Alles fließt

Continuous Integration in der Cloud mit Hudson

Arnd Kleinbeck

Hudson hat als Continuous-Integration-Server (CI-Server) das Ziel, Softwareentwicklern möglichst kurzfristig Feedback über die Konsequenzen ihrer Code-Änderungen im Hinblick auf die Integration ins Gesamtsystem zu geben. Was aber passiert, wenn durch die schiere Größe des Projekts oder durch die Notwendigkeit unterschiedlicher Zielplattformen der Build- und Testprozess zum Flaschenhals wird? Hier bietet sich die Nutzung von Cloud-Diensten an, um auf kostentransparente Weise drastische Performance-Steigerungen zu ermöglichen oder aber um bei Bedarf dynamisch in die Cloud zu skalieren.

Keep Your Build Fast

„Keep your Build fast!" ist eines der wesentlichen Prinzipien des Continuous-Integration-Prozesses [Fowl06], der durch CI-Tools wie Hudson unterstützt wird.

Der CI-Prozess verlangt, dass Veränderungen an einem Softwaresystem durch eine Abfolge kleiner Einzelschritte realisiert werden sollen, deren Umsetzung jeweils einen lauffähigen Zwischenstand ergibt. Damit die Entwickler täglich oder im Idealfall sogar mehrmals täglich ihre Arbeitspakete nach erfolgreichem Bau und Integrationstest in das zentrale Quellcode-Verwaltungssystem einbringen können, ist eine kurzfristige Rückmeldung über die Qualität der umgesetzten Änderungen erforderlich.

Oft findet man allerdings Projektsituationen vor, in denen ein großes Build-Projekt durchaus mehrere Stunden läuft, wenn beispielsweise umfangreiche Datenbank-Setups als Basis für die Integrationstests aufgebaut werden müssen oder aber ein Deployment auf unterschiedliche Zielplattformen notwendig ist. Kann das erforderliche Feedback nicht in angemessener Zeit bereitgestellt werden, ist der Sinn des CI-Prozesses mehr als fragwürdig.

Neben dem Problem der langen Laufzeiten haben CI-Jobs oft eine weitere unangenehme Eigenschaft: Sie tendieren zu einer „zackigen" Ressourcenauslastung, das heißt, Phasen mit extremer Überlast wechseln sich mit Leerlaufzeiten ab, in denen die Build-Prozessoren ungenutzt bleiben.

Um den beschriebenen Problemen zu begegnen, muss für jedes betroffene Projekt individuell die folgende zentrale Frage beantwortet werden: Welche Strategien zur Skalierung der CI-Jobs bieten sich an und welche Randbedingungen ergeben sich daraus? Der vorliegende Artikel liefert das notwendige Rüstzeug zur Beantwortung eben dieser Fragestellung.

Parallelisierung von Build-Projekten: Hudson-Build-Jobs

Die Vorteile der Parallelisierung von Build-Jobs müssen – wie sollte es anders sein – durch zusätzliche Aufwände erkauft werden: Zunächst muss eine sinnvolle Aufteilung des Gesamt-Builds in einzelne Arbeitspakte erfolgen, bei der die Abhängigkeiten zwischen den jeweiligen Teilschritten Berücksichtigung finden. Außerdem ist eine Synchronisation zwischen den parallelen Ausführungssträngen erforderlich sowie schließlich die Aggregation der Teilresultate zu einem Gesamtergebnis.

Hudson kennt zwei Konzepte, die dem Benutzer das Aufsetzen parallelisierbarer Build-Projekte vereinfachen: das Matrix-Projekt und Pipelined Builds.

Bei einem Matrix-Projekt (s. Abb. 1) ergibt sich durch unterschiedliche Komponenten bzw. diverse Zielplattformen eine Matrix von Build-Jobs. Solche Projekte lassen sich sehr einfach konfigurieren und eignen sich im Normalfall hervorragend für eine parallele Ausführung, weil die einzelnen Jobs weitestgehend unabhängig voneinander sind.

Bei Pipelined Builds zerlegt man einen großen Build-Job manuell in sinnvolle Arbeitspakete und definiert die Abhängigkeiten der einzelnen Teilschritte über Build-Trigger und Post-Build-Hooks.

Unter dem Namen Fingerprint bietet Hudson ein weiteres nützliches Feature an, das dabei hilft, Versionsinformationen der einzelnen Artefakte zu verwalten und diese in darauf aufbauenden Build-Jobs zu referenzieren [FingerPrint].

Sehr nützlich ist in diesem Zusammenhang das Plug-in „Promoted Builds", das es ermöglicht, einen erfolgreichen Build-Job zu einem späteren Zeitpunkt als Promoted Build zu kennzeichnen, sofern zusätzliche Anforderungen erfüllt sind [BuildPromotion]. Ein typisches Beispiel für die Build-Promotion ist ein Compile-Job, an den mehrere nachgelagerte Tests ge-

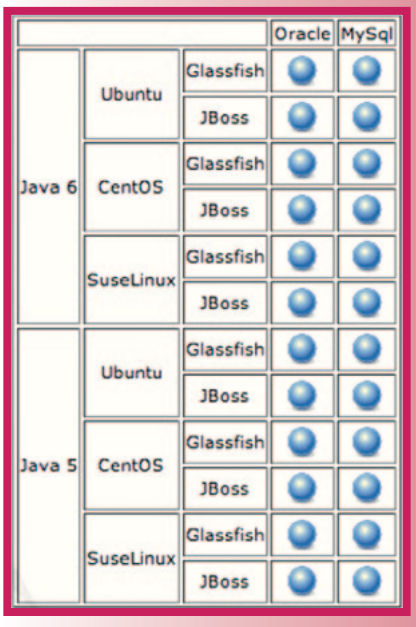

Abb. 1: Beispiel für ein Matrix-Projekt

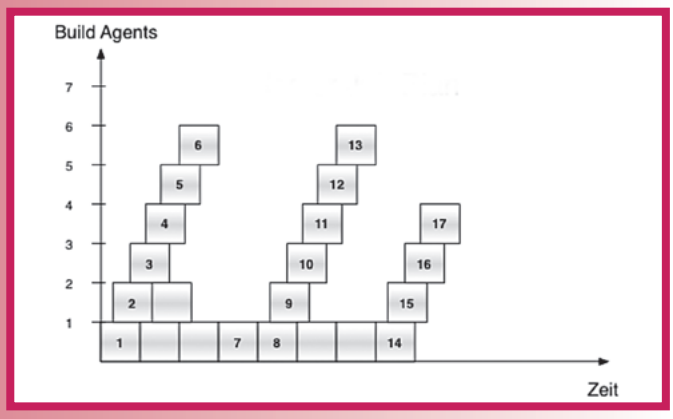

Abb. 2: Idealer Job-Plan

koppelt sind. Erst wenn alle diese Tests erfolgreich durchlaufen sind, bekommt das zugehörige Artefakt des Compile-Jobs das Attribut „promoted" und kann damit für weitere nachfolgende Build-Jobs als Eingabe dienen.

Unter Berücksichtigung der aufgeführten Aspekte ergibt sich für ein beispielhaftes Build-Projekt ein idealer Job-Plan, wie in Abbildung 2 dargestellt, in dem einerseits parallele Abläufe enthalten sind und andererseits Synchronisationspunkte, die für eine Zusammenfassung der Ergebnisse vorgelagerter Jobs verantwortlich sind.

Schaut man sich an, wie eine praktische Zuteilung von Jobs bei Verfügbarkeit von zwei Build-Agenten aussieht (s. Abb. 3), dann stellt man fest, dass die meisten Jobs im Vergleich zum

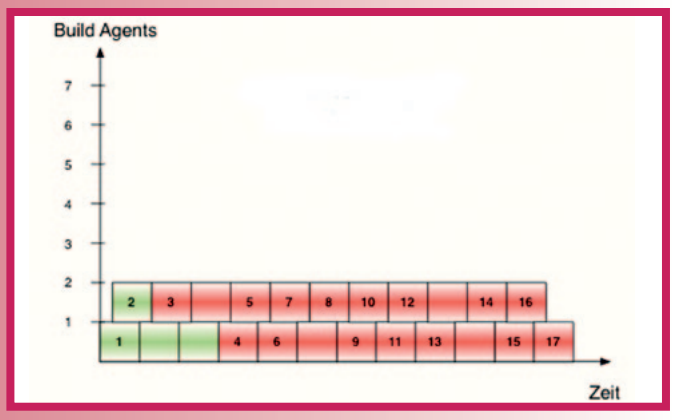

Abb. 3: Job-Zuweisung bei zwei Build-Agenten

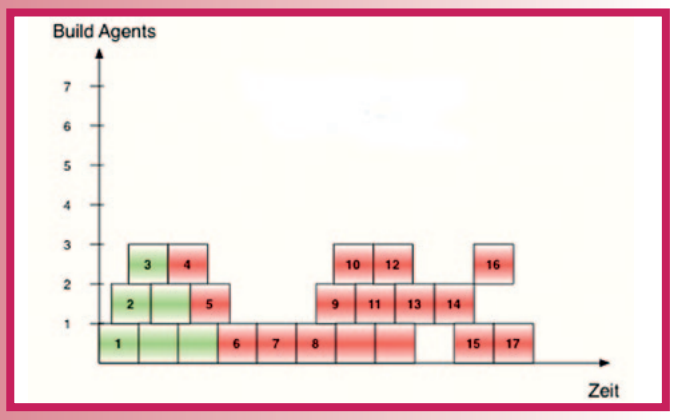

Abb. 4: Job-Zuweisung bei drei Build-Agenten

Idealbild mit einer erheblichen Verzögerung ausgeführt werden (verzögerte Jobs sind rot eingefärbt). Außerdem wird dadurch die Gesamtlaufzeit des Projekts deutlich erhöht.

Wird ein weiterer Build-Server hinzugefügt (s. Abb. 4), zeigt sich, dass nach wie vor die meisten Jobs verspätet ausgeführt werden. Schlimmer noch: Es ergeben sich aufgrund von Abhängigkeiten in der Reihenfolge der Jobs Lücken im Job-Plan. Das wiederum bedeutet, dass vorhandene Kapazitäten teilweise brachliegen. Man bezahlt also für Rechenleistung, die gar nicht effektiv ausgenutzt werden kann.

Cloud-Dienste für Hudson

Um dem idealen Job-Plan in der realen Welt möglichst nahe zu kommen, ohne das eigene Rechenzentrum anhand der maximal zu erwartenden Spitzenlasten zu dimensionieren, bieten sich zwei Cloud-basierte Ansätze an, die in den nachfolgenden Kapiteln weiter vertieft werden:

- Das Hudson-EC2-Plug-in provisioniert und kontrolliert dynamisch und bedarfsorientiert neue Instanzen für Build-Agenten in der Cloud.
- Die DEV@cloud-Plattform von CloudBees beinhaltet eine skalierbare Cloud-basierte Hudson-Umgebung, die ohne zusätzlichen Aufwand direkt zur Konfiguration und Skalierung von eigenen Build-Jobs verwendet werden kann.

Das Hudson-EC2-Plug-in

Das Hudson-EC2-Plug-in ermöglicht es, Build-Jobs auf Instanzen der Amazon Elastic Compute Cloud (EC2) [AmazonEC2] auszuführen. Dazu werden dynamisch und bedarfsorientiert neue EC2-Instanzen in der Cloud bereitgestellt, sobald weitere Build-Agenten benötigt werden. Ist kein Bedarf an zusätzlichen Ressourcen mehr vorhanden, sorgt das Hudson EC2-Plug-in für ein automatisches Herunterfahren der betreffenden EC2-Instanzen.

Die Verwendung des EC2-Plug-ins ist denkbar einfach. Voraussetzung für die Nutzung des Amazon-EC2-Service ist natürlich ein gültiger EC2-Zugang. Wenn man schon einmal ein Buch bei Amazon bestellt hat, erfolgt die Aktivierung des EC2-Accounts durch einen einzigen Mausklick. Ist der Account vorhanden, muss noch ein privater Schlüssel zur Interaktion mit den zu startenden EC2-Instanzen generiert werden. Diese Informationen sind dann auch schon ausreichend, um das EC2-Plug-in in Hudson zu konfigurieren (s. Abb. 5).

Anschließend müssen die Amazon Machine Images (AMIs) konfiguriert werden, die von Hudson aus gestartet werden sollen. Aus einer Liste von mehreren Tausend vorprovisionierten Images kann man sich entweder über die EC2-Kommandozeilen-Schnittstelle oder über die zugehörige Webanwendung (AWS Console) je nach Bedarf AMIs mit dem gewünschten Betriebssystem und sonstigen geforderten Ei-

Abb. 5: Konfiguration des Hudson-EC2-Plug-ins

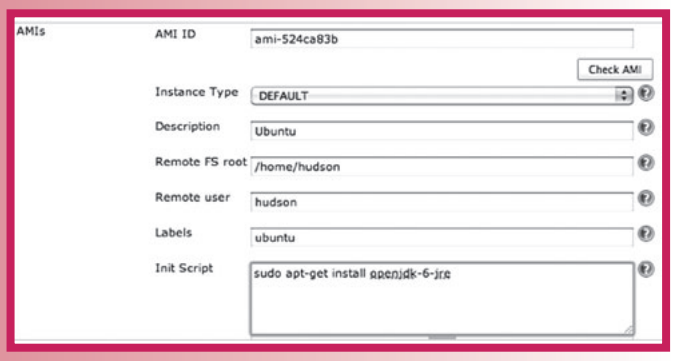

Abb. 6: AMI-Konfiguration

genschaften aussuchen und die zugehörigen IDs in Hudson eintragen (s. Abb. 6).

Sollten vor dem Start der Hudson-Slaves auf den jeweiligen Instanzen noch Initialisierungs-Aufgaben notwendig sein, wie z. B. die Installation einer bestimmten JDK-Version oder eines Git-Clients, kann das an dieser Stelle in Form eines Init-Skripts angegeben werden.

Da Amazon EC2 ein Public-Cloud-Angebot darstellt, hat die Nutzung des zugehörigen Hudson-Plug-ins die Konsequenz, dass die vom Build-Prozess benötigten Artefakte wie Quellcode, Testfälle und Datenbank-Setups außerhalb der physischen Grenzen des eigenen Rechenzentrums verfügbar sein müssen. Diese Tatsache hat je nach Projekt möglicherweise weitreichende rechtliche und sicherheitstechnische Konsequenzen, die von Fall zu Fall abgewogen werden müssen.

Für Anwender, die aus diesem Grund Einwände und Bedenken bei der Nutzung von Public-Cloud-Angeboten haben, könnte folgender Hinweis nützlich sein: Das Private-Cloud-Infrastruktur-Framework Eucalyptus [Euca,KoMü11] ist ein Open-Source-Projekt, das API-kompatibel zu den entsprechenden Amazon-Webservices ist. Es kann problemlos über das Hudson-EC2-Plug-in angesteuert werden. Damit wird es möglich, die Vorteile einer privaten Cloud-Infrastruktur im eigenen Rechenzentrum für Hudson einfach und effizient zu nutzen.

Probleme können auftreten, wenn im Zuge des Build-Vorgangs große Datenmengen zum Build-Agenten übertragen werden müssen, wie z. B. Massendaten für Lasttests. Da hierbei die Bandbreite der Internetverbindung einen limitierenden Faktor darstellt, muss jeweils abgewogen werden, ob sich in diesem Fall das Auslagern in die Cloud aus Zeitgründen überhaupt lohnt.

Die CloudBees-Plattform DEV@cloud

Der zweite Cloud-Ansatz für Hudson geht noch einen Schritt weiter: CloudBees [CloudBees] bietet eine Plattform zum Bauen, Testen und Deployen von Java-Webanwendungen in der Cloud an. Ein wesentlicher Bestandteil dieser Plattform mit dem Namen DEV@cloud ist eine Cloud-basierte und mandantenfähige Hudson-Umgebung, die als Software as a Service (Saas)-Angebot verfügbar ist. Dazu passend werden den Nutzern private und sichere Git-, SVN- und Maven-Repositorys zur Verfügung gestellt, die sich im Zugriff ihrer eigenen Cloud-basierten Hudson-Umgebung befinden.

Anwender können ihre eigenen Build-Projekte direkt über einen Web-Browser konfigurieren. Sie sparen sich dabei die lokale Hudson-Installation sowie deren Feintuning und Pflege und können gleichzeitig je nach Bedarf eine quasi unbegrenzte Kapazität von Build-Agenten in der Cloud nutzen.

Auch wenn die CloudBees-DEV@cloud-Plattform sich zurzeit noch im Aufbau befindet und die CI-Integration aktuell erst seit wenigen Tagen in einer General Availability (GA)-Version vorliegt, wirkt dieses Angebot höchst interessant und vielversprechend. Allein die Tatsache, dass Kohsuke Kawaguchi, Gründer und Chefentwickler von Hudson, nach seinem Weggang von Sun/Oracle seit November 2010 eng mit CloudBees zusammenarbeitet, stärkt die Zukunftsaussichten der DEV@cloud-Plattform immens.

Zurzeit bietet CloudBees drei unterschiedliche Preismodelle auf Basis eines Abonnements an (Base, Dev, Dev Premium), die sich im Leistungsumfang unterscheiden. Das Basisangebot ist kostenlos.

Durch die angebotenen Mehrwertdienste von DEV@cloud ist natürlich die eigene Kontrolle bzw. Einflussnahme stark eingeschränkt. Als Anwender hat man keinen Einfluss auf die Architektur der Plattform oder die verwendete Hardware, sondern ist lediglich in der Lage, die angebotenen Dienste zu konsumieren.

Es ist offensichtlich, dass durch die Nutzung des DEV@cloud-Angebots für den Anwender eine starke Anbieterabhängigkeit entsteht.

Vorteile der Cloud-Ansätze

Wesentliche Vorteile beider Alternativen ergeben sich direkt aus grundlegenden Eigenschaften von Cloud-Diensten:

- *Pay per Use:* Kosten fallen nur dann an, wenn Cloud-basierte Build-Agenten gestartet werden. In Build-Pausen oder wenn die Last komplett durch die Maschinen im eigenen Rechenzentrum bewältigt werden kann, entstehen keinerlei weitere Kosten. Bei CloudBees erfolgt die Abrechnung minutengenau, während bei Amazon EC2 die Kosten stundenweise abgerechnet werden. Man spricht hierbei von einer Verschiebung der Kapitalkosten hin zu den Betriebskosten.
- *Elastizität:* Das Anfordern bzw. Nutzen von Cloud-Angeboten geschieht über entfernt nutzbare Dienste, die programmatisch angesteuert werden können. Je nach Bedarf kann die Verwendung Cloud-basierter Ressourcen kurzfristig angefordert bzw. terminiert werden.
- *Skalierbarkeit:* Durch die hohe Verfügbarkeit von Ressourcen in der Cloud gibt es bezüglich der Skalierbarkeit quasi keine Grenzen. Anbieter sprechen von *unlimited capacity*.
- *Zuverlässigkeit:* Der Betrieb der Cloud-Instanzen erfolgt in einer sehr zuverlässigen und bewährten Umgebung innerhalb des abgesicherten Rechenzentrums des Anbieters. Service Level Agreements (SLAs) treffen verlässliche Zusagen zur Dienstgüte und zur Verfügbarkeit von Ressourcen.

Fazit

Sofern keine rechtlichen Einschränkungen oder Sicherheitsbedenken beim Einsatz von Public-Cloud-Angeboten bestehen, ist die Verwendung von Cloud-Diensten zur Skalierung von großen Hudson-Projekten durchaus sinnvoll. Dabei ist eine Hybrid-Nutzung unter Mitverwendung eigener Ressourcen genauso vorstellbar wie die reine Verwendung von Cloud-Diensten.

Die Nutzung von EC2-Diensten über das Hudson-EC2-Plug-in kombiniert die Vorteile der Cloud, wie Elastizität, Kostentransparenz und die nahezu unbegrenzte Skalierbarkeit, mit den Freiheiten und Optionen, die ein Vollzugriff auf die Hudson-Installation und die Hudson-Slaves gewährt. Außerdem

stellt dieses Angebot aufgrund seiner Dienstreife und Zuverlässigkeit eine gute Basis dar, um Lastspitzen in die Cloud zu skalieren. Alternativ kann über das Hudson-EC2-Plug-in auch eine auf Eucalyptus basierende Private-Cloud-Infrastruktur für die Skalierung von Hudson-Builds verwendet werden.

Die DEV@cloud-Plattform von CloudBees ist ausgerichtet auf die Verlagerung wesentlicher Bestandteile des Softwareentwicklungsprozesses in die Cloud. Ein interessanter Ansatz, der auf jeden Fall weiterverfolgt werden sollte. Aber auch die alleinige Nutzung des Hudson-SaaS-Angebots von CloudBees kann durchaus sinnvoll sein. Diese Lösung bietet sich vor allem dann an, wenn ein Projekt ohne CI-Prozess existiert, oder wenn ein neues Projekt initiiert wird und die komplette Infrastruktur neu aufgesetzt werden muss. Außerdem ist es eine gute Wahl, wenn der Betrieb der eigenen Hudson-Umgebung zu viel Zeit kostet oder um bei Lastspitzen Build-Jobs in die Cloud auszulagern.

Da das CloudBees-Portfolio sich noch im Aufbau befindet und die bisher realisierten Dienste erst seit Kurzem dem Beta-Stadium entwachsen sind, gibt es kaum tragfähige Aussagen über die Service-Qualität, sodass es für einen Produktiveinsatz in kritischen Projekten noch nicht zu empfehlen ist.

Literatur und Links

[AmazonEC2] Amazon Elastic Compute Cloud (EC2), `http://aws.amazon.com/de/ec2/`

[BuildPromotion] Hudson Promoted Builds Plug-in, `http://wiki.hudson-ci.org/display/HUDSON/Promoted+Builds+Plugin`

[CloudBees] Plattform as a Service (Paar)-Angebot zum Bauen, Testen und Deployen von Java Web-Applikationen in der Cloud, `http://cloudbees.com/`

[Euca] Open-Source-Cloud-Plattform, `http://open.eucalyptus.com/`

[FingerPrint] Version Tracking mit Hudson Fingerprint, `http://wiki.hudson-ci.org/display/HUDSON/Fingerprint`

[Fowl06] M. Fowler, Continuous Integration, Mai 2006, `http://www.martinfowler.com/articles/continuousIntegration.html`

[Hudson] Homepage des Hudson CI-Servers, `http://hudson-ci.org/`

[KoMü11] M. Kopp, St. Müller, Migration einer JEE-Umgebung in eine hybride Cloud, in: JavaSPEKTRUM, 2/2011

Arnd Kleinbeck ist Senior Consultant bei der innoQ Deutschland GmbH. Er beschäftigt sich dort aktuell mit skalierbaren Architekturen im Webumfeld und den Einsatzmöglichkeiten von Cloud Computing in Business-Szenarien.
E-Mail: arnd.kleinbeck@innoq.com