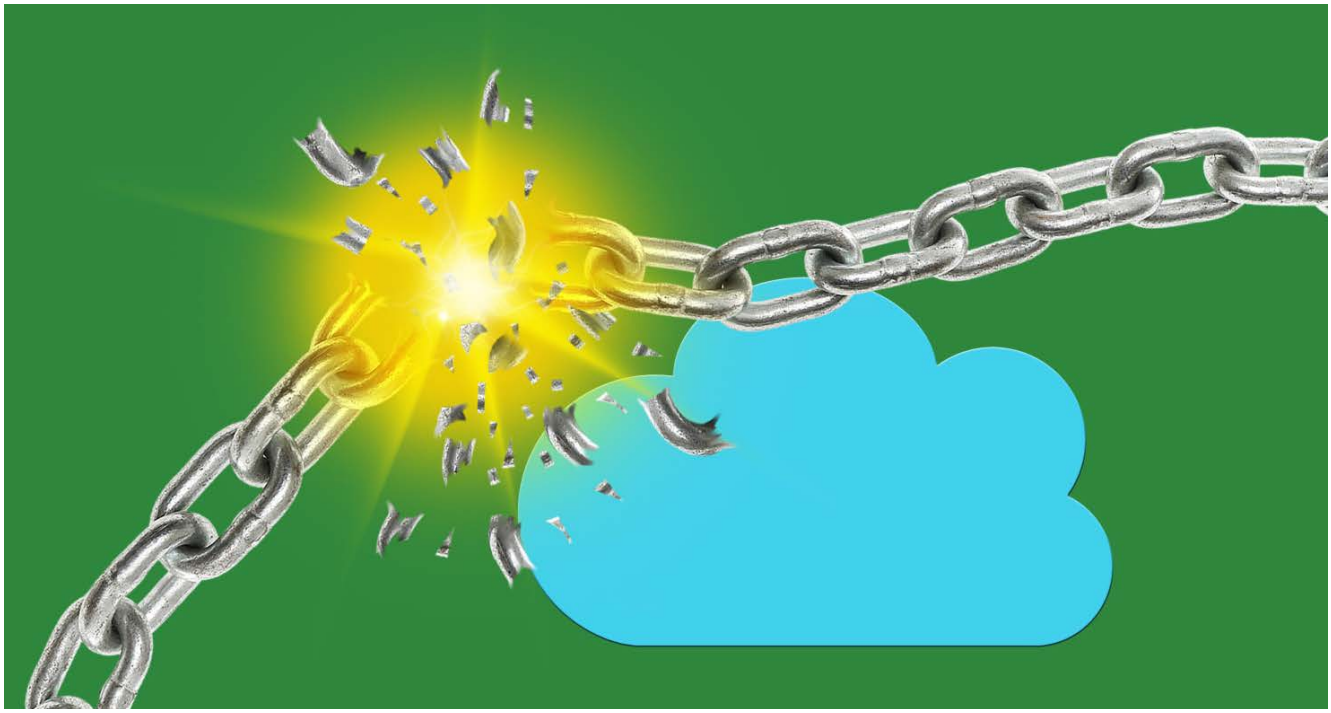


Mit Sovereign Cloud Stack zu mehr digitaler Souveränität

Wolken-Verbund

Kurt Garloff

Das Sovereign-Cloud-Stack-Projekt vereinfacht Bereitstellung und Betrieb standardisierter Cloud- und Containerinfrastrukturen mit einer komplett offenen Softwareplattform. Dieser Artikel erläutert die Grundlagen und leitet durch eine einfache Testinstallation.

Corona machts möglich: Im Frühjahr 2020 rückte die Pandemie die Digitalisierung in den Vordergrund. In der Bildung, in der öffentlichen Verwaltung und in vielen Betrieben waren digitale Arbeitsmittel nicht ausreichend etabliert oder die vorhandenen nicht auf das geforderte sprunghafte Wachstum vorbereitet. Aber viele Unternehmen nutzten die Zeit seit dem ersten Schock und sind heute besser aufgestellt. Allerorten hat man zum Beispiel Videokonferenzen implementiert. Wo die aus cloudbasierter Infrastruktur geliefert werden, ist auch ein Skalieren, also die dynamische Anpassung an stark schwankende Bedürfnisse, Standard.

Der Preis dafür ist in vielen Fällen aber eine noch stärkere Abhängigkeit von großen Plattformanbietern. Von denen gibt es nur wenige und sie operieren auch noch in Rechtsräumen, die kulturell und rechtlich nicht den Erfordernissen des europäischen Datenschutzes genügen. Das EuGH-Urteil zum Thema Privacy Shield (siehe *iX* 9/2020) hat das vielen Unternehmen noch einmal schmerzlich klargemacht. Zudem sind starke Abhängigkeiten auch aus wirtschaftlicher und strategischer Sicht wenig wünschenswert.

Die Zukunft des Abendlandes

Daten und ihre Verarbeitung werden in der Industrie mehr und mehr die Basis jedes Fortschritts. Die Wettbewerbsfähigkeit ist in immer mehr Bereichen davon abhängig, wer besonders gut in der Entwicklung innovativer Software zur Verarbeitung der Daten ist und wer Zugriff und Kontrolle über die Daten hat. Gerade auch für die mittelständisch geprägte Wirtschaft in Deutschland wird sich im kommenden Jahrzehnt entscheiden, ob sie die Kontrolle über ihr Daten behalten kann. Auch die Kompetenz zur datenbasierten Innovation droht verloren zu gehen und damit geriete letztlich auch die Fähigkeit zur Innovation in fremde Hände.

Im Privaten steht nicht weniger auf dem Spiel: Der Datenschutz ist letztlich die Freiheit, nicht bei jeder Handlung der Überwachung durch Daten sammelnde Systeme unterworfen zu sein und somit die Frage stellen zu müssen, welche Informationen über die eigene Person die Datensammler anhäufen. Das westeuropäische System mit der Freiheit des Handelns im Privaten und die Innovationsfähigkeit des Wirtschaftssystems müssen sich gegen andere Entwürfe behaupten – dem ungebremsten Datensammeln der großen

Agile Infrastruktur

Exzellenz in der Softwareentwicklung ist für die Wettbewerbsfähigkeit der Industrie unabdingbar. Die Vorgehensweise in der Softwareentwicklung hat sich dabei in den letzten zwei Jahrzehnten zu agilen Methoden verschoben, die in Kombination mit der Automatisierung von Test- und Betriebsprozessen einen deutlichen Produktivitätssprung ermöglicht haben.

Modern arbeitende Teams schieben dabei die riskanten Integrationsschritte der verschiedenen Komponenten nicht ans Ende des Entwicklungszyklus – vielmehr wird dieser von Anfang an implementiert und kontinuierlich (CI = Continuous Integration) vorangetrieben. Die Integration erfolgt dabei auf einer Infrastruktur, die sich komplett durch Software aufbauen und steuern lässt (Infrastructure as Code oder auch agile Infrastruktur).

Die Integration der Komponenten auf eine vollautomatisch bereitgestellte – und anschließend wieder aufgeräumte – virtuelle Infrastruktur wird dabei mehrmals täglich mit der gleichen Automatisierung durchgeführt wie die (seltener erfolgende) Installation in die Produktionssysteme (CD = Continuous Delivery). Wer hier eine gute Testabdeckung umsetzt, kann gleichzeitig eine höhere Geschwindigkeit und Flexibilität bei geringerem Risiko erzielen.

Solches „Mit"-Entwickeln der Installation auf automatisiert bereitgestellter Infrastruktur verwischt die Grenzen zwischen Entwicklungs- und Betriebsaufgaben, was eine enge Zusammenarbeit von Entwicklern und Admins bis hin zur Auflösung der Teamgrenzen nahelegt. Das ist der Gedanke hinter DevOps. Auch IT-Sicherheitsprozesse lassen sich in diese Strukturen einbinden, zum Beispiel durch Sicherheitstests als Teil der CI. Auf diese Weise bilden sich DevSecOps-Teams.

Plattformanbieter (mit kaum kontrollierbarem Zugriff durch Geheimdienste) wie in den USA einerseits und dem staatlichen Überwachungsmodell Chinas andererseits. Ohne die Kontrolle über die eigenen Daten und die zugehörigen Systeme zur Datenverarbeitung wird das kaum zu erreichen sein. Dies ist der Kern der Diskussion über digitale Souveränität.

Auf dem Digital-Gipfel des BMWi wurde digitale Souveränität als „die Fähigkeit zum selbstbestimmten Handeln und Entscheiden im digitalen Raum definiert. Sie umfasst zwingend die vollständige Kontrolle über gespeicherte und verarbeitete Daten sowie die unabhängige Entscheidung darüber, wer darauf zugreifen darf. Diese umfasst auch die Fähigkeit, technologische Komponenten und Systeme eigenständig zu entwickeln, zu verändern, zu kontrollieren und durch andere Komponenten zu ergänzen."

GAIA-X und Sovereign Cloud Stack

Das vom Ministerium initiierte und mittlerweile von Unternehmen, Forschungseinrichtungen und Institutionen aus Deutschland, Frankreich und anderen EU-Ländern getragene Projekt GAIA-X (siehe Artikel auf Seite 45) hat das Ziel, mittels Transparenz und Standards die Souveränität über die Daten zu bewahren oder überhaupt zu erlangen. Eine sichere und vernetzte Dateninfrastruktur soll den höchsten Ansprüchen an digitale Souveränität genügen und Innovationen fördern. In einem offenen und transparenten digitalen Ökosystem sollen Daten und Dienste verfügbar gemacht, zusammengeführt und vertrauensvoll geteilt werden können. Mit entsprechenden Standards wird es möglich, dass Daten und datenbasierte Anwendungen nicht mehr von einem einzelnen Anbieter abhängig sind. Vielmehr sollen Kunden den Anbieter wechseln und auch Anwendungen aus Bausteinen verschiedener Anbieter zusammensetzen können.

Aber Interoperabilität und Portabilität alleine sind noch kein Garant dafür, dass ein lebendiges Ökosystem von kompatibler, moderner, agiler Infrastruktur in Europa entsteht (siehe Kasten „Agile Infrastruktur"). Zwar gibt es viele kleine und mittelgroße Anbieter sowie interne Cloud-Umgebungen in Unternehmen, die auf Open-Source-Software, typischerweise OpenStack und Kubernetes, oder proprietärer Software – häufig VMware – aufbauen. Jedoch sind diese Umgebungen nicht miteinander vernetzt und aufgrund der endlosen Anpassungsmöglichkeiten auch kaum zueinander kompatibel, der Markt ist stark fragmentiert.

X-TRACT

- Der Markt für Cloud-Angebote jenseits der großen Plattformen aus den USA und China ist groß, aber zersplittert, und kann selten eine Alternative bieten.
- Der Betrieb dynamischer verteilter Systeme wie einer Cloud-Umgebung ist gerade für kleinere Anbieter und IT-Abteilungen nur schwer zu meistern.
- Das Projekt Sovereign Cloud Stack in GAIA-X will durch Vernetzung der Anbieter eine offene, standardisierte Plattform mit dokumentierten Betriebsprozessen schaffen und so eine souveräne, föderierte und interoperable Infrastruktur bereitstellen.

Einen Standard setzen

Für Nutzer, die nach einem Ausweg aus der starken Abhängigkeit von einem der großen Cloud-Anbieter suchen, ist die Alternative – eine ebenso starke Abhängigkeit von einem kleinen europäischen Anbieter – kein offensichtlicher Fortschritt. Kein Admin sollte sich dabei aber der Illusion hingeben, der Betrieb einer hochdynamischen verteilten Umgebung wie einer Cloud-Infrastruktur sei einfach. Gerade kleinere Anbieter tun sich schwer, ein entsprechend qualifiziertes Betriebsteam aufzubauen und die richtige Kultur und Prozesse zu entwickeln, um eine den großen Anbietern vergleichbare Qualität liefern zu können.

Sovereign Cloud Stack (SCS) will das Entstehen eines lebendigen Ökosystems aus vernetzten, föderierbaren Infrastrukturanbietern fördern, die gemeinsam in einem offenen Prozess die Standards, die freie Software ebenso wie die Prozesse und Werkzeuge für den Betrieb der Cloud-Plattformen entwickeln, einem Communityprojekt von GAIA-X. Eine Föderation besteht dabei aus voneinander unabhängigen Netzen, die beispielsweise „nur" eine gemeinsame Authentifizierungsmethode benutzen. Die Infrastrukturanbieter umfassen dabei nicht nur klassische Anbieter öffentlicher Clouds, sondern können ebenso Rechenzentren für die öffentliche Hand oder unternehmensinterne IT-Abteilungen sein.

Aber auch Anwender mit sehr eingeschränkten Nutzergruppen, denen die

Nutzerföderierung nicht nutzt, sollen davon profitieren, eine Standardlösung einzusetzen.

Standards, Code und Ökosystem

Das Sovereign-Cloud-Stack-Projekt (SCS) will dreierlei liefern: Standards, Software und den Aufbau eines Ökosystems. Die Zersplitterung der verschiedenen kleinen Cloud-Plattformen hat sich für die Akzeptanz am Markt als fatal erwiesen. Es fehlt eine Standardisierung, die Softwareentwicklern, Nutzern und letztlich auch Betreibern gegenüber sicherstellt, dass Software, Automatisierung, Wissen, Erfahrung und Testergebnisse ohne nennenswerte Anpassung und doppelte Arbeit übertragbar sind. Diese Standards zu schaffen, ist ein wichtiges Ziel von SCS.

Dabei existieren gerade im Bereich der von SCS eingesetzten Technologien viele Standards. SCS will und wird diese nicht neu erfinden oder gar ersetzen, sondern die am besten passenden auswählen, kombinieren und ergänzen, wo es Lücken gibt. Als Beispiel sei OpenStack erwähnt, dessen Trademark-Zertifizierung (DefCore) allerdings sehr viel Interpretationsspielraum lässt. So ist die Bedeutung von Verfügbarkeitszonen (VZ) dort nicht genau spezifiziert: Ist Block Storage übergreifend oder per VZ definiert? Die gleiche Frage stellt sich rund um den Begriff Netze. Was muss ein Entwickler beachten, der eine Anwendung so konzipieren will, dass sie den Tod einer VZ überlebt?

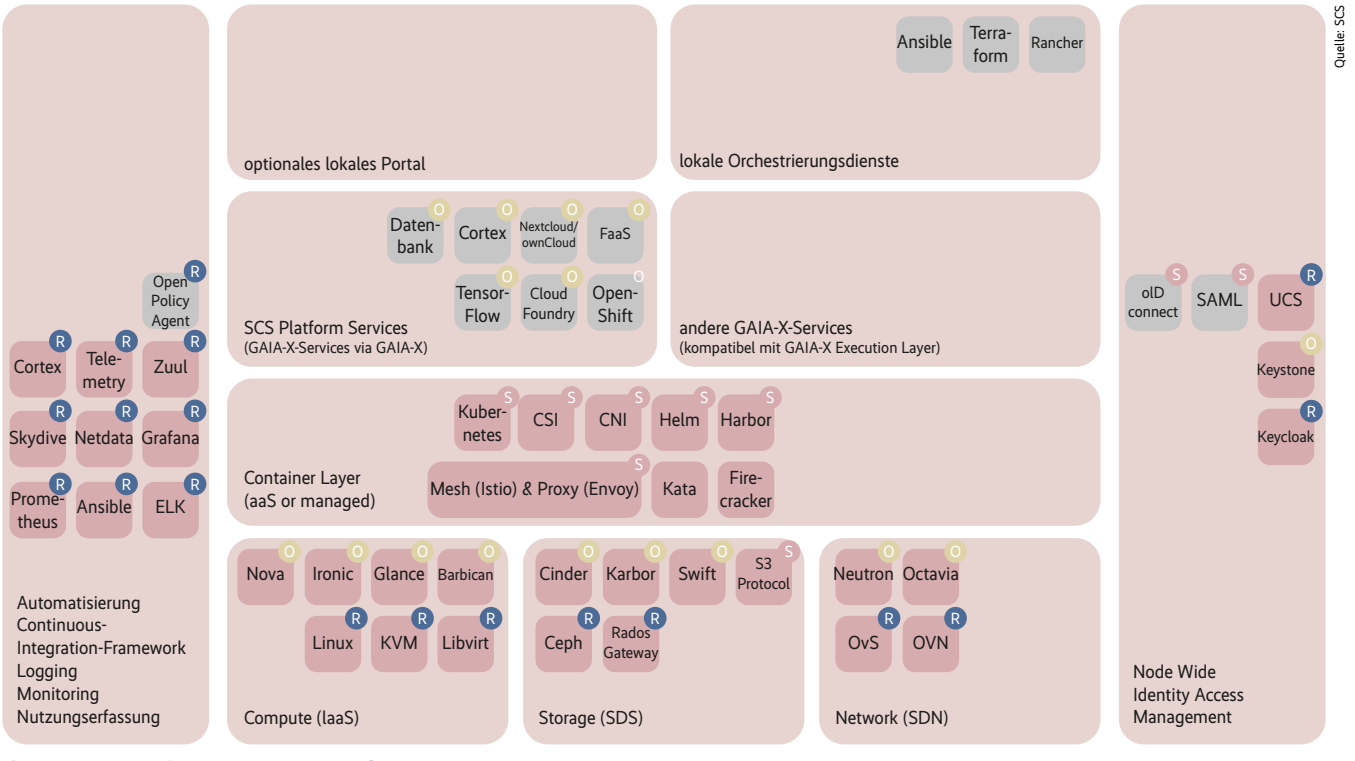
Auch in der Containerwelt finden sich ähnliche Unklarheiten: Ist die Containerorchestrierung durch Kubernetes noch genau definiert, so fehlt es doch an Standards zur Verwaltung der Kubernetes-Cluster. Die Kubernetes Cluster API wird nicht von vielen Anwendern eingesetzt, sie gilt als unzureichend und (noch) nicht ausgereift. Und wie sind die SLAs zu interpretieren?

GAIA-X-Komponenten

Ob die Standards eingehalten werden, muss durch automatisierte Tests überprüfbar sein. Nutzern und Softwareentwicklern gegenüber wird das dann durch entsprechende Zertifizierung zugesichert. Neben für alle verpflichtenden Standards (vor allem im Containerbereich) wird es auch optionale Standards geben – hier können Anbieter entscheiden, ob sie entsprechende Dienste und Schnittstellen anbieten wollen. Wenn sie es tun, ist es aber zertifizierbar kompatibel machbar, was vor allem für die Schnittstellen auf IaaS-Ebene wie auch für Plattformdienste essenziell ist. Die Standards werden behutsam und rückwärtskompatibel weiterentwickelt, was Investitionen in Infrastruktur, Software und Wissen schützt.

Zweitens liefert SCS eine komplette modulare, offene Softwareumgebung. Eine lange Liste an SCS-Komponenten (siehe Abbildung 1) stellt eine auf Standardhardware aufbauende, umfassende Cloud- und Containerinfrastruktur automatisch bereit. Zu der gehören Betriebssystem, Virtualisierung, Speichertechnik, Netzwerk, Datenbank auf Infrastrukturebene ebenso wie die IaaS-Schicht (Infrastructure as a Service), die Self-Service-Clusterverwaltung von Kubernetes-Containerinfrastruktur, ein Werkzeugkasten für die Containernutzer zur Bereitstellung von Service Meshes, Monitoring, Speicher- und Netzwerke aus der Infrastruktur sowie eine vertrauenswürdige Container-Registry und Sicherheitsscans. Ebenso werden Werkzeuge für den Betrieb installiert, die das Testen, die Lebenszyklusverwaltung (Installation, Updates und Dekomissionierung), das Überwachen (Monitoring, Alerting, Trending), die Kapazitätsplanung und die Fehlerdiagnose unterstützen. Und natürlich beherrscht es die Nutzerverwaltung und Nutzerföderierung mit externen Identitätsanbietern und Partnern aus dem GAIA-X-Ökosystem.

Plattformdienste (PaaS) als Bausteine für Anwendungsentwickler sind geplant – Datenbanken, Frameworks für Big Data und



Die Architektur von SCS nutzt bewährte Open-Source-Kompontenten vom lokalen Portal bis zu den verschiedenen Ebenen der Cloud-Infrastruktur (Abb. 1).

für künstliche Intelligenz sowie Function as a Service sind hier die ersten Themen. Hier soll bewusst behutsam vorgegangen werden und nur Basisdienste, die allgemein als Standard erwartet werden (etwa ein relationaler Datenbankdienst), sollen mit in die SCS-Basis einfließen. Die Breite des Angebots kommt durch ein Ökosystem von Partnern zustande, die eigene innovative Dienste auf die Beine stellen, um sich von Konkurrenten zu unterscheiden.

Offen in Lizenz, Entwicklung, Community und Design

Jede integrierte Software muss unter offenen Lizenzen bereitgestellt sein. Mindestanforderung ist dabei eine OSI-kompatible Lizenz. In der Cloud-Welt ist die Apache-2-Lizenz sehr verbreitet, bei der Nutzung von Projekten ist eine offene Lizenz aber nicht ausreichend. SCS orientiert sich an den vier Offenheiten, die sich in der Definition der Open Infrastructure Community finden: offene Lizenzen, offene Entwicklung, offene Community und offenes Design. Darüber hinaus wird SCS Projekte bevorzugen, deren Reifegrad überzeugt und die aktiv weiterentwickelt oder zumindest gepflegt werden.

Auch SCS-Clouds werden Selbstbeschreibungen, wie sie GAIA-X erwartet, bereitstellen. Darüber hinaus bringt die Festlegung auf freie Software ein hohes Maß an Transparenz, was die Vertrauenswürdigkeit erhöht und die Nachvollziehbarkeit fördert. SCS wird – soweit möglich – keine Softwarekomponenten selbst entwickeln. Die Entwicklungsarbeit fließt in Automatisierung, Testabdeckung und durch aktive Mitarbeit zurück in offene Softwareprojekte, die in SCS zum Einsatz kommen.

Drittens arbeitet SCS aktiv mit den SCS-Anbietern zusammen, indem es die Kompatibilität prüft und zertifiziert, die Vernetzung herstellt und die Nutzerföderierung etabliert. Das macht es für die Endanwender möglich, ohne Technologiebruch viele Clouds föderiert, ohne große Komplexität auch in Szenarien zu nutzen, wo eine der SCS-Clouds im eigenen Rechenzentrum läuft. Das SCS-Projekt ist die neutrale Instanz, um eine gemeinsame Wissensbasis und Inspiration zu Betriebsthemen und modernen Prozessen aufzubauen. Sicherheitszertifizierungen erfolgen in der Regel betreiberspezifisch durch Dritte, SCS will dies durch entsprechendes Softwaredesign und dokumentierte Prozesse deutlich erleichtern und die Zusammenarbeit zwischen Betreibern moderieren.

Nur alter Wein ...?

Auf den ersten Blick mag man glauben, dass SCS nur wenig Neues bringt und eigentlich nichts anderes ist als eine integrierte Open-Source-Distribution einiger Infrastrukturtechnologien wie Ceph Storage, OpenStack IaaS und einer Containerplattform mit entsprechenden Tools. Tatsächlich liefert SCS all dies, integriert und aus einem Guss, ein Ziel, das die üblichen Distributionen nicht alle erreichen. SCS setzt aber andere Schwerpunkte:

- Anders als eine klassische Distribution, die dem Nutzer alle Freiheiten lässt, die Installation auf seine spezifischen Bedürfnisse zuzuschneiden, will SCS detaillierte Standards etablieren, die für den Nutzer ein viel besseres, zertifizierbares Kompatibilitätsversprechen gewährleisten können.
- Dank der kompatiblen Technik ergibt auch die Föderierung zu einer gemeinsam nutzbaren Plattform Sinn – mit föderierbarer Nutzerverwaltung und entsprechender Vernetzung. Die gleichzeitige Nutzung mehrerer Anbieter wird für Nutzer angenehmer.
- Die Herausforderungen eines professionellen Betriebs sind im Fokus von SCS. Dazu gehören ein ständiger Testprozess (CI) beim Infrastrukturanbieter, die Fähigkeit zu ständigen Updates sowie Werkzeuge zur Überwachung und Fehleranalyse.
- Die SCS-Anbieter müssen zum Erreichen höherer SCS-Zertifizierungsstufen Transparenz über Betriebsprozesse herstellen. Dies ermöglicht einerseits Kunden den Einblick und ist die Basis für ein hohes Vertrauen. Es erlaubt andererseits eine Zusammenarbeit beim Betrieb, sodass nicht jeder Anbieter hier alle Herausforderungen wieder neu lösen muss. So entsteht ein wachsender Kanon an Best Practices, auf dem alle aufbauen können, was die Qualität aller Anbieter transparent macht und voranbringt.

Projektorganisation

Das SCS-Projekt wurde im November 2019 im Umkreis der Open Source Business Alliance – Bundesverband für digitale Souveränität e. V. (OSB Alliance) initiiert. Nach Diskussionen mit der zur gleichen Zeit entstehenden Agentur für Sprunginnovationen (SPRIN-D), mit dem Ministerium für Wirtschaft und Energie (BMWi), diversen Teilnehmern von GAIA-X und schließlich dem Architekturboard von GAIA-X ergab sich die heutige Arbeitsstruktur mit einem dreiköpfigen Team, gefördert aus Mitteln der SPRIN-D und mit breiter Unterstützung einer stetig wachsenden Zahl von Unternehmen (siehe Abbildung 5). Das Projekt ist als Unterarbeitsgruppe in GAIA-X eingegliedert und bei der Weiterentwicklung von GAIA-X als Arbeitspaket und als Communityprojekt eingeordnet. Derzeit hat Gaia-X 13 Working Groups und drei Communityprojekte, neben SCS beispielsweise auch die Federation Services.

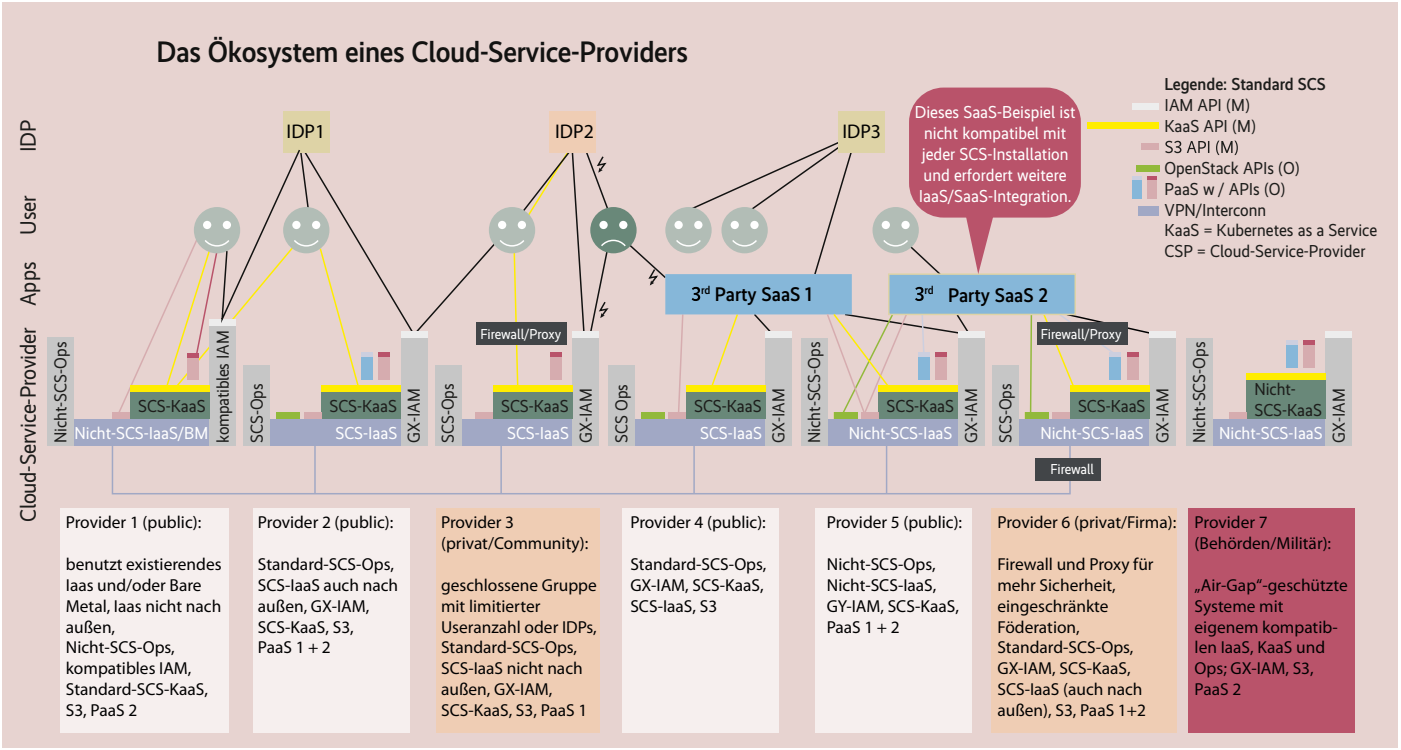
SCS sieht sich als zentrales Team in einer offenen Community – die Abstimmung erfolgt über Task-Boards, Mailinglisten und wöchentliche Termine, in denen Sprintergebnisse bewertet werden, neue User Stories vorbereitet werden und die Planung des nächsten Sprints stattfindet. Um das zentrale Team formte sich so ein regelmäßig zuarbeitender Kern von mehr als einem Dutzend Mitarbeitern aus interessierten Unternehmen.

Eine längerfristige öffentliche Förderung ist angestrebt, auch um das zentrale, neutrale Team mit dem Projekt wachsen zu lassen und um Aufträge für Arbeiten vergeben zu können, die sich aus dem Eigeninteresse einzelner Partner alleine nicht rechtfertigen lassen. Langfristig denkt man zur Finanzierung der zentralen Koordination über genossenschaftliche Strukturen oder eine Stiftung nach.

Abbildung 2 zeigt das Ökosystem von SCS. Nicht alle dort erwähnten Anbieter offerieren Public Clouds; nicht alle bieten die Schnittstellen auf IaaS-Ebene an, auch die beiden optionalen PaaS-Komponenten sind nicht überall vorhanden. Die Föderierung kann über Identity-Provider erfolgen; eine Vernetzung wird vom Projekt GAIA-X Interconnection unterstützt. Die Grafik skizziert zwei beispielhafte Third-Party-SaaS-Anwendungen: Die eine nutzt ausschließlich die vorgeschriebenen SCS-Standards und läuft daher auf jeder zertifizierten SCS-Umgebung (3rd Party SaaS 1). Die andere nutzt optionale Standards – auch sie ist portabel, aber nur über eine Untermenge von SCS-Umgebungen (3rd Party SaaS 2).

Technische Umsetzung

In einem Workshop im Januar 2020 wurde die grundlegende Architektur von Sove-



Die Abbildung zeigt sieben unterschiedliche Einsatzszenarien für SCS mit Blockdiagrammen für die unterschiedlichen APIs (Farben siehe Legende), die sie benutzen. Von links (Public Cloud) nach rechts (Hochsicherheits-Cloud) steigt dabei der Grad an Privatsphäre, die militärisch-administrative Cloud ist nicht nach außen angebunden (Abb. 2).

reign Cloud Stack entwickelt, mit behutsamer Weiterentwicklung ist zu rechnen. Das SCS-Projekt nutzt den Open Source Infrastructure & Service Manager OSISM und konnte dessen Hauptentwickler gewinnen. So stand ein bereits bei vielen Kunden erprobtes System zur Verfügung. Konfiguration und Installation der Basisdienste, der Betriebswerkzeuge und der IaaS-Schicht ließen sich einfach automatisieren.

Abbildung 1 und 2 zeigen einen vorläufigen Blick auf die vielfältigen Komponenten und die Architektur von SCS. Die grundlegende Virtualisierung von Compute, Storage und Netzwerk erfolgt mithilfe der bewährten Technologien KVM, Ceph und OVN, die Verwaltung der Infrastrukturdienste über die zuständigen OpenStack-Komponenten. Die lassen sich standardisiert für Nutzer zur Verfügung stellen oder nur intern von der darüberliegenden Containerschicht benutzen. Auf der Ebene können Kunden per Selbstbedienung Kubernetes-Cluster provisionieren und verwalten.

Die Schnittstellen im Containerbereich sind vorgeschrieben, hier entsteht die Kompatibilität zwischen verschiedenen SCS-Umgebungen. Daneben ist die S3-Schnittstelle für Objektspeicher vorgeschrieben, ebenso wie die Föderierungsmöglichkeiten im Identity Management rechts im Bild. In einem nächsten Schritt kommen als optionale Standards die Plattformdienste dazu. Auf der linken Seite ist eine Untermenge der Werkzeuge skizziert, die dem Betriebsteam die Automatisierung und Überwachung der Plattform erleichtern.

Die Dienste sind alle in Container verpackt, die mittels Docker, demnächst Podman (siehe S. 101), verwaltet werden. Diese Kapselung erlaubt es, Lebenszyklen der einzelnen Dienste unabhängig voneinander zu gestalten und nebenwirkungsfrei zu steuern. Auf dieser Ebene kommt Ansible zum Einsatz, SCS benutzt unter anderem ceph-ansible und kolla-ansible. Auf die Orchestrierung durch Kubernetes verzichtet man, die Vorteile rechtfertigen nach dem Urteil des Entwicklungsteams nicht die zusätzliche Komplexität. Allerdings kann sich diese Bewertung in Zukunft ändern.

Die für Kunden nutzbare Containerschicht ist noch nicht vollständig umgesetzt. Diese fertigzustellen, ist ein wichtiges Ziel für die kommenden Monate. In Zusammenarbeit mit dem Identity-und-Access-Management-Team aus dem GAIA-X-Demonstrator hat SCS OSISM um einen Keycloak-Container ergänzt. Dieser erlaubt Identitätsföderierung und das Abbilden von Attributen auf Rollen und Rechte. Zwar sind die grundlegenden Föderierungsmechanismen auch im OpenStack-Dienst Keystone direkt möglich, aber die Flexibilität in SCS ist höher und erlaubt auch die Föderierung auf verschiedenen Ebenen, nicht nur auf der durch Keystone kontrollierten IaaS-Schicht.

SCS angetestet

Auch ohne Containerschicht lässt sich SCS bereits testen. Für eine produktive Umgebung sollte man den SCS-Stack auf echter Hardware betreiben. Die kleinste sinnvolle Umgebung umfasst drei hyperkonvergente Maschinen mit kombinierten OpenStack-Diensten, OpenStack Compute und Ceph Storage plus einer separaten Maschine fürs Management (Abbildung 3). Die Automatisierung der Installation erfolgt mithilfe von MaaS (Metal as a Service). Das Inventory wird mittels Netbox verwaltet. Alle Informationen werden dann Ansible zur Verfügung gestellt, um den restlichen Prozess zu automatisieren.

Aber zum Testen und für Demos steht noch ein zweiter Weg zur Verfügung: SCS lässt sich auch auf virtualisierter Hardware installieren. Dafür kommt Terraform zum Einsatz – die Schritte mit MaaS und Netbox werden also durch Terraform ersetzt. Die Ansible-Automatisierung, die anschließend übernimmt, bleibt gleich.

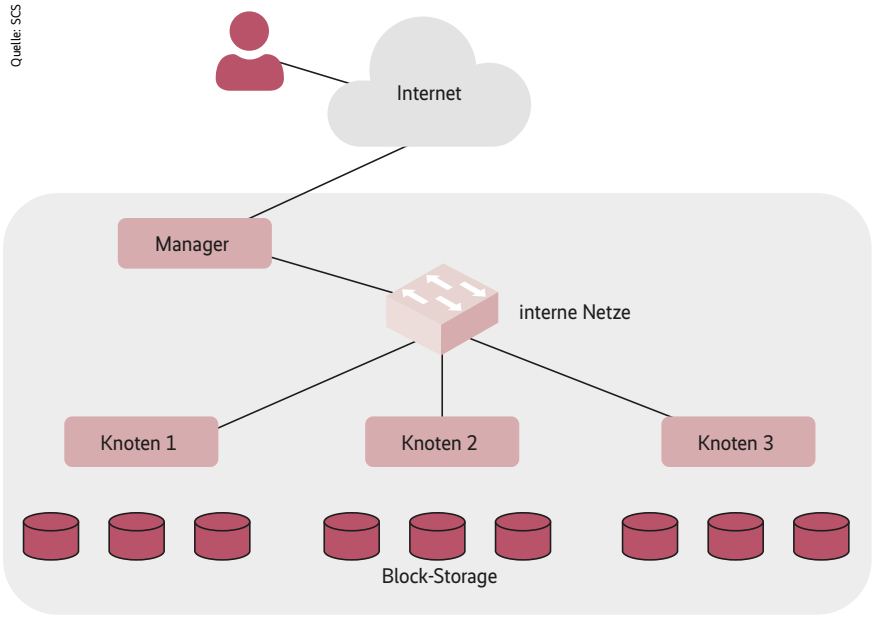
Das SCS-Projekt hat derzeit das Terraform-Deployment auf OpenStack umgesetzt. Somit kann man SCS, das ja OpenStack mitbringt, auch auf OpenStack selbst aufsetzen – bei aktivierter Nested Virtualization überraschend benutzbar.

In SCS wird diese Art der Installation auf virtueller Hardware „testbed" genannt. Der Code dafür ist ebenfalls auf GitHub verfügbar und gut dokumentiert. Dieser Installationsweg ist für Tests und Demozwecke extrem nützlich. Im Rahmen der automatischen Tests (CI) werden täglich automatisierte SCS-Deployments erstellt und durchgetestet.

Zum Ausprobieren nötig sind:

- API-Zugriff auf ein Projekt in einer einigermaßen standardkonformen OpenStack-Cloud mithilfe von Kommandozeilenwerkzeugen. Die Quotas müssen ausreichend groß sein – auf manchen Clouds dürfte man sicher zunächst nicht ausreichend Arbeitsspeicher haben (siehe Kasten „Hardwareanforderungen"). Idealerweise erlaubt die Cloud auch Nested Virtualization oder bringt Bare Metal Flavors. Ohne geschachtelte Virtualisierung ist die Leistung in den inneren VMs nur gering, da hier dann QEMU die vollständige Emulation der virtuellen CPUs erledigt.
- Eine funktionierende Terraform-Installation (0.12 oder neuer) inklusive `terraform-provider-openstack` und `terraform-provider-local` sowie `git` und optional `sshuttle`.

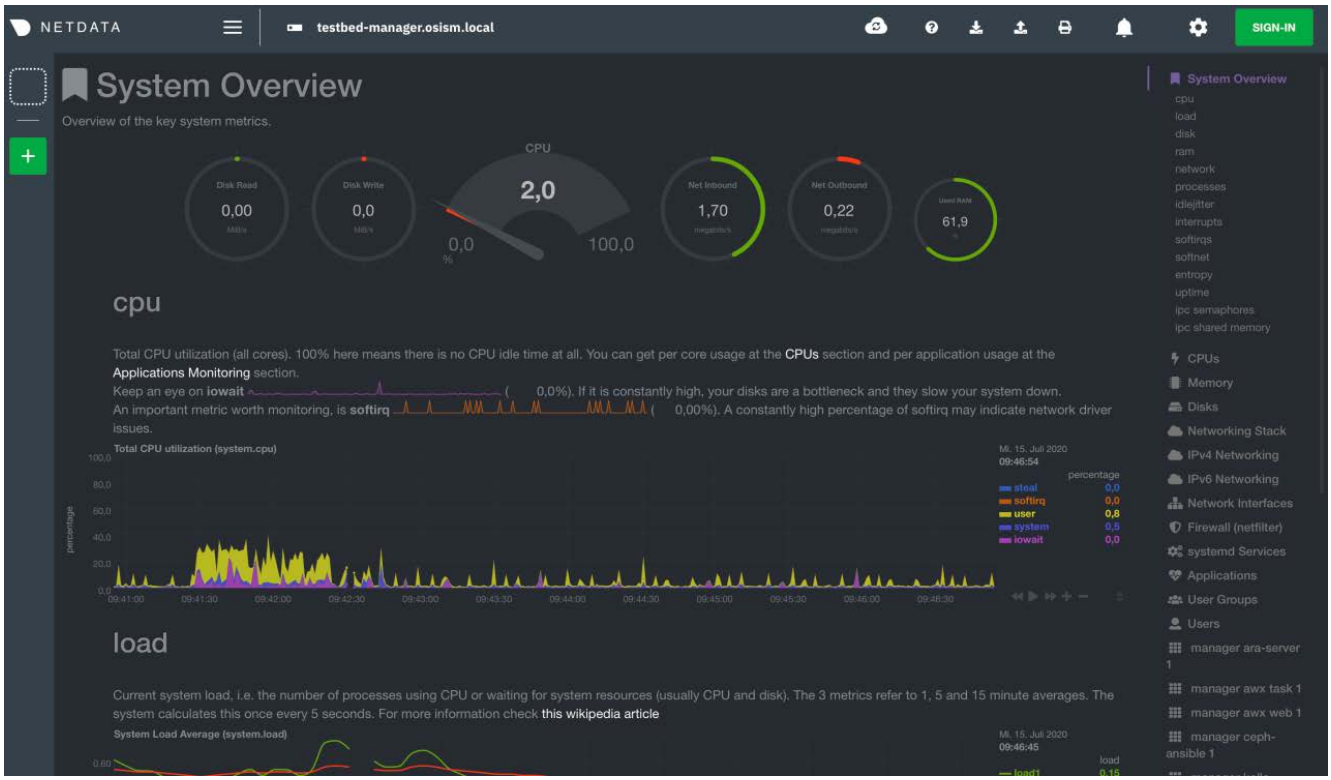
Quelle: SCS



Das minimale SCS-Set-up erfordert drei hyperkonvergente Knoten und ein Managementsystem (Abb. 3).

Grundsätzlich wäre über Terraform auch das Set-up auf einem großen einzelnen Knoten (mittels `terraform-provider-libvirt`) oder auf anderen IaaS-Umgebungen möglich. Umgesetzt wurde dies in SCS noch nicht, Interessenten mögen den Autor des Artikels kontaktieren. Im Verzeichnis terraform des geklonten GitHub-Repository testbed finden sich Dateien mit dem Namen environment-XXX.tfvars. Falls sich hier (noch) keine passende Konfiguration findet, muss diese von Hand erstellt werden. Dafür werden folgende Informationen benötigt:

Quelle: SCS



Netdata stellt das Verhalten der Systeme live und sehr anschaulich dar (Abb. 4).

- der Name der genutzten Cloud in `cloud_provider` – dies sollte dem Namen in clouds.yaml entsprechen;
- der Flavor für die drei hyperkonvergenten Knoten (mit mind. 30 GByte RAM) und der Flavor für den Managementknoten in `flavor_node` und `flavor_manager`;
- der Name des Ubuntu-18.04-Images in `image`;
- der Name des öffentlichen OpenStack-Netzwerks (das man für die öffentlichen Floating IPs nutzt) in `public`;
- die gewünschte Verfügbarkeitszone für Compute und Storage in `availability_zone` und `volume_availability_zone`.
- Falls auch das Netzwerk in Verfügbarkeitszonen aufgeteilt ist, gehört diese in `network_availability_zone`. Falls nicht, muss man in neutron.tf alle `availability_zone_hints` auskommentieren.

Wenn das ausgefüllt ist, wird das Deployment mit

```
make ENVIRONMENT=XXX ↵
                    deploy-openstack watch
```

gestartet. Nachdem Terrraform die Volumes, Netze und VMs angelegt hat, startet dort Ansible die Installation und Konfiguration aller Dienste. Das Watch Target beobachtet dies – nach einigen Minuten steht auf dem Managementknoten auch ein ARA (ARA Records Ansible) bereit, ein Report, den man auf einer Weboberfläche beobachten kann.

Auf manchen OpenStack-Clouds gibt es jedoch Schwierigkeiten: Beim Cloud-Anbieter OVH muss vRacks (private Netze) aktiviert sein und dann das Deployment mehrmals starten, weil die Netzressourcen erst verzögert nutzbar werden. Für die Open Telekom Cloud muss der Tester die Namen der Netzwerkinterfaces an die dortige Realität anpassen, die Bootstrap-Skripte ändern und ein paar Spezialitäten im Netzbereich berücksichtigen (`enable_snat`).

Sobald das Deployment des Manager Node erfolgreich war, ist der Log-in dort möglich (nach einem `make ENVIRONMENT=XXX ssh`) und der Administrator kann selbst prüfen, welche Container laufen oder welche Playbooks erneut anzustoßen sind. Die Automatisierung ist unter `/opt/configuration` einfach einsehbar. Auf einer standardkonformen Cloud wie der von Citynetwork oder der von PlusServer für GAIA-X bereitgestellten SCS-Entwicklungs-Cloud ist das nicht notwendig, auch weil das Deployment hier kontinuierlich getestet wird.

Zur Verwaltung stehen einige Weboberflächen zur Verfügung, die nach einem `make ENVIRONMENT=XXX sshuttle` vom lokalen PC aus erreichbar sind, beispielsweise Cockpit, das Ceph-Dashboard, Zabbix, Netdata (Abbildung 4), Skydive (nach separatem Start), Patchman, Kibana oder auch Keycloak. Für Nutzer ist das OpenStack-Dashboard Horizon verfügbar.

Für die ständige Überwachung kommen Tools wie Prometheus und seine Werkzeuge zum Einsatz. Sie speichern regelmäßig Maschinenwerte als Zeitserien, bilden Trends und lösen bei Überschreitung von Schwellenwerten Alarme oder korrigierende Maßnahmen aus.

In der Grundkonfiguration von testbed sind nur die Kerndienste von OpenStack immer aktiv – andere OpenStack-Komponenten lassen sich bei Bedarf durch Kommandos wie

```
osism-kolla deploy heat,gnocchi, ↵
          ceilometer,aodh,panko,magnum, ↵
                        barbican,designate
```

auf dem Management Knoten aktivieren.

Ein Teil der automatisierten CI-Tests von SCS ist RefStack, das die installierte OpenStack-Umgebung über die API auf Kompatibilität mit den Trademark-Richtlinien der Open Infrastructure Foundation prüft. Von Hand können die Tests auf dem Managementknoten mit `/opt/configuration/contrib/refstack/refstack.sh` gestartet werden. Ein weiterer nützlicher Test ist der `openstack-health-monitor` (ebenfalls im Git-Repository).

Wer für die Ressourcen in der Cloud bezahlen muss, tut gut daran, am Ende wieder alles wegzuräumen – sonst kommen bei den üblichen Tarifen der Cloud-Anbieter schnell Tausende Euro pro Monat zusammen. Das Aufräumen erfolgt Makefile-typisch mittels `make ENVIRONMENT=XXX clean`, was in einem `terraform destroy` mündet.

Noch nicht fertig integriert in SCS ist die Containerschicht. Aber grundlegend klar ist bereits, wie sie aussehen muss: Der Nutzer soll über standardisierte Schnittstellen einen oder mehrere Kubernetes-Cluster bereitstellen und verwalten können, wobei Storage- und Netzwerkfähigkeiten der darunterliegenden IaaS über die entsprechenden Adapter (CSI und CNI, Cloud Storage und Cloud Networking Interface) verfügbar gemacht werden. Zu der Containerplattform gehören auch Werkzeuge zur Bereitstellung von Proxys, einem Service Mesh, einer Container-Registry, Sicherheitsüberprüfungsdienste für Netzwerk und Images sowie die Automatisierung von Container-Deployments.

To-do: Containerschicht

Während Kubernetes selbst sehr gut spezifiziert ist und sich bei den genannten Tools Standards bilden, ist der erste Schritt, die Clusterverwaltung, leider noch nicht durch umfassende und breit akzeptierte Standards möglich. Die Kubernetes Cluster API gilt hier als Kandidat, aber nach Auskunft von Projekten und Anbietern wie SAP Gardener, Kubermatic und Rancher ist der Reifegrad noch nicht hoch genug, sodass die Clusterverwaltung zwar cloudübergreifend, aber nicht herstellerübergreifend möglich ist.

Das ist ein Hemmschuh für das SCS-Projekt. Das Ziel ist ja die Etablierung von Standards, um eine Fragmentierung zu vermeiden. Da ist es keine gute

Mindestanforderungen an die SCS-Hardware

Die kleinste Testbed-Installation benötigt überraschend wenig Ressourcen:

- Die drei hyperkonvergenten Knoten verlangen acht CPU-Kerne, 30 GByte Speicher, drei Festplatten größer als 10 GByte und acht Netzwerke.
- Die Managementknoten müssen mindestens vier CPU-Kerne und 8 GByte Speicher enthalten.
- Dazu kommen die Festplatten für das zu bootende Betriebssystem.

Bei typischen Flavors ergibt das eine RAM-Quota von 104 GByte (4-mal 32 GByte plus 8), was bei einigen Cloud-Providern über den standardmäßigen 50 oder 100 GByte liegt. Dazu kommen 28 vCPUs. Andere Provider begrenzen die Anzahl der Netze oder gar der Sicherheitsgruppenregeln. SCS erstellt 6 Sicherheitsgruppen mit insgesamt knapp 40 Regeln. 90 GByte Blockspeicher stellen für die Anbieter normalerweise keine Herausforderung dar.

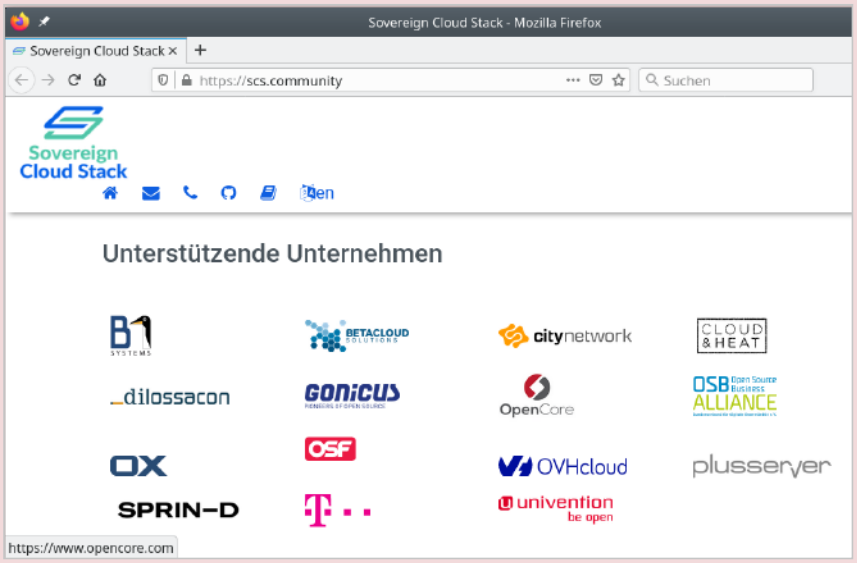
Installationen auf echter Hardware verlangen für die CPUs und den Arbeitsspeicher der hyperkonvergenten Knoten mindestens das Vierfache – wenn signifikante Last erwartet wird, natürlich entsprechend mehr. Es bietet sich dann auch der Einsatz von deutlich mehr Knoten an. Der Managementknoten sollte mit dem Gesamt-Set-up wachsen und muss für produktive Umgebungen redundant ausgelegt werden. Entsprechend viele Netzwerkkarten sind notwendig für redundante (LACP-)Verbindungen zu mindestens sechs Netzen.

Mitmachen!

Sovereign Cloud Stack ist ein offenes Projekt. Während es sich über jeden Benutzer freut, freut es sich noch viel mehr über jeden, der auch mitarbeiten möchte. Momentan kommt die Mitarbeit jenseits der existierenden Open-Source-Projekte zu einem großen Teil aus einer Handvoll von Firmen, vielfach aus dem Kreis potenzieller Cloud-Anbieter (Abbildung 5).

Die Gespräche mit Unternehmen aus dem privaten Sektor und auch mit IT-Betreibern für die öffentliche Hand deuten darauf hin, dass die Unterstützung weiter wächst und auch vielfältigere Interessen und Beiträge integrieren wird.

Eine stetig wachsende Zahl an IT-Firmen, nicht nur aus der Open-Source- oder Hostingbranche unterstützt das SCS-Projekt (Abb. 5).



Option, mehrere alternative sich überlappende Schnittstellen zu implementieren. Ebenso wenig ist es eine gute Option, eine der Lösungen zu unterstützen, weil das unter Umständen andere Anbieter ausschließt und die Gefahr mit sich bringt, dass die gewählte nicht allgemein akzeptiert wird.

Das SCS-Projekt hat erfolgreich drei der möglichen Lösungen – Rancher, Kubermatic und SAP Gardener – auf SCS getestet und die automatisierte Umsetzung erkundet. Alle drei konnten erfolgreich implementiert werden, die Ergebnisse finden sich ebenfalls auf GitHub.

Diese Systeme haben alle den Vorteil, dass sie das Clustermanagement über verschiedene Cloud-Anbieter ermöglichen, seien es die großen amerikanischen Plattformen, private oder öffentliche OpenStack-basierte Clouds (darunter auch SCS) oder direkt auf Hardware implementierte Kubernetes-Cluster. Momentan gibt es Diskussionen mit den Projekten, inwieweit gemeinsam eine standardisierte Schnittstelle weiterentwickelt und zur Reife gebracht werden kann.

Wenn die Fortschritte hier zu lange auf sich warten lassen, ist möglicherweise ein Zwischenschritt notwendig: Mit dem OpenStack Kubernetes Cluster API Provider steht eine Umsetzung des vorläufigen gemeinsamen Standards bereit. Anders als mit den genannten Produkten lassen sich damit aber erst mal nur auf OpenStack-Umgebungen Kubernetes-Cluster verwalten. Immerhin setzt aber auch VMware auf kompatible Schnittstellen, auch Microsoft arbeitet daran.

Das Clustermanagement für die Containerumgebung ist also leider noch nicht standardisiert in SCS. Dies nachzuholen ist eine der wichtigsten Aufgaben für die kommenden Monate. Aber auch sonst herrscht im SCS-Team kein Mangel an Themen.

Zukunft

Ein wesentliches Arbeitsfeld ist die Testabdeckung. Das Ziel von SCS ist es, produktive Umgebungen täglich updaten zu können, ohne damit für Kunden Risiken zu verursachen. Das ist nur möglich, wenn die Testabdeckung so gut ist, dass in den kontinuierlichen Tests keine Probleme durchrutschen.

Dass IT-Sicherheit eine entscheidende Voraussetzung für Datensouveränität ist, zeigt sich in vielerlei Aspekten in SCS. In vielen Fällen muss dies aber noch durch entsprechende Zertifizierungsverfahren bewiesen werden. Erste Vorbereitungen dafür laufen.

Künstliche Intelligenz als wichtiges Anwendungsfeld und Treiber von Innovation spielt auch für SCS eine große Rolle. Die umfangreichen Berechnungen in diesem Bereich profitieren häufig von Beschleunigerhardware, seien es Tensorprozessoren von Grafikkarten, spezialisierte Koprozessoren oder auch programmierbare FPGAs. Standardisierte Schnittstellen auf agiler Infrastruktur müssen diese Anforderungen verfügbar machen, auch die Bereitstellung von Bare-Metal-Systemen über IaaS-Schnittstellen soll ermöglicht werden. Später will SCS auch Architekturen jenseits von x86 unterstützen, insbesondere ARM und RISC-V bieten interessante Perspektiven.

Mehr und mehr Softwareentwickler wollen fertige Bausteine nutzen. Diese lassen sich sehr komfortabel einsetzen, wenn sie als Kubernetes-Operator bereitgestellt sind. Das SCS-Team wird mit Partnern arbeiten, damit diese Bausteine gut auf SCS funktionieren. Standardisierung und Lieferung als Teil der Plattform soll dabei behutsam angegangen werden – insoweit sich hier klare, offene Standardlösungen herausbilden.

Die Architektur von SCS ist für die Nutzung in Rechenzentren optimiert, also Umgebungen, auf denen viele verschiedene USER vielfältige Ressourcen nutzen, die sicher voneinander getrennt und erfasst werden müssen. Für einheitliche Infrastrukturen, die von geschlossenen Anwender genutzt werden, beispielsweise Edge-Clouds, die nahe an Produktionssystemen Daten sammeln und verarbeiten, lässt sich die Architektur vereinfachen. Der Administrator kann auf die Nutzungserfassung verzichten, bei der Isolation Kompromisse eingehen und die Virtualisierungsschicht vereinfachen oder weglassen. (mfe@ix.de)

Kurt Garloff

war bis Ende 2018 als Chefarchitekt für den Aufbau und die Weiterentwicklung der Open Telekom Cloud verantwortlich. Nach einem Jahr als Entwicklungsleiter für Cloud, Storage und Entwicklungs-IT bei SUSE widmet er sich jetzt voll der Leitung des GAIA-X-Community-Projekts Sovereign Cloud Stack.