bbv
MAKING VISIONS WORK.

BOOKLET

SOFTWAREMIGRATION NACH WINDOWS AZURE

PROFITIEREN SIE VON UNSERER ERFAHRUNG!

Kontakt Schweiz
bbv Software Services AG
Blumenrain 10
6002 Luzern
Telefon: +41 41 429 01 11
E-Mail: info@bbv.ch

Kontakt Deutschland
bbv Software Services GmbH
Agnes-Pockels-Bogen 1
80992 München
Telefon: +49 89 452 438 30
E-Mail: info@bbv.eu

INHALT

1	Einleitung	5
2	Stufe 1: Hosting in Windows Azure	8
2.1	Erstellen eines Accounts in Windows Azure	9
2.2	Migration von Code	9
2.3	Migration von Daten	10
2.4	Kosten und Abrechnung	11
3	Stufe 2: Anpassungen an die Cloud	12
3.1	Sicherheit	13
3.2	Lokaler Festplattenplatz	13
3.3	Automatisches Deployment	14
3.4	Latenzzeiten	14
3.5	Diagnostik/Monitoring	15
3.6	Internationalisierung	15
3.7	Kostenoptimierungen	16
3.8	Strukturierte Datenspeicherung	16
3.9	Integration	17
4	Stufe 3: Skalierbarkeit	20
4.1	Mehrere Instanzen	21
4.2	Load-Balancing	21
4.3	Authentifizierung und Autorisierung	22
4.4	Skalierbarkeit	23
5	Stufe 4: Software-as-a-Service	25
5.1	Multi-Instanz vs. Multi-Tenancy	26
5.2	Mandantenfähigkeit/Multi-Tenancy	26
5.3	Partitionierung des Datenbestandes	26
5.4	Anpassbarkeit/Konfigurierbarkeit	27
5.5	Self-Service on Demand	27

6	Abschliessende Bemerkungen	29
7	Anhang	30
7.1	Autor	31
7.2	Literatur	31

1 EINLEITUNG

Cloud Computing ist in aller Munde, insbesondere in den letzten Jahren entstand geradezu ein Hype um diese Technologie. Microsoft ist mit der Azure-Plattform auf diesen Zug aufgesprungen und bietet inzwischen das zurzeit kompletteste Angebot im Bereich Public Cloud. Was sich genau hinter Azure verbirgt, stellt das bbv-Booklet «Die Windows Azure Plattform» vor.

Das vorliegende Booklet zeigt einen möglichen Weg, das Angebot der Azure-Plattform für die Migration von bestehenden Applikationen oder für Neuentwicklungen in der Cloud zu nutzen. Das Vorgehen wird dabei in vier Stufen unterteilt, welche hier einzeln behandelt werden:

Stufe 1: Hosting in Windows Azure
Stufe 2: Anpassung an die Cloud
Stufe 3: Skalierbarkeit
Stufe 4: Software-as-a-Service

Diese vier Stufen stehen jeweils für einen bestimmten Integrationsgrad: Während man mit einem einfachen Hosting in Azure lediglich einen kleinen Teil der Möglichkeiten dieser Cloud nutzt, ist der Integrationsgrad bei einer mandantenfähigen Lösung, welche als Software-as-a-Service vertrieben wird, entsprechend sehr viel höher. Ein niedriger Integrationsgrad bedeutet, dass kaum von den Skaleneffekten der Cloud profitiert werden kann. Ein hoher Integrationsgrad wiederum hat den Nachteil eines hohen Daten- und Code-Lock-ins. Das bedeutet, dass das Wechseln zwischen On-Premise und Cloud oder unter verschiedenen Cloud-Anbietern dann sehr viel schwieriger und aufwendiger wird.

Der Aufwand, eine bestimmte Stufe zu erreichen, wird tendenziell immer grösser, je höher man kommt. Es ist jedoch nicht unbedingt empfehlenswert, diese Stufen stur zu durchlaufen. Es ist lohnenswerter, sich aus den einzelnen Punkten eine eigene Migrations- bzw. Entwicklungsstrategie für das konkrete Projekt zusammenzustellen. Wenn bereits zu Beginn bekannt ist, dass eine mandantenfähige SaaS-Lösung (Stufe 4) das Ziel ist, sollte auch die Architektur entsprechend ausgerichtet werden. Dieses Booklet kann Ihnen dabei helfen.

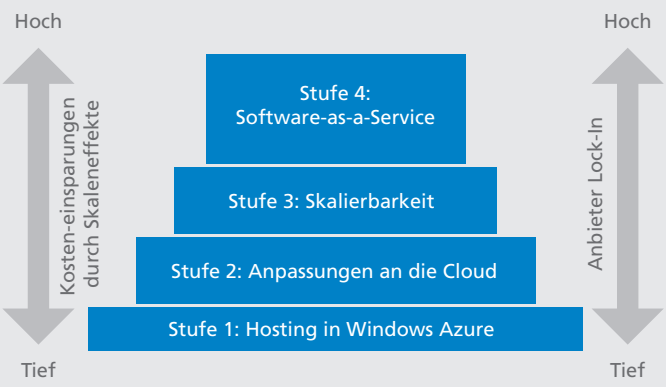


Abbildung 1: Integrationsgrade der vier Stufen.

2 STUFE 1: HOSTING IN WINDOWS AZURE

Ziel der ersten Stufe ist es, eine erste Version der Anwendung in Windows Azure zu betreiben. Dabei spielt es keine Rolle, ob es sich um eine bestehende Anwendung oder den Prototypen einer Neuentwicklung handelt.

Im Folgenden werden die notwendigen Schritte erläutert, um Stufe 1 zu erreichen.

2.1 ERSTELLEN EINES ACCOUNTS IN WINDOWS AZURE

Zunächst muss man sich bei Windows Azure anmelden und einen Account erstellen. Wie die meisten anderen Cloud-Produkte unterstützt auch die Azure-Plattform «Selfservice on Demand», das bedeutet, dass keine menschliche Interaktion seitens Microsoft für den Anmeldeprozess nötig ist. Es genügt dazu eine Windows Live-ID und eine gültige Kreditkarte. Es lohnt sich, vor der Anmeldung die verschiedenen Promotionen von Windows Azure, beispielsweise für Microsoft Partner oder für Inhaber einer MSDN-Subscription, näher anzuschauen.

Anmelden kann man sich unter
www.microsoft.com/windowsazure.

2.2 MIGRATION VON CODE

Ist der Account erstellt, kann mit der Migration der Applikation begonnen werden. Die Migration von Code erfolgt durch die Installation eines entsprechenden Software-Development-Kits auf der Entwicklermaschine. In Azure gibt es diese SDKs für .NET, PHP, Java und Ruby[1].

Die Migration erfolgt, indem die Applikation oder Teile davon innerhalb einer «Web Role» oder einer «Worker Role» paketiert und über das Management-Portal oder direkt aus der IDE nach Windows Azure hochgeladen werden. Praktische Beispiele für das Paketieren und Deployen von Web oder Worker Roles finden Sie im Internet[2].

[1] http://www.microsoft.com/windowsazure/sdk
[2] z. B. unter http://msdn.microsoft.com/en-us/magazine/ee336122.aspx

In Stufe 1 und 2 wird mit nur einer einzigen virtuellen Maschine (Instanz) pro Role gearbeitet, da Applikationen dieser Stufen in den meisten Fällen noch nicht bereit für das Load Balancing sind.

2.3 MIGRATION VON DATEN

Typischerweise verfügen Unternehmensapplikationen über ein Datenbank-Backend. Da die Datenbank noch nicht in der Cloud läuft, muss diese ebenfalls migriert werden. Im Fall von Microsoft SQL Server ist das in den meisten Fällen recht schnell möglich, da mit SQL Azure ein zu grossen Teilen kompatibles Datenbanksystem in der Cloud existiert. Zunächst muss im Management-Portal ein entsprechender Datenbankserver und eine Datenbank angelegt werden. Zu beachten gilt, dass Applikation und Datenbank im selben Datencenter gehostet sein sollten, sowohl aus Kosten- als auch aus Performancegründen. Für die einfache Migration der Daten von SQL Server nach SQL Azure dient das Tool «SQL Azure Migration Wizard», das auf Codeplex[3] gehostet wird. Nach erfolgter Migration muss in der Konfiguration der Applikation lediglich der Datenbank-Connection-String angepasst werden.

[3] http://sqlazuremw.codeplex.com

2.4 KOSTEN UND ABRECHNUNG

Da die Applikation nun in der Cloud läuft, summieren sich ab sofort auch die anfallenden Kosten. Gerade zu Beginn des Entwicklungsprozesses ist es wichtig, diese Kosten im Auge zu behalten, damit man am Ende des Monats keine unliebsamen Überraschungen erlebt. Die ausgestellten Abrechnungen, aber auch die laufenden Kosten können jederzeit eingesehen werden[4]. Hier muss man sich mit derselben Windows Live-ID anmelden, auf welche der Azure-Account lautet.

Damit ist Stufe 1 der Migration erreicht. Die Applikation läuft in der Cloud und kann auf die eigene Datenbank zugreifen, welche ebenfalls in der Cloud gehostet ist.

[4] https://account.windowsazure.com/subscriptions

3 STUFE 2: ANPASSUNGEN AN DIE CLOUD

Obschon das Umfeld von Windows Azure zu grossen Teilen zu Microsoft basierten On-Premise-Infrastrukturen kompatibel ist, gibt es einige Punkte, die man beachten und gegebenenfalls anpassen sollte.

3.1 SICHERHEIT

Da die Anwendung nun öffentlich verfügbar und damit ein potenzielles Ziel für Angreifer ist, sollte sie entsprechend geschützt werden. Die Windows-Azure-Plattform verfügt bereits über einige Schutzmassnahmen, die unter dem Begriff Plattformsicherheit zusammengefasst sind. Andere Schutzmechanismen müssen jedoch selber implementiert werden. Welche Sicherheitsmechanismen bereits vorhanden sind, zeigt das Dokument «Security Best Practices For Developing Windows Azure Applications»[5] und dort insbesondere die Sicherheitsmatrix in Anhang B. Es wird unter anderem dringend empfohlen, die Datenübertragung mit SSL zu sichern. Dabei ist es notwendig, eine eigene URL für die Applikation zu reservieren und nicht *.cloudapp.net zu verwenden, da auf Letztere kein gültiges SSL-Zertifikat ausgestellt werden kann.

3.2 LOKALER FESTPLATTENPLATZ

Die nächste Herausforderung liegt darin, dass der lokale Festplattenplatz einer Role-Instanz flüchtig ist. Das bedeutet, wenn eine Role-Instanz (virtuelle Maschine) in Windows Azure gelöscht wird, gehen sämtliche lokal gespeicherte Daten verloren. Das mag im Fall von temporären Dateien oder Logdateien weniger schlimm sein als wenn produktive Daten verloren gehen, ärgerlich ist es in jedem Fall.

Die bevorzugte Dateiablage in Windows Azure ist in diesem Fall der BLOB-Storage. Dieser kann via REST-Interface angesprochen werden. Der BLOB-Storage bietet Platz für bis zu 200 GB grosse Dateien (1 TB für Page BLOBs) und ist in Containern strukturiert. Die Container und BLOBs können mit individuellen Schreib- und Leserechten versehen werden.

[5] http://www.microsoft.com/download/en/details.aspx?id=7253

3.3 AUTOMATISCHES DEPLOYMENT

Schon früh im Projekt sollte man sich Gedanken dazu machen, wie häufig man während der Entwicklungszeit nach Windows Azure deployen möchte. Für Test- oder Integrationsszenarien wurde von Microsoft die «Extra-Small»-Instanzgrösse entwickelt. Sie kostet rund 1 USD pro Tag, kann aber nicht skaliert werden. Je nach MSDN-Subscription (Ultimate, Premium oder Professional) ist eine bestimmte Anzahl Stunden der Small- oder der Extra-Small-Instanz inbegriffen.

Das Management-API von Windows Azure ist vollständig skriptfähig. Alle Aktionen, die im Management-Portal möglich sind, können auch mit Skript-Werkzeugen wie MSBuild ausgeführt werden. Microsoft stellt Beispiele für ein automatisiertes Deployment nach Windows Azure zur Verfügung[6].

3.4 LATENZZEITEN

In weiten Teilen Europas und Nordamerikas können jeweils zwei Azure-Datencenter innerhalb von maximal 100 ms Latenzzeit erreicht werden. Obschon das relativ schnell ist, dauert es sehr viel länger als bei einer Applikation, die bislang nebenan im unternehmenseigenen Datencenter stand. Dies kann dazu führen, dass ein GUI einer Applikation, welches bislang flüssig lief, nun plötzlich bei der Absetzung eines Requests zum Middletier einfriert. Die Verbesserung der Antwortzeiten ist selbst in einem hochskalierbaren, hochverfügbaren Umfeld wie Windows Azure nicht beliebig möglich, da sie physikalischen Gesetzen unterworfen ist. Dennoch gibt es einige Tipps und Tricks, wie die Performance verbessert werden kann .

[6] http://msdn.microsoft.com/en-us/library/ff803365.aspx
[7] http://channel9.msdn.com/Events/TechEd/NorthAmerica/2011/COS401

Unter anderem sind das:

- Reduktion der Latenz durch die Einführung des asynchronen Design-Patterns
- Optimierung des Datenzugriffs
- Tunen der Applikationsperformance
- Verwendung des Azure Content Delivery Networks (CDN) für BLOBs
- Integration des Windows Azure AppFabric Caching

3.5 DIAGNOSTIK/MONITORING

Das Verhalten herkömmlicher Applikationen wird in der Regel in diversen Logfiles aufgezeichnet, sei es im IIS-Log, im System-Log oder in einer applikationseigenen Logdatei. In Windows Azure ist der Zugriff auf diese Logdateien jedoch kompliziert, da sie auf der virtuellen Maschine liegen und somit nur via Remote-Desktop gelesen werden können und verloren gehen, wenn die virtuelle Maschine gelöscht wird. Die Entwickler der Azure-Plattform mussten sich daher etwas Neues einfallen lassen. In Windows Azure ist es möglich, unterschiedliche Daten zu erfassen und beispielsweise im Table-Storage abzulegen. Diese Daten können für die Fehlerbehebung, beim Messen der Performance, beim Monitoring des Ressourcenverbrauchs, bei der Datenflussanalyse, bei der Kapazitätsplanung oder fürs Auditing verwendet werden.

3.6 INTERNATIONALISIERUNG

Auf stackoverflow.com[8] beklagt sich ein Entwickler, dass seine Applikation, die zuvor auf einem Server in Italien gehostet wurde, in Windows Azure nun plötzlich eine andere Zeitzone besitzt, nämlich diejenige der koordinierten Weltzeit UTC, welche zur MEZ

[8] http://stackoverflow.com/questions/6504153/azure-timezone-utc-problem-how-to-solve

um eine Stunde, im Sommer um zwei Stunden verschoben ist. Um dieses Problem muss man sich also in jedem Fall kümmern, auch wenn die eigene Applikation nur regional eingesetzt werden soll.
Erstreckt sich der Markt des eigenen Produkts über Kulturgrenzen hinweg, muss man sich nochmals intensiv mit der Internationalisierung und den damit verbundenen Themen wie Mehrsprachigkeit, Zeit- und Nummernformate sowie Währungen auseinandersetzen. Hinzu kommen kulturelle Unterschiede, zum Beispiel die unterschiedlichen Bedeutungen von Farben im GUI.

3.7 KOSTENOPTIMIERUNGEN

Gerade während der Entwicklungszeit und der Produkteinführung möchte man die Aufwände möglichst tief halten, da die Azure-Kosten anfangs nur von wenigen zahlenden Nutzern getragen werden müssen. In Windows Azure gibt es verschiedene Möglichkeiten, die Kosten von Single-Instanz-Applikationen zu optimieren. Eine gute Übersicht zeigt Rainer Stropek in seinem Artikel im dotnet-Magazin 2011/11.

3.8 STRUKTURIERTE DATENSPEICHERUNG

Bislang wurde SQL Azure als strukturierter Datenspeicher verwendet. SQL Azure bietet den Vorteil, dass mit der bekannten Technologie des SQL Servers gearbeitet werden kann, die bestehenden Werkzeuge verwendet werden können und der Daten-Lock-in relativ gering ist, da die Daten einfach wieder nach SQL Server portiert werden können. Im Vergleich zu Azure Table Storage ist der Speicher jedoch 70-mal teurer. In SQL Azure kostet ein Gigabyte Speicher rund 10 USD, in Azure Table Storage 0.14 USD[9]. Die Preise liegen für bis zu 50 GB grosse Datenbanken linear zum benötigten

[9] Beim Azure Storage kommen noch Transaktionskosten von 0.10 USD pro 100 000 Speicherzugriffe hinzu.

Speicherplatz. Es kann daher aus finanzieller Sicht Sinn ergeben, den Gebrauch des Azure Table Storage in Erwägung zu ziehen.

Während bei SQL Azure der Speicherplatz nicht nur teuer, sondern auch mit maximal 150 GB pro Datenbank begrenzt ist, kann der Azure Table Storage Terrabytes von strukturierten Daten aufnehmen. Er verteilt diese automatisch auf physikalische Knoten anhand eines festgelegten Schlüssels. Der Table Storage verfügt jedoch nicht über die Möglichkeit, Daten via SQL abzufragen und auch bestehende Werkzeuge des SQL Servers können nicht verwendet werden. Somit ist der Daten-Lock-in beim Azure Table Storage sehr viel grösser: Wurde einmal die Entscheidung getroffen, den Azure Table Storage zu verwenden, ist es schwierig, die Daten zu einem anderen strukturierten Speicher (On-Premise oder in der Cloud) zu portieren. Ob sich ein Wechsel auf den Table Storage lohnt, muss anhand des konkreten Projektes abgeklärt werden.

3.9 INTEGRATION

Es ist selten der Fall, dass eine Anwendung alleine in ihrem Universum ist. Viel häufiger kommt es vor, dass Anwendungen mit anderen Anwendungen oder Diensten interagieren und auf fremde Speichersysteme zugreifen müssen. Windows Azure, SQL Azure und die Windows Azure AppFabric halten hier verschiedene Tools für die Integration von Schnittstellen zu anderen Applikationen bereit:

REST-APIs: Sämtliche Storage-Typen von Windows Azure verfügen über ein REST-API, das es ermöglicht, via HTTP bzw. HTTPS auf die Daten zuzugreifen. Es benötigt jedoch jeweils den geheimen Account-Key, um auf diese Ressourcen zugreifen zu können. Die Herausgabe dieses Keys an einen Client oder ein Fremdsystem ist unter Umständen nicht erwünscht. Hier gilt es im Einzelfall abzuwägen.

Shared-Access-Signatures: Der einzige Storage, welcher auch ohne Account-Key abgefragt werden kann, ist der BLOB-Storage. Hier gibt es einerseits öffentliche BLOBs, die von jedem gelesen werden können. Mittels Shared-Access-Signatures können jedoch einem Client auch differenzierte, zeitlich beschränkte Rechte auf einem BLOB oder einem Container vergeben werden, ohne den geheimen Schlüssel bekannt zu geben. Dies ermöglicht es beispielsweise einem Client, den BLOB direkt vom Storage herunterzuladen. Dies ist in Bezug auf Skalierbarkeit, Performance und Sicherheit von Vorteil.

ODATA: SQL Azure verfügt über eine standardisierte Schnittstelle für den Datenzugriff über das Internet. Das Open-Data-Protocol (OData)[10] ist ein offenes Web-Protokoll basierend auf HTTP, AtomPub und JSON für das Abfragen und Aktualisieren von Daten auf beliebigen Applikationen, Services und Datenspeichern. Neben SQL Azure gibt es bereits eine ganze Reihe von Produkten, die dieses Protokoll unterstützen. Die Zugriffskontrolle erfolgt via SQL-Azure-Benutzerkonten.

Windows Azure Connect: Windows Azure Connect ermöglicht es, virtuelle Maschinen in der Cloud und lokale Computer in einem VPN-Netzwerk zusammenzuschliessen. Damit ist es möglich, dass eine Applikation in der Cloud auf unternehmensinterne Ressourcen wie Active Directory, Dateisystem, Datenbanken, aber auch auf Hardware wie lokal installierte Kameras oder Drucker zugreifen kann. Windows Azure Connect ist im Moment noch im CTP-Stadium. Kosten fallen nur an, wenn über die Verbindung Daten aus der Cloud gelesen werden. Eingehende Datenverbindungen sind gratis.

[10] http://www.odata.org

Azure AppFabric Service Bus: Ein häufiges Problem bei der Integration von Applikationen über Domänengrenzen hinweg besteht darin, dass oft Firewalls im Einsatz sind, die eingehende Datenverbindungen nicht zulassen. Der AppFabric Service Bus behandelt dieses Problem und stellt eine Rendezvous-Adresse in der Cloud zur Verfügung. Applikation A und Applikation B können sich somit über diese Adresse miteinander verbinden, obschon eine oder beide Applikationen in einem Unternehmensnetzwerk laufen und damit von Firewalls geschützt sind. Der AppFabric Service Bus ist eine Erweiterung der Windows Communication Foundation und kann auch von Nicht-Dotnet-Clients genutzt werden.

Damit ist Stufe zwei der Migration erreicht. Die Applikation läuft kostenorientiert und sicher in der Cloud, nutzt weitergehende Services und Storages und kann mit Umsystemen kommunizieren. Es ist jedoch noch immer eine Single-Instanz-Applikation.

4 STUFE 3: SKALIERBARKEIT

Nun erwerben zwei Kunden unser Produkt: Kunde A und Kunde B. Für jeden der beiden Kunden setzen wir eine Installation mit jeweils einer einzelnen Instanz unserer Applikation in der Cloud auf. Dies führt zu folgenden Nachteilen:

- Das SLA von Windows Azure mit der Garantie einer 99.95%igen Verfügbarkeit gilt erst für Applikationen mit mindestens zwei Instanzen pro Role. Somit können wir ein solches SLA nicht an unsere Kunden weitergeben.

- Wir können die Applikationen zwar heraufskalieren, indem wir die Grösse der Instanz verändern, doch bei der Extra-Large-Instanz ist das Ende der Fahnenstange erreicht. Ist die Last, welche durch die Anzahl der Benutzer des Kunden entsteht, grösser, haben wir ein Problem.

Um diesen Nachteilen entgegenzuwirken, müssen wir unsere Applikation multiinstanzfähig machen. Um eine Multi-Instanz-Applikation zu realisieren, benötigt unser System weitere Anpassungen.

4.1 MEHRERE INSTANZEN

Der Azure-Plattform mitzuteilen, dass man die Applikation nicht als Single-Instance, sondern parallel auf mehreren virtuellen Maschinen gehostet haben möchte, bedingt einen entsprechenden Eintrag in der ServiceConfiguration.cscfg-Datei der entsprechenden Role. Dort kann die gewünschte Anzahl Instanzen angegeben werden. Häufig ist es sinnvoll, ein Vielfaches einer «Small»-Instanz zu verwenden, wodurch eine flexiblere Skalierung möglich ist. Die übrigen Instanzgrössen «Medium», «Large» und «Extra-Large» sollten nur bei besonderen Bedürfnissen, wie etwa grosse CPU-/RAM-Voraussetzungen einer Worker Role bei rechenintensiven Tasks verwendet werden. Grundsätzlich gilt jedoch auch hier: Scale-out (d. h. mehrere Instanzen verwenden) ist flexibler als scale-up (d. h. grössere Instanzen verwenden).

4.2 LOAD-BALANCING

Bei der Einführung von mehreren Instanzen tritt unmittelbar ein Problem auf, und zwar kann der Load-Balancer von Windows Azure nicht mit Sessions umgehen. Das bedeutet, dass alle Requests der Clients vom Load-Balancer der Reihe nach an die Instanzen der Web Roles weitergeleitet werden (Round-Robin-Verfahren). Ein Client, der beim ersten Request mit Web-Role-Instanz A kommunizierte, kann somit bei einem weiteren Request auf Instanz B treffen. Die Role-Instanzen müssen daher mit instanzübergreifenden Client-Sessions umgehen können. Dies kann erreicht werden, indem Session-Daten zentral zum Beispiel im Azure Table Storage, im AppFabric Cache, in SQL Azure oder clientseitig in einem Cookie, in einem Token oder in versteckten Formularfeldern gespeichert werden.

4.3 AUTHENTIFIZIERUNG UND AUTORISIERUNG

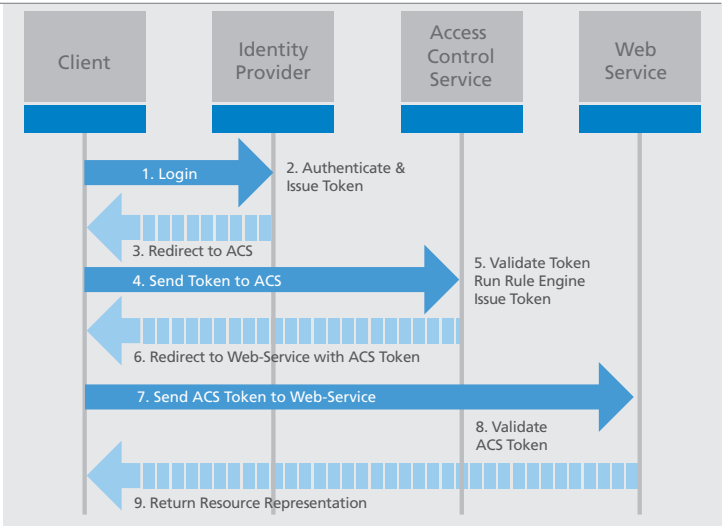
Das statuslose Verfahren des Load Balancers ist insbesondere ein Problem für die Authentifizierung und Autorisierung des Benutzers. Daher wurde in Windows Azure das claimsbasierte Authentifizierungs- und Autorisierungssystem AppFabric Access Control Service (ACS), basierend auf der Windows Identity Foundation, implementiert.

Bei einem claimsbasierten Verfahren wird die Authentifizierung des Benutzers aus der Applikation ausgelagert und einem Identity-Provider übertragen. Dies kann beispielsweise das Active Directory des Kunden sein. Dieses authentifiziert den Benutzer und signiert diese Authentifizierung in einem Token. Der Access-Control-Service überprüft anschliessend dieses Token und autorisiert den Benutzer, gemäss einer zuvor angelegten Rule Engine eine bestimmte Ressource, beispielsweise einen Webservice auf Azure, abzufragen. Diese Autorisierung wird nun ebenfalls in einem Token festgehalten und das neue Token vom Access-Control-Service signiert. Der eigentliche Webservice muss wiederum die Signatur des zweiten Tokens überprüfen. Stimmt die Signatur anhand des Schlüssels des Access-Control-Services überein, dann ist der Benutzer authentifiziert und gemäss den im Token festgehaltenen Rechten (Claims) autorisiert. Diese Art der Benutzerauthentifizierung ist vollständig statuslos und kann daher sehr gut in einem Load-Balancing-Umfeld eingesetzt werden. Zudem bietet sie automatisch auch Single-Sign-on. Abbildung 2 zeigt das Verfahren grafisch.

4.4 SKALIERBARKEIT

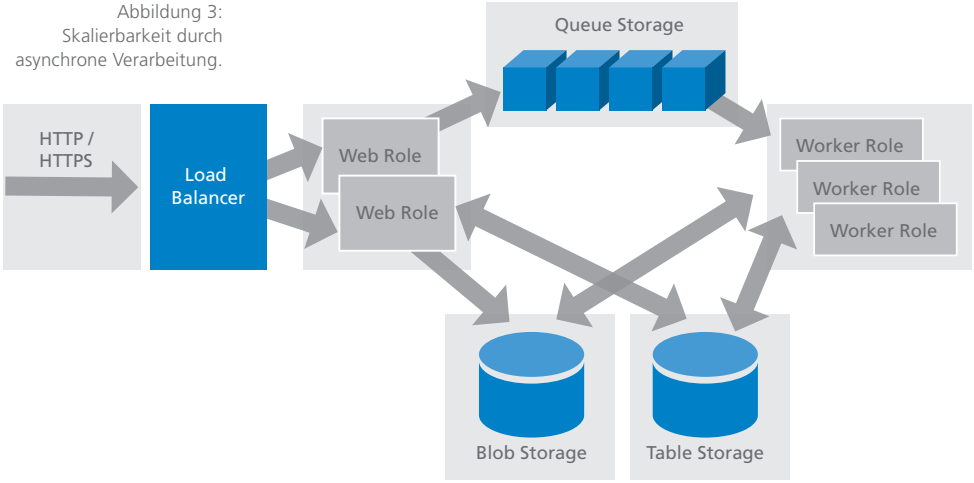
Obschon unsere Applikation mittlerweile bereits gut skaliert, gibt es ein Pattern, das in einem Umfeld wie Windows Azure häufig verwendet wird, um die Skalierbarkeit weiter zu erhöhen. Das Problem, welches dieses Pattern adressiert, besteht darin, dass bislang die Web Roles sowohl für die Entgegennahme als auch für die Verarbeitung der Requests verantwortlich sind. Dies kann dazu führen, dass einige Instanzen ausgelastet sind, während andere nichts zu tun haben, da die Requests unterschiedlich grosse Aktionen auslösen können. Dieses Problem kann gelöst werden, indem die Verarbeitung der Requests an Backend-Worker-Roles ausgelagert wird. Abbildung 3 zeigt die schematische Darstellung. Die Daten der Requests werden dabei von den Front-Ends in einen Storage geschrieben und eine Meldung mit der Beschreibung der Aktion in eine Queue gestellt. Diese Queue wird dann von unabhängigen Worker

Abbildung 2:
Access Control Service.



Roles gelesen und die Daten asynchron verarbeitet. Dies hat den Vorteil, dass die nun relativ schwach belasteten Front-End-Web-Roles unabhängig von den stärker belasteten Back-End-Worker-Roles skaliert werden können.

Abbildung 3:
Skalierbarkeit durch asynchrone Verarbeitung.



5 STUFE 4: SOFTWARE-AS-A-SERVICE

Obschon unsere Applikation jetzt hochskalierbar ist, können wir noch immer kaum von den Skaleneffekten der Cloud profitieren: Nehmen wir an, dass sowohl Kunde A als auch Kunde B ihre jeweiligen Instanzen nur zu 10% auslasten, dann könnten wir die Hälfte an Azure-Computing-Kosten sparen, wenn beide Kunden dieselben Instanzen verwenden würden. Vervielfachten wir die Rechnung auf 100 oder 1000 Kunden, sähen wir deutlich, dass unsere Kosten pro Kunde mit der Anzahl der Kunden abnähmen, da wir nur für jeden zehnten Kunden neue Instanzen unserer Applikation dazumieten müssten. Entsprechend würde unser Gewinn überproportional grösser, je mehr Kunden unsere mandantenfähige Applikation im Vergleich zur bisherigen Single-Tenancy-Architektur nutzen. Das Beispiel ist theoretischer Natur, und die Skaleneffekte können erst bei entsprechend hohen Nutzerzahlen realisiert werden. Es soll jedoch zeigen, was es ökonomisch bedeutet, Software «as-a-Service» anzubieten.

5.1 MULTI-INSTANZ VS. MULTI-TENANCY

Stufe 3 zeigt, wie eine Anwendung multiinstanzfähig gemacht werden kann. Das bedeutet, dass für einen Kunden mehrere Instanzen zur Verfügung gestellt werden können. Der Kunde verwendet jedoch diese Instanzen alleine.

Stufe 4 beschreibt eine Multi-Tenancy-Architektur, bei der mehrere Kunden dieselben Instanzen nutzen. Multi-Tenancy, oder auch Mandantenfähigkeit genannt, benötigt somit mehr als eine multiinstanzfähige Architektur: Es müssen zusätzlich die einzelnen Kunden voneinander getrennt werden können.

5.2 MANDANTENFÄHIGKEIT/MULTI-TENANCY

Wenn mehrere Mandanten (Firmenkunden) dieselben Instanzen einer Applikation verwenden, muss die Architektur sicherstellen, dass die Benutzer der einzelnen Mandanten nicht in die Daten der jeweils anderen Einblick erhalten. Diese Art der Architektur wird Mandantenfähigkeit (engl. multi-tenancy) genannt. Der Mandant bildet damit die oberste Abstraktionsschicht in einer solchen Architektur, und Benutzer und Daten gehören immer nur zu einem einzigen Mandanten. Ausgenommen sind mandantenübergreifende Stammdaten. Es kann durchaus Sinn ergeben, die Daten der einzelnen Mandanten getrennt in unterschiedlichen Datenbanken zu speichern. Die Mandanten teilen sich jedoch eine einzige hochskalierbare Anwendung.

5.3 PARTITIONIERUNG DES DATENBESTANDES

Verwendet man den Azure Table Storage als strukturierten Speicher, ist die Aufteilung der Daten nach Mandanten relativ schmerzlos. Es ergibt Sinn, die Daten jeweils eines Mandaten am selben Ort zu speichern, daher sollte der Mandantenschlüssel Teil des Parti-

tion-Keys sein, nach welchem die Verteilung stattfindet. Dies verhindert Abfragen über Partitionsgrenzen hinweg.

Bei SQL Azure ist das ein wenig komplexer, aber nicht unmöglich. Ein Beispiel zeigt das «Cloud Ninja – SQL Azure Federations Project»[11]. SQL Azure unterstützt das Database-Sharding (horizontales Aufsplitten der Daten von Tabellen auf mehrere Datenbanken, z. B. nach Mandanten) mittels SQL Azure Federations.

5.4 ANPASSBARKEIT/KONFIGURIERBARKEIT

Der Nachteil bei einem mandantenfähigen System gegenüber einer Single-Tenancy-Applikation ist jedoch, dass die Möglichkeit des Customizings verloren geht. Der Quellcode der Applikation kann nicht auf die Bedürfnisse des einzelnen Kunden angepasst werden. Um dies zu verhindern, muss man sich überlegen, wie man die Applikation konfigurierbar macht, damit der Anwender selber die Applikation seinen Bedürfnissen entsprechend anpassen kann. Dies beginnt bei der Gestaltung der GUI-Oberfläche und endet bei einer vollständigen Modularisierbarkeit der Funktionalität der Applikation, die der Kunde bei Bedarf dazumieten kann.

5.5 SELF-SERVICE ON DEMAND

Ein wesentlicher Bestandteil von Software-as-a-Service ist «Self-Service on Demand». Genau gleich wie bei der Plattform von Windows Azure sollten Ihre Kunden in der Lage sein, Ihre Applikation zu nutzen, ohne dass sie dafür einen Vertrag unterschreiben, eine E-Mail schreiben oder etwas installieren müssen. Auf der Herstellerseite sollte es zu keiner menschlichen Interaktion bei diesem Prozess kommen. Dies bedingt, dass Mandanten und alles, was dazugehört,

[11] http://shard.codeplex.com

automatisch angelegt werden und die Benutzung der Applikation automatisch verrechnet wird. Unter Umständen kann man sogar ein System entwerfen, das bei Bedarf automatisch skaliert, indem Role-Instanzen hinzugefügt oder entfernt werden, je nach aktueller oder erwarteter Belastung des Systems.

Die Entwicklung einer SaaS-Anwendung auf Stufe 4 ist nochmals ein weiterer grosser Schritt gegenüber Stufe 3 und die letzten fünf Unterkapitel beschreiben nur grob, was es alles braucht, um eine Multi-Tenancy-Applikation zu entwickeln. Eine solche Anwendung bedarf einer professionellen Analyse der geplanten Architektur.

6 ABSCHLIESSENDE BEMERKUNGEN

Das Booklet zeigt einen Weg für die Migration oder die Neuentwicklung einer Standard-Software auf Windows Azure. Die Vorschläge sind nicht allgemeingültig und müssen auf das konkrete Projekt angepasst werden. Im Sinne einer Checkliste wurden jedoch Probleme und Lösungen gezeigt, die Softwareherstellern und -entwicklern bei der Konzeptionierung der eigenen Software für die Cloud helfen sollen.

Die vier Stufen stellen letztlich die mögliche Palette zur Nutzung von Cloud Computing für die eine Applikation dar. Es kann aus verschiedenen Gründen jedoch auch Sinn ergeben, nur einen Teil der Stufen zu erklimmen. Wir von der bbv Software Services AG beraten Sie dabei gerne.

7 ANHANG

7.1 AUTOR

Roland Krummenacher ist Senior-Softwareingenieur bei der bbv Software Services AG und ist seit 2002 in der Softwareentwicklung tätig. Seit 2010 betreut er mehrere Projekte mit Windows Azure. Seine Schwerpunkte sind Cloud-Computing, .NET-Architekturen und Agile Softwareentwicklung.

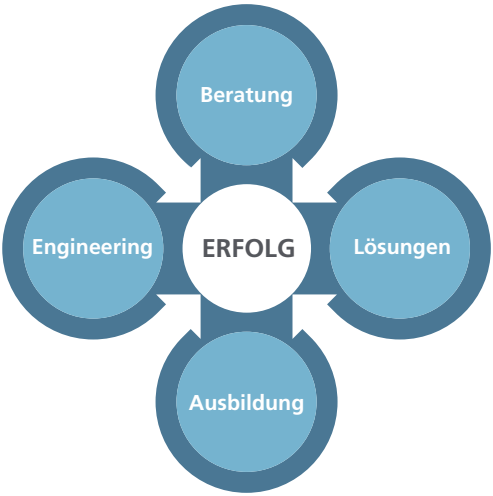
7.2 LITERATUR

Krishnan, S. (2010). Programming Windows Azure. Sebastopol, CA: O'Reilly Media, Inc.

Metzger, C., Reitz, T. & Villar, J. (2011). Cloud Computing. München: Carl Hanser Verlag.

bbv
MAKING VISIONS WORK.

bbv Software Services AG ist ein Schweizer Software- und Beratungsunternehmen, das Kunden bei der Realisierung ihrer Visionen und Projekte unterstützt. Wir entwickeln individuelle Softwarelösungen und begleiten Kunden mit fundierter Beratung, erstklassigem Software Engineering und langjähriger Branchenerfahrung auf dem Weg zur erfolgreichen Lösung.



Unsere Booklets und vieles mehr finden Sie unter
www.bbv.ch/publikationen

MAKING VISIONS WORK.

www.bbv.ch · info@bbv.ch