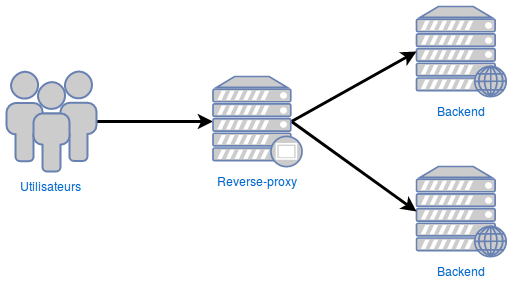
# Mise en place d’un serveur proxy inversé

## Présentation du proxy inverse Nginx HTTPS

Un proxy inverse (*reverse proxy*) est un type de serveur, habituellement placé en frontal de serveurs web. Contrairement au serveur proxy qui permet à un utilisateur d'accéder au réseau Internet, le proxy inverse permet à un utilisateur d'Internet d'accéder à des serveurs internes.

Le proxy inverse est simple à mettre en œuvre et offre à l'utilisateur une sécurité haut de gamme contre les attaques de serveurs Web telles que DDoS et DoS.

Le proxy inverse est installé du côté des serveurs Internet.  
L'utilisateur du Web passe par son intermédiaire pour accéder aux applications de serveurs internes.   
Le proxy inverse est parfois appelé substitut (*surrogate*).



Un proxy inverse HTTP ou HTTPS est donc un service mandataire intermédiaire qui prend une demande client, la transmet à un ou plusieurs serveurs, puis renvoie la réponse du serveur au client.

Alors que la plupart des applications courantes peuvent fonctionner en tant que serveur Web seules, le serveur Web nginx est capable de fournir un certain nombre de fonctionnalités avancées telles que l'équilibrage de charge, les capacités TLS/SSL et l'accélération qui manquent à la plupart des applications spécialisées.

Apache et nginx sont deux serveurs Web open source populaires. Il peut être utile de les exécuter tous les deux sur la même machine lors de l'hébergement de plusieurs sites Web ayant des exigences variées.   
La solution générale pour exécuter deux serveurs Web sur un seul système consiste à utiliser plusieurs adresses IP ou des numéros de port différents.

## Proxy régulier ou Proxy inversé

**Proxy régulier**  
Bien que ce soit un terme très répandu dans la communauté technologique, ce n'est pas le seul endroit où il est utilisé. Un proxy signifie que les informations transitent par un tiers, avant d'arriver à l'emplacement.  
Si on ne souhaite qu'un service connaisse son adresse IP, on peut utiliser un proxy.   
Un proxy est un serveur qui a été configuré spécifiquement à cet effet.   
Si le serveur proxy que l’on utilise se trouve, par exemple, à New York, l'adresse IP qui sera affichée au monde extérieur est l'adresse IP du serveur à New York. Les seuls qui connaîtront son adresse IP sont ceux qui contrôlent le serveur proxy.

**Proxy inversé**

Un proxy ajoute une couche de masquage.   
C'est le même concept dans un proxy inverse, à la différence qu'au lieu de masquer les connexions sortantes (on accède à un serveur web), ce sont les connexions entrantes (personnes accédant à un serveur web) qui seront masquées.   
On fournit simplement une URL comme cmaisonneuve.qc.ca, et chaque fois que des personnes accèdent à cette URL, le proxy inverse se chargera de rediriger la demande vers la bonne destination.

## nginx

nginx est un logiciel open source pour le service Web, le proxy inverse, la mise en cache, l'équilibrage de charge, la diffusion multimédia en continu, …

Il a commencé comme un serveur Web conçu pour des performances et une stabilité maximale. En plus de ses capacités de serveur HTTP, nginx peut également fonctionner comme un serveur proxy pour le courrier électronique (IMAP, POP3 et SMTP) et un proxy inverse et un équilibreur de charge pour les serveurs HTTP, TCP et UDP.

nginx ne nécessite pas la configuration d'un nouveau processus pour chaque requête Web du client. Au contraire, la configuration par défaut consiste à comprendre un processus de travail par processus.

Il peut gérer plus de 10 000 connexions avec une faible empreinte mémoire. nginx peut exploiter plusieurs serveurs Web via une seule adresse IP et envoyer chaque requête au bon serveur au sein d'un LAN.

nginx est l'un des meilleurs serveurs Web pour améliorer les performances du contenu statique. En outre, il peut également être utile de diffuser le contenu du cache et d'effectuer un chiffrement SSL pour réduire la charge du serveur Web.

Cela peut également être utile lors de l'optimisation du contenu en le compressant pour augmenter le temps de chargement

nginx peut effectuer des expériences aléatoires ou des tests A/B[[1]](#footnote-1) sans placer de codes JavaScript dans les pages.

## Avantages d’utilisation

**Équilibrage de charge**Un serveur d’origine unique ne peut pas gérer tout le trafic entrant pour un site web qui compte des millions de visiteurs uniques par jour. Dans ce cas, il est possible de répartir le trafic de manière intelligente entre un ensemble de plusieurs serveurs. En général, tous les serveurs hébergent le même contenu afin d’éliminer un point de défaillance, ce qui rend le site web plus fiable.  
Un proxy inverse peut recevoir le trafic entrant avant qu’il n’atteigne le serveur dorsal (*backend*). Si le serveur d’origine est surchargé ou tombe complètement en panne, il peut distribuer le trafic à d’autres serveurs sans affecter la fonctionnalité du site.  
Le proxy inverse peut également diriger les requêtes entrantes vers plusieurs serveurs, chaque serveur remplissant une fonction spécifique pour laquelle il est optimisé. Le proxy inverse peut alors recueillir les réponses de tous les serveurs et les transmettre au client.  
La plupart des proxys inversé sont principalement pour l’équilibrage de charge, On les identifie alors comme répartiteurs de charge (***load balancers***).  
Les répartiteurs de charge globale des serveurs (*Global Server Load* ou GSLB) permettent de répartir le trafic d’un site web entre de nombreux serveurs placés stratégiquement dans le monde entier. Ils augmentent considérablement la fiabilité et la sécurité du site, mais ils réduisent également les temps de latence et de chargement

**Sécurité renforcée**Le proxy inverse peuvent masquer l’adresse IP et d’autres caractéristiques des serveurs dorsaux. Ainsi, le serveur dorsal d’un site web peut mieux préserver son anonymat, ce qui augmente considérablement sa sécurité.  
Étant donné que le proxy inverse recevra tout le trafic avant qu’il n’atteigne un serveur dorsal, tout attaquant aura plus de mal à cibler le site web avec des menaces de sécurité telles que les attaques DDoS.

**Services sur différents ports**Un des plus grands avantages d'avoir un proxy inverse, c'est qu’il est possible d’avoir des services fonctionnant sur une multitude de ports, mais que l’on a seulement qu'à ouvrir les ports 80 et 443, HTTP et HTTPS respectivement.   
Toutes les demandes arriveront sur le réseau sur ces deux ports et le proxy inverse se chargera du reste.

**Mise en cache**Il est possible d’utiliser un proxy inverse à des fins d’accélération du web en mettant en cache à la fois le contenu statique et dynamique. Cela peut réduire la charge sur le serveur dorsal, ce qui se traduit par un site web plus rapide.

**Compression supérieure**Les réponses des serveurs consomment beaucoup de bande passante. Comprimer les réponses du serveur (par exemple avec gzip) avant de les envoyer au client peut réduire la quantité de bande passante nécessaire, ce qui accélère les réponses du serveur sur le réseau.  
Un proxy inverse est idéal pour compresser les réponses des serveurs car il se situe entre les serveurs d’origine et le client.

**Chiffrement SSL**Le chiffrement et le déchiffrement des requêtes SSL/TLS pour chaque client peuvent être très éprouvants pour un serveur dorsal. Un proxy inverse peut prendre en charge cette tâche afin de libérer les ressources du serveur d’origine pour d’autres tâches importantes, comme la diffusion de contenu.  
Un autre avantage du délestage du chiffrement et du déchiffrement SSL/TSL est de réduire la latence pour les clients qui sont géographiquement éloignés du serveur d’origine.  
On peut également opter pour un proxy inverse avec un matériel d’accélération SSL/TLS spécialisé afin d’optimiser encore plus cette tâche.   
Un tel proxy inverse est appelé proxy de terminaison SSL/TLS.

**Meilleur test A/B**La plupart des outils de test A/B nécessitent l’utilisation de bibliothèques JavaScript externes pour charger leurs fonctions. Cependant, le chargement de scripts tiers peut ralentir le temps de chargement d’une page et créer une expérience difficile pour les utilisateurs.  
Au lieu de cela, on peut utiliser un proxy inverse pour créer deux flux séparés au niveau du serveur lui-même.

**Surveillance et journalisation du trafic**Un proxy inverse saisit toutes les requêtes qui passent par lui.   
On peut donc les utiliser comme un concentrateur (*hub*) central pour surveiller et journaliser le trafic.   
Même si on utilise plusieurs serveurs web pour héberger tous les éléments d’un site, l’utilisation d’un proxy inverse facilitera la surveillance de toutes les données entrantes et sortantes du site.

## Limites d’un proxy inverse

**Sites HTTPS**Un proxy inverse pose un risque de sécurité important car il peut lire et modifier tout le trafic qui le traverse.   
Si on fait passer du trafic HTTPS par le serveur proxy inverse, celui-ci doit déchiffrer et chiffrer à nouveau les données qui passent.   
Cela signifie qu’il doit posséder les clés privées du certificat SSL/TLS.   
Ainsi, si une partie malveillante peut compromettre le proxy inverse, elle peut enregistrer des mots de passe et injecter des logiciels malveillants dans les sites web.

**Point de défaillance unique**Si l’administrateur ou les utilisateurs ne peuvent pas accéder directement au(x) serveur(s) dorsal(aux), l’utilisation d’un proxy inverse peut conduire à un point de défaillance.   
Par exemple, si on utilise un proxy inverse comme serveur frontal pour desservir plusieurs domaines, sa panne peut entraîner la mise hors ligne simultanée de tous les domaines.

**Informations sensibles**Si vous faites appel à un mandataire tiers, on lui transmet des informations sensibles de son site.   
Bien qu’on leur fasse confiance, on ne peut pas prévoir ce qui peut en résulter.

**Sauvegardes**  
La restauration des sauvegardes ou le fait de pousser les sites de *staging* en production sur les sites web qui se chargent par un proxy inverse peut entraîner l’arrêt du chargement correct du site proxy.

## Serreurs proxys populaires

Il existe plusieurs serveurs proxys inverses. Parmi les plus utilisés …

* **nginx** ;
* Varnish ;
* **HAproxy**
* Apache (avec mod\_proxy) ;
* Apache Traffic Server ;
* …

## Mise en place préliminaire Serveur Apache dorsal

Pour cet exemple, on va configurer un proxy inverse nginx (frontal ou *frontend*) devant un serveur Web Apache   
(dorsal ou *backend*).   
En conséquence, il est supposé qu'Apache est déjà installé et configuré …

|  |  |
| --- | --- |
| **Configuration du serveur Apache** | |
| Sur le même hôte …  Modification du fichier /etc/apache2/ports.conf Il va falloir d’abord configurer Apache pour écouter sur d’autres ports. Pour modifier les ports sur lesquels écoute Apache … 7080 pour le HTTP (au lieu de 80) ; 7443 pour le HTTPS (au lieu de 443).  On modifie les fichiers du répertoire /etc/apache2/sites-available/001-default.conf  Pour vérifier la bonne configuration du fichier  On redémarre Apache  (ou on recharge les fichiers de configuration)  On vérifie qu'Apache écoute maintenant 7080  Sur un hôte différent … Aucune modification n’est nécessaire | **Contenu du fichier Listen 7080**  **<IfModule ssl\_module>  Listen 7443 </IfModule>**  **<IfModule mod\_gnutls.c>  Listen 7443 </IfModule>**  **Configuration des hôtes virtuels** (*virtual hosts*)  **<VirtualHost \*:7080>  ServerAdmin webmaster@localhost  DocumentRoot /var/www/html  ErrorLog ${APACHE\_LOG\_DIR}/error.log  CustomLog ${APACHE\_LOG\_DIR}/access.log combined </VirtualHost>**  **>>** **sudo apachectl -t AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message**  **Syntax OK**  **>>** **sudo systemctl restart|reload apache2.service**  **>>** **sudo netstat -tlpn**  **tcp6 0 0 :::7080 :::\* LISTEN 11760/apache2** |

|  |  |
| --- | --- |
| **Installation et configuration rapide de nginx** | |
| On continue avec l’installation du serveur frontal.  On met à jour le cache de paquets APT et on installe le serveur nginx à l’aide du gestionnaire de paquets  On désactive l'hôte virtuel par défaut, qui est préconfiguré lorsque nginx est installé à l’aide du gestionnaire de paquets  On accède au répertoire /etc/nginx/sites-available et on crée un fichier de configuration de proxy inversé  On saisit la configuration nginx suivante dans l'éditeur de texte.  Le serveur proxy redirige toutes les connexions entrantes sur le port 80 vers le serveur Apache, qui écoute sur le port 7080.  On modifie la valeur du port en fonction du port spécifique de l'application.  Remarques … L’adresse locale est 127.0.0.1. Si on fait un lien vers un serveur externe on indique son adresse ou son nom DNS. Les accès et les erreurs se trouvent dans un fichier journal à l'adresse /var/log/nginx. Le serveur Apache, s’il est situé sur un autre hôte, peut écouter sur le port 80.  On copie la configuration de /etc/nginx/sites-available vers /etc/nginx/sites-enabled.  Il est recommandé d'utiliser un lien symbolique  On teste le fichier de configuration nginx    Afin de vérifier le bon fonctionnement, on ouvre un fureteur Web (browser) sur l’ordinateur local et entre l’adresse IP du serveur nginx, qui affichera sa page d'accueil. | **>>** **sudo** **apt update >>** **sudo** **apt install nginx**  **>>** **sudo** **unlink /etc/nginx/sites-enabled/default**  **>>** **cd /etc/nginx/sites-available** **>>** **sudo** **nano reverse-proxy** **server {**  **listen 80;**  **listen [::]:80;**  **access\_log /var/log/nginx/reverse-access.log;**  **error\_log /var/log/nginx/reverse-error.log;**   **location / {**  **proxy\_pass http://127.0.0.1:7080;**  **}**  **}**    **>>** **sudo** **ln -s /etc/nginx/sites-available/reverse-proxy /etc/nginx/sites-enabled/reverse-proxy**  **>>** **sudo** **nginx -t nginx: the configuration file /etc/nginx/nginx.conf syntax is ok nginx: configuration file /etc/nginx/nginx.conf test is successful**  **>>** **sudo** **systemctl restart nginx.service** |

## Installation d’un serveur proxy inversé comme répartiteur de charge

L'équilibrage de charge sur plusieurs instances d'application est une technique couramment utilisée afin d’optimiser l'utilisation des ressources, maximiser le débit, réduire la latence et garantir des configurations tolérantes aux pannes.

Il est possible d'utiliser nginx comme un équilibreur de charge HTTP très efficace pour distribuer le trafic vers plusieurs serveurs d'applications et pour améliorer les performances, l'évolutivité et la fiabilité des applications Web avec nginx.

### Méthodes d'équilibrage de charge

Les mécanismes (ou méthodes) d'équilibrage de charge suivants sont pris en charge avec nginx …

* **round-robin** (par défaut)  
  Les requêtes adressées aux serveurs d'applications sont distribuées de manière circulaire ;
* **le moins connecté** (*least-connected*)  
  La demande suivante est attribuée au serveur avec le moins de connexions actives ;
* **ip-hash**  
  Une fonction de hachage est utilisée pour déterminer quel serveur doit être sélectionné pour la prochaine requête (en fonction de l'adresse IP du client).

### Configuration d'équilibrage de charge par défaut

La configuration la plus simple pour l'équilibrage de charge avec nginx peut ressembler à ce qui suit …  
**>>** **sudo nano /etc/nginx/sites-available/default**

**upstream backend {**  
 **server srv01.<Nom de domaine>:<port>;**  
 **server srv02.<Nom de domaine>:<port>;**  
 **server srv03.<Nom de domaine>:<port>;**  
 **}**

**server {  
 listen 80;**

**location / {  
 proxy\_pass http://backend;  
 }**

**}**

Dans l'exemple ci-dessus, 3 instances de la même application s'exécutent sur srv01, srv01 et srv03.   
Lorsque la méthode d'équilibrage de charge n'est pas spécifiquement configurée, elle est définie par défaut sur round‑robin. Toutes les demandes sont envoyées par proxy au groupe de serveurs monapp1 et nginx applique l'équilibrage de charge HTTP pour distribuer les demandes.

**Exemple** ...

**upstream backend {**  
 **server 192.168.1.142:80;**  
 **server 192.168.1.143:80;**  
 **server 192.168.1.144:80;**  
 **}**

**server {**  
 **listen 80;**  
 **listen [::]:80;**  
 **access\_log /var/log/nginx/reverse-access.log;**  
 **error\_log /var/log/nginx/reverse-error.lo**g;

**location / {**  
 **proxy\_pass** [**http://backend**](http://backend)**;**  
 **}**  
**}**

### Plus de détails

|  |  |
| --- | --- |
| Directive | Description |
| upstream backend | Cette directive définit un bloc “upstream” dont le nom est backend Ce bloc contient les adresses des serveurs dorsaux (de *backend*)  Si plusieurs adresses sont configurées (une directive server par ligne), nginx va automatiquement répartir la charge entre ces serveurs |
| server | Cette directive est l’équivalent de <VirtualHost /> d’Apache C’est dans ce bloc que l’on configure le site |
| listen | Spécifie l’adresse IP et le port correspondant |
| location | Cette précise les règles qui s’appliquent aux adresses qui commencent par /  (après le nom de domaine) |
| proxy\_pass | Cette directive indique d’agir en tant que proxy et de transmettre les requêtes avec le protocole HTTP vers les serveurs définis dans le bloc upstream |

**Conseil**…  
Le **port** est optionnel.

**Remarque** …  
Pour configurer l'équilibrage de charge pour HTTPS au lieu de HTTP, on utilise simplement https comme protocole.

### Équilibrage de charge le moins connecté (*least-connected*)

Une autre sorte d'équilibrage de charge est le moins connectée.   
Le moins connecté permet de contrôler plus équitablement la charge sur les instances d'application dans une situation où certaines des requêtes prennent plus de temps à se terminer.

Avec l'équilibrage de charge le moins connecté, nginx essaiera de ne pas surcharger un serveur d'applications occupé avec des requêtes excessives, en distribuant les nouvelles requêtes à un serveur moins occupé à la place.

Cette méthode fonctionne bien lors du traitement d'un mélange de demandes rapides et complexes.

L'équilibrage de charge le moins connecté dans nginx est activé lorsque la directive less\_conn est utilisée dans le cadre de la configuration du groupe de serveurs …

**upstream bachend {**  
 **least\_conn;**  
 **server srv01.<Nom de domaine>:<port>;**  
 **server srv02.<Nom de domaine>:<port>;**  
 **server srv03.<Nom de domaine>:<port>;**  
**}**  
**…**

### Persistance de session

Avec l'équilibrage de charge circulaire ou le moins connecté, la demande de chaque client suivant peut être potentiellement distribuée à un serveur différent. Il n'y a aucune garantie que le même client sera toujours dirigé vers le même serveur.

S'il est nécessaire de lier un client à un serveur d'applications particulier - en d'autres termes, rendre la session du client collante (*sticky*) ou persistante (*persistent*) afin de toujours essayer de sélectionner un serveur particulier - le mécanisme d'équilibrage de charge ip-hash peut être utilisé.

Avec ip-hash, l'adresse IP du client est utilisée comme clé de hachage pour déterminer quel serveur d'un groupe de serveurs doit être sélectionné pour les demandes du client. Cette méthode garantit que les demandes du même client seront toujours dirigées vers le même serveur sauf lorsque ce serveur est indisponible.

Pour configurer l'équilibrage de charge ip-hash, ajoutez simplement la directive ip\_hash à la configuration du groupe de serveurs (en amont):

**upstream backend {  
 ip\_hash;** **server srv01.<Nom de domaine>:<port>;  
 server srv02.<Nom de domaine>:<port>;  
 server srv03.<Nom de domaine>:<port>;  
}  
…**

### Équilibrage de charge pondéré

Il est également possible d'influencer davantage les algorithmes d'équilibrage de charge nginx en utilisant des pondérations de serveur.

Dans les exemples précédents, les pondérations des serveurs ne sont pas configurées, ce qui signifie que tous les serveurs spécifiés sont traités comme également qualifiés pour une méthode d'équilibrage de charge particulière.

Avec le round-robin en particulier, cela signifie également une répartition plus ou moins égale des requêtes sur les serveurs - à condition qu'il y ait suffisamment de requêtes, et lorsque les requêtes sont traitées de manière uniforme et traitées assez rapidement.

Toutes les méthodes d'équilibrage de charge peuvent être réglées à l'aide d'un paramètre facultatif **weight** dans la directive **server**. Cela a du sens lorsque les serveurs ont des capacités de traitement différentes.

Lorsque le paramètre de poids est spécifié pour un serveur, le poids est pris en compte dans le cadre de la décision d'équilibrage de charge.

**upstream backend {  
 ip\_hash;** **server srv01.<Nom de domaine>:<port> weight=3;  
 server srv02.<Nom de domaine>:<port>;  
 server srv03.<Nom de domaine>:<port>;  
}  
…**

Avec cette configuration, toutes les 5 nouvelles demandes seront distribuées sur les instances d'application comme suit …

* 3 demandes seront dirigées vers srv01 ;
* une demande ira vers srv02   
  et
* une autre vers srv03.

Il est également possible d'utiliser des pondérations avec l'équilibrage de charge le moins connecté et le hachage IP dans les versions récentes de nginx.

### Équilibrage de charge et HTTPS

Afin que le serveur frontal ngnix utilise le protocole HTTPS, il suffit de lui ajouter les paramètres afin d’utiliser ce protocole.

On ajoute (ou modifie) au fichier de configuration de l’hôte virtuel la section suivante …  
**server {  
listen 443 ssl;  
 ssl\_certificate <Chemin et nom du fichier du certificat>  
 ssl\_certificate\_key <Chemin et nom du fichier de la clé privée>  
…  
}**

### Bilans de santé (*health check*)

S'il y a une erreur ou un délai d'expiration lorsque nginx tente de se connecter à un serveur dorsal, de lui transmettre une requête ou de lire l'en-tête de réponse, il réessaye la demande de connexion avec un autre serveur.

L'implémentation du proxy inverse dans nginx inclut des vérifications de l'état du serveur intrabande (ou passives).   
Si la réponse d'un serveur particulier échoue avec une erreur, nginx marquera ce serveur comme ayant échoué et essaiera d'éviter de sélectionner ce serveur pour les demandes entrantes ultérieures pendant un certain temps.

Il est possible d’inclure la directive **proxy\_next\_upstream** dans la configuration afin de définir d'autres conditions pour réessayer la demande.

De plus, nginx peut retirer le serveur défaillant de l'ensemble des serveurs et envoyer occasionnellement des requêtes pour détecter le moment où il est de retour en service.   
Les paramètres **max\_fails** et **fail\_timeout** de de la directive **server** contrôlent ce comportement

* La **directive max\_fails** définit le nombre de tentatives infructueuses consécutives de communication avec le serveur qui doivent se produire pendant fail\_timeout.  
  Par défaut, max\_fa**ils** est défini sur 1.   
  Lorsqu'il est défini sur 0, les vérifications d'état sont désactivées pour ce serveur.
* Le **paramètre fail\_timeout** définit également la durée pendant laquelle le serveur sera marqué comme ayant non fonctionnel. Après l'intervalle fail\_timeout suivant la panne du serveur, nginx commencera à sonder normalement le serveur avec les demandes du client en direct.

Si les coups de sondes ont réussis, le serveur est marqué comme un serveur actif.

## Blocage de l'accès direct vers Apache (serveur dorsal)

Étant donné qu'Apache écoute sur le port 8080 sur une adresse IP publique, il est accessible à tous.   
Il peut être bloqué en exécutant la commande iptables suivante dans le jeu de règles de pare-feu.  
**>> sudo iptables -I INPUT -p tcp --dport 8080 ! -s <Adresse IP de nginx> -j REJECT --reject-with tcp-reset**

**Remarques** …  
Dans cette règle, **! -s** inverse la directive, c'est-à-dire que toutes les interfaces spécifiées sont exclues de cette règle.  
Les règles iptables ne survivent pas à un redémarrage du système par défaut.   
Il existe plusieurs façons de conserver les règles iptables, mais la plus simple est de les utiliser iptables-persistent.

1. Un test A/B (également connu sous le nom de test fractionné ou test de compartiment) est une méthode de comparaison de deux versions d'une page Web ou d'une application afin de déterminer celle qui fonctionne le mieux.   
   Le test AB est essentiellement une expérience dans laquelle deux ou plusieurs variantes d'une page sont présentées aux utilisateurs au hasard, et une analyse statistique est utilisée pour déterminer quelle variante fonctionne le mieux pour un objectif de conversion donné. [↑](#footnote-ref-1)