A quoi sert un reverse proxy ?

Le problème quand on héberge plusieurs services

Quand on commence à héberger des services sur un serveur, on se retrouve assez vite avec plusieurs applications différentes. Par exemple **un wiki**, **une interface** de supervision, **un cloud** personnel, un gestionnaire de mots de passe etc.

Au début, on peut se dire que chaque application est facilement accessible avec son **adresse IP** et son **port**. Par exemple une application peut répondre sur le port `8080`, une autre sur le port `3000`, et une autre sur le port `5000`, etc.

On se retrouve alors avec des adresses de ce genre :

http://192.168.1.50:8080
http://192.168.1.50:3000
http://192.168.1.50:5000

Techniquement, c'est vrai, ça fonctionne. Mais, ce n'est pas très propre, pas très pratique à retenir, et ce n'est pas forcément l'idéal niveau sécurité. C'est dans ce cas-là que l'on peut utiliser un **Reverse Proxy**.

Qu'est-ce qu'un reverse proxy

Un **Reverse Proxy** est un serveur qui se place devant **ns web**. Il reçoit les

demandes des utilisateurs, puis les transmet au bon service en interne.

L'utilisateur ne contacte donc pas directement l'application finale, il contacte le **reverse proxy**.

C'est lui qui va se charger de faire le lien avec la bonne application !

On peut l'imaginer comme par exemple, l'accueil d'un bureau administratif. Quand une personne arrive, elle ne connaît pas forcément le bureau ou elle doit se rendre. Elle donne simplement le nom de la personne ou du service qu'elle cherche, et l'accueil l'oriente au bon endroit. Le **reverse proxy** fait un peu la même chose, mais avec des sites web et des applications

Exemple concret avec plusieurs services

Si l'on reprend les adresses de tout à l'heure, imaginons que l'on ait trois services sur le même serveur

Un wiki sur le port 8080
Une supervision sur le port 3000
Une application web sur le port 5000

Sans **reverse proxy**, il faudait accéder directement à chaque port. Avec un **reverse proxy**, on peut utiliser des adresses beaucoup plus propres

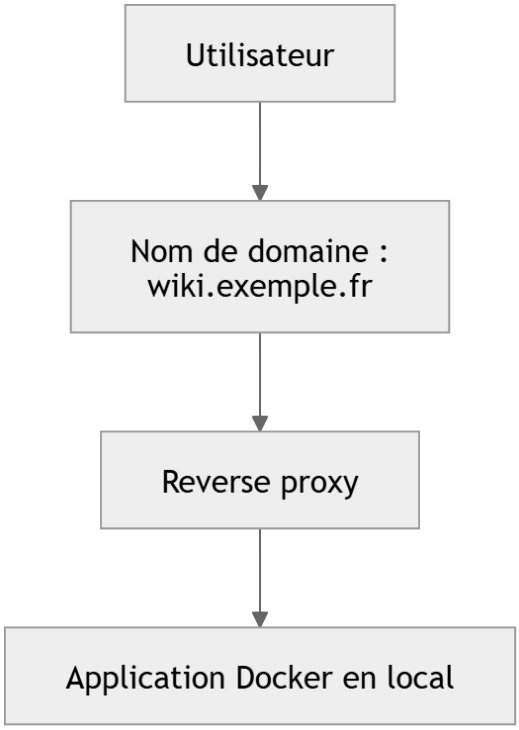
https://wiki.exemple.fr
https://supervision.exemple.

https://app.exemple.fr

Le **reverse proxy** reçoit la requête, regarde le **nom de domaine** demandé, puis l'envoie vers le bon service. Si l'utilisateur demande `wiki.exemple.fr`, le **reverse proxy**, paf ! Il connait où se trouve le wiki et transmet la demande. Si l'utilisateur demande `supervision.exemple.fr`, il transmet la demande vers l'outil de supervision. Tout ça peut se faire sur un seul et même serveur.

Le cas particulier de Docker

C'est particulièrement pratique quand on utilise **Docker**. Beaucoup d'application Docker écoutent sur un **port interne**. Par exemple, une application peut fonctionner dans un **conteneur** sur le port 3000. On pourrait directement exposer ce port mais ce n'est pas idéal. Avec un **reverse proxy**, on peut garder l'application accessible uniquement en local sur le serveur puis laisser le **reverse proxy** s'occuper de l'accès depuis l'extérieur.



L'utilisateur n'a pas besoin de savoir que l'application tourne sur le port 3000, dans un conteneur Docker, sur une machine précise, c'est tout sauf pratique. L'utilisateur, pour lui, il visite juste une **adresse web classique**.

Gestion du HTTPS et des certificats

Un autre avantage du **reverse proxy**, c'est la gestion du **HTTPS**. Quand on héberge un site ou une application, on veut généralement éviter le simple **HTTP**, surtout si des informations personnelles ou des identifiants peuvent être entrées. Le **reverse proxy** peut gérer les **certificats SSL/TLS** et permettre aux utilisateurs d'accéder aux services en **HTTPS**.

Ce qui est cool, c'est que l'application derrière le **reverse proxy** peut parfois fonctionner en HTTP uniquement en local. Le **reverse proxy**, lui, s'occupe de présenter une connexion **HTTPS** sécurisée à l'utilisateur.

On peut donc avoir quelque chose comme ceci :

Utilisateur → HTTPS → Reverse proxy → HTTP local → Application

Vu de l'extérieur, la connexion est sécurisée. En interne sur le serveur, le **reverse proxy** discute avec l'application. Il permet donc de simplifier la gestion des certificats. Super bien quand on héberge plusieurs services

Réduction des ports exposés

Le **reverse proxy** permet également d'éviter d'ouvrir trop de **ports** vers internet. Au lieu d'exposer les ports 3000,5000,8080 et ainsi de suite, on expose juste les ports "classiques du web".

80 → HTTP
443 → HTTPS

C'est tout ! Ensuite, c'est le **reverse proxy** qui va s'occuper de rediriger les demandes vers les bons ports internes. Ça permet d'avoir une architecture beaucoup plus propre. Les applications ne sont pas toutes visibles depuis internet, et le **reverse proxy** devient le point d'entrée principal

Les outils de reverse proxy

Il existe plusieurs outils capables de faire **reverse proxy**. Le plus connu est probablement **Nginx**, très utilisé sur les serveurs **Linux**. **Apache** peut aussi faire ce rôle. **Traefik** est souvent utilisé avec **Docker**, car il peut détecter automatiquement les conteneurs et créer les routes correspondantes. **Caddy** est apprécié pour sa simplicité et sa gestion automatique du **HTTPS**. Il existe aussi **Nginx Proxy Manager**, qui permet de gérer des redirections depuis une interface web, ce qui peut être plus accessible quand on ne veut pas écrire toute la configuration à la main.

Reverse proxy vs proxy classique

Il faut aussi faire la différence avec un **proxy classique**. Un proxy classique est plutôt utilisé côté utilisateur. Par exemple, un ordinateur peut passer par un proxy pour accéder à Internet. Le **reverse proxy**, lui, est côté serveur. Il reçoit les demandes qui viennent d’Internet et les transmet aux applications hébergées derrière lui.

On peut résumer simplement comme ceci :

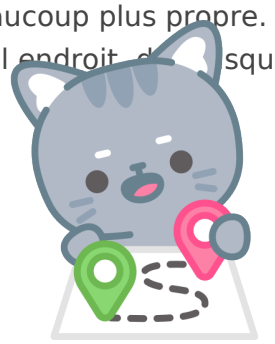
```
Proxy classique :
Ordinateur → Proxy → Internet


Reverse proxy :
Internet → Reverse proxy → Applications
```

Le proxy classique aide un utilisateur à sortir vers Internet. Le **reverse proxy** aide les utilisateurs à entrer proprement vers des services hébergés.

Résumons

Au final, un **reverse proxy** sert surtout à rendre l'hébergement de plusieurs applications beaucoup plus propre. Il permet d'utiliser de beaux **noms de domaine**, de gérer le **HTTPS** à un seul endroit, de masquer les ports internes, et d'envoyer chaque demande vers le



bon service.

C'est un outil discret, mais extrêmement important dans une **infrastructure web**. Quand tout fonctionne bien, on ne le remarque presque pas. Pourtant, sans lui, beaucoup d'installations seraient vite moins pratiques, moins propres et plus compliquées à maintenir.

En résumé, le **reverse proxy** est un peu le GPS de votre serveur. L'utilisateur arrive avec une adresse, comme `wiki.exemple.fr`, et le **reverse proxy** sait exactement vers quelle application l'envoyer. Il simplifie l'accès, améliore l'organisation, et permet d'héberger plusieurs services proprement derrière une seule **porte d'entrée**.

Révision #2
Créé 2026-07-04 14:29:15 UTC par Renard
Mis à jour 2026-07-04 15:07:17 UTC par Renard