Einfaches Einbetten in das Internet der Dinge:

IoT von der Stange

(Bild: everything possible – Shutterstock)

Heute sollte am besten jedes Gerät in die große Rechnerwolke eingebettet werden. Doch so einfach ist das nicht, denn Cloud ist viel mehr als nur das Internet der Dinge. Als Anwender muss man sich also fragen, welche Dienste man möchte und welchem Anbieter man sein Vertrauen schenkt.

Von Dr. Jörg Wollert und Andreas Booke

Bereits seit einigen Jahren werden in der Informationstechnik am häufigsten die Begriffe Cloud und Big Data benutzt. Doch was verbirgt sich dahinter? Sind die Begriffe sogar synonym zu verwenden? Das sind Fragen, die eigentlich gestellt werden müssen, um den Hype rund um das Cloud Computing (**Bild 1**) zu verstehen.

Betrachtet man die historische Entwicklung der Datentechnik, so wird schnell deutlich, warum Cloud so spannend ist. Ursprünglich waren in den 1960er und Anfang der 1970er Jahre Zentralrechner mit Terminalschnittstellen Stand der Technik. Computer waren teuer, doch nur die Verwendung von mehreren Terminals ermöglichte eine Verteilung der Rechenleistung auf mehrere User. Für die industrielle Datentechnik wurde die DEC PDP11 in den 70er Jahren das Synonym für einen Prozessrechner. Eine Revolution in der Rechnertechnik erfolgte in den späten 70er Jahren. 1976 gründete Steve Jobs mit Steve Wozniak und Ronald Wayne Apple und 1981 führte IBM den PC (Personal Computer) ein. Erstmals erhielten Arbeitsplatzrechner eine eigene Intelligenz und ermöglichten ein lokales Arbeiten jenseits des Großrechners. Lokales Arbeiten, Spielen und Programmieren waren fortan Stand der Technik. Erst in den 90er Jahren mit der Entwicklung des Internets war auf einmal wieder vernetztes Computing angesagt. Sogenannte Client- und Server-Architekturen stellten dedizierte Dienste zur Verfügung, die temporär, per Einwahl genutzt werden konnten. Teure Einwahlverbindungen begrenzten die ferne Nutzung von Diensten auf ein Minimum. Erst in den späten 1990er Jahren wurde mit der Dotcom-Welle und der Internet-Flatrate eine ganz neue, digitale Welt geschaffen. „Immer online“ wurde zum Standard.

Der Trend führte zu einer deutlichen Zunahme von Anbietern für Internettechnik. Server mussten gehostet und Speicherplatz musste bereitgestellt werden. Um dem zunehmenden Bedarf an Speicher- und Rechnerkapazität gerecht zu werden, wurde die Hardware zunehmend virtualisiert und auf skalierbare Rechenzentren ausgelagert. Durch

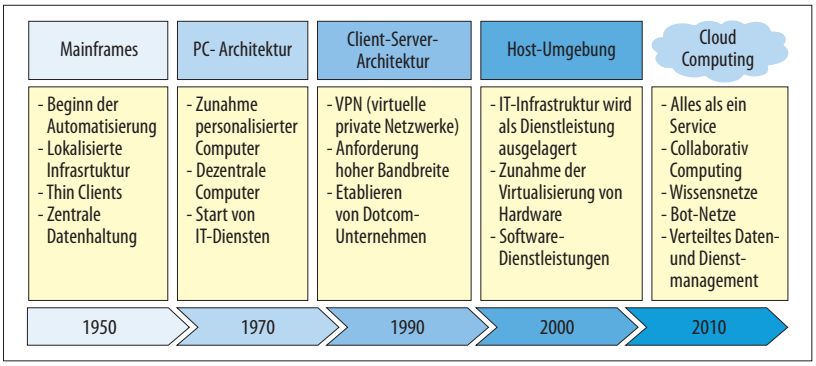

Bild 1. Cloud Computing ist der nächste Schritt in der IT-Entwicklung. (Quelle: Wollert, Booke)

den Wechsel von Hardware-Zentrierung hin zu Software-basierten Systemen war der Schritt zu Mehrwertdiensten auf den Server-Systemen offensichtlich. Warum den ganzen Server mieten, wenn eine Anwendung ausreicht? Der dienstleistungsorientierte Ansatz wurde in der letzten Dekade zum Standard. Skalierbare Dienste, die auf einem oder vielen Rechnern arbeiten können, nutzen Speicher, der hinter einem virtuellen Server verborgen ist und keine physikalische Beziehung zu einer Hardware hat. Dienste und Infrastruktur sind irgendwo in der Cloud.

Unter einer Cloud (**Bild 2**) versteht man eine Ansammlung von Infrastrukturkomponenten und unterschiedlichen Diensten, die nicht mehr lokal innerhalb eines Netzwerkes sind. Stattdessen werden sie global aus der Ferne betrieben. Ganze IT-Infrastrukturen können über die Cloud genutzt und organisiert werden, ohne dass der Rechner physisch in der Nähe verfügbar sein muss. Die Grundideen stammen von den schnell wachsenden großen Internet-Dienstleistern wie Amazon, Google oder Yahoo. Die Firmen mussten teilweise sehr starke Spitzenlastzeiten mit der vorhandenen IT-Infrastruktur abfangen, obwohl die Grundlast deutlich geringer war. Ein anschauliches Beispiel liefert das alljährliche Weihnachtsgeschäft bei Amazon, bei dem die IT-Infrastruktur deutlich stärker beansprucht wird als zu allen anderen Zeiten. Die Anforderungen führen zu einer skalierbaren Infrastruktur, in der Ressourcen bereitgestellt werden, wenn sie tatsächlich benötigt werden.

Eine Cloud besteht aus Front Ends, also dem Benutzer-Client – zumeist ein Rechner mit einem Webbrowser –, und dem Back End. Das Back End einer Cloud stellt ein komplettes Ökosystem bereit, das skaliert durch den Betreiber oder Nutzer zur Anwendung kommt. 2009 veröffentlichte das National Institute of Standards and Technology (NIST) eine Definition für Cloud Computing, die auf hohe Übereinstimmung innerhalb der IT-Branche stieß [1]. Darin werden verschiedene Servicemodelle unterschieden:

Infrastructure as a Service (IaaS)

IaaS ist die grundlegende Basis in einem Back End. Die „Rechnerwolke" bietet Nutzungszugang von virtualisierten Computer-Hardware-Ressourcen wie Rechnern, Netzen und Speicher. Mit IaaS gestalten sich Nutzer frei ihre eigenen virtuellen Computer-Cluster und sind daher für die Auswahl, die Installation, den Betrieb und das Funktionieren ihrer Software selbst verantwortlich.

Platform as a Service (PaaS)

Werden spezifische Software-Pakete bereits als Infrastruktur mitgegeben, spricht man von PaaS. Hier kommt zu der eigentlichen skalierbaren, virtualisierten Hardware noch eine Dienstleistungsschicht zum Nutzungszugang von Programmierungs- oder Laufzeitumgebungen hinzu. Nutzer können so ihre eigenen Software-Anwendungen entwickeln und ausführen oder es können Software-Umgebungen vom Dienstanbieter (Service Provider) bereitgestellt und unterhalten werden.

Software as a Service (SaaS)

Bei SaaS werden Software-Pakete durch den Provider einsatzbereit zur Verfügung gestellt. Nicht nur selbst entwickelte Software kann in der Cloud gehostet werden. Mittlerweile werden viele Anwendungsprogramme in der Cloud angeboten – wie Office, Datenbanken und Webshops. SaaS wird auch als Software on Demand (Software bei Bedarf) bezeichnet.

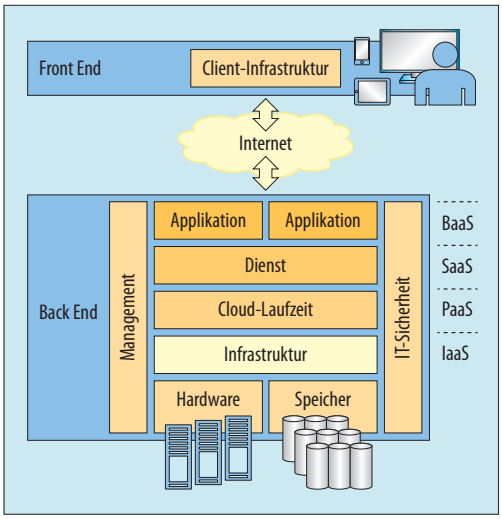

Bild 2. Eine Cloud beschreibt ein gangzes Ökosystem rund um Serverdienste. (Quelle: Wollert, Booke)

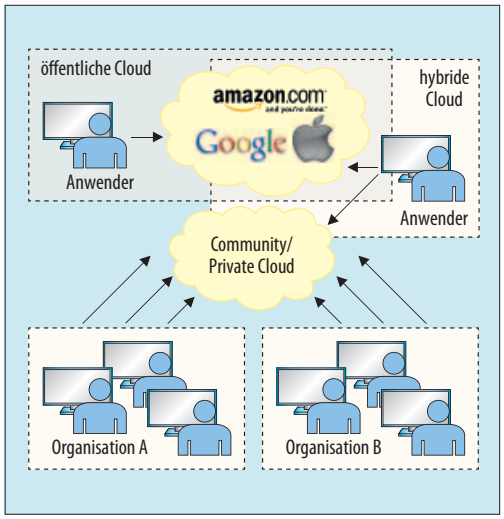

Bild 3. Unterschiedliche Liefermodelle bestimmen die Öffentlichkeit der Dienste. (Quelle: Wollert, Booke)

Business as a Service (BaaS)

Das nächste Level stellt BaaS, eine Anhäufung unterschiedlicher Software-Pakete, dar. Werden verschiedene Software-Dienste miteinander verknüpft, können auch komplexe Geschäftsprozesse abgewickelt werden. Das kann zum Beispiel die Verknüpfung eines Webshop mit Banktransaktionen und den Diensten eines Logistikdienstleisters sein. Hier sind beliebige Kombinationen denkbar, die sehr stark von der individuellen Bedarfslage abhängen.

Everything as a Service (XaaS)

Heute geht man so weit, dass eigentlich alles als Dienst angeboten werden kann. Das XaaS ist sicherlich nicht ganz ernst zu nehmen, zeigt aber, wohin die Service-Welt geht – sie ist quasi grenzenlos und nur noch von der Kreativität des Entwicklers und Betreibers abhängig. Nichts ist unmöglich.

Neben den vorhergehenden Service-Modellen werden auch sogenannte Liefermodelle unterschieden. Unter Liefermodellen (**Bild 3**) versteht man die Form, wie die Nutzer einen Zugang zu den Diensten erhalten.

Public Cloud – Zugriff für alle

Die öffentliche Cloud bietet Zugang zu abstrahierten IT-Infrastrukturen für die breite Öffentlichkeit über das Internet. Public-Cloud-Dienstanbieter erlauben ihren Kunden, IT-Infrastruktur auf einer flexiblen Basis zu mieten. Sie müssen nur den tatsächlichen Nutzungsgrad bzw. Verbrauch bezahlen (pay as you go), ohne Kapital in Rechner- und Datenzentrumsinfrastruktur investieren zu müssen. Anbieter sind die großen Vier: Amazon, Microsoft, IBM und Google.

Private Cloud – die ganz private Rechnerwolke

In einer privaten Cloud wird ein Zugang zu abstrahierten IT-Infrastrukturen innerhalb der eigenen Organisation – z.B. Behörde oder Firma – gewährt. Eine Private Cloud bietet größte Sicherheit, denn nur Mitglieder der spezifischen Cloud haben einen Zugang zu den Diensten.

Hybrid Cloud – skalierbare Dienste mit privatem und öffentlichem Zugang

Häufig ist der Zugriff auf öffentliche und private Dienste erforderlich. Dafür bieten hybride Clouds einen kombinierten Zugang zu abstrahierten IT-Infrastrukturen aus den Bereichen der öffentlichen und privaten Clouds, ganz nach den Bedürfnissen ihrer Nutzer.

Community Cloud – die gemeinschaftliche Rechnerwolke

Immer attraktiver werden Gemeinschafts-Clouds. Sie bieten einen Zugang zu abstrahierten IT-Infrastrukturen wie bei der öffentlichen Cloud – jedoch für einen genau bestimmten Nutzerkreis, der sich, meist örtlich verteilt, die Kosten teilt – z.B. mehrere städtische Behörden, Universitäten, Betriebe oder Firmen mit ähnlichen Interessen, Forschungsgemeinschaften und Genossenschaften. Somit wird es möglich, sehr spezifische Daten und Dienste einer eingeschränkten Nutzergruppe mit einem hohen Sicherheitsgrad zugänglich zu machen.

Cloud konkret

Die bisherigen Ausführungen sind die wichtigsten Begriffe in der Cloud-Technik. Wie eine Cloud aber tatsächlich funktioniert, kann nur an einem konkreten Beispiel erläutert werden. Es gibt nicht nur die eine Cloud, sondern viele verschiedene von unterschiedlichen Anbietern. Unangefochtener Marktführer ist die Amazon-Cloud. Seit 2006 vermarktet Amazon seine eigene Infrastrukturtechnik als Dienstleistung. Als Amazon Web Services (AWS) werden die Dienste auch anderen Nutzern zur Verfügung gestellt. Das Preismodell sieht eine transparente Abrechnung nach dem „Pay only for what you use"-Modell vor – nur die tatsächlich genutzten Ressourcen werden abgerechnet. Aktuell besitzen die vier großen Cloud-Anbieter Amazon, Microsoft Azure, IBM und Google zusammen einen Marktanteil von etwa 51 Prozent und wachsen aktuell schneller als alle anderen Anbieter. AWS ist mit etwa 31 Prozent der absolute Marktführer, während Microsoft Azure mit 9 Prozent noch vor IBM mit 7 und Google mit 4 Prozent auf dem zweiten Platz rangiert [2].

Um dem sehr amerikanisch geprägten Markt ein Gegengewicht auf dem europäischen Markt zu geben, hat 2015 die Deutsche Telekom die Open Tele-

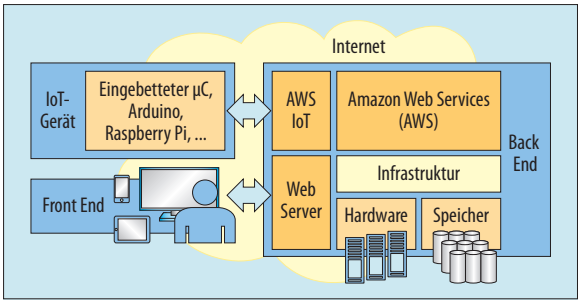

Bild 4. Wird ein IoT-Gerät mit einer Cloud verbunden, wird eine Datenquelle, das Front End und das Back End benötigt. (Quell: Wollert, Booke)

kom Cloud angekündigt, die ab 2016 eine europäische Alternative zu den etablierten US-Unternehmen darstellen soll [3].

Systementwurf für eine Cloud

Ein normaler Computer hat eine Benutzerschnittstelle – einen Webbrowser wie den Internet Explorer, Google Chrome, Firefox oder Safari. Damit im Internet zu stöbern ist quasi selbstverständlich. Der Trend rund um Industrie 4.0 und Cloud-Dienste setzt jedoch verstärkt auf intelligente und eingebettete Steuerungen, die nicht unbedingt wie konventionelle Computer aussehen. Das kann die Kaffeemaschine sein, das E-Bike, das Auto oder der Pulsmesser. Alles ist möglich. Allein die Webschnittstelle macht eine eingebettete Steuerung zum IoT-Gerät (Internet der Dinge). Sie ermöglicht die Kommunikation mit dem Internet über WiFi, Ethernet oder 3G und ist eine standardisierte High-Level-Kommunikation zu den unterstützten Web Services. Drei Beteiligte sind notwendig, um ein Gerät mit einer Cloud zu verbinden (**Bild 4**):

- das IoT-Gerät selbst, in der Regel der Datenproduzent,
- das Back End, das die Cloud-Dienste bereit stellt und
- das Front End, z.B. ein Smartphone oder Computer, der die Cloud-Dienste via Webserver nutzt.

Das Bindeglied zwischen den unterschiedlichen Instanzen ist das Toolkit vom Cloud Provider. Hier wird das AWS IoT Toolkit verwendet, um eine beispielhafte Verknüpfung von physischem IoT-Gerät und der öffentlichen Cloud von Amazon zu demonstrieren. Das Beispiel kann innerhalb des kostenlosen Nutzungs-Kontingents von Amazon (Free Usage Tier) nachvollzogen werden. AWS hat gemeinsam mit Partnern eigene Entwicklungs-Kits entwickelt, die direkt mit dem Cloud-Anbieter verknüpft werden können. Mit dem Beitrag und den weiterführenden Links soll es dem Leser ermöglicht werden, selbst ein erfolgreiches Beispiel in der AWS-Umgebung aufzubauen und in Betrieb zu nehmen [4].

Das IoT-Gerät – die Datenquelle

Das IoT-Gerät ist in der Regel der Produzent der Daten. Als eingebettetes Gerät übernimmt es vollkommen unabhängig einfache Kontroll-, Steuerungs- oder Überwachungsaufgaben und stellt die Verbindung mit dem Internet her. Hierfür eignen sich die Maker-Plattformen Arduino Yun oder Raspberry Pi. Beide verfügen über die notwendigen Schnittstellen, um Internetverbindungen betreiben zu können; neben WiFi und Ethernet ist auch mit einfachen Mitteln eine Erweiterung um die gängigen 2G- oder 3G-Techniken möglich. Der Arduino Yun zeichnet sich dabei durch seine leichte Programmierung und Handhabung aus, während der Raspberry Pi mit einer deutlich höheren

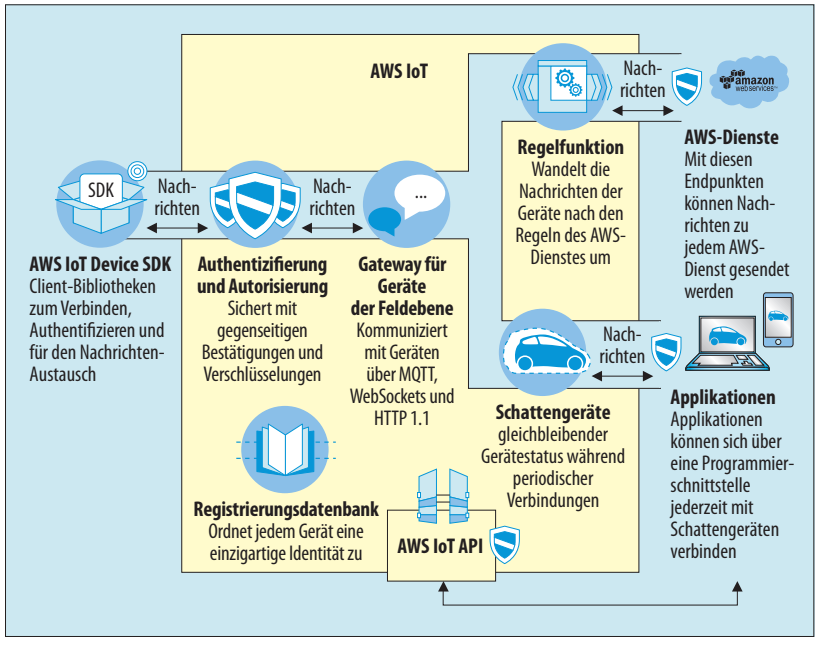

Bild 5. AWS IoT ist eine verwaltete Plattform für universelle Cloud-Dienste. (Quelle: Amazon Web Services)

MQTT – Message Queue Telemetry Transport

Das Internet der Dinge (IoT) braucht kleine und leichtgewichtige Protokolle. Die konventionellen textbasierten Protokolle wie HTTP und WSDL und SOAP sind viel zu groß, als dass kleine Geräte und Maschinen genug Kapazitäten zur leistungsstarken Interpretation hätten. Genau dort setzt MQTT an. 1999 entwickelte IBM das Protokoll für die Satellitenkommunikation, um möglichst effizient Daten zwischen Geräten mit wenigen Funktionen und geringer Rechenleistung auszutauschen. Das Protokoll ist leichtgewichtig, einfach zu implementieren, besitzt QoS- und Sicherheitsaspekte und kann als Publisher-Subscriber-Protokoll verwendet werden. Es wird über die TCP/IP-Protokollfamilie übertragen; damit ist es eine Alternative zu anderen webbasierten Protokollen. Herzstück der MQTT-Kommunikation ist der Broker. Er verwaltet die Publisher und Subscriber und die durch sie versendeten und abonnierten Topics. Ein Topic kann nahezu alles sein. Es kann ein Datum sein oder eine ganze Gruppe. Hier sind den Entwicklern quasi keine Grenzen gesetzt – MQTT ist damit datenagnostisch. Was natürlich dazu führt, dass es keine spezifischen Profile für eine Anwendungsgruppe gibt. Generell kann man feststellen, dass aktuell für die Maschine-zu-Maschine-Kommunikation OPC UA und MQTT als schlanke Datenaustausch-Protokolle für Maschinen in der Diskussion sind. Der Blick auf die Amazon-Cloud und generell in die USA zeigt hier klare Vorteile für das sehr schlanke und flexible MQTT.

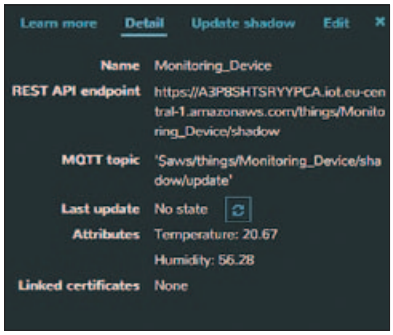

Bild 7. In der Detail-Ansicht können die Eigenschaften des Dings noch einmal überprüft werden. (Bild: Wollert, Booke)

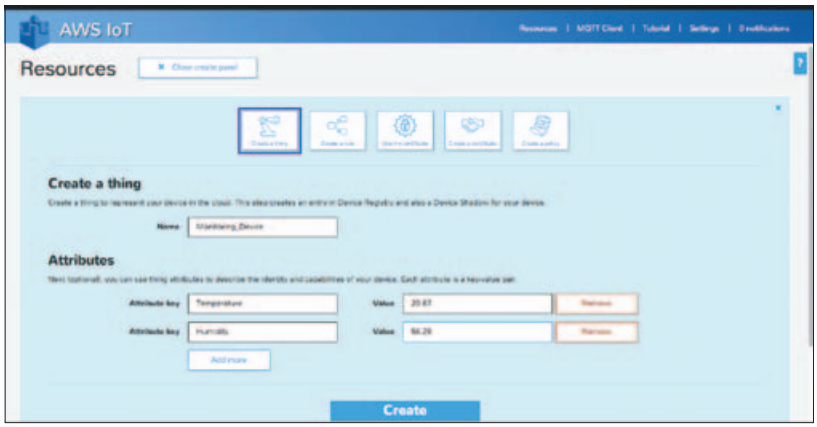

Bild 6. Über den Einstiegsdialog wurde ein neues Ding mit dem Namen „Monitoring_Device" und den Attributen „Temperature" und „Humidity" angelegt. Mit ihnen können Regeln erstellt werden, die bestimmte Aktionen auslösen. (Bild: Wollert, Booke)

Leistung größeren Anforderungen gerecht wird. Auf beiden Plattformen kann das AWS Device SDK (Software-Entwicklungs-Kit) eingesetzt werden, um eine sichere Verbindung zu der AWS-Cloud aufzubauen. Das IoT-Gerät-SDK ist als Embedded-C-SDK für C-basierte Plattformen wie Linux, RTOS, mit OpenSSL- und mbedTLS-Unterstützung, als JavaScript SDK für Node.js und als Arduino Yun SDK verfügbar.

AWS-IoT-Plattform

AWS IoT (**Bild 5**) [5] ermöglicht es, die unterschiedlichen IoT-Geräte mit AWS-Diensten mit dem Back End zu verbinden und Daten für Front Ends aufzubereiten. Daten und Interaktionen können somit sicher ausgetauscht, Gerätedaten verarbeitet und in andere Dienste eingebunden werden. Außerdem übernimmt AWS IoT das gesamte Gerätemanagement, auch wenn die Verbindung vorübergehend unterbrochen ist. AWS IoT unterstützt unterschiedliche Datenaustauschprotokolle wie HTTP, WebSockets oder MQTT (siehe **Kasten** *MQTT – Message Queue Telemetry Transport*), das speziell für eingebettete Geräte entwickelt worden ist. Innerhalb von AWS IoT können Geräte miteinander kommunizieren, auch wenn sie unterschiedliche Protokolle verwenden.

Bild 5 zeigt einen Überblick über die AWS-IoT-Dienste [5]. Eingebettete Geräte werden mit AWS IoT Device SDK zu einem Ding und können sich über ein gesichertes Gateway mit der Cloud verbinden. Dinge (Things) sind Geräte von unterschiedlichen Typen, die innerhalb von AWS als Anwendung, physisches Objekt oder als verbundener Service eingesetzt werden können. Dinge sind Sensoren oder Aktoren innerhalb ihrer lokalen Umgebung und führen dort bestimmte Aktionen durch. Das AWS-IoT-Modell basiert auf Zuständen (States) und State-Änderungen und erlaubt eine durchgängige Funktionsweise – auch wenn vorübergehend die Verbindung unterbrochen ist. Anwendungen agieren durch Schattengeräte mit anderen Diensten innerhalb der AWS-Cloud. Jedes Ding besitzt dabei einen spezifischen Namen, verschiedene Attribute und ein Schattengerät.

Schattengeräte (Thing Shadows) sind virtuelle, Cloud-basierte Organisationseinheiten von Dingen. Der digitale Schatten lebt auch dann weiter, wenn die physischen Dinge einmal offline sind. Der Schatten in der Cloud enthält den letzten Status des Geräts, sodass Anwendungen oder andere Geräte Nachrichten lesen und mit dem Gerät interagieren können. Schattengeräte bleiben im letzten gemeldeten Status und gewünschten zukünftigen Status jedes Geräts erhalten, auch wenn das Gerät offline ist. Der zuletzt gemeldete Status kann abgerufen oder ein gewünschter zukünftiger Status festlegt werden. Das geschieht entweder über die Programmierschnittstelle (API) oder über die Regelfunktion.

Die Regelfunktion (Rules Engine) ermöglicht es, auf globaler Basis IoT-Anwendungen einzusetzen, die Daten von den verbundenen Geräten zu sammeln, verarbeiten, analysieren und für weitere Aktionen zu berücksichtigen, ohne dass die Infrastruktur verwaltet werden muss. Die Regelfunktion wertet im AWS IoT ankommende Nachrichten aus, formt sie um und sendet sie je nach

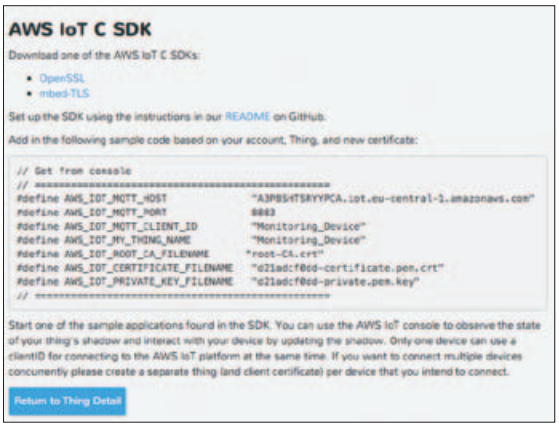

Bild 8. Mit dem Setup Wizard können im Laufe der Installation Informationen über Authentizität und Verschlüsselung angezeigt werden. (Bild: Wollert, Booke)

festgelegten Regeln an ein anderes Gerät oder einen Cloud-Dienst weiter. Eine Regel kann auf Daten von einem oder vielen Geräten angewendet werden und eine oder mehrere Aktionen parallel vornehmen.

Ein Beispiel macht die theoretische Betrachtung etwas deutlicher: Angenommen das IoT-Gerät ist ein Temperatursensor. Er würde nichts anderes tun als die Temperaturwerte in die Cloud zu übermitteln. Überschreiten die Temperaturwerte einen bestimmten Schwellenwert, kann mit der Regelfunktion eine Regel definiert werden, die einen weiteren Cloud-Dienst auslöst. Zum Beispiel könnte der Temperaturdienst die nächst gelegenen Schulen informiert, dass Hitzefrei zu geben ist. Es könnten auch statistisch komplexe Auswertungen gemacht werden. Hier sind nahezu unbegrenzte Möglichkeiten für Dienste gegeben, die durch die Cloud erbracht werden können. Die Regeln werden bei AWS in einer SQL-ähnlichen Syntax verfasst. Das Gateway für Geräte oder auch Message Broker ist der Nachrichtenmanager, der alle relevanten Webprotokolle unterstützt. MQTT, WebSockets und HTTP 1.1 sind die hier üblichen Standards. Darüber hinaus bietet der Message Broker einen sicheren Mechanismus für Dinge und IoT-Anwendungen um Nachrichten und Informationen auszutauschen. Über das MQTT-Protokoll wird ein Publisher-Subscriber-Mechanismus aufgebaut. So ermöglicht es AWS IoT einem verbundenen IoT-Gerät Daten an mehrere Abonnenten zu senden. Über die http-REST-Schnittstelle (siehe **Kasten** *REST – Representional State Transfer*) können Nachrichten an verschiedene Endpunkte gesendet werden. Neben den Standardprotokollen ist auch die Integration eigener oder älterer Protokolle möglich.

AWS IoT bietet ein eigenes Device SDK. Es ermöglicht, dass ohne nähere Kenntnisse ein Gerät oder eine mobile Anwendung einfach und schnell mit einer Cloud verbunden werden kann. Das Device SDK unterstützt C, JavaScript und die Arduino-Maker-Plattform. Es enthält zudem die Client-Bibliotheken, ein Entwicklerhandbuch und das Portierungs-Handbuch für Gerätehersteller.

Außerdem ermöglicht AWS IoT die gegenseitige Authentifizierung und Autorisierung an allen Verbindungspunkten, sodass Daten nie ohne geprüfte Identität zwischen Geräten und AWS IoT ausgetauscht werden

Bild 9. Das Dashboard einer Web-App. (Bild: Wollert, Booke)

REST – Representational State Transfer

REST ist ein Architekturstil oder Design-Muster für die sinnvolle Nutzung einer Servicearchitektur im Internet. Es ist damit weder Protokoll noch Standard, sondern beschreibt ausschließlich, wie Webstandards eingesetzt werden sollen. Im Wesentlichen wird die Semantik des HTTP-Protokolls übernommen, das jedoch nicht zwingend verwendet werden muss.

- GET – fragt die Repräsentation einer Ressource ab. GET-Anfragen müssen nicht zwingend beantwortet werden.
- POST – mit dem POST-Befehl kann einer Ressource etwas hinzugefügt werden. POST ermöglicht Veränderungen in der Ressource durch das Ändern von Daten oder Eigenschaften.
- PUT – ermöglicht das Erstellen neuer Ressourcen, oder der Inhalt von Ressourcen kann ersetzt werden.
- DELETE – mit der Lösch-Anweisung können Ressourcen gelöscht werden.

Mit den Dienstmethoden können alle relevanten Transaktionen von Systemen durchgeführt werden. Wie die Interaktion konkret aussieht, bleibt den Anwendern von REST überlassen. Heute werden durch nahezu alle Webserver, wie Appache und IIS, oder Webcontainer, wie Tomcat, mit einer generischen REST-Schnittstelle ausgerüstet. REST wird auch von Amazon, Google und FlickR verwendet und ist in der Welt des Internets nicht mehr wegzudenken.

können. Es unterstützt die AWS-Methode der Authentifizierung – mit der Bezeichnung „SigV4" – sowie eine Authentifizierung auf der Basis von X.509. Bei HTTP-Verbindungen können beide Methoden verwendet werden. Dagegen wird bei MQTT-Verbindungen eine auf Zertifikaten basierende Authentifizierung verwendet. Bei Web-Socket-Verbindungen kann ebenfalls SigV4 verwendet werden. Zudem kann einem einzelnen Gerät unverzüglich der Zugriff entzogen werden, sollte das erforderlich sein. AWS IoT unterstützt darüber hinaus Verbindungen von mobilen Anwendungen der Benutzer über Amazon Cognito. Der Service übernimmt alle erforderlichen Schritte zum Erstellen eines eindeutigen Bezeichners für die Benutzer und der jeweiligen Anwendung. Zudem ruft er temporäre AWS-Anmelde-Informationen mit eingeschränkten Berechtigungen ab.

Die Registrierungsdatenbank erstellt eine Identität für Geräte und verwaltet Metadaten, wie Attribute und individuellen Fähigkeiten der Geräte. Sie weist jedem Gerät eine eindeutige Identität zu, die unabhängig von der Art des Geräts oder der Art der Verbindungsaufnahme einheitlich formatiert ist. Außerdem werden Metadaten unterstützt, die die Fähigkeiten eines Geräts beschreiben – z.B. ob ein Sensor Temperaturen meldet und ob die Daten in Fahrenheit oder Celsius gemessen werden.

Realisierung in der Amazon-Cloud

Zunächst sollte ein eingebettetes Gerät verwendet werden – es muss die notwendigen Protokolle prinzipiell beherrschen. Auf den Amazon-Cloud-Seiten [4] stehen entsprechende Entwicklungs-Kits zur Verfügung. Sie sind von unterschiedlichen Halbleiterherstellern wie Arduino Yun, Broadcom, Renesas, Intel und TI. Hier im Speziellen wurde eine Seeeduino-Arduino-YUN-Plattform verwendet. Auch hier ist ein passendes Device SDK für die Inbetriebnahme notwendig. Amazons „Erste Schritte"-Leitfaden beschreibt, wie die jeweilige Plattform individuell eingerichtet werden kann. In der Regel wird man auf ein GitHub-Verzeichnis verwiesen, welches die notwendigen Bibliotheken und Kurzanleitungen beinhaltet. War die Inbetriebnahme erfolgreich und konnte die Registrierung bei AWS durchgeführt werden, ist man in der Amazon-Cloud-Welt angekommen. Unter dem Menüpunkt IoT kann ein neues Ding angelegt werden (**Bild 6**).

Nun wird durch AWS IoT automatisch ein neues Ding mit entsprechendem REST-API-Endpunkt und MQTT Topic erstellt. REST und MQTT sind spezifische Verhaltensmuster und Protokolle, die für eine Cloud-Kommunikation wichtig sind. Zusätzlich sind weitere Protokolle für eine Service-orientierte Architektur denkbar. Ziel ist es, eine standardisierte und portable Schnittstelle zu schaffen, auf die andere Cloud-Dienste Zugriff haben und mit dem – zunächst virtuellen – Ding kommunizieren können. Welche Dinge dem virtuellen Schatten eine physische Identität geben und wie die Verknüpfung zur Regelbasis erfolgt, wird mit der Detail-Ansicht realisiert (**Bild 7**). Mit „Connect a device" wird ein spezifisches Zertifikat erstellt, welches dem eingebetteten Gerät eine eindeutige Identität gibt. Während der Installation werden darüber hinaus verschiedene Konfigurationen vorgenommen, um die jeweiligen unterstützten Sicherheitsprotokolle einzubinden und damit alle relevanten Informationen bezüglich Authentizität und Verschlüsselung bereit zu stellen. Im Zuge der Installation werden die Informationen angezeigt (**Bild 8**). Anschließend ist das eingebettete Gerät als Ding über ein gesichertes Gateway mit der Cloud verbunden. Security by Design ist damit ein wichtiger essenzieller Baustein für die Cloud-Konnektivität. Der Prozess lässt sich auch über die Kommandozeile durchführen (AWS CLI) oder durch entsprechende Skripte über die AWS-API automatisieren.

Visualisierung mit dem Front End

Nachdem die eingebetteten Geräte als Ding mit der Cloud-Infrastruktur kommunizieren können, sollten die durch das IoT-Gerät bereitgestellten Daten für den Anwender visualisiert werden. Das kann nur durch einen Webserver-Zugriff erfolgen. Nur Webdienste sind plattformunabhängig und können somit alle Endgeräte ohne Installation zusätzlicher Software mit Daten versorgen. Ein Webserver im Cloud Back End (**Bild 4**) ist damit verbindlich. Er muss den Zugang zu den bereitgestellten Daten des IoT-Geräts ermöglichen. Die Verbindung schafft bei AWS der Elastic Beanstalk, eine Dienstleistung von Amazon. Sie ermöglicht die Skalierung von Webanwendungen und Webdiensten in den unterschiedlichsten Programmiersprachen von Java über .NET und PHP zu Phyton, Ruby oder Docker. Wird der Anwendungs-Code hochgeladen, übernimmt Elastic Beanstalk automatisch die Kapazitätsbereitstellung, Lastverteilung, automatische Skalierung und die Statusüberwachung der Anwendung. Elastic Beanstalk kann dabei auf durch AWS IoT genutzte Ressourcen – z.B. Datenbanken – zurückgreifen und die Daten entsprechend der eigenen Anwendungslogik bereitstellen [6].

Das eigentliche Front End kann entweder durch eine Web-App oder eine App für das Smartphone oder Tablet realisiert werden. **Bild 9** zeigt das Dashboard einer Web-App. Das Dashboard basiert auf HTML5 und dem JavaScript Framework ReactJS. Die Daten werden über Elastic Beanstalk aus der Datenbank gelesen. Für unterschiedliche Techniken sind Dashboard-Vorlagen verfügbar, sodass eine entsprechende Seite mit kurzer Entwicklungszeit realisiert werden kann. Und damit ist der Service fertig – Sensor to Cloud ist Realität geworden.

Eine Frage des Vertrauens

Embedded to Cloud ist gar nicht so schwierig. Amazon bietet mit dem AWS ein umfassendes Angebot mit hoher Integrationsdichte zwischen den einzelnen Diensten. Wie bei allen Cloud-Anbietern gibt es nur eine Möglichkeit: Man muss sich dem Anbieter und seinen Diensten ausliefern, um die Plug-and-Play-Funktion nutzen zu können. Zwar sind die Protokolle der unterschiedlichen Cloud-Anbieter identisch, aber wie die Instrumentierung erfolgt, ist Gegenstand des Framework. Die gute Nachricht: Security by Design ist Standard. Zwar sind unterschiedliche Verschlüsselungsverfahren mit verschieden sicheren Verfahren möglich, aber das betrifft hauptsächlich die eingesetzten Mikrocontroller. Hat man sich für eine Cloud entschieden, dann gilt der Blick den Rechenzentren und dem Vertrauen, das man dem Anbieter entgegenbringt. Bei Amazon ist das Haupt-Rechenzentrum in Frankfurt. Es bietet mit drei zugeordneten Zonen eine hohe Ausfallsicherheit bei gleichzeitiger Datenspeicherung in Deutschland. Mit dem AWS IoT bietet Amazon zahlreiche Möglichkeiten, Geräte aus der realen Welt mit der Cloud und den verschiedenen AWS-Diensten zu verknüpfen. Die unterschiedlichen Dienste von AWS werden immer enger zusammenwachsen und somit wird für die Zukunft eine umfassende Dienstleistungs-Plattform verfügbar sein. *cd*

Literatur

[1] *Mell, P. et alii*: The NIST Definition of Cloud Computing. NIST – National Insitute of Standards and Technology. U.S. Department of Commerce. 2011. nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

[2] AWS Remains Dominant Despite Microsoft and Google Growth Surges. 2016. www.srgresearch.com/articles/aws-remains-dominant-despite-microsoft-and-google-growth-surges

[3] *Marwan, P.*: Open Telekom Cloud soll ab 2016 mit AWS konkurrieren. 2015. www.zdnet.de/88250050/open-telekom-cloud-soll-ab-2016-mit-aws-konkurrieren/

[4] AWS IoT-Starterkits. aws.amazon.com/de/iot/getting-started/#kits

[5] AWS IoT. aws.amazon.com/de/iot

[6] AWS Elastic Beanstalk – PaaS Application Management. aws.amazon.com/de/elasticbeanstalk/

Andreas Booke
hat seinen Abschluss im Fach Mechatronik bei Dr.-Ing. Jörg Wollert an der Fachhochschule Aachen absolviert. Anschließend arbeitete er bei Wollert als wissenschaftlicher Mitarbeiter im Bereich Embedded Systems, IoT und Wireless Technologies und ist seit Mai 2016 Mitgründer eines Unternehmens für die Anbindung von dezentralen Energiespeicherlösungen.

Prof. Dr.-Ing. Jörg Wollert
ist Professor für Mechatronik und eingebettete Systeme an der Fachhochschule Aachen und als Dozent und Berater in den Themengebieten industrielle Kommunikation, Echtzeitsysteme und Embedded Systems tätig. Seit gut 20 Jahren beschäftigt er sich mit dem Design und der Implementierung verteilter objektorientierter Echtzeitsysteme sowie Gateway-Technologien zwischen kabelgebundenen und funkbasierten Systemen in industriellen Anwendungen.
wollert@fh-aachen.de