2020 Leitfaden für

Schnellere Builds mit kontinuierlicher Integration

Von: Jim Holmes

INCREDIBUILD

Inhaltsverzeichnis

Wir hassen langsame Builds ohnehin – jetzt gibt es hierfür noch einen weiteren Grund

Langsame Builds bereiten jedem Projektteam schon bei der Arbeit an einem nur mäßig komplexen Softwareprojekt Kopfschmerzen. Delivery-Teams haben Schwierigkeiten, in einen gleichbleibenden Arbeitsrhythmus zu finden: Anforderungen festlegen, die Arbeit besprechen, die Arbeit erledigen, die Software erstellen und überprüfen, an die entsprechende Umgebung senden. Es ist schwierig, an einen Punkt zu kommen, an dem Ihr Team mit diesem Arbeitsrhythmus vertraut ist. Alles, was diesen Arbeitsrhythmus verlangsamt, wirkt sich direkt auf die Gesamtqualität und den Wert eines Produkts aus.

Noch schwieriger ist es, diesen Arbeitsrhythmus beizubehalten, wenn das Projekt weiter wächst und immer ausgereifter wird. Weitere Funktionen werden hinzugefügt, Abhängigkeiten und Komplexität nehmen zu, und eine Reihe anderer Probleme treten auf. Oft entwickelt sich der Build-Prozess zum Haupthindernis für den Erfolg eines Projekts. Waren Sie schon einmal an einem Projekt beteiligt, bei dem der All-Inclusive-Build nur einmal pro Nacht durchgeführt wurde oder, noch schlimmer, bis zum Wochenende liegen gelassen wurde?

Eine solche Umgebung stellt ein ernstes Risiko für die Qualität des Projekts dar: Verzögerungen bei Builds führen dazu, dass Teams in kritischen Bereichen Abstriche machen. Tests werden nicht aufgeschrieben. Es dauert zu lange, bis ein reibungsloser Ablauf rot – grün - Refactoring einer guten Test-First-Entwicklung erreicht ist. Stattdessen entsteht eine Test-Whenever-Entwicklung. Tests werden nicht bei jedem lokalen Build ausgeführt. Die zusätzlich benötigte Zeit ist bei einem bereits langsamen Build einfach zu lang. Tests sind nicht Teil des festgelegten Testumfangs. Man hat nur genug Zeit, um wenige Tests auszuführen und prüft Integrationstests nicht auf Nebenwirkungen

Auf den folgenden Seiten werden wir verschiedene qualitätsbezogene Probleme bei langsamen Builds diskutieren und auf Lösungsansätze eingehen.

Die Zeit als entscheidender Faktor

24 Stunden pro Tag, sieben Tage pro Woche, vier Wochen pro Monat. Diese zeitlichen Begrenzungen sind unumgänglich. Sie können weitere Mitarbeiter engagieren, aber die Arbeit, die in einem festgelegten Zeitraum erledigt werden kann, ist beschränkt. Wenn sich die Build-Zyklen verlängern, müssen die Teams überlegen, was sie am Prozess ändern können. Oft werden Kompromisse geschlossen, die die Gesamtqualität eines Projekts gefährden, dafür aber mehr Funktionen, verknüpfte Ressourcen, zusätzliche Projekte usw. hinzufügen.

Im folgenden Abschnitt werden wir uns die Auswirkungen solcher Entscheidungen genauer ansehen.

So optimieren Sie Ihre Delivery-Prozesse

Es gibt verschiedene Methoden der Softwareentwicklung – agil, Wasserfall, Scrummerfall, chaotisch usw. Doch egal, welche sie nutzen: Ihr Team und Ihre Kunden profitieren davon, dass Sie an einer Verbesserung der Softwarebereitstellung arbeiten. Wenn Sie Ihren Delivery-Prozess reibungslos gestalten, entsteht zum Release-Zeitpunkt weniger Stress. Die Verbesserung Ihres Deliver-Prozesses führt zu weniger Angst beim Hinzufügen von Funktionen oder Ändern der Codebasis. Durch einen verbesserten Deliver-Prozess erhalten Ihre Kunden schneller einen besseren Mehrwert – das ist kein Mythos!

Kontinuierliche Verbesserung (Continous Improvement)[1] ist ein Konzept, das Teams und Organisationen bei der Optimierung ihrer Softwarebereitstellung unterstützen soll. Sie müssen nicht alles, was im referenzierten Artikel steht, übernehmen. Es kann für Sie schon ausreichen, sich auf wenige Dinge zu konzentrieren – Ihr Build-Zyklus sollte dabei ganz oben auf der Liste stehen!

Feedback-Zyklen gehören zu den entscheidenden Themen, wenn es um kontinuierliche Verbesserung, Lean-Software, agile Software und andere Philosophien/Methoden geht. Ein Feedback-Zyklus (oder eine Feedback-Schleife) ist ein Ausdruck für die Zeit, die benötigt wird, bis sich eine Änderung im System widerspiegelt. Feedback-Zyklen existieren auf der langen/langsamen Skala (wurde der von einem Kunden gemeldete Fehler in der neuesten Version behoben?) und der viel kleineren/schnelleren Skala (hat der Komponententest, den ich gerade geschrieben habe, bei der Ausführung zu einem positiven Ergebnis geführt?).

Es wurde viel über Feedback-Zyklen geschrieben. Die drei unten aufgeführten Beiträge sind ein guter Ausgangspunkt, wenn Sie mehr darüber erfahren möchten:

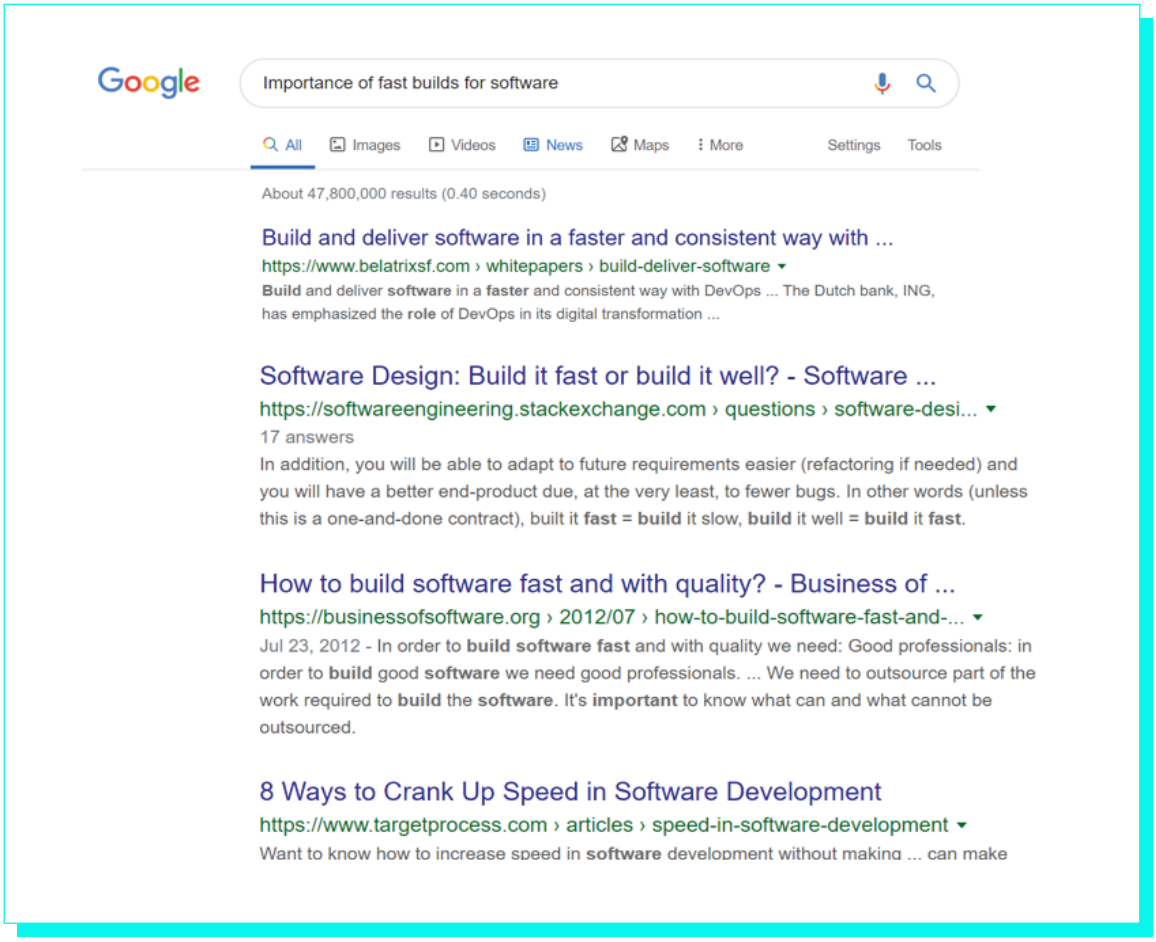
- Scott Amblers[2] Artikel über die Wichtigkeit schneller Feedback-Zyklen für die agile Methode
- Ron Jeffries[3] Text über die Wichtigkeit von Tests für ein gutes Feedback

1 Einen großartigen Einführungsartikel finden Sie unter http://leankit.com/kanban/continuous-improvement/
2 www.ambysoft.com/essays/whyAgileWorksFeedback.html
3 ronjeffries.com/xprog/what-is-extreme-programming/

Vordenker der Branche über die Leistungsfähigkeit schneller Builds

Zum Glück gibt es in der Softwareindustrie Vordenker, die seit langem auf die Bedeutung eines schnellen Build-Zyklus hinweisen. Über dieses Thema wurde viel geschrieben und diskutiert. In Michael Feathers Standardwerk „Working Effectively with Legacy Code“ wird wiederholt von einer verkürzten Build-Zeit als Teil der Einschränkung der Codebasis gesprochen. James Shores und Shane Wardens wegweisende Arbeit „The Art of Agile Development“ (und andere Bücher) spricht von einem zehnminütigen Build-Zyklus.

Eine einfache Google-Suche zeigt, wie sich das Denken darüber in den letzten zehn Jahren verändert hat. Bereits 2001 betonten Joel Spolsky und andere die Bedeutung eines Builds mit täglicher Integration.



Die Branche hat sich seitdem weiterentwickelt und konzentriert sich nun auf die kontinuierliche Integration (Continous Integration). Die Philosophie der kontinuierlichen Integration[4] befürwortet tägliche Builds und Integrationen. Teams werden jedoch ermutigt, auf viel häufigere Builds/Integrationen zu setzen. Ganze Tool-Plattformen wurden entwickelt, um Teams dabei zu

4 https://www.thoughtworks.com/continuous-integration

unterstützen, häufigere Builds/Integrationen/Deployments nahezu kontinuierlich, oft alle fünf bis zehn Minuten, durchzuführen. Dadurch wird der Feedback-Zyklus erheblich beschleunigt. Teams erhalten bei der Integration fast sofort einen Pass/Fail-Status, ohne tagelang warten zu müssen!

Unsere Branche entwickelt sich weiter. Mittlerweile haben wir erkannt, dass selbst ein tägliches Feedback zu Integrationen, Builds und Tests des gesamten Systems zu langsam sein kann. Jez Humble[5] und andere Vordenker der Branche betonen den Wert einer kontinuierlichen Delivery, nicht nur einer kontinuierlichen Integration!

Auswirkungen langsamer Builds

Langsamere Builds haben viele Auswirkungen auf die Bemühungen eines Teams. Diese Auswirkungen betreffen die Arbeitsmoral, die Kosten und die Fähigkeit, qualitativ hochwertige Software zu erstellen.

Langsamere Wertlieferung an den Kunden

Letztendlich kommt es nur darauf an, dass Softwareteams ihren Kunden einen Nutzen liefern. Alles, was die Weiterentwicklung von Codes für die Produktion und die Weitergabe an Endbenutzer behindert, bremst das Projekt. In „Lean Software“ fragen Tom und Mary Poppendieck bekanntlich: „Wie lange dauert es, eine Codezeile für die Produktion bereitzustellen?“

Was passiert, wenn Sie einen schwerwiegenden Defekt in Ihrer Produktionsumgebung haben, beispielsweise einen, der Ihren Umsatz gefährdet – vielleicht eine entscheidende Sicherheitslücke? Können Sie das Problem in angemessener Zeit beheben und Lösungen voranbringen, erstellen, testen und bereitstellen? Oder sind Sie durch Ihren Build-Prozess so eingeschränkt, dass es Stunden oder – noch schlimmer – Tage dauern kann, eine Änderung einzuführen?

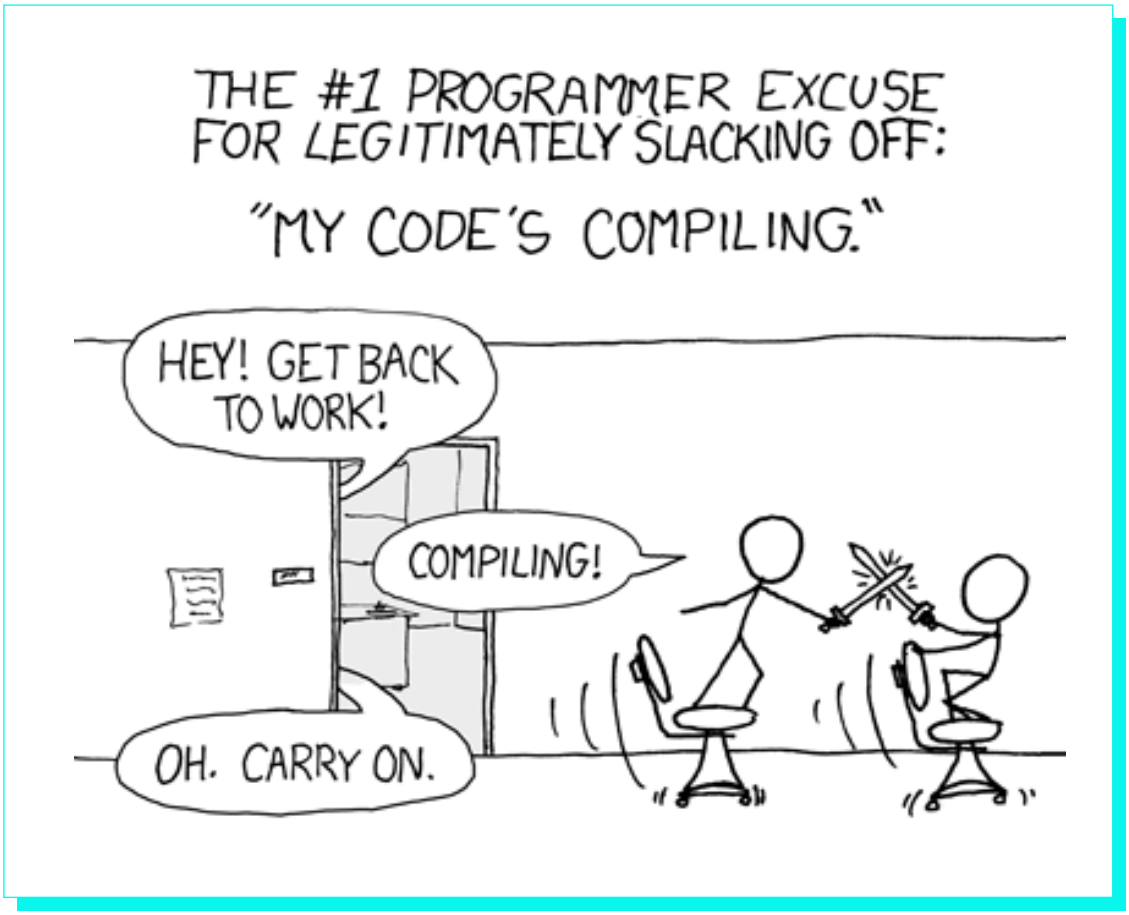
Nichterreichen des Flow-Zustands

Softwaretester und insbesondere Entwickler reden gerne davon, einen Flow-Zustand erreichen zu wollen. Der Flow ist ein Zustand höchster Konzentration und Fokussierung, in dem eine Person extrem effizient arbeiten kann. Programmierer möchten den Flow-Zustand erreichen, weil sie dann in der Lage sind, komplexe Probleme schnell zu lösen, schwierige Aufgaben zu erledigen und in sehr kurzen Zeiträumen Nutzen zu liefern.

Nicht alle sind glücklich, wenn der Build beschleunigt wird. XKCDs berühmter Comic[6] ist die humorvolle Version eines Teams, das die Zeit mit langsamen Builds auf „interessante“ Weise füllt. Aber vielleicht finden Sie einen anderen Weg, um die Bedürfnisse Ihres Teams nach kreativem Anachronismus zu erfüllen!

5 http://continuousdelivery.com/

6 https://xkcd.com/303/ (Creative Commons BY-NC 2.5)



Source: xkcd

Schlechtere Wartbarkeit

Menschen vermeiden schmerzhafte Situationen. Dies gilt für körperliche Verletzungen, emotionalen Stress, SharePoint-Entwicklungsprojekte usw. Aber auch extrem lange Build-Zyklen passen sicherlich in diese Kategorie. Der Wunsch, schmerzhaft lange Integrations- und Testbuilds zu vermeiden, kann sich tatsächlich negativ auf die Wartbarkeit von Projekten auswirken. Die Teams werden es vermeiden, einen Integrations-Build auszuführen, der mehrere Komponenten, Zweige, Projekte usw. enthält, wenn die Ausführung dieses Builds zu lange dauert. Die Integration wird auf einen nächtlichen Build, dann auf einen wöchentlichen Build, dann auf eine Iteration usw. verschoben.

Teams in diesem Zustand haben sich mit der Komplexität abgefunden, statt daran zu arbeiten, sie auflösen. Martin Fowler hat einen großartigen Artikel darüber geschrieben, wie man Integrationen so oft wie möglich durchführt.[7] Sein Motto lautet „Wenn es weh tut, tun Sie es öfter“. Das heißt, arbeiten Sie sich durch die unangenehmen Teile der Delivery Pipeline, damit Sie sich darauf konzentrieren können, so schnell wie möglich Mehrwert zu liefern.

7 http://martinfowler.com/bliki/FrequencyReducesDifficulty.html

Nichtbeachtung automatisierter Tests

Leider können langsame Builds auch dazu führen, dass sich Teams immer weniger auf automatisierte Testsuiten verlassen. Denken Sie nach: Wenn Sie bereits unter langen Build-Zeiten leiden, besteht eine natürliche (aber FALSCHE!) Reaktion darin, alles zu vermeiden, das die Build-Zeit weiter verlängert. Dies hat sich in der Softwareindustrie gezeigt, wo Teams weniger Tests schreiben und diese Tests seltener ausführen. Dies führt zu vermehrten Regressionen, insbesondere in komplexeren Systemen, da Teams die Vorteile ihrer automatisierten Testsuiten verlieren.

Ein praktisches Beispiel hierfür ist ModuleWorks, ein Anbieter von CAD/CAM-Komponenten. Für die komplexe Verteilungsarchitektur von ModuleWorks müssen 32 verschiedene Sätze freigebbarer Binärdateien erstellt werden, um Kundenkonfigurationen zu unterstützen. Die Build-Zeiten waren so lang, dass Entwickler nur einmal am Tag integriert haben – und ihre GTest-Suite für den C++ Code wurde selten ausgeführt.[8] Erst nach der Implementierung von Incredibuild (siehe unten) konnte ModuleWorks eine kontinuierliche Integration und automatisierte Tests implementieren.

Eingeschränkter Umfang ausgeführter Tests

Eine andere Möglichkeit für Teams, die Build-Zeiten zu verkürzen, besteht darin, den Umfang der ausgeführten Tests zu verringern. Anstatt alle Tests in Einheiten- oder Integrationssuiten auszuführen, können Entwickler dazu neigen, den Testausführungsbereich auf die spezifische Komponente zu beschränken, an der sie gerade arbeiten.

Diese schlechte Praxis führt dazu, dass potenzielle Nebenwirkungen weniger auffallen, wenn Teams Code schreiben. Denken Sie nach: Einer der Hauptgründe für eine automatisierte Testsuite ist, ein Sicherheitsnetz für die weitere Entwicklung zu haben! Eine gut geschriebene, durchdachte Testsuite gibt uns die Gewissheit, Teile des Systems zu ändern, während das Verhalten in anderen Bereichen des Systems durch Tests blockiert wird.

Retalix, ein Geschäftsbereich von NCR Retail, bietet eine optimierte Suite von Systemen für Online-, Mobil- und In-Store-Einzelhandelssysteme. Retalix verfügte über eine Suite, die 15.000 Komponententests umfasste, deren Ausführung 12 Minuten dauerte – nach einer vollständigen Optimierung des Systems jedes Entwicklers. 12 Minuten für die Durchführung von Komponententests sind viel zu lang. Die Entwicklungsteams von Retalix haben die Tests daher nicht so oft durchgeführt, wie sie sollten. Lesen Sie die dazugehörige Fallstudie, um zu erfahren, wie Retalix die Testläufe auf 1 Minute 20 Sekunden reduzieren konnte![9]

8 ModuleWorks beschleunigt Tests und kontinuierliche Integration und ermöglicht Advanced Manufacturing, Fallstudie von Incredibuild abrufbar unter https://www.incredibuild.com/case-studies/moduleworks
9 Siehe „Retalix Case Study“ unter https://www.incredibuild.com/case-studies/retalix

Nichtbeachtung statischer Analysen und anderer Qualitätsansätze

Tools wie die Code Analysis Tools von Visual Studio, NDepend, SonarQube oder ähnliche Werkzeuge können eine große Hilfe sein, um die Qualität und Wartbarkeit einer Codebasis sicherzustellen. Diese Tools kennzeichnen Qualitätsprobleme wie Komplexität, Abhängigkeiten, Codemetriken usw. und können als Gate zur Codeverschiebung durch die Delivery Pipeline verwendet werden.

Leider dauert die Ausführung dieser Tools einige Zeit, was bedeutet, dass sie in Umgebungen, in denen Builds bereits die Produktivität und Bereitstellung einschränken, sicherlich nicht verwendet werden.

Manuelle Beschleunigung von Builds

Teams lernen, langsame Builds auf vielfältige Weise zu überwinden, anstatt nur unter ihnen zu leiden. Diese Anstrengungen erstrecken sich manchmal über ein Jahr oder länger und beziehen sich auf die Neukonfiguration komplexer Buildzyklen, das Aufstellen neuer Hardware und die Überarbeitung ganzer Testsuiten.

Mehr Hardware behebt alle Probleme!

Natürlich setzen viele Unternehmen zunächst auf Hardware, um langsame Builds zu beschleunigen. In vielen Fällen kann dies sinnvoll sein. Es ist zudem einfach, ein oder zwei Build-Server auszutauschen. Was passiert jedoch, wenn das Problem dadurch nicht wesentlich verbessert wird? Das Beschaffen, Einrichten, Konfigurieren und Warten von Hardware kostet Zeit.

Wenn Sie einen neuen Pool von Remote-Agenten erstellen möchten, müssen Sie auch für diese die Beschaffung, Einrichtung, Konfiguration und Wartung übernehmen! Darüber hinaus kann die Integration dieser Remote-Agenten in Ihr Build-System äußerst schwierig sein. Sie müssen sicherstellen, dass alle Ihre Toolketten parallelisiert werden: Build-Server, Build-System, Test-Frameworks usw. Teams, die ein ausgeglichenes Aufwand-Nutzen-Verhältnis anstreben, können den nötigen Aufwand nur schwer rechtfertigen. Unabhängig davon besteht die Beschränkung durch die Geschwindigkeit Ihrer stärksten Build-Hardware weiterhin. Und diese reicht möglicherweise nicht aus. Der Kauf zusätzlicher Hardware, die eventuell nur für Spitzenzeiten benötigt wird, kann teuer sein. Stattdessen können Sie auch Ihre bereits vorhandene Hardware und Verarbeitungsleistung optimieren.

Manuelle Aufteilung in Komponenten/Aufgaben

Builds können manchmal beschleunigt werden, indem der Build-Job manuell in mehrere Teile aufgeteilt und diese Teile manuell ausgeführt werden.

Diese Lösung birgt jedoch Gefahren. Teams, deren Arbeitsbelastung bereits durch die Softwarebereitstellung hoch ist, müssen den Build-Prozess manuell über Abhängigkeiten, Aktualisierungen und neue Aufgaben hinweg verwalten. Solche Dinge manuell zu erledigen, ist kostspielig und zwingt die Teams, ein tiefgreifendes Fachwissen über die Build-Pipeline aufzubauen.

Das Abhängigkeitsmanagement ist ein anstrengender, aber entscheidender Bestandteil jedes Softwaresystems. Werkzeuge für das Plattform-Abhängigkeitsmanagement wie Bundler für Ruby, Maven für Java, NuGet für .NET und andere sind komplexe Tools, mit denen das Abhängigkeitsmanagement einfacher gestaltet werden kann.
Teams, die Builds aufteilen, sind für einen Großteil dieses Abhängigkeitsmanagements verantwortlich, da die Build-Pipeline nicht weiß, wie mit Abhängigkeiten zwischen separaten Build-Jobs umgegangen werden soll.

Kosten und Aufwand

Es ist wichtig, dass die Teams die Kosten für die Build-Beschleunigung berücksichtigen. Viele Teams verbringen Monate, wenn nicht Jahre damit, ihre Build-Prozesse zu optimieren. Emanuil Slavov erzählte in seinem Vortrag „Need for Speed“ auf der ISTA-Konferenz 2015 die Geschichte einer 18-monatigen Reise, deren Ziel es war, die Testausführungszeit von drei Stunden auf drei Minuten zu verkürzen. Das ist eine großartige Verbesserung. Es wurde jedoch unglaublich viel Aufwand dafür betrieben.

War das Ergebnis am Ende diesen Aufwand wert? Für Emanuils Team hat es sicherlich gelohnt. Man kann sich jedoch fragen, welche anderen Optionen sie möglicherweise gehabt hätten. Optionen, bei denen sie Hunderte oder Tausende von Stunden gespart hätten und sich auf die Bereitstellung statt auf die Infrastruktur konzentriert hätten.

Schnellere Builds und Tests mit Incredibuild

Unternehmen, die ihre Build-Zeiten verkürzen möchten, stehen vor einer Gratwanderung: Sie müssen überlegen, wie sie die Geschwindigkeitssteigerungen erreichen können, ohne zu viele Mitarbeiter und Ressourcen dafür zu aufzuwenden. Emanuil Slavov und sein Team brauchten, wie zuvor erwähnt, 18 Monate, um ihre Ziele zu erreichen. Sie erzielten enorme Ergebnisse, nachdem sie lange und hart an der Aktualisierung der Infrastruktur, der Build-Konfigurationen, der Datenverwaltung und der Testautomatisierungsstrategie gearbeitet hatten. Obwohl die Ergebnisse großartig waren, hätte man diesen Arbeitsaufwand möglicherweise besser direkt in die Wertschöpfung investieren können, wenn andere Optionen für die Build-Optimierung verfügbar gewesen wären.

Überblick über die Architektur und Lösungen von Incredibuild

Incredibuild bietet Teams und Organisationen genau diese Möglichkeit: die Build-Infrastruktur schnell zu skalieren, um die Build-Zeiten drastisch zu verbessern – alles mit minimalen Investitionen in Einrichtung, Konfiguration und Verwaltung der Build-Ressourcen. Mit Incredibuild können Unternehmen mit einem zentralen Build-Koordinator schnell Build-Agentenpools aufbauen. Entwickler, Tester oder andere Mitarbeiter, bei denen Incredibuild installiert ist, können einen Build initiieren, der über den Agentenpool verteilt wird.
Die Aufteilung eines langfristigen Builds in Tasks und die parallele Ausführung dieser Tasks bringt enorme Vorteile, benötigt zugleich aber nur wenig Zeit und Ressourcen.

Darüber hinaus kann Incredibuild andere Tasks mit langer Laufzeit ausführen. Spieleentwickler verwenden Incredibuild, um ihre Grafik- und Audio-Rendering-Jobs zu beschleunigen. Diese Aufgaben nehmen bekanntermaßen übermäßig viel Zeit in Anspruch. Andere Unternehmen nutzen Incredibuild, um die Ausführungszeiten für automatisierte Testsuiten drastisch zu verkürzen und die Codeanalyse zu beschleunigen.

Es ist wichtig zu verstehen, dass die Incredibuild-Agenten nicht mit denen von Jenkins, Team Foundation Server oder TeamCity identisch sind. Die Agenten dieser Tools ergänzen die Agenten von Incredibuild. Incredibuild befreit Jenkins Build-Agenten beispielsweise von den Leistungsbeschränkungen durch die eigene Hardware. Die beiden Agentensätze können zusammen verwendet werden, um eine Build-Workstation in ein System mit potentiell Hunderten von Prozessorkernen und Giga-Bytes an Speicherplatz umzuwandeln.

Mit den Funktionen von Incredibuild müssen Unternehmen nicht in mehrere neue Systeme investieren, um ihre langfristigen Tasks zu skalieren. Stattdessen können Unternehmen die Agenten von Incredibuild auf vorhandenen Systemen verwenden. Incredibuild verteilt Tasks nur an Systeme, die über eine angemessene Menge an freien CPU-Zyklen und Speicher verfügen. Das bedeutet, dass Agenten auf Systemen ausgeführt werden können, die gerade im Leerlauf sind, auf Systemen, die von Administratoren oder Programmmanagern nur wenig verwendet werden, und sogar auf Systemen, die von Entwicklern aktiv verwendet werden. Weiter unten werden Sie sehen, wieso auf diesen Agentensystemen nicht einmal Quelldateien, Bibliotheken oder Build-Tools installiert werden müssen.

Bei Bedarf können Sie den Agentenpool von Incredibuild problemlos auf Cloud-Ressourcen skalieren, um Hunderte oder sogar Tausende von Kernen für Ihre Build-Aufgaben zur Verfügung zu haben!

Skalieren in die Cloud

Mit Cloud-Plattformen wie Amazon AWS, Cloud Foundry von Pivotal oder Azure von Microsoft können Unternehmen ihre Systeminfrastruktur schnell skalieren. Warum sollten dieselben Plattformen nicht dazu dienen, die Build- und Lieferpipelines eines Unternehmens zu unterstützen?

Ehrlich gesagt, ist es Incredibuild egal, wo Sie Ihre Build-Agenten definiert haben – es muss nur in der Lage sein, mit ihnen zu kommunizieren. Incredibuild kann jede cloud-basierte Infrastruktur verwenden, solange sie die Bereitstellung virtueller Maschinen unterstützt. Im Allgemeinen bedeutet dies, dass Sie eine Art VPN benötigen, das mit diesen cloud-basierten Systemen eingerichtet wurde. Wenn Sie dieselben Plattformen für Ihre Systementwicklung verwenden, ist dies wahrscheinlich bereits vorhanden.

Wenn Sie Microsoft Azure verwenden, ist es noch einfacher: Incredibuild bietet einen Out-of-the-Box-Support für die Arbeit mit Azure. Richten Sie Ihr VPN ein, übertragen Sie die Incredibuild-Agentensoftware auf Ihre Azure-Instanzen, und schon können Sie diese Instanzen in Ihren Agentenpool ziehen. Sie können Incredibuild sogar verwenden, um bei Bedarf automatisch neue Instanzen bereitzustellen.

Diese Benutzerfreundlichkeit erstreckt sich über den gesamten Funktionsumfang von Incredibuild. Das Ziel von Incredibuild ist es, sicherzustellen, dass Unternehmen weniger Zeit für die Einrichtung und Konfiguration der Build-Infrastruktur aufwenden und sich mehr darauf konzentrieren, ihren Kunden Mehrwert zu liefern.

Low-Impact-Lösung

Incredibuild konzentriert sich auf Benutzerfreundlichkeit, damit Teams schnell wieder an ihre Arbeit zurückkehren können. Die Installation und Konfiguration von Incredibuild ist selbst in großen Umgebungen sehr einfach. Die Standardeinstellungen werden sorgfältig ausgewählt, damit die Teams bei einer Out-of-the-Box-Installation von großen Vorteilen profitieren. Beispielsweise sind sinnvolle Mindestanforderungen für CPU-Auslastungsgrenzen, verfügbaren RAM, Festplatten-Caching usw. vordefiniert.

Incredibuild geht sogar noch einen Schritt weiter, indem Tools zum Erstellen eines Agenten-installationspakets bereitgestellt werden, das projektspezifische Konfigurationen enthält. Wenn Sie dieses Installationsprogramm ausführen, wird die Agentensoftware eingerichtet, es werden projektspezifische Anpassungen vorgenommen und es wird eine Verbindung zum Incredibuild-Controller hergestellt – alles automatisch.

Pool-Agenten benötigen zudem keine zusätzliche projektspezifische Software. Sie müssen nichts installieren – weder Visual Studio, noch Xbox-Entwicklungsumgebungen, gcc, abhängige Bibliotheken, Vorlagen oder ähnliches. Durch die Prozessvirtualisierung von Incredibuild werden diese Ressourcen gebündelt und an den Agenten weitergeleitet. Der Agent erhält dieses Bundle und kann alle zugewiesenen Aufgaben ausführen, ohne jemals auf lokale Ressourcen angewiesen zu sein. Dies ist eine enorme Zeitersparnis für Unternehmen, da sich Teams nie darum kümmern müssen, Bibliotheken zu aktualisieren, Tools zu patchen oder Service Packs auf komplexe Systeme anzuwenden. (Hinweis: Ressourcen auf Systemebene wie Antiviren- und Betriebssystem-Patches müssen weiterhin verwaltet werden!)

Dieser Ansatz hat es den Kunden von Incredibuild ermöglicht, ihre Build- und Paketerstellungszeiten schnell und dramatisch zu verbessern, wie aus mehreren praktischen Beispielen hervorgeht.

Beispiele für reale Zeitersparnisse

Unternehmen auf der ganzen Welt haben sich an Incredibuild gewandt, um Hilfe bei der Verbesserung langer Build-Zyklen in einer Vielzahl von Bereichen zu erhalten.

Reifung von langen Builds zu kontinuierlicher Integration

Algotec entwickelt die Carestream Vue-Produktsuite, die von Tausenden medizinischen Einrichtungen auf der ganzen Welt genutzt wird. Die Vue-Suite hilft bei der Optimierung der medizinischen Bildgebung und basiert auf einer komplexen Codebasis von über 400 C++ Projekten, von denen viele mehr als 30 Dateien in jedem Projekt enthalten. Quelldateien nutzen in großem Umfang verarbeitungsintensive Funktionen wie Makros, TLB-Importe und benutzerdefinierte Build-Schritte.

Infolgedessen konnte der Build für diese Suite nur einmal pro Nacht ausgeführt werden. Die Integration von Änderungen durch die Entwickler war langsam und aufwendig. Dies führte zu sehr langen Feedback-Zyklen für die Entwickler. Lange Feedback-Zyklen führen häufig zu einer Abwärtsspirale bei der Qualität. Für ein Team, das sich gerade weiterentwickelt und Prozesse wie kontinuierliche Integration oder Delivery übernimmt, können sie ein Hemmnis sein.

Algotec unternahm den einfachsten Schritt zur Implementierung einer Incredibuild-Lösung: einfach einen Agenten auf dem System jedes Entwicklers installieren und vorhandene Systeme im Algotec-Netzwerk nutzen. Dieser eine Schritt führte bei kleineren Projekten zu Verbesserungen von 90%. Bei größeren Projekten wurde zunächst eine geringere Verbesserung festgestellt: von 140 Minuten auf 40 Minuten. Algotec und Incredibuild arbeiteten zusammen, um Abhängigkeiten und andere Projekteinstellungen zu optimieren. Schließlich wurden die Builds auf 34 Minuten reduziert, was einer Verbesserung von 80% entspricht.

Durch diese Zeitersparnis konnte Algotec in einen kontinuierlichen Integrationsfluss übergehen und seine Builds mehrmals täglich ausführen. Algotec konnte auch mit der Umsetzung automatisierter Tests beginnen, was ohne die außergewöhnlichen Zeitersparnisse durch die Incredibuild-Implementierung nicht möglich gewesen wäre.

Zeitersparnis bedeutet besseres Testen

ModuleWorks entwickelt äußerst komplexe, hochgradig kundenspezifische Lösungen für die computergestützte Fertigungsindustrie weltweit. Viele Kunden benötigen sehr spezielle Konfigurationen, sodass ModuleWorks für jeden wichtigen Release mindestens 32 verschiedene Binärsätze erstellen muss.

Aufgrund der komplexen Build-Anforderungen mussten ModuleWorks-Entwickler die Integrationen nacheinander durchführen. Schlimmer noch: Durch die Länge des Builds (mehrere Tage!) konnten sie die Integration nur einige Male pro Woche durchführen. Darüber hinaus wurden Tests, die mit dem GTest-Framework von Google geschrieben worden waren, durch die Ausführungsdauer beeinträchtigt. Neue Tests wurden nicht geschrieben und vorhandene Tests wurden nicht häufig genug ausgeführt, um zunehmende Qualitätsprobleme zu vermeiden.

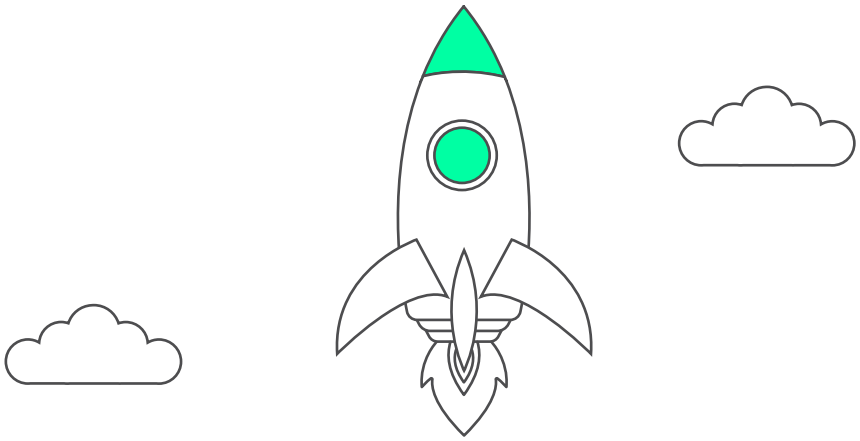
Durch die simple Einführung von Incredibuild konnte ModuleWorks die Build-Zeiten von 30 Minuten auf drei Minuten verkürzen. Die Testausführungszeiten für die GTest-Suiten gingen um 86% zurück. Vor der Implementierung von Incredibuild dauerte ein vollständiger Build- und Testzyklus mehr als drei Stunden. Nach der Integration von Incredibuild beträgt der vollständige Build-Zyklus weniger als 30 Minuten.

Durch die Einsparung von mehr als 2,5 Stunden pro Build-Zyklus hatten die Entwickler von ModuleWorks wieder freie Kapazitäten und konnten mehr automatisierte Tests durchführen. Besser noch, ModuleWorks verzeichnete signifikante Rückgänge bei den Regressionen, da die Testsuiten wachsen und häufiger ausgeführt werden. Die Gesamtqualität der von ModuleWorks gelieferten Produkte hat sich stark verbessert.

Geben Sie sich nicht mit langen Builds zufrieden. Liefern Sie bessere Produkte!

Lange Build-Zyklen (unabhängig davon, ob es sich um Build, Build und Test oder Build und Test und andere Aufgaben handelt) wirken sich negativ auf die Produktivität eines Teams aus. Wie Sie in diesem Dokument gelesen haben, wirken sich lange Build-Zeiten auch auf das Qualitätsniveau eines Unternehmens aus.

Jedes Unternehmen kann sich dazu entschließen, viel Zeit und Ressourcen in die Verkürzung seines Build-Zyklus zu investieren. Es gibt jedoch Alternativen, mit denen die Build-Zeiten ohne so hohe Investitionen in Zeit und Kapital drastisch verkürzt werden können. Jede Organisation, die ihre Build-Zeit schnell optimieren möchte, sollte Incredibuild in Betracht ziehen. Am besten holen Sie sich direkt eine kostenlose Testlizenz von Incredibuild und installieren sie. Der minimale Aufwand beim Einrichten und Konfigurieren von Incredibuild macht es Ihnen leicht. Sie können es selbst ausprobieren!



Hier können Sie eine kostenlose, ASERVO-exklusiv erweiterte* Testlizenz für Incredibuild oder einen PoC (Proof of Concept) anfordern:

ASERVO Software GmbH

Konrad-Zuse-Platz 8 I 81829 München
Tel.: 089 7167182-40 I Fax: 089 7167182-55
Email: kontakt@aservo.com I www.ASERVO.com

*Die ASERVO-exklusive Incredibuild-Testlizenz gilt für eine unbeschränkte Anzahl von Maschinen.