Überblick
Software Defined »X«

Grundlagen und Status Quo

www.bitkom.org

bitkom

Herausgeber

Bitkom e. V.
Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V.
Albrechtstraße 10 | 10117 Berlin

Ansprechpartner

Christian Herzog | Bitkom e. V.
T 030 27576-270 | c.herzog@bitkom.org

Redaktion

- Frank Beckereit | Dimension Data Germany AG & Co.KG
- Ingolf Wittmann | IBM Deutschland GmbH
- Lorenz Keller | Oracle Deutschland
- Georg Mey | Netapp Deutschland GmbH
- Ulrich Hamm | Cisco Deutschland GmbH
- Thomas Harrer | IBM Deutschland
- Dr. Georgios Rimikis | Hitachi Data Systems Deutschland GmbH

Verantwortliches Bitkom-Gremium

AK Server-, Storage und Netze

Inhaltsverzeichnis

Verzeichnis der Abbildungen

1 Überblick

Software Defined »X«

In letzter Zeit werden Begriffe wie »Software Defined Datacenter«, »Software Defined Networking«, »Software Defined Storage« etc. zunehmend genannt. Wenngleich jeder der Begriffe eine unterschiedliche Bedeutung hat, so verbindet sie alle mit der Bezeichnung »Software Defined X« ein wiederkehrendes Konzept: die (einfache) Modellierung und Bereitstellung von zum Teil sehr komplexen Ressourcen bzw. Diensten.

Traditionell sind die Ressourcen Server, Storage & Netzwerk einzelne, in sich eng verwobene Funktionsblöcke aus Hardware & Software. Dezentral administriert arbeiten sie überwiegend eigenständig in Silos. Die Bereitstellung von benötigten Diensten aus diesen Ressourcen ist fest an die verwendete Hardware gekoppelt. Die damit einhergehende Einschränkung bei der Flexibilität der Nutzung wird offensichtlich. Server, Storage und Netzwerke in Rechenzentren müssen häufig neu konfiguriert werden, um veränderten Kundenanforderungen zu entsprechen, Anwendungen anzupassen oder auf veränderte Arbeitslast-Anforderungen reagieren zu können. Die Starrheit hat konkrete Auswirkungen auf organisatorische Ergebnisse, Geschäftsanforderungen. Fehlende Flexibilität verlängert Projektlaufzeiten. Dieses führt zwangsläufig zu höheren zusätzlichen Kosten. Zugleich können Gewinne mitunter gar nicht erst realisiert werden, da aufgrund mangelnder Agilität Geschäftsideen nicht oder nicht zeitgerecht umgesetzt werden können. Um flexibler und agiler zu werden bedarf es der Entkopplung der Ressourcen von den physischen Geräten. Eine Virtualisierungsschicht ermöglicht den Zugriff auf die Ressourcen. In der Folge ist es möglich mehrere, netzwerkbezogene virtualisierte Ressourcen zu bündeln, zu verknüpfen und zu höherwertigen Diensten zu veredeln. Je hochwertiger und spezieller die bereitgestellten Dienste, desto komplexer sind in der Regel die dafür benötigten Architekturen.

Diese komplexen hochwertigen (Rechen-)Dienste möchte der Nutzer auf der anderen Seite einfach, zuverlässig, skalierbar und mit hoher Qualität nach dem einfachen Abo-Prinzip nutzen. Die so beschriebene Idee des Cloud Computing[1] unterscheidet zunächst zwischen drei Arten bzw. Schichten von Diensten:

- Infrastruktur (IaaS – Infrastructure as a Service): Sie stellt die unterste Schicht im Cloud Computing dar. Der Benutzer greift hier auf bestehende Dienste innerhalb des Systems zu, verwaltet seine Recheninstanzen (siehe virtuelle Server) allerdings weitestgehend selbst.

- Plattform (Platform as a Service): Der Entwickler erstellt die Anwendung und lädt diese in die Cloud. Diese kümmert sich dann selbst um die Aufteilung auf die eigentlichen Verarbeitungseinheiten. Im Unterschied zu IaaS hat der Benutzer hier keinen direkten Zugriff auf die Recheninstanzen. Er betreibt auch keine virtuellen Server.

1 Vgl. ↗NIST Definition of Cloud Computing

- Anwendung (Software as a Service): Die Anwendungssicht stellt die abstrakteste Sicht auf Cloud-Dienste dar. Hierbei bringt der Benutzer seine Applikation weder in die Cloud ein, noch muss er sich um Skalierbarkeit oder Datenhaltung kümmern. Er nutzt eine bestehende Applikation, die ihm die Cloud nach außen hin anbietet.

Cloud-Dienste wie Database as a Service (DBaaS), Middleware as a Service (MWaaS), Java as a Service (JaaS) lassen sich in die zuvor beschriebenen Schichten unterordnen. Sie zeigen aber auch die Vielschichtigkeit, Komplexität und Verbundenheit mit herstellerspezifischen Diensten.

Auch wird schnell klar, dass Server-Virtualisierung allein nicht die Lösung ist, um diese höherwertigen Dienste zu erstellen, anzubieten und zu betreiben. Wie lassen sich diese komplexen Dienste und Ressourcen einfach und flexibel konfigurieren, automatisieren, standardisieren, zuweisen, voneinander absichern, skalieren, überwachen, entziehen, abrechnen? Es bedarf entsprechender Software, die in der Lage ist, die Ressourcen und Dienste einfach und flexibel abrechenbar darzubieten.

»Software Defined Storage« umfasst das Konzept der Speichervirtualisierung. Die Software zur Verwaltung des Speichers wird von der darunterliegenden Hardware getrennt. Die Software bietet dabei zusätzliche Dienste, wie beispielsweise Speicher unterschiedlicher Güte- und Schnelligkeitsklassen, zu unterschiedlichen Kosten und für unterschiedliche Aufgaben (Speicher für Transaktionssysteme, Speicher für Backups, ...) mit Intelligenten Mechanismen (De-Duplizierung, Replikation, ...) bereitzustellen.

Auch beim »Software Defined Networking« werden die zwei wesentlichen Komponenten, die Control-Plane, die entscheidet, wohin Daten geschickt werden sollen und die Data-Plane, die für den Transport der Daten zum Ausgewählten Bestimmungsort verantwortlich ist getrennt. Dadurch lassen sich auch hier virtuelle Netzwerke über Applikationstechnik flexibler und agiler verwalten als bisher.

Das »Software Defined Datacenter« bedient sich der zuvor genannten Visualisierungs-Konzepte, um Ressourcen und Dienste (Services) zu bündeln und am Ende die Vision von IT as a Service einfach, agil und flexibel umsetzen zu können. Aufgrund der Vielschichtigkeit der Dienste und Ressourcen zum Einen, der großen Zahl an Technologieanbietern, Produkten und Standards auf der anderen Seite, wird das SDDC von Kritikern in der Praxis mehr als Marketing Hype gesehen, als eine bereits realisierbare Option. Ergänzend kommt hinzu, das viele Hersteller Ihrerseits der Komplexität und Vielschichtigkeit der Dienste begegnen, indem sie integrierte Systeme oder Appliances anbieten, welche die komplexen und höherwertigen Dienste über alle Architekturschichten integriert anbieten (→ Vergleiche Bitkom-Papier »Integrierte Systeme«). Verlockend hierbei ist nicht zuletzt, dass die Integrationsaufwände, wie auch Support und Gewährleistung bei dieser Lösung auf Seiten der Hersteller und nicht auf Seiten der Rechenzentrums-Betreiber liegen.

2 Software Defined Environment

2.1 Einführung

Nach Server- und Storagevirtualisierung, Cloud Computing dreht sich die aktuelle Diskussion um das Managen dieser Umgebungen mit einem neuen Ansatz den sogenannten »Software Defined Environments« (SDE). Eine softwaredefinierte Umgebung soll es Unternehmen ermöglichen IT-Services auf eine besonders effiziente Art und Weise bereitzustellen und sich unabhängiger von proprietären Hardwareplattformen und Softwareadministrationswerkzeugen zu machen. Dabei werden nicht nur Server oder Speicherkomponenten betrachtet, sondern alle Komponenten welche zu einer IT Infrastruktur gehören. Also auch Netzwerke und Workloads.

Man kann es auch wie folgt definieren:

- Software – abstrahiert und virtualisiert IT-Infrastruktur-Ressourcen, welche von Software verwaltet werden.
- Defined – Anwendungen definieren automatisch die Anforderungen an die Infrastruktur und deren Konfigurationen.
- Environment – IT-Infrastruktur, welche sich über mehrere Umgebungen erstreckt und über das Rechenzentrum hinausgeht.

Dabei wird der Ansatz verfolgt, Hardwarefunktionalitäten in Software abzubilden. Dies basiert auf Standards bzw. Industriestandards unter Verwendung von Open Source als Basistechnologie um die Interoperabilität zwischen den Herstellern und den unterschiedlichen HW/SW Architekturen zu ermöglichen.

Damit untergliedert sich SDE in drei Teilbereiche:

- Software Defined Computing – umfasst die Serverkomponenten als »Bare Metall« und deren virtuellen Umgebungen unabhängig von der eingesetzten Technologie. Damit soll die Auslastung gesteigert, die Compliance sichergestellt und die Kosten im Rechenzentrum gesenkt werden.
- Software Defined Storage – umfasst die Speicherkomponenten zum Managen der Datenzunahme z. B. durch Mobil- und Social-Media-Technologien durch das Zusammenfassen von physischen Speicherressourcen in anbieterunabhängige Pools um eine bessere Auslastung zu erreichen.
- Software Defined Networking – umfasst die Netzwerkkomponenten und bietet eine zukunftsorientierte Alternative für das Management auf Ebene der physikalischen Geräte.

Häufig werden diese drei Teilbereiche in den Unternehmen noch als Silos betrachtet, sowohl von der organisatorischen als auch architektonischen Seite. SDE hat den Anspruch alle Aspekte in einem ganzheitlichen Ansatz zu erfassen und managen. Dafür sind De-facto-Standards wie Openstack, TOSCA oder OpenFlow wichtig, auf denen die Implementierungen wie das Open-Daylight Projekt für den SDN-Bereich basieren.

Im Anwendungsbereich halten sogenannte Patterns Einzug, welche eine komplette Anwendungsumgebung inclusive Middleware wie z. B. Datenbanken und Webapplikationen und die entsprechenden Virtualisierungs- und Netzwerkdefinitionen umfassen. Diese Patterns können mit SLAs versehen werden und von dem Managementbereich der SDE je nach Last vergrößert oder verkleinert, oder sogar auf andere IT-Infrastrukturen umgezogen werden. Insbesondere private und Hybrid-Cloud-Umgebungen sind hier im Fokus von SDE, um mit den Automatisierungsmechanismen mehr Sicherheit und Verfügbarkeit in die Anwendungsumgebungen zu bekommen bzw. Kosten einzusparen.

2.2 Bedeutung

Für das einzelne Unternehmen bedeutet dies seine IT-Infrastruktur ganzheitlich zu betrachten und zu organisieren. Ein architektonischer Blueprint sollte die Ziel-IT-Infrastruktur für die Unternehmensprozesse und Anwendungsumgebungen definieren und wie diese schrittweise erreicht werden kann.

Da die unterschiedlichen IT-Unternehmen im Markt ihre eigenen Definitionen und Sichtweisen von SDE haben, ist bei einem Transformationsprojekt auf die Integrationsfähigkeiten von heterogenen Infrastrukturen und die Unterstützung von (Industrie)-Standards ein besonderes Augenmerk zu legen.

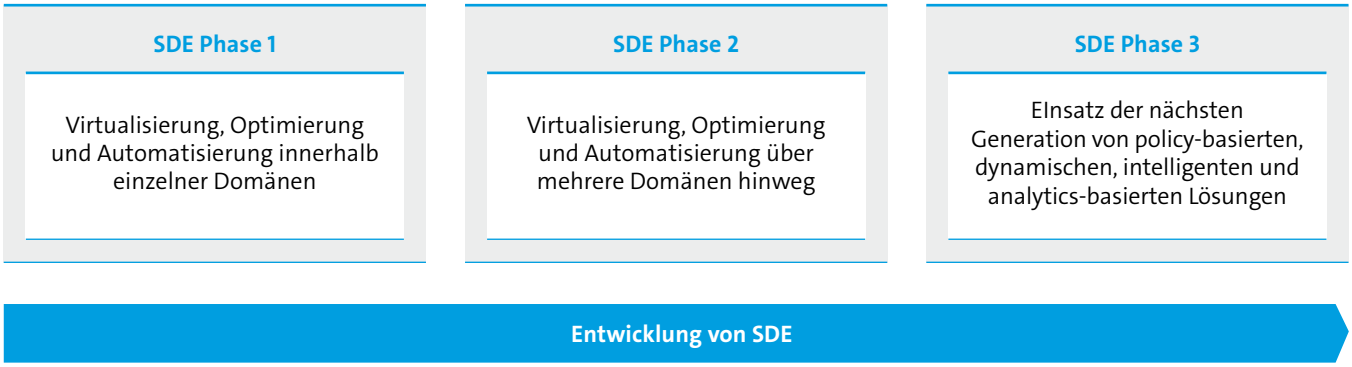

Abbildung 1: Phasen der Entwicklung von Software Defined Environments

Viele Unternehmen haben in den letzten Jahren Virtualisierung von IT-Anwendungen und IT-Infrastrukturen vorangetrieben. Für die IT-Infrastruktur spielen die Ansätze der Server-Virtualisierung eine wichtige Rolle bei der Transformation in Richtung einer Industrialisierung der IT. Bitkom bietet seit 2006 einen Glossar zur Servervirtualisierung an. Um die Bedeutung der Servervirtualisierung zu unterstreichen, bietet der Bitkom-AK Server, Storage und Netzwerk die aus vier Teilen bestehenden Leitfäden zur Servervirtualisierung zur Verfügung.

3 Software Defined Computing

3.1 Einführung

Wenn es bei Software Defined Environments darum geht, die richtigen IT-Ressourcen für bestehende und neue Anwendungen flexibel, dynamisch und automatisiert zur Verfügung zu stellen, dann bezeichnet der Bereich Software Defined Computing die Compute-Ressourcen, die für die Laufzeitumgebungen der Anwendungen genutzt werden. Der Begriff Software Defined drückt dabei aus, dass die Eigenschaften der Compute-ressourcen durch Software und Parameter definiert werden. In der Regel können im Software Defined Computing alle Compute-Ressourcen über Application Programming Interfaces (APIs) oder andere SW-Schnittstellen erzeugt und verworfen werden sowie Änderungen an den Charakteristika der laufenden Umgebungen vorgenommen werden.

Die Compute-Ressourcen nutzen dabei jeweils Netzwerk- und Storage-Ressourcen. Mehrere Software-Defined-Umgebungen können über Orchestrierung in eine größere Infrastruktur zusammengebunden, gemeinsame bereitgestellt und automatisch angepasst werden.

Ein besonderer Anspruch von Software Defined Computing ist, dass die Optimierung der Umgebungen nach vorgegebenen Service-Level-Anforderungen der Anwendungen automatisch erfolgt. Traditionell werden Umgebungen in einer Bottom-Up-Methode verwaltet: ausgehend von einem statischen Infrastruktur-Design werden Umgebungen manuell bereitgestellt, installiert, konfiguriert und optimiert. In einem Software Defined Environment kommt ein Top-Down-Ansatz zum Tragen. Ausgehend von den Anforderungen der Anwendung wird eine Infrastruktur automatisch bereitgestellt, die sich während des Lebenszyklus der Anwendung laufend optimiert und anpasst. Um diese Anpassung leisten zu können, müssen die Elemente und Ausprägungen des Software Defined Computings flexibel anpassbar sein.

3.2 Ausprägungen

Compute-Ressourcen können in unterschiedlichen Ausprägungen und Abstraktionen zum Einsatz kommen. Die nachfolgende Abbildung zeigt die drei im Folgenden erläuterten Ausprägungen von Software Defined Computings, Bare-Metal-Systeme, virtualisierte Systeme und Container:

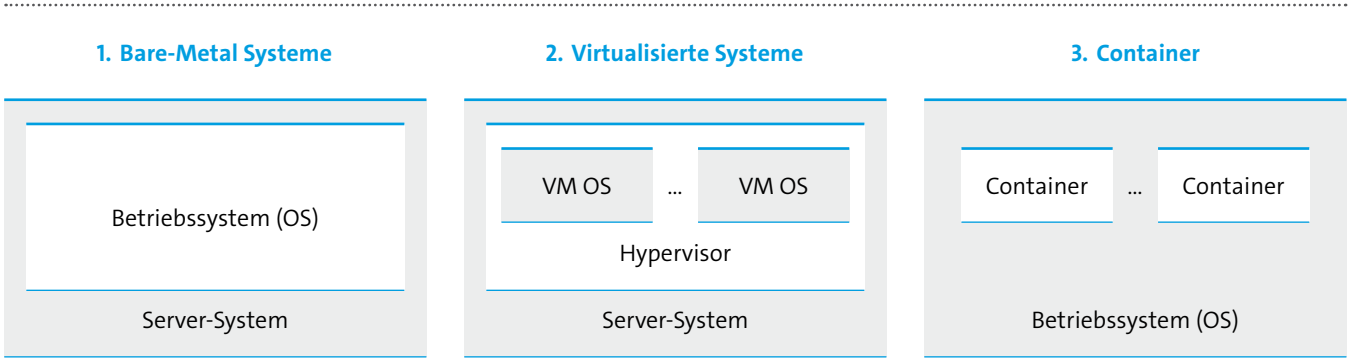

Abbildung 2: Typen von virtualisierten Systemen

Das Betriebssysteme für Container kann entweder in einem Bare-Metal-System laufen oder in einem Virtualisierten System.

3.2.1 Bare-Metal Systeme

Die Basis jeder Verarbeitung sind physische Server, die aus Prozessoren, Hauptspeicher und I/O-Bereichen zusammengesetzt sind. Diese lassen sich direkt als Basis für Laufzeitumgebung für Anwendungen verwenden. Der dafür erforderlichen Software-Stack läuft in einem Betriebssystem, das direkt auf der Hardware läuft. Diese Ausprägung wird heute üblicherweise Bare-Metal-Systeme genannt und kommt in Bereichen zum Einsatz, bei denen die Performance von dedizierten physischen Ressourcen und strikte Abschottung gegenüber anderen Laufzeitumgebungen erforderlich ist.

3.2.2 Virtualisierte Systeme

Heute stellen virtualisierte Systeme den Standard für Software Defined Computing dar. Mit dem Einsatz von Hypervisoren, teilweise auch Partitionierung, können mehrere Laufzeitumgebungen mit jeweiligem Software-Stack und Betriebssystem auf einem physischen Server laufen. Die virtuellen Umgebungen teilen sich die physischen Compute-Ressourcen, so dass diese Ressourcen gut ausgelastet werden können. Virtualisierte Systeme kommen dort zum Einsatz, wo eine flexible, kosteneffiziente Bereitstellung von kompletten Laufzeitumgebungen einschließlich eigenem Betriebssystem erforderlich ist, gleichzeitig aber das Sharing von Ressourcen mit anderen virtuellen Gästen auf einem physischen System angestrebt wird.

3.2.3 Container

Eine dritte Klasse von Software Defined Computing stellt die Container-Technologie dar. Innerhalb eines Betriebssystems schaffen Container isolierte Laufzeitumgebungen für verschiedene Anwendungen und Mandanten. Alle diese Umgebungen nutzen Ressourcen des Betriebssystems gemeinsam; jede Umgebung ist für sich abgeschottet, so dass sie getrennt administriert werden kann. Container sind im Vergleich zu virtuellen Systemen sehr effizient, da sie kein eigenes Betriebssystem benötigen. Container kommen dort zum Einsatz, wo sehr viele Laufzeitumgebungen schnell und kostengünstig bereitgestellt werden müssen und keine strikte Trennung auf Betriebssystem-Ebene erforderlich ist.

3.3 Funktionalitäten für Software Defined Computing

Software Defined bedeutet, dass Infrastruktur-Elemente durch Software-Aufrufe erzeugt, verändert, genutzt und verworfen werden können. In der Regel stehen die erforderlichen Funktionalitäten durch eine API zur Verfügung.

3.3.1 Erzeugen von Umgebungen

Beim Erzeugen von Umgebungen unterscheiden sich die drei Ausprägungen von Software-Defined Computing. In Bare-Metal-Umgebungen bedeutet ein software-definiertes Erzeugen eine automatisierte Basisinstallation eines Betriebssystems auf einem Serversystem, bei dem die I/O-Anbindung für verschiedene Netze und für Storage so erfolgt, dass die für das System erforderlichen I/O-Ressourcen genutzt werden können. Dazu gehören auch die Bereitstellung des Software-Stacks in diesem Betriebssystem, die Einrichtung von Benutzern und vieles mehr.

In virtualisierten Systemen ist der Aufsatzpunkt die vergleichbare Bare-Metal-Implementierung des Hypervisors. Ein laufender Hypervisor kann einen neues Gastsystem erzeugen und mit virtuellen Ressourcen ausstatten, in das ein Betriebssystem installiert werden kann oder ein vorgefertigtes Image angepasst und gestartet werden kann. Auch hier erfolgen Parametrisierungen für I/O, Software, Benutzer und vieles mehr.

Das Erzeugen von Containern setzt voraus, dass bereits eine Bare-Metal-Umgebung generiert wurde und ggf. darin ein virtualisiertes System implementiert wurde, in dem ein Betriebssystem läuft und parametrisiert ist. Innerhalb dieses Betriebssystems lässt sich ein neuer Container einfach und schnell erzeugen. Dabei sind Parametrisierungen für Ressourcen, Software, Benutzer und anderes möglich.

3.3.2 Verändern von Umgebungen

Einer der Nutzenversprechen von Software-Defined Environments ist, Anwendungsumgebungen dynamisch aufgrund von vorhandenen Service Level Policies anzupassen um kontinuierliche Optimierung der Umgebungen mittels der am besten verfügbaren Ressourcen zu erreichen. Dafür müssen Software Defined Compute Umgebungen Verändert werden können.

Innerhalb des Lebenszyklus einer Laufzeitumgebung erfolgen viele Änderungen. Sei es, dass Ressourcen hinzugenommen werden oder entfernt werden oder dass sich die installierte Software ändert, dass Benutzer und Security-Parameter verändert werden. Bei den Ressourcen unterscheiden sich die Umgebungen: Bare-Metal-Umgebungen sind relativ unflexibel bezüglich CPU-, Hauptspeicher und IO-Adapter-Ressourcen, da die Änderung dieser Komponenten mit manuellen Umbauten verbunden wäre. Ausnahmen sind Modelle einzelner Hersteller, Kapazitäten bedarfsgerecht freizuschalten. Diese Art von Ressourcen für virtualisierte Umgebungen und Container lassen sich dagegen solange einfach verändern, solange insgesamt Ressourcen zur Verfügung stehen. Weitere Änderungen betreffen die Netzwerke und die Storage-Umgebung, deren Anbindung innerhalb der Compute-Umgebung verändert werden kann.

Im Falle von virtuellen Systemen kann die Verlagerung auf einen anderen Host, im Falle von Containern die Verlagerung auf ein anderes Betriebssystem dazu beitragen, dass Ressourcen optimiert werden können. Auch die Laufzeitumgebung innerhalb von Bare-Metal-Server kann prinzipiell von einem physischen System auf ein anderes umgezogen werden, Umzüge im laufenden Betrieb sind aber eher bei virtualisierten Systemen und bei Containern etabliert.

3.3.3 Löschen von Umgebungen

Am Ende des Lebenszyklus von Umgebungen werden Ressourcen wieder freigegeben. Das Löschen und Freigeben von Containern und virtuellen Umgebungen innerhalb von bestehenden Systemen schafft die Möglichkeit, die freiwerdenden Ressourcen zu andere Umgebungen nutzbar zu machen. Im Falle von Bare-Metal-Systemen kann sich entweder der Lebenszyklus über die gesamte Nutzungsdauer von Servern erstrecken. In dem Fall werden die Laufzeitum-gebungen auf neue Server migriert. Wenn die Lebenszyklen der Umgebungen kürzer als die Nutzungsdauer der Server ist, dann werden nur die Ressourcen freigegeben. Die freien Server können danach für neue Umgebungen genutzt werden.

3.4 Status und Ausblick

Heutige virtualisierte Compute-Umgebungen aus relativ statischen und homogenen Ressourcen Pools aufgebaut. Die Umgebungen basieren in der Regel auf vordefinierten, virtuellen Maschinen (VMs). Anwendungs-Workloads werden statisch und manuell mittels der vordefinierten VMs auf diese Ressourcen Pools verteilt.

Software Defined Environments werden aufgrund der spezifizierten Charakteristika der Anwendungen die erforderlichen Compute-Ressourcen automatisch erzeugen, verändern und am Ende des Lebenszyklus Ressourcen wieder zurückgeben. Dabei kann z. B. die Anzahl der Umgebungen automatisch mit den jeweiligen Performance-Anforderungen skalieren.

3.5 Offene Standards und Initiativen

Offene Standards fördern Interoperabilität und Kompatibilität von Systemen. Für Software Defined Computing entstanden neben zahlreichen herstellerspezifischen Implementierungen eine Reihe von offenen Standards und Intiativen am Markt, von denen wichtige Vertreter im Folgenden kurz skizziert werden.

OpenStack als umfassende offene Initiative für SDX umfasst mehrere SDC-Projekte: OpenStack Compute (Codename Nova) provisioniert und managed virtuelle Maschinen; OpenStack Image Services (Codename Glance) verwaltet u. a. Images für Computeumgebungen und OpenStack Bare-Metal Provisioning (Codename Ironic) implementiert Verfahren zur Bereitstellung von Bare-Metal-Systemen. Ein offener Standard der Distributed Management Task Force (DMTF) für virtualisierte Images bzw. virtuelle Appliances ist das Open Virtualization Format (OVF). Im Sommer 2015 haben zahlreiche Hersteller die Open Container Initiative (OCI) gegründet, um offene Industriestandards um Container und Laufzeitumgebungen zu entwickeln.

4 Software Defined Networking

4.1 Allgemeines

Die klassischen Netzwerkgeräte wie Router und Switches enthalten jeweils ihr eigenes Betriebssystem und alle notwendigen Informationen für die Auswahl der Wege und das weiterleiten der Datenpakete, für Layer 2 und Layer 3. Diese Informationen werden auch über entsprechende Protokolle erzeugt und im Netzwerk koordiniert. Die Konfiguration und das Management werden in vielen Fällen über CLI (Command Line Interface) oder einen Element-Manager einzeln für die Switche und Router durchgeführt. Die Control-Plane ist für die Konfiguration und Verwaltung der Weiterleitungsinformationen sowie für das zugehörige Regelwerk zuständig. Die Data-Plane sorgt dabei für das eigentliche Weiterleiten.

Durch Server-Virtualisierung, Cloud, Service-Catalog, Self-Service Portale, etc. haben sich die Anforderungen an Automatisierung, Orchestrierung und Netzwerk-Infrastruktur natürlich deutlich erhöht. Das bisher in vielen Fällen verwendete Betriebsmodell der Konfiguration mittels CLI oder Element-Manager oder auch Scripts ist für diese Anforderungen nicht immer geeignet.

Das war einer der Auslöser für SDN (Software Defined Networking), wenn auch nicht der einzige. In vielen Fällen wurde unter SDN die Trennung von der Controll- und Data-Plane verstanden.

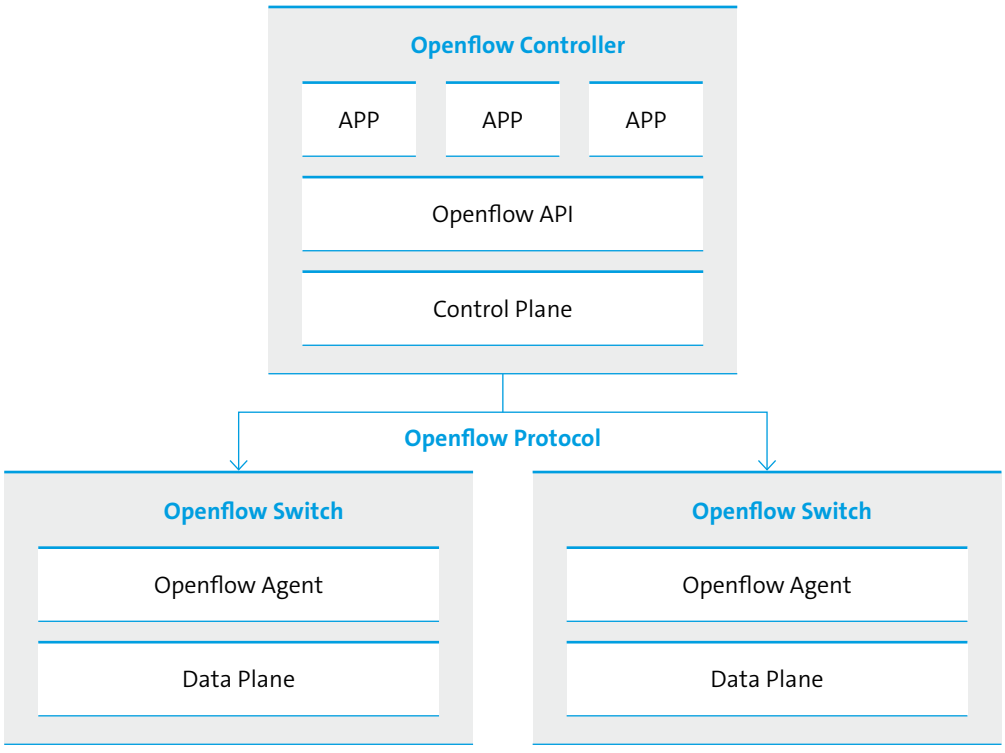

Abbildung 3: Trennung Control- und Data-Plane

Für die Kommunikation zwischen Controller und Infrastruktur wird überwiegend der offene Standard OpenFlow verwendet. Er wurde an der Stanford University in Kalifornien entwickelt und wird inzwischen von der Open Networking Foundation (ONF) überwacht.

Über den Controller und die Schnittstellen nach Norden und Süden kann eine zentrale Provisionierung und dadurch auch eine einfacherer Automatisierung und Orchestrierung erreicht werden. Der Nachteil dieser Lösung ist, dass alles vom zentralen Controller abhängt und sich in diesem Zusammenhang auch Fragen bzgl. der Sicherheit und Skalierbarkeit ergeben.

Nahezu alle Netzwerkhersteller haben solche SDN Controller im Angebot. Es hat sich aber auch ein Herstellerkonsortium gebildet – »OpenDaylight« das einen offenen SDN Controller entwickelt hat und von Kunden genutzt werden kann.

Neben der bereits erwähnten SDN »Underlay« Lösung, welche über einen SDN-Controller die Verkehrsflüsse in den darunter liegenden physikalischen Switchen regelt, stellen die »Overlay« Konzepte eine weitere Möglichkeit zur Netzwerkvirtualisierung zur Verfügung. Ein SDN-Overlay, auch bekannt als Network Virtualization Overlay (NVO), regelt Verkehrsflüsse zwischen virtuellen Switches, welche von der Hypervisor Umgebung direkt auf den Servern bereitgestellt werden. Die nativen Datenpakete der VMs werden auf den Servern gekapselt (»encapsulated«) und durch das physikalische Netzwerk durchgetunnelt. Neue Switches oder Konfigurationsänderungen im darunter liegenden physikalischen Netzwerk sind nicht notwendig. Kommunikationsbeziehungen zwischen virtuellen Maschinen können auf diese Weise sehr einfach implementiert und automatisiert werden.

4.2 Status Quo

Beide oben angesprochenen Konzepte sind nützliche Technologien auf dem Weg zu mehr Effizienz in virtualisierten IT-Umgebungen. Overlays beginnen sich, aufgrund von anderen Anforderungen als im WAN und Campusnetzwerken, im RZ zu etablieren. Wichtige Punkte sind hier die Workload Mobilität, also das Verschieben von virtuellen Maschinen auf verschiedene physikalische Hosts innerhalb eines Rechenzentrums oder auch Rechenzentrumsübergreifend. Virtuelle Maschinen (VM) können in sehr kurzer Zeit zur Verfügung gestellt werden, in vielen Fällen dauert dabei jedoch die Provisionierung der Netzwerkinfrastruktur länger, bis die VM benutzt werden kann. Auch das Bereitstellen von Virtual Private Clouds stellt Herausforderungen an die Provisionierung der Netzwerke. Hier können Overlay-Netzwerke verwendet werden. Dabei handelt es sich gewissermaßen um Netzwerke, die über ein anderes Netzwerk gelegt werden. Die Basis ist die physikalische Netzwerkinfrastruktur oder auch Underlay-Netzwerk genannt. Darauf können dann die Overlay-Netzwerke abgebildet werden. Dafür wurde der VXLAN (Virtual Extensible LAN) Standard entwickelt. VXLAN bietet die höhere Skalierbarkeit der Infrastruktur sowie eine Möglichkeit zur Erstellung isolierter, mandantenfähiger Broadcast-Domänen innerhalb des Rechenzentrumsnetzwerkes oder Underlay-Netzwerk. Dazu erstellt es logische Layer-2-Netzwerke, die in Layer-3-Standard-IP-Paketen gekapselt werden.

Ein weiterer Vorteil von VXLAN ist die Aufhebung der VLAN-Begrenzung von 4096 VLANs, das kann bei grösseren Umgebungen ein durchaus limitierender Faktor sein. VXLAN basierte Overlay-Netzwerke können auch über einen zentralen Controller provisioniert werden. Es exisitieren rein Software basierte Lösungen, was bedeutet, dass die Overlay-Netzwerke auf den Hypervisor aufgesetzt werden und das Underlay-Netzwerk dabei nur noch das IP-Forwarding durchführt.

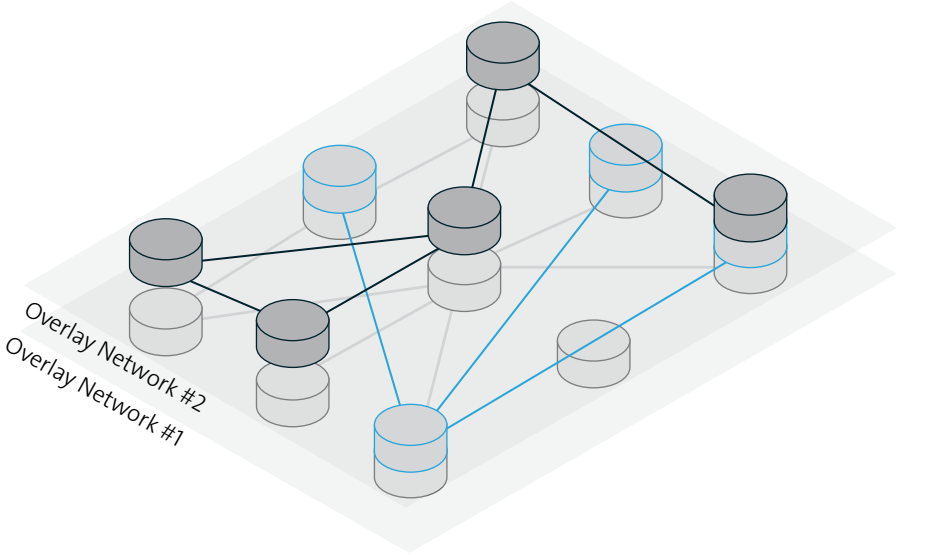

Abbildung 4: Overlay Netzwerke

Punkte auf die zu achten sind:

- Da es neben der virtualisierten Umgebung in den meisten Fällen auch noch nicht virtualisierte Systeme gibt, muss für ein Gateway zwischen dem/den VXLAN-Overlay-Netzwerken und den »normalen« VLAN-basierten Netzen implementiert werden. Dies wird häufig auf x86-Servern implementiert und kann ein potentieller Engpass sein. Auch ist bei Gateways immer auf die möglichen Redundanzen zu achten

- Bei den rein Software basierten Lösungen fehlt immer die Interaktion mit dem Underlay Netzwerk. Das erschwert das Management und auch die Fehlersuche.

- Es müssen zwei Netzwerke betrieben werden, das Overlay und das Underlay.

Neben den rein Software basierten Lösungen gibt es auch Hardware gestützte Lösungen. Dabei wird das VXLAN Overlay auf den Switchen aufgesetzt, in denen auch die Gateway Funktion enthalten ist. Dieser Ansatz adressiert zumindest zum Teil die oben angesprochenen Punkte.

Eine dritte Option ist ein Verzahnung von der rein software- und hardwarebasierten Lösung. Das bietet eine nahtlose Integration der Virtualisierten und physikalischen Server-Infrastrukur und die Kombination der Flexibilität von Software und der Performance, Skalierbarkeit und Sicherheit einer Hardware basierten Lösung.

Unabhängig von den Vor- und Nachteilen der verschiedenen Optionen – alle bieten Lösungen für Automation und Orchestrierung, sollte eine genaue Analyse der Anforderungen im Vordergrund stehen. Es kann auch durchaus eine Kombination von verschiedenen Ansätzen sinnvoll sein. Wichtig ist das der Controller über eine offene Schnittstelle nach »North« besitzt damit eine Anbindung in übergreifende Automatisierungs- und Orchestrierungsanwendungen erfolgen kann.

Neben den reinen Netzwerkfunktionen muss auch ein Augenmerk auf die Netzwerkservices wie Firewall und Loadbalancing gelegt werden. Nur mit diesen Funktionen können umfassende Cloud Lösungen bereitgestellt werden.

4.3 Software Defined Networking Glossar

Abstraktion:
Im Zusammenhang mit SDN wird Abstraktion verwendet, um mit einer Software-Ebene zwischen dem Controller und den Netzwerkelementen die Steuerungsfunktionen von der Weiterleitung der Pakete zu trennen.

Control-Plane:
Ist die zentrale Steuerungsfunktion in einem Switch oder Router. Darüber wird die Konfiguration der Switche/Router gesteuert und die »Forwarding«-Tabellen für das Weiterleiten der Pakete programmiert. Bei SDN kann diese Funktion ausgelagert werden, so dass der Switch/Router lokal nur noch das weiterleiten der Pakete durchführt.

Data-Plane:
Ist bei Switchen und Routern für das Weiterleiten der Pakete verantwortlich und wird von der »Control-Plane« gesteuert.

Flow-Management:
Ist das Steuern der einzelnen Datenströme in einem Netzwerkelement (Switch/Router). Von der ONF (Open Network Foundation) wurde dazu Openflow entwickelt, das den Zugriff auf die Forwarding Funktion in einem Switch oder Router regelt.

Netzwerk-Virtualisierung:
Darunter versteht man die Möglichkeit, ein Netzwerk in logisch aufzuteilen. Es wird also das physikalische Netzwerk in verschiedene unabhängige Netze unterteilt. Es wird dazu auch der Begriff Overlay Netzwerk verwendet.

Network-Function-Virtualization (NFV):
Ist das Virtualisieren von Netzwerkservice-Funktionen wie z. B. Firewalls oder Loadbalancer. Es werden also keine physikalischen Appliances mehr verwendet. Der Loadbalancer oder die Firewall sind dann virtuelle Maschinen, die auf einem Hypervisor betrieben werden.

OpenDaylight:
Ist ein Herstellerkonsortium mit dem Ziel, einen offenen SDN-Controller zu entwicklen, um damit eine standardbasierte Controller-Lösung Kunden zu Verfügung zu stellen. Die verschiedenen Hersteller tragen unterschiedliche Funktionen zum OpenDaylight Controller bei. Siehe auch: http://www.opendaylight.org/

Openflow:
Ein von der ONF (Open Network Foundation) entwickeltes Protokoll für die Kommunikation bzw. Programmierung der Forwarding-Tabellen in einem Switch oder Router, das auch Funktionen für das Traffic-Management enthalten kann.

SDN – Software defined Networking:
In vielen Fällen wird damit die Trennung der Control-Plane und Data-Plane-Funktionen in einem Netzwerkelement gemeint. Es wird die Steuerungsfunktion (Control-Plane) von der Forwarding-Funktion (Data Plane) getrennt. Befindet sich also nicht mehr gemeinsam in einem System (Switch/Router). Die Steuerungsfunktion wird von einem zentralen Controller übernommen.

5 Software Defined Storage

5.1 SDS versus klassische Speichersysteme

Ein klassisches Controller-basierendes Speichersystem besteht aus spezialisierter Hardware und einer darauf laufenden proprietären Software (Betriebssystem oder Firmware). Die Aufgaben (Storage Services) dieser Software gliedern sich in drei wesentliche Bestandteile:

- Daten-/Kontroll-Pfad: Zugriffsprotokolle (Beispiel: iSCSI, SMB)
- Daten-Management (Beispiel: Pooling, Deduplizierung, Cloning)
- Daten-Ablage (Beispiel: RAID-Schutz, Striping)

SDS abstrahiert diese Storage Services und Funktionen von der untergelagerten Speicher-Hardware. Die Hardware dient lediglich als persistente Ablageschicht für die Daten – ohne deren eigene Storage Services zu nutzen (soweit überhaupt vorhanden). Alle Services werden in Software abgebildet. SDS ist jedoch keine reine Software-Lösung, vielmehr handelt es sich um eine softwarezentrische Lösung die typischerweise auf »Commodity Hardware« läuft. Je nach Implementierung nutzt SDS dieselbe Hardware wie die anderen Komponenten der SDDC oder es werden dedizierte Systeme verwendet.

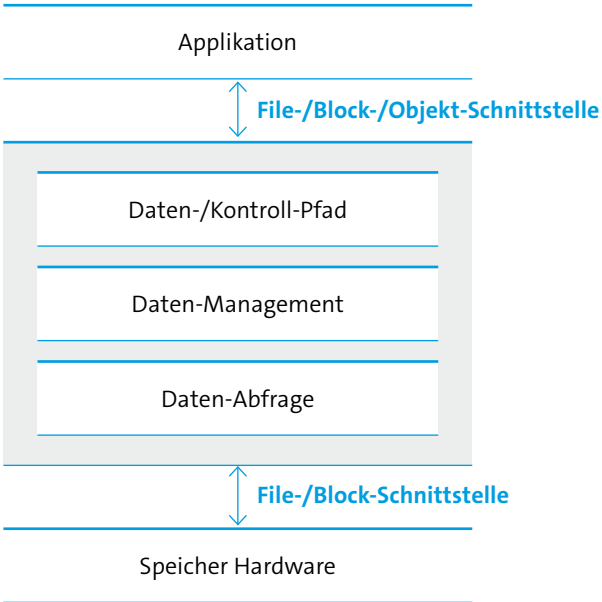

Abbildung 5: Speichermodell Abstraktionslevel

Ein wesentliches Merkmal einer SDS-Lösung ist der starke Applikationsbezug. SDS stellt Speicher-Pools mit gewissen Qualitätsmerkmalen zur Verfügung. Applikationen bedienen sich aus diesen Pools je nach Anforderung. Die Provisionierung ist eng mit der Applikation verknüpft. Ein hohes Maß an Flexibilität soll sicherstellen, dass die Daten innerhalb ihres Lebenszyklus immer adäquat abgelegt werden. Der SDS-Ansatz verwendet im Kern Storage-Virtualisierungs-Techniken, ist aber weit mehr als eine reine Storage-Virtualisierungs-Lösung.

5.2 Anforderungskriterien an eine SDS Lösung

Eine SDS-Implementierung muss eine Reihe von Basisanforderungen erfüllen (siehe auch (↗SNIA Whitepaper vom April 2014))

1. Automation: Einfache Bereitstellung/Provisionierung der Storage-Infrastruktur
2. Standard-Schnittstellen: APIs für das Management, die Provisionierung und Wartung/Monitoring der Storage Services
3. Zugriffs-Pfade: File-, Block- und Objekt-Schnittstellen für den Zugriff auf die Daten.
4. Skalierbarkeit: Das System muß ohne Beeinträchtigung erweiterbar sein
5. Mobilität/Flexibilität: Das System muß es erlauben Daten transparent zwischen Leistungsklassen zu verschieben, um auf geänderte Applikationsanforderungen flexibel reagieren zu können.

5.3 Ausprägungsvarianten

Software-Defined-Storage wird heute in unterschiedlichen Ausprägungsvarianten implementiert und angeboten. Man kann diese Lösungen in drei grundsätzliche Ansätze unterteilen:

Hypervisor-Implementierung:
SDS läuft vollständig als Virtuelle Storage Appliance unter der Kontroller eines Hypervisors. Für die Zugriffspfade wird die Netzwerk-Infrastruktur des Hypervisors genutzt – damit sind die Zugriffsprotokolle typischerweise auf IP-basierende Protokolle beschränkt. Die Ablage der Daten erfolgt auf virtuellen Plattenlaufwerken, die der Hypervisor zur Verfügung stellt.

Storage Appliance:
Basierend auf Standard-Server-Hardware und Software (oftmals OpenSource) werden Storage Services angeboten. Die Ablage der Daten erfolgt auf lokal angeschlossen Speichermedien (in der Regel Festplatten). Je nach Einsatzgebiet werden Datei-, Block- oder objektbasierende Protokolle implementiert. Der Zugriff erfolgt über die Netzwerkschnittstellen der Server-Hardware – damit ist eine Vielzahl unterschiedlicher Standardprotokolle möglich.

Hybrid:
Auf einem proprietären Speichersystem werden einzelne virtuelle Instanzen definiert. Eine solche Instanz wirkt nach außen wie ein eigenständiges Speichersystem mit eigenen Ressourcen und einer eigenen Identität. Jede Instanz kann potentiell alle Data-Management-Funktionen und Zugriffs-Protokolle zur Verfügung stellen. Die Betriebssoftware des Speichersystems agiert wie ein Hypervisor. Eine solche Instanz bedient in der Regel eine bestimmte Applikation.

Da diese Technologie noch sehr neu ist und stark auf Funktionen in den Hardware-Komponenten zurückgreift, ist die Einhaltung der jeweiligen Hardware Compatibility List (HCL) zwingend erforderlich.

6 Orchestrierung und Automatisierung

6.1 Einführung

SDx Plattformen und Technologien sind kein Selbstzweck oder eine interessante Modeerscheinung. Vielmehr bilden sie die Basis um alle SDx-Teildisziplinen in ein übergreifendes Managementsystem einzubinden und damit erst Automatisierung im großen Stil zu ermöglichen. Um die Komponenten einer SDx-Umgebung zusammenzuführen und letztlich die Integration von Technik und Prozessen in einer wiederholbaren, automatisierten Form zu ermöglichen, werden sogenannte Orchestrierungs- und Automatisierungsplattformen eingesetzt.

Vereinfacht gesagt: SDx wird nicht simpler, nur weil alles In Software statt in Silizium gegossen wurde. Die Komplexität der SDx-Lösungen im Einzelnen muss durch eine durchgängige Management- und Automatisierungslösung aufgefangen werden. Einzelne Inseln nur zu automatisieren, macht wenig Sinn.

6.2 Funktion

Für die einzelnen Teildisziplinen existieren viele verschiedene Managementsysteme, welche sich um die Verwaltung der Komponenten im Detail kümmern. Hierbei kommen häufig Herstellerspezifische Programme und Plattformen zum Einsatz, übergreifende Produkte sind kaum zu finden. Dies ist unter anderem in der spezifischen Ausprägung der APIs und Schnittstellen begründet, für die überwiegend keine detaillierten Standards existieren. Das nachfolgende Schaubild zeigt exemplarisch die Integration der Komponenten in das Gesamtsystem.

Alle Komponenten der Teildisziplinen müssen APIs zur Verfügung stellen um in Orchestrierungs und Automatisierungs-Frameworks integriert werden zu können. Die APIs teilen sich u. a. auf in die Bereiche:

- Provisionierung
- Steuerung (z. B. Snapshots/Clones erzeugen)
- Überwachung (Kapazität, Effizienz, Performance)
- De-Komissionierung

Die meisten APIs sind dabei als REST-APIs ausgeführt, um die Interaktion mit weiteren Managementsystem zu vereinfachen.

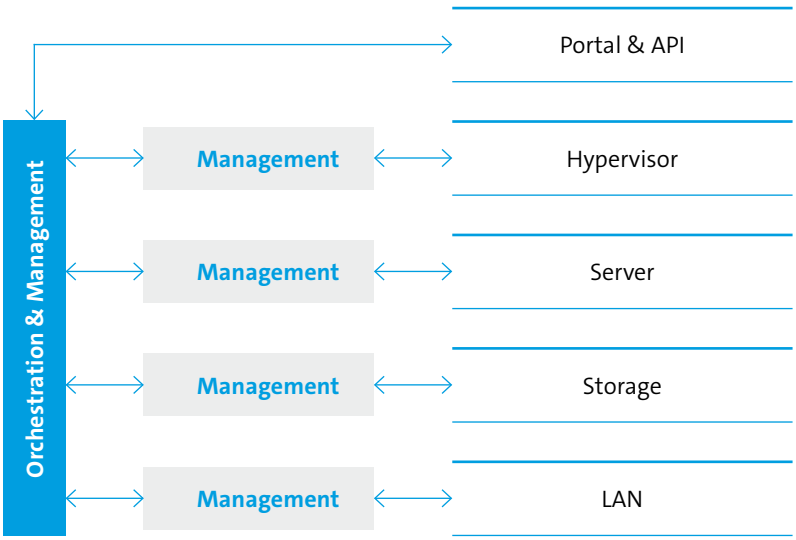

Abbildung 6: Orchestration & Management

Für die Orchestrierungs- und Automationsschicht haben sich mittlerweile einige Quasi-Standards entwickelt, die allerdings bislang keiner offiziellen Normierung unterliegen. Hierzu zählen z. B. OpenStack, bekannt als Cloud-Management-Plattform, oder auch Puppet als Automationsplattform. Wo es jedoch offene (Industrie-)Standards gibt, sollten diese adaptiert werden.

Dies gilt insbesondere für die Frameworks, die eine offene Plugin/Treiber-Architektur verfolgen. Hier liefert der Hersteller der Komponente einen vollständigen Adapter – das Framework selber muss nicht angepasst werden (Beispiel: OpenStack Cinder/Manila/Swift Architektur) – damit bleibt das Framework herstellerunabhängig und die Verantwortung für die Integration liegt beim Hersteller der Komponente.

6.3 Voraussetzungen

Basis für die Umsetzung von automatischen Abläufen und die Bereitstellung von Systemen in Self-Service-Katalogen ist die Definition von Standards und Servicelevel Agreements (SLA). Erstaunlich viele Unternehmen beschäftigen sich bereits mit der technischen Auswahl von Automationsprodukten, ohne sich vorab Gedanken über die dafür notwendigen Standards ihrer Systeme gemacht zu haben:

- Wie sieht z. B. ein Standard-Fileserver aus? (CPUs, Speicher, RAM, NICs, Plattengrößen, OS Konfiguration, Standardsoftware, Backupprozeduren, etc.)
- Wie sieht der Prozess von Beantragung bis Betriebsübergang aus? (Wer darf was beantragen oder genehmigen? Wie werden erforderliche Kapazitäten sichergestellt? Wie erfolgt das Lizenzmanagement? etc ...)
- Wie werden die Ressourcen abgerechnet? (Interne Leistungsverrechnung, projektbezogene Verrechnung, Flat Fee, monatliche Kosten, pay as you use, etc?)
- Wie schnell muss das System verfügbar sein? Wie definiert sich die Verfügbarkeit für das spezifische System? Wie wird es überwacht und was passiert im Fehlerfall und durch wen? etc ...

Die Zusammenhänge zwischen verschiedenen Disziplinen werden durch Policies definiert. So müssen z. B. zur Bereitstellung eines neuen Servers auch die entsprechenden Firewalls und VLANs konfiguriert, Backup und Monitoring eingebunden und die CMDB aktualisiert werden, ehe das System produktiv genutzt werden kann. Auch hier müssen vorab die Abläufe und Standardprozeduren definiert werden ehe diese in einem System technisch abgebildet werden können.

6.4 Zusammenfassung

Technische Unterschiede im Detail gibt es derzeit noch sehr viele zwischen den verschiedenen Produkten. Welche davon jedoch für eine konkrete Umgebung relevant sind, und welche Produkte und Plattformen dann optimaler Weise gewählt werden sollten, hängt entscheidend an der Beschreibung der zugrunde gelegten Ziele und der abzubildenden Prozesse.

Erst wenn klar ist, welches Ergebnis erreicht werden soll, ist im nachfolgenden Schritt die Auswahl einer passenden technischen Plattform möglich, die dann mit den verschiedenen Managementkomponenten der SDx-Teildisziplinen in optimaler Weise zusammenspielt.

7 Nutzen

Die verschiedenen technologischen Komponenten des Software Defined Environments decken eine Vielzahl der geschäftsbezogenen Anforderungen ab, die an eine Unternehmens-IT gestellt werden. Gemeinsam bilden sie ein einheitliches Framework, das die Unternehmen für die Zukunft rüstet. In diesem Kapitel wird die Nutzung dieses Ansatzes zusammengefasst, wobei die Vorteile nicht sehr scharf voneinander getrennt werden können.

Flexibilität

Voraussetzung einer softwaredefinierten Umgebung ist, dass die Systeme es erlauben, Ressourcen transparent zwischen Leistungsklassen zu verschieben, um auf geänderte Applikationsanforderungen flexibel reagieren zu können. Die Bereitstellung dieser Ressourcen ist unabhängig vom Ort und von der Hardware.

Hardwareeingriffe in einem System sind zeit- bzw. kostenaufwändiger als Softwareänderungen. Eine Software Defined Umgebung ermöglicht eine schnelle Markteinführung von Produkten und Dienstleistungen.

Vereinfachung

Eine Trennung zwischen der logischen und der physikalischen Schicht erlaubt das Vornehmen von Änderungen in der einen Ebene, ohne die Andere zu beeinflussen. Eine Vereinfachung erfolgt auch durch die Multiprotokollfähigkeit und Konvergenz, die wesentliche Elemente eines Software-Defined-Konzepts sind.

Richtlinienbasierte Konfiguration und Bereitstellung der notwendigen Ressourcen sind zentrale Elemente eines SDEs, die, verglichen mit klassischen Methoden, Zeit und Kosten reduzieren. An dieser Stelle soll auch die Reduzierung der potentiellen Fehlerquellen durch menschliche Eingriffe erwähnt werden.

In einem SDDC steht die Service-Orientierung im Mittelpunkt. Alle Infrastrukturkomponenten sind virtualisiert, werden als Service bereitgestellt und die Eingriffe sowie das Management sind mittels Software automatisiert. Anwendungen bekommen den passenden Level an Leistung und Verfügbarkeit. Die bedarfsorientierte Bereitstellung der Infrastruktur beschleunigt somit die Einführung von neuen Geschäftsfeldern im Unternehmen mit klar definierten und für den jeweiligen Anwendungsfall verfügbaren Service Level.

Wirtschaftlichkeit

Weniger Aufwand für Konfiguration, Provisionierung und Management, eine höhere Reaktionsgeschwindigkeit sowie bessere Nutzung von Ressourcen und der damit verbundene Energieverbrauch tragen dazu bei, dass Kosten reduziert werden. In vielen Fällen können Unternehmen die vorhandene Infrastruktur nutzen, dadurch ihre Investitionen schützen und effizient gegenüber den sich ändernden Anforderungen aufstellen.

Virtualisierung ist ein wesentliches Instrument für die Realisierung einer Softwaredefinierten Umgebung. Das Rechenzentrum wird durch diesen Ansatz unabhängig von spezialisierter Hardware und deren Konfiguration bzw. Programmierung.

Standard-Schnittstellen und Protokolle (Bsp. OpenStack und RESTful APIs, etc.) sorgen dafür, dass das Management, die Bereitstellung von Ressourcen und das Monitoring gewährleistet sind.

Der gesamte Ansatz ist so konzipiert, dass künftige Funktionalitäten schnell, einfach und transparent implementiert werden können, ohne die Hardware direkt zu beeinflussen.

8 Ausblick

Software Defined Environments sind die Grundlage für Cloud-Computing im Bereich Infrastructure-as-a-Service. Insbesondere für die Nutzung von Hybrid-Cloud-Szenarien, bei denen Unternehmens-IT mit öffentlichen oder privaten Cloud-Angeboten von Cloud-Anbietern oder Outsourcing-/Outtasking-Anbietern kombiniert wird, werden Software Defined Environments in der Zukunft eine immer wichtigere Rolle spielen.

Bitkom vertritt mehr als 2.300 Unternehmen der digitalen Wirtschaft, davon gut 1.500 Direktmitglieder. Sie erzielen mit 700.000 Beschäftigten jährlich Inlandsumsätze von 140 Milliarden Euro und stehen für Exporte von weiteren 50 Milliarden Euro. Zu den Mitgliedern zählen 1.000 Mittelständler, 300 Start-ups und nahezu alle Global Player. Sie bieten Software, IT-Services, Telekommunikations- oder Internetdienste an, stellen Hardware oder Consumer Electronics her, sind im Bereich der digitalen Medien oder der Netzwirtschaft tätig oder in anderer Weise Teil der digitalen Wirtschaft. 78 Prozent der Unternehmen haben ihren Hauptsitz in Deutschland, 9 Prozent kommen aus Europa, 9 Prozent aus den USA und 4 Prozent aus anderen Regionen. Bitkom setzt sich insbesondere für eine innovative Wirtschaftspolitik, eine Modernisierung des Bildungssystems und eine zukunftsorientierte Netzpolitik ein.

Bundesverband Informationswirtschaft,
Telekommunikation und neue Medien e.V.

Albrechtstraße 10
10117 Berlin
T 030 27576-0
F 030 27576-400
bitkom@bitkom.org
www.bitkom.org

bitkom