Continuous Integration für Oracle-DB und APEX

Peter Busch, Dominic Ketteltasche
MT AG
Balcke-Dürr-Allee 9, 40882 Ratingen

Schlüsselworte
Continuous Integration, Oracle-DB, APEX, HUDSON

Einleitung

Immer kürzer werdende Intervalle von Releases und Hotfixbearbeitung machen eine Versionierung von Datenbankobjekten und Anwendungen erforderlich. Um die entsprechenden Auslieferungen und Sicherheitsmechanismen effektiv nutzen zu können, werden unterschiedliche Werkzeuge benutzt. Continuous Integration (CI) ist der Oberbegriff, unter dem der gesamte Ablauf abgehandelt wird. Eine Automatisierung der Prozesse ergibt zusätzlich Laufzeitverbesserungen und eine einheitliche, projektweite Verfahrensweisen für die unterschiedlichen Umgebungen.
Das Projekt, über das im Zuge des Vortrages berichtet wird, hat vorgesehen, dass schon in der Entwicklungsphase fertige Teile an den Kunden geliefert werden. Eine Versionierung war somit unerlässlich. Das Einspielen der DB-Objekte sowie der Applikation musste so einfach wie möglich durchgeführt werden. Die Lösung dafür ist CI gewesen. Anhand des Projekts werden im Vortrag die verschiedenen Elemente von CI praktisch aufgezeigt. Als Tools sind TortoiseSVN für die Versionierung, der Mantis Bug Tracker und das webbasierte System Hudson eingesetzt worden. Das Projekt basiert auf Windows-Server und Oracle-Datenbanken.

CI für Oracle-DB und APEX im Praxiseinsatz

Die Datenintegrität stand im Vordergrund. Aus diesem Grund wurde die APEX-Applikation auf der Testumgebung entwickelt und wurde nur als Ganzes gesichert und ausgeliefert.

Die Architektur in dem Projekt hat folgendermaßen ausgesehen und besteht aktuell noch fort:

Die Test- und Entwicklungsumgebung sind auf unseren hauseignen Servern installiert.

Die Integrations- und Produktionsumgebung befindet sich beim Kunden vor Ort.

Eine Umgebung, die der Produktion entspricht, ist ebenfalls bei uns eingerichtet.

Für die Release-Arbeiten innerhalb unserer Firma wird TortoiseSVN für die Versionierung eingesetzt, ebenso wie Batch-Prozesse für die Verarbeitung und Hudson für die Automatisierung und das Überwachen. Dem Kunden wird jeweils eine ZIP-Datei zur Verfügung gestellt. In dieser sind alle Skripte enthalten, die für die Auslieferung notwendig sind. Über ein zentrales Startskript wird die Installation ausgeführt.

Um sicher zu stellen, dass alle Skripte für eine Auslieferung bereitgestellt werden, ist eine Deployment-Anwendung geschrieben worden. Diese enthält im Groben, die entwickelten Skripte und die Release-Nummer der jeweiligen Auslieferung. Parallel zu der Anwendung werden die Skripte mit dem entsprechenden Label in das Versionierungssystem eingecheckt.

Mit Hilfe der Deployment-Anwendung und dem Label, wird bei einer Auslieferung per Skript geprüft, ob alle Skripte in der Deployment-Anwendung mit dem erforderlichen Label enthalten sind. Diese müssen im Versionierungstool enthalten sein und umgekehrt.

Sofern ein Release ansteht, werden zwei Phasen ausgeführt:

Phase 1:

Die Entwicklung eines Release ist abgeschlossen, alle Skripte sind erstellt und mit dem Versions-Label getagt. Außerdem sind die Skripte in die Deployment-Anwendung eingegeben worden.

Der Job von Phase1 wird angestoßen. Nach dem Aufruf werden die folgenden Prüfungen durchlaufen:

- Stimmen die Anzahl und das Format der übergebenen Parameter?
- Sind mit den Zugangsdaten auch die erforderlichen Systeme zu erreichen?
- Sind die vorgegebenen Verzeichnisse zu erreichen?
- Sind alle Batch-Skripte für das Deployment vorhanden?

Für den Fall, dass die Prüfungen erfolgreich sind, wird ein Ausführungsskript mit den Skriptnamen der eingetragenen Skripte der Deployment-Anwendung erstellt. Die aufgeführten Skripte müssen, wie oben beschrieben, in dem Release-Verzeichnis enthalten sein und umgekehrt. Die Abhängigkeit der Skripte werden durch die Ordnerstruktur festgelegt – DB-Schema, DDL, DML, PLSQL, APEX-Seiten-Objekte (Views, Packages, …). Innerhalb dieser Verzeichnisse kann eine zusätzliche Abhängigkeit über die Eingabe innerhalb der Deployment-Umgebung gebildet werden.

Tritt hierbei kein Fehler auf, so ist die Phase1 erfolgreich gelaufen.

Bevor das Release an den Kunden ausgeliefert wird, müssen die Änderungen im Testsystem noch Prüfungen durchlaufen. Das Einspielen in das Testsystem dient dabei ebenfalls als Testlauf für das Deployment beim Kunden.

Danach wird der Job von Phase 2 angestoßen. Nach dem Aufruf werden die folgenden vier Prüfungen durchlaufen, welche so auch in Phase 1 vorgekommen sind (Die übrigen Prüfungen innerhalb der beiden Phasen sind unterschiedlich.):

- Stimmen die Anzahl und das Format der übergebenen Parameter?
- Sind mit den Zugangsdaten auch die erforderlichen Systeme zu erreichen?
- Sind die vorgegebenen Verzeichnisse zu erreichen?
- Sind alle Batch-Skripte für das Deployment vorhanden?

Ab hier wird der eigentliche Prozess ausgeführt.

- Der Start des Releases wird in eine LOG-Tabelle eingetragen.
- Der derzeitige Release-Stand der DB muss der Vorversion entsprechen.
- Die Skripte des Releases werden aufgerufen.
- Die einzelnen Skripte werden in der LOG-Tabelle protokolliert.
- Es werden drei Recompiles durchgeführt, um fehlerhafte DB-Objekt zu entfernen.
- Es findet eine Abschlussprüfung statt.
- Der neue Release-Stand wird in die Steuerungstabelle eingetragen.
- Die LOG-Dateien werden in ein zentrales Verzeichnis abgestellt.

Jedes Batch-Skript der Phase 2 erzeugt eine LOG-Datei. Diese Dateien werden am Ende des Laufs auf Fehler durchsucht und die Fehlertexte sowie die Logfile-Namen gesammelt ausgegeben. Gegebenenfalls erfolgt auch ein Abbruch der Phase 2.

Phase 1 und Phase 2 auf der Testumgebung finden bei uns vor Ort ohne Mitwirken des Kunden statt, so dass eine freie Toolauswahl möglich gewesen ist. Für die Automatisierung der Prozesse ist die Wahl auf Hudson gefallen. Um den Prozess bei uns und beim Kunden gleichzuhalten, werden über

Hudson immer nur Batch-Prozesse gestartet, so wie es der Kunde auch durchführen muss.
Es ist allerdings möglich über Parameter zum Beispiel die Release-Version oder die Zielumgebung vorzugeben.

Da unsere Versionierung unabhängig von der des Kunden ist, besteht für uns die Möglichkeit notwendige SVN-Arbeiten zu automatisieren. So werden die Tag-Verzeichnisse für die einzelnen Releases per Skript angelegt, vor dem Start der Phase 1 das Verzeichnis aktualisiert und die in Phase 2 erzeugten LOG-Dateien zentral ins SVN eingecheckt.
Aktuell gibt es noch Teile des Deployments, die bislang nicht in das CI mit aufgenommen wurden. In erster Linie sind es JavaScript Dateien, welche noch manuell auf die einzelnen Server zu übertragen sind. Im Prinzip betrifft es alle Objekte, die keine DB-Objekte sind oder zur APEX-Anwendung gehören. Es gehört zu den zukünftigen Zielen, diese ebenfalls mit in den automatisierten Prozess aufzunehmen.

Nachdem bei uns alle Tests erfolgreich durchgelaufen sind, wird das Release beim Kunden eingespielt. Hierfür stellt ein Batchprozess alle erforderlichen Skripte in einem ZIP-File zusammen, welches dem Kunden bereitgestellt wird.

Der Kunde entpackt das ZIP-File und führt den Prozess der Phase 2 aus. In dem ZIP-File sind alle notwendigen Account-Dateien für die Umgebungen des Kunden enthalten. Dieser braucht somit nur noch die Phase 2 mit der entsprechenden Account-Datei zu starten. Auf Grund der Tatsache, dass nur Batch-Skripte verwendet werden, findet beim Kunden der gleiche Prozess wie bei uns im Hause statt. Potentiell entstehende Fehler beim Release können somit nicht auf den übergebenen Skripten beruhen.

Nach Fertigstellung der APEX-Applikation auf der Entwicklungsumgebung, erfolgt der Export der Applikation mit Hilfe der JavaAPI APEXExport, welche in der APEX-Installation enthalten ist. Der Export stellt die Applikation als SQL-Skript in das File-System. Dieser Prozess wird über ein Batchskript ausgeführt und wie bei den DB-Objekten über Hudson angestoßen.

Für den Import der Applikation in das Testsystem werden die Parameter WORKSPACE_ID, APPLICATION_ID, SCHEMA, APPLICATION_ALIAS und der OFFSET entsprechend gesetzt. Über SQL-Plus erfolgt dann die Ausführung des SQL-Files der Applikation. Im Nachgang daran erfolgen das „UNAVAILABLE“-Setzen der veralteten Applikationen und das “AVAILABLE“-Setzen der aktuellen Version. Somit ist ein Start der Applikation mit dem gleichen Link immer dann möglich, wenn die APPLICATION_ID oder der APPLICATION_ALIAS beibehalten wird.

Da das Export-Skript der Anwendung nicht verändert wird, ist ein Einspielen mit der Import-Funktion des Applikation Builders in andere Umgebungen möglich.
Sobald die Applikation bereit zur Auslieferung ist, wird das SQL-File ins SVN eingecheckt, entsprechend der Version getagt und in das ZIP-File der DB-Skripte mit aufgenommen.

Durch die Anforderungen des Kunden sind im Laufe des Projektes zwei weitere Umgebungen bei uns im Hause notwendig geworden. Das hat es erforderlich gemacht, die DB-Schematas per datapump oder auch Skripte als Releases einzuspielen. An dieser Stelle ist die Steuerung über Hudson sehr hilfreich gewesen. Über Variablen und Auswahlboxen können die einzelnen Umgebungen bedient werden. Ein zentrales Verzeichnis, das von allen Servern erreicht werden kann, dient als Drehscheibe für alle Daten, bei denen keine Versionierung notwendig ist.

Mittlerweile werden vier Umgebungen für das Projekt vorgehalten, bei denen es mittels Hudson möglich ist, die Applikationen von jeder Umgebung aus zu exportieren und in jede zu importieren.

Daraus ergeben sich die folgenden Vorteile. Innerhalb unserer eigenen Umgebungen sind wir variabler und können schneller auf Kundenwünsche reagieren. Wenn z.B. ein Hotfix neben den normalen Releases notwendig ist, kann ohne großen Aufwand eine aktuelle Umgebung aufgebaut werden, die auch dem Produktionsstand entspricht. Die Laufzeiten der Skripte haben sich durch den Aufruf über Hudson deutlich reduziert.

Da der Kunde für seine Installation die gleichen Skripte verwendet wie wir, profitiert er in Bezug auf Sicherheit von den laufenden Änderungen (z.B. erweiterte Fehlerroutinen).

Im Bug-Tracker Mantis werden alle gemeldeten Änderungen eingetragen und mit der Release-Nummer gekennzeichnet. Über dessen Auswertungsmöglichkeiten ist der Stand der Release-Arbeiten schnell abzulesen. Die eingetragenen Skripte in der Deployment-Anwendung sind mit den entsprechenden Mantis-Nummern versehen. So ist zu erkennen, welche Skripte für einen Mantis-Eintrag erstellt wurden und wer für dieses Skript verantwortlich ist.

Im Laufe des Projekts musste die Vollauslieferung auf ein inkrementelles Verfahren umgestellt werden. Dieses Verfahren hat eine besondere Herausforderung dargestellt, da die erzeugten Skripte von da ab restartfähig sein mussten. So mussten von da an bei DDL-Skripten überprüft werden, ob im Repository die Änderungen bereits enthalten sind. Bei DML-Skripten kam die Prüfung hinzu, ob die Änderungen im Datenbankschema schon vorgenommen wurden. Für Sonderfälle musste ein Exception-Handling mit dem Abfangen des Fehlers eingebaut werden.

Fazit und Ausblick

Als Fazit kann festgehalten werden, dass bereits viele Optimierungen geschehen sind, sich CI aber trotzdem immer noch im Wandel befindet, sei es durch eigene Ideen oder durch Anforderungen vom Kunden.

Für die Zukunft sind somit noch einige Erweiterungen denkbar bzw. wünschenswert:

- Hudson bzw. allgemein – die Installation von Dumps bzw. der Rücksprung auf die Initialinstallation
- die Einbindung der Installation der Anwendung als Hudson Job
- die Verbesserung der Handhabung von Abhängigkeiten bei Installationsskripten
- die Anpassung der Installationsreihenfolge primär über die Vorgängerbeziehung und sekundär über die Vorgabe der Ordnerstruktur (DDL – PLSQL – APEX – DML)
- die inkrementelle Auslieferung der APEX-Anwendung
- allgemein ein höherer Grad der Automatisierung.

Kontaktadresse:

Peter Busch, Dominic Ketteltasche
MT AG
Balcke-Dürr-Allee 9
D-40882 Ratingen

Telefon:	+49 (0) 2102-30961 0
Fax:	+49 (0) 2102-30961 101
E-Mail	peter.busch@mt-ag.com, dominic.ketteltasche@mt-ag.com
Internet:	www.mt-ag.com