Teil 1: Einen PostgreSQL-Cluster installieren und konfigurieren

Aller Anfang ist leicht ...

Es ist ein Mythos, dass der Einsatz der PostgreSQL-Datenbank nur Advanced Users vorbehalten ist. Die Installation und Konfiguration eines PostgreSQL-Clusters ist kein Buch mit sieben Siegeln. Überzeugen Sie sich selbst von dieser Tatsache. In Teil 1 dieser Artikelserie werden die Installation und grundlegende Konfiguration behandelt.

▶ von Andreas Wenk

Lassen Sie uns zuerst einmal mit einem Vorurteil aufräumen. Nein, es ist nicht kompliziert, einen PostgreSQL-Datenbankcluster zu installieren bzw. aufzusetzen. Weder auf Windows, Mac OS X noch auf Unix-artigen Systemen wie Linux. Ja, es erfordert im Gegensatz zu einer MySQL-Installation etwas mehr Aufwand, den Cluster zum Laufen zu bringen. Aber auch das ist kein Hexenwerk und lässt sich mit ein wenig Motivation und Interesse realisieren. In diesem Artikel erhalten Sie das grundlegende Verständnis, wie das geht. Dass unter den Datenbanken die PostgreSQL nicht genauso verbreitet ist wie die MySQL, hängt, so denke ich, mit der fehlenden Verfügbarkeit bei Hosting-Anbietern zusammen. Daraus ergibt sich dann wiederum die fehlende Unterstützung von PostgreSQL in vielen beliebten Projekten wie z. B. XAMPP (was dann XAPPP heißen müsste), TYPO3, Joomla! oder WordPress. Aus meiner Sicht ist

dies sehr schade, denn PostgreSQL bietet einen sehr großen Leistungsumfang und ist definitiv „ein bisschen mehr Datenbank" als MySQL. Letztlich liegt die Wahl der Datenbank aber immer in den Händen des Benutzers – also in Ihren. Genug der Datenbankphilosophie … lassen Sie uns ans Werk gehen und sehen, wie wir das Ding auf den Server bekommen.

Installation der PostgreSQL

Die Installation der PostgreSQL-Datenbank [1] kann auf unterschiedlichen Wegen erfolgen. Im Folgenden zeige ich Ihnen, wie die Installation mit einem Paketmanager (Linux, Mac OS X), per One-click Installer (Linux, Windows, Mac OS X) und aus den Sourcen zum Ziel führt. Die übliche Einschränkung auf Grund der mannigfaltigen Linux-Distributionen muss leider auch hier wieder getroffen werden: Die Installation aus den Sourcen und mit dem Paketmanager zeige ich exemplarisch auf einem Debian-basierten Linux-System (z. B. Debian Linux, Ubuntu, Kubuntu). Natürlich kann die PostgreSQL aber auch genauso problemlos mit den Paketmanagern von Suse, RedHat, FreeBSD oder anderen Linux-Distributionen ausgeführt werden.

Installation mit dem Paketmanager

Jeder Sysadmin wird Ihnen dringend empfehlen, Programme immer mit dem der Distribution zugehörigen Paketmanager zu installieren. Unter Debian-basierten Linux-Systemen gibt es *aptitude* (empfohlen) oder den Vorgänger *apt* (um korrekt zu sein: Diese beiden Programme sind ein Frontend bzw. Interface für den eigentlichen Paketmanager *dpkg*). Das hat einen großen Vorteil: Bugfixes oder Updates spielen Sie durch den simplen Befehl *aptitude update* und *aptitude upgrade* ein. Außerdem werden die Programme und deren Dateien an die Orte im System abgelegt, die in das Konzept der Distribution passen. Für jedes dieser Pakete gibt es bei Debian einen so genannten *Maintainer* (zumindest meistens), der sich um die Betreuung des Pakets kümmert. Geben Sie einfach mal folgenden *aptitude*-Befehl ein: *$ aptitude show postgresql-8.4*. Dann sehen Sie z. B., dass Martin Pitt der Maintainer dieses Pakets ist. Nun aber zur Installation der PostgreSQL.

Das gestaltet sich ziemlich einfach. Suchen Sie zuerst mit *aptitude* nach den PostgreSQL-Paketen: *$ aptitude search ^postgresql-*. Mit dem Befehl finden Sie alle Pakete, deren Name mit *postgresql* beginnt und von einem – Zeichen gefolgt wird. Sie sehen jetzt diverse Pakete für die PostgreSQL in den Versionen 8.3 und 8.4. Von Interesse ist hier das Paket *postgresql-8.4*. Interessant sind außerdem die Pakete *postgresql-contrib-8.4* mit Erweiterungen der PostgreSQL, *postgresql-doc-8.4* für die Dokumentation der PostgreSQL und Pakete wie *postgresql-plperl-8.4* oder *postgresql-plpython-8.4*, die die Datenbank um eine prozedurale Sprache erweitern. Die Installation erledigen Sie dann mit folgendem Befehl: *$ aptitude install postgresql-8.4 [und weitere von Ihnen gewünschte Pakete]*. Wenn Sie zuerst einmal sehen wollen, was genau installiert würde, nutzen Sie die Option *--simulate*. Ist alles o.k., lassen Sie *--simulate* weg und installieren die Software endgültig. Und das war's auch schon.

Diese Installationsart nimmt Ihnen viel Arbeit ab (wie Sie später noch im Abschnitt „Die Sourcen selbst übersetzen" sehen werden). Denn es wurde bereits ein Systembenutzer *postgres*, das Verzeichnis */etc/postgresql/8.4/main/*, in dem die Konfigurationsdateien liegen, und das Verzeichnis */var/lib/postgresql/8.4/main/*, in dem alle zum Cluster gehörigen Dateien zu finden sind, angelegt. Außerdem sei noch erwähnt, dass Sie unter */usr/share/postgresql/8.4/* ebenfalls ein zur PostgreSQL gehöriges Verzeichnis finden. Dort finden Sie einige Beispieldateien und Unterstützung für die Volltextsuche (*tsearch_data*). Erwähnenswert ist schließlich noch das Verzeichnis */usr/lib/postgresql/8.4/*, in welchem im Unterverzeichnis */bin/* alle zur PostgreSQL gehörenden Programme zu finden sind. Ganz wichtig zum Thema „Arbeit sparen" ist auch, dass bei der Installation per *aptitude* alle Abhängigkeiten zu anderen Paketen mit installiert bzw. aufgelöst werden.

Sehr angenehm ist auch die Tatsache, dass Ihnen in der Debian-Distribution einige zusätzliche *pg_*-Programme zur Verfügung stehen. Sehen wir uns kurz an, welche davon standardmäßig im PATH vorhanden sind (alle übrigen finden Sie im oben erwähnten Verzeichnis */usr/lib/postgresql/8.4/bin/*):

```
$ pg_ [TAB][TAB]
$ pg_createcluster pg_ctlcluster pg_dropcluster pg_dump
    pg_dumpall pg_lsclusters pg_restore pg_updatedicts
                                        pg_upgradecluster
```

Da es hier ausschließlich um die Installation von PostgreSQL geht, sehen wir uns nur kurz *pg_ctlcluster* an. Das ist das Debian-Pendant zu *pg_ctl* (sehen Sie die Verwendung im Abschnitt „Die Sourcen selbst übersetzen" weiter unten). Mit diesem Programm starten Sie einen PostgreSQL-Cluster: *$ pg_ctlcluster 8.4 main start*. Und analog, um ihn zu stoppen: *$ pg_ctlcluster 8.4 main stop*. Sie können natürlich auch die Variante über */etc/init.d/* wählen: *$ /etc/init.d/postgresql-8.4 start* und *$ /etc/init.d/postgresql-8.4 stop*. Anhand des Programms *pg_ctlcluster* und des Aufrufs der PostgreSQL-Versionsnummer können Sie erkennen, dass es unter Debian möglich ist, PostgreSQL in unterschiedlichen Versionen parallel (natürlich auf unterschiedlichen Ports) laufen zu lassen. Gerade bei der Migration zu einer neueren Version ist der Parallelbe-

Installation auf Mac OS X mit MacPorts

Freunde des Mac (so wie ich) kennen natürlich den Paketmanager *MacPorts* [6] und haben ihn mit aller größter Wahrscheinlichkeit installiert. Mit einer Suche in den *MacPorts* über das *Command Line*-Programm finden Sie die PostgreSQL in der Version 8.4:

```
$ port search postgresql
[…]
postgresql84 @8.4.1 (databases)
  The most advanced open-source database available
                                        anywhere.
[…]
```

Die Installation starten Sie dann einfach durch den Befehl *$ port install postgresql84*.

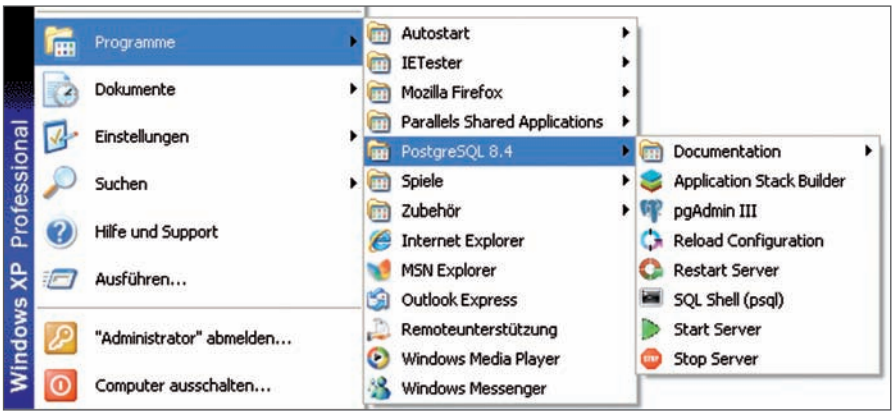

Abb. 1: Menüeinträge von PostgreSQL nach der Installation mit dem One-click Installer

trieb oft sehr hilfreich. Und damit haben wir auch schon alles zur Installation mit dem Paketmanager gesagt.

Installation mit dem One-click Installer

Die Firma EnterpriseDB [3] bietet professionelle Services rund um die PostgreSQL-Datenbank. Bruce Momjian und Dave Page gehören dieser Firma an und sind beide Entwickler im Core-Team der PostgreSQL-Datenbank. EnterpriseDB hat ein plattformunabhängiges Installations-Bundle erstellt. Damit lässt sich PostgreSQL mit einem interaktiven Installer auf Windows, Mac OS X und Linux (32 und 64 Bit) komfortabel installieren. Mitgeliefert wird außerdem das PostgreSQL-Datenbankverwaltungstool *pgAdmin III* [4] und der *StackBuilder Package Manager*, der es Ihnen erlaubt, weitere Programme und Erweiterungen für PostgreSQL zu installieren.

Laden Sie also von der EnterpriseDB-Downloadseite [5] den One-click Installer herunter. Im Folgenden zeige ich Ihnen beispielhaft den Installationsprozess unter Windows. Nach dem Download extrahieren Sie den Inhalt des zip-Archivs in einen beliebigen Ordner und wechseln in das extrahierte Verzeichnis. Um die Installation nun zu starten, doppelklicken Sie die Installer-Datei *postgresql-8.4.1-1-windows.exe*. Während der Installationsroutine werden Sie aufgefordert, verschiedene Informationen anzugeben. Standardmäßig wird auch hier ein Systembenutzer *postgres* angelegt, dem Sie noch ein Passwort geben müssen. Ansonsten ist die Installation trivial.

Wie auf Windows-Systemen üblich, wird im Startmenü unter Programme ein Ordner zur Installation untergebracht. Dort finden Sie alle Programme, die Sie für das Starten, Stoppen und Verwalten der PostgreSQL benötigen. In Abbildung 1 sehen Sie im rechten Fenster übrigens auch einen Eintrag für *pgAdmin III*. Sehr schön – wie oben erwähnt, bekommen Sie das Programm gleich frei Haus mitgeliefert. Am häufigsten werden Sie wohl die Einträge Dienst starten und Dienst beenden nutzen, außerdem natürlich die beiden PostgreSQL-Verwaltungstools *pgAdmin III* und *psql nach 'postgres'* (was nicht mehr bedeutet als „*rufe psql als Benutzer postgres auf*"). Damit haben wir auch schon die Installation mit dem One-click Installer besprochen. Beachten Sie bitte, dass die Installation auf dem Mac ein klein wenig abweichend ist. Lesen Sie dazu einfach die Hinweise unter [5].

Die Sourcen selbst übersetzen (kompilieren)

Dies ist wohl die rudimentärste, aber auch kniffligste Installation der PostgreSQL-Datenbank. Deshalb sollten Sie das nur tun, wenn Sie Erfahrung auf diesem Gebiet haben und es einen guten Grund dafür gibt. Das wäre zum Beispiel der Fall, wenn das verfügbare Paket von PostgreSQL in der jeweiligen Distribution nicht über die erforderlichen Features wie z. B. SSL-Unterstützung verfügt, oder wenn die neueste Version von PostgreSQL noch nicht über den jeweiligen Paketmanager installiert werden kann. Im Folgenden gehe ich davon aus, dass Sie sich auf der Linux-Kommandozeile befinden und root-Rechte besitzen.

Hinweise zur Windows-Kommandozeile

Natürlich können Sie die Programme, die mit PostgreSQL mitgeliefert wurden, auch auf der Windows-Kommandozeile ausführen. Nutzen Sie dazu einfach die Verknüpfung Befehlszeile im Startmenü (Abb. 1). Sie gelangen beim Öffnen des Eingabefensters sofort in das *bin*-Verzeichnis der PostgreSQL-Installation. Geben Sie dort den Befehl dir ein, sehen Sie alle vorhandenen PostgreSQL-Programme.

Optionen für das configure-Skript

Sie haben die Möglichkeit, auf das Erstellen der Makefiles, und damit schließlich auf die Installation an sich, durch das Angeben von Optionen Einfluss zu nehmen. Denkbar sind zum Beispiel folgende Optionen:

```
./configure --prefix=/usr/local/pgsql --with-tcl --
    with-perl --with-python --with-krb5 --with-pam --
        with-openssl --with- libxslt –with-gssapi
```

Dabei geben Sie mit der Option *--with-prefix=* das Verzeichnis an, in dem PostgreSQL installiert werden soll. Mit den Optionen *--with-tcl*, *--with-perl* und *--with-python* geben Sie an, dass die Sprachen PL/Tcl, PL/Perl und PL/Python als prozedurale Sprachen für *user defined functions* zur Verfügung stehen sollen. Weiterhin dienen *--with-krb5* für die Kerberos-, *--with-pam* für die PAM- und *--with-gssapi* für die GSSAPI-Authentifizierungen. *--with- openssl* bietet die Unterstützung für SSL-verschlüsselte Verbindungen und *--with-libxslt* dient der Möglichkeit, unter Nutzung der contrib/xml2-Library XSL-Transformationen von XML durchzuführen. Sie können die Installation natürlich auch ohne Nutzung der Optionen problemlos durchführen, haben dadurch aber einen geringeren Funktionsumfang.

MAYFLOWER

THE PHP EXPERTS

AGILITÄT & EXZELLENZ

www.mayflower.de

Im ersten Schritt müssen Sie die Quellen von [2] herunterladen. Nachdem Sie das Archiv heruntergeladen haben, entpacken Sie dieses in einem Ordner Ihrer Wahl (ich habe das mit *gzip* komprimierte *tar*-Archiv gewählt – es liegt auch ein mit *bzip2* komprimiertes *tar*-Archiv vor): *$ tar xfz postgresql-8.4.1.tar.gz*. Es wird ein Ordner */postgresql-8.4.1/* erstellt, in den Sie jetzt wechseln. Innerhalb des Ordners finden Sie ein *configure*-Skript, das Sie folgendermaßen ausführen können – lesen Sie aber bitte zuerst weiter: *$ ./configure*. Das *configure*-Skript ist ein Teil von *automake* bzw. *autoconf*, das dazu dient, das Makefile für die Installation der PostgreSQL-Datenbank zu erstellen. Die Ausführung des *configure*-Skripts ist notwendig, um herauszufinden, in welcher Umgebung bzw. auf welchem System die PostgreSQL-Datenbank installiert werden soll, ob alle notwendigen anderen Programme und Bibliotheken für die Installation vorhanden sind und ob die Installation überhaupt möglich ist. Der Aufruf von *./configure* ist der denkbar einfachste. Lesen Sie im Kasten „Optionen für das configure-Skript", welche Optionen Sie angeben sollten, um einen ausreichend großen Funktionsumfang der PostgreSQL zu erhalten.

Das *configure*-Skript gibt Fehlermeldungen zurück, wenn es zu Problemen kommt. Sie müssen diese Probleme beheben, um fortfahren zu können. Sehr wahrscheinlich werden Sie bei der Angabe der im Kasten angegebenen Optionen für *configure* nicht in der glücklichen Lage sein, dass alles glatt geht. Mit großer Wahrscheinlichkeit erhalten Sie eine Fehlermeldung, dass zum einen das Paket *readline* und zum anderen das Paket *zlib* fehlt. Suchen Sie mit dem Paketmanager *aptitude* nach den Paketen und installieren Sie sie:

```
$ aptitude search readline-dev ...
                    libreadline5-dev - GNU readline and history \
libraries, development files ...
$ aptitude search zlib-dev libghc6-zlib-dev -
                              Compression and decompression in \
the gzip and zlib formats
$ apt-get install libreadline5-dev libghc6-zlib-dev
```

Dann werden Sie ebenso wahrscheinlich auf Probleme mit Abhängigkeiten von PostgreSQL zu anderen Paketen stoßen. Hier hilft ein kleiner Trick. Sie können davon ausgehen, dass die Version 8.3 von PostgreSQL die gleichen Abhängigkeiten hat wie die 8.4. Deshalb installieren Sie die abhängigen Pakete:

```
$ aptitude build-dep postgresql-8.3
[...]
```

Die folgenden *neuen* Pakete werden installiert: *cdbs*, *fdupes*, *hardening-wrapper*, *intltool*, *libossp-uuid-dev*, *libossp-uuid15*, *libpam0g-dev*, *libperl-dev*, *libxslt1-dev*, *python-dev*, *python2.6-dev tcl8.5-dev*. So weit, so gut. Aber *configure* wird sich nochmals beschweren, und zwar bzgl. *libxml* und *tcl* – *configure* kann den Ort der C-Header-Dateien nicht finden. Na gut – sagen wir's *configure*: *--with-tclconfig=/usr/lib/tcl8.5 --with-includes=/usr/include/libxml2*. Der gesamte *configure*-Befehl inklusive Optionen sieht dann folgendermaßen aus:

```
$ ./configure --prefix=/usr/local/pgsql
                          --with-tcl –with-perl –with-python \
--with-krb5 --with-pam --with-openssl –with-libxslt
                                                --with-gssapi \
--with-tclconfig=/usr/lib/tcl8.5 –with-includes=/usr/
                                                include/libxml2
```

Im nächsten Schritt führen Sie dann das Programm *make* aus, das nun auf Grundlage des Makefile die PostgreSQL-Datenbank kompiliert: *$ make*. Das Kompilieren nimmt einige Zeit in Anspruch (Tipp: nächsten Kaffee holen ...). Nach Beenden des Programms finden Sie im Verzeichnis */src/* die für Ihr System kompilierten Dateien und Programme. Um PostgreSQL nun endgültig zu installieren, führen Sie abschließend das Programm *make* mit dem Parameter *install* aus und beachten dabei, dass Sie je nach angegebenem Ort mit der Option *--with-prefix=* dafür *root*-Rechte brauchen. Dies ist der eigentliche Installationsprozess, denn nun wird das Programm an die von den Entwicklern vorgesehenen Stellen im System kopiert: *$ make install*. Et voilà. Die PostgreSQL-Datenbank ist auf Ihrem System installiert. Übrigens sollten Sie unbedingt einen Blick in die Dateien *INSTALL* und *README* werfen. Sie erhalten dort wichtige Informationen und Hilfe zu üblichen Installationsproblemen. Insbesondere erfahren Sie in der Datei *INSTALL*, welche Abhängigkeiten samt Header-Dateien installiert werden müssen.

Um nun prinzipiell mit PostgreSQL arbeiten zu können, müssen Sie noch ein paar Tasks abarbeiten. Zum einen wird ein Benutzer *postgres* benötigt:

```
$ useradd -m postgres
[...]
```

Dann benötigt PostgreSQL ein Datenverzeichnis. In diesem Verzeichnis werden alle Datenbanken und deren Datenbankobjekte unseres Clusters wie Tabellen, Sequenzen usw. abgelegt:

```
$ mkdir /usr/local/pgsql/data
$ chown postgres /usr/local/pgsql/data
```

Jetzt wechseln Sie zum Systembenutzer *postgres* und springen in dessen Heimverzeichnis: *$ su – postgres*. Schließlich richten wir den PostgreSQL-Cluster ein, indem wir das Programm *initdb* aufrufen und das eben erstellte Verzeichnis als Datenverzeichnis initialisieren: *postgres$ initdb /usr/local/pgsql/data*. Jetzt fehlt uns nur noch die Möglichkeit, den

Tabelle 1: **Orte, an denen die pg_hba.conf und postgresql.conf zu finden ist**

OS	Installationsart	Pfad
Linux	Paketmanager	/etc/postgresql/8.4/main
Linux	Sourcen	/usr/local/pgsql/ (wenn so angegeben als --prefix=)
Linux	One-click Installer	/var/lib/postgresql/8.4/data (oder wie angegeben)
Windows	One-click Installer	C:\Program Files\PostgreSQL\8.4 (oder wie angegeben)
Mac OS X	One-click Installer	/Library/PostgreSQL/8.4/data (oder wie angegeben)
Mac OS X	MacPorts	/opt/local/var/db/postgresql84/main

Cluster zu starten. Dafür nutzen wir als Systembenutzer *postgres* das Programm *pg_ctl* und geben ihm mit dem Parameter *-D* den Pfad zum Datenverzeichnis und die Option *start* mit: *postgres$ /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data start*. Um den Cluster zu stoppen, geben wir *pg_ctl* die Option *stop* mit: *postgres$ /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data stop*. Gratulation! Wenn Sie hier angelangt sind und alles geklappt hat, haben Sie PostgreSQL erfolgreich kompiliert und zum Laufen gebracht. Damit haben Sie alle unterschiedlichen Installationsarten kennen gelernt und können nun im nächsten Abschnitt sehen, wie der PostgreSQL-Cluster konfiguriert wird.

Kaffeepause

So, dann haben wir den ersten Schritt erledigt. Sie haben nun auf Ihrem Lieblingssystem auf die von Ihnen favorisierte Art und Weise einen PostgreSQL-Cluster in der Version 8.4.1 installiert. Zumindest die Installation per Paketmanager oder per One-click Installer war doch wirklich nicht schwer. Sehen wir uns nun im nächsten Abschnitt an, was Sie tun müssen, damit Sie mit Ihrem neuen Baby sprechen können. Aber zuerst sollten Sie sich den nächsten, wohlverdienten Kaffee holen.

Konfiguration der PostgreSQL

Es gibt prinzipiell zwei Dateien, die dazu dienen, die wichtigsten Konfigurationseinstellungen vorzunehmen. Das ist zum einen die Datei *postgresql.conf* und zum anderen die Datei *pg_hba.conf*. In Tabelle 1 finden Sie eine Übersicht, wo Sie diese, je nach Betriebssystem und Installationsart, finden.

Die Datei pg_hba.conf

In der Datei *pg_hba.conf* wird der Zugriff auf den PostgreSQL-Cluster geregelt. *pg_hba* heißt ausgeschrieben *PostgreSQL Hostbased Authentication*. Wenn zu PostgreSQL eine Verbindung hergestellt wird, geschieht dies über zwei unterschiedliche Wege. Zum einen über eine TCP/IP-Verbindung oder über einen lokalen Unix-Domain-Socket (Kasten: „TCP/IP und Unix-Domain-Sockets“). Ob diese Verbindungen zugelassen werden oder nicht, wird in der *pg_hba.conf* festgelegt. Lassen Sie uns zuerst einmal eine *pg_hba.conf* ansehen:

```
1 # Database administrative login by UNIX sockets
2 local     all                 postgres                        ident
3 # TYPE    DATABASE            USER        CIDR-ADDRESS        METHOD
4 # "local" is for Unix domain socket connections only
5 #local    all                 all                             ident
6 # IPv4 local connections:
7 host      all                 all         127.0.0.1/32        password
8 host      all                 all         216.104.34.90/32    md5
9 # IPv6 local connections:
10 host     all                 all         ::1/128             md5
```

Die Datei ist in fünf Spalten aufgeteilt: *TYPE* gibt die Art der Verbindung an, *DATABASE* die betreffende Datenbank, durch ein Komma separierte einzelne Datenbanken oder alle Datenbanken, *USER* den betreffenden Datenbankbenutzer, durch ein Komma separierte Datenbankbenutzer oder alle Datenbankbenutzer, *CIDR-ADDRESS* die IP-Adresse, eine IP-Adressen-Range oder keine Angabe bei *local*-Verbindungen, über welche Adresse eine Verbindung zugelassen wird und schließlich *METHOD*, die Methode der Authentifizierung. Alle Unix-Domain-Socket-Verbindungen werden durch *local* und alle TCP/IP-Verbindungen durch das einleitende *host* gekennzeichnet. Wie Sie sehen können, wird in Zeile 2 zuallererst sichergestellt, dass *local*-Anfragen durch den Systembenutzer *postgres* akzeptiert werden. Denn *ident* bedeutet, dass die Verbindung nur zugelassen wird, wenn es auch den gleichen Systembenutzer gibt, wie den Benutzer, der gerade versucht, mit PostgreSQL zu kommunizieren. Diese Zeile sollten Sie nie entfernen, denn sonst können Hintergrundjobs von PostgreSQL, ausgeführt durch den Benutzer *postgres*, nicht erledigt werden.

Zeile 5 ist auskommentiert und bezieht sich ebenfalls auf Unix-Domain-Socket-Verbindungen. Sie steht hier nur um zu zeigen, dass Sie natürlich mehrere unterschiedliche Definitionen angeben können. Beachten Sie in diesem Zusammenhang, dass die Reihenfolge wichtig ist. Der erste Eintrag, der zutrifft, wird genutzt – nachfolgende Einträge werden nicht weiter berücksichtigt.

Damit Sie nun z. B. mit *psql*, dem mitgelieferten und sehr mächtigen *Command Line Interface* der PostgreSQL, eine Verbindung zum Cluster herstellen können, müssen Sie dies als Systembenutzer *postgres* tun. Der Eintrag in Zeile 2 würde greifen. Wenn Sie z. B. eine Webanwendung erstellt haben und auf das lokale PostgreSQL zugreifen wollen (localhost), müssen Sie einen Eintrag wie in Zeile 7 erstellen. Und wenn der Zugriff schließlich über einen entfernten Server erfolgt, sprich, Sie Ihre Applikation sinnvoller Weise in Webserver und Datenbankserver aufgetrennt haben, müssen Sie einen Eintrag wie in Zeile 8 mit der entsprechenden IP-Adresse angeben.

An dieser Stelle noch ein Satz zu der METHOD *md5* im Vergleich zu *password*. *md5*, was bedeutet, dass PostgreSQL ein MD5-gehashtes Passwort erwartet. Das heißt, dass Sie sicherstel-

TCP/IP und Unix-Domain-Sockets

TCP/IP ist ein Netzwerkprotokoll und heißt ausgeschrieben *Transmission Control Protocol/Internet Protocol*. Dieses Protokoll beschreibt einen Weg, wie Datenströme (TCP) und Datenpakete (IP) in Netzwerken übertragen werden. TCP ist also ein *Transport Layer* und IP ein *Internet Layer*. Ein Unix-Domain-Socket stellt eine Schnittstelle zwischen einzelnen Prozessen für den Austausch von Daten auf einem Unix-(POSIX-)artigen System dar. Diesen Austausch nennt man auch *Inter Process Communication* (IPC). Im Unterschied zu TCP/IP wird dabei kein Netzwerkprotokoll genutzt und er findet immer auf dem gleichen Rechner statt.

len müssen, dass Sie ein so codiertes Passwort „over the wire“ schicken. Glücklicherweise hasht die *php5-pgsql*-Erweiterung das Passwort bereits mit MD5. Intern speichert PostgreSQL Benutzerpasswörter übrigens standardmäßig MD5-gehasht, außer Sie geben explizit an, dass das nicht geschehen soll. Die METHOD *password* bedeutet in diesem Fall, dass generell ein Passwort erforderlich ist, das auch *plain* sein kann. Der Gegensatz dazu wäre die METHOD *trust*, die im Prinzip sagt, dass kein Passwort erforderlich ist. In produktiven Systemen schlage ich vor, von dieser Methode eher die Finger zu lassen.

Leider würde eine genaue Untersuchung aller möglichen Werte für die unterschiedlichen Angaben den Rahmen dieses Artikels sprengen. Deshalb möchte ich an dieser Stelle auf die Onlinedokumentation von PostgreSQL unter [7] verweisen. Prinzipiell werden Sie aber mit der obigen *pg_hba.conf*-Datei als Grundlage erfolgreich Verbindungen auf Ihren Cluster zulassen können.

Die Datei postgresql.conf

Die zweite wichtige Datei in einem PostgreSQL-Cluster ist die Datei *postgresql.conf*. Wenn die Datei *pg_hba.conf* die Authentifizierung regelt, werden in der *postgresql.conf* alle möglichen Einstellungen für den laufenden Betrieb vorgenommen. Im nächsten Teil dieser Artikelserie werden wir die Datei wesentlich näher unter die Lupe nehmen als wir es hier tun. Unser Ziel ist es momentan, die grundlegenden Einstellungen vorzunehmen, sodass wir mit der PostgreSQL-Datenbank verbinden und arbeiten können.

Ein ganz wichtiger Punkt ist die Tatsache, dass die Einstellungen in der *postgresql.conf* bzgl. des Speicherverbrauchs und ähnlichen Einstellungen sehr konservativ gehalten ist. Das Ziel der Entwickler ist es, PostgreSQL auf so vielen Systemen mit so unterschiedlichen Hardwareausstattungen als möglich lauffähig zu halten. Das bedeutet für Sie, dass Sie die Einstellungen unbedingt an Ihre Hardware und Art von Applikation anpassen sollten. Das wird unter anderem ein Thema des zweiten Teils dieser Artikelserie sein. An dieser Stelle der Hinweis, dass viele Parameter auskommentiert sind. Die Werte, die hier angegeben sind, sind die Standardwerte. Wollen Sie also einen Wert ändern, nehmen Sie das Kommentarzeichen (#) weg und ändern den Wert.

Hier interessieren uns drei wichtige Dinge. Zum einen muss sichergestellt werden, dass PostgreSQL auf dem richtigen Port lauscht. Standardmäßig ist das der Port 5432 und wird über den Parameter *port = 5432* eingestellt. Dann müssen wir einstellen, auf welchen IP-Adressen die Datenbank lauschen soll. Standardmäßig ist hier *localhost* eingestellt. Wenn wir aber von einem entfernten Rechner zugreifen, wird das nicht reichen. Deshalb können Sie der Einfachheit wegen angeben, dass auf allen IP-Adressen gelauscht werden soll. Nein, das ist kein Sicherheitsrisiko, denn weder die *postgresql.conf* noch die *pg_hba.conf* sind dafür da, eine Firewall zu ersetzen. Dafür gibt es andere Tools. Also: *listen_addresses* = '*'. Dann ist ja des Sysadmins Lieblingsfrage bei auftretenden Problemen: „Was sagt denn das Log?“ Also prüfen wir, dass ordentlich geloggt wird. Wenn Sie ins *syslog* loggen möchten, geben Sie *log_destination* = '*syslog*' an. Standardmäßig loggt PostgreSQL in ein eigenes Verzeichnis. Die Parameter, die für das Loggen zuständig sind, heißen:

```
data_directory = '/var/lib/postgresql/8.3/main'
                                        (ganz oben)
log_destination = 'stderr'
logging_collector = on
log_directory = 'pg_log
log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log'
log_rotation_age = 1d
log_rotation_size = 10MB
```

Die Einstellungen geben an, dass in das Verzeichnis */var/lib/postgresql/8.3/main/pg_log/* Logfiles mit dem Namen *postgresql-2009-07-06_131426.log* geschrieben werden. Ob Sie das so wollen oder lieber die *syslog*-Variante bevorzugen, bleibt Ihnen überlassen. Wichtig ist nur, dass geloggt wird. Natürlich gibt es noch viele weitere Parameter. Allerdings ist die Betrachtung der genannten hier ausreichend, um den Cluster so zu konfigurieren, dass es losgehen kann.

Fazit und Ausblick

The job is done. Nach der Lektüre dieses Artikels und entsprechendem Einsatz Ihrerseits haben Sie einen PostgreSQL-Cluster installiert und so konfiguriert, dass Sie damit arbeiten können. Ich denke, der Mythos, dass PostgreSQL schwer zum Laufen zu bringen ist, „busted“ ist. Alles in allem sollte das Lesen des Artikels nicht mehr als eine halbe Stunde dauern, und das Installieren und Einrichten von PostgreSQL nicht mehr als 30 bis 45 Minuten. Ist das zu viel, um eine professionelle Datenbank zu installieren? Ich denke, nein. Im nächsten Teil dieser Artikelserie werde ich unter anderem näher auf die *postgresql.conf* und das Tuning der Datenbank eingehen. Im dritten Teil der Artikelserie sollen Themen wie Backup, Recovery und andere weiterführende Themen nicht fehlen. Ich hoffe, Sie sind weiterhin dabei und lassen sich überraschen, denn ... PostgreSQL rocks!

Links & Literatur

[1] PostgreSQL: http://www.postgresql.org

[2] Source der Version 8.4.1: http://www.postgresql.org/ftp/source/v8.4.1/

[3] EnterpriseDB – The Enterprise Postgres Company: http://www.enterprisedb.com

[4] pgAdmin: http://www.pgadmin.org/

[5] PostgreSQL-8.8-Installer: http://www.enterprisedb.com/products/pgdownload.do

[6] MacPorts-Paketmanager: http://www.macports.org/

[7] Informationen zur Datei pg_hba.conf: http://www.postgresql.org/docs/8.4/interactive/auth-pg-hba-conf.html

Andreas Wenk

Andreas Wenk ist Softwarearchitekt bei der NMMN in Hamburg und hat mit Thomas Pfeiffer zusammen das Buch „PostgreSQL 8.4“ für den Galileo Computing Verlag geschrieben. Sie erreichen ihn unter a.wenk@netzmeister-st-pauli.de und können Ihm unter @awenkhh folgen.

Private Network & VPN

serverloft

Sie haben die Wahl:
AMD oder Intel

Private Network und VPN

- Cluster mit beliebig vielen Servern
- Verbindung der Server als Private Network
- Verschlüsselter VPN-Zugang
- Bis zu 1000 MBit/s Bandbreite

pro Server ab EUR/Monat **9,–**

Kein Setup!

149,–
Euro gespart!

Willkommen bei serverloft

serverloft steht für professionelle Root-Server ohne Kompromisse. Wir richten uns an Anwender, die Wert auf das Wesentliche legen: Performante Hardware mit Server-Prozessoren von AMD oder Intel und PRIMERGY Servern von Fujitsu, ein schnelles, stabiles Netzwerk und transparente Angebote mit einem außergewöhnlich guten Preis-Leistungsverhältnis.

Remote Management kostenlos inklusive:
ServerView von Fujitsu
iRMC inklusive Reboot, Recovery, etc.
Remote-Console und Server-Monitoring

FUJITSU

Wir überlassen Ihnen die Konfiguration und die Verwaltung Ihres Servers vollständig, garantieren aber bei Hardware-Defekt einen Austausch Ihres Servers innerhalb von 4 Stunden – 24 Stunden am Tag, 7 Tage die Woche.

Gehen Sie keine Kompromisse ein.

Ihr
Thomas Strohe
Thomas Strohe, Geschäftsführer

Powered by plusserver

PerfectServer L – Xeon
Fujitsu PRIMERGY RX200
1x Xeon DP E5405, 1x Quadcore
48GB DDR2-RAM
2x 250 GB SATA II-HDDs, 7.2 k
5.000 GB Datentransfer inkl.

PerfectServer L – Opteron
Fujitsu PRIMERGY RX330
1x Opteron 2344 HE, 1x Quad-Core
4 GB DDR2-RAM
2x 250 GB SATA II-HDDs, 7.2 k
5.000 GB Datentransfer inkl.

Keine Mindestlaufzeit!
Keine Einrichtungsgebühr!
Preis/Monat: **79,–**

PerfectServer XXL – Xeon
Fujitsu PRIMERGY RX200
2x Xeon DP E5405, 2x Quadcore
16 GB DDR2-RAM
2x 146 GB SAS-HDDs, 15 k
15.000 GB Datentransfer inkl.

PerfectServer XXL – Opteron
Fujitsu PRIMERGY RX330
2x Opteron 2344 HE, 2x Quad-Core
16 GB DDR2-RAM
3x 500 GB SAS-HDDs, 7.2 k
15.000 GB Datentransfer inkl.

Keine Mindestlaufzeit!
Keine Einrichtungsgebühr!
Preis/Monat: **179,–**

Bei allen Angeboten haben Sie freie Wahl zwischen openSuSE 11.1, Ubuntu 8.04 LTS, Debian 4.0 & 5.0 LAMP, CentOS 5 oder Windows Webserver 2008 (zzgl. 30 EUR/Monat). Zur Administration ist Plesk 8.x oder 9.x „10 Domains" auf Wunsch kostenlos erhältlich.

Keine Einrichtungsgebühr bis 31.01.2010. Jedes zusätzliche 1 GB Datentransfer über Freikontingent EUR 0,19. Alle Preise in Euro inklusive 19% Mehrwertsteuer.

Tel. 0800 100 4082

www.serverloft.de