IoT + Cloud = Money

Effektiver Business-nutzen mit der CloudLine Reference Architecture

Richard Hubert

Die Cloud ist längst Realität – vor allem deshalb, weil sie zahlreiche Innovationen für viele neue Geschäftsmodelle sowie Mehrwertoptionen für vorhandene Systeme bietet. Grundvoraussetzung für nachhaltig erfolgreiche Projekte ist eine auf die neuen Anforderungen, Prozesse und Technologien angepasste Architektur. Die Grundüberlegungen, die jede IT-Architektur treiben, gelten dabei in gleichem Maße für Cloud-Architekturen. Dieser Artikel stellt eine praxisbewährte Referenzarchitektur für Cloud-Projekte vor.

Cloud-Referenzarchitektur statt Chaos

In der Cloud trifft man zunächst auf zahlreiche Bekannte der Service-Oriented Architecture, manche davon in neuen Gewändern: Aus RPC wurde CORBA, dann SOAP, dann REST. XMI wurde zu JSON und SSL wurde zu TLS. Mit den neuen Möglichkeiten und Anforderungen von Cloud-Anwendungen haben sich andere Bereiche so gewandelt, dass ihre Cloud-Vorgänger kaum noch wiederzuerkennen sind. Dazu gehören zum Beispiel Autorisierung und Authentisierung (Identity Management), Sicherheit und Datenschutz. Daneben sind aufgrund neuer Möglichkeiten und Anwendungen auch völlig neue Technologieklassen wie No-SQL-Datenbanken, In-Stream Analytics oder Hybrid Storage APIs getreten.

Der Fokus dieses Artikels liegt primär auf neu entstandenen Designaspekten. Beleuchtet werden deren Potenziale und Herausforderungen und die Art und Weise, wie sie in einem umfassenden Architekturkontext, der „CloudLine Reference Architecture“, zum Einsatz kommen. Das Ziel dieser Referenzarchitektur besteht darin, den Weg zum effektiven Business-Nutzen der Cloud so kurz und hindernisfrei wie möglich zu halten. Das Fundament wird dabei so gestaltet, dass auch längerfristig und nachhaltig von der ständigen Evolution der Cloud-Services und Plattformen profitiert werden kann.

Die hier präsentierte Referenzarchitektur hat sich in unterschiedlichen Projekten in der Praxis bewährt. Es wird primär der Microsoft Azure Stack als Basis genutzt, weil er unter anderem aufgrund der hohen vertikalen Integration in der großen Mehrzahl der Cloud-Produkt- und Cloud-SOA-Szenarien am schnellsten zum Return on Investment (ROI) führt (laut Gartner-Studien, s. [Hers15]). Programmiersprachen spielen hier eine untergeordnete Rolle, denn es geht in der Cloud primär um Microservices, die jeweils in einer der diversen web-fähigen Sprachen geschrieben werden können, sowie um REST-APIs, die sprachneutral genutzt werden können. Microsoft hat sich verändert: .NET ist Open Source und Java, Javascript, PHP usw. sind genauso in den Azure Services vertreten wie C#. Betriebssysteme und die darunter liegende Infrastruktur tauchen übrigens in der Welt der Cloud-Microservices gar nicht mehr auf und müssen deshalb auch nicht mehr von einer Cloud-Referenzarchitektur berücksichtigt werden. Signifikante Vorteile entstehen schon alleine dadurch, dass Microservices jeweils in der am besten geeigneten oder vom Entwickler bevorzugten Sprache geschrieben werden können, während das Gesamtdesign hauptsächlich auf die Partitionierung und Service-Levels der APIs und Microservices fokussiert.

Um die Frage nach Zweck und ROI eines Architectural Framework für IoT und Cloud-Services gleich vorweg zu beantworten: In diesem Zusammenhang stellt die Cloud definitiv keine Revolution dar. Es gilt grundsätzlich auch hier, was in anderen Bereichen der Softwareentwicklung längst etablierter Standard ist, auch wenn eine Cloud- Architektur sich naturgemäß von anderen Architekturen unterscheidet. Ein professionelles Architectural Framework zu entwickeln und zu pflegen, ist auf jeden Fall eine Business-Entscheidung für nachhaltige Wertschöpfung und schnellen ROI.

Die CloudLine Reference Architecture orientiert sich an einem Big Picture. Neben dem Design des zentralen Cloud-Systems selbst, das in Abbildung 1 blau hinterlegt ist, werden zwei weitere Bereiche einbezogen, die für einen langfristig erfolgreichen Lebenszyklus der entwickelten Komponenten und der verwendeten Cloud-Services sorgen: eine DevOps-Plattform und die Operational-Systems-Plattform.

Die Referenzarchitektur betrachtet den Lebenszyklus und die Design-Evolution des Gesamtsystems. Sie definiert und transportiert grundsätzlich wichtige Top-Down-Aspekte wie eine integrierte und wiederverwendbare Systemstruktur als Baustil, eine effektive Werkzeugausstattung, qualitätssichernde Referenzimplementierungen sowie Entwicklungs-Governance bis in die Produktion und Pflege hinein. Das alles sind konkrete und ROI-relevante Themen, die ohnehin im Rahmen von Projekten adressiert werden müssen. Hierbei gibt es letztlich nur das klassische „Entweder-oder-Paradigma“: entweder ad hoc, rein projektbezogen und dadurch tendenziell zu projektübergreifendem Chaos führend oder im Rahmen einer stetig besser werdenden, projektübergreifenden Referenzarchitektur.

Wenn das Service-Design jeder der in der Abbildung gezeigten Plattformen definiert ist, dann sind Referenzimplementierungen für Anwendungen/Apps und Microservices möglich.

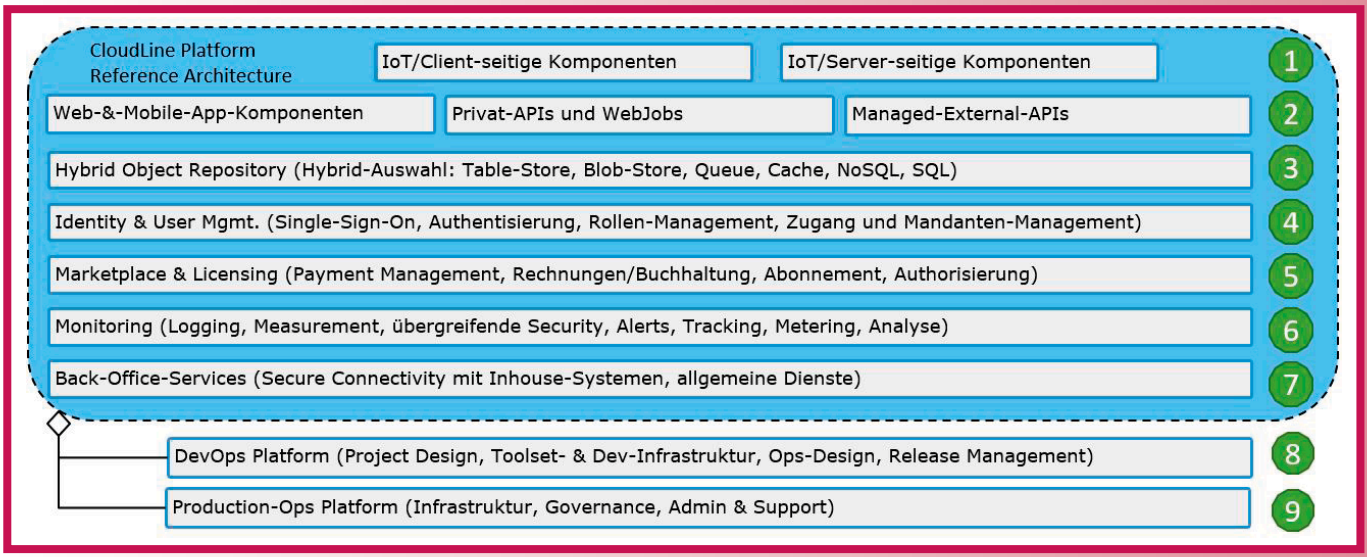

Abb. 1: CloudLine Reference Architecture

Diese Referenzimplementierungen ähneln qualitätsgesicherten Fertigbausteinen. Sie sind in vielerlei Hinsicht äußerst nützlich, erfordern aber die Überwindung einer singulären Fokussierung auf Einzelprojekte, die eine weitsichtige Vorbereitung und Optimierung nicht in ausreichendem Maße ermöglichen. Die Referenzimplementierungen reduzieren die Kosten und erhöhen sowohl die Qualität als auch die Geschwindigkeit jedes einzelnen Projekts, weil sie Erlerntes aus Einzelprojekten in qualitätsgesicherten Fertigbausteinen konsolidieren. Das erwartet man ja auch von zugekauften Komponenten – und die eigenen Referenzimplementierungen bringen unternehmensspezifischen Mehrwert hinzu. Damit tragen sie zu nachhaltigen Innovationszyklen im Unternehmen bei [Bosch13]. Die Projektteams verbringen dadurch signifikant weniger Zeit mit nicht-wertschöpfenden Tätigkeiten.

Ein langfristig angelegtes Zieloptimum – eine „long-term architectural vision" – als Leitbild darf bei keiner Referenzarchitektur fehlen, denn ohne Ziel wäre auch der Weg nicht bewertbar. Die CloudLine Reference Architecture verwendet das bewährte Convergent Engineering von David A. Taylor [Tayl95,Hub02] als Architectural Style sowie zahlreiche Aspekte aus „Dependency-Oriented Thinking" von Ganesh Prasad [Pras14] und das hervorragende und schnell erlernbare „Three Layered Product Model" von Jan Bosch [Bosch13].

Abbildung 1 zeigt die neun Hauptbestandteile der Referenzarchitektur, die in den folgenden Abschnitten mit der jeweiligen Bezugsnummer erläutert werden. Die Quadrate in der Abbildung stellen Plattformen dar, die von allen anderen Plattformen im Sinne eines Service-Oriented Abhängigkeitsmanagements [Bosch13,Pras14] verwendet werden. Die Plattformen besitzen eine Komponentenstruktur (sind deshalb auch selbst Komponenten), die nicht nur Systemelemente definiert, sondern auch die entsprechenden Entwicklungs- und Product-Lifecycle-Elemente.

IoT/Client-seitige und IoT/Server-seitige Komponenten (1)

Die IoT/Client-seitige Komponente stellt die geräteseitige Kommunikationslogik dar (Handshake, Security, Verbindungsmanagement), die auf einer Vielfalt von im Internet-of-Things (IoT) verteilten Geräten eingesetzt werden kann. Heute handelt es sich meistens um gerätespezifische Kernanwendungen, die eine sichere und robuste Kommunikation mit zentralen Cloud-Service-APIs verwenden. Managed Cloud-Services wie Azure Service Bus oder Azure IoT-Hub bieten hier eine für IoT und Cloud optimierte Kommunikationslösung „vom Regal".

Die IoT/Server-seitige Komponente bietet eine Standardplattform für Empfang und Weiterverteilung der IoT-Datenströme. Wichtige Aspekte wie Kommunikations- und Fehler-Management, Security-Logik und Logging sind hier angesiedelt. Auch hier gibt es Managed Cloud-Services wie Azure Stream Analytics, Machine Learning oder Automation, die „vom Regal" eingesetzt werden können.

Web-&-Mobile-App, API und WebJobs-Komponenten (2)

Aufbauend auf die Informationsströme in (1) findet sich hier die spezifische Geschäftslogik in Form von Webanwendungen, Apps und Managed REST-APIs. Auch hier nehmen Cloud-Service-Bausteine wie die Azure App Services, Mobile Services, WebJobs und Api Management Services einen großen Anteil der traditionellen Implementierung und Infrastruktur ab.

Besonders interessant ist der Api Management Service, weil er gleichsam sowohl ein Entwickler-Partnernetz als auch ein „Entwickler-Ökosystem" (also auch einen Datendienst-Marktplatz) ermöglicht. Mit diesem Service lassen sich neue Cloud-Geschäftsmodelle leichter entwickeln und auch ein starker Inbound-Marketing-Effekt bei Fremdanbietern generieren.

Hybrid Object Repository (3)

Die konsolidierten Informationen von Ebene (1) und (2) dienen als wichtigstes wertschöpfendes Rohmaterial für weitere Anwendungen. Mit der neuen Vielfalt der Datenhaltung in der Cloud trägt das Service-Design eines Hybrid Object Repositorys wesentlich zu Produktwert und Marktdifferenzierung bei. Fähigkeiten wie Datenstromverarbeitung oder -analyse in

Echtzeit, kosteneffektives Big-Data-Management oder eine flexible Suche in Quasi-Echtzeit (Elastic Search) sind heute auch über den Einsatz von fertigen Managed Cloud-Services leicht erreichbar. Diese Fähigkeiten können nicht mehr kosteneffektiv über eine Zentraldatenbank implementiert werden.

Die neue Cloud-Datenlandschaft ist vielfältig: Blob-, Queue-, Table-Storage, JSON-NoSql DB, Redis-Cache, Hadoop, SQL Data Warehouse, um nur einige der verfügbare Managed Cloud-Services zu erwähnen. Wo und wie die Daten gespeichert werden, wird dadurch zu einem exponentiellen Faktor bei Kosten und Leistungsfähigkeit eines Cloud-Systems. Praktisch jede Anwendung hat hier einen anderen Anforderungs-Mix. Die Komponente Hybrid Object Repository übernimmt deshalb nicht nur die üblichen Kapselungsdienste einer guten Plattformkomponente, sondern bietet dem Entwickler auch eine einfache Möglichkeit, die gesamte Palette der unterschiedlichen Datendienste zu nutzen – bei gleichzeitig hoher Design-Integrität. Dadurch sind Kundenlösungen mit anspruchsvollen Anforderungen schneller lieferbar und qualitativ solider.

Auch die operative Seite profitiert: Dienste wie Geo-Redundant-Storage, bei dem sechs Kopien der Daten über zwei Geo-Regionen gehalten werden, liefern nicht nur Leistungssteigerungen, sondern auch einen sehr hohen Service-Level und Kontinuitätsstandard. Notfallpläne oder Rollbacks zur Fehlerkorrektur sind auch durch Services wie Azure Backup/Restore und Point-in-Time Snapshots sehr einfach geworden.

Identity, User & Tenant Management (4)

Systeme, die eine Vielfalt von sicheren Mehrwertdiensten in der Cloud anbieten wollen, müssen die Systemnutzer (Endkunden, Admins, aber auch ihre Geräte) über das gesamte verteilte System hochsicher authentisieren und identifizieren, ohne den Endanwender ständig zu „nerven“. Compliance, das heißt, die nachweisbare Einhaltung eines oder auch mehrerer gegebener rechtlicher Rahmen sowie eine lückenlose Sicherheit bei diversen Informations- und Prozessarten, spielt hier ebenfalls eine zentrale Rolle.

In diesem äußerst komplexen Bereich des „Access and Identity Management“ (AIM) sollte man möglichst nicht als „Erfinder“ unterwegs sein – es sei denn, gerade dies gehört zum Kerngeschäft. Die Auswahl des geeigneten AIM-Anbieters ist eine der zentralen Vorbereitungsmaßnahmen einer angewandten Referenzarchitektur. Jeder Anbieter in diesem komplexen Umfeld erfüllt sehr unterschiedliche Anforderungen mit verschiedensten Kompromissen. Im Bereich der gehobenen Freeware sind Shibboleth und Thinktecture zu nennen. Bei den starken Frameworkanbietern ForgeRock und WSO2 muss ein ausfallsicherer Serverbetrieb dazugerechnet werden, egal ob in der Cloud oder Inhouse. Es gibt aber auch hier hervorragende Managed Cloud-Services wie Windows Azure Active Directory, Amazon AIM und Auth0.

Marketplace & Licensing (5)

Aufbauend auf (4) kann dann ein Lizenzierungs- und Rechte-Management als „Marktplatz“-Service etabliert werden. „Marktplatz” deshalb, weil – unabhängig davon, ob Geld verlangt wird oder nicht – nach einer Authentisierung in (4) auch eine Autorisierung und Zuordnung der Nutzungsrechte stattfinden muss. Das gilt nicht nur für Menschen, sondern auch für Geräte, die mit dem System interagieren. Ein Beispiel hierfür sind Geräte in modernen Automobilen, die sich unterwegs ständig identifizieren und über einen „Marktplatz“ die Freigabe von Mehrwert-Features für den aktuellen Fahrer abfragen. Bekannte Marktplätze dieser Art in der Cloud sind die Shops von Apple, Windows, Amazon AWS oder der Azure Marketplace. Die Granularität der Rollen und Systemeigenschaften hängt vom Zweck und Umfang des Service ab, der erfahrungsgemäß mit der Zeit wächst. Dies gilt insbesondere in Cloud-Systemen, wo den Endkunden im Prinzip fast täglich neue Features sehr einfach angeboten werden können. Hier bestehen also eigentlich keine „natürlichen Grenzen“ mehr, und deshalb empfiehlt sich hier wie bei (4) eine professionelle und fokussierte Lösung als Basis.

Monitoring (6)

Neben Stabilität und Leistung ist jede Komponente im System für ihren eigenen Schutz zuständig. Sicherheit ist Designpriorität Nummer eins. Die Monitoring-Komponente unterstützt jede Einzelkomponente in der Aufgabe der Intra-Komponente Monitoring und Logging das Gesamtsystem in der Aufgabe der Inter-Komponente oder „end-to-end“ Monitoring und Analyse. Auch alle Formen der Messung (z. B. Metering) finden sich hier.

Zum Monitoring gehören neben reaktiven Warnmeldungen (Alerts) auch die Definition und das Management des proaktiven Handelns. Zum Beispiel lässt sich bei fast allen Azure Services ein automatisches „Elastic Scaling“ einstellen. Sind die Regeln definiert (z. B. CPU-Last, Antwortzeit, Max.-/Min-./Schwellwerte), werden Cloud-Dienste, Datenbanken, Verbindungsrouting usw. automatisch nach den Regelvorgaben hoch- und herunterskaliert, verteilt und protokolliert. Diese faszinierenden Cloud-Features bringen ein ganz neues Optimierungspotenzial für kosteneffektives Betriebsmanagement und vor allem eine einfache Gewährleistung der Service-Qualität ins Spiel.

Ähnliches gilt für Komponenten übergreifende Überwachung. Die Monitoring-Komponente definiert hier eine wiederverwendbare Instrumentierung, die alle Komponenten und Referenzimplementierungen (im Code und in der Konfiguration) für eine granulare Qualitätssicherung, Überwachung und Steuerung der Systemintegrität verwenden können. Die Datenströme von Pre- und Post-Conditions im Code fließen zentral zusammen, zum Beispiel in der Azure Table Storage oder bei On-Demand-Services wie Logentries oder Splunk. Sie können dabei von Azure Stream Analysis- und Azure Machine-Learning-Algorithmen auf Alarme, Trends oder Anomalien überwacht werden. Ebenso fließen Messströme (Metering Data) aus den Anwendungen über Azure Stream Analysis, um eine Konsolidierung, Auswertung und Verdichtung auf dem Weg zur Speicherung zu ermöglichen.

Back-Office-Services & Konnektivität (7)

In den meisten Cloud-Szenarien gibt es weiterhin Enterprise-Systeme, die aus diversen Gründen hinter einem Peripherieschutz – in der Cloud oder On-Premise – laufen oder besondere Anforderungen stellen, zum Beispiel an die Art der Kommunikation. Ein Beispiel hierfür sind SAP-Systeme oder Workflow-Anwendungen. Die Back-Office-Services-Komponente schützt und vereinfacht die Verbindung mit diesen Systemen. Auch andere wiederverwendbare Basisdienste wie Druck oder Mail, die durchaus auch in der Cloud laufen können, werden von dieser Komponente als wiederverwendbare Services angeboten.

DevOps-Plattform und Production-Ops-Plattform (8/9)

Wie in Abbildung 1 zu sehen, ist die DevOps-Plattform kein inhärenter Teil der Cloud-Referenzarchitektur, sondern stellt zusammen mit der Production-Ops-Plattform (9) ein wichtiges Fundament dar. Wenn dieses Fundament nicht ausreichend dimensioniert ist, ist alles, was auf ihm aufbaut, ebenfalls instabil – und zwar unabhängig davon, wie gut das individuelle Design auch sein mag. Deshalb sind die Themen (8) und (9) dennoch aus der Sicht der Referenzarchitektur zu adressieren, um eine erfolgreiche Umsetzung und Evolution zu sichern.

Diese Plattformen umfassend zu beschrieben, würde den Rahmen dieses Artikels sprengen. Zusammenfassend sei gesagt, dass eine DevOps-Vorgehensweise zahlreiche Unternehmen heute vor ähnlich viele Herausforderungen stellt, wie sie den Unternehmen Vorteile bringen würde. Zunächst sind rein projektgerichtete und projektgetriebene Strukturen in Unternehmen oft nur in geringem Maße auf eine kontinuierliche Evolution hin ausgelegt, vor allem im Bereich Software. Zweitens stehen das Design und die Umsetzung von umfassenderen, projektübergreifenden Themen wie DevOps, Operations und Enterprise Architecture gar nicht in angemessenem Maße auf der Agenda, weil sie nicht direkt einem Einzelprodukt zugeordnet werden können. Je nach Priorisierung bleiben dadurch Themen, die oft den signifikantesten Beitrag zur Wertschöpfung leisten könnten, unberücksichtigt.

Solche Themen können nicht durch gut gemeinte Ad-hoc-Aktionen erfolgreich adressiert werden, im Gegenteil. Ein typisches Beispiel hierfür ist das wichtige Thema der Systemmodellierung. Eine Modellierungsinitiative zu starten, sei es UML oder TLA+ [TLA+], ohne vorab über die Metaebene eines Modellierungsstils nachgedacht zu haben, führt praktisch zwangsläufig zur Einstellung solcher Modellierungsversuche, weil sie sich ohne die erforderlichen Grundlagen im Sinne einer „Self Fullfilling Prophecy“ schnell als „unproduktiv“ erweisen. Eine ebenso aufschlussreiche wie brillant geschriebene Studie hierzu ist Bertrand Meyers „Agile! The Good, the Hype and the Ugly“ [Mey14].

Schlussanmerkungen

Mit der CloudLine Referenzarchitektur kann ein geeigneter Mix aus neuen On-Demand-Services, zugekauften Komponenten und Eigenentwicklungen so definiert und orchestriert werden, dass hochwertige Cloud-Anwendungen schnell, aber auch nachhaltig „auf die Straße“ gebracht werden können. Die Cloud wird aufgrund solider Innovationen eine erhebliche Rolle bei Unternehmen und Institutionen spielen. Die Einstiegshürden scheinen hoch zu sein, aber mit dem Schritt zu einer ganzheitlichen Cloud-Referenzarchitektur kommen Organisationen jeder Größenordnung sehr schnell auf die Zielgerade.

Literatur und Links

[Bosch13] J. Bosch, Achieving Simplicity with the Three-Layer Product Model, in: Computer, vol.46, no. 11, pp. 34-39, Nov. 2013

[Hers15] N. Herskowitz, Microsoft – the only vendor named a leader in Gartner Magic Quadrants for IaaS, Application Paas, and Cloud Storage, Microsoft, 22.5.2015, `https://azure.microsoft.com/blog/2015/05/22/microsoft-the-only-vendor-named-a-leader-in-gartner-magic-quadrants-for-iaas-application-paas-cloud-storage-and-hybrid/`

[Hub02] R. Hubert, Convergent Architecture: Building Model-Driven J2EE Systems with UML, Wiley, 2002

[Mey14] B. Meyer, The Good, the Hype and the Ugly, Springer, 2014

[Pras14] G. Prasad, Dependency-Oriented Thinking: Volume 2 – Governance and Management, InfoQ Enterprise Software Document Series, August 2014 (`http://www.infoq.com/minibooks/dependency-oriented-thinking-2`)

[Tayl95] D. Taylor, Business Engineering with Object Technology, Wiley, 1995

[TLA+] Temporal Logic of Actions, `http://research.microsoft.com/en-us/um/people/lamport/tla/tla.html`

Richard Hubert ist IT-Architekt mit zahlreichen Auszeichnungen, IEEE CSDP und IASA Fellow (An Association for All It Architects, http://iasaglobal.org/iasa-fellowship) sowie Autor.
E-Mail: hubert@acm.org