puppet

OPatch ferngesteuert: Automatisiertes Patchen mit Open Source Puppet

Raphael Daum, DBConcepts GmbH

Das Patchen einer Datenbank gehört zu den regelmäßig wiederkehrenden Aufgaben eines Datenbank-Administrators. Eine innige Beziehung wird aber selten daraus. Auf die Frage, ob das Patchen denn Spaß mache, pendelt die Antwort zwischen „na ja“ und „nein“. Aber wieso ist das so? Kann es nicht auch anders sein? Wenn viele der einzelnen und mühsamen Schritte automatisiert werden könnten, würde Patchen sogar Spaß machen. Dieser Artikel stellt mit dem Open-Source-Configuration-Management-Tool Puppet eine Möglichkeit vor, um das Patchen von Datenbanken quer über alle Oracle-Editionen kostengünstig zu automatisieren, sodass es zu einer entspannten Tätigkeit wird.

Jedes Quartal stellt Oracle neue Patches für die unterstützten Datenbank-Versionen zur Verfügung. Nicht wenige Unternehmen sind auf diesen Zug aufgesprungen und verlangen ein regelmäßiges und zeitnahes Patchen ihrer Datenbank-Systeme. Das ist in den meisten Fällen mit einigem Arbeitsaufwand für die involvierten Abteilungen verbunden. Es beginnt mit der Evaluierung des Ist-Zustands, der Vorbereitung der Systeme und endet mit der Ausarbeitung von detaillierten Wartungsfenstern und Rückstiegs-Szenarien. Durch viel Übung geht manches schneller von der Hand und kleinere Arbeitsschritte lassen sich mit Skripten vereinfachen. Trotzdem ist der Prozess ein langwieriger und birgt jedes Quartal neue Risiken: Welche Sicherheitslücken schließe ich mit dem Patch und welche bisher unbekannten Probleme könnten durch den aktuellen Patch entstehen? Was tue ich, wenn es schief geht?

Wo wollen wir hin? Wie schnell muss es gehen?

Bevor man sich intensiver damit auseinandersetzt, wie das Patchen automatisiert werden kann, muss geklärt sein, wie der aktuelle Ist-Zustand ist und wie der gewünschte Soll-Zustand aussehen soll. Was verhindert, den gewünschten Soll-Zustand zu erreichen? In den meisten Fällen ist es wohl eine Kombination von unterschiedlichen Faktoren: fehlende Zeit, unklare Prioritäten, mangelnde Praxis, fehlende Patch-Strategie, um nur einige zu nennen.

Die Automatisierung des Patch-Vorgangs kann zwar diese grundlegenden Probleme nicht lösen, sehr wohl aber entscheidend zu einer Lösung beitragen. Denn sobald sich das Unternehmen oder die zuständige Abteilung auf eine verbindliche Patching-Strategie geeignet hat, ist aus unternehmenspolitischer Sicht bereits alles Notwendige getan. Nun beginnt die Phase, in der die technischen Fragen geklärt werden müssen. Während die Patching-Strategie die Frage nach dem Wann beantwortet, gibt die Patch-Automatisierung eine Antwort auf die Frage „Wie patchen?“

Die Qual der Wahl: PSU oder CPU oder SPU?

Bevor wir zur technischen Realisierung kommen, steht noch die Klärung der Frage im Raum, welche Patches die Grundlage für die Automatisierung darstellen können. Oracle bietet eine Fülle verschiedener Patches für Datenbanken an, aber welcher Patch ist der richtige? Critical Patch Update (CPU), Security Patch Update (SPU), Patch Set Update (PSU), Combo Patch (PSU plus OJVM) oder GI-Patch?

Die Erfahrung hat gezeigt, dass ein PSU den meisten Mehrwert und das ge-

ringste Risiko enthält. Aus diesem Grund beruht die hier vorgestellte Lösung auch darauf, ein PSU automatisiert zu installieren. Das hat damit zu tun, dass im PSU sowohl sicherheitsrelevante als auch performancerelevante Bugs geschlossen werden. Zusätzlich listet Oracle drei Vorteile des PSU auf:

- Low-Risk, High-Value Content
- One Integrated, well tested Patch
- Baseline-Version for easier Tracking

Der erste Punkt nimmt Bezug darauf, dass ein PSU keine Konfigurationsänderungen erforderlich macht und das Verhalten der Datenbank nicht verändert, wodurch auch keine Re-Zertifizierung notwendig wird. Im zweiten Punkt wird darauf angespielt, dass bis zu hundert Bugs pro PSU behandelt werden und die Fixes aufeinander abgestimmt und getestet wurden. Punkt drei erleichtert die eindeutige Identifizierung des Patch-Levels der Datenbank. Wird etwa der PSU vom 18. Oktober 2016 eingespielt, ändert sich das Patch-Level der Datenbank auf 12.1.0.2.161018 beziehungsweise 11.2.0.4.161018. Nachdem das aktuelle PSU immer alle vorigen PSUs enthält, ist somit ein damit eindeutiger Patch-Stand gewährleistet.

Spaß am Patchen

Damit wir uns dem Ziel nähern – nämlich das Patchen zu einer entspannten Tätigkeit zu machen –, müssen lediglich zwei Voraussetzungen erfüllt sein: Erstens brauchen wir ein Test-Umgebung und zweitens eine Automatisierung, die uns die ganze händische und oft fehleranfällige Arbeit bei der Patch-Vorbereitung und dem eigentlichen Patchen abnimmt. Durch die Test-Umgebung lässt sich alles im Vorfeld gefahrlos testen und evaluieren, durch die Automatisierung ist Patch-Apply und Patch-Rollback in schnellen Zyklen kein Problem mehr. Automatisieren bringt unter anderem folgende Vorteile mit sich:

- Reproduzierbarkeit
- Die Konfiguration erfolgt an einer Stelle
- Deklaratives Arbeiten (Zielzustand)
- Wiederholte und wiederholbare Ausführung („idempotent“)
- Workflow ist im Code
- Individuelles und personengebundenes Know-how wird in Code übersetzt
- Code ist überwachbar (VCS)
- Sofortige Skalierungsgewinne

Deklarativ statt imperativ

Die Automatisierung von Patch-Vorgängen lässt sich mit sogenannten „Configuration Management Systemen“ implementieren. Deren zentrale Aufgabe ist es, auf einem Zielsystem einen gewünschten (deklarativen) Zustand herzustellen. In unserem Fall verwenden wir so ein System, um auf einem Oracle-Datenbank-Zielsystem einen gewünschten Patch-Level zu realisieren. Als Configuration Management System kommt dabei „Puppet“ zum Einsatz. Es ist Open Source und unterstützt eine Reihe von Plattformen und Architekturen (Linux, Windows, Solaris etc.). Puppet hat sich in den letzten Jahren als äußerst vielseitiges und robustes Configuration Management System einen Namen gemacht und besticht durch eine rein deklarative Ausrichtung, was in komplexen IT-Umgebungen ein starker Trumpf ist: Wir geben nur den gewünschten Zielzustand eines Systems an und müssen uns nicht darum kümmern, was alles erforderlich ist, um diesen zu erreichen.

Es ist Aufgabe von Puppet, für jedes System den korrekten Weg zu wählen (imperativ). Uns interessiert lediglich der Zielzustand (deklarativ). Um diesen zu erreichen, vergleicht Puppet bei jeder Ausführung den Ist- mit dem Soll-Zustand. Besteht hier eine Differenz, werden die notwendigen Schritte durchgeführt, um das System in den Soll-Zustand zu bringen. Entspricht der Ist- bereits dem Soll-Zustand, passiert nichts; nichts anderes ist mit dem Begriff „idempotent“ gemeint.

Automate, automate, automate!

Die Vorteile einer Open-Source-Lösung liegen auf der Hand: Alle nachfolgend aufgeführten Aufgaben erfordern keine Management-Packs von Oracle. Die Patch-Automatisierung kann mit allen Oracle-Editionen (SE, SE ONE, SE2, EE) umgehen und unterstützt die Datenbank-Versionen 11.2.0.4, 12.1.0.1 und 12.1.0.2. Support für neue Releases kann einfach hinzugefügt werden. Oracle bietet mit dem hauseigenen „Database Lifecycle Management Pack“ für den Enterprise Manager eine eigene Lösung für das automatisierte Patchen an. Es umfasst wesentlich mehr Funktionalität als das hier vorgestellte Patching, setzt aber auch den Einsatz der kostspieligen Oracle Enterprise Edition samt Management Pack auf allen Datenbank-Zielen voraus. Der Einsatz für die Datenbanken der Standard Edition (SE, SE ONE, SE2) ist lizenzrechtlich nicht erlaubt.

Abbildung 1 skizziert den Patching-Workflow mit Puppet. Überblicksmäßig realisiert die vorgestellte Open-Source-Lösung folgende Aufgaben:

- *Patch-Inventory*
  In einem zentralen Patch-Inventory sind alle relevanten Informationen für einen Patch hinterlegt. Das betrifft unter anderem die Anwendbarkeit von Patches für verschiedene Architekturen und Versionen, die notwendigen Pre-Requi-

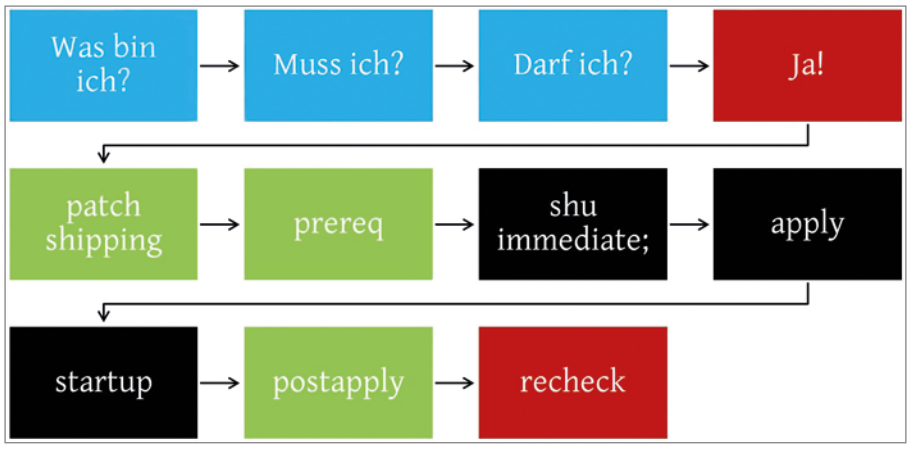

Abbildung 1: Patching-Workflow

```
# 12.1.0.2.1611118 (Oct2016)
12.1.0.2::psu::ensure:      present
12.1.0.2::psu::identifier:  12.1.0.2.161118
12.1.0.2::psu::patchid:     24591642
12.1.0.2::psu::patchfile:   p24922906_121020_MSWIN-x86-64.zip
12.1.0.2::psu::opatch:      12.1.0.1.2
12.1.0.2::psu::apply:       apply
12.1.0.2::psu::postapply:   datapatch.bat -verbose
12.1.0.2::psu::postremove:  datapatch.bat -verbose
12.1.0.2::psu::prereq:
  - CheckConflictAgainstOHWithDetail
  - CheckApplicable
```

Listing 1

```
"oradb_homes_c_oracle_product_11.2.0_dbhome_1": {
  "orabase": "C:\Oracle",
  "homepath": "C:\Oracle\product\11.2.0\dbhome_1",
  "homeowner": "ORA_DBA",
  "homegroup": "S-1-5-32-544",
  "homekey": "OraDb11g_home1",
  "opatchversion": "11.2.0.3.15",
  "oracleversion": "11.2.0.4",
  "oracleproduct": "rdbms",
  "psudesc": "No PSU descrption found",
  "psu": "No PSUs detected"
},

"oradb_homes_c_oracle_product_12.1.0.2_dbhome_1": {
  "orabase": "C:\Oracle",
  "homepath": "C:\Oracle\product\12.1.0.2\dbhome_1",
  "homeowner": "ORA_DBA",
  "homegroup": "S-1-5-32-544",
  "homekey": "OraDB12Home2",
  "opatchversion": "12.1.0.1.3",
  "oracleversion": "12.1.0.2",
  "oracleproduct": "rdbms",
  "psudesc": "WINDOWS DB BUNDLE PATCH 12.1.0.2.161118(64bit):24922906",
  "psu": "24922906"
},
```

Listing 2

rement-Checks, die minimal notwendige OPatch-Version und Schritte zum Installieren und Deinstallieren des Patches.

- *Patch-Storage*
  Das Patch-Inventory verweist auf ein zentrales Patch-Storage, auf dem nicht nur die Patches hinterlegt sind, sondern auch weitere Utilities und Skripte, die für das Patchen notwendig sind.

- *Rollout*
  Die Patches werden auf die frei definierbaren Zielsysteme ausgerollt und entpackt.

- *Dry-Run*
  Ein Dry-Run-Modus ermöglicht es, alle Schritte für das Patchen („apply“ oder „rollback“) zu simulieren und die Pre-Requirement-Checks auszuführen. Das ist eine Aufgabe, die bereits vor dem eigentlichen Wartungsfenster ausgeführt werden kann und soll. Falls es hier zu Problemen kommt, die sich nicht automatisiert lösen lassen, besteht noch ausreichend Zeit, um sie zu korrigieren. Das könnten etwa vorher installierte Oneoff-Patches sein, die mit dem PSU in Konflikt stehen. Hier hilft dann nur ein Eingriff von Hand. Sollte der Dry-Run erfolgreich gewesen sein, sind bereits viele Stolpersteine für das Wartungsfenster aus dem Weg geräumt.

- *Patch*
  Durch das Patch-Shipping sowie die Prüfung der Voraussetzungen und möglichen Patch-Konflikte wird die Downtime auf die reine Patch-Dauer („patch apply“) reduziert. Patches können automatisiert installiert („apply“) und deinstalliert („rollback“) werden.

- *Kill or Report?*
  Sollte das Patchen trotz erfolgreichem Dry-Run scheitern, liegt das meistens an offenen File-Handles und an von anderen Prozessen verwendeten DLLs, was zur Folge hat, dass OPatch die gewünschte Datei nicht aktualisieren kann. Die vorgestellte Patch-Automatisierung verfügt deshalb über die Funktionalität, solche Showstopper ausfindig zu machen und entweder die verantwortlichen Prozesse gleich zu beenden oder lediglich zu berichten.

Komponenten und Konfiguration

Puppet arbeitet mit einer klassischen Client-Server-Architektur: Auf der einen Seite steht ein Puppet-Server (der in unserem Fall auch als Patch-Inventory und Patch-Storage dient), auf dem die gewünschte Ziel-Konfiguration für jedes System hinterlegt ist. Die Konfiguration liegt in einfachen YAML-Files. Auf der anderen Seite läuft am Zielsystem der Puppet-Agent, dessen Aufgabe es ist, regelmäßig den Ist-Zustand des Systems an den Puppet-Server zu schicken und vom Puppet-Server übermittelte Aufträge zur Erreichung des Soll-Zustands umzusetzen. Der Puppet-Agent ist die einzige Software, die am Zielsystem installiert werden muss, um die skizzierte Patch-Automatisierung umzusetzen. Für jeden Patch gibt eine Konfigurationsdatei, in der alle notwendigen Informationen hinterlegt sind *(siehe Listing 1)*.

Am Zielsystem läuft ein Prozess mit dem Namen „facter“, der – wie der Namen bereits erahnen lässt – den aktuellen Ist-Zustand erhebt und an den Puppet-Server schickt. *Listing 2* zeigt einen Auszug des Outputs.

Fazit

Die vorgestellte Lösung einer Patch-Automatisierung unternimmt den bescheidenen Versuch, Patch-Abläufe stressfreier

und entspannter zu gestalten, und greift dafür auf die etablierte Open-Source-Software „Puppet“ zurück. Durch die Automatisierung kann eine große Anzahl von Datenbanken schnell und regelmäßig auf das gewünschte Patch-Level gehoben werden.

Im Gegensatz zu anderen Lösungen funktioniert das Patchen mit Puppet unabhängig von Datenbank-Version und -Edition und macht es damit zu einer universellen, sehr kostengünstigen Lösung, weil keine zusätzlichen Lizenzen erforderlich sind. Schließlich: Regelmäßiges Patchen spart Nerven (weil das Know-how bereits im Code ist), Zeit (aufgrund der Automatisierung) und Geld (weniger Downtime, weniger Stillstände aufgrund von Bugs).

Raphael Daum
raphael.daum@dbconcepts.at

Entspannt virtualisieren mit dem offenen Oracle-Multipurpose-Hypervisor

Nico Henglmüller und Dr. Thomas Petrik, Sphinx IT Consulting GmbH

Wünschen Sie sich einen stabilen Betrieb Ihrer Virtualisierungslösung sowie Oracle-Datenbanken und Applikationen, und das Ganze in einem sicheren Lizenz-Umfeld? Alles kein Problem, legen Sie die Beine hoch.

Virtualisierungsmethoden sind aus der modernen Betriebsführung nicht mehr wegzudenken. Aktuelle Virtualisierungslösungen können nach ihren Voraussetzungen bei der Installation in zwei Gruppen eingeteilt werden:

- *Bare-Metal*
  Diese Art der Hypervisor wird direkt auf die vorhandene Hardware installiert, verwaltet die Ressourcen-Aufteilung und stellt die Treiber für den Hardware-Zugriff selbst bereit. Vertreter dieser Gruppe sind Xen, VMware ESXi, KVM oder Microsoft Hyper-V.

- *Betriebssystem*
  Diese Gruppe benötigt ein vorinstalliertes Betriebssystem, da dieses die Treiber für den Hardware-Zugriff bereitstellt. Zu dieser Gattung zählen Oracle VM VirtualBox, VMware Workstation oder Parallels Desktop.

Lösungen, die ein Betriebssystem voraussetzen, werden hauptsächlich auf Workstations eingesetzt, während Bare-Metal-Installationen im Server-basierten Umfeld ihren Einsatz finden. In der Gruppe der Bare-Metal-Installationen befinden sich zwei Open-Source-Virtualisierungslösungen: Xen und KVM. Die Referenzliste von Xen reicht von Amazons Webservices über diverse Google-Produkte bis hin zu den Cloud-Angeboten von Rackspace und Yahoo.

Seit dem Jahr 2007 erweitert Oracle schrittweise die Xen-Implementierung des Xen-Projekts und stellt die in Oracle VM verwendeten Quelltexte öffentlich unter der GNU GPL v2 bereit. Das Resultat dieser Anstrengungen ist der Oracle-VM-Server, der Xen den Schrecken nahm und das Herz der Oracle-Virtualisierungslösung ist. Um diesen Kern baute Oracle eine grafische Oberfläche zur Verwaltung sowie ein API zur Automatisierung, ohne den schlanken Hypervisor um zusätzliche Funktionen aufzublähen. Für den hochausfallsicheren Betrieb der Virtualisierungslösung spielt OCFS2 als Cluster-File-System eine zentrale Rolle. Unter anderem stellt es elementare Funktionen der Virtualisierung, beispielsweise „Copy on write“-Snapshots für Thin Clones oder das dynamische Allokieren von Speicherplatz für virtuelle Festplatten, bereit. Zudem ist der OCFS2-Layer für den vollautomatisierten HA-Betrieb (inklusive Heartbeat) verantwortlich.

Architektur

Die Architektur von Oracle VM besteht aus den folgenden Komponenten *(siehe Abbildung 1)*. Der Oracle-VM-Server wird innerhalb weniger Minuten Bare-Metal installiert. Neben der vertikalen Skalierung innerhalb eines Servers bietet ein Oracle-VM-Server-Pool, bestehend aus mehreren Oracle-VMServern, die Mög-