MENIER Thomas Wanchaï
GENETET Maud

IUT de Gradignan

Rapport de la SAÉ S1.03
Machine virtuelle

Sommaire

1] Dossier d’étude et de choix des solutions

a) Définition de la machine virtuelle

Afin de mettre en place notre machine virtuelle, nous avons utilisé1 le logiciel VMWare. Le client demandait à avoir une machine avec le système d’exploitation Linux ainsi qu’avec un bureau léger. Nous avons donc déterminé que le client souhaitait avoir l’installation la plus légère possible.

Ainsi, nous avons choisi d’installer le système d’exploitation XUbuntu, qui est non seulement léger (requiert près de 8,5 Go)[1] mais qui consomme notamment moins de mémoire vive au démarrage. De plus, puisque c’est un dérivé de l’OS Ubuntu, le client aura plus de facilité à trouver des informations en cas de problème sur sa machine, Ubuntu étant un des noyaux linux les plus populaires. Il est à noter qu’il possède le bureau XFCE

Dans la configuration de la machine, il a été décidé de créer la machine en un seul fichier, afin que celle-ci soit plus performante. La contrepartie est que si son fichier est déplacé, il y a plus de risques de bugs.

En ce qui concerne les caractéristiques de la machine virtuelle en elle-même, nous lui avons attribué 2 Go de RAM et un processeur de 4 coeurs. Une grande quantité de RAM n’était pas nécessaire car le client ne souhaitait que 3 programmes. Cependant pour le processeur, nous avons préféré lui donner 4 coeurs afin que la compilation des programmes Rust soit plus rapide, et que le développeur puisse être plus productif.

Pour l’espace disque, nous l’avons limité à 12GB, car les programmes à installer (voir choix logiciels) requièrent au total 1,5GB[2] d’espace disque. L’USB Controller a été défini à USB 3.1, ce qui était plus judicieux car ce type de bus USB est rétro-compatible. La machine utilise la connexion Internet de la machine hôte, c’est donc la même adresse IP qui est utilisée.

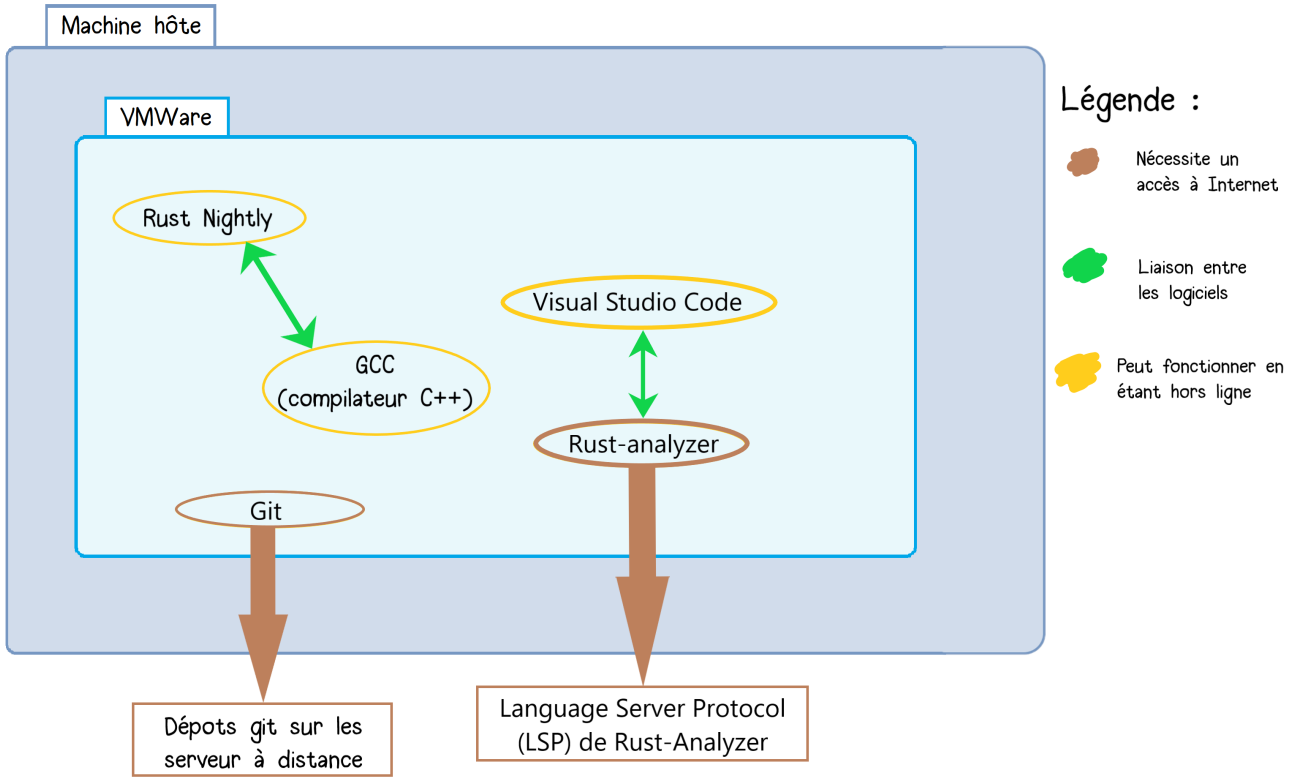
1 Source: https://xubuntu.org/requirements/
2 Tailles des programmes récoltées pendant leur installation sur la machine

b) Choix logiciels

Le choix du système d’exploitation Xubuntu a déjà été justifié précédemment : plus léger, le bureau XFCE y est déjà présent et popularité.
Le reste des programmes nécessaires a été défini par le client :
Git pour le gestionnaire de sources, Visual Studio Code avec l’extension rust-analyzer ainsi que le langage de programmation Rust et son compilateur associé. Il était aussi nécessaire d’installer la commande « gcc » sur la machine virtuelle afin de permettre la compilation de programmes Rust.

2] Architecture logicielle

Schéma de l'architecture logicielle



3] Installation de la machine

Voici la démarche à suivre afin d'installer et de configurer la machine virtuelle demandée par le client :

- Télécharger l'OS

Tout d'abord, télécharger la version 20.04 du système d'exploitation XUbuntu, que vous pouvez trouver ici **http://ftp.free.fr/mirrors/ftp.xubuntu.com/releases/20.04/release/** et téléchargez la version : **xubuntu-20.04.3-desktop-amd64.iso**

- Création de la machine virtuelle

Lancez **VMWare Workstation Player**, cliquez sur « Create a New Virtual Machine ». Ensuite, cliquez sur « Browse » et sélectionnez le fichier .iso que vous avez téléchargé préalablement.

Cliquez sur « Next >> », puis dans les champs à remplir, inscrivez « Alice » dans le premier champ pour le nom familier du profil administrateur. Renseignez ensuite le nom d'utilisateur et le mot de passe de ce compte administrateur.

Cliquez encore sur « Next >> », mettez la taille du disque à **12.0** dans le champ « Maximum disk size (GB) » et veillez à cocher « **Store virtual disk as a single file** ».

Cliquez de nouveau sur « Next >> », cliquez sur « **Customize Hardware** ». Sélectionnez **2GB** à l'aide de votre curseur. Allez ensuite dans « **Processors** » et sélectionner **2** pour le champ « **Number of processor cores** ». Enfin, cliquez sur l'onglet « USB Controller », sélectionnez **USB 3.1** dans le champ « **USB Compatibility** » et **cochez toutes les cases** intitulées « Show all USB input devices » et « Share Bluetooth devices with the virtual machine ». Vous pouvez enfin cliquer sur « Close ».

Pour terminer, cliquez sur « Finish » et laissez la machine s'initialiser et s'installer.

- Changer le clavier en français

Il est possible qu'à l'issue du lancement de la machine virtuelle, celle-ci soit configurée avec un clavier anglais. Vous pouvez le vérifier en essayant de vous connecter une fois arrivé sur l'écran de connexion. Si tel est le cas, suivez ces instructions :

Appuyez simultanément sur les touches **CTRL**+**Alt**+**F1.** Un invite de commandes devrait remplacer l'écran de connexion. Lorsque celui-ci vous demande le nom d'utilisateur, tapez sur votre clavier **qlice** pour le nom d'utilisateur puis **qlice~but** pour le mot de passe. Tapez ensuite `sudo dpkg-reconfigure keyboard-configuration .` Une interface graphique devrait apparaître, appuyez sur **Tab** à répétition jusqu'à que le mot « OK » soit en surbrillance. Lorsque c'est le cas, appuyez sur la touche **Entrée**. Faites ceci jusqu'à que vous voyez apparaître des noms de langue sur votre écran.

A ce moment-là, sélectionnez la langue **French** à l'aide de vos flèches directionnelles puis appuyez sur **Entrée**. Continuez la combinaison de touches **Tab** et **Entrée** jusqu'à ne plus voir l'interface graphique. Une fois de retour sur l'invite de commande, appuyez simultanément sur **CTRL**+**ALT**+**F7** aifn de sortir de cet invite, puis redémarrez la machine virtuelle.

La machine est dorénavant configurée avec un clavier français.

- Ajouter l'utilisateur Bob

Une fois connecté avec le compte d'Alice, appuyez simultanément sur les touches **CTRL**+**ALT**+**T** pour ouvrir un invite de commande. Tapez ensuite `sudo adduser bob` puis suivez les instructions à l'écran.

Après avoir ajouté l'utilisateur, il faut s'assurer que celui-ci n'ait pas accès aux fichiers d'Alice. Toujours dans l'invite de commandes, entrez les commandes suivantes :

`chmod a-rwx ~/` **et** `chmod u+rwx ~/`

Désormais, l'utilisateur Bob n'a pas accès aux fichiers d'Alice

- Installer les paquets nécessaires

Dans un invite de commandes, entrez les commandes suivantes pour installer les différentes commandes nécessaires au bon fonctionnement de la machine virtuelle :

```
sudo apt install curl
sudo apt install rustc
sudo apt install gcc
sudo apt install git
```

Une fois rentrées, déconnectez vous du compte d'Alice et connectez vous avec le compte de Bob défini précédemment.
Sous le compte de Bob, ouvrez un invite de commande et installez Rust Nightly en entrant la commande suivante :
`curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh`
Faites **Entrée**, **t**apez **2** puis **Entré**e. Continuez d'appuyer sur cette touche jusqu'à voir apparaître la phrase « Default toolchain ? » à l'écran. A ce moment-là, écrivez `nightly` puis continuez à appuyer sur **Entrée**. Une fois de retour sur le menu des choix, tapez 1 puis **Entrée** et le programme d'installation va installer Rust Nightly sur la machine.

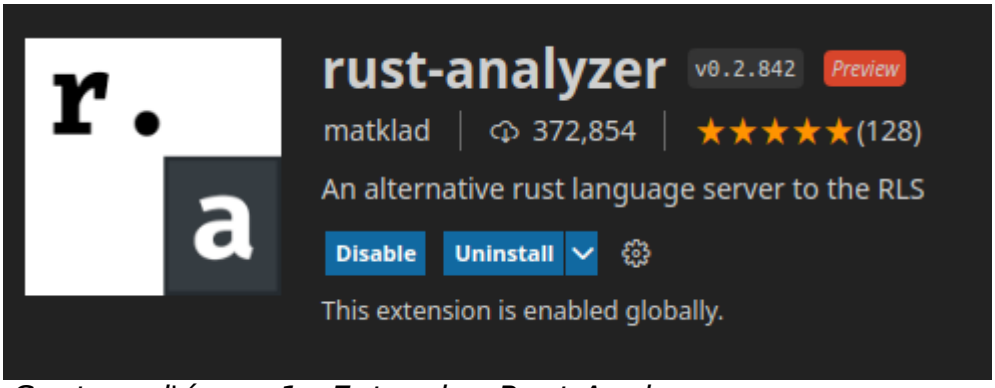
- Cloner un dépôt git

Ouvrez le répertoire de fichiers qui contiendra ce dépôt, ouvrez-y un terminal en faisant un clic droit → « *Open a terminal here* » dans le menu déroulant. Tapez ensuite `git clone lien` en remplaçant le mot « lien » par le lien de votre dépôt.

- Installer VSCode et l'extension rust-analyzer

Ouvrez un terminal, et rentrez la commande suivante :
`sudo snap install code –classic`
Ensuite ouvrez Visual Studio Code en rentrant la commande `code` dans le terminal.

Allez sur les extensions (icône de cube sur la gauche), tapez **rust-analyzer** dans la barre de recherche et installez l'extension suivante (voir capture d'écran 1).



Capture d'écran 1 : Extension Rust-Analyzer

L'extension est dorénavant installée et vous pouvez l'utiliser après avoir redémarré Visual Studio Code.

- Indiquer l'état du dépôt dans le terminal

Ouvrez un terminal, tapez nano `~/.bashrc` et cherchez dans le fichier le texte suivant :

```
if [ "$color_prompt" = yes ]; then
   PS1='${debian_chroot:+($debian_chroot)}\[\033[01;36m\]\u@\h\[\033[00>
else
   PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
```

et mettez à la place :

```
if [ "$color_prompt" = yes ]; then
   PS1='${debian_chroot:+($debian_chroot)}$(git -C ~/hello.sae4.but1.iut/ status)\n\[\
033[01;36m\]\u@\h\[\033[00>
else
   PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
```

Puis faites **Ctrl+X**, taper **Y**, puis faite Entrée, redémarrer le terminal et le prompt à été mis à jour.

- Si vous avez mit votre dossier git autre part qu'à la racine, il faut modifier:

$(git -C ~/hello.sae4.but1.iut/ status)

- Par :

$(git -C [votre_chemin]/hello.sae4.but1.iut/ status)

- [votre_chemin] étant le chemin pour aller dans le dossier git.

4] Notice d'utilisation

Comment créer un projet rust :

- Mettez vous dans le répertoire parent du projet
- Tapez : `cargo new [nom_du_projet]` et voilà votre projet est créé.

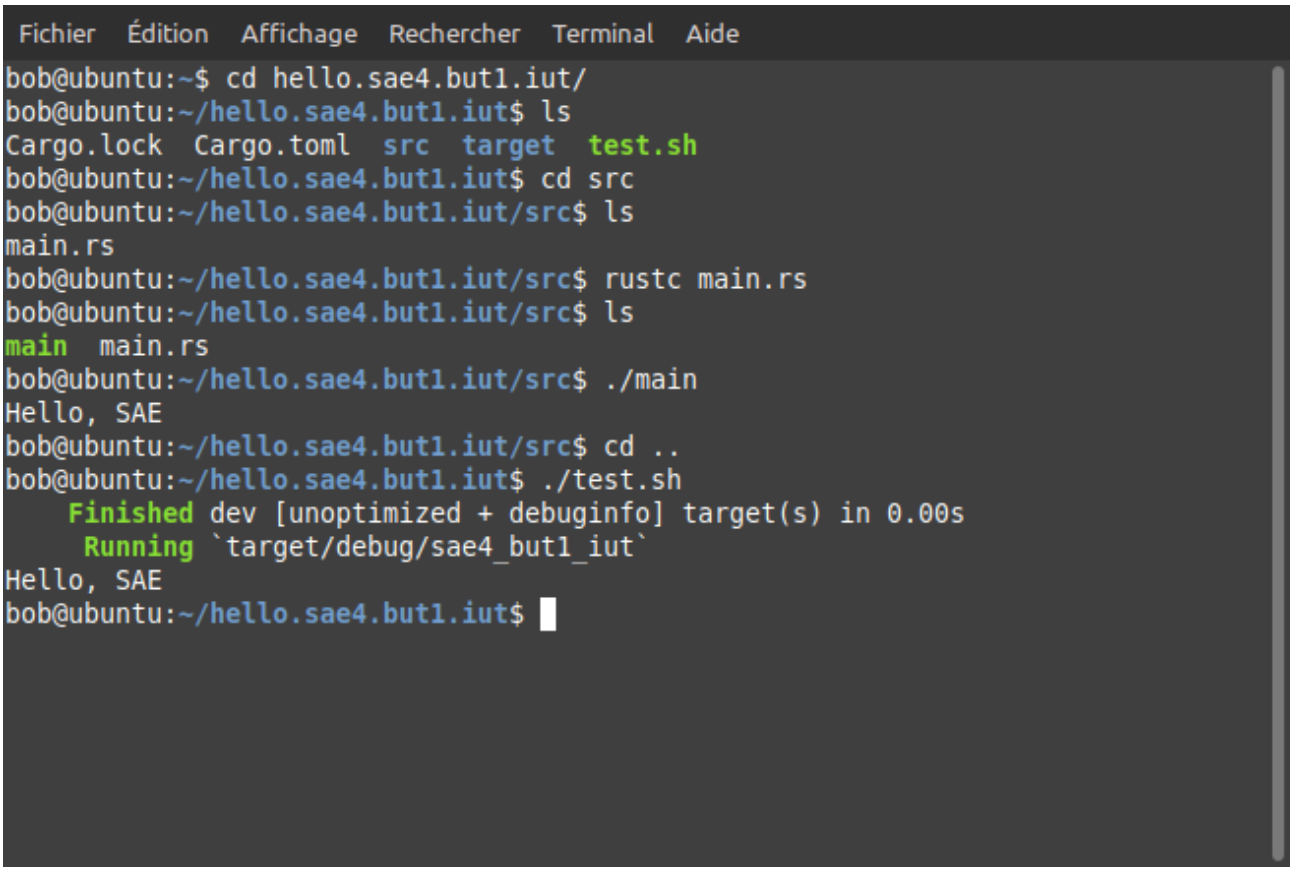
Le projet est composé de :

- Un fichier Cargo.lock
- Un fichier Cargo.toml
- Un répertoire src avec dedans un fichier main.rs, c'est le fichier contenant notre code, celui qu'on devra compiler , on peut l'ouvrir avec nano main.rs ou avec code main.rs.
- Un fichier target contenant des fichiers : `CACHEDIR.TAG` `.rustc_info.json` et des répertoires nommé : `debug`, `release` , contenant tout les deux des répertoires nommés : `build`, `deps`, `examples`, `incremental`.

Exécuter un projet :

- Ouvrez un terminal et tapez: `rustc main.rs` (le fichier se trouvant dans le répertoire src).
- Un fichier exécutable est donc créé, il est nommé main.
- Tapez ./main dans un terminal puis faites **Entrée** et votre projet sera lancé et affichera ce qu'il fait dans le terminal de commande.

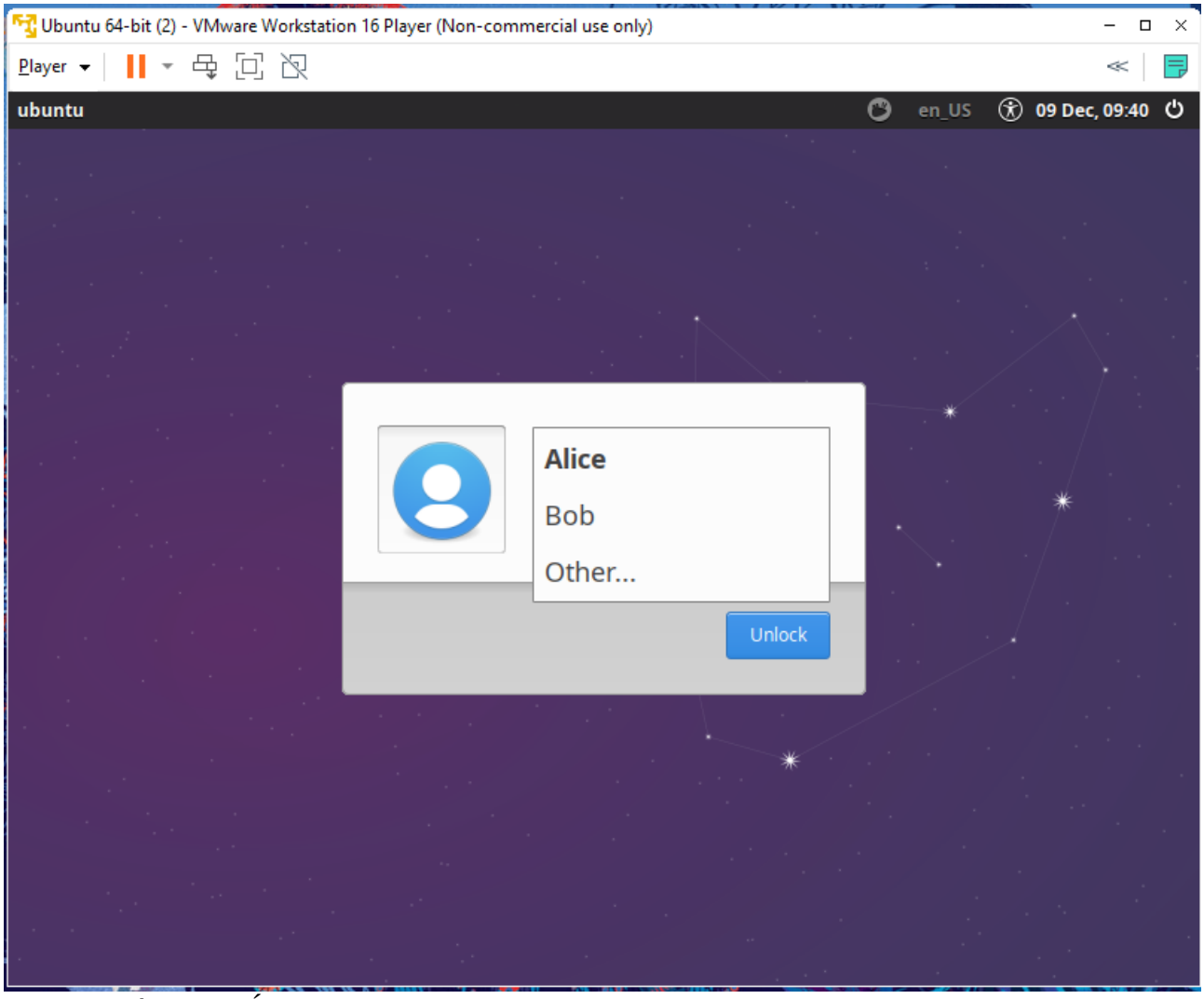
Test d’installation :



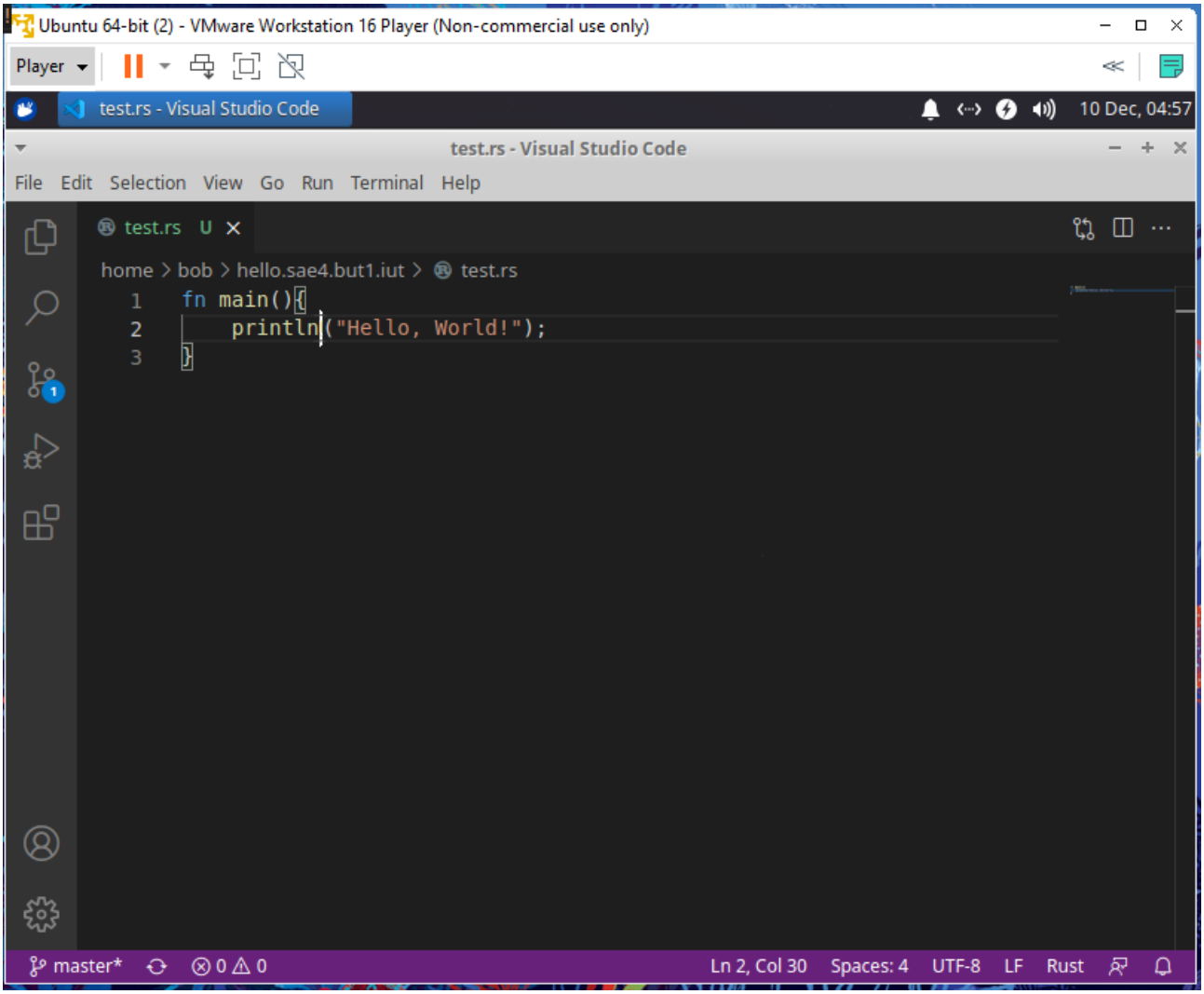
Test installation

5] Portflio

Création d'une machine virtuelle avec 2 utilisateurs : Alice l'Admin et Bob le développeur et installation de VSCode, Rust et du gestionnaire de sources Git et changement des composants.



Capture d'écran 2 : Écran de connexion Bob & Alice



Capture d'écran 3 : Projet Rust ouvert dans VSCode

6] Ce que nous avons appris :

Ce projet nous a permis d’apprendre la gestion de plusieurs utilisateurs sur une même machine et les quantités minimes nécessaires à l’installation d’une machine virtuelle. Nous avons appris de commandes Linux et leur fonctionnement. Nous avons de plus appris à nous habituer à une autre distribution Linux et comprendre ses différences ainsi que ses besoins.