MySQL-Auswahl gegen die Distribution

Jörg Brühe, FromDual GmbH

Von MySQL gibt es verschiedene Varianten (Oracle Community, Oracle Enterprise, MariaDB, Percona) jeweils in mehreren Versionen, und die Linux-Distributionen liefern ihre eigenen Builds auf Basis einer dieser Varianten. Wenn der System- oder Datenbank-Administrator eine andere Variante als die der Distribution installieren will, gibt es häufig Abhängigkeitskonflikte. Dieser Artikel beschreibt, warum das so ist, wie man diese Konflikte analysiert und sie dann löst, ohne das Paket-Management zu umgehen. Im Ergebnis wird so genau die gewünschte Variante installiert, und alle Abhängigkeiten sind erfüllt.

Der Begriff „MySQL“ wird mit zwei verschiedenen Bedeutungen benutzt: Zum einen bezeichnet er ein ganz bestimmtes Produkt der Firma Oracle, das sie in zwei Varianten anbietet („Community Edition“ und „Enterprise Edition“), zum anderen alle Produkte, die von den GPL-Quellen dieses Produkts abgeleitet wurden. In diesem Artikel wird MySQL im zweiten Sinn benutzt; bei den abgeleiteten Produkten geht es um „MariaDB“ und „Percona Server“. Die Namen gehören markenrechtlich natürlich den jeweiligen Anbietern, das hat aber die allgemeine Nutzung (und begriffliche Ausweitung) von „MySQL“ nicht verhindert.

Wer die Wahl hat …

Wie bereits angedeutet, stehen als MySQL-Datenbanksystem etliche Varianten zur Wahl:

- MySQL Enterprise Edition (Oracle)
- MySQL Community Edition (Oracle)
- Percona Server (Percona)
- Percona XtraDB Cluster (Percona)
- MariaDB Server (MariaDB)
- Galera Cluster (Codership)

Meist sind zwei Releases als „allgemein verfügbar“ deklariert, also bei Oracle derzeit MySQL 5.6 und MySQL 5.7. Nähere Informationen zu den Versionen, ihren

Features etc. gibt es bei den jeweiligen Anbietern.

In diesem Artikel geht es um MySQL auf Linux. Hier wird die Vielfalt noch dadurch vergrößert, dass die meisten Distributionen selbst MySQL-Pakete bauen und ausliefern, der Anwender beziehungsweise Administrator ein MySQL-Paket also nicht separat vom Anbieter (Oracle, Percona oder MariaDB) holen muss. Typischerweise nutzt die Distribution diese Pakete dann auch, um anderen Komponenten ein Datenbank-System verfügbar zu machen. So gibt es Abhängigkeiten, durch die das MySQL-Paket der Distribution zwingend installiert wird.

Jede Distribution entscheidet sich vor der Herausgabe eines Release für eine MySQL-Variante und nimmt davon das Release, das als „allgemein verfügbar" erklärt ist. Innerhalb der Support-Zeit für das Distributions-Release wird dann die MySQL-Version auch aktualisiert, es gibt jedoch keinen Release-Wechsel. Das gilt selbst dann, wenn der Anbieter („Upstream") dieses MySQL-Release nicht mehr unterstützt.

Ein Beispiel dafür ist zurzeit Red Hat Enterprise Linux 6 (RHEL, einschließlich kompatibler Distributionen wie CentOS, Scientific Linux und Oracle Linux). RHEL 6.0 wurde am 10. November 2010 veröffentlicht, das Release soll bis zum 30. November 2020 Support erhalten (Angaben laut Wikipedia). Damals war MySQL 5.1 das aktuelle Release (5.5 wurde erst im Dezember 2010 als „GA" erklärt), und so enthält RHEL 6 weiterhin MySQL 5.1 – während Oracle den Support dafür im Dezember 2013 eingestellt hat. Hier zeigen sich die unterschiedlichen Interessen von Distributionen und Software-Herstellern:

- Eine Distribution liefert einen Satz von aufeinander abgestimmten Paketen, die auf gemeinsamen Basis-Funktionen aufbauen und für die jeweils höhere Schicht ihre Leistung durch konstante Schnittstellen (APIs) verfügbar machen. Das Ziel einer Distribution ist, die insgesamt angebotenen Funktionen über die gesamte Lebensdauer für den Anwender stabil bereitzustellen.
- Ein Software-Hersteller liefert eine bestimmte Komponente (wie MySQL), die gewisse Basis-Funktionen (System Calls, C-Bibliothek) voraussetzt und so ihre Leistung für andere Komponenten (Anwendungen) anbietet. Ziel des Herstellers ist, seine Komponente stets aktuell zu halten, sodass der Anwender möglichst schnell von Verbesserungen profitiert.

Während also für die Distribution die stabile Nutzbarkeit (zu der natürlich auch Fehlerbehebungen, also Upgrades, gehören) aller Komponenten gemeinsam an erster Stelle steht, ist es für den Hersteller die Aktualität seiner Komponente (unter der Einschränkung einer für den Anwender hinreichend langen Lebensdauer).

... hat die Qual

„Qual" ist hier die „Qual der Entscheidung": Selbstverständlich bietet jede MySQL-Variante und -Version die üblichen Grundfunktionen, ist also für viele Zwecke ausreichend. Ebenso selbstverständlich führt aber die Weiterentwicklung der Anbieter zu Änderungen (meist: Verbesserungen) durch funktionale Erweiterungen und Fehlerbehebungen, die für manche Zwecke entscheidend sind. Als Beispiele seien genannt:

- Seit MySQL 5.6 unterstützt auch InnoDB die Volltext-Suche; Anwendungen, für die dieses Feature wesentlich ist, sind also nicht mehr zur Benutzung von MyISAM gezwungen
- Seit MySQL 5.7 unterstützt auch InnoDB die speziellen Indizes für geografische Objekte und die dazugehörigen Funktionen
- MySQL 5.7 bringt weitere neue Features wie den Datentyp „JSON" mit dazu passenden Funktonen sowie „generierte Spalten"
- Nur die Enterprise Edition von Oracle enthält die zusätzlichen Tools wie „Enterprise Monitor" und „Audit Plug-in"
- Percona hat MySQL um „TokuDB" als weiteren Table-Handler erweitert und weitere Performance- sowie Diagnose-Features implementiert
- Wer für Hochverfügbarkeit oder Lastverteilung eine produktionsreife Cluster-Lösung haben möchte, muss eine Variante mit Galera-Cluster nehmen (von Codership auf Basis Oracle-Community, von MariaDB oder von Percona)
- Der Hersteller einer Anwendung kann eine bestimmte MySQL-Version vorschreiben, und eine Anwendung kann eine bestimmte Fehlerbehebung benötigen
- Das Auslaufen des Anbieter-Supports kann ein Kriterium für den Einsatz einer neueren Version sein
- Auch Vorgaben aus der Firmenpolitik oder Einschätzungen der Anbieter-Qualität können die Wahl der Variante beeinflussen

Pakete und Lieferdienst

Üblicherweise gibt es MySQL-Versionen für Linux in zwei unterschiedlichen Formaten: als „tarball" („.tar.gz") und als Distributionsspezifisches Paket („.rpm" beziehungsweise „.deb"). Für tarball kann man den Installationsort frei wählen (muss dann gegebenenfalls die Variable „$PATH" anpassen), für ein RPM- oder DEB-Paket wird er beim Paketbau festgelegt.

Die meisten Installationen benutzen Pakete, weil für sie die Anpassung an die Distribution sichergestellt ist und weil die Installation und das Upgrade ohne manuelle Eingriffe erfolgen. Voraussetzung für ein automatisiertes Upgrade ist, dass das jeweilige Repository im System eingerichtet ist, sodass das Upgrade-Tool („yum", „zypper", „apt-get", „synaptic" etc.) die Existenz einer neueren Version feststellen und die Pakete dann auf die Maschine holen kann. Pakete bringen allerdings die Einschränkung, dass nur eine Version installiert sein kann, sonst käme es ja zu Namenskonflikten – es kann auf einer Maschine nur ein „/usr/bin/mysql" geben.

Sofern nicht ungewöhnliche Gründe die parallele Installation mehrerer MySQL-Versionen erzwingen, ist die Entscheidung für Pakete die bessere Wahl. Wenn also eine bestimmte MySQL-Variante in einer bestimmten Version installiert werden soll, dann muss dafür ein etwa bereits installiertes MySQL-Paket entfernt werden. Das kann schwierig werden, wenn andere installierte Pakete von diesem MySQL-Paket abhängen: Wird es entfernt, dann werden diese Abhängigkeiten verletzt. Die Tools zum Paket-Management („rpm" beziehungsweise „dpkg") verhindern das, wenn der Administrator es nicht durch eine „force"-Option explizit verlangt –dann ist jedoch die Konsistenz des Gesamtsystems nicht mehr gesichert.

Solange im System nur die Pakete der Distribution benutzt werden, ist das alles kein Problem: Die Distribution stellt

die Upgrade-Möglichkeit sicher, und durch ihre Festlegung auf ein bestimmtes MySQL-Release ist auch das korrekte API für die installierten Anwendungen garantiert. Schwieriger wird es, wenn die Pakete der Distribution durch Pakete eines anderen Anbieters ersetzt werden sollen, typischerweise also durch solche von Oracle, MariaDB, Percona oder Codership: Meistens führt ein solcher Versuch zu einer Fehlermeldung wegen Datei-Konflikten oder unerfüllten Abhängigkeiten und scheitert dann. Nachfolgend ist das näher beschrieben, analysiert und dann sauber umgangen.

Wie ein Upgrade scheitert

Das Beispiel in *Listing 1* zeigt den Versuch, auf einem System mit CentOS 7 die Community Edition von MySQL 5.7 (Oracle) zu installieren. Zunächst wird der Ausgangszustand bestimmt und im CentOS-Repository nach MySQL-Paketen gesucht.

Man sieht, dass CentOS 7 keine Pakete auf Basis von Oracles MySQL liefert, sondern MariaDB 5.5 enthält. Bei Oracle endete der Support für MySQL 5.5 im Dezember 2015, bei MariaDB ist das für April 2017 angekündigt. Die gewünschten Pakete müssen also direkt von Oracle geholt werden. Das könnte über „yum" erfolgen, aber mit einer Trennung zwischen Download und lokaler Installation (mit „rpm") gibt es bessere Diagnose-Möglichkeiten. Oracle bietet einen „tarball" aller RPM-Pakete („bundle") an, durch Download und Auspacken erhält man diese Dateien (siehe *Listing 2*, weitere sind ausgelassen). Ein Versuch, das Client- und das Server-Paket zu installieren, scheitert gründlich *(siehe Listing 3)*.

Der erste Schritt ist natürlich, die fehlenden System-Pakete aus dem CentOS-Repository zu installieren; das ist einfach und gelingt mit „demo$ sudo yum install libaio net-tools". Nun wird erneut versucht, die MySQL-Pakete zu installieren, außer „client" und „server" auch die benötigten Pakete „libs" und „common" *(siehe Listing 4)*.

Diese Fehlermeldung ist durch ihre schlechte Formulierung nur schwer verständlich. Gemeint ist: Die Metadaten von „mysql-community-libs" verlangen, „mariadb-libs" zu ersetzen, aber die „install"-

```
demo$ cat /etc/redhat-release
CentOS Linux release 7.2.1511 (Core)
demo$ rpm -qa | egrep -i 'mysql|maria'
mariadb-libs-5.5.35-3.el7.x86_64
demo$ yum info 'mysql*'
...
Verfügbare Pakete
Name      : MySQL-python
...
Name      : mysql-connector-java
...
Name      : mysql-connector-odbc
...
```

Listing 1

```
demo$ ls -1 /Download/
mysql-5.7.14-1.el7.x86_64.rpm-bundle.tar
mysql-community-client-5.7.14-1.el7.x86_64.rpm
mysql-community-common-5.7.14-1.el7.x86_64.rpm
...
mysql-community-libs-5.7.14-1.el7.x86_64.rpm
mysql-community-libs-compat-5.7.14-1.el7.x86_64.rpm
mysql-community-server-5.7.14-1.el7.x86_64.rpm
...
```

Listing 2

```
demo$ sudo rpm -i /Download/mysql-community-{client,server}-5.7.14-1.el7.x86_64.rpm
Warnung: ... NOKEY
Fehler: Fehlgeschlagende Abhängigkeiten:
mysql-community-libs(x86-64) >= 5.7.9 wird benötigt von mysql-community-client-5.7.14-1.el7.x86_64
libaio.so.1()(64bit) wird benötigt von mysql-community-server-5.7.14-1.el7.x86_64
libaio.so.1(LIBAIO_0.1)(64bit) wird benötigt von mysql-community-server-5.7.14-1.el7.x86_64
libaio.so.1(LIBAIO_0.4)(64bit) wird benötigt von mysql-community-server-5.7.14-1.el7.x86_64
mysql-community-common(x86-64) = 5.7.14-1.el7 wird benötigt von mysql-community-server-5.7.14-1.el7.x86_64
net-tools wird benötigt von mysql-community-server-5.7.14-1.el7.x86_64
```

Listing 3

```
demo$ sudo rpm -i /Download/mysql-community-{client,common,libs,server}-5.7.14-1.el7.x86_64.rpm
Warnung: ... NOKEY
Fehler: Fehlgeschlagende Abhängigkeiten:
mariadb-libs wird ersetzt durch mysql-community-libs-5.7.14-1.el7.x86_64
```

Listing 4

```
demo$ sudo rpm -U /Download/mysql-community-{client,common,libs,server}-5.7.14-1.el7.x86_64.rpm
Warnung: ... NOKEY
Fehler: Fehlgeschlagende Abhängigkeiten:
libmysqlclient.so.18()(64bit) wird benötigt von (installiert) postfix-2:2.10.1-6.el7.x86_64
libmysqlclient.so.18(libmysqlclient_18)(64bit) wird benötigt von (installiert) postfix-2:2.10.1-6.el7.x86_64
```

Listing 5

Aktion von „rpm“ (Option „-i“) tut das nicht: Nur ein „update“ (Option „-U“) führt eine solche Verdrängung durch. Also ein neuer Versuch, mit „update“ statt „install“ *(siehe Listing 5)*.

Hier ist nun das inhaltliche Problem sichtbar: Das Paket „postfix“ (das vom Upgrade nicht direkt erfasst wird) benötigt die Datei „libmysqlclient.so.18“. Wenn man ahnt, dass diese Bibliothek vom Upgrade erfasst wird, kann man diese Vermutung sehr einfach prüfen *(siehe Listing 6)*.

Ohne diese Ahnung muss man anders forschen *(siehe Listing 7)*. Beide Wege zeigen: Die Bibliothek ist im zu verdrängenden Paket enthalten, würde also gelöscht werden; dieser Bruch der Abhängigkeiten ist nicht zulässig.

```
demo$ rpm -ql mariadb-libs | fgrep libmysqlclient
/usr/lib64/mysql/libmysqlclient.so.18
/usr/lib64/mysql/libmysqlclient.so.18.0.0
```

Listing 6

```
demo$ find /usr/lib* -name libmysqlclient.so.18 2>/dev/null
/usr/lib64/mysql/libmysqlclient.so.18
demo$ rpm -qf /usr/lib64/mysql/libmysqlclient.so.18
mariadb-libs-5.5.37-1.el7_0.x86_64
```

Listing 7

Server	Client	Anwendung
mysql-server		
	mysql	
mysql-libs	mysql-libs	mysql-libs

Tabelle 1

Server	Client	Anwendung
mysql-ED-server		
mysql-ED-client	mysql-ED-client	
mysql-ED-libs	mysql-ED-libs	mysql-ED-libs
mysql-ED-common	mysql-ED-common	mysql-ED-common

Tabelle 2

Exkurs: Unterpakete und Abhängigkeiten

Für das bessere Verständnis der folgenden Abschnitte ist es hilfreich, die Einteilung der MySQL-Komponenten in Unterpakete zu kennen. Die Vorstellung, es gebe einfach ein Paket „MySQL“, stimmt nicht – sowohl bei den RPM- als auch bei den DEB-Paketen wird zwischen dem Server, den mitgelieferten Client-Programmen und den Bibliotheken für Anwendungen unterschieden. Die genaue Einteilung entscheidet jeder Paket-Ersteller selbst. *Tabelle 1* zeigt, wie diese etwa bei Red Hat 6 mit MySQL 5.1 war.

Das Client-Paket erkennt man also nicht direkt am Namen, und sowohl das Server- als auch das Client-Paket hängen vom Bibliotheks-Paket ab. Oracle hat die Einteilung verfeinert und „common“ für gemeinsame Dateien eingeführt, benennt das Client-Paket ausdrücklich und erzwingt die Installation der Client-Programme auch beim Server *(siehe Tabelle 2)*. „ED“ steht für „Edition“, also „community“ beziehungsweise „commercial“ im Paketnamen, die Einteilung ist bei beiden gleich.

```
demo$ sudo rpm -e mariadb-libs
Fehler: Fehlgeschlagende Abhängigkeiten:
libmysqlclient.so.18()(64bit) wird benötigt von (installiert) post-
fix-2:2.10.1-6.el7.x86_64
libmysqlclient.so.18(libmysqlclient_18)(64bit) wird benötigt von (in-
stalliert) postfix-2:2.10.1-6.el7.x86_64
```

Listing 8

Ursachenforschung

Die bisherigen Erkenntnisse sind:

- Sowohl das Server- als auch das Client-Paket lassen sich nicht ohne das Libs-Paket installieren
- Das Paket „mysql-community-libs“ kollidiert mit „mariadb-libs“ und will es verdrängen. Das beruht auf Gegenseitigkeit, „mariadb-libs“ würde ein installiertes „mysql-community-libs“ ebenfalls verdrängen.
- Diese Verdrängung scheitert, weil sie die Anforderungen des installierten „postfix“-Pakets verletzen würde

Den dritten Punkt sollte man sich noch einmal genauer anschauen *(siehe Listing 8)*. Diese Fehlermeldung sieht etwas selt-

sam aus, weil sie „mariadb-libs“ nicht erwähnt. Was ist die genaue Abhängigkeit? *Listing 9* zeigt, was „rpm“ antwortet. Wenn das Paket „mariadb-libs“ nicht verlangt wird, was ist dann die Abhängigkeit? Auch das sagt „rpm“ in *Listing 10*.

```
demo$ rpm -q --whatrequires mariadb-libs
Kein Paket benötigt mariadb-libs
```

Listing 9

```
demo$ rpm -q --requires postfix
...
libmysqlclient.so.18()(64bit)
libmysqlclient.so.18(libmysqlclient_18)(64bit)
...
```

Listing 10

```
demo$ rpm -q --provides mariadb-libs
...
libmysqlclient.so.18()(64bit)
libmysqlclient.so.18(libmysqlclient_18)(64bit)
...
```

Listing 11

Die Abhängigkeit besteht also nicht gegenüber dem Paket, sondern gegenüber der enthaltenen Bibliothek. Unabhängig davon, ob der Paket-Ersteller das angibt, erzeugt „rpm“ standardmäßig für jede im Paket enthaltene Bibliothek auch „provides“-Angaben *(siehe Listing 11)*. Die Lösung besteht also darin, die Bibliothek „libmysqlclient.so.18“ aus einem anderen Paket zu installieren, sodass dann „mariadb-libs“ zugunsten der Oracle-MySQL-Pakete verdrängt werden kann.

Die saubere Lösung

Unter den ausgepackten Oracle-MySQL-RPMs waren auch weitere, die bisher nicht betrachtet wurden *(siehe Listing 12)*. „rpm“ liefert die Auskunft darüber, was „libs-compat“ und „libs“ zur Verfügung stellen und wie sie sich dabei unterscheiden *(siehe Listing 13)*.

Während also das „libs“-RPM die Version 20 der Client-Bibliothek enthält, ist in „libs-

compat“ noch die Version 18. Das ist genau der Zweck eines solchen „compatibility“-Pakets: Nach dem Wechsel auf neue Software-Versionen (hier: von MySQL) soll den anderen, benutzenden Programmen noch ein Stand angeboten werden, der mit ihrer Übersetzung kompatibel ist. Das Postfix-Paket von CentOS 7 ist ja übersetzt worden für die Client-Bibliothek aus MySQL (genauer: MariaDB) 5.5, die trägt als „shared object“ die Version 18.

Anders formuliert: Das „libs-compat“-Paket von MySQL 5.7 für CentOS 7 enthält diejenige „libmysqlclient“, die mit dem Original von CentOS 7 kompatibel ist, also die aus MySQL 5.5. Es gibt auch ein „libs-compat“-Paket von MySQL 5.7 für CentOS 6, das enthält die „libmysqlclient“ aus MySQL 5.1, weil CentOS 6 noch MySQL 5.1 benutzt.

Ein „libs-compat“-RPM hat seinerseits wieder Abhängigkeiten – aber nicht zur alten Version 5.5 (die soll ja durch das Upgrade ersetzt werden), sondern zur aktuellen Version 5.7 *(siehe Listing 14)*. „libs-compat“ muss also zusammen mit „libs“ (und darum mit „common“) installiert werden und die Aktion muss „upgrade“ sein, um „mariadb-libs“ zu verdrängen *(siehe Listing 15)*.

Diese Verdrängung ist erfolgt, „mariadb-libs“ ist durch „mysql-community-libs-compat“ ersetzt. Jetzt können das Server- und das Client-Paket installiert werden *(siehe Listing 16)*. Bei Bedarf können natürlich auch weitere Pakete installiert werden, etwa die Tests und die Header für Software-Entwicklung.

Fazit

- Distributionen und Software-Hersteller liefern unterschiedliche Pakete
- Die Hersteller-Pakete sind neuer als die Distributions-Pakete und bieten die größere Auswahl
- Wenn schon ein Distributions-Paket installiert ist, kann die Installation eines Hersteller-Pakets seine Verdrängung erfordern. Bei „rpm“ und „yum“ ist dafür eine „Upgrade“-Aktion nötig
- Falls das zu verdrängende Paket (im Beispiel „mariadb-libs“) von einem anderen Paket (im Beispiel „postfix“) benötigt wird, ist diese Abhängigkeit zu analysieren und es muss ein passender Ersatz gefunden werden (im Beispiel „mysql-libs-compat“).

```
demo$ ls -1 /Download/
mysql-5.7.14-1.el7.x86_64.rpm-bundle.tar
mysql-community-client-5.7.14-1.el7.x86_64.rpm
mysql-community-common-5.7.14-1.el7.x86_64.rpm
...
mysql-community-libs-5.7.14-1.el7.x86_64.rpm
mysql-community-libs-compat-5.7.14-1.el7.x86_64.rpm
mysql-community-server-5.7.14-1.el7.x86_64.rpm
...
```

Listing 12

```
demo$ rpm -qp --provides mysql-community-libs-compat-5.7.14-1.el7.
x86_64.rpm
...
libmysqlclient.so.18()(64bit)
...
demo$ rpm -qp --provides mysql-community-libs-5.7.14-1.el7.x86_64.rpm
...
libmysqlclient.so.20()(64bit)
...
```

Listing 13

```
demo$ rpm -qp --requires mysql-community-libs-compat-5.7.14-1.el7.
x86_64.rpm
...
mysql-community-libs(x86-64) >= 5.7.9
...
demo$ rpm -qp --requires mysql-community-libs-5.7.14-1.el7.x86_64.rpm
...
mysql-community-common(x86-64) >= 5.7.9
...
```

Listing 14

```
demo$ sudo rpm -U /Download/mysql-community-{libs-
compat,libs,common}-5.7.14-1.el7.x86_64.rpm
Warnung: ... NOKEY
demo$ rpm -qa | egrep -i 'mysql|maria'
mysql-community-common-5.7.14-1.el7.x86_64
mysql-community-libs-compat-5.7.14-1.el7.x86_64
mysql-community-libs-5.7.14-1.el7.x86_64
```

Listing 15

```
demo$ sudo rpm -i /Download/mysql-community-{client,server}-5.7.14-1.
el7.x86_64.rpm
Warnung: ... NOKEY
demo$ rpm -qa | egrep -i 'mysql|maria'
mysql-community-common-5.7.14-1.el7.x86_64
mysql-community-libs-compat-5.7.14-1.el7.x86_64
mysql-community-server-5.7.14-1.el7.x86_64
mysql-community-libs-5.7.14-1.el7.x86_64
mysql-community-client-5.7.14-1.el7.x86_64
```

Listing 16

- Die Abhängigkeit besteht nicht unbedingt zwischen den Paketen, sondern oft zu einer enthaltenen Bibliothek („shared library“). Dabei ist auf die „so“-Version zu achten
- Häufig ist die Analyse einfacher, wenn man zunächst die Pakete herunterlädt und sie dann lokal analysiert (mit „rpm“ beziehungsweise „dpkg“), als wenn man das Repository befragt (mit „yum“ beziehungsweise Apt-Tools).

Mit dieser Analyse ist es möglich, installierte MySQL-Pakete durch solche eines anderen Paketierers zu ersetzen (die natürlich für die gegebene Plattform passend erstellt sein müssen), ohne dabei die Abhängigkeiten des Paket-Managements zu übergehen oder zu brechen. Der kritische Bereich sind die Anforderungen von installierten Anwendungen, die durch Kompatibilitäts-Pakete (meist „libs-compat“) erfüllt werden.

Falls das (mangels Angebot) nicht möglich ist, ist die Installation aus „tarball“ („tar.gz“) die saubere Alternative. Dafür gibt es allerdings keine Upgrade-Unterstützung, sodass der Administrator das selbst periodisch prüfen muss.

Jörg Brühe
joerg.bruehe@fromdual.com

MySQL® in der Oracle Public Cloud – den Durchblick behalten

Mario Beck und Carsten Thalheimer, ORACLE Deutschland B.V. & Co. KG

Während die MySQL-Datenbank bereits seit einigen Jahren im Amazon-, Microsoft- und Google-Cloud-Angebot ist, hat sich Oracle als Eigentümer von MySQL erstaunlich lange mit einem MySQL Cloud Service Zeit gelassen und den bereits seit Juli 2016 laufenden Testbetrieb während der Oracle OpenWorld freigegeben.

Auch im Jahr 2016 nutzte die Oracle-MySQL-Gruppe wieder die OpenWorld als Plattform, um praktisch alle Produkte der MySQL Enterprise Product Suite auf den neuesten Stand zu bringen. Der Fokus der rund fünfzehn MySQL Product Announcements *(siehe „http://forums.mysql.com/list.php?3“)* lag vor allem auf der neuen Version 8.0, die wie üblich als Development Milestone Release 1 (Beta Quality) als Binary und im Source Code für gängige Plattformen zur Verfügung gestellt wurde. Die Versionsnummern MySQL 6 (Namenskonflikt mit älteren Sun-Microsystems-MySQL-Plänen) und MySQL 7 (reserviert für MySQL NDB Cluster) werden somit übersprungen.

Nicht minder interessant sind die Neuigkeiten in der Oracle Public Cloud (OPC). Das Presse-Release liest sich zunächst recht nüchtern: Es wird von einem sicheren, kosteneffektiven und unternehmensorientierten MySQL Cloud Service (CS) gesprochen, ohne allerdings auf die technischen Details einzugehen.

Die in der Cloud verwendete MySQL-Version basiert auf MySQL Enterprise 5.7.16 und ist somit 100 Prozent kompatibel zu den im Jahr 2015 veröffentlichten MySQL 5.7 Community und kommerziellen Versionen. Somit ist gewährleistet, dass die Daten sowohl in der Cloud als auch On-Premise in eigenen, lokalen Rechenzentren verwendet werden. Eine Daten-Migration in die Cloud beziehungsweise zurück in das Rechenzentrum ist somit aus Datenbank-Sicht denkbar einfach.

Das im MySQL Cloud Service eingesetzte Release ist die kommerzielle Version von MySQL. Das bedeutet, dass Anwender über die normale MySQL-Funktionalität hinaus noch weitere Features nutzen können. Hervorzuheben sind hier vor allem die Security-Erweiterungen wie transparente Datenverschlüsselung (TDE), Auditing und die integrierte SQL-Firewall.

Das erste Aufsetzten einer MySQL-Instanz in der Oracle-Cloud geht schnell von der Hand. Auf Basis eines vorkonfigurierten und MySQL-optimierten Oracle-Compute-Images wird eine virtuelle Betriebssystem-