docs.postgresql.fr

20.1. Le fichier pg_hba.conf

23–29 minutes

L'authentification du client est contrôlée par un fichier, traditionnellement nommé `pg_hba.conf` et situé dans le répertoire data du groupe de bases de données, par exemple `/usr/local/pgsql/data/pg_hba.conf` (HBA signifie « host-based authentication » : authentification fondée sur l'hôte.) Un fichier `pg_hba.conf` par défaut est installé lorsque le répertoire data est initialisé par `initdb`. Néanmoins, il est possible de placer le fichier de configuration de l'authentification ailleurs ; voir le paramètre de configuration hba_file.

Le format général du fichier `pg_hba.conf` est un ensemble d'enregistrements, un par ligne. Les lignes vides sont ignorées tout comme n'importe quel texte placé après le caractère de commentaire #. Un enregistrement est constitué d'un certain nombre de champs séparés par des espace et/ou des tabulations. Les enregistrements ne peuvent pas être continués sur plusieurs lignes. Les champs peuvent contenir des espaces si la valeur du champ est mise entre guillemets doubles. Mettre entre guillemets un des mots-clés dans un champ base de données, utilisateur ou adresse (par exemple, `all` ou `replication`) fait que le mot perd son interprétation spéciale, ou correspond à la base de données, à l'utilisateur ou à l'hôte ayant ce nom.

Chaque enregistrement précise un type de connexion, une plage d'adresses IP (si approprié au type de connexion), un nom de base de données, un nom d'utilisateur et la méthode d'authentification à utiliser pour les connexions correspondant à ces paramètres. Le premier enregistrement qui correspond au type de connexion, à l'adresse client, à la base de données demandée et au nom d'utilisateur est utilisé pour effectuer l'authentification. Il n'y a pas de suite après une erreur (« fall-through » ou « backup ») : si un enregistrement est choisi et que l'authentification échoue, les enregistrements suivants ne sont pas considérés. Si aucun enregistrement ne correspond, l'accès est refusé.

Un enregistrement peut avoir l'un des sept formats suivants.

local

database

user

auth-method

[

auth-options

]

host

database

user

address

auth-method

[

auth-options

]

hostssl

database

user

address

auth-method

[

auth-options

]

hostnossl

database

user

address

auth-method

[

auth-options

]

host

database

user

IP-address

IP-mask

auth-method

[

auth-options

]

hostssl

database

user

IP-address

IP-mask

auth-method

[

auth-options

]

hostnossl

database

user

IP-address

IP-mask

auth-method

[

auth-options

]

La signification des champs est la suivante :

`local`

Cet enregistrement intercepte les tentatives de connexion qui utilise les sockets du domaine Unix. Sans enregistrement de ce type, les connexions de sockets du domaine Unix ne sont pas autorisées.

`host`

Cet enregistrement intercepte les tentatives de connexion par TCP/IP. Les lignes `host` s'appliquent à toute tentative de connexion, SSL ou non.

Note

Les connexions TCP/IP ne sont pas autorisées si le serveur n'est pas démarré avec la valeur appropriée du paramètre de configuration listen_addresses. En effet, par défaut, le serveur n'écoute que les connexions TCP/IP en provenance de l'adresse `loopback` locale, `localhost`.

`hostssl`

Cet enregistrement intercepte les seules tentatives de connexions par TCP/IP qui utilisent le chiffrement SSL.

Pour utiliser cette fonction, le serveur doit être compilé avec le support de SSL. De plus, SSL doit être activé en positionnant le paramètre de configuration ssl (voir la Section 18.9 pour plus d'informations). Dans le cas contraire, l'enregistrement `hostssl` est ignoré à l'exception d'une alerte dans les traces indiquant qu'il n'y a aucune connexion correspondante.

`hostnossl`

Cet enregistrement a un comportement opposé à `hostssl` : il n'intercepte que les tentatives de connexion qui n'utilisent pas SSL.

`database`

Indique les noms des bases de données concernées par l'enregistrement. La valeur `all` indique qu'il concerne toutes les bases de données. Le terme `sameuser` indique que l'enregistrement coïncide si la base de données demandée a le même nom que l'utilisateur demandé. Le terme `samerole` indique que l'utilisateur demandé doit être membre du rôle portant le même nom que la base de données demandée (`samegroup` est obsolète bien qu'il soit toujours accepté comme écriture alternative de `samerole`.). Les super-utilisateurs ne sont pas considérés

comme membres d'un rôle dans le cadre de `samerole` à moins qu'ils ne soient explicitement membres du rôle, de manière directe ou indirecte, et non pas juste par ses droits de super-utilisateur. La valeur `replication` indique que l'enregistrement établit une correspondance si une connexion de réplication physique est demandée (notez que les connexions de réplication ne ciblent pas une base de données particulière). Dans tous les autres cas, c'est le nom d'une base de données particulière. Plusieurs noms de base de données peuvent être fournis en les séparant par des virgules. Un fichier contenant des noms de base de données peut être indiqué en faisant précéder le nom du fichier de @.

`user`

Indique les utilisateurs de bases de données auxquels cet enregistrement correspond. La valeur `all` indique qu'il concerne tous les utilisateurs. Dans le cas contraire, il s'agit soit du nom d'un utilisateur spécifique de bases de données ou d'un nom de groupe précédé par un + (il n'existe pas de véritable distinction entre les utilisateurs et les groupes dans PostgreSQL ; un + signifie exactement « établit une correspondance pour tous les rôles faisant parti directement ou indirectement de ce rôle » alors qu'un nom sans + établit une correspondance avec ce rôle spécifique). Ainsi, un super-utilisateur n'est considéré comme membre d'un rôle que s'il est explicitement membre du rôle, directement ou indirectement, et non pas juste par ses droits de super-utilisateur. Plusieurs noms d'utilisateurs peuvent être fournis en les séparant par des virgules. Un fichier contenant des noms d'utilisateurs peut être indiqué en faisant précéder le nom du fichier de @.

address

Indique l'adresse IP ou la plage d'adresses IP à laquelle correspond cet enregistrement. Ce champ peut contenir soit un nom de machine (FQDN), soit le suffixe d'un domaine (sous la forme .exemple.com), soit une adresse ou une plage d'adresses IP, soit enfin l'un des mots-clés mentionnés ci-après.

Une plage d'adresses IP est spécifiée en utilisant la notation numérique standard (adresse de début de plage, suivi d'un slash (`/`) et suivi de la longueur du masque CIDR. La longueur du masque indique le nombre de bits forts pour lesquels une correspondance doit être trouvée avec l'adresse IP du client. Les bits de droite doivent valoir zéro dans l'adresse IP indiquée. Il ne doit y avoir aucune espace entre l'adresse IP, le `/` et la longueur du masque CIDR.

À la place du *`CIDR-address`*, vous pouvez écrire `samehost` pour correspondre aux adresses IP du serveur ou `samenet` pour correspondre à toute adresse du sous-réseau auquel le serveur est directement connecté.

Une plage d'adresses IPv4 spécifiée au format CIDR est typiquement `172.20.143.89/32` pour un hôte seul, `172.20.143.0/24` pour un petit réseau ou `10.6.0.0/16` pour un réseau plus grand. Une plage d'adresses IPv6 spécifiée au format CIDR est par exemple `::1/128` pour un hôte seul (dans ce cas la boucle locale IPv6) ou `fe80::7a31:c1ff:0000:0000/96` pour un petit réseau. `0.0.0.0/0` représente toutes les adresses IPv4, et `::0/0` représente l'ensemble des adresses IPv6. Pour n'indiquer qu'un seul hôte, on utilise une longueur de masque de 32 pour IPv4 ou 128 pour IPv6. Dans une adresse réseau, ne

pas oublier les zéros terminaux.

Une entrée donnée dans le format IPv4 correspondra seulement aux connexions IPv4, et une entrée donnée dans le format IPv6 correspondra seulement aux connexions IPv6, même si l'adresse représentée est dans la plage IPv4-in-IPv6. Notez que les entrées au format IPv6 seront rejetées si la bibliothèque C du système n'a pas de support des adresses IPv6.

La valeur `all` permet de cibler n'importe quelle adresse IP cliente, `samehost` n'importe quelle adresse IP du serveur ou `samenet` pour toute adresse IP faisant partie du même sous-réseau que le serveur.

Si un nom d'hôte est renseigné (dans les faits tout ce qui ne correspond pas à une plage d'adresse ou une plage d'adresses IP, ni à un mot clé, sera traité comme un nom d'hôte), ce nom est comparé au résultat d'une résolution de nom inverse de l'adresse IP du client (ou une recherche DNS inverse si un DNS est utilisé). Les comparaisons de noms d'hôtes ne sont pas sensibles à la casse. En cas de correspondance, une nouvelle recherche récursive de nom sera lancée afin de déterminer que le nom d'hôte concorde bel et bien avec l'adresse IP du client. L'enregistrement n'est validé qu'en cas de concordance entre la résolution inverse et la résolution récursive pour l'adresse IP cliente. (Le nom d'hôte fourni dans le fichier `pg_hba.conf` doit donc correspondre à au moins l'une des adresses IP fournies par le mécanisme de résolution de noms, sinon l'enregistrement ne sera pas pris en considération. Certains serveurs de noms réseau permettent d'associer une adresse IP à de multiples noms d'hôtes (alias DNS), mais

bien souvent le système d'exploitation ne retourne qu'un seul nom d'hôte lors de la résolution d'une adresse IP.)

Un nom d'hôte débutant par un point (`.`) ciblera le suffixe du nom d'hôte du poste client. Du coup, indiquer `.exemple.com` correspondra à la machine `foo.exemple.com` (mais pas au client `exemple.com`).

Lorsque vous spécifiez des noms d'hôtes dans le fichier `pg_hba.conf`, vous devez vous assurer que la résolution de noms soit raisonnablement rapide. À défaut, il peut être avantageux de configurer un serveur-cache local pour effectuer la résolution de noms, tel que `nscd`. Vous pouvez également valider le paramètre de configuration `log_hostname` afin de retrouver dans les journaux le nom d'hôte du client au lieu de sa simple adresse IP.

Ce champ ne concerne que les enregistrements `host`, `hostssl` et `hostnossl`.

Note

Les utilisateurs se demandent parfois pourquoi les noms d'hôte sont gérés de cette manière apparemment si compliquée, avec deux résolutions de nom incluant une résolution inverse de l'adresse IP du client. Cela complique l'utilisation de cette fonctionnalité dans le cas où l'entrée de reverse-DNS n'est pas remplie ou retourne un nom d'hôte indésirable. Cela est fait essentiellement pour raison d'efficacité : de cette manière, une tentative de connexion nécessite au plus deux recherches de résolution, dont une inversée. S'il y a un problème de résolution avec une adresse, cela devient le problème du client. Une alternative d'implémentation hypothétique qui ne ferait pas de

recherche inverse se verrait obligée de résoudre chaque nom d'hôte mentionné dans `pg_hba.conf` à chaque tentative de connexion. Cela serait plutôt lent si de nombreux noms étaient listés. De plus, s'il y a un problème de résolution pour un seul des noms d'hôte, cela devient le problème de tout le monde.

De plus, une résolution inverse est nécessaire pour implémenter la fonctionnalité de correspondance par suffixe dans la mesure où le nom d'hôte du candidat à la connexion doit être connu afin de pouvoir effectuer cette comparaison.

Enfin, cette méthode est couramment adoptée par d'autres implémentations du contrôle d'accès basé sur les noms d'hôtes, tels que le serveur web Apache ou TCP-wrapper.

`IP-address`

`IP-mask`

Ces champs peuvent être utilisés comme alternative à la notation *`adresse IP/longueur masque`*. Au lieu de spécifier la longueur du masque, le masque réel est indiquée dans une colonne distincte. Par exemple, `255.0.0.0` représente une longueur de masque CIDR IPv4 de 8, et `255.255.255.255` représente une longueur de masque de 32.

Ces champs ne concernent que les enregistrements `host`, `hostssl` et `hostnossl`.

`auth-method`

Indique la méthode d'authentification à utiliser lors d'une connexion via cet enregistrement. Les choix possibles sont résumés ici ; les détails se trouvent dans la Section 20.3.

Toutes les options sont en minuscules et traitées avec une sensibilité à la casse, donc même les acronymes comme `ldap` doivent être écrits en minuscule.

`trust`

Autorise la connexion sans condition. Cette méthode permet à quiconque peut se connecter au serveur de bases de données de s'enregistrer sous n'importe quel utilisateur PostgreSQL de son choix sans mot de passe ou autre authentification. Voir la Section 20.4 pour les détails.

`reject`

Rejette la connexion sans condition. Ce cas est utile pour « filtrer » certains hôtes d'un groupe, par exemple une ligne `reject` peut bloquer la connexion d'un hôte spécifique alors qu'une ligne plus bas permettra aux autres hôtes de se connecter à partir d'un réseau spécifique.

`scram-sha-256`

Réalise une authentification SCRAM-SHA-256 afin de vérifier le mot de passe utilisateur. Voir Section 20.5 pour les détails.

`md5`

Réalise une authentification SCRAM-SHA-256 ou MD5 afin de vérifier le mot de passe utilisateur. Voir Section 20.5 pour les détails.

`password`

Requiert que le client fournisse un mot de passe non chiffré pour l'authentification. Comme le mot de passe est envoyé en clair sur le réseau, ceci ne doit pas être utilisé sur des réseaux non dignes de confiance. Voir la

Section 20.5 pour les détails.

`gss`

Utilise GSSAPI pour authentifier l'utilisateur. Disponible uniquement pour les connexions TCP/IP. Voir Section 20.6 pour les détails.

`sspi`

Utilise SSPI pour authentifier l'utilisateur. Disponible uniquement sur Windows. Voir Section 20.7 pour plus de détails.

`ident`

Récupère le nom de l'utilisateur en contactant le serveur d'identification sur le poste client, et vérifie que cela correspond au nom d'utilisateur de base de données demandé. L'authentification Ident ne peut être utilisée que pour les connexions TCP/IP. Pour les connexions locales, elle sera remplacée par l'authentification peer.

`peer`

Récupère le nom d'utilisateur identifié par le système d'exploitation du client et vérifie que cela correspond au nom d'utilisateur de base de données demandé. Peer ne peut être utilisée que pour les connexions locales. Voir la Section 20.9 ci-dessous pour les details.

`ldap`

Authentification par un serveur LDAP. Voir la Section 20.10 pour les détails.

`radius`

Authentification par un serveur RADIUS. Voir Section 20.11 pour les détails.

`cert`

Authentification par certificat client SSL. Voir Section 20.12 pour les détails.

`pam`

Authentification par les Pluggable Authentification Modules (PAM) fournis par le système d'exploitation. Voir la Section 20.13 pour les détails.

`bsd`

Authentification utilisant le service BSD Authentication fourni par le système d'exploitation. Voir Section 20.14 pour plus de détails.

`auth-options`

Après le champ *`auth-method`*, on peut trouver des champs de la forme *`nom`*`=`*`valeur`* qui spécifient des options pour la méthode d'authentification. Les détails sur les options disponibles apparaissent ci-dessous pour chaque méthode d'authentification.

En plus des options spécifiques à une méthode listées ci-dessous, il existe une option d'authentification indépendante de la méthode, appelée `clientcert`, qui peut être indiquée dans tout enregistrement `hostssl`. Une fois configurée à `1`, cette option requiert le client à présenter un certificat SSL valide (de confiance), en plus des autres nécessités de la méthode d'authentification.

Les fichiers inclus par les constructions @ sont lus comme des listes de noms, séparés soit par des espaces soit par des virgules. Les commentaires sont introduits par le caractère # comme dans `pg_hba.conf`, et les constructions @ imbriquées sont autorisées. À moins que le nom du fichier qui suit @ ne soit

un chemin absolu, il est supposé relatif au répertoire contenant le fichier le référençant.

Les enregistrements du fichier `pg_hba.conf` sont examinés séquentiellement à chaque tentative de connexion, l'ordre des enregistrements est donc significatif. Généralement, les premiers enregistrements ont des paramètres d'interception de connexions stricts et des méthodes d'authentification peu restrictives tandis que les enregistrements suivants ont des paramètres plus larges et des méthodes d'authentification plus fortes. Par exemple, on peut souhaiter utiliser l'authentification `trust` pour les connexions TCP/IP locales mais demander un mot de passe pour les connexion TCP/IP distantes. Dans ce cas, l'enregistrement précisant une authentification `trust` pour les connexions issues de 127.0.0.1 apparaît avant un enregistrement indiquant une authentification par mot de passe pour une plage plus étendue d'adresses IP client autorisées.

Le fichier `pg_hba.conf` est lu au démarrage et lorsque le processus serveur principal reçoit un signal SIGHUP. Si le fichier est édité sur un système actif, on peut signaler au postmaster (en utilisant `pg_ctl reload`, en appelant la fonction SQL `pg_reload_conf()`, ou `kill -HUP`) de relire le fichier.

Note

L'information précédente n'est pas vraie sous Microsoft Windows : ici, tout changement dans le fichier `pg_hba.conf` est immédiatement appliqué à toute nouvelle connexion.

La vue système `pg_hba_file_rules` peut aider pour pré-tester les changements dans le fichier `pg_hba.conf`, ou pour diagnostiquer des problèmes si le rechargement du fichier n'a

pas eu les effets escomptés. Les lignes dans la vue avec des champs `error` non vides indiquent des problèmes dans les lignes correspondantes du fichier.

Astuce

Pour se connecter à une base particulière, un utilisateur doit non seulement passer les vérifications de `pg_hba.conf` mais doit également avoir le droit `CONNECT` sur cette base. Pour contrôler qui peut se connecter à quelles bases, il est en général plus facile de le faire en donnant ou retirant le privilège `CONNECT` plutôt qu'en plaçant des règles dans le fichier `pg_hba.conf`.

Quelques exemples d'entrées de `pg_hba.conf` sont donnés ci-dessous dans l'Exemple 20.1. Voir la section suivante pour les détails des méthodes d'authentification.

Exemple 20.1. Exemple d'entrées de `pg_hba.conf`

```
# Permettre à n'importe quel utilisateur du système local de se connecter
# à la base de données sous n'importe quel nom d'utilisateur au travers
# des sockets de domaine Unix (par défaut pour les connexions locales).
#
# TYPE  DATABASE        USER            ADDRESS                 METHOD
local   all             all                                     trust

# La même chose en utilisant les connexions TCP/IP locales loopback.
#
# TYPE  DATABASE        USER            ADDRESS
```

```
METHOD
host    all             all             127.0.0.1/32            trust

# Pareil mais en utilisant une colonne netmask distincte.
#
# TYPE  DATABASE    USER        IP-ADDRESS        IP-mask
METHOD
host    all         all         127.0.0.1         255.255.255.255     trust

# Pareil mais en IPv6.
#
# TYPE  DATABASE        USER            ADDRESS
METHOD
host    all             all             ::1/128                 trust

# À l'identique en utilisant le nom d'hôte (qui doit typiquement
fonctionner en IPv4 et IPv6).
#
# TYPE  DATABASE        USER            ADDRESS
METHOD
host    all             all             localhost               trust

# Permettre à n'importe quel utilisateur de n'importe quel hôte
d'adresse IP
# 192.168.93.x de se connecter à la base de données
"postgres" sous le nom
# d'utilisateur qu'ident signale à la connexion (généralement le
# nom utilisateur du système d'exploitation).
#
# TYPE  DATABASE        USER            ADDRESS
METHOD
host    postgres        all             192.168.93.0/24         ident
```

```
# Permet à un utilisateur de l'hôte 192.168.12.10 de se
connecter à la base de
# données "postgres" si le mot de passe de l'utilisateur est
correctement fourni.
#
# TYPE  DATABASE        USER            ADDRESS
METHOD
host    postgres        all             192.168.12.10/32        scram-
sha-256

# Permet la connexion à n'importe quel utilisateur depuis toutes
les machines du
# domaine exemple.com à n'importe quelle base de données si
le mot de passe
# correct est fourni.
#
# Require SCRAM authentication for most users, but make an
exception
# for user 'mike', who uses an older client that doesn't support
SCRAM
# authentication.
#
# TYPE  DATABASE        USER            ADDRESS
METHOD
host    all             mike            .example.com            md5
host    all             all             .example.com            scram-sha-256

# Si aucune ligne "host" ne précède, ces deux lignes rejettent
toutes
# les connexions en provenance de 192.168.54.1 (puisque cette
entrée déclenche
```

```
# en premier), mais autorisent les connexions GSSAPI de
n'importe où
# ailleurs sur l'Internet. Le masque zéro signifie qu'aucun bit de
l'ip de
# l'hôte n'est considéré, de sorte à correspondre à tous les
hôtes.
#
# TYPE  DATABASE        USER            ADDRESS
METHOD
host    all             all             192.168.54.1/32         reject
host    all             all             0.0.0.0/0               gss

# Permettre à tous les utilisateurs de se connecter depuis
192.168.x.x à n'importe
# quelle base de données s'ils passent la vérification
d'identification. Si,
# par exemple, ident indique que l'utilisateur est "bryanh" et qu'il
# demande à se connecter en tant qu'utilisateur PostgreSQL
"guest1", la
# connexion n'est permise que s'il existe une entrée dans
pg_ident.conf pour la
# correspondance "omicron" disant que "bryanh" est autorisé à
se connecter en
# tant que "guest1".
#
# TYPE  DATABASE        USER            ADDRESS
METHOD
host    all             all             192.168.0.0/16          ident
map=omicron

# Si ces trois lignes traitent seules les connexions locales, elles
# n'autorisent les utilisateurs locaux qu'à se connecter à leur
```

```
propre
# base de données (base ayant le même nom que leur nom
# d'utilisateur) exception faite des administrateurs
# et des membres du rôle "support" qui peuvent se connecter à
toutes les bases
# de données. Le fichier $PGDATA/admins contient une liste de
noms
# d'administrateurs. Un mot de passe est requis dans tous les
cas.
#
# TYPE  DATABASE        USER            ADDRESS
METHOD
local   sameuser        all                             md5
local   all             @admins                         md5
local   all             +support                        md5

# Les deux dernières lignes ci-dessus peuvent être combinées
en une seule ligne :
local   all         @admins,+support                md5

# La colonne database peut aussi utiliser des listes et des noms
de fichiers :
local   db1,db2,@demodbs  all                       md5
```