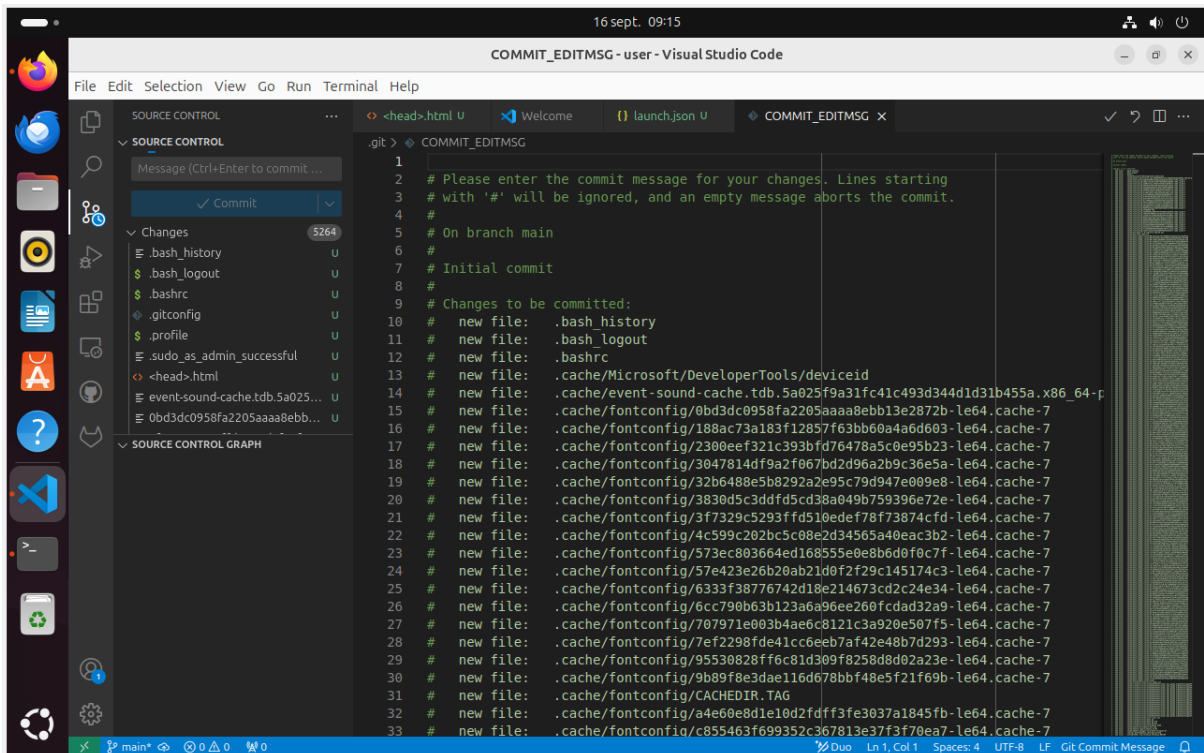
En cas de changement de code :

Appuyer sur Commit pour enregistrer les modifications si vous modifiez votre code (vous pouvez mettre un nom au Commit).

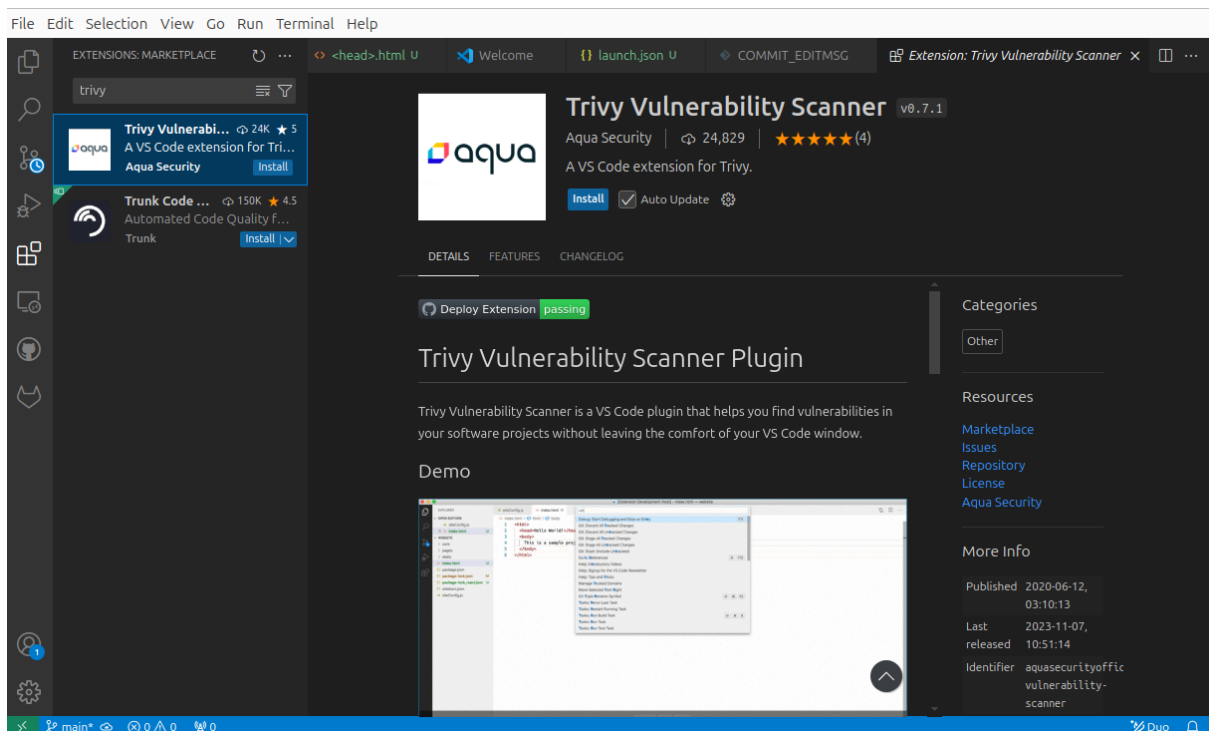
Push

Appuyer ensuite sur push pour le mettre/pousser sur GitHub.

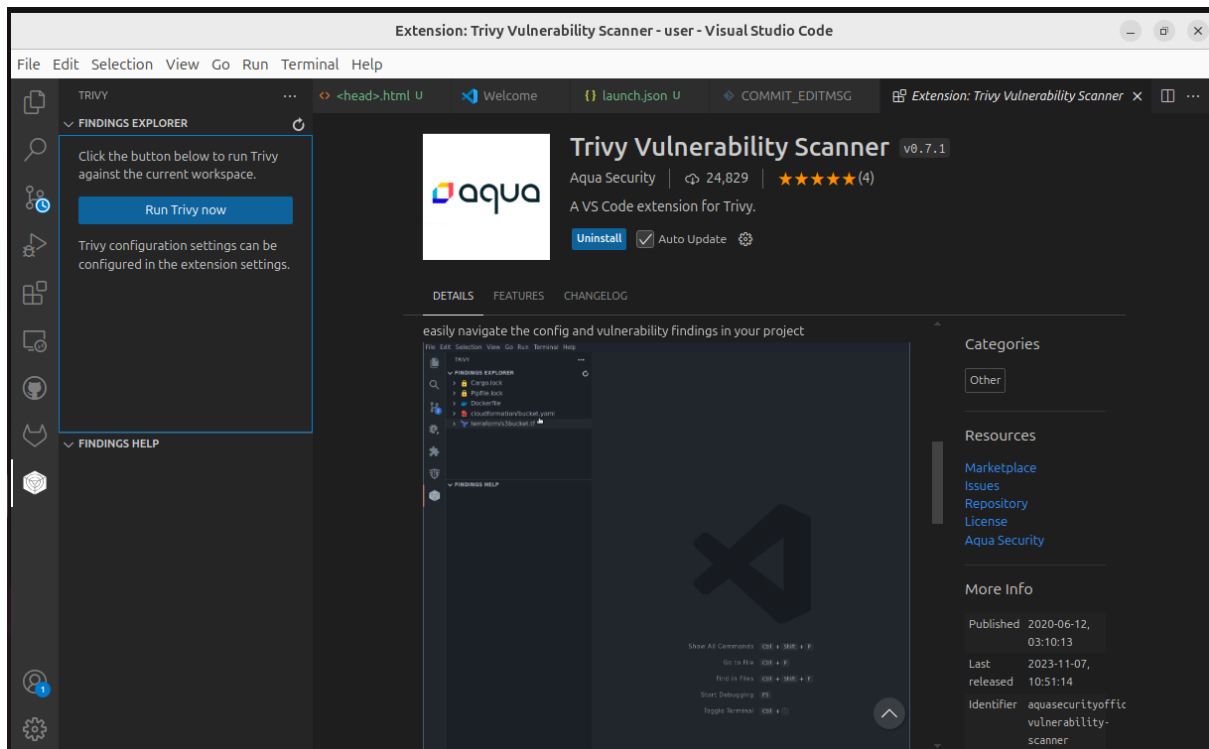




Les changements sont bien pris en comptes et **enregistrer sur GitHub**. Le projet est mis à jour en temps réel sur GitHub. Le test est donc conclu.



Installation de **Trivy** pour plus de sécurité.



Trivy est un outil de sécurité open-source utilisé principalement pour la **détection de vulnérabilités** dans différentes ressources telles que :

1. **Images de conteneurs** (Docker, Podman, etc.)
2. **Dépôts de code** (Git, GitLab, etc.)
3. **Systèmes de fichiers**
4. **Infrastructure-as-Code** (Terraform, Kubernetes, etc.)

Fonctionnalités principales de Trivy :

5. **Analyse des vulnérabilités** : Trivy scanne des images de conteneurs et d'autres cibles pour identifier des failles de sécurité dans les bibliothèques et dépendances utilisées.
6. **Scanning de fichiers de configuration** : Il permet de détecter des configurations non sécurisées dans des environnements comme Kubernetes ou Terraform.
7. **Rapports de compatibilité** : Trivy génère des rapports détaillant les vulnérabilités trouvées et leur gravité, aidant ainsi les équipes à prioriser les correctifs.

8. **Support multi-environnements** : Il peut être utilisé à la fois en local ou intégré dans des pipelines CI/CD pour une surveillance continue de la sécurité des applications.

En résumé, Trivy aide les développeurs et les administrateurs à identifier et à corriger les vulnérabilités dans les infrastructures et les applications avant leur déploiement en production.

5. Les bonnes pratiques de sécurité sur un poste de travail :

La sécurité informatique au sein des postes de travail en entreprise est essentielle pour protéger les données sensibles et assurer la continuité des activités. Voici quelques bonnes pratiques pour améliorer la sécurité des postes dans une entreprise :

1. Gestion des mots de passe

- **Mots de passe forts** : Utiliser des mots de passe complexes, longs (minimum 12 caractères), comprenant lettres majuscules/minuscules, chiffres, et caractères spéciaux.
- **Gestionnaire de mots de passe** : Encourager l'utilisation d'un gestionnaire de mots de passe pour éviter les mots de passe faibles et réutilisés.
- **Authentification multi-facteurs (MFA)** : Activer l'authentification à deux facteurs pour renforcer la protection des comptes (code OTP, applications d'authentification).

2. Mises à jour régulières

- **Mises à jour du système d'exploitation** : Configurer les postes pour appliquer automatiquement les mises à jour de sécurité.

- **Mises à jour des logiciels** : S'assurer que tous les logiciels (navigateurs, suites bureautiques, etc.) sont également maintenus à jour.

3. Antivirus et pare-feu

- **Antivirus/antimalware** : Installer et maintenir à jour des solutions antivirus sur chaque poste pour détecter et supprimer les menaces.
- **Pare-feu** : Activer un pare-feu sur chaque machine pour limiter les accès non autorisés et surveiller le trafic réseau.

4. Chiffrement des données

- **Chiffrement des disques** : Activer le chiffrement complet du disque (ex. BitLocker, FileVault) pour protéger les données en cas de vol ou de perte d'un poste.
- **Chiffrement des communications** : Utiliser des connexions sécurisées (ex. VPN, HTTPS) pour garantir la confidentialité des données échangées.

5. Politique de verrouillage automatique

- **Verrouillage automatique des sessions** : Configurer un verrouillage automatique de l'écran après quelques minutes d'inactivité afin de protéger l'accès aux postes non supervisés.
- **Verrouillage manuel** : Encourager les utilisateurs à verrouiller manuellement leur poste (ex. raccourci clavier) dès qu'ils s'éloignent de leur poste de travail.

6. Accès minimal aux ressources

- **Principe du moindre privilège** : Ne donner aux utilisateurs que les droits nécessaires à leur travail (éviter de donner des accès administratifs non nécessaires).
- **Contrôle des accès** : Utiliser des systèmes de contrôle d'accès basés sur des rôles pour s'assurer que seuls les utilisateurs autorisés accèdent aux données sensibles.

7. Surveillance et journalisation

- **Journalisation des activités** : Activer la journalisation des actions sur les postes pour détecter des comportements anormaux ou malveillants.
- **Supervision en temps réel** : Utiliser des outils de surveillance du réseau et des postes pour détecter des incidents de sécurité en temps réel.

8. Sensibilisation et formation des utilisateurs

- **Formation à la cybersécurité** : Former régulièrement les employés aux bonnes pratiques de cybersécurité (phishing, sécurité des mots de passe, comportements en ligne).
- **Simulations d'attaques** : Organiser des simulations (ex. campagnes de phishing simulé) pour tester et renforcer la vigilance des employés.

9. Contrôle des périphériques externes

- **Contrôle des ports USB** : Limiter l'utilisation des périphériques externes (clés USB, disques durs) pour éviter l'introduction de logiciels malveillants.
- **Politique d'utilisation des périphériques** : Mettre en place une politique claire concernant l'utilisation des périphériques personnels et externes.

10. Sauvegarde des données

- **Sauvegarde régulière** : Mettre en place des solutions de sauvegarde régulière et automatisée des données critiques (localement et sur le cloud sécurisé).
- **Test des restaurations** : Tester régulièrement les restaurations pour s'assurer que les sauvegardes fonctionnent et sont fiables en cas de sinistre.

11. Politiques de BYOD (Bring Your Own Device)

- **Sécurisation des appareils personnels** : Si les employés utilisent leurs appareils personnels pour accéder aux ressources de l'entreprise, il faut mettre en place des solutions de sécurité (ex. gestion des appareils mobiles, MDM).
- **Segmentation des accès** : Restreindre l'accès aux ressources sensibles depuis les appareils non gérés.

12. Politique de réponse aux incidents

- **Plan de réponse aux incidents** : Mettre en place un plan de réponse en cas d'incident de sécurité (cyberattaque, vol de données, etc.) afin de réagir rapidement et limiter les dégâts.
- **Équipe de réponse dédiée** : Avoir une équipe désignée ou un prestataire pour traiter les incidents et gérer la communication en cas de crise.

13. Isolement des postes sensibles

- **Segmentation du réseau** : Isoler les postes critiques sur des segments réseau distincts pour limiter la propagation des attaques.

- **Environnements virtuels** : Utiliser des machines virtuelles pour les utilisateurs qui doivent accéder à des environnements ou données sensibles, réduisant ainsi les risques d'intrusion.

Ces bonnes pratiques permettent de protéger efficacement les postes de travail contre une variété de menaces informatiques tout en garantissant la sécurité des données de l'entreprise.