REPUBLIQUE DU SENEGAL

Un peuple-un but-une foi

Ministère de l'Enseignement Supérieur, de la Recherche et de l'Innovation
Direction de l'Enseignement Supérieur Privé

Institut Supérieur d'Informatique

ISI

Rapport pour l'examen d'administration système Linux semestre 5

Configuration d'un serveur web sécurisé Apache2 et Nginx en interopérabilité sur Ubuntu.

Année Académique : 2024-2025

Présenté et soutenu par :

M. NGOUWA Bimbounza
Emmanuel Sevy

Sous la direction de :

Professeur MASSAMBA LO

TABLE DE MATIÈRES

Chapitre 1 : Introduction

1.1. Contexte

Ce rapport a pour objectif de présenter l'implémentation d'un serveur Linux dans le cadre de la modernisation de l'infrastructure informatique d'entreprises de taille intermédiaire, telles que les PME et PMI ou de taille robuste tel que les data centers. Le serveur vise principalement à centraliser les services de stockage de données et à renforcer la sécurité des accès réseau eux différents services.

1.2. Objectif

Les objectifs de ce projet incluent :

- *La configuration du service de nom de domaine DNS*
- *La configuration d'un serveur web sous apache2*
- *La configuration d'un serveur web sous Nginx*
- *La création d'un mini site web*
- *Configuration de Https pour apache et Nginx*

1.3. Choix du Système d'exploitation

Le choix de Linux comme système d'exploitation repose sur ses qualités indéniables de stabilité, de sécurité et de flexibilité, avec une préférence marquée pour la distribution Ubuntu Server en raison de sa compatibilité avec les outils open source
et de sa vaste communauté de soutien.

1.4. Environnement

Nous allons déployer notre serveur Ubuntu sur VirtualBox, un logiciel de virtualisation open source créé par Oracle. Ce dernier permet de créer et de gérer des machines virtuelles (VM) sur un ordinateur principal. VirtualBox s'avère particulièrement utile pour les étudiants, les administrateurs système et les développeurs souhaitant acquérir des compétences en installation et en configuration de serveurs Linux, tout en préservant l'intégrité de leur système hôte.

Ainsi, cette solution offre la possibilité de simuler un environnement de production, permettant de tester des configurations de serveurs Linux avant de les mettre en œuvre sur des machines physiques.

1.5. Public cible

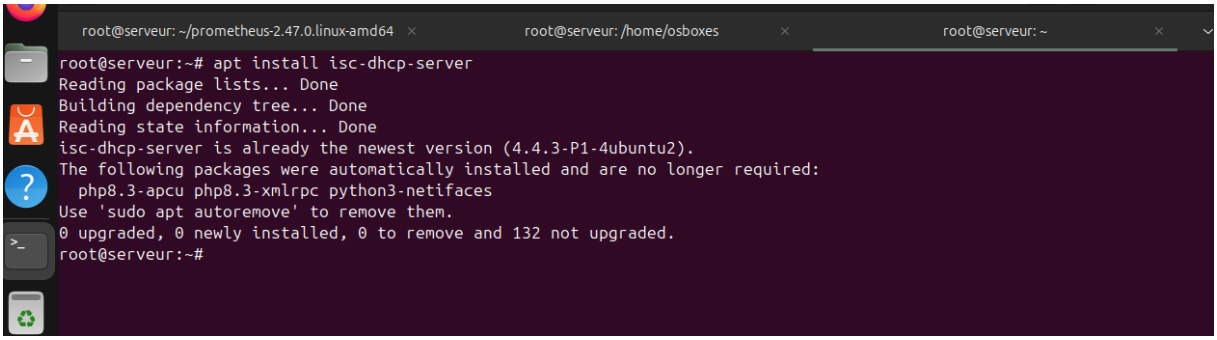
Ce document s'adresse à l'équipe technique chargée de la gestion des systèmes, ainsi qu'aux ingénieurs réseaux, afin de leur offrir une vision d'ensemble du projet. Il est organisé en cinq sections : planification, installation, configuration, tests et conclusion.

1.6. Structure et méthodologie

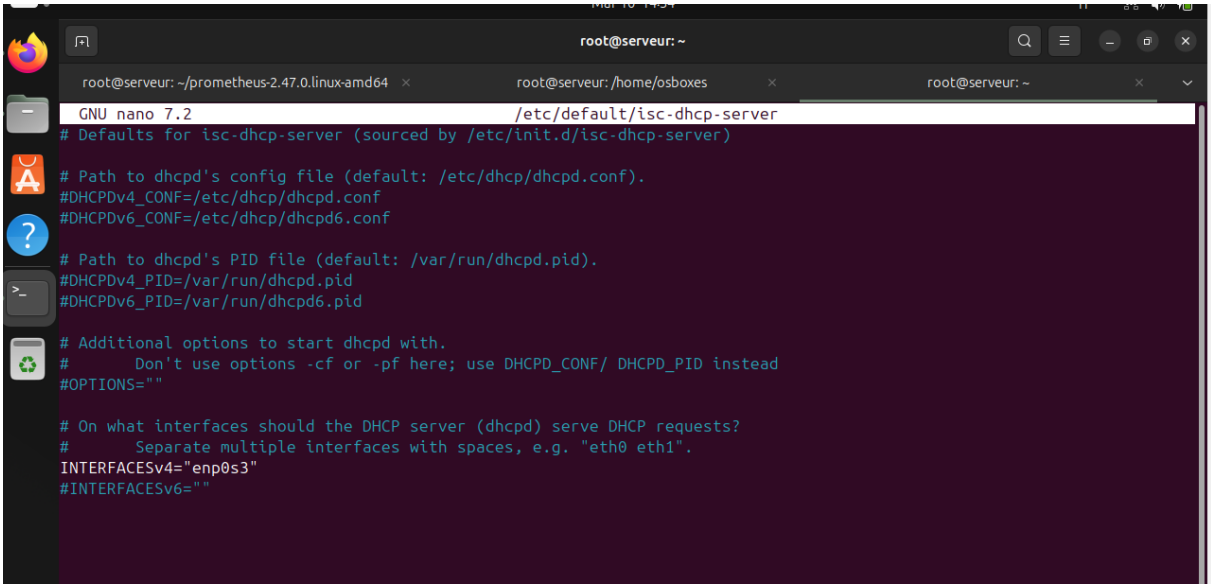
Le rapport présente une approche méthodique étape par étape, incluant l'utilisation de scripts Bash pour automatiser certaines tâches et la ligne de commande pour réaliser les diverses configurations.

DHCP

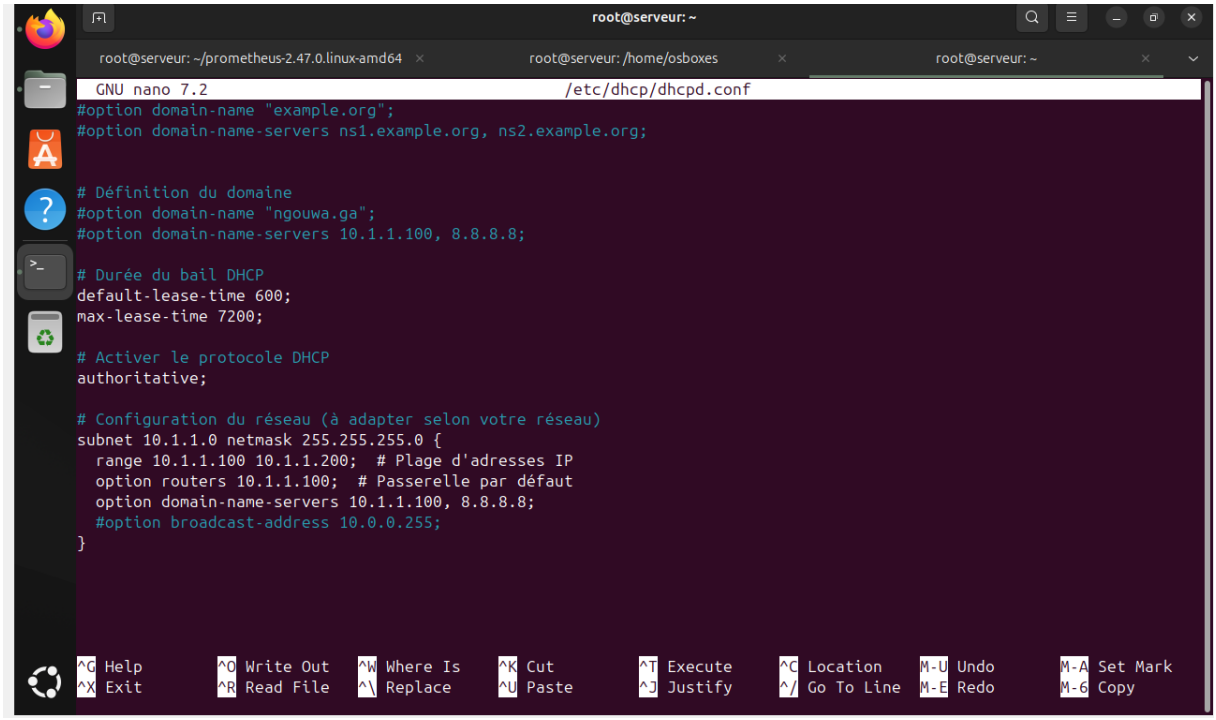
Téléchargement du paquet isc-dhcp-server



Assignations de l'interface pour le serveur dhcp



Configuration du réseaux dhcp



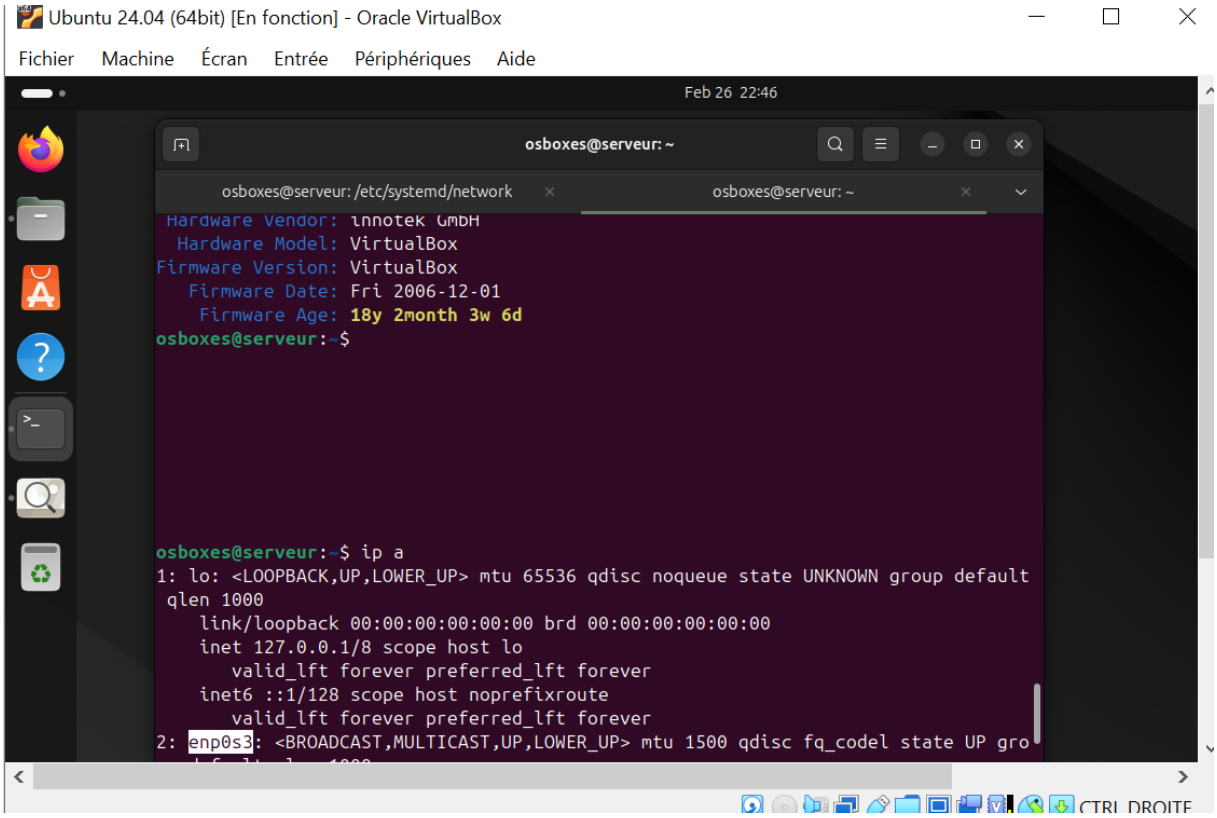
Chapitre 2 : Configuration DNS

2.1. Identification de l'interface réseaux

Une interface réseau sous Linux constitue un point de communication essentiel entre le système et le réseau. De ce fait, l'identification de son nom est une étape basique mais importante pour la suite de notre travail sachant bien qu'une machine peut avoir plusieurs interface réseaux.

Pour ce faire nous tapons la ligne de commande suivante :

- ➔ ip a



Constat : Nous relevons l'interface nommée enp0s3 sur laquelle nous ferons des configurations dans la séquence suivante.

2.2. Configuration du fichier d'interface enp0s3

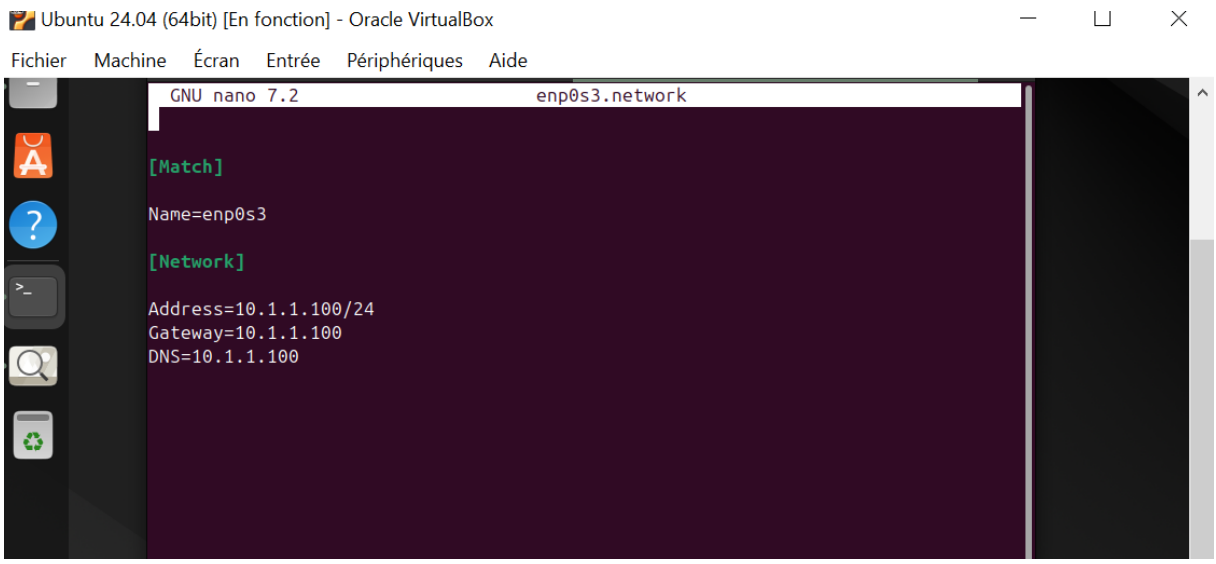
Elle permet la transmission de données grâce à des adresses IP et à des protocoles spécifiques. La configuration adéquate de cette interface est fondamentale pour le bon fonctionnement d'un serveur web, car elle garantit que celui-ci est accessible depuis d'autres machines, que ce soit sur le réseau local ou sur Internet. De plus, une

configuration précise contribue à la sécurité, à la performance et à la stabilité du serveur, des éléments indispensables pour l'hébergement de services en ligne.

Dans cette séquence nous choisissons une add **IPV4** tel que **10.1.1.100** avec pour masque **255.255.255.0** Cette adresse sera statique sur l'interface du serveur.

Pour ce faire tapons la ligne de commande suivante et modifions **le fichier enp0s3.network :**

➔ nano /etc/systemd/network/enp0s3.network



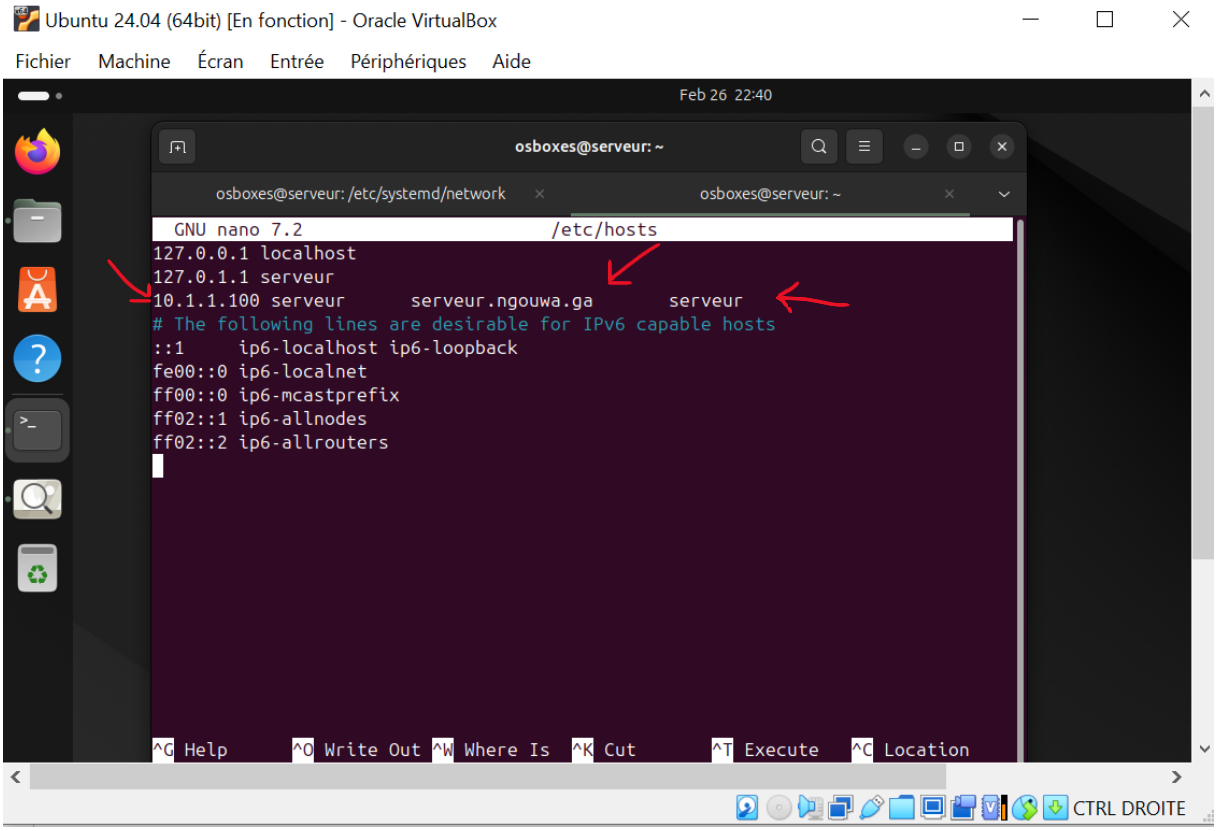
Constat : Pour configurer un serveur maître et pour favoriser la haute disponibilité, nous avons choisis de fixer une même add IP pour le serveur, la passerelle et le DNS. Par ailleurs la plage d'adresse du DHCP seront pris en fonction de cette même add 10.1.1.100.

2.3. Définir un nom de serveur personnalisé (hostname)

Dans notre cas nous choisissons le nom « *serveur* » que nous allons définir dans le fichier « hosts »

Ligne de commande :

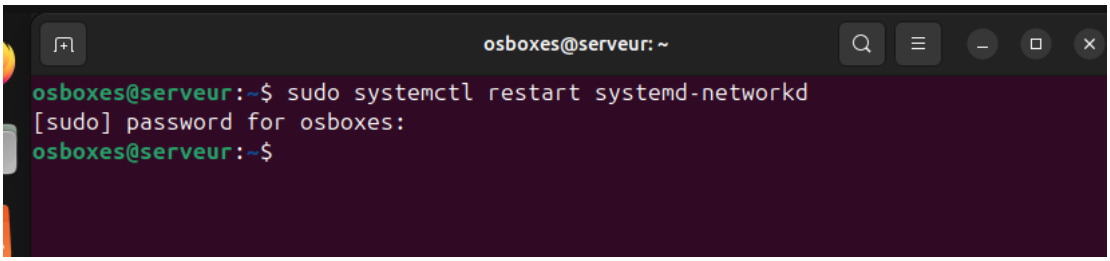
➔ nano /etc/hosts



Constat : Vous pouvez remarquer que nous le nom du serveur est définis conjointement avec l'adresse ip du serveur et également le futur nom de domaine que nous allons configurer dans les étapes suivantes, « ngouwa.ga » .

2.4. Vérification des informations du serveur

Après avoir fait nos configurations de base, nous allons redémarrer les services réseaux grâce à la commande « systemctl restart systemd-networkd » ,

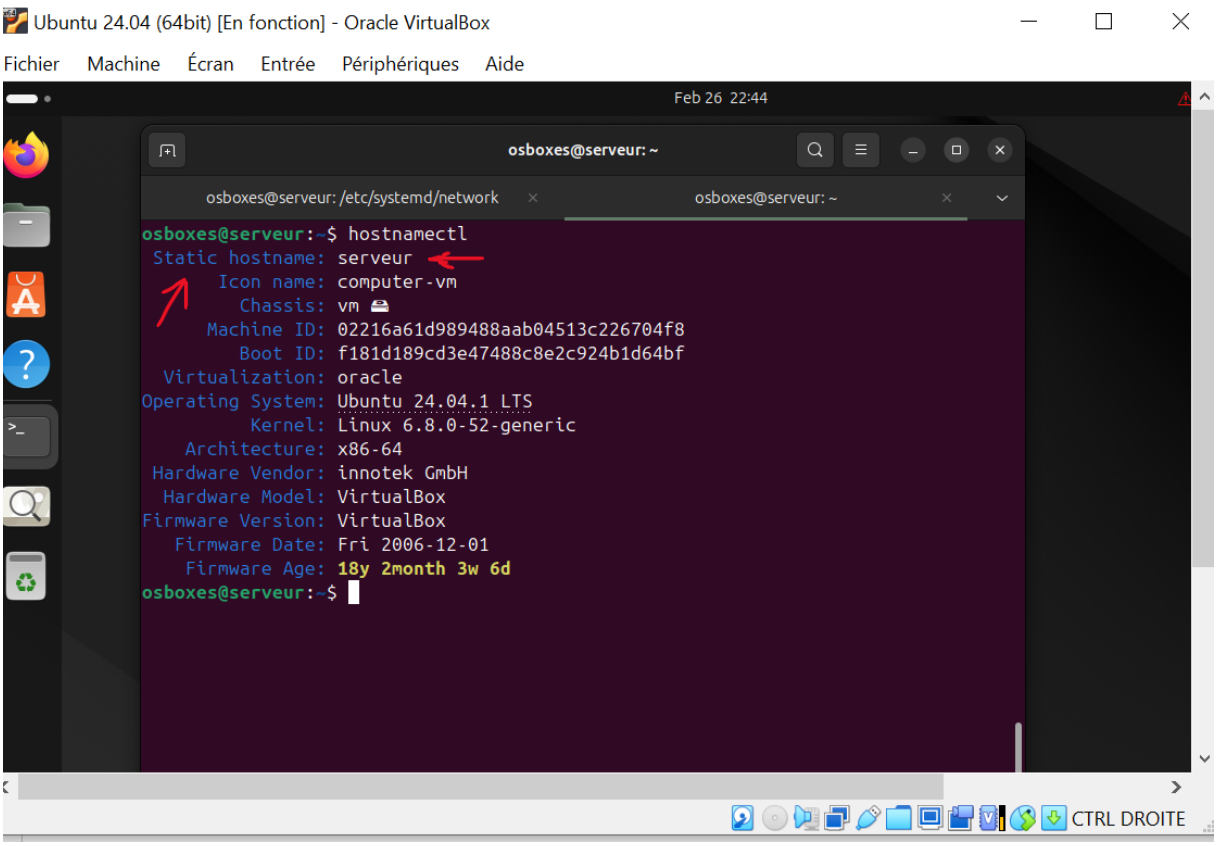


Constat : contrairement aux anciens systèmes linux qui ont utilisé le gestionnaire « networking », les dernières distributions tel que Ubuntu 24.04, prennent en charge un nouveaux gestionnaire appelé « systemd-networkd ».

Nous pouvons enfin vérifier que notre serveur porte bien le nom que nous avons donné et que celui-ci est statique.

Ligne de commande :

➔ hostnamectl



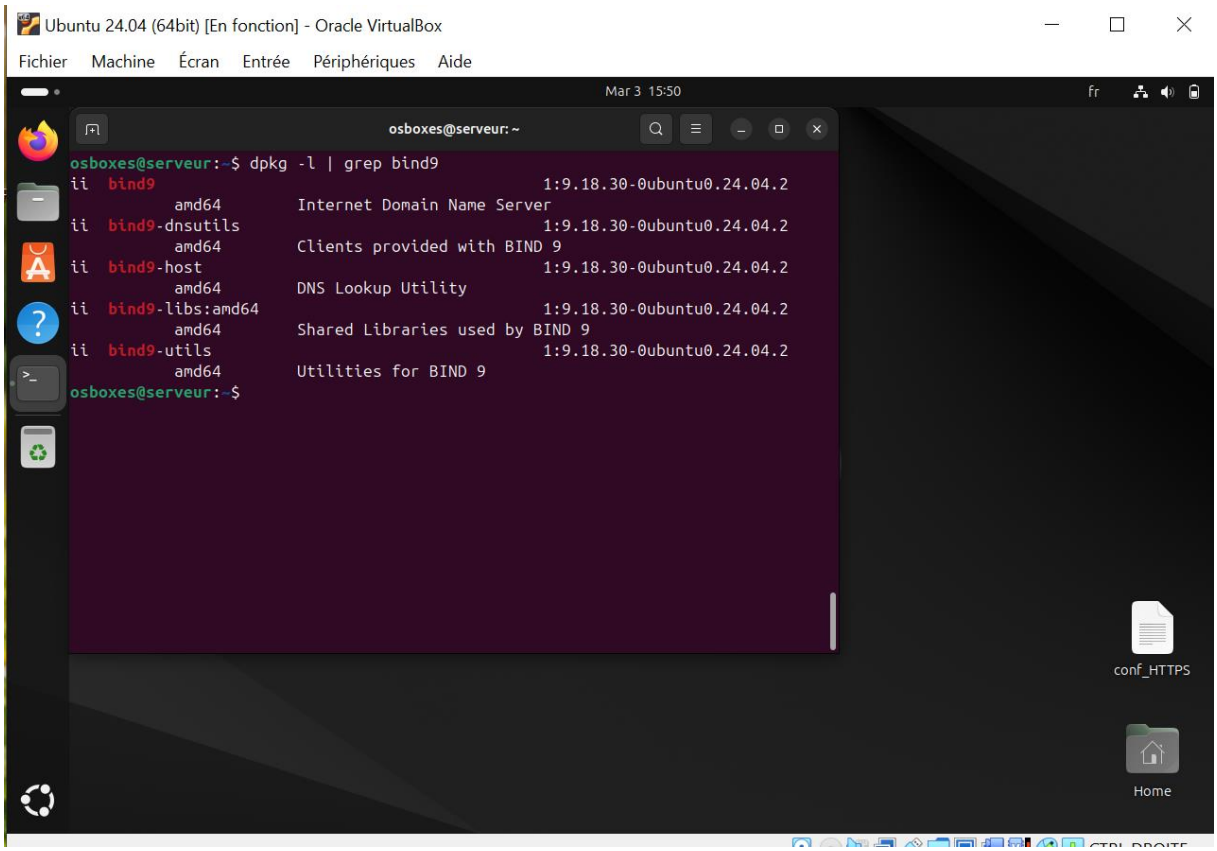
2.5. Téléchargement et installation du paquet Bind

Dans cette étape nous parlons du paquet Bind9, qui joue un rôle important dans la configuration du DNS et son fonctionnement.

Par contre, parce que nous avons déjà eu à le télécharger et l'installer, nous allons juste vérifier son existence sur le serveur .

Ligne de commande :

➔ dpkg -l | grep Bind9



2.6. Configuration propre au service du nom de domaine .

Dans le cadre de la configuration du DNS sur Debian, les zones et les fichiers de zone occupent une place centrale. Une zone DNS représente une portion de l'espace des noms de domaine pour laquelle un serveur DNS est chargé de la gestion. Chaque zone est définie par un fichier de zone, qui est généralement situé dans

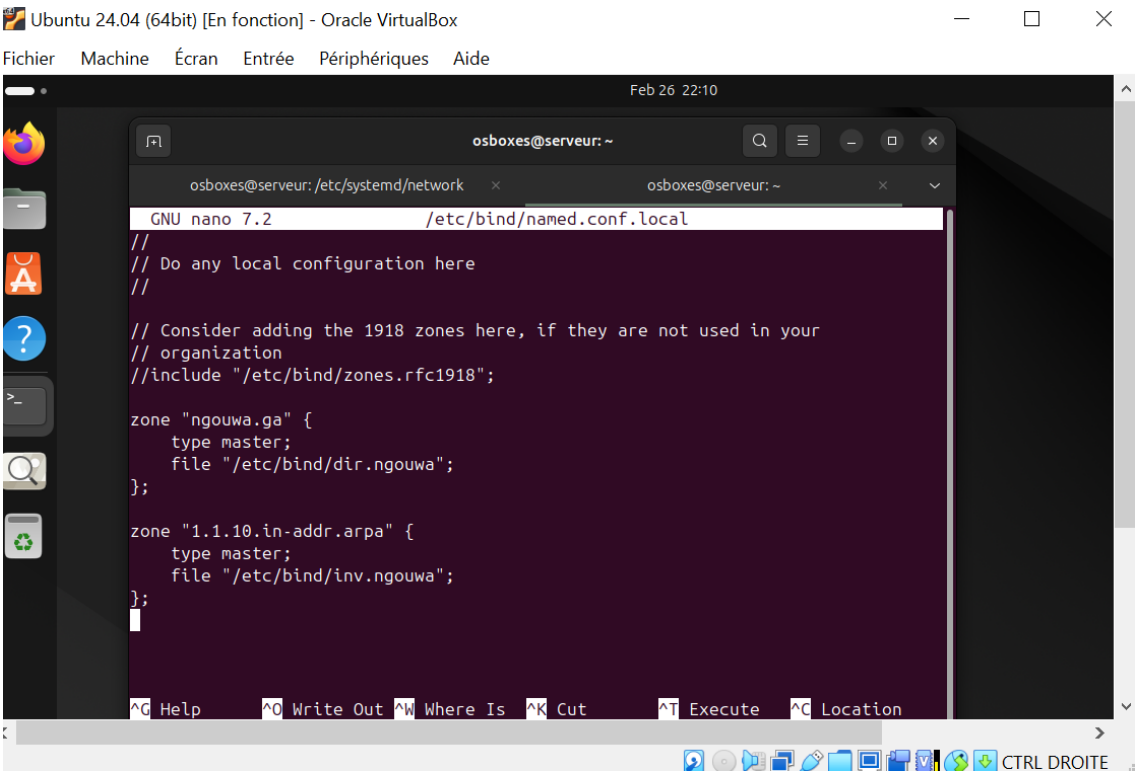
le répertoire /etc/bind/ ou dans un sous-répertoire spécifique. Ce fichier contient divers enregistrements DNS, tels que les enregistrements A, MX, CNAME, NS, etc. , permettant de lier les noms de domaine à des adresses IP ou à d'autres types d'informations. Par exemple, un fichier de zone pour le domaine example. com pourrait inclure des entrées pour www. example. com ou mail. example. com.

Dans BIND, le serveur DNS le plus couramment utilisé sur Debian, les zones sont déclarées dans le fichier de configuration principal (named. conf ou named. conf. local). C'est ici que l'on définit le chemin vers le fichier de zone et le type de zone (maître, esclave ou forward). Une gestion adéquate des fichiers de zone est essentielle pour garantir une résolution efficace des noms de domaine et assurer la disponibilité des services réseau.

2.7. Déclaration des Zones (fichier named.conf)

Dans cette étape de notre rapport, nous allons déclarer les zones en spécifiant le chemin vers lequel les fichiers de zone seront créer et sauvegardés. Pour cela modifions le fichier « named.conf » situé dans le répertoire bind.

➔ nano /etc/bind/named.conf.local



2.8. Configuration du fichier de zone directe

Qu'est-ce qu'une zone directe en matière de DNS ?

Une zone directe, également connue sous le nom de zone de recherche directe, est une base de données au sein du système DNS (Domain Name System) qui établit les correspondances entre les noms de domaine, tels que site. com, et leurs adresses IP respectives, par exemple 192. 168. 1. 1 .

Comment ça fonctionne ?

Lorsque vous entrez un nom de domaine dans votre navigateur, la zone directe est consultée afin de traduire ce nom en une adresse IP, ce qui permet d'établir une connexion avec le serveur. Cette zone contient différents types d'enregistrements DNS, tels que :

- A : Associe un nom de domaine à une adresse IPv4.

- AAAA : Associe un nom de domaine à une adresse IPv6.

- CNAME : Crée un alias ou redirige un domaine vers un autre.

- MX : Spécifie le serveur de messagerie pour le domaine.

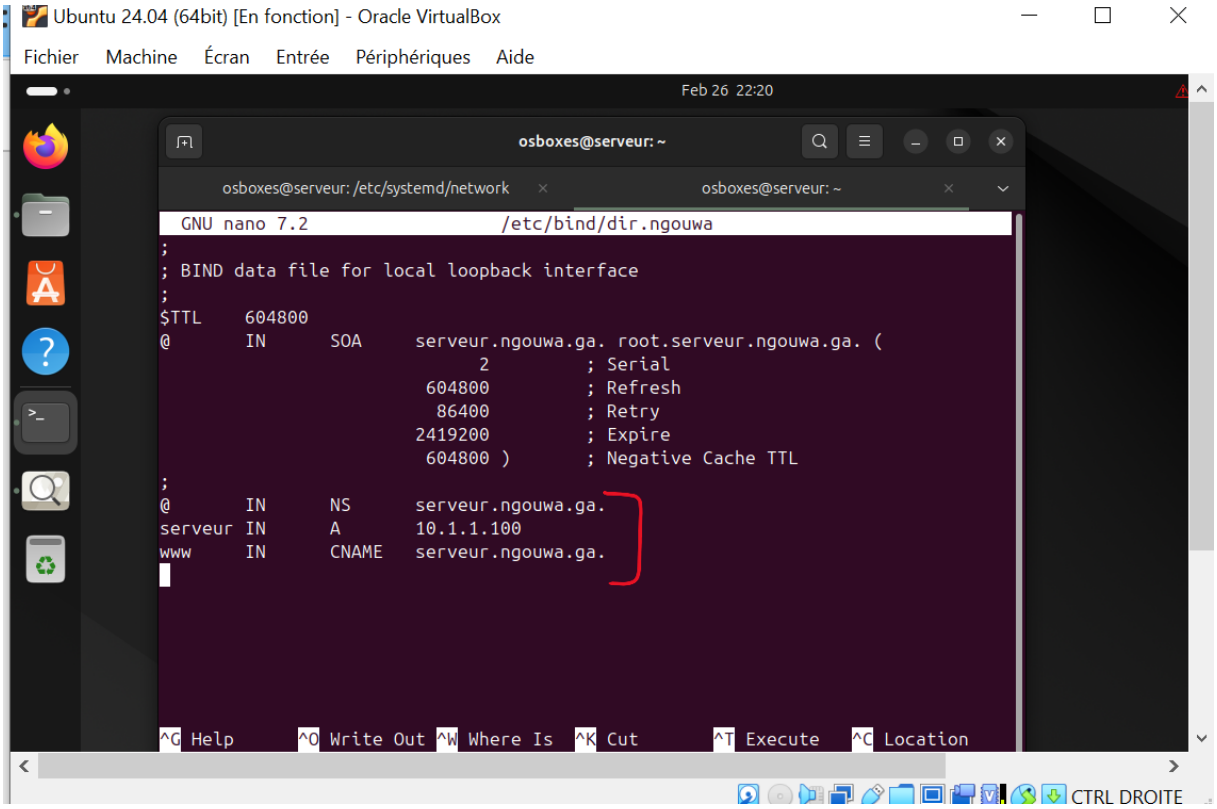
Exemple concret :

Supposons que vous interrogez www. ngouwa.ga ; dans ce cas, la zone directe vous fournira l'adresse IP correspondante, comme 10.1.1.100 , permettant ainsi à votre navigateur de se connecter au site.

En résumé, la zone directe joue un rôle essentiel en reliant les noms de domaine, compréhensibles par l'être humain, aux adresses IP nécessaires pour que les machines puissent communiquer entre elles.

Pour appliquer ces paramètres, nous allons créer un fichier nommé « dir.ngouwa » eg ajouter des lignes de configuration portant sur le nom de domaine et l'add IP choisie (serveur.ngouwa.ga et 10.1.1.100)

➔ nano /etc/named/dir.ngouwa



2.9. Configuration du fichier de zone indirecte

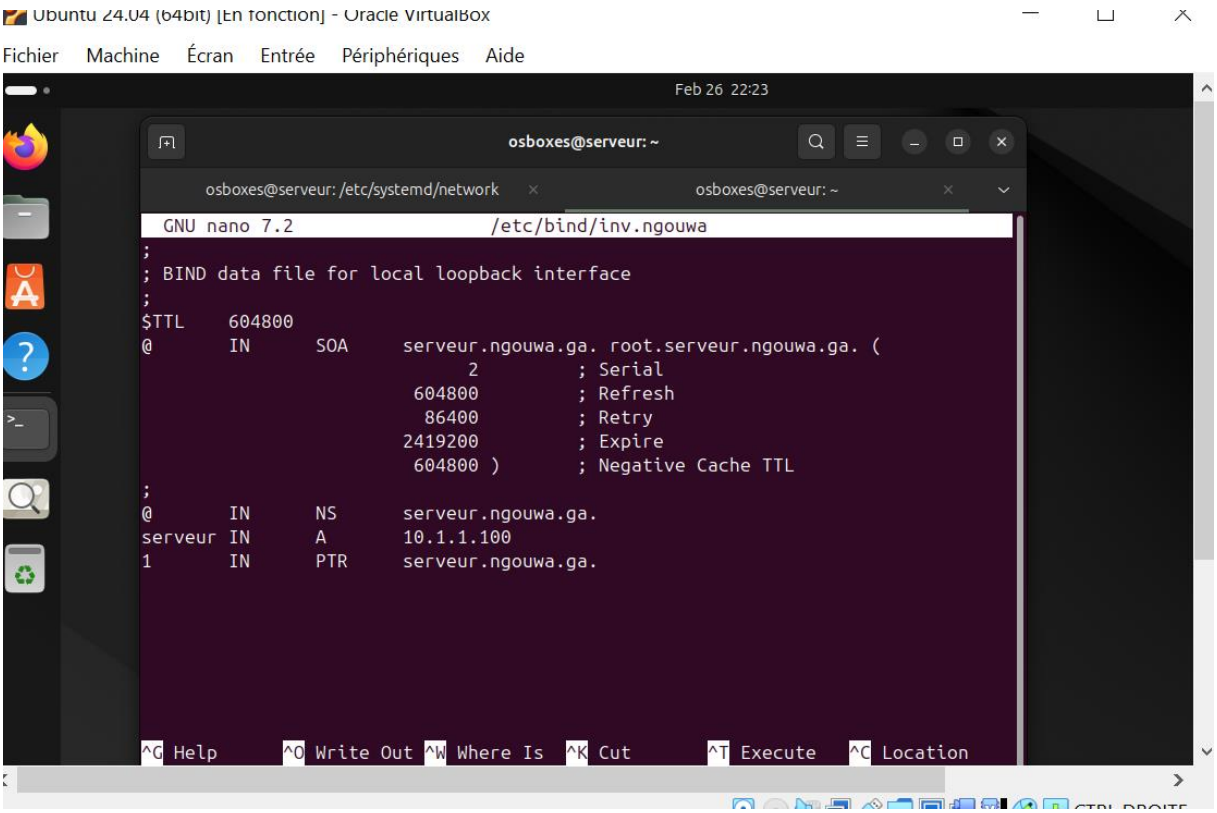
Dans le système DNS (Domain Name System), une zone indirecte est communément désignée sous le terme de zone de recherche inverse. Elle a pour fonction de traduire une adresse IP en un nom de domaine, à l'opposé de la zone directe qui effectue l'inverse, c'est-à-dire associe un nom de domaine à une adresse IP.

La zone de recherche inverse revêt une importance cruciale dans la conversion des adresses IP en noms de domaine, facilitant ainsi le diagnostic réseau et renforçant la sécurité.

Dans notre cas nous créons le fichier inv.ngouwa et nous écrivons toutes les configurations nécessaires dans ce fichier.

Ligne de commande :

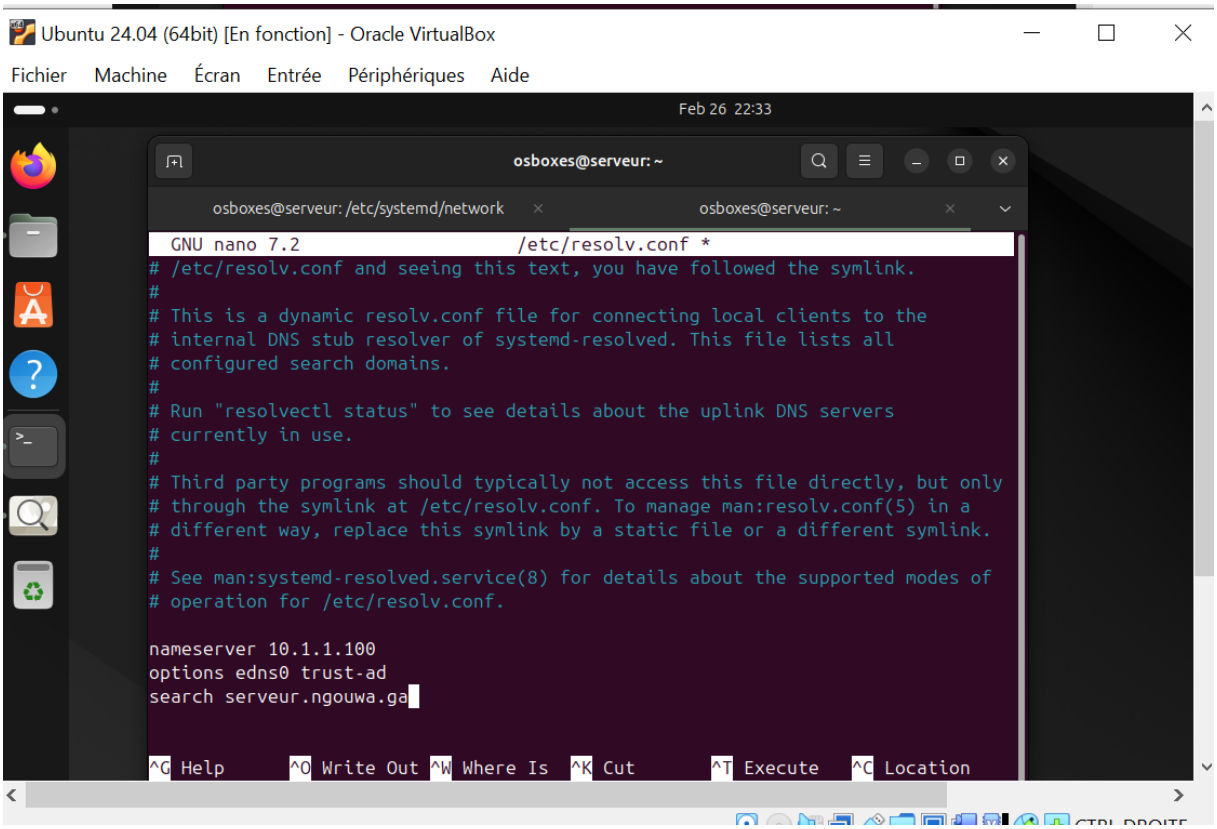
➔ nano /etc/named/inv.ngouwa



2.10. Configuration du fichier de résolution de nom de domaine

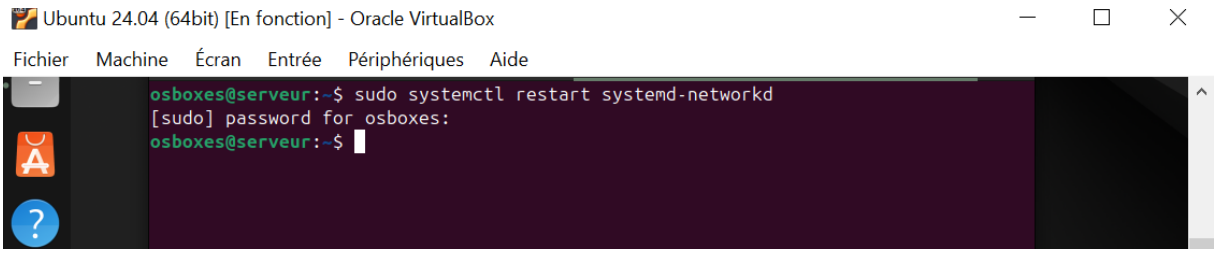
Ligne de commande

➔ nano /etc/resolv.conf



2.11. Redémarrage des services réseaux

Après chaque configuration de ce type nous devons toujours redémarrer les services réseaux afin que le serveur puisse prendre en compte toute mes modifications apportées sur les fichiers ainsi que les interfaces.

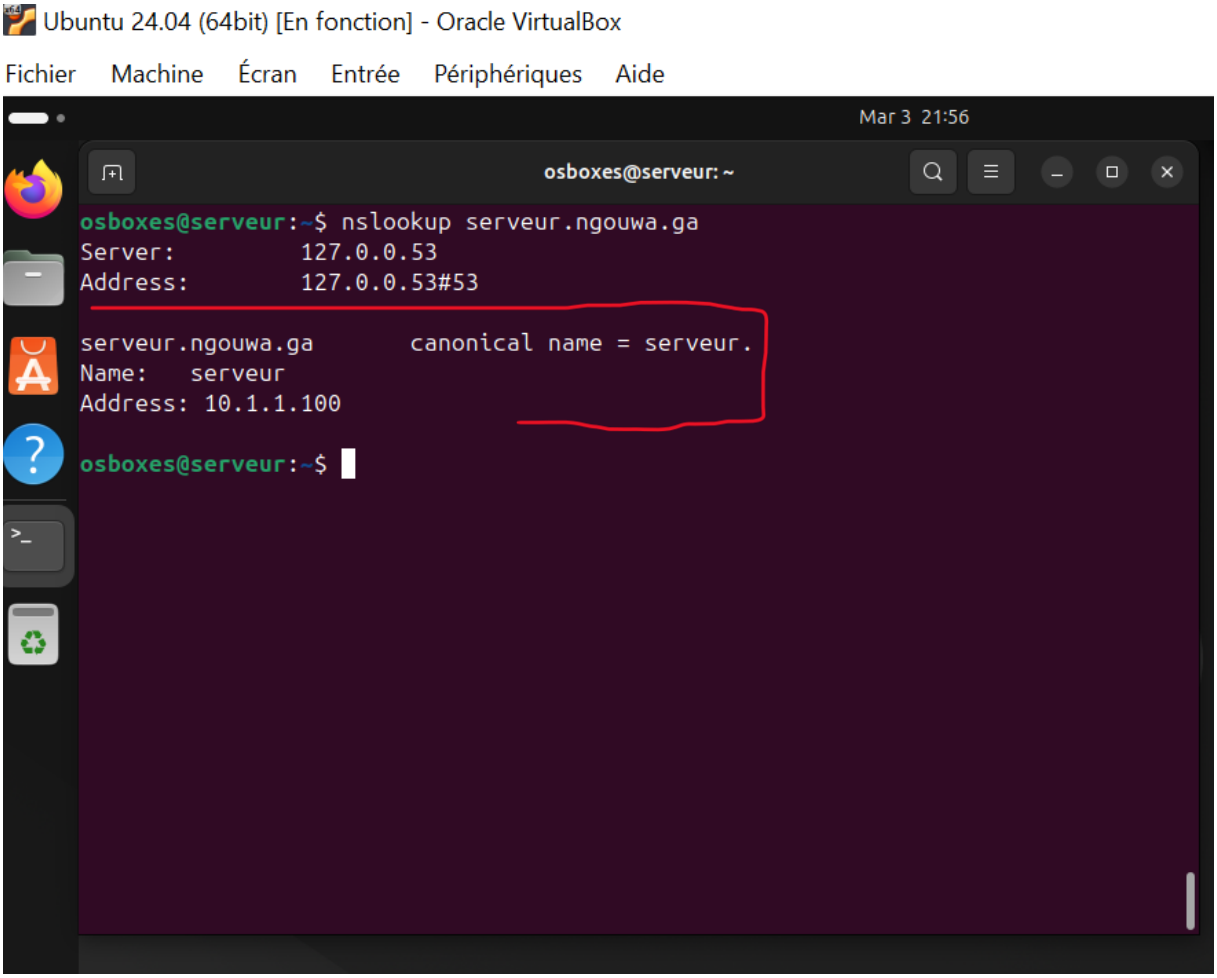


2.12. Vérification de l'existence et du bon fonctionnement du nom de domaine précédemment crée (nslookup).

La commande nslookup est un outil précieux pour interroger les serveurs DNS et obtenir des informations sur la résolution des noms de domaine, en établissant la correspondance entre un nom et son adresse IP. Accessible sur la plupart des systèmes d'exploitation, tels que Linux, macOS et Windows, elle est particulièrement utile pour diagnostiquer et résoudre divers problèmes liés au DNS.

Pour se faire nous allons taper la commande nslookup suivie de notre nom de domaine.

➔ nslookup serveur.ngouwa .com



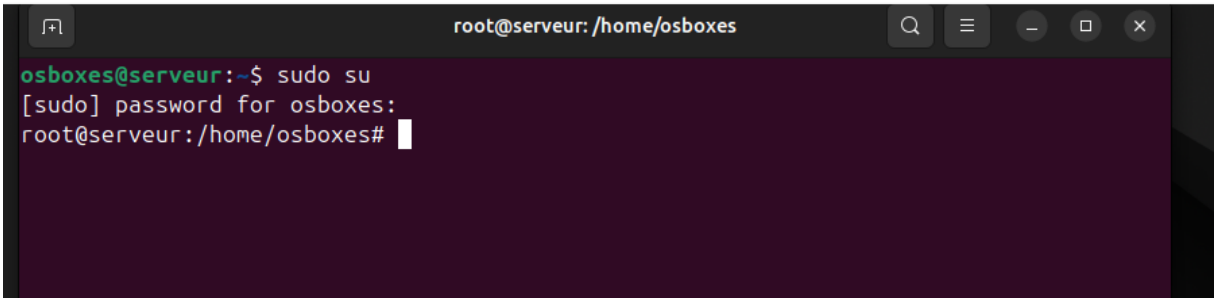
Constat : Notre domaine est belle et bien existant sur notre serveur avec la haute disponibilité à l'adresse IP static 10.1.1.100 .

Chapitre 3 : Installation et configuration Apache2

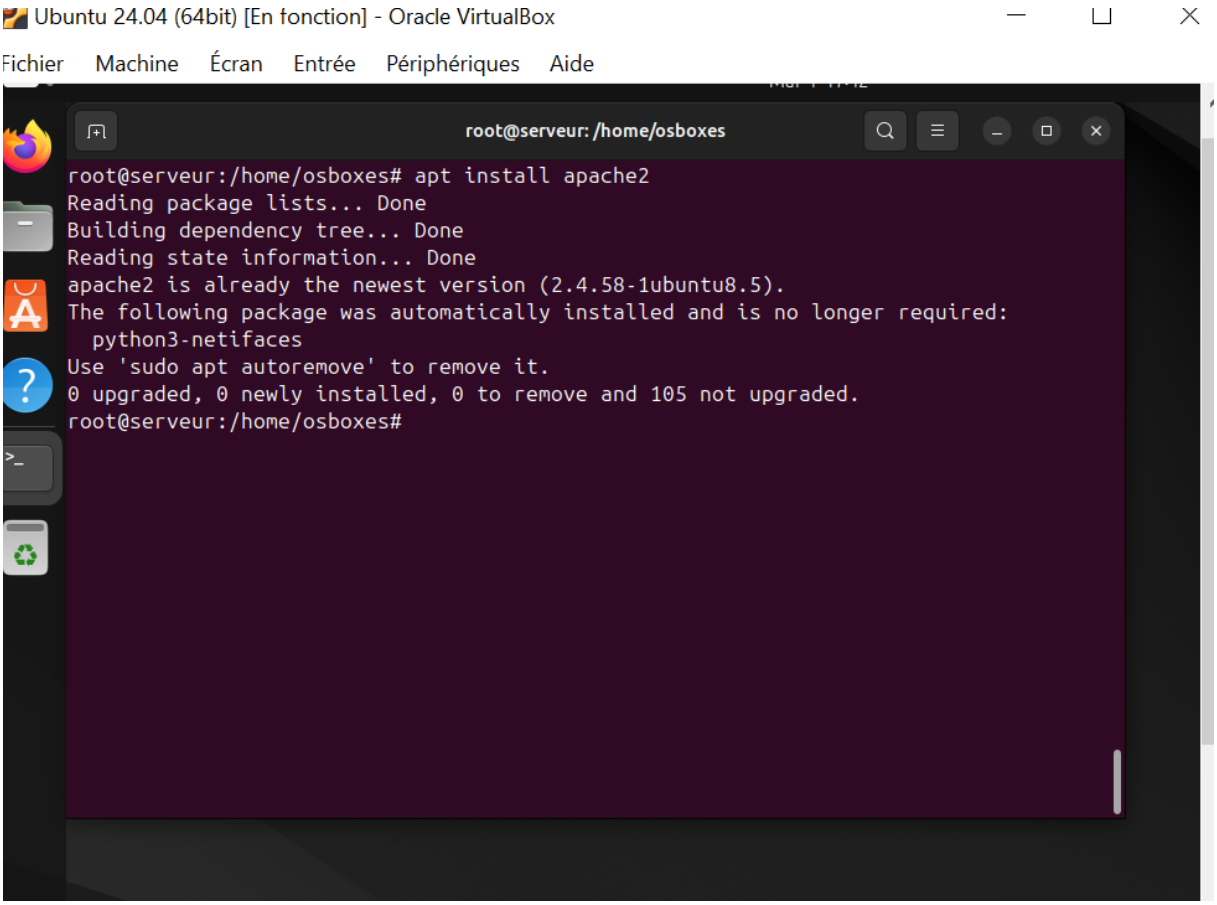
3.1. Elévation des privilèges sur le système

Tout d'abord passons en mode SuperUtilisateur pour la suite de nos travaux.

➔ sudo su



3.2. Téléchargement et installation du paquet apche2

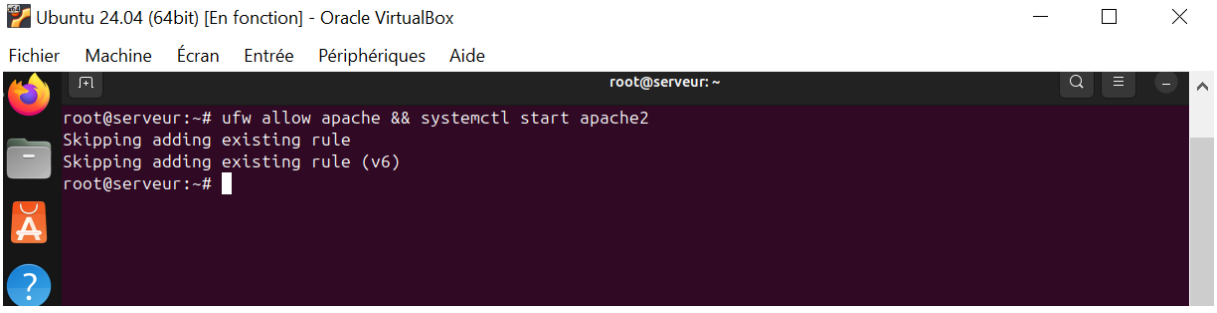


Constat : Nous remarquons qu'il existe déjà la dernière version d'apache2 sur notre serveur. Effectivement nous avions déjà téléchargé le paquet.

Du coup, dans la prochaine étape nous allons autoriser apache2 (port 80 et port 443) via le pare-feu et sur la même ligne de commande nous démarrons les services d'apache2.

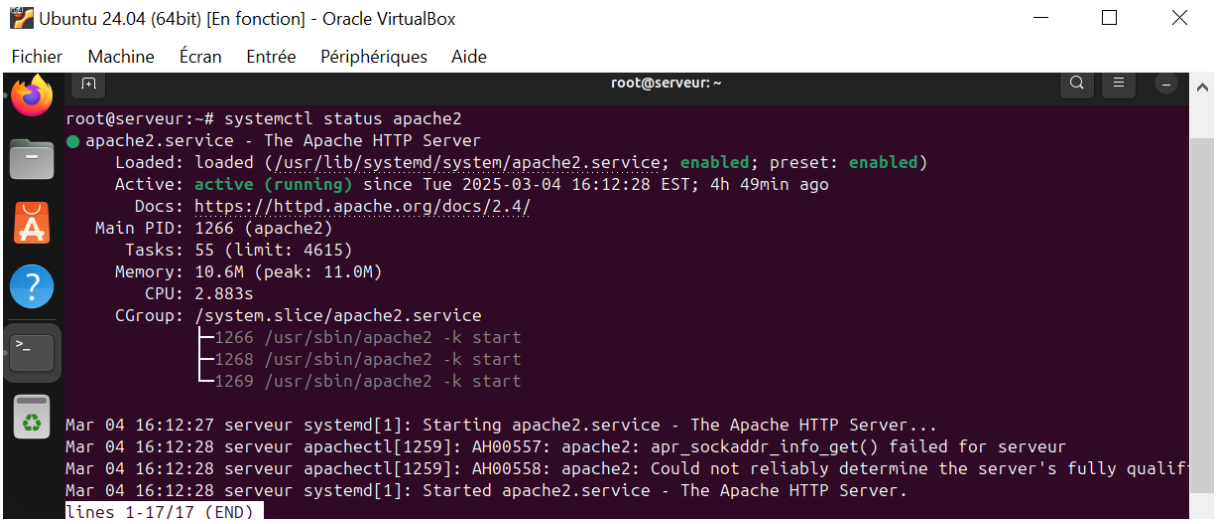
3.4. Accès pare-feu et lancement des services d'apache2

➔ ufw allow apache && systemctl restart apache2



3.5. Vérifions le fonctionnement du paquet apache2

➔ systemctl status apache2



Constat : apache2 est activé et en cours d'exécution sur notre serveur.

Maintenant que Apache2 est installé et actif, en principe la prochaine étape doit consister à créer un nouvelle hôte virtuel (un site web personnalisé) mais cela n'est pas vraiment nécessaire dans la mesure ou le nom de domaine précédemment créer et sa résolution à l'adresse 10.1.1.100 est automatique le substitut du localhost (127.0.0.1) adresse sur laquelle pointe le serveur web apache dans le navigateur.

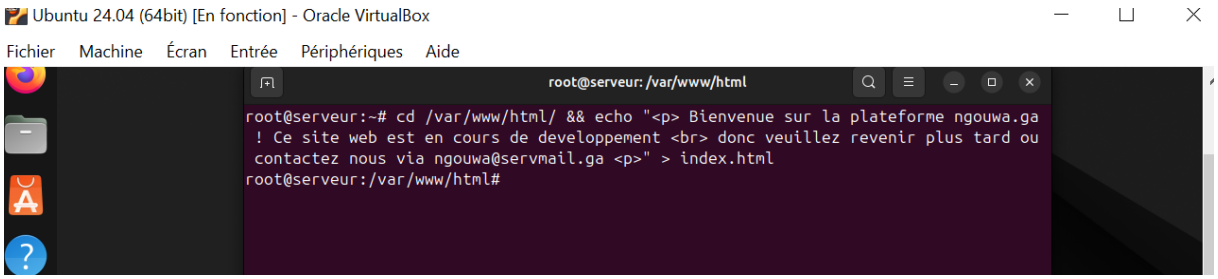
Par conséquent nous allons juste devoir soit remplacer ou sinon modifier et personnaliser la page index d'apache (index.html) situé dans le répertoire suivant le chemin /var/www/html/.

3.6. Création d'une page web sur le serveur

Pour cette étape nous allons créer un page web simple contenant le message suivant : « Bienvenue sur la plateforme ngouwa.ga ! Site en cours de développement donc veuillez revenir plus tard ou contactez nous sur ngouwa@mail.ga » Dans un bout de code HTML.

Ligne de commande :

➔ cd /var/www/html/ && echo ‘’<p> Bienvenue sur la plateforme ngouwa.ga ! Ce site web est en cours de développement
 donc veuillez revenir plus tard ou contactez nous via ngouwa@mail.ga <p>‘’ > index.html



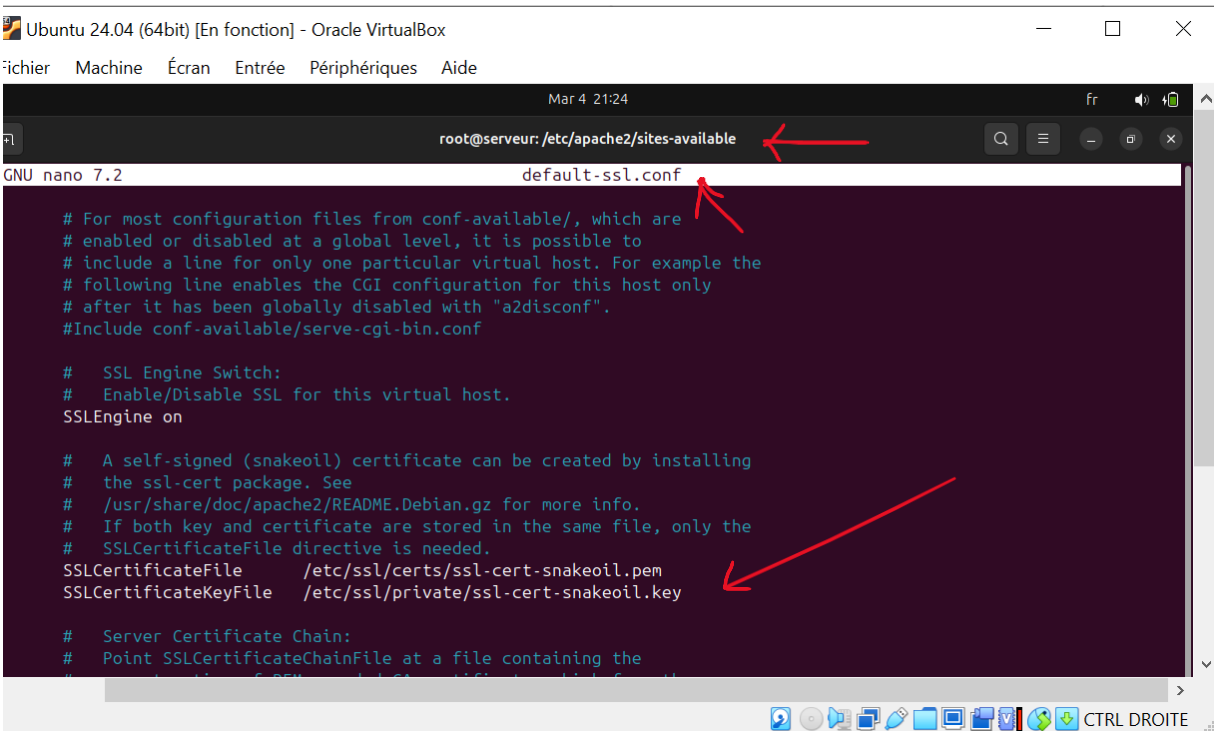
Constat : Aucun message d’erreur donc notre ligne de commandes a bien fonctionné parcontre avant d’afficher notre page web dans un navigateur, nous allons d’abord sécuriser la connexion à notre serveur web grâce au protocole https via le port 443.

3.7. Configuration de HTTPS

Cette nouvelle étape de notre travail consiste à chiffrer les communications entre tout clients qui contacte notre serveur web .

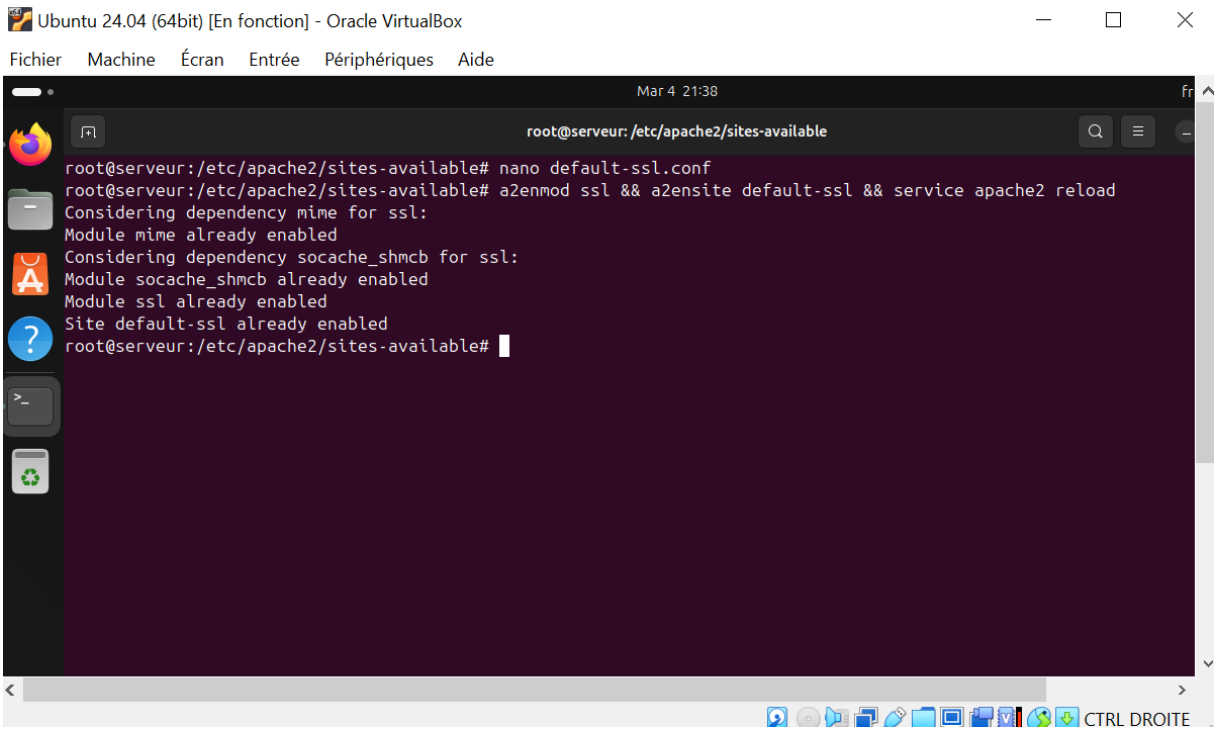
Il est important de souligner que la dernière version d’apache intègre de base un certificat auto signé contenue dans le fichier default-ssl.conf.

➔ nano /etc/apache2/sites-available/default-ssl.conf



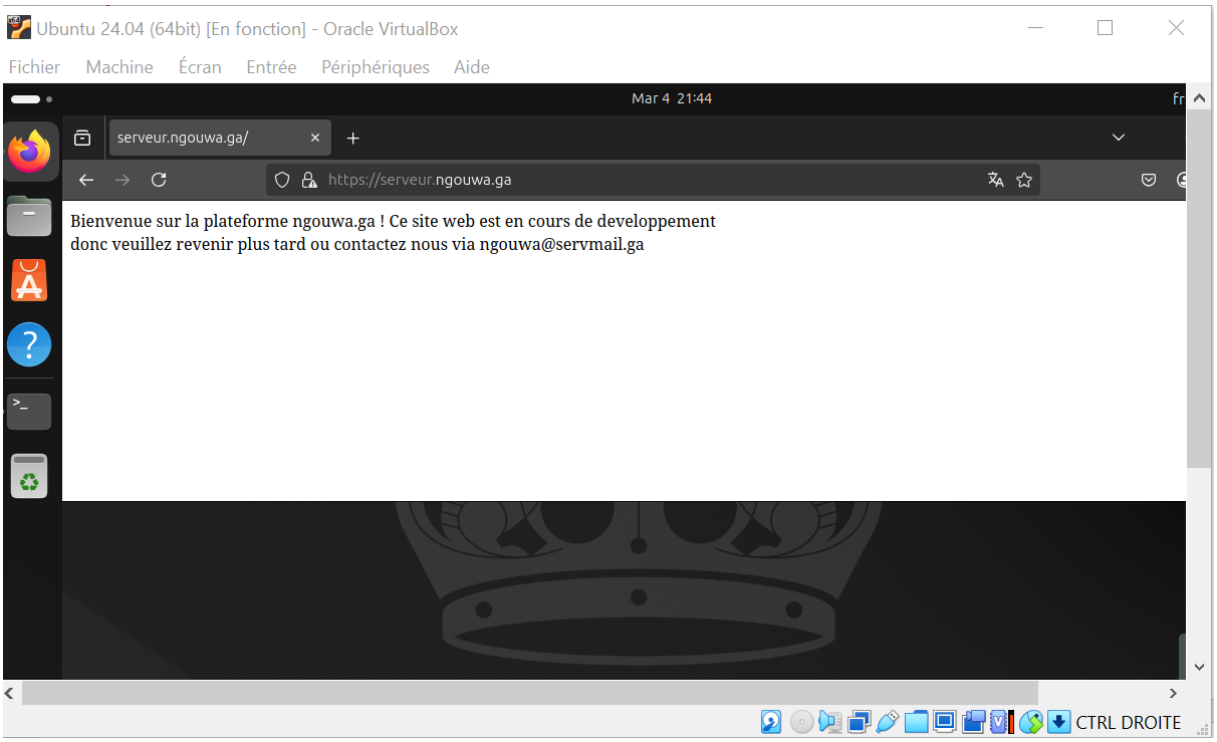
Constat : L'existence d'un certificat et de la clé de certificat est vérifié. De ce fait, nous refermons l'éditeur de texte nano sans modifier le fichier.

La prochaine étape va consister à juste activer les modules SSL afin que le certificat soit intégré à notre plateforme web.



Constat : Contrairement à l'utilisation de Openssl pour la création manuelle de certificat , la méthode que nous venons de voir est la plus rapide et la plus pratique pour un serveur web de test en local.

Ainsi, dans la prochaine étape nous allons dans un navigateur , accéder à notre site web en mode https.

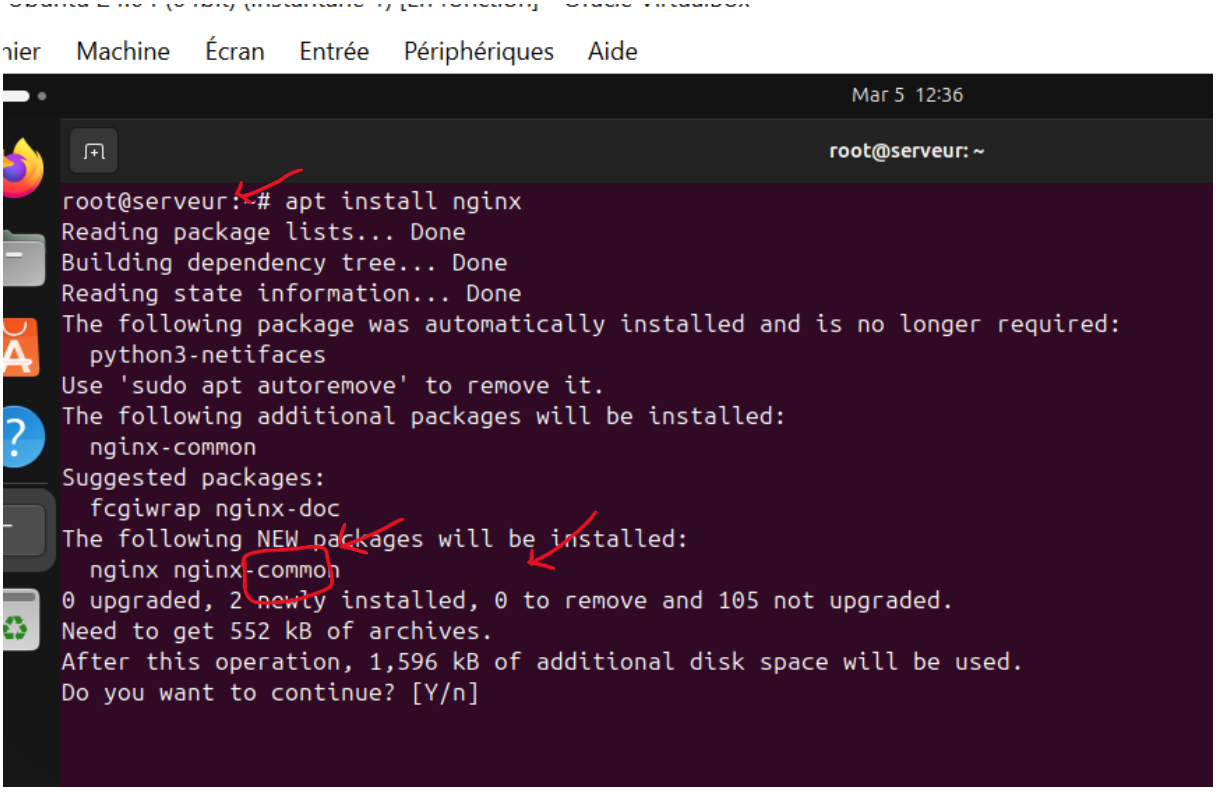


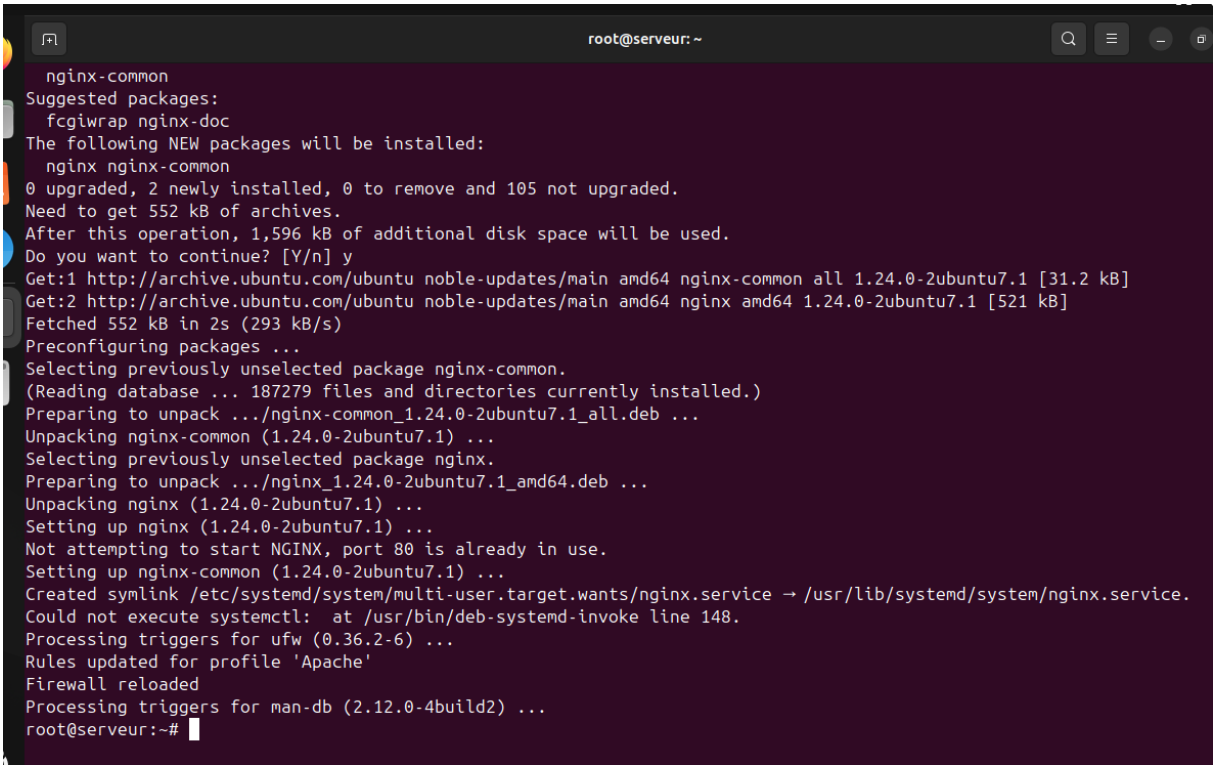
Constat : Le protocole https est bien fonctionnel avec notre site web sauf que le navigateur signale la présence de notre certificat en tant que certificat auto signé et non reconnue. De ce fait il va falloir paramétrer le navigateur pour qu'il facilite l'accès https malgré le certificat auto signé (Ajouter une exception) .

Chapitre 4 : Installation et configuration de Nginx

4.1. Téléchargement et installation de Nginx

➔ apt-get install nginx





Constat : Le paquet Nginx est bien installé mais si nous observons bien le résultat de la commande , le message affiché indique que le lancement du serveur Nginx rencontre un problème à cause d'un conflit au niveau du port 80.

Cela s'explique à cause de l'existence du serveur apache qui écoute occupe déjà le port 80 pour http et 443 pour https.

De ce fait, la prochaine séquence apportera pour solution le changement de port d'coute dans le fichier de configuration de Nginx .

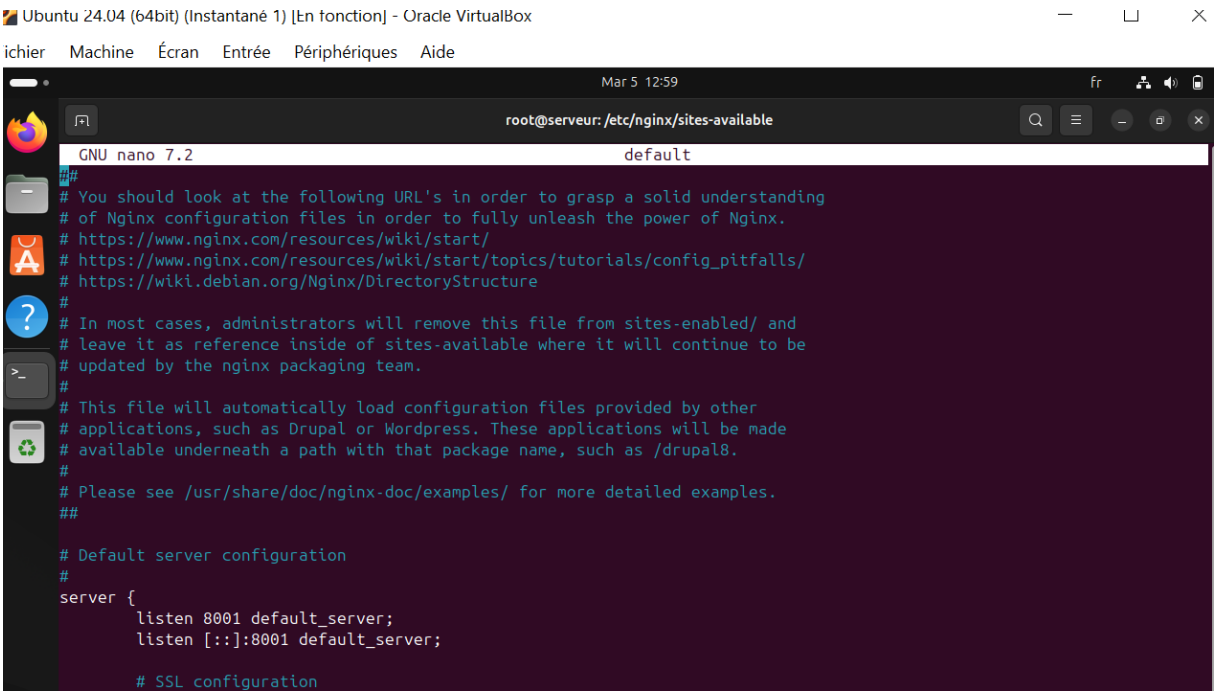
4.2. Changement du numéro de Port

Le fichier de configuration principal de Nginx est généralement situé dans /etc/nginx

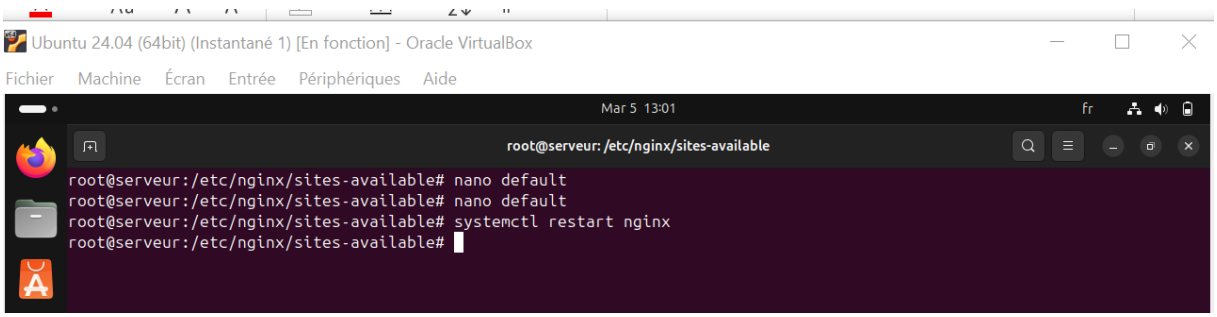
Nous chercherons le fichier « *default* » à l'intérieur des sous répertoires de ce répertoire . Ensuite nous l'ouvrirons et nous le modifierons pour changer les ports.

➔ . cd /etc/nginx/sites-available && nano default

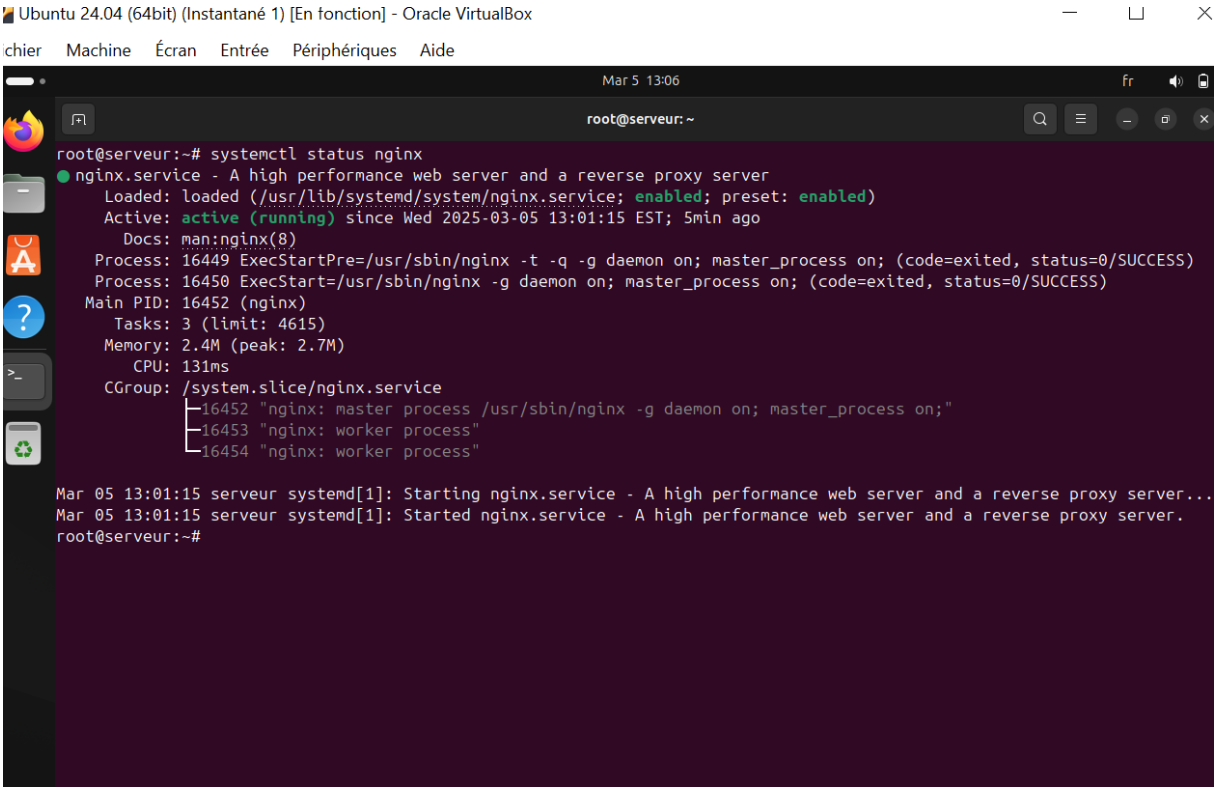
Constat : Nous changer le port d'écoute **80** contre le port **8001** choisi.



Maintenant que le numéro de port à été changé, nous allons redémarrer le serveur Nginx et accéder à la page index via un navigateur de notre choix.



4.3. Vérification du statut actif de Nginx



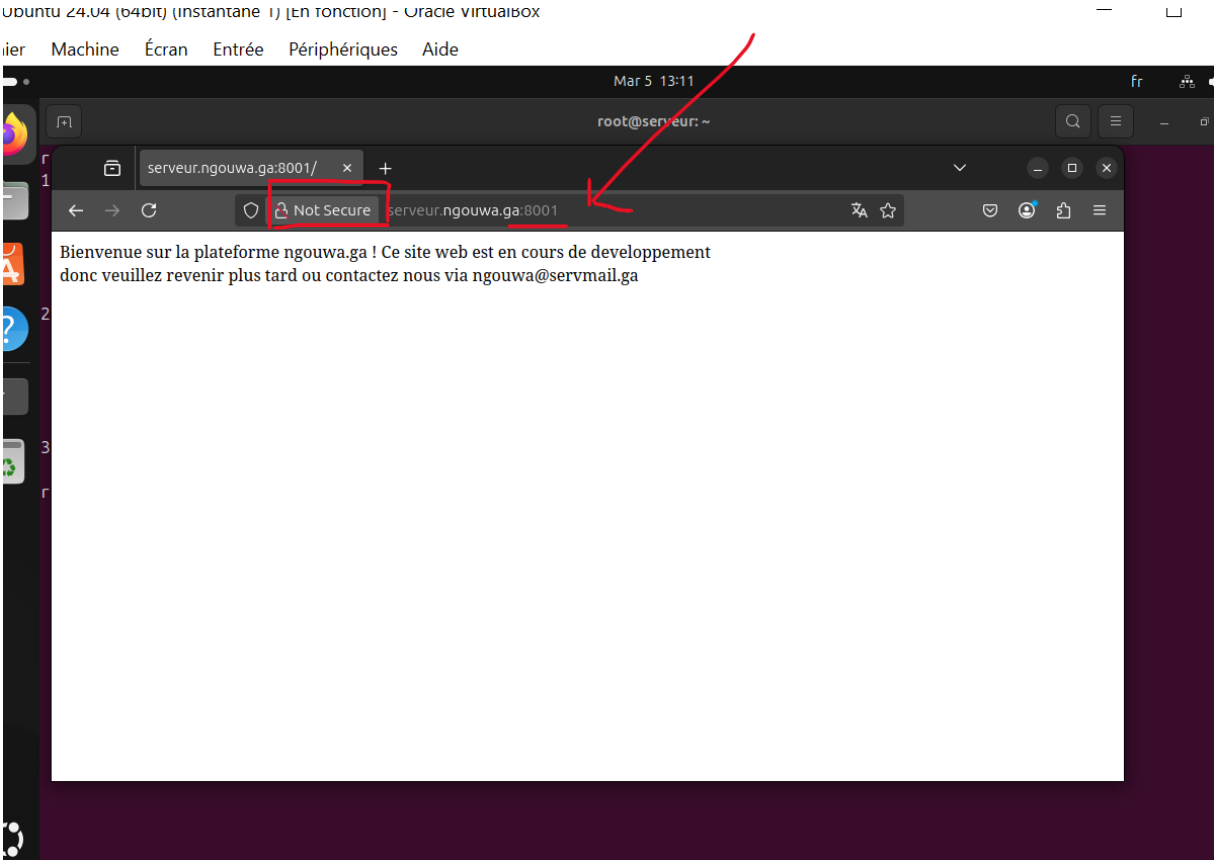
Constat : Maintenant que Nginx est actif nous donc désormais pouvons ouvrir le navigateur Firefox et accéder à notre domaine via le port 8001.

4.4. Accès au domaine via le port 8001

Pour cela , dans le cham url nous allons juste écrie :

http://serveur.ngouwa.ga:8001

Voir la capture ci-dessous



Constat : Notre domaine est bien accessible via le port 8001 mais la page web n'est pas sécurisé.

De ce fait, pour sécuriser l'accès à notre site web nous procéderons aux configurations de https pour Nginx.

4.5. Configuration de HTTPS

Configurer le Https consiste fondamentalement à la création de certificat SSL et à l'association au port d'écoute.

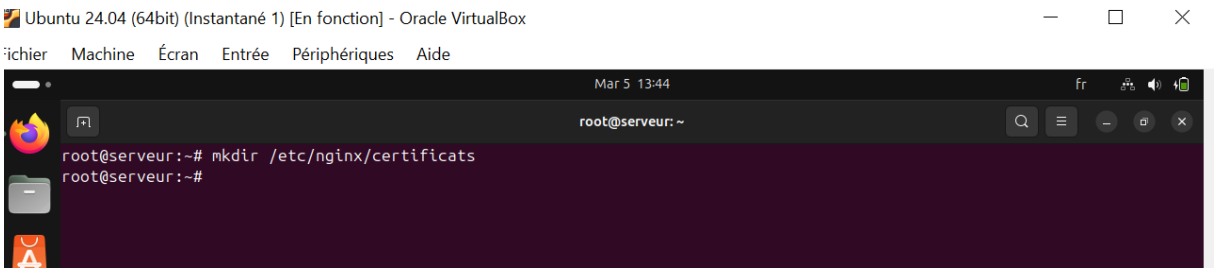
Nonobstant, nous savons que tout les serveurs web classique utilise le port 443 pour https et sachant bien que nous avons déjà un serveur apache qui écoute sur ce port, nous allons devoir positionner les certificats SSL de nginx sur un autre port de nos choix, le port 8443 .

Et contrairement à la méthode utilisé pour le serveur web apache2, nous utiliserons cette fois-ci l'outil cryptographique OpenSSL afin de créer une clé privé et un certificat auto-signé pour notre site web .

4.5.1. Création du dossier de sauvegarde du certificat

Dans le répertoire de Nginx, tapons la commande suivante :

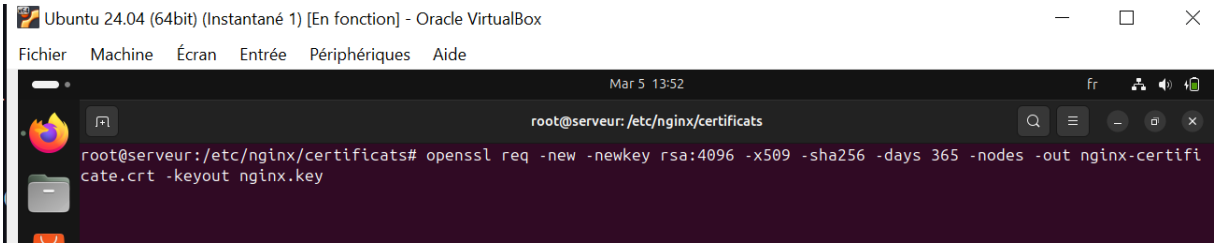
➔ . mkdir /etc/nginx/certificats

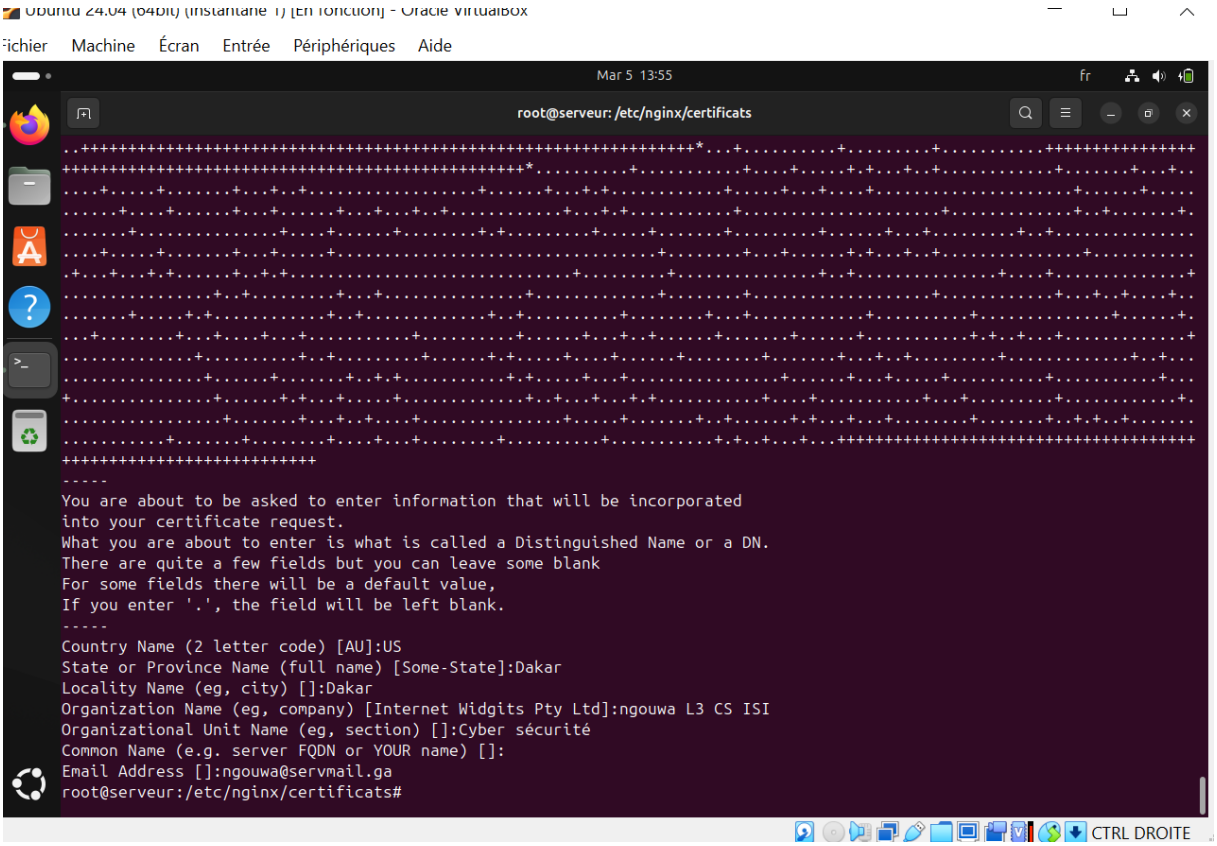


Constat : le dossier est créé, nous pouvons accéder au répertoire et ensuite y créer notre certificat.

4.5.2. Création de la clé privée et du certificat

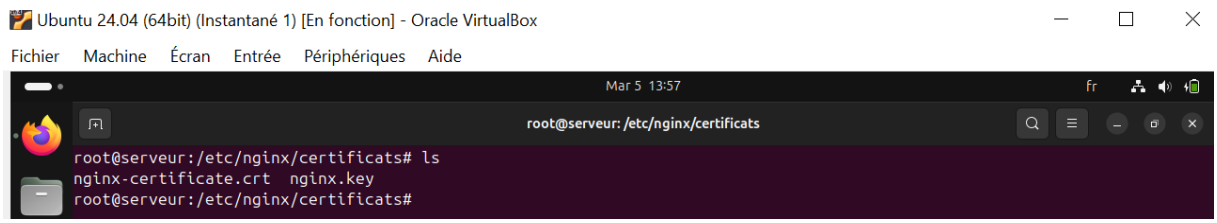
Pour la création du certificat et de la clé privée nous avons saisi la ligne de commande ci-dessous en capture :





Constat : OpenSSL demande quelques informations pour valider le certificats, nous avons saisi quelques une et ignoré d'autres.

Pour vérifier que cela a fonctionné, nous restons dans le même répertoire et nous tapons la commande « ls » afin d'afficher le certificat crée ainsi que sa clé.



Constat : Cela a bien été crée . En plus le nom du fichier de certificat ainsi que la clé nous seront utiles pour configurer le fichier nginx pour https.

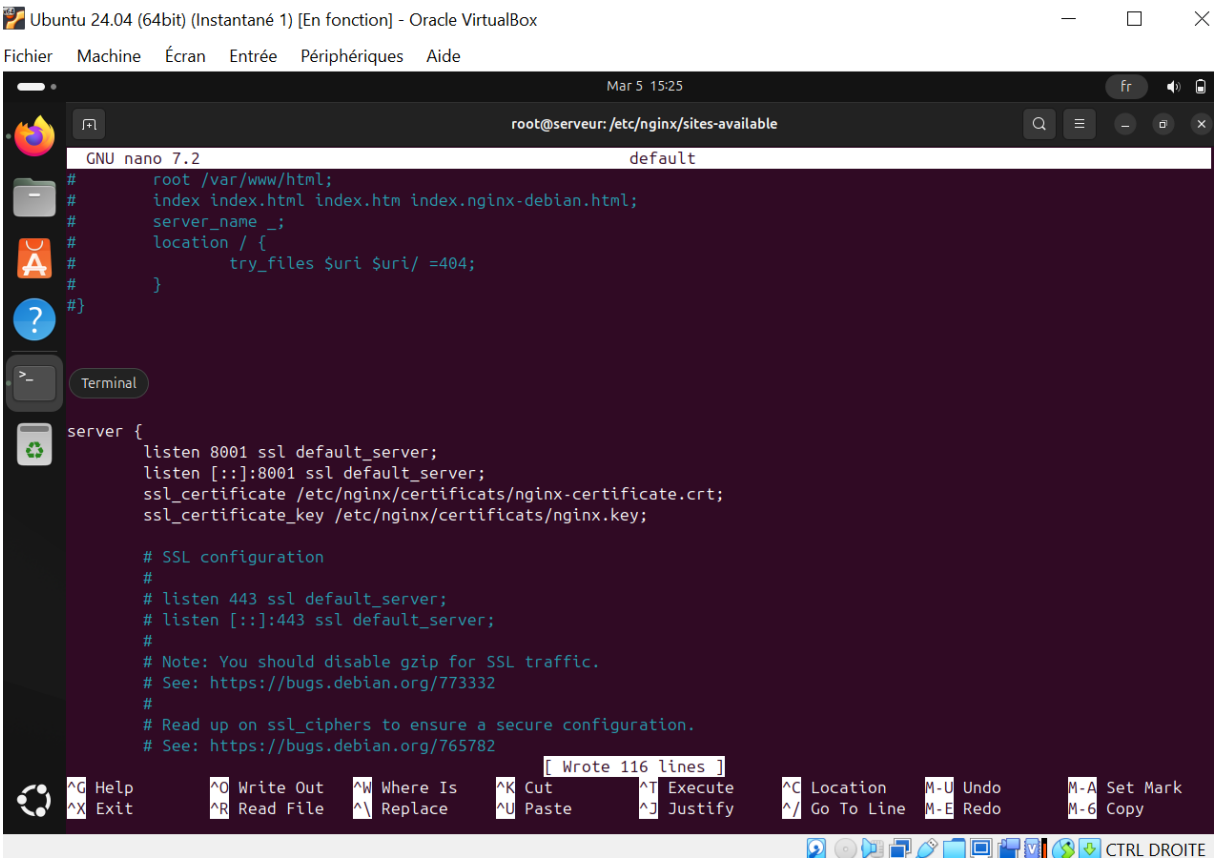
4.5.3 Mention du chemin du certificats dans le fichier default de Nginx

Pour cela nous retournons encore apporter des modifications dans le fichier de configuration par défaut de nginx. Par conséquent nous allons juste ajouter deux lignes à l'intérieur du fichier et dans le bloc de configuration du serveur.

ssl_certificate /etc/nginx/certificate/nginx-certificate.crt;

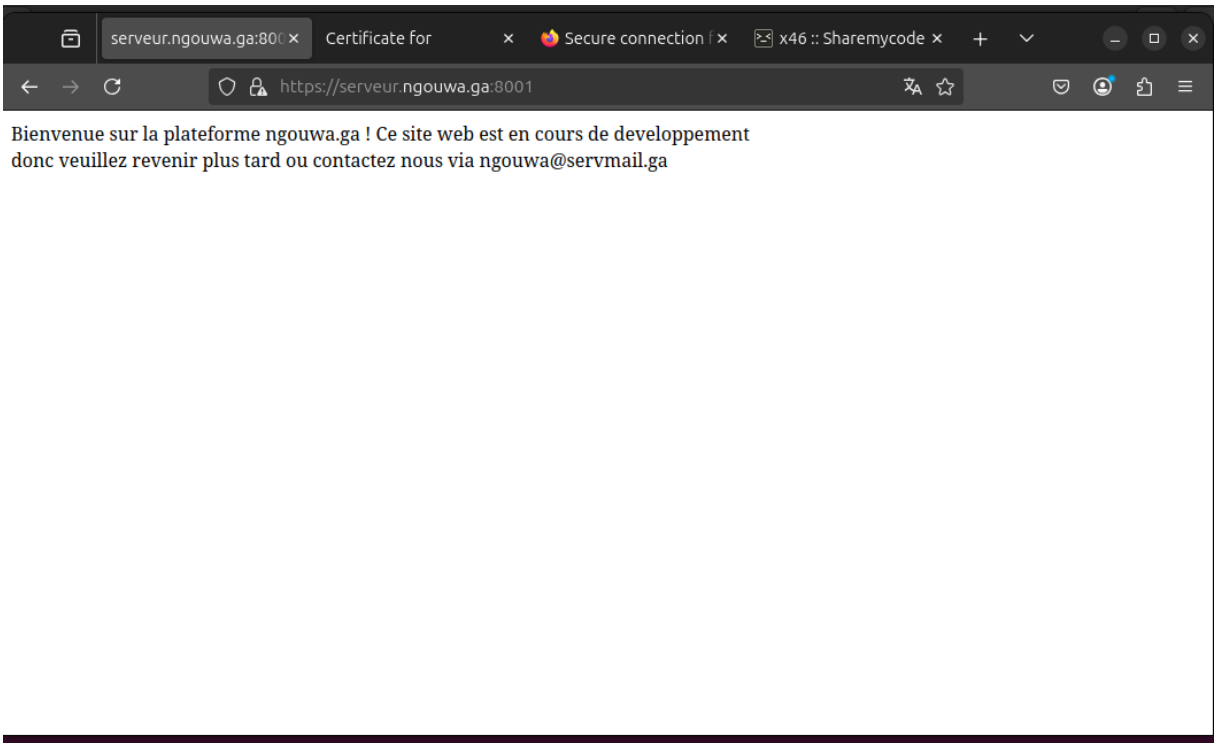
ssl_certificate_key /etc/nginx/certificate/nginx.key ;

➔ . nano /etc/nginx/sites-available/default

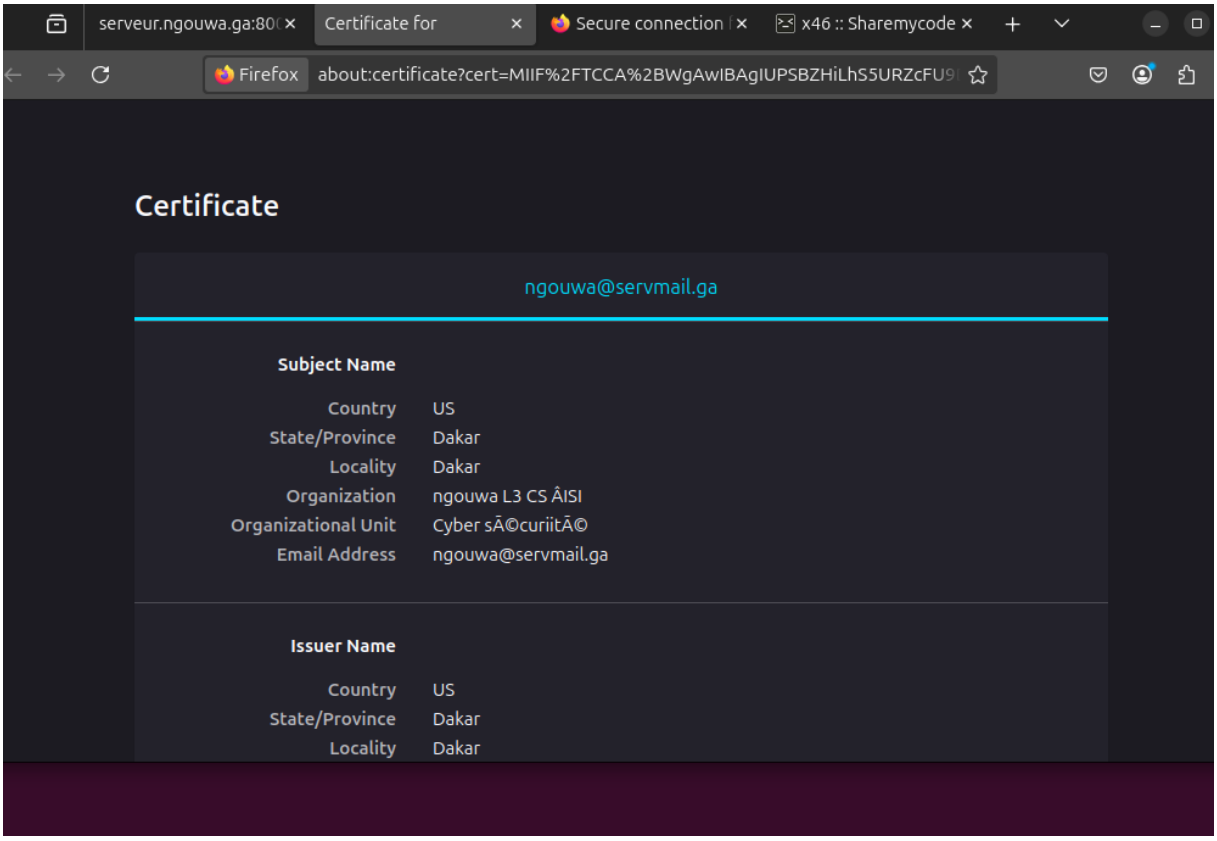


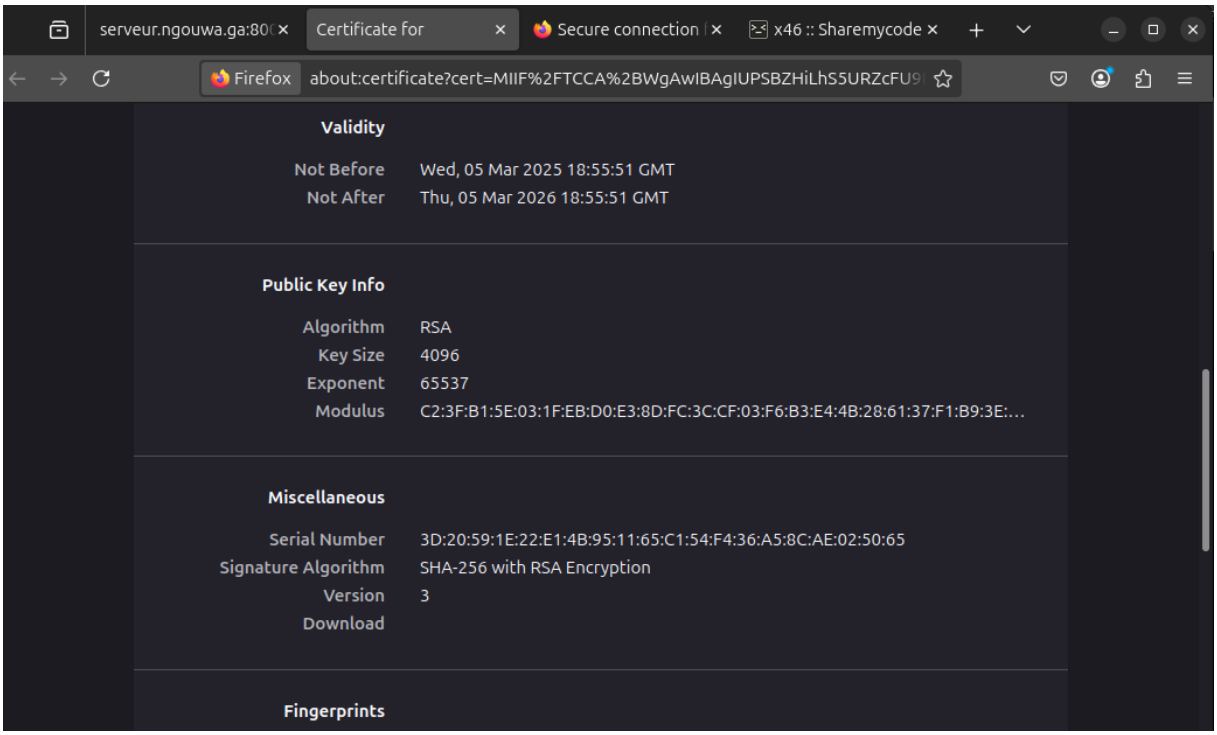
Constat : Dans ce bloc de ligne nous avons indiquer au protocole SSL le chemin vers notre certificat et notre clé privé pour chiffrer la communication entre les clients et le serveur nginx . Nous avons tout de même conservé le numéro de port 8001 qui sera désormais associé au protocole https à cause de la présence du certificat dans nos configurations.

Pour la suite du travail nous allons enregistrer le fichier, redémarrer le serveur Nginx et ensuite accéder à notre domaine via le port 8001 désormais sécurisé.



4.5.4 Affichage du certificat





Constat : Depuis le navigateur nous avons affichons les informations relatives à notre certificat. On peut y voir les informations personnelles saisit lors de la création (nom de l'organisation, mail etc..), il y a également le type d'algorithme de chiffrement dans notre c'est RSA 4096 et également SHA-256 pour la signature etc....

Conclusion

Dans le cadre de ce projet, nous avons réalisé l'installation et la configuration de deux serveurs web parmi les plus répandus, à savoir Apache2 et Nginx, sur un même serveur Ubuntu. L'objectif principal était de garantir la coexistence harmonieuse de ces services tout en prévenant les conflits de ports, et d'assurer la sécurité des communications grâce au protocole HTTPS.

Points clés accomplis :

- Installation et configuration d'Apache2 et Nginx :

- Les deux serveurs ont été installés avec succès sur le même serveur.

- Les ports d'écoute par défaut ont été modifiés afin d'éviter toute interférence. Ainsi, Apache2 écoute désormais sur des ports personnalisés (par exemple, 8080 pour HTTP

et 8443 pour HTTPS), tandis que Nginx a été configuré pour utiliser d'autres ports (comme 8081 pour HTTP et 8444 pour HTTPS).

- Activation du protocole HTTPS :

- Des certificats SSL/TLS ont été générés et configurés pour les deux serveurs, garantissant ainsi la sécurité des échanges.

- Les configurations SSL ont été rigoureusement testées et validées pour assurer un chiffrement efficace des données.

- Gestion des pare-feu et des ports :

- Les ports personnalisés ont été ouverts dans le pare-feu (ufw) afin de permettre un accès fluide aux services.

- Les règles du pare-feu ont été vérifiées pour garantir une sécurité maximale tout en permettant l'accès aux services web.

- Tests de fonctionnement :

- Chaque serveur a été testé individuellement pour confirmer son bon fonctionnement sur les ports attribués.

- Les connexions HTTPS ont été validées pour s'assurer que les certificats SSL/TLS étaient correctement appliqués.

Résultats :

Apache2 et Nginx opèrent simultanément sans conflit, chacun étant assigné à ses ports dédiés. Les services sont accessibles tant en HTTP qu'en HTTPS, offrant ainsi une flexibilité et une sécurité accrues. Cette configuration permet de bénéficier des atouts des deux serveurs web, tirant parti de la modularité d’Apache2 et des performances de Nginx dans un environnement unifié.

Perspectives d'amélioration :

- Déployer un reverse proxy (par exemple, avec Nginx) pour rediriger les requêtes vers Apache2 ou Nginx en fonction des besoins.

- Automatiser le renouvellement des certificats SSL/TLS via Let's Encrypt et Certbot.

- Surveiller les performances des deux serveurs pour optimiser leur utilisation en fonction de la charge.

-Configuration de php

-Configuration de MariaDB pour faire communiquer le site web avec une base de donnée sécurisé.

En somme, ce travail a démontré qu'il est tout à fait possible de faire coexister Apache2 et Nginx sur un même serveur tout en assurant une configuration sécurisée et optimisée. Cette approche confère une grande flexibilité pour répondre à des besoins spécifiques en matière de services web.