Produktionsdeployments im DWH

Cornel Brücher
SHS VIVEON GmbH
Düsseldorf

Schlüsselworte

Deployment, Downtime

Einleitung

Produktionsdeployments im DWH (und anderswo) rangieren nicht gerade an der Spitze der Beliebtheitsskala. Die Entwickler gehen vorsichtshalber in Deckung oder sind schon weg, die Fachabteilungen möchten nicht einen Tag auf ihr System verzichten, die Deploymentverantwortlichen hätten gerne alles einen Monat im Voraus eingefroren, und die Systemverantwortlichen hätten gerne gar kein Deployment.

In diesem Vortrag betrachten wir den Spagat zwischen Risiko- und Aufwandsminimierung durch zyklische Deployments auf der einen und maximaler Kundenorientierung/Time-to-Market durch immediate Deployments auf der anderen Seite. Wir gehen dabei auf die Voraussetzungen, die organisatorischen und personellen Unterschiede sowie die Möglichkeiten des Outsourcings bei beiden Vorgehensweisen ein.

Zyklische Produktionsdeployments

Egal wie lange man es herauszögert, irgendwann ist die Entwicklung neuer Schnittstellen, ETL-Prozeduren, Verarbeitungslogik und DataMarts abgeschlossen, und es nähert sich der Tag der Wahrheit. Alle Beteiligten des DWH-Projekts wissen, dass trotz gründlicher Tests und Testdeployments immer etwas schiefgehen kann – bis hin zum Ausfall. Die bewährteste Taktik zur Risikominimierung ist, den drohenden ungeplanten Ausfall durch einen geplanten Ausfall zu ersetzen. Dazu wird dann pro Monat maximal ein Deployment-Termin festgelegt. Dieser liegt dann zwischen den „dann bitte nicht, da brauchen wir die Reports“-Tagen der Fachabteilungen und den „an diesem Tag müssen wir Betriebssystem-Patches einspielen“-Tagen der Systemadministratoren.

Das Verfahren der zyklischen Deployments hat sich in großen Anwendungen wie z.B. CRM-Systemen bewährt. Mit einem hohen Grad an Formalisierung ist genau festgelegt, bis wann vor dem Termin alle „Deliverables“ (Code, Dokumentation, Testdokumentation, Deploymentscripte etc.) vorliegen müssen, damit der Deploymentverantwortliche sie noch prüfen kann. Von außen betrachtet ist eine Woche Code Freeze vor dem Deployment recht viel, aber bei der Menge an laufenden Projekten in einem großen DWH ist eine Woche als Vorbereitungszeit schon knapp. Wenn Sie den Deploymentverantwortlichen nach dem idealen Zeitraum aus seiner Sicht fragen, erhalten Sie als Antwort wahrscheinlich eine Größenordnung von 2-4 Wochen. Fragen Sie einen Entwickler oder Projektleiter, so hat dieser dafür überhaupt kein Verständnis, und hält eine Abgabe am Tag vor dem Deployment für ausreichend. Was könnte man in dieser Woche noch alles programmieren. Manchem Entwickler fällt genau in dieser Woche auf, dass er noch dringend etwas ändern muss. Daher ist es auch schon Tradition, dass nach Ablauf der Deadline mindestens ein Teilprojekt versucht, sich noch in das aktuelle Deployment hineinzumogeln oder hineinzueskalieren.

Am Tag des Deployments werden alle Schnittstellen stillgelegt, alle Datenbank-Jobs gestoppt, und die vorbereiteten Deploymentscripte eingespielt. Bei einem großen DWH kann das ein langer Tag werden. Anschließend werden die Jobs wieder in Betrieb genommen, die in der Zwischenzeit aufgelaufenen Daten an den Schnittstellen abgearbeitet, und das System läuft wieder an.

Voraussetzungen

Die Entscheidungen der Qualitätssicherung und der Deploymentverantwortlichen müssen unanfechtbar sein. Wenn es gelingt, das Deployment von Projekten mit fehlenden Voraussetzungen wie verspäteter Lieferung, mangelnder Dokumentation oder Qualitätssicherung zu erzwingen, gefährden diese Projekte das gesamte Deployment. (z.B. ungeplante Langläufer oder Abbrüche bei den Deploymentscripts), und die rechtzeitige Wiederinbetriebnahme des ganzen Systems. Erfolgreiche Testdeployments auf einem Produktionsspiegel sind Pflicht. Selbstverständlich sind eigentlich die Verfügbarkeit der aktuellen Produktionsversion in der Versionsverwaltung und (durch die Entwickler) vorbereitete Rollback-Scripte, um ein nicht mehr zu rettendes Deployment schnell zurücksetzen zu können.

Arbeitsteilung

Typisch für zyklische Deployments ist die personelle und zeitliche Trennung von Entwicklung, Qualitätssicherung, Deployment und Wartung/Troubleshooting. Der fachliche Hintergrund der einzelnen Projekte bleibt auf die Köpfe der an Analyse, Design und Entwicklung beteiligten Personen beschränkt; die genannten Tätigkeiten sind zeitlich strikt nacheinander angeordnet. Das Wartungsteam lernt den Quellcode erst im Fehlerfall kennen – falls überhaupt. Der Know-how-Träger über alle Phasen hinweg ist nur die Dokumentation. Die Beteiligten müssen die Folgen eventueller Fehler und Nachlässigkeiten nicht selbst tragen. Beispielsweise gibt die Qualitätssicherung ein Projekt für das Deployment frei, ist aber an Deployment und Wartung nicht beteiligt. Was zuvor schiefgeht, darf das Wartungsteam als unteres Ende der Nahrungskette wieder geradebiegen.

Outsourcing

Da hier ohnehin kein fachliches Know-how vorgehalten wird, lassen sich Analyse, Design, Entwicklung und Test leicht outsourcen. Qualitätssicherung über den Test hinaus (Programmierstil, Kommentare, Umgang mit Systemressourcen), Deployment und Wartung hingegen sollten im Hause bleiben, damit sich diese Stellen gegenüber den Outsourcing-Dienstleistern durchsetzen können.

Vorteile zyklischer Deployments:

- Planbarkeit: Der geplante Ausfall verursacht keinen Ärger mit den Fachabteilungen, wenn er im Zeitrahmen bleibt. Findet das Deployment an einem Freitag statt, hat man das Wochenende für Nachkorrekturen und Nachverarbeitung, und zu Wochenbeginn sind die Zahlen wieder aktuell.
- Aufwandsminimierung:
    - Geringer Aufwand für das Deployment selbst; das Deployment kann (und sollte) nur von ein bis zwei Personen durchgeführt werden.
    - Das System wird als Ganzes einmal für alle betroffenen Projekte sauber herunter- und wieder hochgefahren.
- Risikominimierung:

    - Keine Blockade des Deployments durch laufende Objekte.
    - Kein ungeplanter Ausfall durch Produktionseingriffe
- Kleines Wartungsteam: Nur im Fehlerfall Einarbeitung in Code und Dokumentation

Nachteile zyklischer Deployments:

- Black Box Deployment: Die fachlichen Hintergründe des Deployments kennen nur die Designer und Entwickler. Code-Merges finden ohne Kenntnis der Fachlichkeit statt. Die neuen Zwischen- und Endergebnisse auf dem produktiven DWH können nur mit Hilfe der Fachabteilung überprüft werden.
- Hotfixes: Sind Nacharbeiten an den neuen oder geänderten Objekten erforderlich, beispielsweise Performancetuning oder Bugfixes, so können diese bei genügend Argumentationsdruck als Hotfixes eingespielt werden, oder müssen bis zum nächsten Deploymenttermin warten. Diese Möglichkeit lockt natürlich auch Change Requests an, die dann als Hotfix getarnt werden. Wenn man diese Entwicklung zulässt, hat man irgendwann mehr Hotfix-Deployments als reguläre Deployment-Objekte. (Lässt man sie nicht zu, gilt man als unflexibel.)
- Datenstau: Während des Deployments werden die Daten aller Schnittstellen aufgestaut, auch wenn nur eine vom Deployment betroffen ist. Die gleichzeitige Nachverarbeitung aller Daten eines Tages belastet das System und kann zu weiteren Verzögerungen führen.
- Leerlauf in Projekten: Bei monatlichem Deployment mit einer Woche Code-Freeze vorab müssen Projekte nach Fertigstellung 1-5 Wochen auf das Deployment warten.
- Know-how-Verlust: Entwickler können nicht im bezahlten Leerlauf auf das Deployment warten. Sie arbeiten schon in einem neuen Teilprojekt oder sind - wenn extern - vielleicht schon bei einem neuen Kunden. Damit stehen sie für Rückfragen während des Deployments nur eingeschränkt oder nicht mehr zur Verfügung.
- Schwierige Entstörung: Bei späteren Störungen steht niemand zur Verfügung, der den Quellcode bereits kennt. Fehler müssen anhand der Dokumentation und in unbekanntem Quellcode gesucht und behoben werden. Eine Beurteilung fachlicher Fehlerzustände durch das Wartungsteam ist nicht praktikabel.

Die beiden letztgenannten Punkte könnten theoretisch entschärft werden, wenn die Verfügbarkeit des Entwicklers budgettechnisch und vertraglich abgesichert wäre. Praktikabel ist das nicht, denn die Beseitigung von Störungen darf nicht von der Verfügbarkeit des Entwicklers abhängen. Davon abgesehen haben Entwickler auf der Produktion nichts zu suchen (Datenschutz!).

Spielen wir das mal für ein Projekt zeitlich durch, und gehen von einem Deployment-Termin am 26.4.2013 aus. Der späteste Abgabetermin für Code und vollständige Dokumentation zu Code, Test und Betrieb wäre dann am 19.4.2013. Wer diesen Termin um einen Tag verpasst, wäre dann erst am 24.5. auf der Produktion. Ein direkt anschließender, noch so kleiner Change-Request, von einem ehrlichen Projektleiter nicht als Hotfix getarnt, benötigt einen weiteren Monat, bis er produktiv umgesetzt ist. Beispielsweise ein Hinzufügen einer Gruppierungsspalte in einem SELECT-Statement. 15 Minuten für die Änderung im Code, 1 Tag für den Test, 2 Tage für die Anpassung der Dokumentation. Das ist schon ein verhältnismäßig hoher administrativer Aufwand, aber von einsehbarer Notwendigkeit. Danach noch Wochen auf das Deployment zu warten, ist für alle Beteiligten eine Zumutung. Das erinnert an COBOL-Programmierung und Deployment mit Lochkarten.

Dabei haben wir bis jetzt noch ignoriert, dass Projekte schon vor dem Deployment aus dem Zeitplan laufen können. Bei Verzögerungen, die in Wochen oder gar Monaten gemessen werden, fangen die Fachabteilungen an, Zwischenlösungen mit Excel oder Access zu basteln. Für langlebige Provisorien dieser Art gibt es einen Namen: Schatten-IT. Das kann so weit gehen, dass die DWH-Reports zwar erzeugt, aber nicht mehr gelesen werden.

Was spricht eigentlich dagegen, Projekte im DWH sofort nach Fertigstellung auf Produktion einzuspielen? Klassische Gegenargumente sind:

Aussage: „Die Auswirkungen sind nicht zuverlässig abschätzbar!“
Bedeutung: Wir haben die Dokumentation nicht im Griff.

Aussage: „Die Architektur lässt das nicht zu!“
Bedeutung: Das DWH ist historisch gewachsen. Es hat keine geplante Architektur.

Aussage: „Unmöglicher Aufwand!“
Bedeutung: Es ist mehr Arbeit.

Last Line of Defense
Aussage: „Das haben wir noch nie so gemacht.“
Bedeutung: Keine Änderung vor der Rente!

Immediate Deployments

Sind die Entwicklungen zu einem DWH-Projekt abgeschlossen, kann es bei kundenorientierter Einstellung eigentlich nur einen Deployment-Termin geben: Sofort. Jedes Mal das ganze DWH anzuhalten, ist unmöglich, also kann das nur im laufenden Betrieb stattfinden. Es werden nur rechtzeitig vorher die Verarbeitungsketten deaktiviert, die vom Deployment betroffen sind.

Das ist nicht so schlimm, wie es sich anhört. Hier fliegen keine echten Zahnräder aus dem Getriebe. Es ist alles nur virtuell…

Voraussetzungen

Qualität: Während zyklische Deployments auch in DWHs mit zweifelhafter Architektur und unvollständiger Dokumentation funktionieren, verlangt ein Deployment im laufenden Betrieb eine saubere Architektur (nicht zuletzt uneingeschränkt restartfähige Programme), aktuelle und vollständige Dokumentation, und die genaue Kenntnis aller Abhängigkeiten. Es reicht nicht, nach Checkliste vorzugehen und die Dokumentation danebenzulegen. Das ist kein Black-Box-Deployment. Man muss den Code, der eingespielt wird, kennen und beurteilen können. An erfolgreichen Testdeployments auf einem Produktionsspiegel, der Verfügbarkeit der aktuellen Produktionsversion in der Versionsverwaltung und vorbereiteten Rollback-Scripts führt hier kein Weg vorbei.

Personelle Struktur

Die Arbeit muss hier nicht primär nach Tätigkeiten, sondern nach Projekten und Fachgebieten aufgeteilt werden. Die klassische Arbeitsteilung funktioniert hier nicht mehr und endet bei den Entwicklern und Testern.

Neu ist die Rolle des Data Warehouse Engineers, der

- die Entwicklung begleitet und technisch unterstützt,
- die Qualitätssicherung des Programmcodes übernimmt,
- die Deploymentfreigabe erteilt,
- das Deployment durchführt,
- die Wartung übernimmt
- und das Projekt sowohl technisch als auch fachlich weiter betreut.

Der DWH Engineer wird so zum zentralen technischen und fachlichen Know-How-Träger des Projekts. Er wird ein Projekt nur zum Deployment freigeben, wenn er es qualitativ verantworten kann. (Ist er hier zu nachlässig, muss er es später selbst ausbaden.) Wenn später im laufenden Betrieb Störungen auftreten, kennt er bereits die Hintergründe und kann die Fehler schneller beheben. Im Gegensatz zum klassischen Wartungsteam kann er auch fachliche Ursachen bewerten.
Für jedes Projekt/fachliches Thema muss ein DWH Engineer namentlich und ein zweiter stellvertretend verantwortlich sein.

Outsourcing

Entwicklung und Test lassen sich leichter (und weiter weg) outsourcen, da die DWH Engineers für die Sicherstellung der Qualität sorgen. Das DWH Engineering muss im Hause bleiben, da das Know-how in den Köpfen erhalten bleiben muss.

Vorteile von immediate Deployments:

- Keine Verzögerungen durch Wartezeiten bis zum Deployment
- Know-how ist in der Dokumentation und den Köpfen der DWH Engineers vorhanden
- Schnelle Entstörung. (Wenn die Zahlen zum Monatsende nicht stimmen, hat man nicht bis zum nächsten Monat Zeit, das Ticket herumzureichen)
- Begrenzter Datenstau →Schnellere Nachverarbeitung, keine Überlastung des Gesamtsystems
- Kompetente Ansprechpartner für Fachabteilungen im DWH-Team

Nachteile immediate Deployments:

- Ungeplanter Kollateralschaden ist möglich
  - aufgrund fehlerhafter Dokumentation.
  - aufgrund eines Fehlers beim Deployment.
- Höherer Personalbedarf: Da ein DWH Engineer nur eine begrenzte Anzahl von Projekten gleichzeitig betreuen kann, ist das Team größer als QS + Deployment + Wartung beim alten Verfahren.
- Das Know-how in den Köpfen kann zu Nachlässigkeiten bei der Dokumentation führen.
- Das System ist personenabhängiger.

Fazit

Ein Data Warehouse muss aktuell sein, um seine Anforderungen erfüllen zu können. Es muss flexibel sein, um neue Anforderungen schnell umsetzen zu können. Antiquierte Verfahren wie zyklische Deployments behindern diese Ziele.

Die beliebige Austauschbarkeit von Personal im DWH ist ohnehin nur eine Illusion. Dokumentation kann Know-how nicht vollständig ersetzen.

You get what you pay for! Qualität und Aktualität haben ihren Preis.

Kontaktadresse:
Cornel Brücher
SHS VIVEON GmbH
Bennigsen-Platz 1
40474 Düsseldorf

Telefon:	+49 (0) 211 / 91 31 33 - 0
Fax:	+49 (0) 211 / 91 31 33 - 10
E-Mail	Cornel.Bruecher@SHS-VIVEON.com
Internet:	www.SHS-VIVEON.com