Nico Orschel

(nico.orschel@aitgmbh.de)
ist Software Process Consultant, Autor und Referent im Umfeld Microsoft ALM bei der AIT GmbH & Co. KG Stuttgart und wurde von Microsoft als Most Valuable Professional (MVP) für Visual Studio ALM ausgezeichnet. Er hilft Unternehmen, auf Basis von Microsoft Visual Studio Team Foundation Server effizienter Software zu entwickeln und zu testen und so ein höheres Qualitätsniveau bei kürzeren Release-Zyklen zu erreichen.

Cloud ist nicht nur etwas für Entwickler und IT-Profis

In der Cloud zu sein beziehungsweise etwas mit der Cloud zu machen, ist hip und modern. Gleich nach Scrum, Agile und DevOps ist die „Wolke" derzeit eines der Schlagwörter in der IT-Welt. Je nach Blickwinkel durchaus mal im positiven wie auch negativen Licht. Erstaunlicherweise taucht der Begriff aber verhalten wenig im Zusammenhang mit Testen auf. In anderen Bereichen wie E-Mail, Webservices, Apps usw. ist es bereits üblich, externe Dienste einzukaufen und diese tief in die eigenen internen Infrastrukturen und Prozesse einzubetten.
An dieser Stelle setzt der Artikel an und zeigt Szenarien auf, in denen es sinnvoll ist, Teile von Testprozessen mit Cloud-Diensten zu ergänzen. Neben den Vorteilen soll auch die Betrachtung der Grenzen nicht zu kurz kommen. Die zentrale Frage ist hier stets: Wann ist die Integration sinnvoll und wann nicht? Inhaltlich werden zunächst die verschiedenen Typen von Cloud-Diensten vorgestellt. Zu den einzelnen Diensttypen werden anschließend ausgewählte Dienste aus der Microsoft Azure Cloud für den Testbereich herausgepickt. Als Szenarien werden exemplarisch Testmanagement, Bereitstellung von Testumgebungen und Lasttests in der Cloud betrachtet.

Typen an Cloud-Diensten

Cloud als Begriff wird recht inflationär genutzt. Jedes Produkt, jede Software, Hardware und selbst Mobiltelefone und Uhren müssen aktuell Cloud-fähig sein. Jeder spricht also über die Cloud, aber alle verstehen etwas anderes darunter. Bevor dieser Artikel auf Cloud-Dienste auf Basis von Microsoft Azure eingeht, werden eingangs zunächst die verschiedenen Cloud-Typen betrachtet. In der Literatur (siehe [Car11], [Rümm12]) werden Cloud-Dienste in drei grundlegende Typen eingeteilt: *Plattform as a Service (PaaS)*, *Infrastructure as a Service (IaaS)* und *Software as a Service (SaaS)* (siehe **Abbildung 1**).

Den meisten Nutzern ist wahrscheinlich IaaS am besten bekannt. Bei diesem Cloud-Typ stellt der Dienstanbieter die Hardware-Infrastruktur zur Verfügung und der Kunde kann seine Systeme direkt darauf betreiben. Ein typisches Beispiel für diese Kategorie ist der Betrieb von virtuellen Maschinen (VM) mit unterschiedlichsten Betriebssystemen bei Microsoft Azure oder Amazon. Der Kunde hat hier die einfache Möglichkeit, Rechenkapazität bedarfsgerecht auch für sehr kurze Zeiträume zu mieten und seine Systeme auf der externen Infrastruktur zu betreiben. Von Vorteil ist hier, dass der Kunde keine eigene Hardware anschaffen muss, aber dennoch eine hohe Flexibilität hat – kann er doch alles nach Bedarf in einer VM installieren. Flexibilität kann dabei Vorteil und gleichzeitig Nachteil sein, Betriebsaufgaben wie Sicherungen, Updates und Skalierung der Dienste und VMs verbleiben in der Verantwortung des Kunden. Der Anbieter verwaltet bei IaaS lediglich die zugewiesenen Hardware-Ressourcen und garantiert deren fehlerfreien Betrieb.

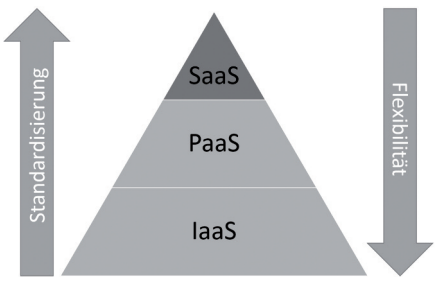

Abb. 1: Cloud-Typen: Flexibilität vs. Standardisierung

Möchte man sich als Unternehmen primär auf die Entwicklung konzentrieren und einfach nur eine verwaltete Plattform nutzen, dann bieten sich Angebote der Kategorie PaaS an. In diesem Fall übernimmt der Dienstanbieter die Verwaltung und den Betrieb der Plattform. Betriebsaufgaben wie Updates, Sicherungen, Sicherheitsmanagement und Skalierung werden dann nicht mehr durch den Anwender, sondern den Dienstanbieter realisiert. Eine Plattform bildet eine Kombination aus Betriebssystem, Datenbanken, Drittanbieter-Komponenten und Frameworks wie PHP, Ruby, Java, .NET usw. Vorteilhaft für den Kunden bei PaaS sind die geringen Wartungsaufwände bei Hard- und Software. Die Vorteile werden allerdings durch eine etwas geringere Flexibilität erkauft. Als Kunde lassen sich nur die vom Dienstanbieter unterstützten bezie-

hungsweise bereitgestellten Schnittstellen, Komponenten und Frameworks verwenden. Ein Beispiel für PaaS aus der Praxis sind die Webhoster.

Der dritte noch verbleibende Cloud-Typ ist SaaS. Bei diesem Typ stellt der Anbieter Applikationen als standardisierte Dienste bereit. Der Anwender kann in nur sehr geringem Umfang Einfluss auf die bereitgestellten Dienste nehmen. Ein Beispiel für diese Art von Diensten ist Office 365 von Microsoft, bei dem SharePoint-Portale und Exchange-Postfächer als Dienst genutzt und in die eigene Infrastruktur integriert werden können.

Nachdem die grundsätzlichen Diensttypen vorgestellt wurden, werden jetzt Dienste in den verschiedenen Kategorien für Tester auf Basis von Microsoft Azure betrachtet. Azure ist in den Projekten der AIT der Favorit, da diese Plattform eine sehr tiefe Integration in die Microsoft Visual Studio Entwicklungs- und Testwerkzeuglandschaft hat.

Testmanagement

Stark vereinfacht betrachtet besteht ein nach ISTQB definierter Testprozess aus fünf Prozessschritten: Planung und Steuerung, Analyse und Design, Realisierung und Durchführung, Auswertung und Bericht sowie Abschluss. In unseren Projekten setzen wir zur Verwaltung von Projekt-, Entwicklungs- und Testartefakten auf die Plattform Microsoft Visual Studio, bestehend aus Visual Studio für die Entwickler, Microsoft Testmanager für die Tester sowie MS Word (siehe [AIT]), Excel und Project für den Projektmanager sowie Web Access für das gesamte Team. Alle Werkzeuge legen ihre Daten im Team Foundation Server (TFS) als zentralem System ab.

Die Ablage in einem zentralen System hat den Vorteil, dass jeder Beteiligte transparent auf alle Daten zu jedem Zeitpunkt zugreifen kann. Der Zugriff ist dabei nicht auf Quellcode beschränkt, sondern gilt auch für Planungsdaten. Letztere heißen im TFS Work Items und können je nach Prozess unterschiedliche Begriffe aufweisen. Beispiele sind Product Backlog Items, Requirements, Bugs, Tasks oder Change Requests. Durch den transparenten Zugriff lassen sich Fehler frühzeitig korrigieren. Man kann hier auch von einem leichtgewichtigen Review-Prozess sprechen.

Testmanagement als Funktionalität im Speziellen findet beim TFS über den dedizierten Microsoft Testmanager (MTM) beziehungsweise web-gestützt über den neuen TFS Web Access Test-Hub statt. Als Verwaltungselemente kommen dabei Testplan, Testsuiten und Testfälle zum Einsatz (Detailartikel zum MTM siehe [Orsch10]). Testpläne gruppieren TFS-Testfälle (Test Cases) nach zeitlichen Aspekten (z. B. Projektphase). Sie kann man vereinfacht als Backlog oder ToDo-Liste der Tester beschreiben. Im Gegensatz zum Umfang eines Testplanes im klassischen Sinn zum Beispiel nach ISTQB wird hier nur eine Untermenge in Werkzeugform abgebildet. Die nächste Stufe der Gruppierung nach fachlichen Aspekten (Funktionsbereiche der Anwendung, Module usw.) findet über Testsuiten statt. Die eigentlichen manuellen Testfälle werden in Form von Testschritten und erwartetem Ergebnis in Test Case Work Items abgelegt. Der Mechanismus ist technologisch ähnlich zu den anderen Work Item-Typen wie Anforderungen oder Bugs. Test Cases haben hier lediglich spezielle Felder.
Damit der MTM funktioniert, ist ein lauffähiger TFS Voraussetzung. Die lokale Bereitstellung eines TFS kann für kleine Teams beziehungsweise sehr kurze Projekte verhältnismäßig aufwendig sein. Es gilt Hard- und Software zu beschaffen und die Installation nach aktuellen Microsoft Best Practices durchzuführen. Demgegenüber kann sich der Aufwand für eine Bereitstellung mittels Cloud-Angeboten auf ein Minimum begrenzen, denn Microsoft bietet auch den TFS unter dem Namen Visual Studio Online (VSO) als Cloud-Dienst auf der Azure-Plattform an. Entsprechend der vorgestellten Kategorisierung handelt es sich hier um SaaS. Funktional entspricht der Dienst dabei einem lokalen TFS-Server. Es stehen alle Funktionalitäten wie Work Items, Source Control, Testmanagement und Web Access zur Verfügung (siehe Abbildung 2).

Aufgrund des Umstandes, dass VSO nicht lokal läuft, sondern auf Microsoft Azure betrieben wird, gibt es für VSO keine Möglichkeit, lokale Domäneninformationen zur Anmeldung zu verwenden. Positiv formuliert kann man auch sagen, dass keine Domäne zur Anmeldung notwendig ist. Die Anmeldung der Nutzer gegenüber dem Dienst erfolgt standardmäßig via Microsoft Account (ehemals Live ID bzw. Microsoft Passport). Neben dieser Möglichkeit kann man seit ein paar Wochen auch ein Azure Active Directory (AAD) nutzen. Dies ist vereinfacht gesprochen ein spezieller Domänentyp für Azure-Dienste. Eine Synchronisierung von lokalen Domäneninformationen zu AAD ist möglich, sodass die Anmeldedaten lokal und remote für den Nutzer stets gleich sind. Die Steuerung der einzelnen Berechtigungen innerhalb der jeweiligen kundenspezifischen VSO-Instanz (Team-Projekt, Source Control, Builds, Work Items usw.) entspricht denen eines lokalen TFS-Servers. Die Verwendung von Microsoft Accounts hat in der täglichen Projektwelt den Charme, dass sich Kunden und externe Mitarbeiter sehr einfach in Projekte integrieren lassen. Externe Nutzer benötigen zur Zusammenarbeit keine lokalen Accounts mehr und müssen deshalb auch

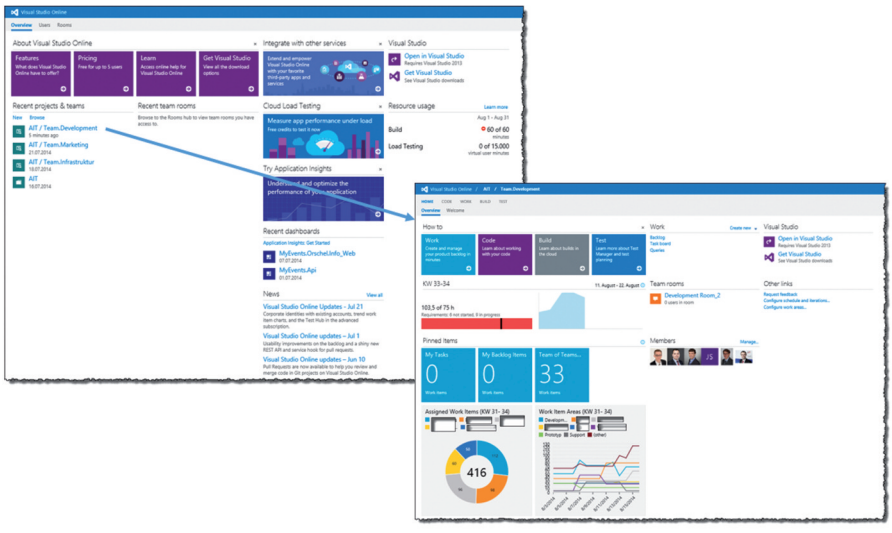

Abb. 2: Visual Studio Online Portal und Web Access

keinen Zugriff auf lokale Netzwerkressourcen haben.

Ein großer Vorteil sind geringe Lizenzkosten, denn bei den meisten Nutzern ist der Microsoft Account auch an eine MSDN Subscription gebunden. Jeder Kunde mit einer aktiven Subscription ist immer für alle eigenen und fremden (Kunden-)VSO-Instanzen lizenziert. Die Notwendigkeit der Beschaffung von Zugriffs(CAL)- und Serverlizenzen für lokale TFS-Systeme entfällt.

Unterschiede zwischen lokalem TFS und VSO betreffen derzeit im Wesentlichen die fehlende Möglichkeit zur Anpassung von Work Items und fehlende Unterstützung von virtuellen Lab Management-Testumgebungen (Detailartikel zu Lab Management siehe [Orsch11]). Die Anbindung von virtuellen Maschinen über den Mechanismus der Standardumgebungen ist aber auch bei VSO möglich.

Die VSO-Plattform hat nicht nur Einschränkungen, sondern ist in einigen Punkten einem lokalen TFS sogar überlegen. Microsoft lebt das Motto *„Mobile First, Cloud First"* ([Nad14]). Neue Produkt-Funktionalitäten und -Releases werden erst als Cloud-Version und dann als lokales Produkt veröffentlicht. In der VSO-Welt ist dieses Vorgehen bereits fester Bestandteil seit dem Visual Studio 2012 Release (vor ca. 2 Jahren). Die TFS-Produktgruppe rollt alle drei Wochen ein neues Release auf die VSO-Plattform aus. Kleine und größere Änderungen und Neuheiten finden so sehr schnell den Weg zum Nutzer. Als Beispiel aus der jüngeren Vergangenheit sei hier aus dem Testbereich die Wiederverwendbarkeit von Testdaten durch *Shared Parameters* genannt (siehe [VSOGA]). Dieses Feature war ca. acht Wochen vor dem offiziellen Update Release bereits auf VSO für alle Nutzer ohne eigenes Zutun zugänglich.

Bereitstellung von Testumgebungen

Neben der Bereitstellung eines TFS ist in vielen Projekten die Bereitstellung von Rechnern zum Testen und Entwickeln ein Problem. Die Bandbreiten möglicher Probleme reichen vom fehlenden Budget, Hardware-Ressourcen, Software-Lizenzen bis zu komplexen und langwierigen Bereitstellungszeiten. Teilweise wurden Maschinen erst mit mehrwöchiger Verzögerung bereitgestellt, sodass Projekte faktisch schon in der Endabnahmephase steckten.

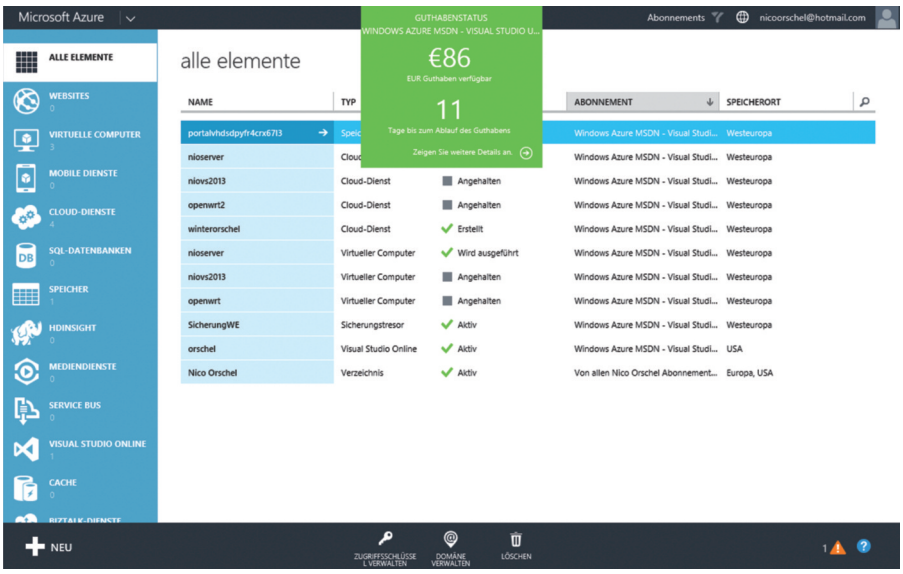

Abb. 3: Schaltzentrale - Azure Management Portal

Auch beim Thema Infrastrukturbereitstellung für Testmaschinen kann Microsoft Azure unterstützen. Auf Azure gibt es die einfache Möglichkeit, webgestützt per Azure Management-Portal (siehe **Abbildung 3**) virtuelle Maschinen auf Basis von Vorlagen anzulegen. Dieser Aspekt von Azure entspricht dem Cloud-Typ IaaS. Am Beispiel eines TFS Build-Servers haben wir diesen Prozess Schritt für Schritt auf dem AIT-Weblog (siehe [Wehr14]) dokumentiert.

Microsoft stellt in seiner Gallery verschiedene Vorlagen für unterschiedlichste Bereiche bereit. Die Bandbreiten der Templates reichen dabei von unterschiedlichsten Windows-Systemen, Oracle-Datenbanken bis hin zu Visual Studio-, Java-, NOSQL DB-, SharePoint- und sogar Linux-Versionen. Mit wenigen Mausklicks kann man sich hier als Tester eine Maschine bereitstellen lassen. Die Bereitstellung einer kompletten SharePoint-Testumgebung dauert beispielsweise ca. eine Stunde. Selbst installiert, dauert dies normalerweise mehrere Tage. Neben den Vorlagen haben Sie auch alternativ die Option, eigene Images hochzuladen. Voraussetzung ist lediglich, dass die Images auf Hyper-V erzeugt wurden.

Der Zugriff auf diese Maschinen erfolgt dabei über das Internet mit bekannten Protokollen wie Remote Desktop, SSH oder VNC. Standardmäßig laufen die VMs in Azure getrennt vom lokalen Firmennetzwerk und sind frei vom Internet erreichbar. Seit einigen Monaten gibt es auch bei Azure VMs die Möglichkeit, diese an das eigene Firmennetzwerk über eine VPN-Verbindung anzubinden. Die VMs in der Cloud sind dadurch wie lokale Rechner erreichbar. Sie können lokale Ressourcen wie DNS und Active Directory nutzen. Um die Sicherheit noch weiter zu erhöhen, können diese VMs auch in einem privaten Verbund ohne direkten Internet-Zugang betrieben werden. In diesem Fall spricht man von einer Hybrid Cloud (siehe [Rous]).

Vergleicht man an dieser Stelle wieder lokale Virtualisierung mit Azure VM, dann fällt aktuell nur eine Limitierung im Zusammenhang mit Testen auf. Beim Testen kann der Einsatz von Snapshots eine Menge Arbeit beim Zurücksetzen von Testumgebungen sparen. Direkte Snapshots werden bei Azure VMs nicht unterstützt. Azure hat dafür einen eigenen Ansatz. Snapshots sind in Azure stets eigenständige VM-Kopien.

Eine noch verbleibende Frage bei den Azure VMs betrifft die Kosten. VMs werden bei Microsoft Azure nach verbrauchten Ressourcen (CPU, Arbeitsspeicher), Speicherplatz und ausgehendem Netzwerkverkehr berechnet (siehe [AzurePC]). Vielen MSDN-Abonnenten unbekannt ist hier jedoch, dass er über ein monatliches Frei-Budget zwischen 75 und 150 $ für beliebige Azure-Dienste verfügt. Das Budget kann genutzt werden, um ca. zwei VM mit zwei Kernen und 7 GB RAM einen ganzen Monat laufen zu lassen. Abonnenten können sogar noch mehr Maschinen pro Monat laufen lassen, wenn sie die Maschine nach erledigter Arbeit am Abend einfach herunterfahren. Verrechnet wird

nur die Rechenleistung für laufende Rechner.

Lasttests

Lasttests sollten in der aktuellen Welt von Smartphones und Webanwendungen zur Standard-Methodik gehören. Unter Lasttests fallen prinzipiell zwei Aspekte:

- Test der Applikation unter extremen Lastbedingungen und ihres korrekten Verhaltens beziehungsweise
- Test einzelner Transaktionen auf das gewünschte nicht-funktionale Verhalten wie Verarbeitungsdauer.

Visual Studio bietet schon seit vielen Jahren die Möglichkeit, im Rahmen der Visual Studio Ultimate Edition Lasttests mit einer unlimitierten Anzahl an Benutzern zu erstellen.

Die Basis für Lasttests bilden dabei Web-Performance- und Unit-Tests auf Basis von MSTest. Web-Performance-Tests testen Applikationen auf Basis von http-Aufrufen und sind deshalb auf das Testen von Web-Applikationen beziehungsweise -Diensten beschränkt. Web-Performance-Tests können entweder per Recorder oder via .NET-Code erstellt werden.

Eine etwas unterschätzte Art, Lasttests aufzubauen, besteht darin, MSTest-Unit-Tests als Basis zu verwenden. In Unit-Tests kann jegliche Art von API durch Code angesprochen werden. Sind Web-Performance-Tests noch bei der Anwendbarkeit auf reine Webanwendungen beschränkt, kann mit Unit-Tests faktisch jedes Protokoll einem Lasttest zugeführt werden. Beispiele sind hier Tests von proprietären Protokollen, wie denen von SAP- und CRM-Systemen, oder offenen Protokollen, wie REST- und JSON-basierende Systeme.

Neben dem reinen Aufbau der Lasttests ist ein wichtiger Faktor die Infrastruktur zum Ausführen von Lasttests. Lasttests lassen sich auf drei Arten ausführen: lokal im Visual Studio, remote über Test Agents und seit VS 2013 auch über VSO. Die empfohlene Anzahl an virtuellen Nutzern pro CPU-Kern ist 500 (siehe [MSDN]). Die Gesamtnutzeranzahl kann mit weiteren Test Agents beliebig angehoben werden. Die Lizenzierung der Agents ist praktisch kein Problem, sind doch mit VS 2013 Ultimate unlimitiert viele virtuelle Lasttest-Nutzer lizenziert. Der begrenzende Faktor ist vielmehr die Anzahl der laufenden Server und deren Setup. Die Aufwände für eine lokale Bereitstellung können sich dabei schnell auf mehrere Tage addieren.

Mit Lasttests via Azure kann man diese Aufwände gerade für ad hoc beziehungsweise unregelmäßige Einsätze reduzieren. Das TFS-Team stellt die notwendige Infrastruktur auf Azure als PaaS zur Verfügung. Das Schöne an diesem Dienst ist, dass je nach Anzahl gewünschter virtueller Nutzer die notwendigen Ressourcen flexibel bereitgestellt werden. Aktuell unterstützt Azure ca. 10.000 virtuelle Nutzer (20 Knoten x 500 Nutzer). Das Limit soll in naher Zukunft noch auf ca. 25.000 Nutzer erhöht werden (50 Knoten). In der Praxis sind die Lasttests nur durch das zur Verfügung stehende Budget limitiert. Jeder MSDN-Abonnent hat pro Monat 15.000 Virtual User Minutes (VUM) zur Verfügung. 15.000 VUM entspricht der Ausführung eines Lasttests mit 1.000 virtuellen Nutzern für 15 Minuten. Wird mehr Laufzeit benötigt, so ist dies separat vergütungspflichtig (siehe [VSOPD]). Diese natürliche Begrenzung sorgt auch indirekt dafür, dass der Dienst nicht für DoS-Attacken genutzt werden kann.

Um nun einen Lasttest auf Azure auszuführen, ist gar nicht viel Arbeit notwendig. Zum Start ist lediglich eine separate Test-Einstellungsdatei (testsettings) notwendig. In dieser Datei wird „Azure“ als Verbindungstyp ausgewählt und schon ist der Lasttest bereit für Azure. Wird anschließend ein Lasttest im Visual Studio gestartet, so wird dieser Lasttest in den Azure-Dienst hochgeladen und ausgeführt. In regelmäßigen Abständen während der Ausführung werden dem Nutzer dann ausgewählte Metriken (z. B. Response Time) im Visual Studio angezeigt (siehe **Abbildung 4**). Am Ende der Ausführung wird ein detailliertes Log für weitere spätere lokale Analysen im Visual Studio angeboten. Nach der Ausführung werden die Analysedaten im Cloud-Dienst gelöscht.

Zu guter Letzt wieder die Frage: Wo liegen die wichtigen Einschränkungen gegenüber einer lokalen Ausführung? Anwendungen, welche über den Dienst getestet werden sollen, müssen über das Internet erreichbar sein. Der Zugriff auf Anwendungen in geschlossenen Netzwerken ist nicht möglich. Wird eine Anwendung extern verfügbar gemacht, so sollten Maßnahmen zum Schutz der Daten implementiert sein (z. B. Verschlüsselung). Ein weiterer Punkt betrifft die Erfassung von Systemkennzahlen. Die Azure-Infrastruktur hat ohne weitere Anpassungen keine Möglichkeit, Windows-Performance-Counter (Speicherverbrauch, CPU-Auslastung sowie weitere Metriken) über das Internet auszulesen. Die Analyse ist hier standardmäßig auf extern messbare Metriken (Response-Zeiten, Transfervolumen usw.) beschränkt. Mit dem kürzlich veröffentlichten VS 2013 Update 3 wurde dieses Limit teilweise aufgehoben. Integriert man auf dem zu testenden System das Microsoft-Telemetriedaten-Framework *Application Insights* (siehe [VSOAI]), so lassen sich die Telemetriedaten aus den beiden Cloud-Diensten Load-Testing und Application Insights lokal zusammenzuführen. Durch die Zusammenführung ist das angezeigte Endergebnis eines Azure-Lasttests sehr ähnlich zu einer reinen lokalen Ausführung. Application Insights als

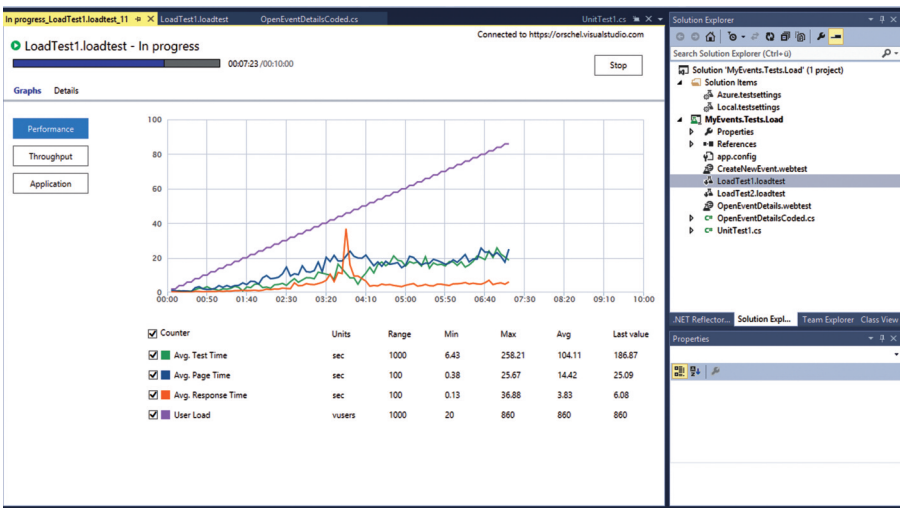

Abb. 4: Azure-Lasttest in Visual Studio 2013

Cloud-Dienst ermöglicht das Einsammeln und Verarbeiten von Telemetriedaten wie Nutzungsverhalten, Exception-, Log- und Trace-Daten einer Anwendung.

Datenschutz

Datenschutz ist eine wichtige und kontroverse Fragestellung im Zusammenhang mit den vorgestellten Diensten. Diskussionen rund um dieses Thema sind meistens von Extremen geprägt. Aus Sicht des Autors ist Schutz von Projektdaten mehrstufig und anbieterunabhängig zu betrachten. Eine grundsätzliche Fragestellung steht hier bei allen Diensten am Anfang: Vertraue ich als Kunde dem Dienste-Anbieter? Als Kunde muss man sich bei jedem Anbieter stets vor Augen führen: Der Anbieter kontrolliert die Plattform und kann theoretisch alle Daten einsehen. Der zugrunde liegende Cloud-Typ ist dabei nebensächlich. Die Anbieter versuchen, das Problem durch vertragliche Zusicherungen gegenüber ihren Kunden zu lösen. Im Fall der vorgestellten Azure-Dienste finden Sie beispielsweise unter [VSOFAQ] Zusicherungen zu Datenschutz und Eigentum. Datenschutzrichtlinien werden nach Industriestandards auf der gesamten Azure-Plattform eingehalten und alle Daten verbleiben stets im Kundeneigentum.

Die zweite grundlegende Frage ist: Zwingen mich gesetzliche oder vertragliche Anforderungen, einen Dienst lokal zu betreiben, sodass Sicherungsmaßnahmen (z. B. Verschlüsselung, Backup) auf allen Schichten implementiert und überwacht werden können. Bei Cloud-Diensten lassen sich diese nur partiell umsetzen. Dies ist proportional zur Flexibilität der Plattform (siehe Abbildung 1). Mehr Flexibilität bedeutet mehr kundenseitige Verantwortung und Aufwand. Es ist hier letztendlich eine Abwägung zwischen Aufwand oder Nutzen.

Die dritte Frage im Kontext Schutz ist: Kann ich durch Prozessänderungen die Speicherung von kritischen Daten in meinem Entwicklungs- und Testprozess vermeiden? Exemplarisch sei hier die Anonymisierung von Testdaten oder die Verwendung von dedizierten Testumgebungen mit Nicht-Produktions-Nutzerkonten genannt. Beides sind Vorgehen, welche auch bei lokalen Diensten selbstverständlich sein sollten.

Fazit

Microsoft Azure-Dienste haben sich im Verlauf der letzten Jahre massiv weiterentwickelt. Die im Artikel vorgestellten Dienste hatten dabei primär den Tester im Blick. Es wurden exemplarisch drei unterschiedliche Dienste vorgestellt. Den Anfang markierte VSO als SaaS-Alternative zu einem lokalen TFS. Alle Projektdaten aus den Bereichen Work Items, Source Control, Build und Testmanagement liegen dabei auf VSO. Dem Tester bietet dies die Möglichkeit, Testmanagement mit den Visual Studio-Werkzeugen ohne viel Aufwand und eigene Infrastruktur durchzuführen.

Als Alternative zu einer lokalen Bereitstellung von Testumgebungen wurden zweitens Azure VMs (IaaS) vorgestellt. Mit einer MSDN-Subscription sind VSO und Azure VM sogar zum Großteil kostenlos nutzbar.

Den Abschluss der vorgestellten Dienste bildete der Azure-Lasttest-Dienst als PaaS-Alternative zum lokalen Betreiben einer großen Lasttest-Infrastruktur.

Eine spannende und kontroverse Fragestellung bei diesen Diensten ist allerdings offen: Wie viel Cloud darf es denn sein? Pauschale Antworten gibt es nicht und Diskussionen enden hier oft ohne Ergebnis in einer großen religiös angehauchten Diskussion.

Als Antwort gibt es zwei Extreme: „Absolut keine Cloud-Dienste“ oder „Alles in die Cloud“. Es ist jedoch nicht förderlich, die Frage ohne Berücksichtigung von Grautönen zu beantworten. In Projekten ist es letztendlich wie immer im Leben: Man trifft sich irgendwo in der Mitte. Eine mögliche Option für kritische Bereiche besteht darin, kritische Kunden-Projektdaten im lokalen TFS abzulegen. Die Bereitstellung der Testumgebungen für Abnahme- und Lasttests wird aber selektiv im Azure betrieben. Faktoren für die Beurteilung, wo, wann und wie ein Dienst integriert werden kann, sind dabei letztendlich abhängig vom Budget, von der Art der Daten (datenschutzkritisch, vertraulich, unkritische Daten) und ob die notwendigen Ressourcen und Dienste eventuell bereits lokal vorhanden sind. ■

Literatur & Links

[AIT] AIT WordToTFS, siehe: http://www.aitgmbh.de/nc/downloads/team-foundation-server-tools/ait-wordtotfs.html

[AzurePC] Azure Pricing Calculator, siehe: http://azure.microsoft.com/en-us/pricing/calculator/

[Car11] J. Caruso, IaaS vs. PaaS vs. SaaS - Cloud computing flavors designed to meet almost any need, 2011, siehe: http://www.networkworld.com/article/2182527/virtualization/iaas-vs--paas-vs--saas.html

[MSDN] Test Controller and Test Agent Requirements for Load Testing, siehe: http://msdn.microsoft.com/en-us/library/ff937706.aspx

[Nad14] S. Nadella, Mobile First, Cloud First Press Briefing, 2014, siehe: http://www.microsoft.com/en-us/news/speeches/2014/03-27nadella.aspx

[Orsch10] N. Orschel, Ausweg aus der Kommunikationskrise oder das Ende von „Bei mir funktioniert's“?, Advertorial im Online-Themenspecial Testing 2010 von OBJEKTspektrum, siehe: http://www.sigs.de/publications/os/2010/Testing/orschel_OS_TESTING_2010.pdf

[Orsch11] N. Orschel, Kürzere Testvorbereitungsphasen durch integrierte Testlabore, Advertorial im Online-Themenspecial Testing 2011 von OBJEKTspektrum, siehe: http://www.sigs-datacom.de/fileadmin/user_upload/zeitschriften/os/2011/Testing/orschel_OS_testing_11.pdf

[Rous] M. Rouse, hybrid cloud, siehe: http://searchcloudcomputing.techtarget.com/defintion/hybrid-cloud

[Rümm12] Th. Rümmler, Cloud Computing – was bringt's?, 2012, siehe: http://blog.aitgmbh.de/2012/10/08/cloud-computing-was-bringts/

[VSOAI] Application Insights for Visual Studio Online, siehe: http://msdn.microsoft.com/en-us/library/dn481095.aspx

[VSOFAQ] Frequently asked questions - Security & Privacy, siehe: http://www.visualstudio.com/en-us/support/faq-vs.aspx

[VSOGA] General Availability of Visual Studio Online, siehe: http://www.visualstudio.com/en-us/news/2014-apr-3-vso.aspx

[VSOPD] Visual Studio Online Pricing Details, siehe: http://azure.microsoft.com/en-us/pricing/details/visual-studio-online/

[Wehr14] B. Wehrle, Ideen für die Verwendung des MSDN Azure Freikontingents: Bereitstellung eines Visual Studio Online Build-Servers, 2014, siehe: http://blog.aitgmbh.de/2014/07/09/ideen-fr-die-verwendung-des-msdn-azure-freikontingents-bereitstellung-eines-visual-studio-online-build-servers/