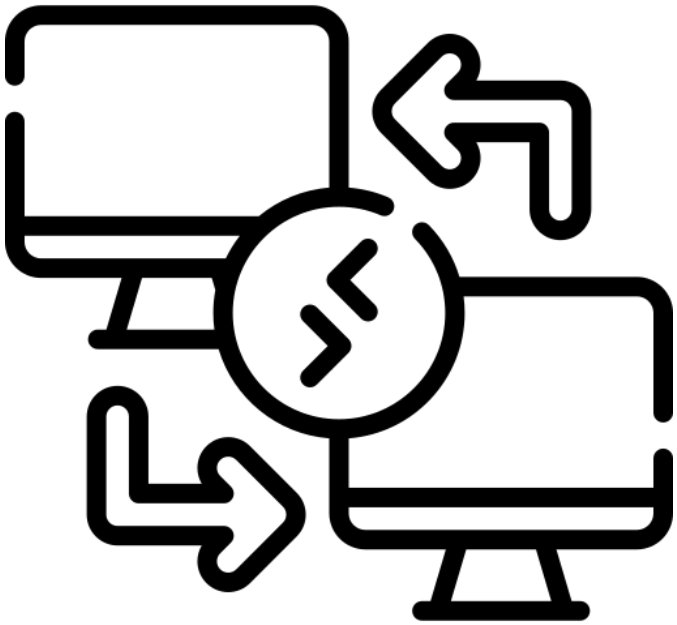


Solutions de connexion à distance

Tamara Crétard - 24.09.2025

Sommaire:

1. Contexte

Dans le cadre d'un cours du BTS SIO, l'objectif est de comprendre l'utilité et le fonctionnement de la connexion à distance. Cet apprentissage comprend notamment l'installation, l'utilisation et la comparaison de diverses solutions de connexion à distance telles que *Remmina* combinée à *xrdp* et *Bureau à distance*, *UltraVNC* avec *x11vnc* et *Apache Guacamole*.

2. Environnement de travail

Afin de pouvoir expérimenter l'utilité et le fonctionnement de la connexion à distance, l'utilisation de diverses solutions est nécessaire. Avant de procéder à leur installation, la première étape est d'analyser correctement notre environnement de travail.

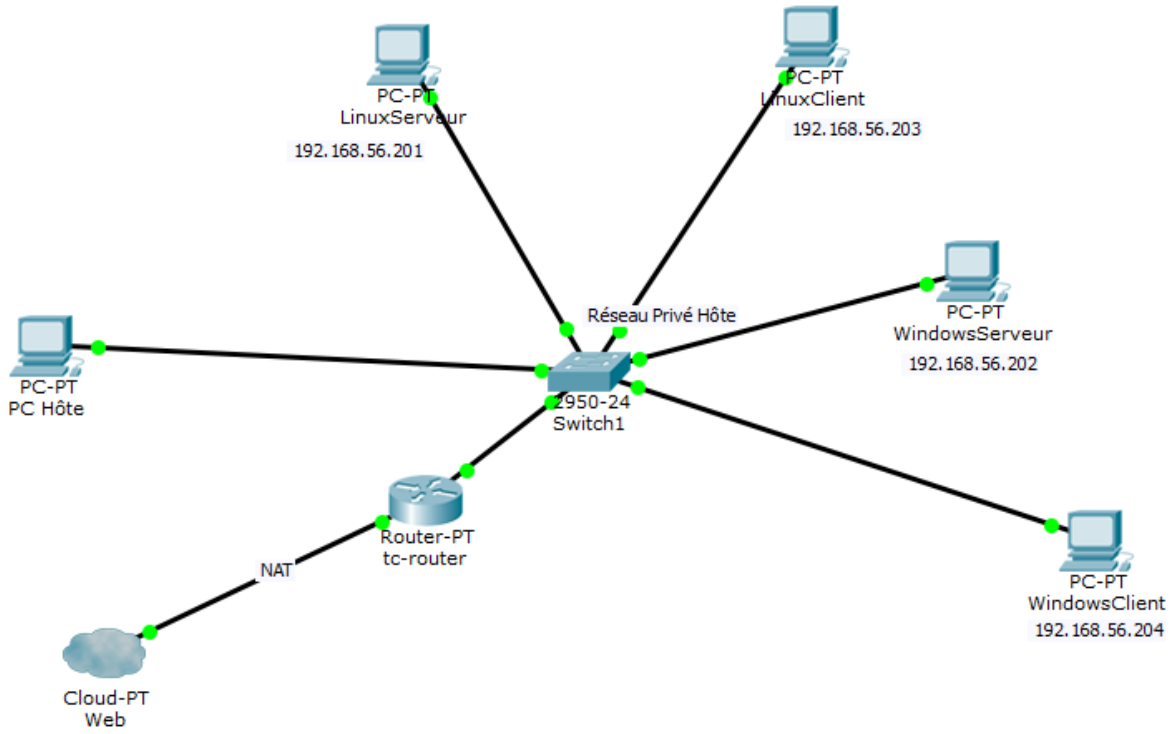
2.1. Configuration utilisée

Tout d'abord, il est intéressant de connaître la configuration utilisée pour permettre l'installation des lanceurs. Dans notre cas, la configuration utilisée est la suivante:

Processeur	11th Gen Intel(R) Core(TM) i7-11700K @ 3.60GHz 3.60 GHz
RAM	32 Go
Carte graphique	NVIDIA GeForce RTX 3080
SSD	2 To
Système d'exploitation	Microsoft Windows 11 Famille
Architecture	x64

2.2. Schéma du réseau virtuel

Afin de visualiser plus clairement la structure de notre environnement de test, un schéma réseau virtuel a été conçu à l'aide de Cisco Packet Tracer:



Dans notre cas, nous avons besoin d'une machine hôte, de quatre machines virtuelles en Réseau Privé Hôte, dans le même réseau avec le routeur:

- une machine *LinuxServeur* en 192.168.56.201 sans interface graphique
- une machine *LinuxClient* en 192.168.56.203 avec interface graphique LXDE
- une machine *WindowsServeur* en 192.168.56.202 (Windows Serveur 2022)
- une machine *WindowsClient* en 192.168.56.204 (Windows Pro 10)

3. Connexion à distance

La connexion à distance désigne l'ensemble des technologies permettant d'accéder et de contrôler un ordinateur, un serveur ou un système depuis un autre appareil, sans être physiquement présent sur le lieu où il se trouve. Elle permet à un utilisateur de visualiser le bureau d'une machine, d'exécuter des programmes, de transférer des fichiers ou d'administrer un système comme s'il y était directement connecté.

Cette méthode est très utilisée en entreprise pour la maintenance informatique, le support technique, la supervision de serveurs ou encore le télétravail. Grâce à la connexion à distance, les administrateurs peuvent gérer des infrastructures situées dans différents sites, tandis que les employés peuvent accéder à leur environnement de travail depuis n'importe où.

Selon les besoins, cette connexion peut être réalisée en mode graphique (via des protocoles comme RDP, VNC, ou des solutions comme *Apache Guacamole* et *RustDesk*) ou en mode ligne de commande. Elle doit toujours être sécurisée afin de protéger les données

et les accès contre les intrusions, notamment grâce au chiffrement des communications et à l'authentification des utilisateurs.

3.1. Protocole RDP

Le Remote Desktop Protocol (RDP) a été développé par Microsoft dans les années 90. Il permet de se connecter à distance à un ordinateur et de contrôler son interface graphique comme si l'on était devant la machine. RDP est largement utilisé pour le télétravail, l'administration à distance et le support technique.

Le protocole utilise par défaut le port TCP 3389 pour établir la connexion entre le client et le serveur, permettant ainsi d'accéder aux applications, de transférer des fichiers et de gérer l'ordinateur distant.

3.2. Protocole VNC

Le Virtual Network Computing (VNC) a été créé par AT&T dans les années 90. Il permet d'accéder à distance à l'interface graphique d'un ordinateur et de le contrôler depuis un autre appareil. VNC est souvent utilisé pour la maintenance à distance, le support technique et l'accès à des postes distants.

Par défaut, VNC utilise le port TCP 5900, avec la possibilité de définir des ports différents pour chaque session graphique.

4. Solutions

La connexion à distance peut se faire de plusieurs manières. En effet, différents protocoles existent et il est possible entre autres d'utiliser SSH ou Telnet. Cependant, de nombreuses autres solutions sont utilisées dans le milieu professionnel.

En entreprise, divers systèmes d'exploitation sont utilisés. Effectivement, on retrouve notamment des serveurs Linux et Windows, mais aussi des postes clients que ce soit Linux avec ou sans interface graphique, ainsi que des postes avec Windows.

Pour ce projet, trois solutions gratuites et open source qui répondent à des besoins complémentaires ont été sélectionnées. Le choix s'est basé sur plusieurs critères: la compatibilité multiplateforme (permettant des connexions croisées entre Linux et Windows), la diversité des approches techniques (protocoles RDP et VNC, architecture client-serveur et accès web), et la pertinence au niveau professionnel (solutions réellement utilisées en entreprise pour le travail quotidien, le support technique et l'administration centralisée).

Ces solutions permettent également d'approfondir la compréhension des protocoles de connexion à distance et des concepts d'administration système.

Nous allons donc voir particulièrement trois solutions de connexion à distance: *Remmina - xrdp - rdp*, *UltraVNC - x11vnc* et *Apache Guacamole,* et tester les connexions entre les différents postes présents en entreprise.

4.1. Remmina - xrdp - Bureau à distance

xrdp est un serveur RDP pour Linux. Il permet d'accéder à un bureau graphique Linux à distance, comme si l'utilisateur était physiquement devant la machine. Grâce à *xrdp*, un poste Windows peut utiliser l'outil intégré *Bureau à distance* pour se connecter à un serveur Linux, et un poste Linux peut se connecter via un client RDP, comme *Remmina*. *xrdp* traduit les protocoles RDP en commandes compréhensibles par l'environnement graphique Linux, ce qui rend possible une connexion fluide entre systèmes différents.

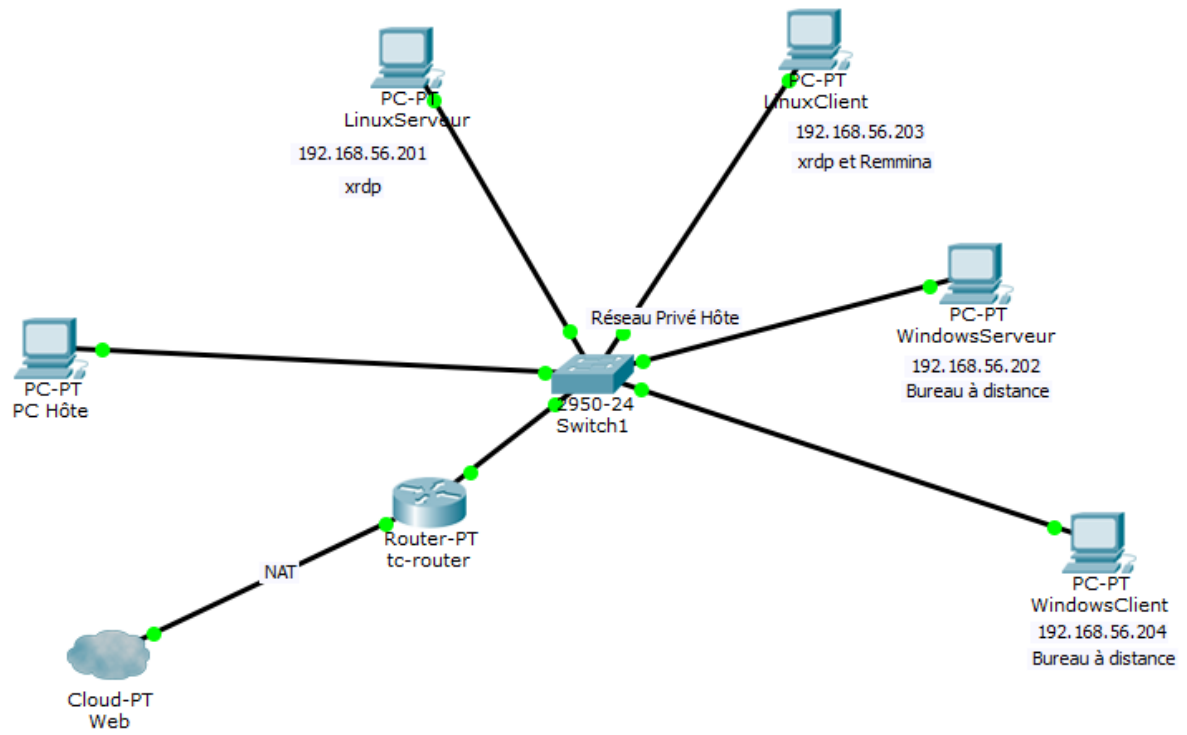
Remmina est un client de connexion à distance pour Linux. Il supporte plusieurs protocoles : RDP (Windows ou *xrdp*), VNC, SSH... Dans notre contexte, *Remmina* sera principalement utilisé comme client RDP, permettant :

- de se connecter à un poste Windows avec le *Bureau à distance* activé,
- de se connecter à un poste Linux disposant de *xrdp*, permettant ainsi une compatibilité bidirectionnelle entre Linux et Windows.

Bureau à distance (ou Remote Desktop sur Windows) est une fonctionnalité native de Windows qui permet d'accéder à un poste Windows depuis un autre ordinateur via le protocole RDP. Associé à *xrdp* et *Remmina*, il permet de prendre le contrôle et d'utiliser un ordinateur distant de manière équivalente à une session locale, avec accès au bureau, aux applications et aux fichiers.

4.1.1. Schéma du réseau virtuel

Pour mieux comprendre la structure et la communication entre les différentes machines lors de l'utilisation de *Remmina*, *xrdp* et *Bureau à distance*, un schéma réseau virtuel présentant les différentes installations a été réalisé:



Pour nos expérimentations, nous allons installer *xrdp* sur *LinuxClient* et *LinuxServer*, *Remmina* sur *LinuxServeur* et utiliser *Bureau à distance* qui est natif sur *WindowsClient* et *WindowsServeur*.

4.1.2. Installation

Avant d'utiliser cette solution, il est nécessaire d'installer les outils indispensables sur les machines concernées. Nous allons donc détailler les étapes d'installation de *xrdp* et de *Remmina*.

4.1.2.1. Installation de xrdp

La connexion à distance depuis une machine Windows vers une machine Linux, ainsi que depuis une machine Linux vers une autre machine Linux, nécessite d'installer *xrdp* sur toutes les machines Linux auxquelles on souhaite accéder.

Pour cela, voici les étapes à suivre :

Etape	Description
1	Mettre à jour les paquets avec la commande **apt update**.
	Représentation
	`root@linuxserveur:~# apt update`
Etape	Description
2	Installer *xrdp* avec la commande **apt install xrdp**.
	Représentation
	`root@linuxserveur:~# apt install xrdp`
Etape	Description
3	Démarrer le service *xrdp* avec **systemctl start xrdp**.
	Représentation
	`root@linuxserveur:~# systemctl start xrdp`
Etape	Description
4	Activer le service *xrdp* au démarrage avec **systemctl enable xrdp**.

	Représentation
	`root@linuxserveur:~# systemctl enable xrdp`
Etape	Description
5	Ajouter l'utilisateur "sio" au groupe SSL pour l'accès RDP avec **adduser sio ssl-cert**.
	Représentation
	`root@linuxserveur:~# adduser sio ssl-cert`

4.1.2.2. Installation de Remmina

Une fois *xrdp* configuré sur les serveurs, il faut installer *Remmina* sur les postes clients Linux. Ce logiciel servira à initier les connexions vers les différentes machines du réseau.

Pour y arriver, il suffit de suivre simplement les deux étapes suivantes:

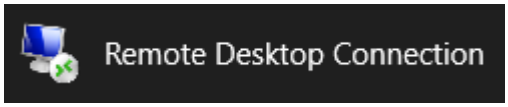
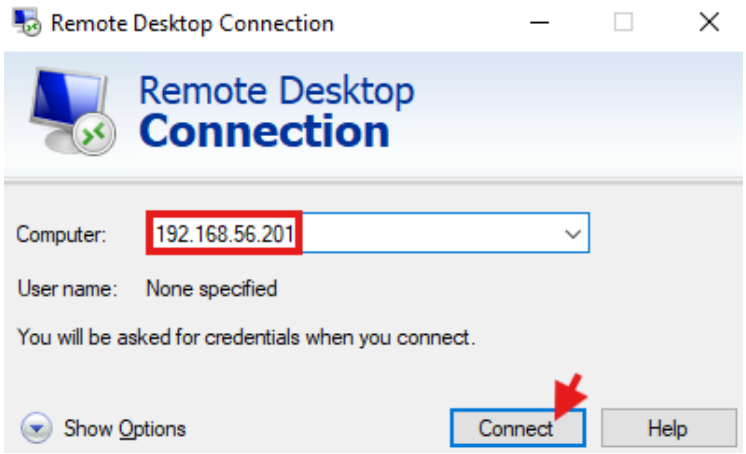
Etape	Description
1	Mettre à jour les paquets avec la commande **apt update**.
	Représentation
	`root@linuxclient:~# apt update`
Etape	Description
2	Installer Remmina avec la commande **apt install remmina**.
	Représentation
	`root@linuxclient:~# apt install remmina`

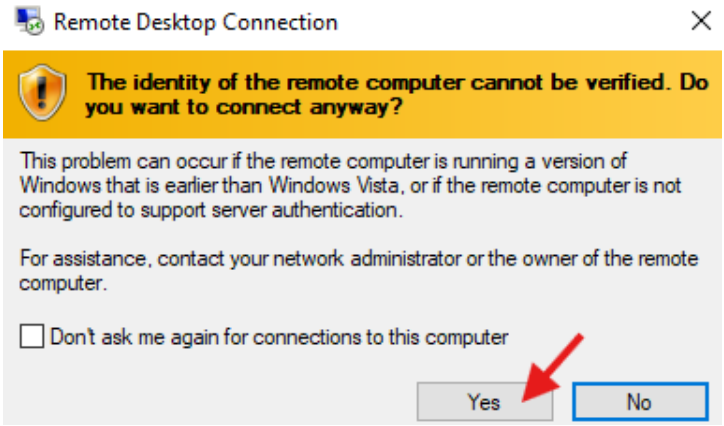
4.1.3. Utilisation

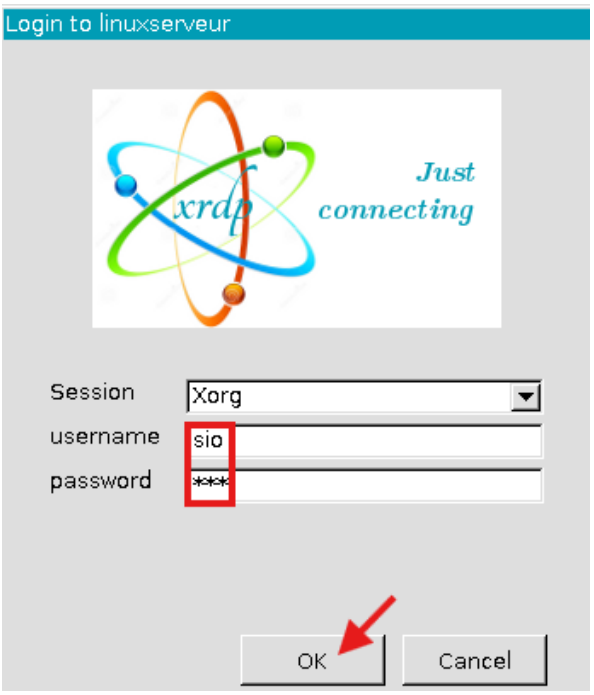
Après l'installation des outils nécessaires, il est temps de tester la connexion entre les différents systèmes. Les cas suivants présentent les connexions possibles entre machines Windows et Linux.

4.1.3.1. Connexion depuis WindowsClient vers LinuxServeur

Tout d'abord, il est intéressant de voir comment un poste Windows peut se connecter à un serveur Linux via le protocole RDP grâce à *Bureau à distance* installé nativement sur Windows et *xrdp* installé sur Linux à l'étape 4.1.2.1:

Etape	Description
1	Ouvrir l'application "Connexion Bureau à distance".
	Représentation
	
Etape	Description
2	Entrer l'adresse IP du serveur Linux, dans notre cas, 192.168.56.201 et cliquer sur "Connect".
	Représentation
	

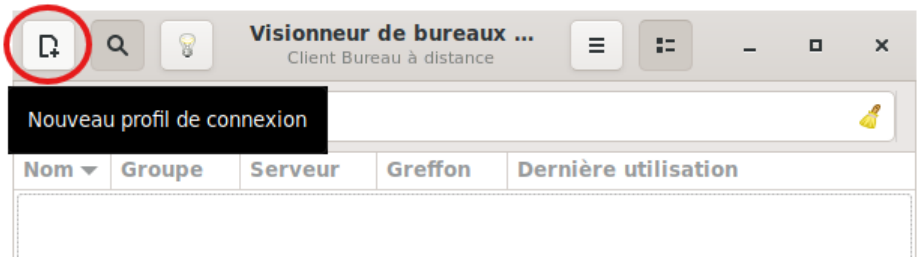
Etape	Description
3	Accepter le certificat de sécurité si demandé.
	Représentation
	

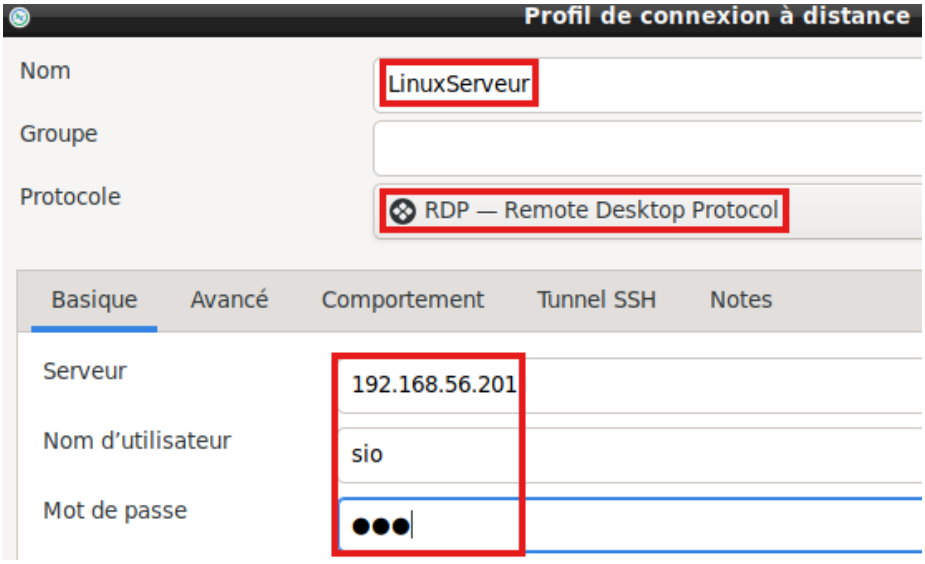
Etape	Description
4	Saisir le nom d'utilisateur Linux et le mot de passe de l'utilisateur Linux, puis cliquer sur "OK".
	Représentation
	

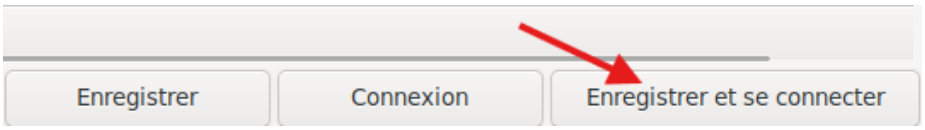
Nous pouvons observer que la connexion de *WindowsClient* vers *LinuxServeur* a bien fonctionné:

4.1.3.2. Connexion depuis LinuxClient vers LinuxServeur

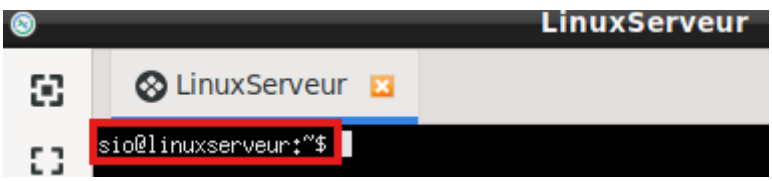
Dans un second temps, la connexion peut s'effectuer entre deux machines Linux, dans notre cas de *LinuxClient* vers *LinuxServeur*:

Etape	Description
1	Lancer Remmina avec **remmina &.**
	Représentation
	`root@linuxclient:~# remmina &`
Etape	Description
2	Cliquer sur l'icône "Nouveau profil de connexion".
	Représentation
	

Etape	Description
3	Entrer un nom de connexion *LinuxServeur*, choisir le protocole *RDP*, entrer l'adresse IP du serveur *192.168.56.201*, le nom de l'utilisateur *sio* et le mot de passe de l'utilisateur.
	Représentation
	

Etape	Description
4	Cliquer sur "Enregistrer et se connecter".
	Représentation
	

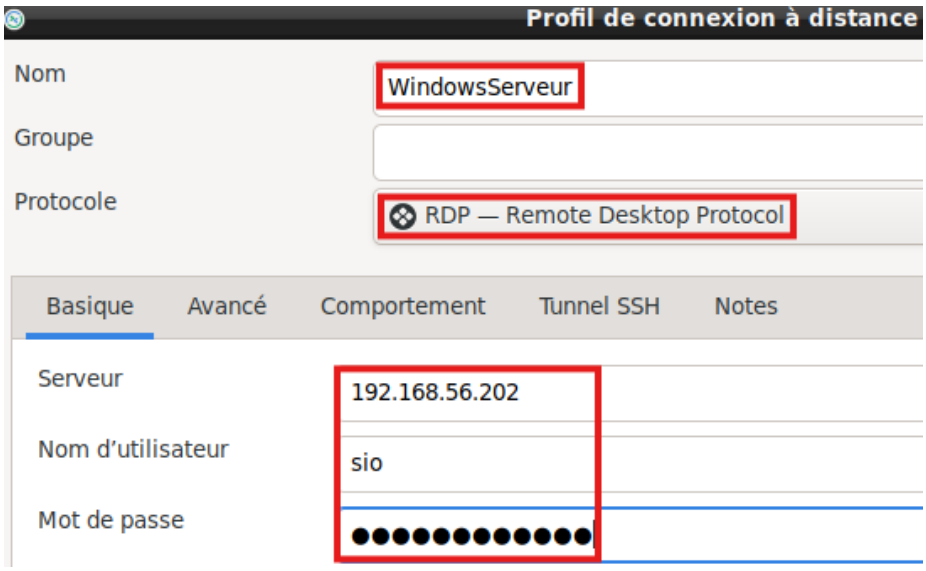
Nous pouvons observer que la connexion de *LinuxClient* vers *LinuxServeur* a bien fonctionné:

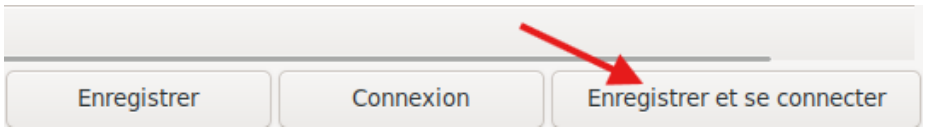


4.1.3.3. Connexion depuis LinuxClient vers WindowsServeur

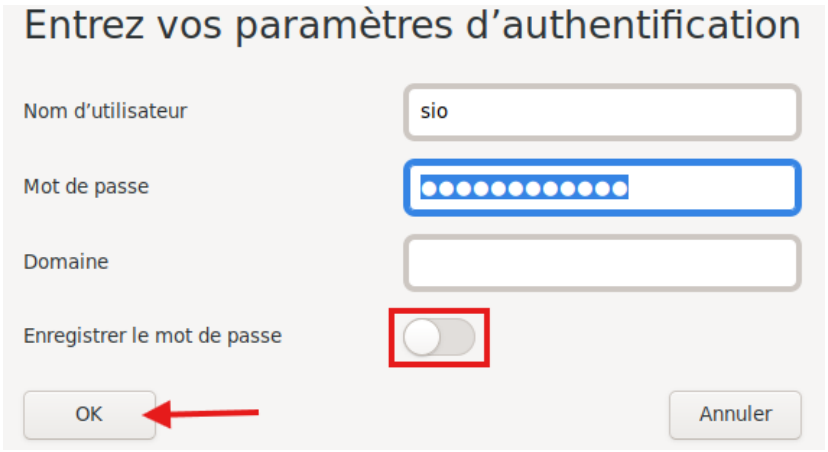
Il est également possible, depuis un poste Linux, d'accéder à un poste Windows via RDP, toujours grâce à Remmina:

Etape	Description
1	Lancer Remmina avec **remmina &.**
	Représentation
	`root@linuxclient:~# remmina &`
Etape	Description
2	Cliquer sur l'icône "Nouveau profil de connexion".
	Représentation
	
Etape	Description
3	Entrer un nom de connexion *WindowsServeur*, choisir le protocole *RDP*, entrer l'adresse IP du serveur *192.168.56.202*, le nom de l'utilisateur *sio* et le mot de passe de l'utilisateur.

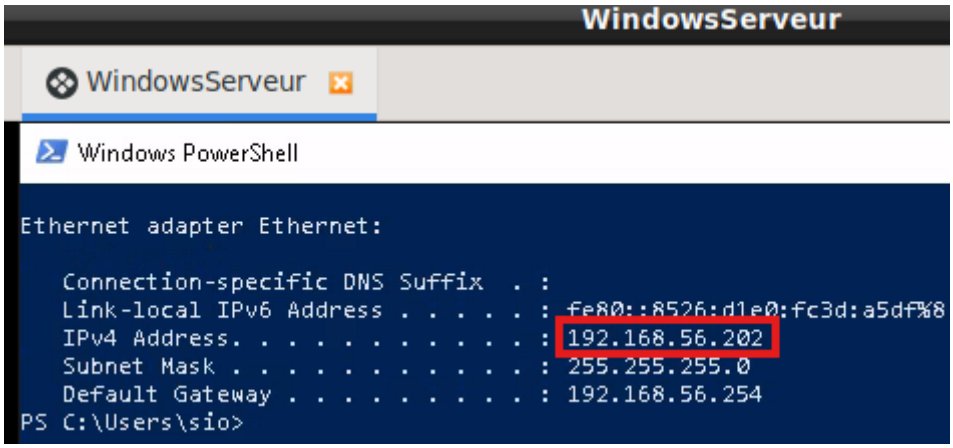
	Représentation
	

Etape	Description
4	Cliquer sur “Enregistrer et se connecter”.
	Représentation
	

Etape	Description
5	Accepter l’avertissement de certificat.
	Représentation
	

Etape	Description
6	Décocher la case “Enregistrer le mot de passe” pour plus de sécurité et cliquer sur “OK”.
	Représentation
	

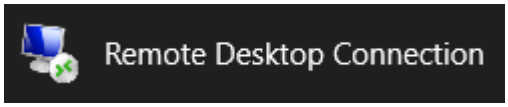
Nous pouvons à présent observer que la connexion de *LinuxClient* vers *WindowsServeur* a bien fonctionné:

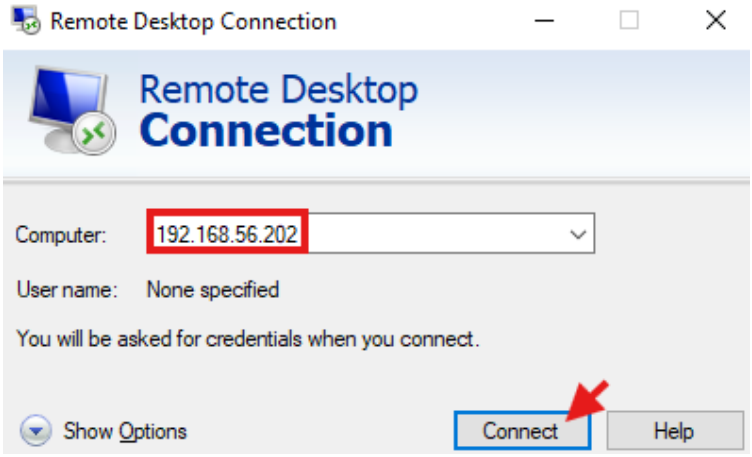


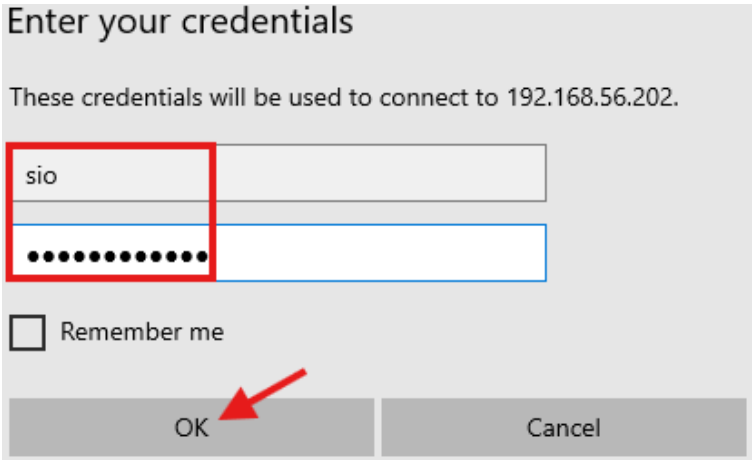
4.1.3.4. Connexion depuis WindowsClient vers WindowsServeur

Enfin, en entreprise, il peut être nécessaire de se connecter d’un poste client vers un serveur, dans notre cas de *WindowsClient* vers *WindowsServeur*:

Etape	Description
1	Ouvrir l’application “Bureau à distance”.

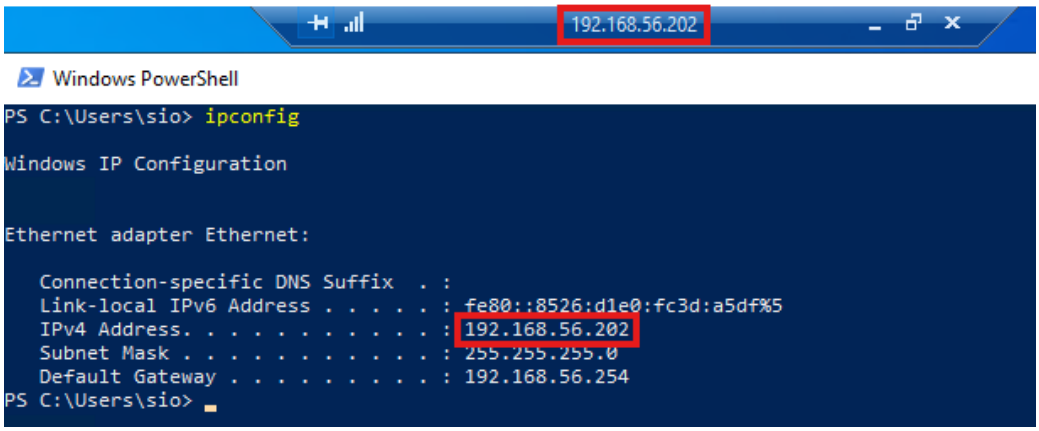
	Représentation
	

Etape	Description
2	Entrer l'adresse IP du serveur, dans notre cas, 192.168.56.202 et cliquer sur “Connect”.
	Représentation
	

Etape	Description
3	Saisir le nom de l’utilisateur *sio* et le mot de passe de l’utilisateur, puis cliquer sur “OK”.
	Représentation
	

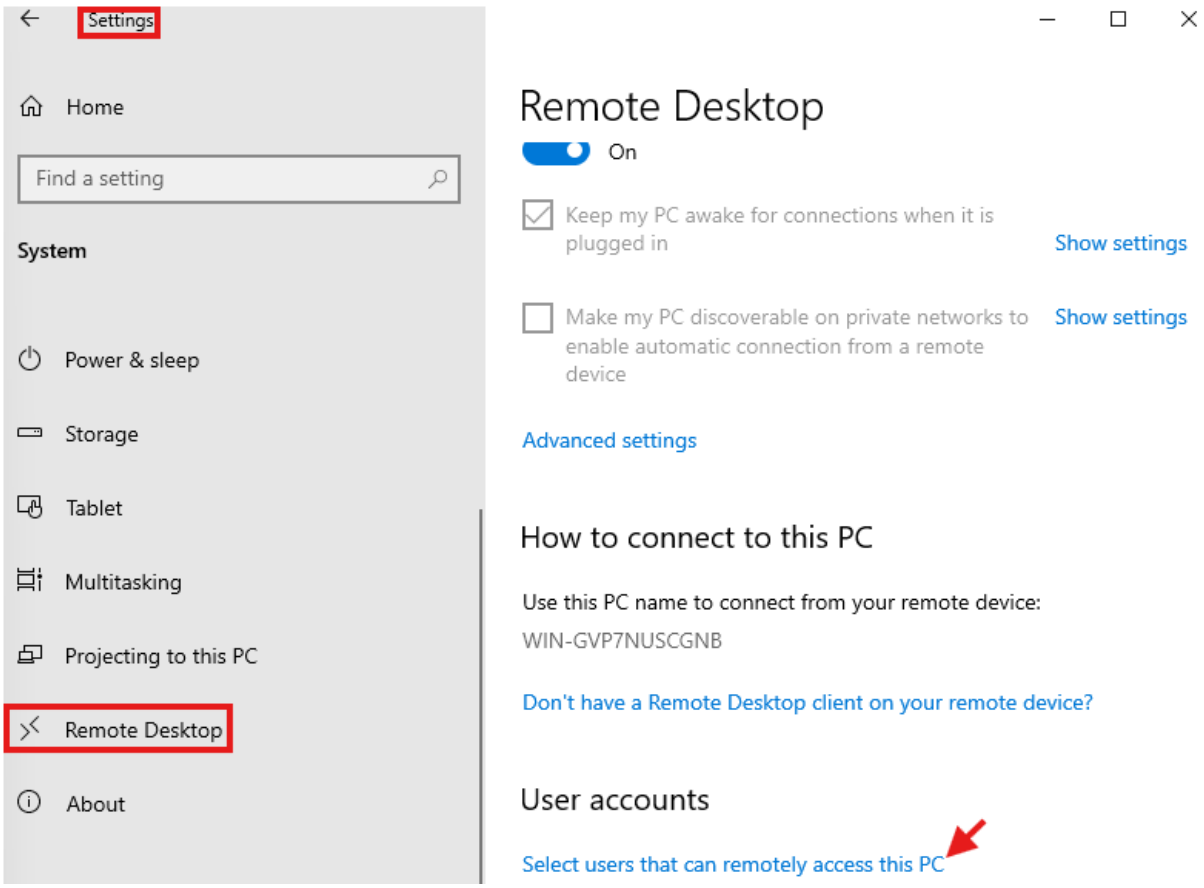
Etape	Description
4	Accepter le certificat de sécurité si demandé.
	Représentation
	

Nous pouvons observer que la connexion de *WindowsClient* vers *WindowsServeur* a bien fonctionné:

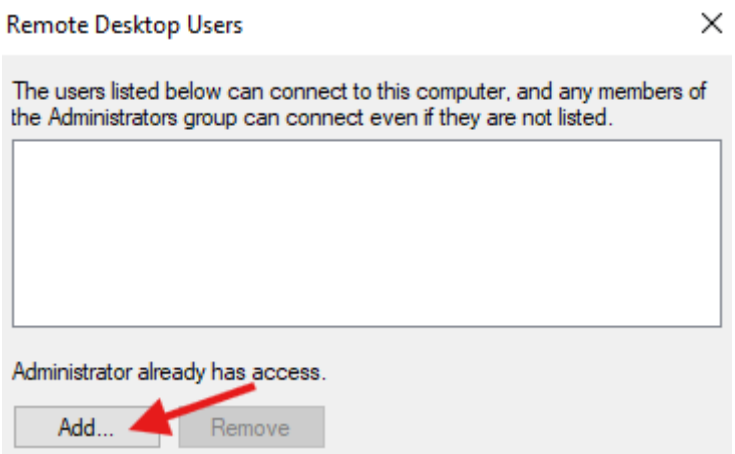


Attention, pour pouvoir se connecter à un compte non-administrateur, il faut d'abord autoriser la connexion RDP pour ce compte.

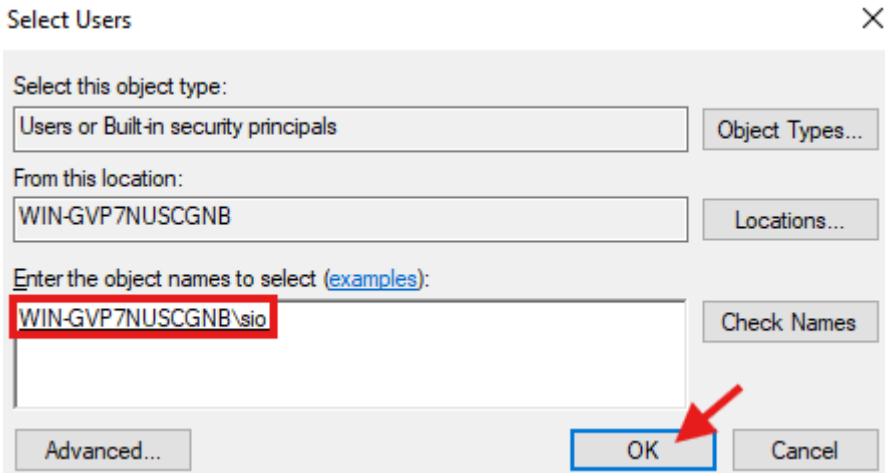
Pour cela, il faut d'abord aller dans les paramètres de la machine, dans notre cas *WindowsServeur*, puis dans *System* et *Remote Desktop* et enfin, cliquer sur "Select users that can remotely access this PC":



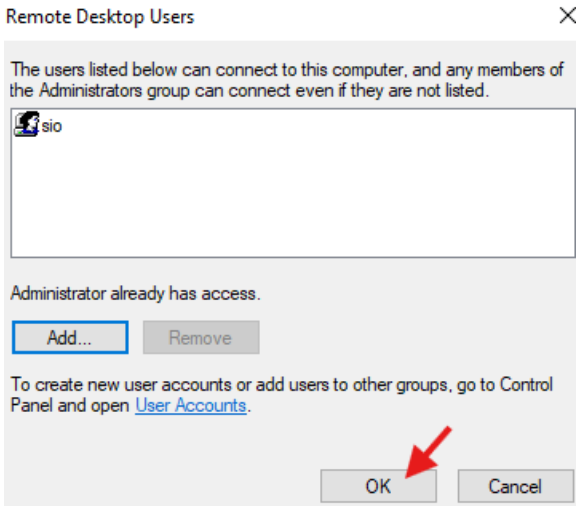
Après cela, cliquer sur "Add":



Ensuite, chercher l'utilisateur à ajouter et cliquer sur "OK":



Enfin, cliquer à nouveau sur "OK":



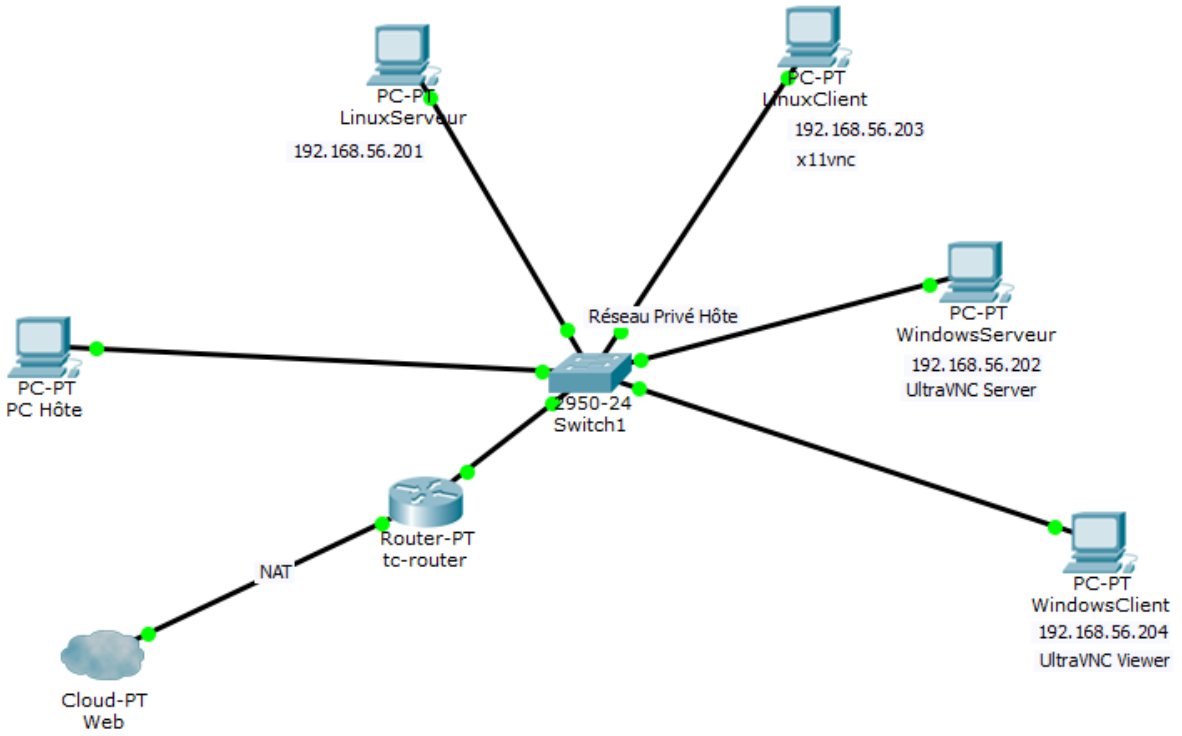
4.2. UltraVNC - x11vnc

UltraVNC est une solution open source qui permet de prendre le contrôle à distance d'un ordinateur via le protocole. Elle permet d'afficher et d'interagir avec le bureau d'un ordinateur distant comme si l'utilisateur était devant la machine. *UltraVNC* se compose d'un serveur, installé sur l'ordinateur à contrôler, et d'un client, installé sur l'ordinateur distant. Il est principalement conçu pour Windows et présente une interface simple pour le support technique, l'administration à distance et la collaboration.

Pour Linux, il existe des alternatives comme *x11vnc*, qui permet d'accéder à distance à l'interface graphique d'une machine Linux. Cependant, il n'est pas possible de se connecter à un serveur Linux via VNC si celui-ci n'a pas d'interface graphique installée, car VNC nécessite un environnement graphique pour afficher le bureau et interagir avec lui.

4.2.1. Schéma du réseau virtuel

Comme pour la première solution, un schéma du réseau virtuel présentant les différentes installations permet de visualiser l'infrastructure dans le cadre de l'utilisation d'*UltraVNC*:



Dans notre cas, nous allons installer *x11vnc* sur *LinuxClient*, *UltraVNC Server* sur *WindowsServer* et *UltraVNC Viewer* sur *WindowsClient*.

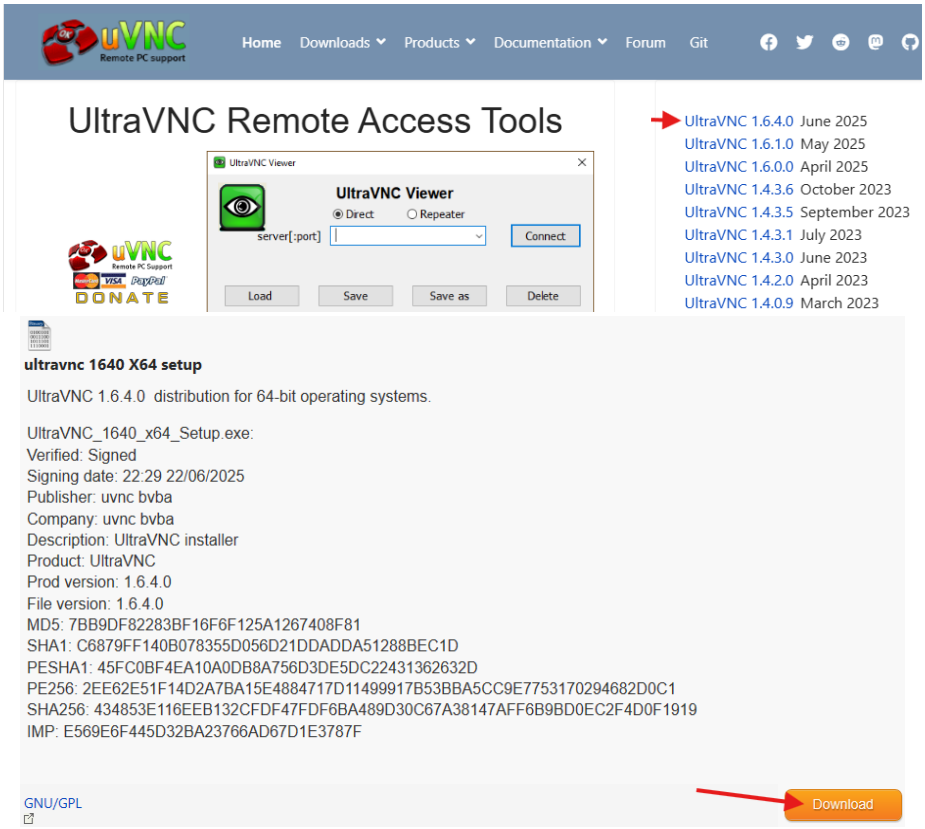
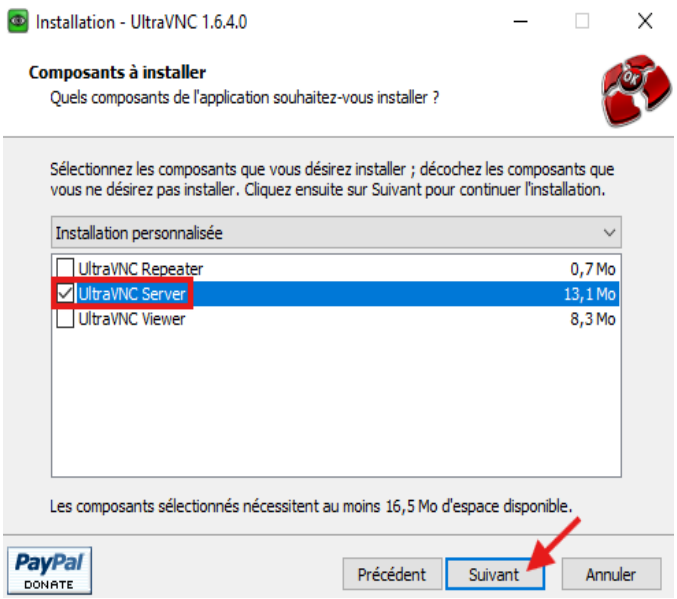
4.2.2. Installation d'UltraVNC

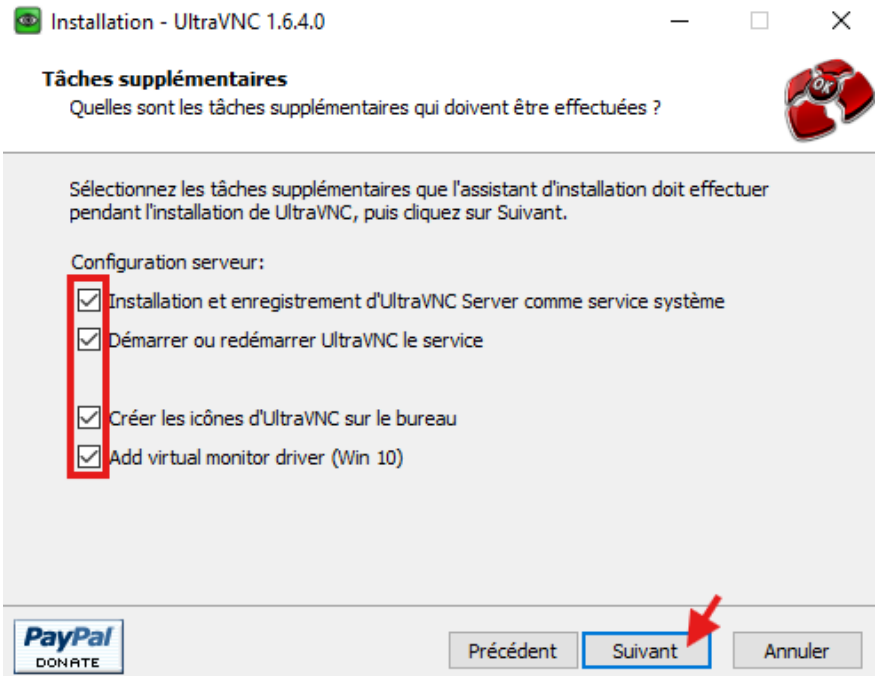
L'installation d'*UltraVNC* se déroule en deux parties : d'abord sur le serveur à contrôler, puis sur le poste client qui lancera la connexion.

4.2.2.1. Installation sur WindowsServeur

L'installation sur la machine serveur permet de déployer UltraVNC Server, qui rend possible le contrôle à distance de ce poste. Dans notre cas, nous allons réaliser cela sur *WindowsServeur*:

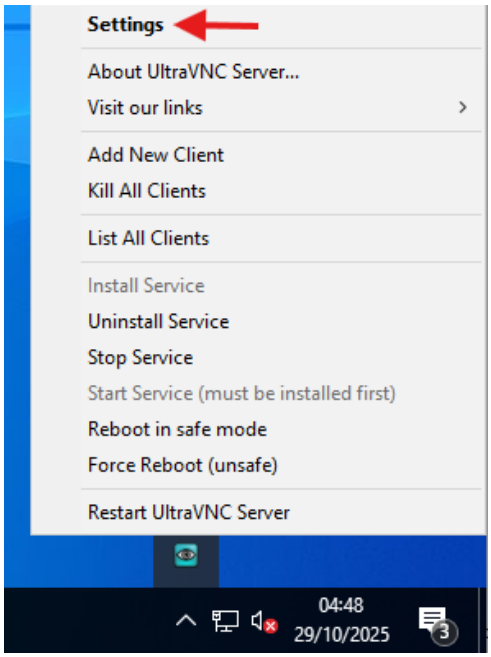
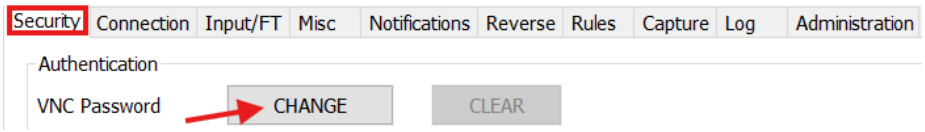
Etape	Description
1	Télécharger UltraVNC depuis uvnc.com

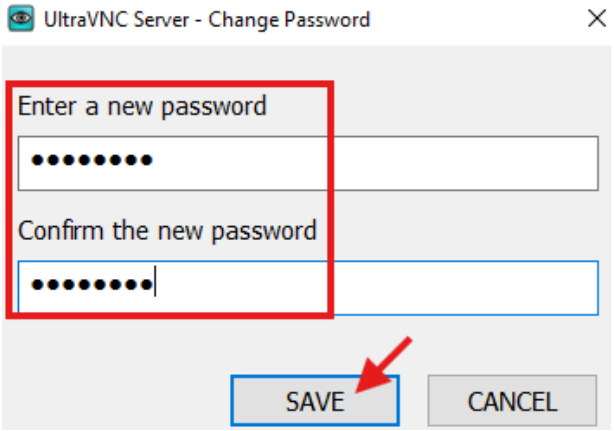
	Représentation
	
Etape	Description
2	Lancer l'installation et lorsque cela est demandé, cocher seulement "UltraVNC Server" et cliquer sur "Suivant".
	Représentation
	

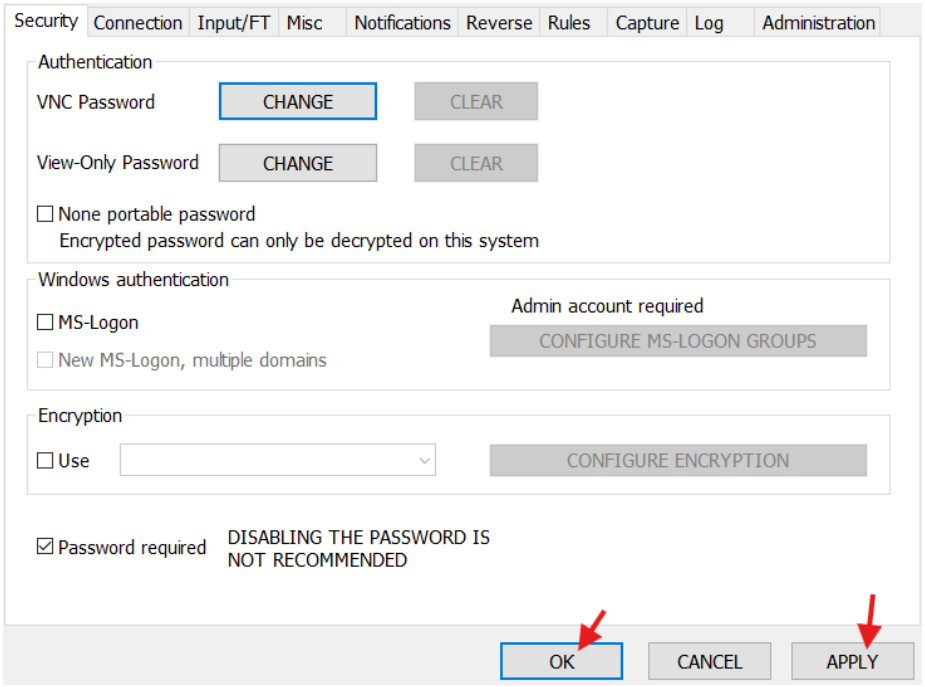
Etape	Description
3	Cocher toutes les cases, lorsque cela est demandé et cliquer sur “Suivant”. La première case installe *UltraVNC Server* comme un service Windows, c’est-à-dire qu’il démarre automatiquement au démarrage de la machine. La deuxième permet de lancer le service *UltraVNC* immédiatement après l’installation. La troisième ajoute le raccourci vers *UltraVNC Server* sur le bureau. Enfin, la dernière case permet d'installer un pilote d’écran virtuel permettant d’afficher un bureau même si aucun moniteur n’est branché sur la machine.
	Représentation
	

Une fois l’installation réalisée, le raccourci vers *UltraVNC Server* apparaît sur le bureau:

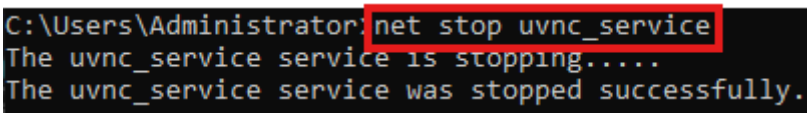
Il est ensuite nécessaire de choisir un mot de passe UltraVNC qui permettra à *WindowsClient* d'avoir accès à *WindowsServeur*:

Etape	Description
1	Faire clic-droit sur l'icône *UltraVNC* en bas à droite de l'écran et choisir "Settings".
	Représentation
	
Etape	Description
2	Dans l'onglet "Security", cliquer sur "CHANGE" pour changer le "VNC Password".
	Représentation
	
Etape	Description
3	Entrer un mot de passe pour *UltraVNC* et cliquer sur "SAVE".

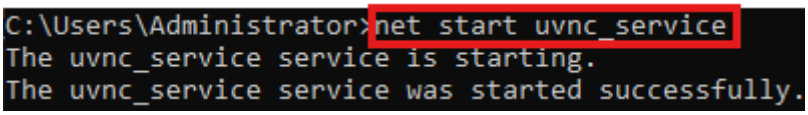
	Représentation
	

Etape	Description
4	Enfin, cliquer sur “APPLY” et “OK”.
	Représentation
	

Enfin, il est nécessaire d’arrêter le service *UltraVNC* avec la commande **net stop uvnc_service**:



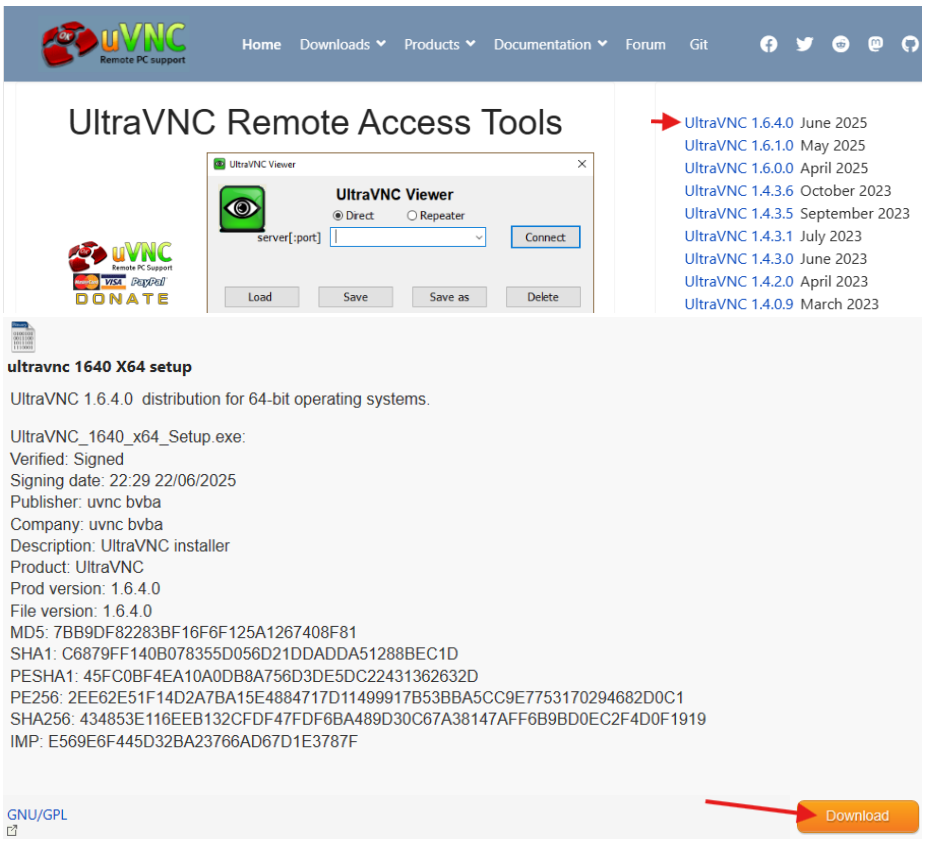
Puis, de le redémarrer avec la commande **net start uvnc_service**:



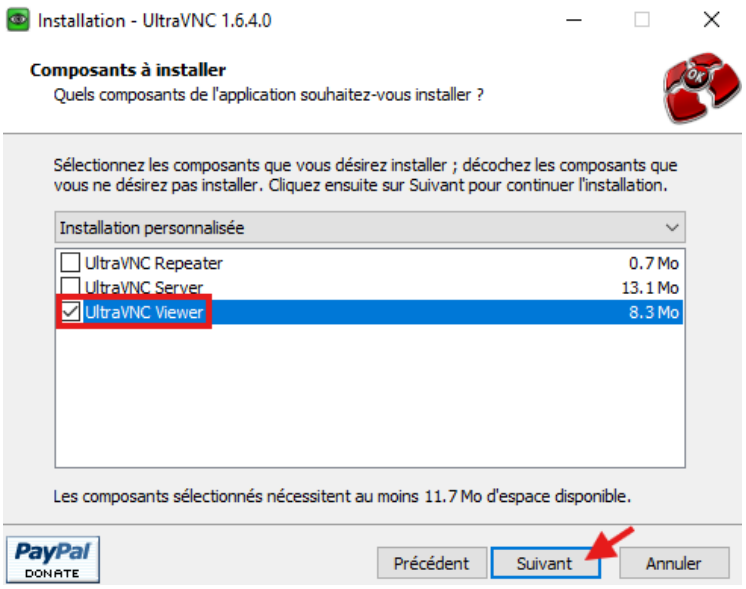
4.2.2.2. Installation sur WindowsClient

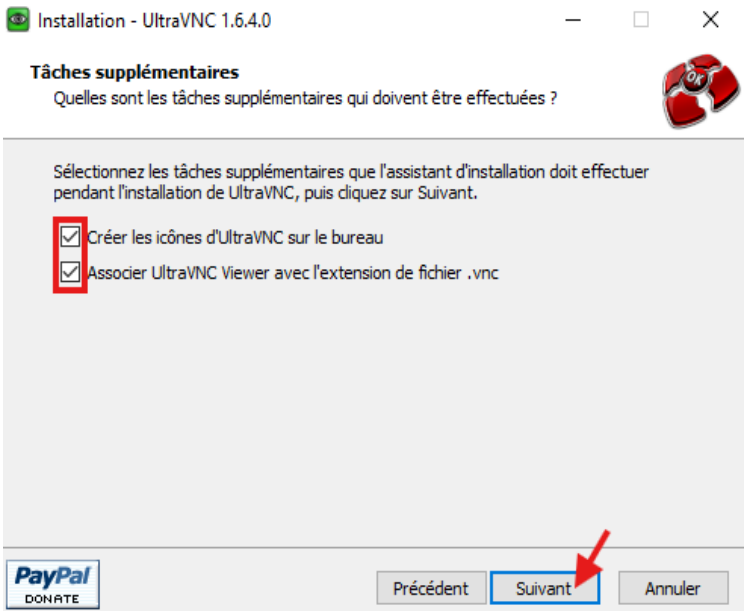
L'installation du client *UltraVNC Viewer* sur la machine distante permet ensuite de se connecter au serveur et d'interagir avec son interface graphique.

Pour y arriver, il est nécessaire de suivre la procédure suivante pour les installations sur *WindowsClient*:

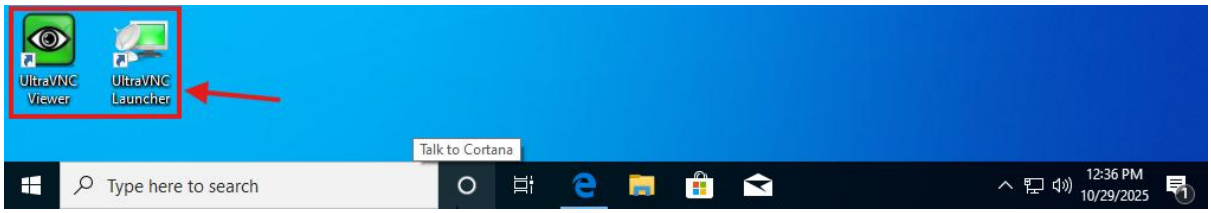
Etape	Description
1	Télécharger UltraVNC depuis uvnc.com
	Représentation
	

Etape	Description
2	Lancer l'installation et lorsque cela est demandé, cocher seulement "UltraVNC Viewer" et cliquer sur "Suivant".

	Représentation
	

Etape	Description
3	Cocher toutes les cases, lorsque cela est demandé et cliquer sur “Suivant”. La première case ajoute les raccourcis vers *UltraVNC Viewer* sur le bureau. La deuxième signifie que tous les fichiers avec l’extension .vnc (qui contiennent des configurations de connexion VNC enregistrées) seront automatiquement ouverts avec *UltraVNC Viewer* en double-cliquant dessus.
	Représentation
	

Une fois l'installation réalisée, le raccourci vers *UltraVNC Viewer* ainsi que celui vers *UltraVNC Launcher* apparaissent sur le bureau:



4.2.3. Installation de x11vnc

La mise en place du serveur *x11vnc* sur la machine *LinuxClient* permet d'établir une connexion distante et d'interagir avec son interface graphique.

Pour installer *x11vnc*, il est nécessaire de suivre la procédure suivante pour les installations sur *LinuxClient*:

Etape	Description
1	Mettre à jour les paquets avec la commande **apt update**.
	Représentation
	`root@linuxclient:~# apt update`
Etape	Description
2	Ensuite, installer le serveur VNC x11vnc avec **apt install x11vnc**.
	Représentation
	`root@linuxclient:~# apt install x11vnc`
Etape	Description
3	Définir un mot de passe avec **x11vnc -storepasswd**.
	Représentation
	`root@linuxclient:~# x11vnc -storepasswd`
Etape	Description
4	Entrer un mot de passe VNC et le vérifier.

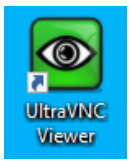
	Représentation
	Enter VNC password: Verify password:
Etape	Description
5	Écrire le mot de passe dans le fichier passwd de vnc en entrant **y**.
	Représentation
	Write password to /root/.vnc/passwd? [y]/n y
Etape	Description
6	Lancer le serveur avec **x11vnc -forever -usepw -display :0.**
	Représentation
	root@linuxclient:~# x11vnc -forever -usepw -display :0

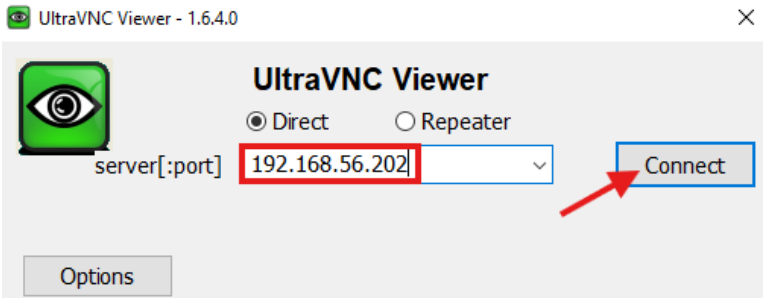
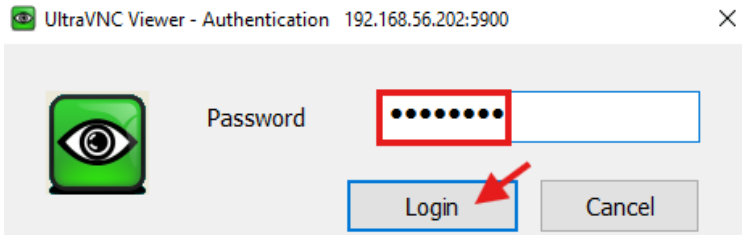
4.2.4. Utilisation

Après l'installation des outils nécessaires, il est temps de tester la connexion entre les différents systèmes. Les scénarios suivants présentent les connexions entre machines Windows et Linux.

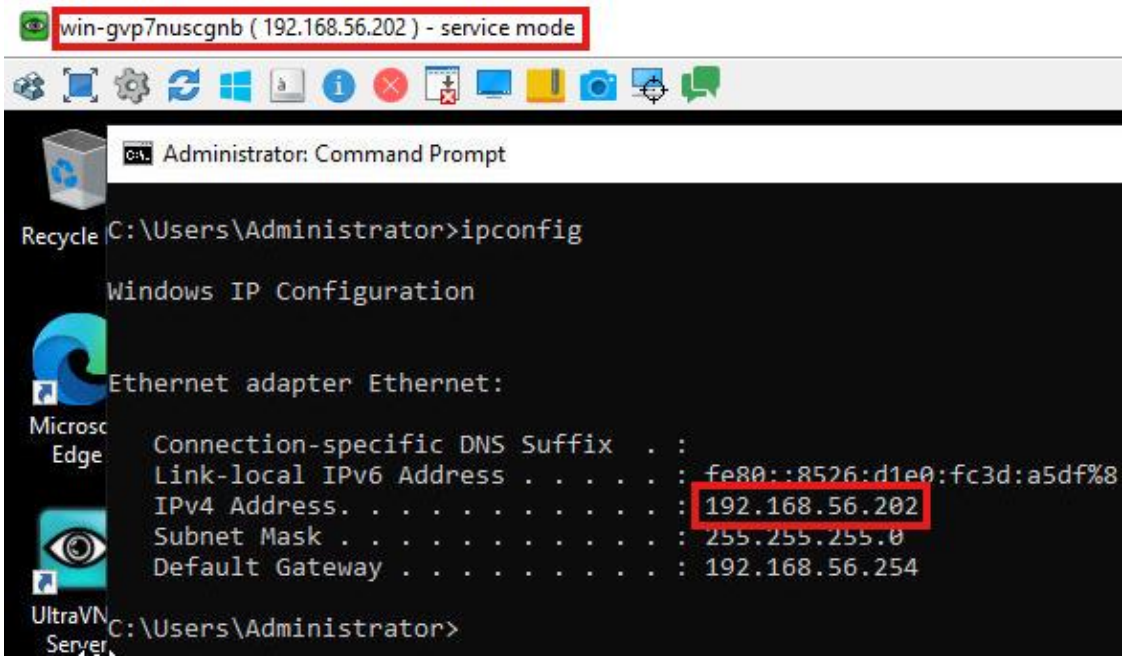
4.2.4.1. Connexion de WindowsClient vers WindowsServeur

Une fois *UltraVNC* installé, il est possible d'établir une connexion depuis le poste client, dans notre cas *WindowsClient,* vers le serveur *WindowsServeur*. Les étapes suivantes montrent la démarche complète de connexion et d'authentification:

Etape	Description
1	Ouvrir *UltraVNC Viewer*.
	Représentation
	

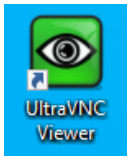
<table>
<tr><th>Etape</th><th>Description</th></tr>
<tr><td rowspan="3">2</td><td>Entrer l'adresse IP du serveur, dans notre cas, 192.168.56.202 et cliquer sur “Connect”.
En cliquant sur “Options”, il est possible d’accéder à une multitude d'options en lien avec la connexion VNC, mais dans notre cas, nous ne changerons rien.</td></tr>
<tr><td>Représentation</td></tr>
<tr><td> 
</td></tr>
<tr><th>Etape</th><th>Description</th></tr>
<tr><td rowspan="3">3</td><td>Entrer le mot de passe UltraVNC définit lors de l’installation de UltraVNC Server sur WindowsServeur et cliquer sur “Login”.</td></tr>
<tr><td>Représentation</td></tr>
<tr><td> 
</td></tr>
</table>

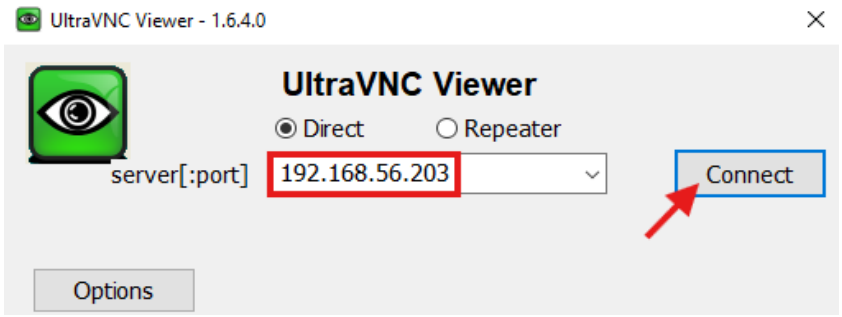
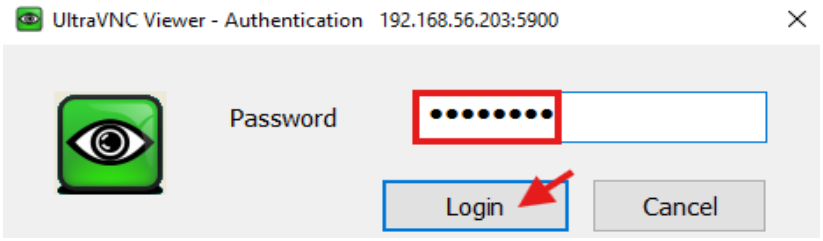
Nous pouvons observer que la connexion de *WindowsClient* vers *WindowsServeur* a bien fonctionné:



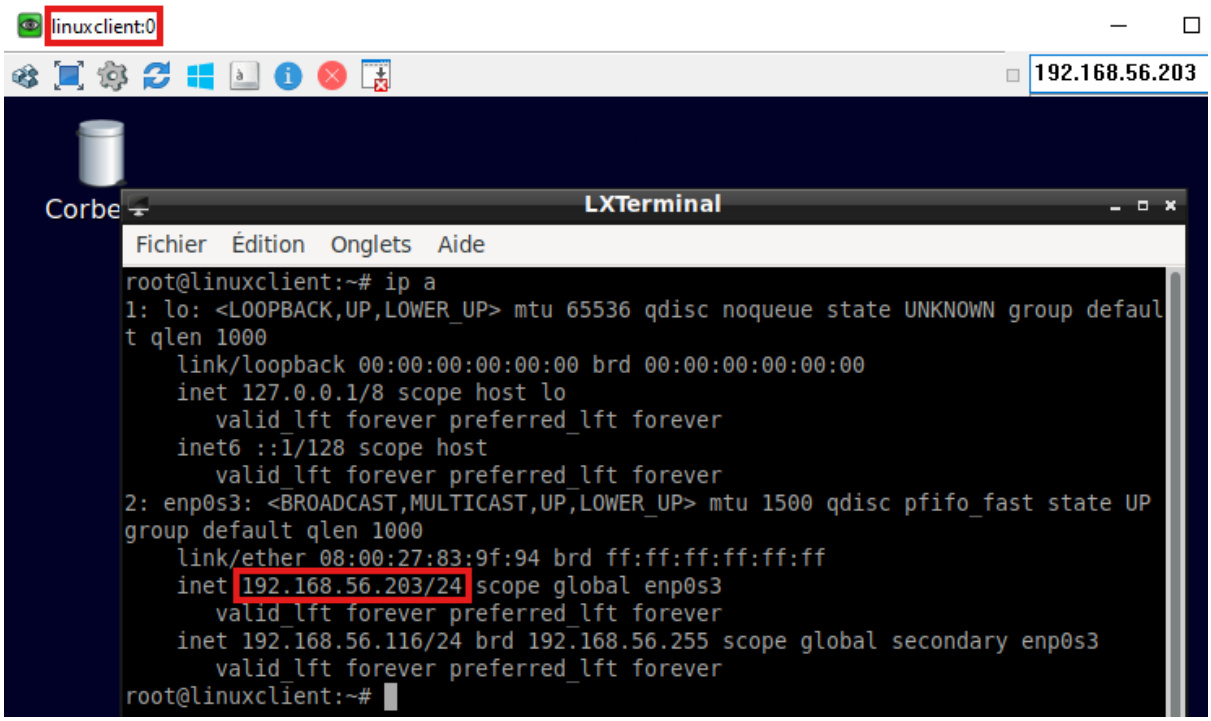
4.2.4.2. Connexion de WindowsClient vers LinuxClient

Une fois *UltraVNC* installé, il est possible d'établir une connexion depuis le poste client, dans notre cas *WindowsClient*, vers *LinuxClient*, sur lequel nous avons installé *x11vnc*. Les étapes suivantes montrent la démarche complète de connexion et d'authentification:

Etape	Description
1	Ouvrir *UltraVNC Viewer*.
	Représentation
	
Etape	Description
2	Entrer l'adresse IP du client, dans notre cas, 192.168.56.203 et cliquer sur "Connect". En cliquant sur "Options", il est possible d'accéder à une multitude d'options en lien avec la connexion VNC, mais dans notre cas, nous ne changerons rien.

	Représentation
	
Etape	**Description**
3	Entrer le mot de passe VNC définit lors de l'installation de *x11vnc* sur *LinuxClient* et cliquer sur "Login".
	Représentation
	

Nous pouvons observer que la connexion de *WindowsClient* vers *LinuxClient* a bien fonctionné:



4.3. Apache Guacamole

Apache Guacamole est une solution libre de connexion à distance, dite *clientless* (sans client local à installer).

Elle permet d'accéder à un bureau distant via un simple navigateur web.

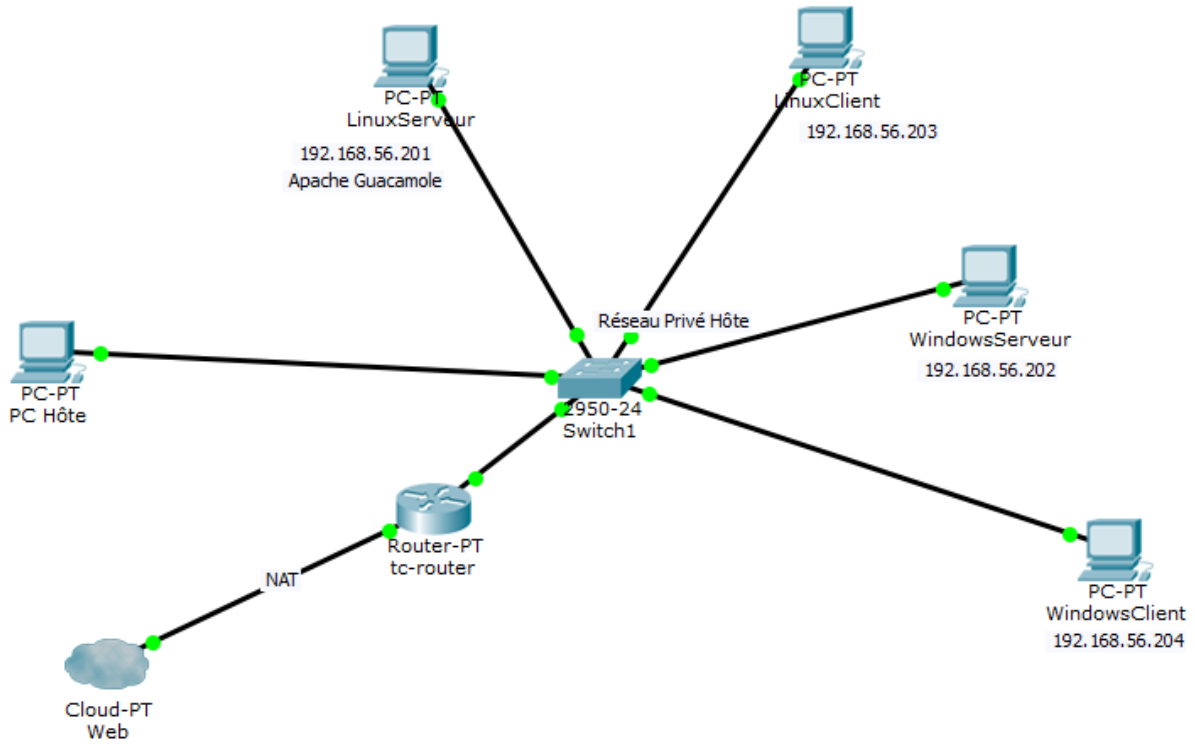
Guacamole supporte les protocoles RDP, VNC et SSH.

Cette solution se compose de trois éléments principaux :

- *guacd* : le daemon d'exécution qui fait le lien entre le navigateur et la machine distante.
- *Tomcat* : le serveur web Java qui héberge l'application web.
- *guacamole.war* : le fichier de l'application web elle-même.

4.3.1. Schéma du réseau virtuel

Comme pour les précédentes solutions, un schéma du réseau virtuel permet de montrer l'infrastructure, cette fois-ci en lien avec l'installation d'*Apache Guacamole*:



Pour nos expérimentations, nous allons simplement installer *Apache Guacamole* sur *LinuxServer* et nous pourrons y accéder depuis un navigateur web.

4.3.2. Installation

L'installation d'*Apache Guacamole* est plus complexe que les précédentes, car elle implique plusieurs composants: *Tomcat*, *guacd* et l'interface web.

Nous allons passer par différentes étapes pour réaliser cette installation sur *LinuxServeur*.

4.3.2.1. Installation des prérequis

Avant de pouvoir installer *Guacamole*, il est nécessaire de mettre en place certains éléments fondamentaux, tels que le serveur *Tomcat* et les bibliothèques de compilation nécessaires à *guacd*:

Etape	Description
1	Mettre à jour les paquets avec la commande **apt update**.
	Représentation
	`root@linuxserveur:~# apt update`
Etape	Description
2	Installer le serveur *Tomcat* (serveur web Java) avec **apt install tomcat9 tomcat9-admin -y**.
	Représentation
	`root@linuxserveur:~# apt install tomcat9 tomcat9-admin -y`
Etape	Description
3	Installer les paquets nécessaires pour *guacd* avec la commande **apt install build-essential libcairo2-dev libjpeg62-turbo-dev libpng-dev libtool-bin libossp-uuid-dev freerdp2-dev libssh2-1-dev libtelnet-dev libvncserver-dev libwebsockets-dev -y**
	Représentation
	`root@linuxserveur:~# apt install build-essential libcairo2-dev libjpeg62-turbo-dev libpng-dev libtool-bin libossp-uuid-dev freerdp2-dev libssh2-1-dev libtelnet-dev libvncserver-dev libwebsockets-dev -y`

4.3.2.2. Installation de guacd

Le daemon *guacd* constitue la base du système *Guacamole*. Il agit comme un intermédiaire entre le navigateur web et la machine distante.

Il est donc indispensable de l'installer et de le configurer à l'aide des étapes suivantes:

Etape	Description
1	Télécharger les sources d'*Apache Guacamole* avec la commande **wget https://downloads.apache.org/guacamole/1.5.5/source/guacamole-server-1.5.5.tar.gz**
	Représentation
	`root@linuxserveur:~# wget https://downloads.apache.org/guacamole/1.5.5/source/guacamole-server-1.5.5.tar.gz`
Etape	Description
2	Extraire l'archive téléchargée avec la commande **tar -xzf guacamole-server-1.5.5.tar.gz**
	Représentation
	`root@linuxserveur:~# tar -xzf guacamole-server-1.5.5.tar.gz`
Etape	Description
3	Entrer dans le dossier extrait avec **cd guacamole-server-1.5.5**
	Représentation
	`root@linuxserveur:~# cd guacamole-server-1.5.5`
Etape	Description
4	Compiler le serveur en tapant **./configure --with-init-dir=/etc/init.d**
	Représentation
	`root@linuxserveur:~/guacamole-server-1.5.5# ./configure --with-init-dir=/etc/init.d`
Etape	Description
5	Lancer la compilation avec **make**

	Représentation
	`root@linuxserveur:~/guacamole-server-1.5.5# make`
Etape	Description
6	Installer *guacd* en tapant la commande **make install**
	Représentation
	`root@linuxserveur:~/guacamole-server-1.5.5# make install`
Etape	Description
7	Mettre à jour les liens et le cache des bibliothèques avec **ldconfig**
	Représentation
	`root@linuxserveur:~/guacamole-server-1.5.5# ldconfig`
Etape	Description
8	Crée manuellement le fichier de service *systemd* avec **nano /etc/systemd/system/guacd.service**
	Représentation
	`root@linuxserveur:~# nano /etc/systemd/system/guacd.service`

Etape	Description
9	Entrer les informations permettant à Linux de savoir comment lancer et gérer le programme *guacd* automatiquement comme un service système.
	Représentation
	GNU nano 5.4 /etc/systemd/system/guacd.service * [Unit] Description=Guacamole proxy daemon (guacd) After=network.target [Service] Type=simple ExecStart=/usr/local/sbin/guacd -f -b 0.0.0.0 Restart=on-abort [Install] WantedBy=multi-user.target
Etape	Description
10	Recharger la configuration des services systemd avec **systemctl daemon-reload**
	Représentation
	root@linuxserveur:~# systemctl daemon-reload
Etape	Description
11	Démarrer le service *guacd* avec la commande **systemctl start guacd**
	Représentation
	root@linuxserveur:~/guacamole-server-1.5.5# systemctl start guacd
Etape	Description
12	Activer le service *guacd* au démarrage avec la commande **systemctl enable guacd**
	Représentation
	root@linuxserveur:~# systemctl enable guacd

A présent, en tapant la commande **systemctl status guacd**, on observe que le service est bien actif:

```
● guacd.service - Guacamole proxy daemon (guacd)
     Loaded: loaded (/etc/systemd/system/guacd.service; enabled; ven>
     Active: active (running) since Wed 2025-10-29 14:54:20 CET; 10s>
   Main PID: 24887 (guacd)
      Tasks: 1 (limit: 1115)
     Memory: 11.5M
        CPU: 9ms
     CGroup: /system.slice/guacd.service
             └─24887 /usr/local/sbin/guacd -f
```

4.3.2.3. Installation de l'interface web Guacamole

Enfin, l'installation de l'interface web rend l'accès distant possible depuis un simple navigateur. Les étapes ci-dessous permettent de finaliser la mise en place de *Guacamole* et de créer les fichiers de configuration nécessaires:

Etape	Description
1	Télécharger le fichier WAR de Guacamole avec la commande **wget https://downloads.apache.org/guacamole/1.5.5/binary/guacamole-1.5.5.war**
	Représentation
	`root@linuxserveur:~# wget https://downloads.apache.org/guacamole/1.5.5/binary/guacamole-1.5.5.war`
Etape	Description
2	Copier le fichier dans le répertoire web de Tomcat avec **cp guacamole-1.5.5.war /var/lib/tomcat9/webapps/guacamole.war**
	Représentation
	`root@linuxserveur:~# cp guacamole-1.5.5.war /var/lib/tomcat9/webapps/guacamole.war`

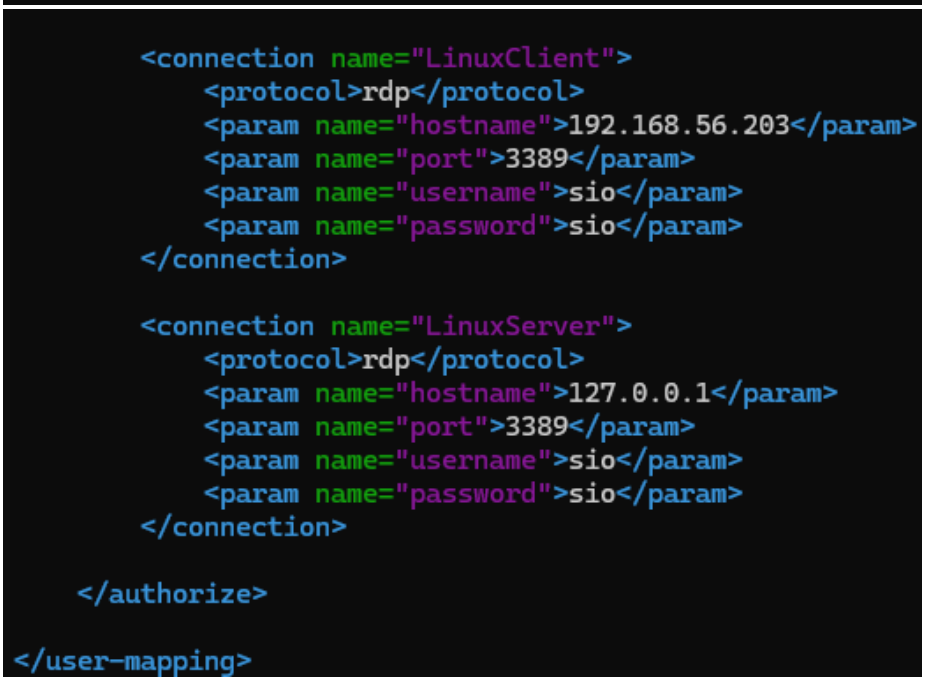
Etape	Description
3	Créer le répertoire de configuration avec la commande **mkdir /etc/guacamole**
	Représentation
	`root@linuxserveur:~# mkdir /etc/guacamole`
Etape	Description
4	Créer le fichier de configuration principal en tapant la commande **nano /etc/guacamole/guacamole.properties**
	Représentation
	`root@linuxserveur:~# nano /etc/guacamole/guacamole.properties`
Etape	Description

5	Ajouter les lignes suivantes au fichier:
	<pre>guacd-hostname: localhost
	guacd-port: 4822
	user-mapping: /etc/guacamole/user-mapping.xml</pre>

	Représentation

	`GNU nano 5.4 /etc/guacamole/guacamole.properties *`
	`<pre>guacd-hostname: localhost`
	`guacd-port: 4822`
	`user-mapping: /etc/guacamole/user-mapping.xml</pre>`

Etape	Description
6	Créer le fichier d'utilisateurs avec la commande **nano /etc/guacamole/user-mapping.xml**
	Représentation
	`root@linuxserveur:~# nano /etc/guacamole/user-mapping.xml`
Etape	Description
7	Ajouter les connexions vers chaque machine à laquelle on souhaite accéder à distance.

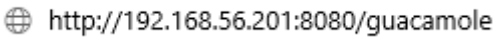
	Représentation
	
	
Etape	Description
8	Redémarrer *Tomcat* avec la commande **systemctl restart tomcat9**
	Représentation
	

A présent, en tapant la commande **systemctl status tomcat9**, on observe que le service est bien actif:

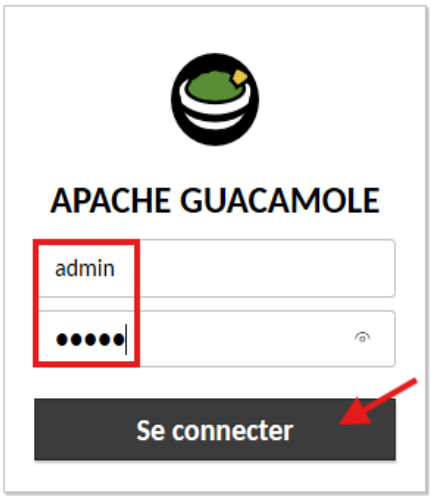
```
● tomcat9.service - Apache Tomcat 9 Web Application Server
     Loaded: loaded (/lib/systemd/system/tomcat9.service; enabled; v>
     Active: active (running) since Wed 2025-10-29 14:37:58 CET; 18m>
       Docs: https://tomcat.apache.org/tomcat-9.0-doc/index.html
   Main PID: 14356 (java)
      Tasks: 31 (limit: 1115)
     Memory: 140.0M
        CPU: 11.086s
     CGroup: /system.slice/tomcat9.service
             └─14356 /usr/lib/jvm/default-java/bin/java -Djava.util.>
```

4.3.3. Utilisation

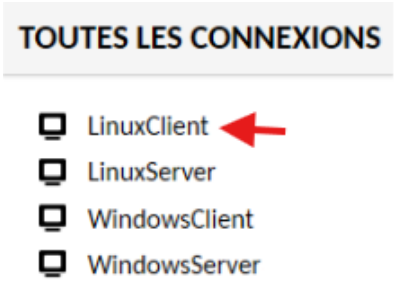
Afin de pouvoir se connecter à distance à une machine grâce à Apache Guacamole, il faut simplement entrer http://192.168.56.201:8080/guacamole (correspond à *LinuxServeur* sur lequel on a installé *Guacamole*) dans un navigateur:



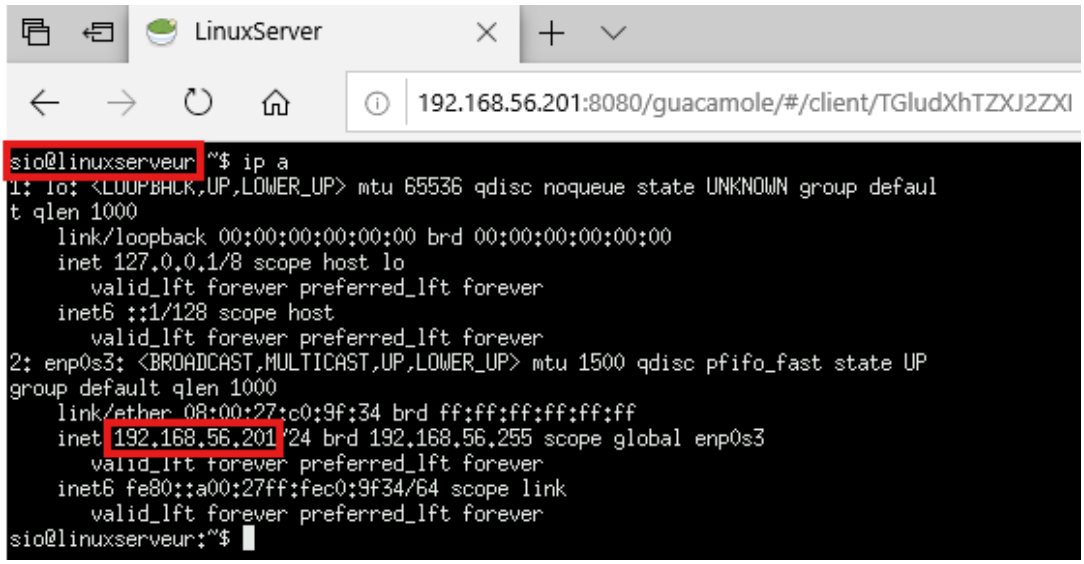
Dans la page qui s'ouvre alors, entrer les identifiants de connexion spécifiés dans le fichier **user-mapping.xml** dans la partie 4.3.2.3, dans notre cas, admin/admin:



Choisir ensuite la machine à laquelle se connecter, par exemple *LinuxClient*:



Nous pouvons observer que la connexion vers *LinuxClient* a bien fonctionné:

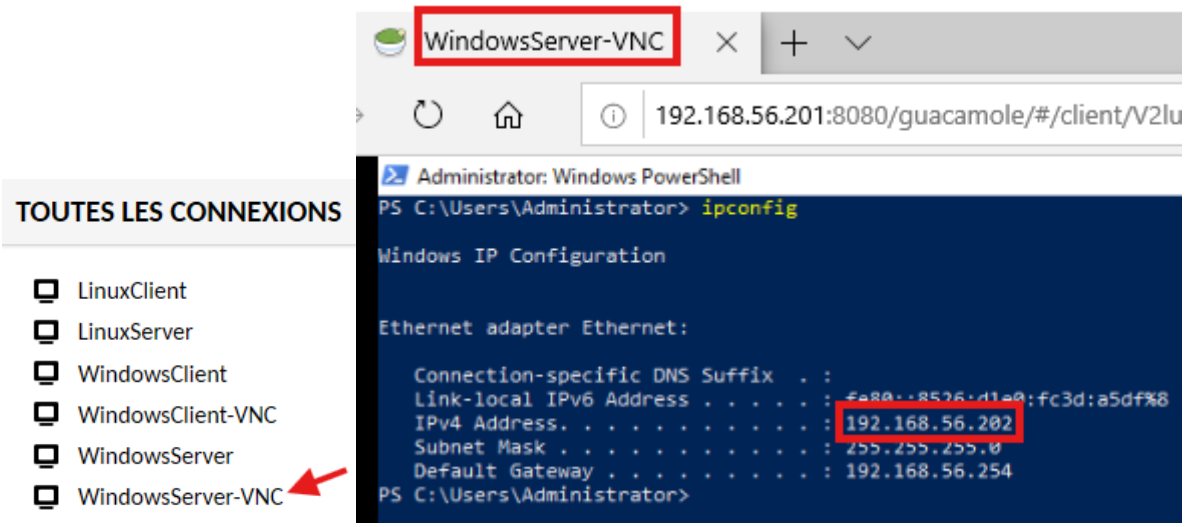


Il faut aussi noter qu'en ajoutant une connexion VNC vers *WindowsServer* (et *WindowsClient*) dans le fichier **user-mapping.xml**:

```
<connection name="WindowsServer-VNC">
    <protocol>vnc</protocol>
    <param name="hostname">192.168.56.202</param>
    <param name="port">5900</param>
    <param name="password">UltraVNC</param>
</connection>

<connection name="WindowsClient-VNC">
    <protocol>vnc</protocol>
    <param name="hostname">192.168.56.204</param>
    <param name="port">5900</param>
    <param name="password">UltraVNC</param>
</connection>
```

Il est possible de s'y connecter avec l'interface web d'*Apache Guacamole*:



5. Avantages et inconvénients

Les trois solutions testées (*Remmina - xrdp - rdp*, *UltraVNC - x11vnc* et *Apache Guacamole*) proposent des approches différentes pour établir des connexions à distance. Chacune présente des avantages et des limites selon le contexte d'utilisation, le système d'exploitation et le niveau de sécurité recherché.

5.1. Remmina - xrdp - Bureau à distance

Remmina, associé à *xrdp*, est une solution Linux permettant l'utilisation du protocole RDP. Elle se distingue par sa polyvalence, supportant RDP, VNC, SSH, SPICE et d'autres protocoles. Sur Debian ou Ubuntu, c'est un outil standard pour les connexions graphiques à distance, que ce soit vers des serveurs Linux ou Windows.

Avantages :

- Gratuit et open source
- Multi-protocole côté Linux
- Compatibilité bidirectionnelle entre Linux et Windows
- Performance correcte et qualité graphique fluide sur réseau local

Inconvénients :

- Nécessite une interface graphique pour se connecter depuis le serveur Linux
- Un PC sous Windows Home peut se connecter à distance à un autre ordinateur, mais ne peut pas être contrôlé à distance via RDP

5.2. UltraVNC - x11vnc

UltraVNC est une solution conçue à l'origine pour Windows, utilisant le protocole VNC. Elle permet d'afficher et de contrôler un bureau à distance, avec une installation et une configuration simples : le serveur s'installe sur la machine à contrôler et le client sur la machine distante.

Pour Linux, *x11vnc* est l'équivalent serveur VNC : il permet d'accéder à l'interface graphique d'une machine Linux. Cependant, si le serveur Linux n'a pas d'environnement graphique installé, il est impossible d'utiliser VNC pour se connecter, car le protocole nécessite un bureau graphique pour fonctionner.

Avantages :

- Gratuit
- Installation rapide et configuration intuitive
- Compatible avec Linux via *x11vnc*
- Simple et efficace pour un usage bureautique ou support technique

Inconvénients:

- Optimisé pour Windows, moins fluide sous Linux
- Sécurité limitée : le protocole VNC n'est pas chiffré par défaut
- Performances graphiques inférieures à RDP
- Impossible d'accéder à un serveur Linux dépourvu d'interface graphique

5.3. Apache Guacamole

Apache Guacamole adopte une approche moderne et centralisée. Cette solution “clientless” ne nécessite aucune installation sur le poste client : un navigateur suffit. L’accès se fait via un portail web hébergé par le serveur Guacamole, qui permet des connexions RDP, VNC ou SSH.

Avantages :

- Gratuit et open source
- Accessibilité universelle depuis n’importe quel appareil
- Sécurité renforcée grâce à un serveur central et possibilité d’authentification centralisée
- Aucun logiciel à installer côté client

Inconvénients:

- Installation serveur plus complexe (*Tomcat*, *guacd*, base de données possible)
- Performances légèrement inférieures à une connexion RDP directe

6. Comparaison des solutions

Pour mieux visualiser les différences entre les solutions choisies et d’autres non-explorées, mais présentes sur le marché, un tableau comparatif a été réalisé. Les informations concernant *TeamViewer* et *AnyDesk* proviennent de la documentation officielle et de comparatifs en ligne:

	Remmina - xrdp - Bureau à distance	UltraVNC - x11vnc	Apache Guacamole	TeamViewer	AnyDesk
Licence	Gratuit et open source	Gratuit et open source	Gratuit et open source	Gratuit pour usage personnel / payant pro	Gratuit pour usage personnel / payant pro
Plateformes supportées	Linux, Windows	Principalement Windows (clients multi-OS)	Multi-plateforme via navigateur web	Windows, macOS, Linux, Android, iOS	Windows, macOS, Linux, Android, iOS
Installation côté client	Oui (Remmina sur Linux)	Oui (client VNC)	Non (navigateur web)	Oui	Oui
Installation côté serveur	Oui (xrdp sur Linux)	Oui (serveur VNC)	Oui (Tomcat + guacd)	Oui	Oui

Protocole principal	RDP / VNC / SSH	VNC	RDP / VNC / SSH	Protocole propriétaire	Protocole propriétaire
Chiffrement intégré	Oui (TLS pour RDP)	Non par défaut	Oui (HTTPS, SSH, RDP chiffré)	Oui (AES 256 bits)	Oui (TLS 1.2 + RSA 2048)
Niveau de sécurité	Bon (renforcé avec VPN)	Moyen sans tunnel SSH	Très bon (HTTPS + authentification centralisée)	Très élevé	Très élevé
Performances graphiques	Très bonnes en local	Moyennes	Bonnes (selon le navigateur)	Excellentes	Excellentes
Accès web sans client	Non	Non	Oui	Oui (version web)	Oui (version web)
Compatibilité Linux/Windows croisée	Partielle (pour utiliser *Remmina*, une interface graphique est nécessaire)	Partielle (pour utiliser *x11vnc*, une interface graphique est nécessaire)	Oui	Oui	Oui
Prise en main / simplicité	Moyenne (configuration technique)	Facile	Moyenne à complexe	Très simple	Très simple
Utilisation en entreprise	Oui (infrastructures mixtes)	Moyenne	Oui (administration centralisée)	Oui	Oui
Usage personnel / support rapide	Moyen	Bon	Moyen	Excellent	Excellent

7. Difficultés rencontrées

Au cours de la mise en œuvre des différentes solutions de connexion à distance, quelques difficultés ont été rencontrées.

La première concernait la configuration réseau des machines virtuelles sous Linux. Lors des premiers tests, certaines machines ne parvenaient pas à communiquer entre elles. Après analyse, le problème provenait de la présence de plusieurs adresses IP assignées automatiquement par VirtualBox sur une même interface réseau. Ce conflit d'adressage empêchait la communication entre les machines. Pour corriger ce dysfonctionnement, j'ai

supprimé l'ensemble des adresses IP à l'aide de la commande **ip addr flush dev enp0s3**, puis attribué manuellement une adresse fixe avec **ip addr add [ip_machine/masque] dev enp0s3**. Ensuite, j'ai ajouté une route par défaut avec **ip route add default via [ip_passerelle]**. Cette manipulation a permis de rétablir une connectivité stable et fiable entre toutes les Machines Virtuelles.

Un autre aspect qui a compliqué la mise en œuvre des solutions de connexion à distance a été la nécessité de comprendre en détail le fonctionnement de chaque outil utilisé. Il ne suffisait pas d'installer un logiciel et d'essayer de s'y connecter : il a fallu analyser le fonctionnement des communications propres à chaque solution et les paramètres de configuration spécifiques.

En particulier, la configuration d'*Apache Guacamole* a demandé un travail approfondi. J'ai dû me renseigner sur la structure de l'application et comprendre comment *Guacamole* gère les utilisateurs et les connexions. Une partie importante était le fichier **user-mapping.xml**, qui définit les comptes, les mots de passe et les accès aux différentes machines distantes. Au cours des tests, certaines connexions échouaient, ce qui m'a obligé à revenir sur ce fichier pour y ajouter ou corriger les lignes nécessaires, en respectant la syntaxe XML et les identifiants des machines. Cette étape a été essentielle pour assurer que chaque utilisateur pouvait se connecter correctement à ses machines cibles via *Guacamole*.

Enfin, trouver certaines machines virtuelles adaptées a constitué une difficulté supplémentaire, notamment pour *WindowsServeur*. Les images officielles n'étant pas toujours disponibles, j'ai dû rechercher et télécharger une VM compatible via des archives fiables de machines préconfigurées. Ce processus a été nécessaire pour disposer d'un environnement de test stable et pouvoir effectuer les connexions à distance sans erreur.

Ces difficultés ont demandé du temps, de la méthode et de la persévérance, mais elles ont surtout amené à renforcer mes compétences techniques et ma capacité d'adaptation.

8. Introspection

Ce projet m'a permis de gagner en autonomie et de développer une vraie rigueur dans la gestion des problèmes techniques. J'ai appris à mettre en place une démarche méthodique: observer le problème, formuler des hypothèses, vérifier les journaux système (**journalctl**, **systemctl status**) et valider chaque étape avant de passer à la suivante. Cela m'a aidé à mieux comprendre le fonctionnement des services réseau et à identifier rapidement la cause d'un échec de connexion.

Pour *Apache Guacamole*, j'ai utilisé le fichier **user-mapping.xml**, qui permet de définir tous les utilisateurs et leurs connexions dans un simple fichier XML. Cela présente l'avantage d'être très simple à comprendre et à mettre en place, et ne nécessite aucune configuration supplémentaire. Cette solution est idéale pour des tests ou des démonstrations car elle permet de voir clairement la structure des connexions.

Cependant, elle présente des inconvénients majeurs, notamment le fait que les mots de passe sont stockés en clair dans le fichier, ce qui représente un problème de sécurité.

De plus, il faut redémarrer le serveur *Tomcat* à chaque fois que l'on modifie un utilisateur ou une connexion. Il n'existe pas d'interface graphique pour gérer les utilisateurs,

tout se fait manuellement dans le fichier, et il est impossible de voir qui s'est connecté et quand, car il n'y a pas d'historique.

Enfin, la gestion des permissions reste très limitée.

Une autre méthode pour gérer les utilisateurs, que je n'ai pas utilisée dans ce projet, repose sur une base de données MariaDB. Elle consiste à stocker toutes les informations dans une base de données relationnelle. Au lieu de modifier un fichier XML, on passe par une interface web intégrée à *Guacamole* pour gérer les utilisateurs et les connexions. Cette méthode présente de nombreux avantages en termes de sécurité et de fonctionnalités. Tout d'abord, les mots de passe sont chiffrés dans la base de données grâce à des algorithmes de hachage, ce qui les rend illisibles même si quelqu'un accède à la base.

Ensuite, l'interface web intuitive permet de créer, modifier ou supprimer des utilisateurs sans avoir à toucher aux fichiers de configuration, ce qui facilite grandement l'administration. L'un des atouts majeurs de cette méthode est l'historique complet des connexions: on peut consulter qui s'est connecté, à quelle machine, à quel moment et pendant combien de temps, ce qui est essentiel pour l'audit et la traçabilité. Les modifications sont également appliquées en temps réel sans nécessiter de redémarrage des services, ce qui évite les interruptions de service.

La gestion des permissions est beaucoup plus avancée, ce qui permet de créer des groupes d'utilisateurs et de limiter l'accès à certaines machines selon les profils. Il est même possible d'ajouter l'authentification à deux facteurs pour renforcer encore la sécurité.

Enfin, plusieurs administrateurs peuvent gérer le système simultanément sans risque de conflit. En revanche, la configuration initiale est beaucoup plus complexe car elle nécessite de créer la base de données, d'importer un schéma SQL, d'installer des extensions Java et de maîtriser le fonctionnement des bases de données relationnelles. Cette mise en place demande donc plus de temps et de connaissances techniques.

Sur le plan personnel, cette expérience m’a appris la patience, la persévérance et la logique. En effet, j’ai cherché à comprendre le fonctionnement de chaque solution avant de les installer, puis j’ai pris le temps de comprendre les erreurs qui empêchaient les connexions avant d’agir, tester et de documenter chaque étape pour garder une trace claire du travail réalisé.

Enfin, j’ai globalement réfléchi à l'utilisation de la connexion à distance en entreprise et sélectionné 3 solutions gratuites et open source, mais d’autres solutions peut-être plus utilisées telles que AnyDesk, ou TeamViewer existent. Il serait intéressant de les mettre en place pour pouvoir les comprendre et réellement observer les différences notées dans la partie 6.

En résumé, ces difficultés m’ont appris à travailler de manière structurée, à mieux comprendre le fonctionnement des réseaux et des systèmes, et à améliorer ma capacité à résoudre des situations techniques complexes.

9. Conclusion

Les trois solutions testées, *Remmina - xrdp - Bureau à distance*, *UltraVNC - x11vnc* et *Apache Guacamole*, proposent chacune une manière différente de se connecter à distance, avec leurs points forts et leurs limites.

Remmina - xrdp - Bureau à distance est pratique pour les environnements mixtes Linux/Windows. C'est une solution gratuite et open source, avec de bonnes performances graphiques. Elle est assez sûre si l'on utilise un VPN ou un tunnel sécurisé. Son principal point faible est qu'il faut que le serveur Linux ait une interface graphique, et Windows Home ne peut pas être utilisé comme serveur RDP.

UltraVNC - x11vnc est facile et rapide à installer sur Windows. C'est une solution idéale pour du support ou du travail à distance sur un bureau Windows. Cependant, elle est moins sécurisée par défaut et ses performances graphiques sont un peu moins bonnes que RDP. Sur Linux, *UltraVNC* n'existe pas, mais il est possible d'utiliser *x11vnc* pour accéder à une machine Linux équipée d'une interface graphique. Si le serveur Linux n'a pas d'interface graphique installée, la connexion VNC est impossible.

Apache Guacamole est moderne et fonctionne directement depuis un navigateur, sans rien installer côté client. Il est très flexible et sécurisé, et permet de gérer plusieurs utilisateurs facilement. Dans ce projet, la gestion des utilisateurs a été réalisée via un fichier XML simple (**user-mapping.xml**), mais Guacamole peut également utiliser une base de données MariaDB pour une gestion plus avancée : chiffrement des mots de passe, historique des connexions, gestion des permissions et administration via interface web. Globalement, l'inconvénient d'*Apache Guacamole* est que l'installation du serveur est plus complexe et les performances sont un peu moins rapides qu'une connexion RDP directe.

Les solutions commerciales telles que *TeamViewer* et *AnyDesk* sont très simples à utiliser, sont rapides et sécurisées. Elles sont adaptées pour du support ou un accès rapide depuis n'importe quel appareil, mais leurs versions gratuites sont limitées et elles dépendent d'un service externe plutôt que d'être auto-hébergées.

En résumé, le choix de la solution dépendra principalement, du type d'environnement (Linux, Windows ou mixte), du niveau de sécurité requis, de la volonté ou non de gérer l'infrastructure soi-même et du besoin d'accessibilité depuis n'importe quel appareil ou navigateur.

Pour un environnement Linux/Windows auto-hébergé avec contrôle des données, *Remmina - xrdp - Bureau à distance* et *Apache Guacamole* sont recommandés.
Pour une utilisation rapide, universelle et sans installation serveur complexe, *TeamViewer* et *AnyDesk* restent les plus pratiques.

Enfin, en dehors de l'aspect technique, ce projet m'a permis d'acquérir une meilleure compréhension des services réseau et des outils de connexion à distance. Les difficultés rencontrées m'ont fait adopter une méthode structurée : observer, tester, documenter, et comprendre avant d'agir. Il serait intéressant de tester d'autres outils afin de comparer plus précisément leurs performances, leurs limites et leurs modes de configuration.