# Continuous Delivery von APEX-Applikationen

**Christian Klingbacher**
**Altran Concept Tech GmbH**
**Graz**

**Schlüsselworte**
Oracle APEX, Continuous Delivery, Docker, Git, Version Control

## Einleitung

Mit dem Rapid Application Development Tool Oracle APEX können professionelle Business-Applikationen in nur sehr kurzer Zeit erstellt und um neue Features erweitert werden. Continuous Delivery bietet dem Entwickler eine Möglichkeit, Änderungen an komplexer Software automatisiert und zuverlässig dem Anwender zur Verfügung zu stellen.
Im Folgenden wird beschrieben, wie APEX Applikationen mit nur einem Klick automatisch auf unterschiedliche Instanzen ausgerollt werden können. Unter Verwendung von Git, Apache Maven, sowie Atlassians´ Bitbucket wird eine vollständige Deployment Pipeline mit nur wenigen Klicks erzeugt und getestet. Bis zum Ende erhält der Leser zudem eine vollständige Schritt-für-Schritt-Anleitung zur Erstellung der ersten eigenen Pipeline[1]. Damit sollte einem automatischem Deployment der eigenen APEX Applikationen in Zukunft nichts mehr im Wege stehen!

## Continuous Integration vs. Continuous Delivery vs. Continuous Deployment

Continuous Delivery oder Continuous Deployment „bezeichnet eine Sammlung von Techniken, Prozessen und Werkzeugen, die den Softwareauslieferungsprozess verbessern“[2] – Continuous Integration stellt dabei einen Teilbereich der beiden vorangegangenen Begriffe dar.

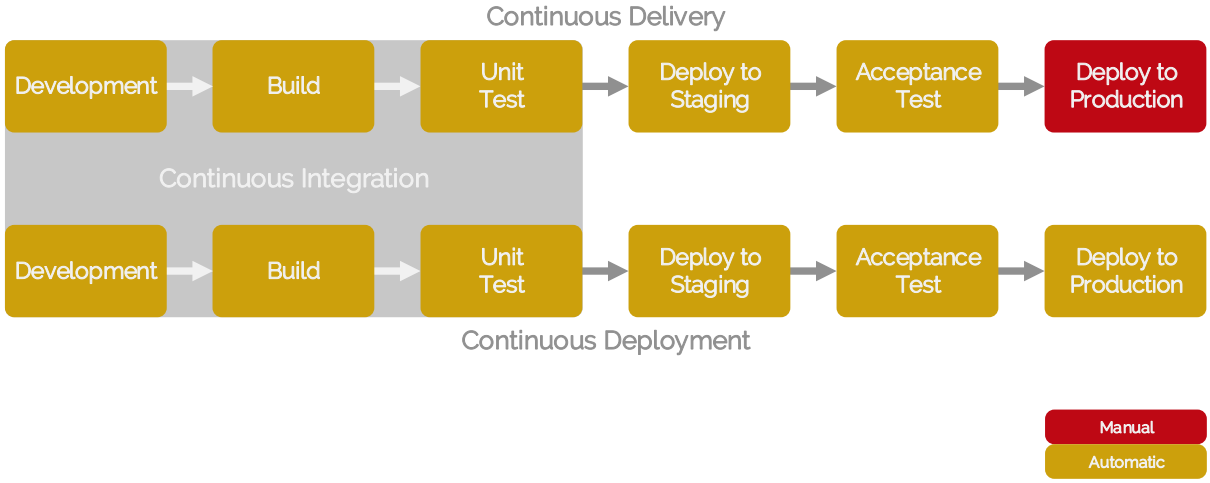


*Abb. 1: Unterschiede Continuous Integration/Delivery/Deployment*[3]

---

[1] Die Erstellung automatischer Tests ist nicht Teil der Präsentation.

[2] https://de.wikipedia.org/wiki/Continuous_Delivery

[3] Vgl.: https://www.atlassian.com/continuous-delivery/ci-vs-ci-vs-cd

Wie aus Abbildung 1 ersichtlich, umfasst der Begriff Continuous Integration die Entwicklung und Erstellung von Software, sowie das automatisierte Testen des Source Codes mit Hilfe von Unit Tests. Ist es möglich, dass von einer geänderten Software ein Build erstellt wird und alle Tests erfolgreich verlaufen, so kann diese automatisch auf einer eigenen Instanz für User Acceptance Test bereitgestellt werden. Verlaufen auch diese Tests durch die User positiv, so kann ein Deployment auf die Produktivumgebung stattfinden. Der Unterschied zwischen Continuous Delivery und Contiuous Deployment ist hierbei, dass der Prozess des Deployments entweder manuell oder automatisch gestartet wird.
Der gesamte Prozess vom Development bis hin zum Deployment auf die Produktivumgebung wird dabei Release-Pipeline genannt.

**Source Control**
Um Continuous Delivery in einem Softwareentwicklungsprozess umsetzen zu können, ist die Einführung einer Versionskontrolle für Source Code zwar nicht unbedingt notwendig, aber auf jeden Fall sehr hilfreich. Abbildung 2 zeigt, wie eine Versionskontrolle für Oracle APEX Projekte durch Einsatz von Git implementiert werden kann.

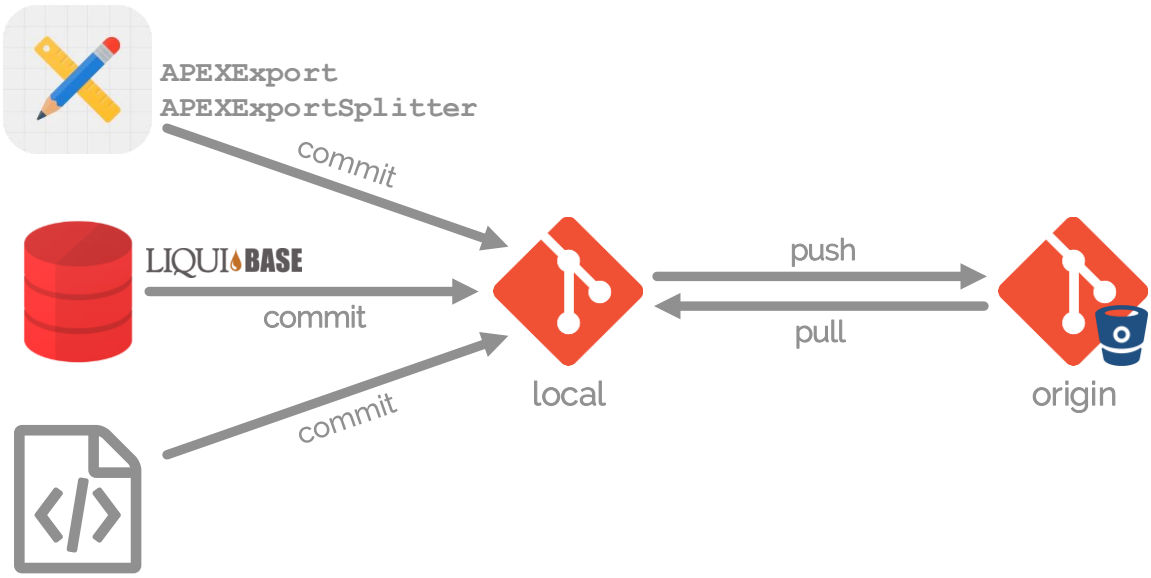


*Abb. 2: Implementierung einer Versionskontrolle mit APEX Applikationen*

Um APEX Applikationen automatisiert versionieren zu können, ist ein Export der Applikationen mit Hilfe der Java-Klassen APEXExport und APEXExportSplitter durchzuführen[4] - dies gewährleistet auch, dass alle Änderungen nachvollziehbar sind. Die Datenbankobjekte werden in diesem Beispiel mit Hilfe des Open-Source-Tools *Liquibase* versioniert – damit kann das Erstellen und Ändern von Tabellen, View, Packages, usw. sehr genau gesteuert werden.[5] Sämtliche Dateien, die an anderer Stelle im Projekt benötigt werden (z.B. CSS, JavaScript), können ohne weitere Vorbereitungen im Projektverzeichnis abgelegt werden.

---

[4] Für mehr Informationen siehe:
https://apex.oracle.com/pls/apex/germancommunities/apexcommunity/tipp/4901/index.html

[5] Eine detaillierte Beschreibung findet man unter http://www.liquibase.org/

Nachdem alle Dateien im lokalen Repository commitet bzw. gemergt wurden, werden die Änderungen in das Remote-Repository auf Bitbucket gepusht. Dies ist auch der Zeitpunkt an dem die Pipeline startet.

**Setup der Pipeline**
Um die Pipeline-Funktion auf Bitbucket nutzen zu können, braucht diese nur in den Einstellungen des Repositorys aktiviert zu werden – um damit auch APEX-Applikationen deployen zu können, sind allerdings noch ein paar zusätzliche Schritte notwendig. Alle Schritte für eine erfolgreiche Konfiguration werden im Folgenden detailliert beschrieben:

1. Benutzerkonto erstellen

Um Bitbucket-Pipelines nutzen zu können, muss ein Benutzerkonto auf https://bitbucket.org/ erstellt werden. Die Nutzung des Dienstes ist für kleine Teams bis zu fünf Personen kostenlos – hierbei sind auch 50 Build-Minuten pro Monat inkludiert. Details zu den Funktionen und Preisen findet man unter https://de.atlassian.com/software/bitbucket.

2. Repository erstellen

Nachdem man sich bei Bitbucket angemeldet hat, kann ein neues Repository erstellt werden. Dazu klickt man auf der Seite links oben auf + und anschließend auf *Repository*. Nachdem alle Einstellungen getroffen wurden, wird das Repository durch einen Klick auf *Create repository* erstellt.

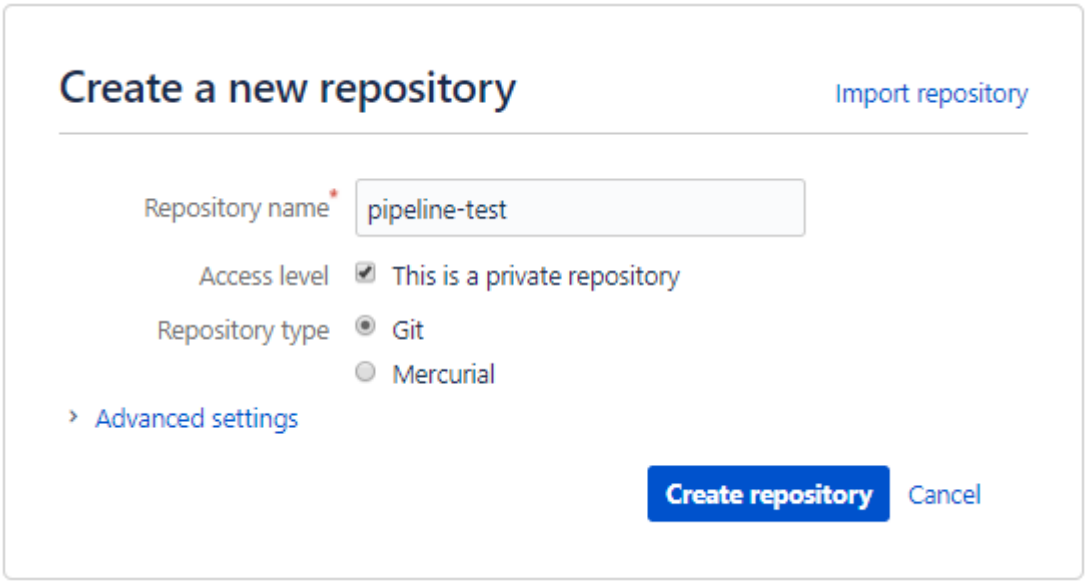


*Abb. 3: Repository erstellen*

3. Pipeline aktivieren

Im nächsten Schritt muss die Pipeline-Funktion in den Settings aktiviert werden. Hierzu öffnet man die *Settings* des Repositorys und darin die *Pipelines settings*, in welchen unter der Option *Enable Pipelines* die Aktivierung durchzuführen ist.

4. Pipeline konfigurieren

Im letzten allgemeinen Task muss die Datei bitbucket-pipelines.yml – welche sämtliche Pipeline-Schritte enthält – konfiguriert werden. Hierzu klickt man im Anschluss an die Aktivierung einfach auf den Button *Configure bitbucket-pipelines.yml* (siehe Abbildung 4).

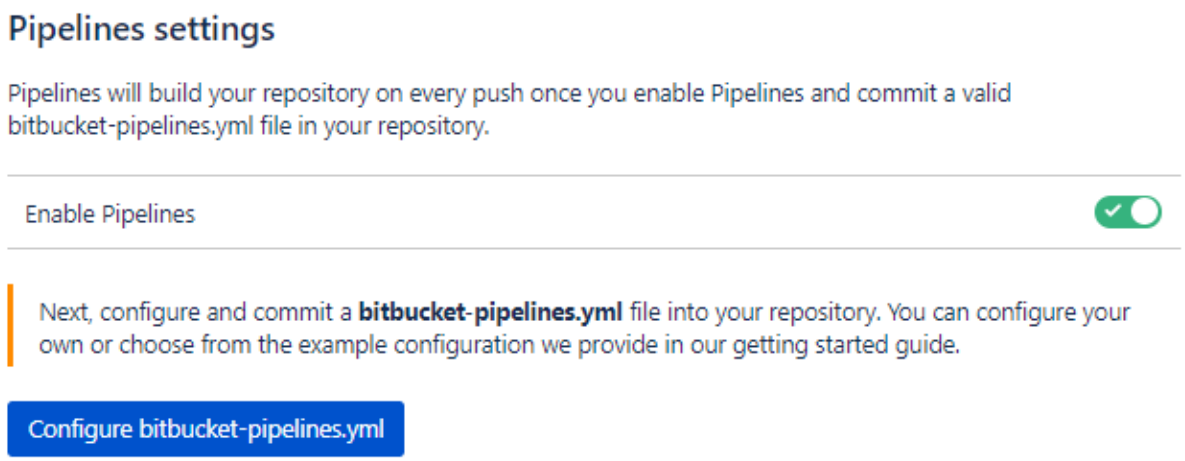


*Abb. 4: Pipelines settings*

Nun braucht nur mehr *Other* als Language Template ausgewählt und auf *Commit file* geklickt werden – fertig ist die (erste Version) der Pipeline. Auf der anschließenden Seite wird die Pipeline #1 angezeigt und automatisch für einen ersten Run gestartet. Da diese zu diesem Zeitpunkt nur einen Text ausgibt, läuft sie auch erfolgreich bis zum Ende.

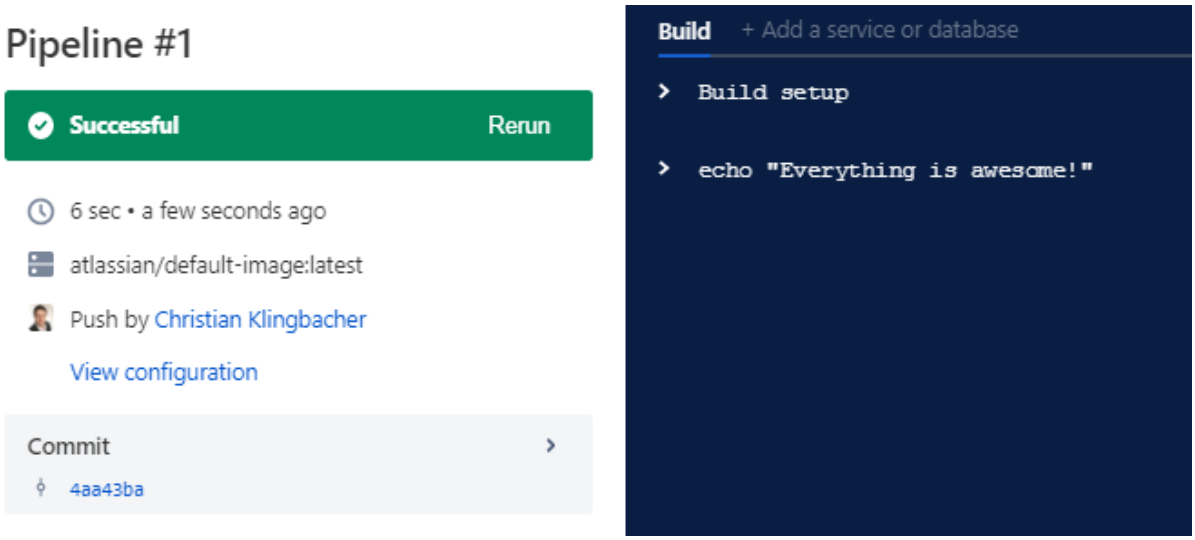


*Abb. 5: Erster Lauf der soeben erzeugten Pipeline*

Mit *View configuration* werden nun die Pipeline erneut geöffnet und sämtliche Schritte für das Deployment einer APEX Applikation definiert (siehe Präsentation).

5. Maven Project Object Model erstellen

Für das Deployment auf einen Zielserver wird Apache Maven eingesetzt – hierfür steht sowohl für Liquibase, als auch für APEX ein entsprechendes Plugin zur Verfügung.
Beim Start der Pipeline wird von Bitbucket ein Docker-Container erstellt, wobei das hierfür verwendete Standard-Image bereits Maven beinhaltet. Im Anschluss werden alle Dateien aus dem Git-Repository in den neuen Container geladen, wofür es notwendig ist, dass die entsprechende Maven-Projektstruktur, sowie das Project Object Model[6] zur Verfügung stehen.
Das Project Object Model ist eine XML-Datei im Root-Verzeichnis des Repositorys mit dem Namen *pom.xml*. Wie auch in der Präsentation ersichtlich, ruft diese Datei zwei Plugins auf – eines für das Ausführen von Liquibase und ein weiters für das Deployen der APEX Applikationen. Für die Ausführung des jeweiligen Plugins werden einfach die benötigten Parameter angegeben – alle Informationen zum Verbindungsaufbau werden von der Pipeline beim Starten von Maven übergeben.

6. Environment Variables erstellen

Um Benutzernamen und Passwörter nicht im Klartext in Konfigurationsdateien speichern zu müssen, bietet Bitbucket die Möglichkeit *Environment Variables* anzulegen. Diese können Informationen sowohl in Klartext, als auch verschlüsselt speichern. Um automatisiert ein Deployment auf unterschiedliche Server durchführen zu können, sollten je Instanz zumindest folgende Variablen angelegt werden:

- *BRANCHNAME*_CONNECTION_STRING
- *BRANCHNAME*_URL
- *BRANCHNAME*_USERNAME
- *BRANCHNAME*_PASSWORD

Voraussetzung hierfür ist allerdings, dass je Zielserver ein eigener Branch existiert (z.B. Development, Test, Acceptance, Production). Will man die Software auf mehrere Server deployen, so wäre es sinnvoll, einen Master-Branch zu erstellen, welcher mit keinem Server „verbunden“ ist – auf diesem wird jede Version der Software mit der entsprechenden Nummer getaggt. Weiters kann für jeden Produktivserver ein eigener Branch angelegt werden (z.B. Kunde_A, Kunde_B, Kunde_C). Will man nun beispielsweise Version 3 der Software an Kunden B ausliefern, so braucht man nur jenes Commit mit dem entsprechenden Tag am Master-Branch in den Branch Kunde_B mergen.

7. Hilfsprogramme zur Verfügung stellen

Da sowohl in der Pipeline, als auch im Project Object Model, verschiedene Hilfsprogramme zur Ausführung des Deployments aufgerufen werden, muss sichergestellt sein, dass diese auch unter dem jeweiligen Pfad verfügbar sind. Hierbei handelt es sich um folgende Programme:

- APEXExport und APEXExportSplitter: Diese Java-Klassen werden für den automatischen Export von APEX Applikationen benötigt und sind in jeder APEX-Installation vorhanden.
- Oracle Instant Client mit SQL*Plus[7]: SQL*Plus wird für das Ausführen von PLSQL-Code auf einer Oracle Datenbank benötigt. Da im geladenen Docker Image kein Oracle-Client zur Verfügung steht, ist die Installation des Oracle Instant Clients inkl. SQL*Plus notwendig.
- OJDBC[8]: Treiber für den Verbindungsaufbau mit einer Oracle Datenbank.

---

[6] Siehe auch https://maven.apache.org/pom.html

[7] http://www.oracle.com/technetwork/database/features/instant-client/index-097480.html

- Oracle APEX Maven Plugin[9]: Da dieses Plugin nicht im Maven Plugin Repository gelistet ist, ist es auch nicht möglich, dieses beim Kompilieren automatisch zu laden. Aus diesem Grund müssen die benötigten Dateien manuell hinzugefügt werden.

8. Pipeline starten

Wurde die Konfiguration wie beschrieben durchgeführt, so startet die Pipeline bei jedem Commit oder Merge und durchläuft alle Schritte für den jeweiligen Branch – dabei werden alle Änderungen des Datenbankschemas, sowie alle APEX Applikationen auf den zugehörigen Zielserver ausgeliefert.

**Literatur**

Nick Buytaert (2015): Lifecycle Management, in: John Scott, Nick Buytaert, Karen Cannell, Martin D'Souza, Doug Gault, Dimitri Gielis, Roel Hartman, Denes Kubicek, Raj Mattamal, Dan McGhan, Francis Mignault, Tom Petrus, Jorge Rimblas and Christoph Ruepprich, Expert Oracle Application Express, Apress

Eberhad Wolff, Continuous Delivery: Der pragmatische Weg, 2. Auflage (2016), dpunkt.verlag

**Kontaktadresse:**
Christian Klingbacher
Altran Concept Tech GmbH
Engelgasse 3-5
A-8010 Graz

Telefon: +43 664 80935192
Fax: +49 316 231123-2303
E-Mail christian.klingbacher@altran .com
Internet: www.altran.at

[8] http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html

[9] https://github.com/nbuytaert1/orclapex-maven-plugin