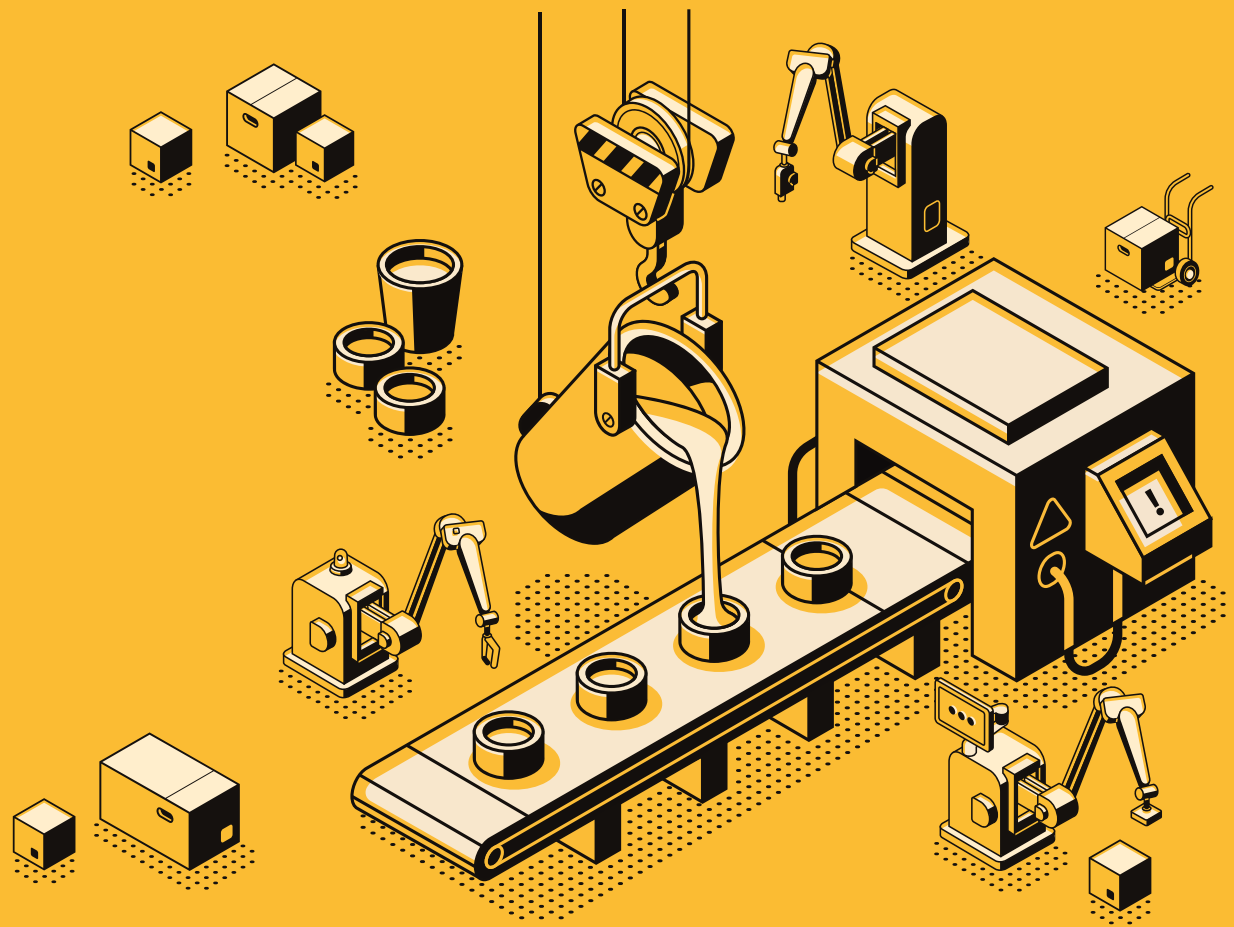
Nr. 3 | Juni 2019 | ISSN 2366-7915 | www.doag.org

Red Stack Magazin

DOAG SOUG swiss oracle user group AOUG AUSTRIAN ORACLE USER GROUP

AUTOMATISIERUNG



Aus der Praxis

Systematische Datenbank-Performanceoptimierung

Im Interview

Ing. Daniel Hafner und Ing. Klaus-Michael Hatzinger, DBConcepts GmbH

Automatisierung

Automatisierung von Datenbank-Backups mit der Zero Data Loss Recovery Appliance

DOAG UNIVERSITY

Finden Sie die passende Schulung im Oracle-Umfeld auf

university.doag.org

- Oracle-Technologien
- IT-Methoden
- IT-Management

Erhalten Sie als **DOAG-Mitglied** einen **exklusiven Rabatt** auf den regulären Kurspreis.

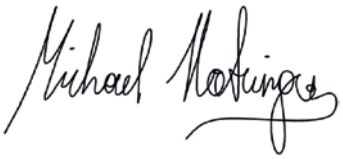
Klaus-Michael Hatzinger
AOUG-Präsident

Liebe Mitglieder, liebe Leserinnen und Leser,

Automatisierung im Bereich der Infrastruktur ist heute im Cloud-Zeitalter nicht mehr wegzudenken. Stetig steigende und immer komplexer werdende Anforderungen an das Management, die Verfügbarkeit, Skalierung und die Provisionierung von Systemen und Services machen es notwendig, hier weitgehend zu automatisieren. Auch die künstliche Intelligenz hält beim Thema Automatisierung Einzug und übernimmt zum Beispiel vollautomatisch administrative Tätigkeiten zur Optimierung und den Betrieb von Oracle-Datenbanken. Durch maschinelles Lernen werden die Systeme überwacht und laufend analysiert. Aufkommende Probleme und Optimierungspotenzial können dadurch schon frühzeitig erkannt, vorhergesehen und präventiv automatisch korrigiert werden.
Zahlreiche Hersteller beschäftigen sich mit Infrastruktur-Automatisierung und auch die Open-Source Community hat hier einiges zu bieten. In der aktuellen Ausgabe des Red Stack Magazins haben wir uns dem Thema gewidmet und für Sie interessante Anwendungsfälle und Technologien zusammengetragen.

Ich wünsche Ihnen viele interessante Momente beim Lesen dieser Ausgabe.

Ihr



Ein „Platzhirsch“ im Bereich des Deployens von Infrastrukturen ist Terraform

APEX hat den neuen Release-Zyklus und das neue Versions-Schema für Oracle-Produkte übernommen.

Wie lassen sich unerwartete Performance-Probleme in Oracle-Datenbanken lösen?

Cloud

Automatisierung

APEX

Open Source

Datenbank

Tuning

Intern

Timeline

4. April 2019

Im Vorfeld der Collaborate 19 haben die Usergruppen bekannt gegeben, dass die IOUG mit der Quest Oracle Community fusionieren wird und ab Anfang Mai unter den Organisationsrahmen der Quest Oracle Community schlüpfen wird. Damit bietet die Quest Oracle Community dann die weltweit weitaus größte Usergruppe mit zusammen mehr als 70.000 Mitgliedern.

7. bis 11. April 2019

Die Konferenz COLLABORATE 19 fand vom 7. bis 11. April in San Antonio, Texas statt. Das Technology and Applications Forum for the Oracle Community ist eine Live-Veranstaltung, die von Anwendern für Anwender konzipiert wurde und von der Independent Oracle Users Group (IOUG), der Oracle Applications Users Group (OAUG) und der Quest Oracle Community (Quest) präsentiert wird. Die Collaborate bietet als "Flaggschiffveranstaltung" der drei US-amerikanischen Usergroups OAUG, QUEST und IOUG ein umfassendes Programm für Applications- und Technologienutzer im Oracle-Markt. In der Begrüßungssession wurde eine Umbenennung angekündigt. Aus der Oracle Application User Group wird die Oracle Applications & Technical Users Group (OATUG). In den anschließenden Besprechungen der Usergroups erklärte sich die OATUG bereit, intensiver mit anderen Usergruppen (speziell auch DOAG) zusammenzuarbeiten.

25. und 26. April 2019

Rund 70 Leitungskräfte und Delegierte haben sich in Berlin zum DOAG-Leitungskräfteforum versammelt, um unter dem Stichwort „DOAG 2024" die strategische Ausrichtung des Vereins für die kommenden Jahre zu definieren. Die thematische Öffnung der DOAG und das Immobilienprojekt „DOAG Haus" standen dabei im Fokus. Der erste konkrete Schritt zur Öffnung der DOAG wird eine neue, herstellerunabhängige Konferenz zum Thema Cloud darstellen. Die Versammlung einigte sich auf den Namen „Cloud Con". Die Konzeption und Abwicklung der Veranstaltung soll eine dafür eingerichtete Arbeitsgruppe übernehmen.

27. April 2019

Im Berliner Hotel Estrel fand die Ordentliche Delegiertenversammlung statt. Die strategische Ausrichtung des Vereins für die kommenden Jahre wurde unter dem Namen „DOAG 2024" beschlossen. Es wurden außerdem mit Niels de Bruijn und Björn Bröhl zwei neue Community-Leiter als Vorstandsmitglieder gewählt. Diese treten die Nachfolge der zurückgetretenen Vorstandsmitglieder Robert Szilinski (Development Community) und Rolf Scheuch (Data Analytics Community) an. Die zwei neugewählten Community-Leiter sind bereits seit vielen Jahren bei der DOAG aktiv. Die ordentliche Delegiertenversammlung der DOAG findet jährlich in Berlin im Anschluss an das DOAG-Leitungskräfteforum statt.

6. bis 9. Mai 2019

Auf der APEX connect 2019 in Bonn gab es diesmal ein Wiedersehen der Community im Kameha Grand Hotel direkt am Rhein. Über 70 Vorträge zu den drei Streams APEX, SQL & PL/SQL und JavaScript boten eine vielfältige Auswahl an interessanten Themen. Auch in diesem Jahr bot die Veranstaltung wieder die ideale Umgebung für Wissensaustausch und Networking. Die ausgezeichnete Location des Kameha Grand Hotels, die gute Atmosphäre und eine stimmungsvolle abendliche Rheinschifffahrt sorgten für allseits zufriedene Besucher. Mit 375 Teilnehmern konnte die fünfte Ausgabe der Veranstaltung einen neuen Besucherzahlen-Rekord aufstellen und zu einem vollen Erfolg werden. In diesem Jahr konnten zudem allein sechs internationale Keynote-Speaker gewonnen werden.

14. Mai 2019

Im Kreativbezirk Berlin-Kreuzberg fand mit 66 Teilnehmern der erste Netsuite User Day statt. Flexibilität und Zuverlässigkeit sind nur zwei von vielen Eigenschaften, die Unternehmen von einem ERP-System erwarten. Deswegen unterzog sich NetSuite beim ersten NetSuite User Day dem Härtetest der Community – ungeschminkte Erfahrungsberichte, Lesson Learned und Best Practices standen dabei auf der Agenda. In sechs Sessions wurden vielfältige Aspekte von NetSuite behandelt. So stellte Matthias Reinecke (Regional Product Manager DACH – NetSuite Germany) beispielsweise Neuerungen der Version 2019.1 vor. Christian Walch (KPMG AG) vermittelte einen Einblick hinsichtlich der Herausforderungen eines multinationalen Pharma-Unternehmens bei der Einführung von NetSuite.

16. bis 18. Mai 2019

In Rottach-Egern am Tegernsee trafen sich die Vorstände zur Vorstandssitzung 2/19. Aufbauend auf den Beschlüssen der Delegiertenversammlung stand die Ausarbeitung und Umsetzung der Strategie „DOAG 2024“ auf der Agenda. Die beiden Projekte „DOAG Haus“ und „CloudCon 2020“ wurden auf den Weg gebracht.

29. bis 31. Mai 2019

Rund 15 Oracle-Anwendergruppen aus der EMEA-Region sprachen auf dem Riga DevDays-Meeting in Riga über die Intensivierung ihrer zukünftigen Zusammenarbeit. Da sich Oracle zunehmend weniger für die Usergruppen engagiert, wollen die Vertreter der europäischen Anwender die Organisation und Durchführung ihrer halbjährlichen Treffen mehr und mehr in die eigenen Hände nehmen. Das nächste Treffen findet dann im Rahmen der DOAG Konferenz + Ausstellung 2019 in Nürnberg statt.

Dr. Dietmar Neugebauer
Ehemaliger DOAG-Vorstandsvorsitzender

Aus der Ferne betrachtet: Nur wo DOAG drauf steht, ist auch DOAG drin …

Der Gründung der DOAG Deutsche ORACLE-Anwendergruppe lag sicherlich die Idee zugrunde, den Anwendern der Oracle-Produkte als Plattform für ihre Interessen zu dienen, einen Erfahrungsaustausch anzubieten und die Vernetzung der Nutzer zu unterstützen. Bemerkenswert war dabei, dass sich die DOAG von Anfang an eine Unabhängigkeit von Oracle bewahren wollte. Kritischer Umgang mit der Weiterentwicklung und Qualität der Produkte sowie mit der Effizienz des Supports hatten von Anfang an einen hohen Stellenwert innerhalb der DOAG. Auch der Positionierung von Oracle gegenüber anderen Datenbankprodukten stand man offen gegenüber. So kann ich mich gut erinnern, dass die DOAG in den 90er Jahren in einer Seminarreihe einen Vergleich zu den damals gängigen Datenbankprodukten mit entsprechenden Experten angeboten hat.

Mit den Zukäufen von Oracle wie Peoplesoft, Hyperion, Primavera, Siebel oder auch Bea und Sun hat sich das Anwenderspektrum für die Oracle-Produkte natürlich immens erweitert. Die DOAG hat versucht, allen diesen hinzugekommenen Anwendern eine Plattform zu bieten. Dies ist jedoch nicht immer erfolgreich gewesen. Irgendwann musste auch die DOAG einsehen, dass ein Java-Entwickler sich nur schwer oder gar nicht mit Oracle identifiziert und sich auch nicht von der DOAG vertreten lassen will. Hier hat man dann mit der Gründung des IJUG (Interessenverbund der Java-Usergruppen) eine Lösung gefunden, die außerhalb und doch mit der DOAG zusammen die Interessen dieser Anwender erfolgreich vertreten kann.

Inzwischen haben sich in den letzten Jahren viele neue Ideen in der DOAG entwickelt. Konferenzbesucher erwarten nicht mehr nur rein technische Vorträge, sondern sie nutzen die Gelegenheit, sich Wissen über Projektmanagement, Teambildungsprozesse, agile Softwareentwicklung, Präsentationstechniken, Menschenführung einzuholen und auch hier ihre Erfahrung mit anderen auszutauschen. Mit der Gründung der NextGen Community hat die DOAG auch einen wichtigen Schritt in Richtung Nachwuchsförderung getan. Hierbei ist sicherlich ein wichtiger Punkt, den Nachwuchs bereits in der Ausbildung Informationen zu Oracle-Produkten zukommen zu lassen. Andererseits hat man hier schnell erkannt, dass in der Ausbildung Oracle nur einen relativ geringen Stellenwert einnimmt und man sich auch hier mit den vorhandenen Interessen der Auszubildenden auseinandersetzen muss.

Mit dem inzwischen immer stärker wachsenden Markt der Anwender von Cloud-Produkten wird sich das Angebot der DOAG wohl auch hier weiter öffnen müssen. Die Anwender erwarten hier von der DOAG das, was auch in der Satzung der DOAG steht: „Der Verein fördert den Dialog, die Beratung und Zusammenarbeit mit Oracle und Anbietern anderer Produkte und Dienstleistungen“. Der Wissens- und Erfahrungsaustausch in Verbindung zu anderen Cloud-Anbietern sowie anderen Datenbank- und Softwareprodukten sollte für die DOAG keine Abgrenzung mehr sein. Wichtig ist dabei nicht, ob Oracle draufsteht, sondern wo und wie die DOAG als unabhängige Institution ihre Mitglieder am besten vertreten kann, denn wo DOAG draufsteht, muss auch weiterhin DOAG drin bleiben …

Ing. Daniel Hafner *Ing. Klaus-Michael Hatzinger*

Es geht immer mehr in den ununterbrochenen 24/7-Betrieb mit geplanten Wartungen und nur kurzen Unterbrechungen

Das Thema Hochverfügbarkeit beschäftigt immer mehr Unternehmen, deren Systeme jahrelang einwandfrei liefen, die sich aber nun doch darüber Gedanken machen, wie sie ihre Verfügbarkeit sicherstellen und verbessern können. Dies ist nicht nur eine Kostenfrage. Martin Meyer, Redaktionsleiter des Red Stack Magazins, sprach darüber mit Ing. Daniel Hafner und Ing. Klaus-Michael Hatzinger, DBConcepts

Was bedeutet Hochverfügbarkeit?

Daniel Hafner: Hochverfügbarkeit ist ein weiter Begriff. Es gibt da zwei unterschiedliche Terminologien. Generell ist HV darin definiert, dass man Gesamtsysteme hochverfügbar gestaltet. Gemäß einer Definition heißt es, dass man ein System, das selbstständig Fehlerkorrekturen durchführen kann und wieder in Betrieb geht, als hochverfügbar bezeichnen kann. Aus Oracle-Sicht ist die Hochverfügbarkeits-Definition sehr eingeschränkt, das heißt, wenn man Hochverfügbarkeit hört, dann hat man fast schon automatisch Oracle RAC (Real Application Cluster) im Sinn, das ist jedoch nicht das Einzige. Es gibt ja noch andere Technologien, die dahinterstehen. Replikationen möglicherweise, die auch Hochverfügbarkeit schaffen können. Diese sind unterschiedlicher Natur: einmal logischer und ein anderes Mal physischer Art.

Welche Voraussetzungen müssen für die Herstellung eines hochverfügbaren Systems gegeben sein?

Daniel Hafner: Das ist immer eine spannende Frage! Ausreichend Geld ist immer ein Thema. Hochverfügbarkeit steht in Korrelation zu den Kosten – sehr eng im Regelfall –, wobei man mittlerweile die Hochverfügbarkeit sehr kostengünstig erreichen kann. Es gibt zum Beispiel Aktiv-Passiv-Systeme, die relativ günstig zu realisieren sind. Es gibt allerdings größere Systeme, die hohe Verfügbarkeitsansprüche haben, die auch entsprechend

kosten. Als Voraussetzung sollte sich der Kunde zunächst im Klaren darüber sein, welche Verfügbarkeitsansprüche er denn überhaupt hat. Das ist – so glaube ich – eines der Kernthemen. Wir stellen immer diese Frage und meistens erhalten wir als Antwort: „Ja das müssen wir jetzt mal definieren"! Bei manchen Kunden ist es klar. Man sieht die Entwicklung der letzten Jahre und diese hat oft ergeben, dass die Kunden jetzt bereits Single-Server-Systeme usw. im Einsatz haben, die jahrelang einwandfrei laufen und noch nie Probleme gemacht haben, sich aber trotzdem Gedanken über deren Verfügbarkeit machen. Zentral ist die Frage danach, was es bedeutet und welche negativen Auswirkungen es haben kann, wenn beispielsweise wirklich die Produktion für eine Stunde stillsteht.

Was ich sehr oft erlebe, ist, dass es sich bei Produktionsbetrieben praktisch eingebürgert hat, dass man 4 Stunden als grobes Limit für eine maximale Produktionsunterbrechung betrachtet. Dies kann im konkreten Fall beispielsweise bedeuten, dass man beim Ausfall einer Maschine oder einer Datenbank für vier Stunden in einer Gießerei die Hochöfen herunterfahren, ein- und ausbauen und wieder neu hochfahren muss und so viele Zehntausende Euro Schaden entstehen. In solchen Fällen bekommt eine Kosten-Nutzen-Rechnung schnell eine besondere Bedeutung. Der wichtigste Punkt ist, sich Klarheit über das Ziel zu verschaffen. Danach kann man dann Konzepte aufbauen, die dementsprechend kostenangepasst sind und teilweise den Verfügbarkeitsansprüchen entsprechen, die der Kunde wirklich benötigt. Ein Thema sind meistens auch Standorte. Es gibt zum Teil Konzepte, die man innerhalb eines Standortes realisiert. Das heißt beispielsweise, dass man im Rahmen eines Hochverfügbarkeitskonzepts an einem Hochschul-Campus mehrere Räume findet und eigene Brandabschnitte auswählt. Häufig ist ebenfalls, dass man die Daten auf mehrere Data Center verteilt, die ein paar Kilometer auseinanderliegen oder je nach Firmengröße international oder interkontinental verteilt sind. Es gibt große Kunden, die auch zwischen Kontinenten replizieren. Das hat Vor- und Nachteile bezüglich der Verfügbarkeit, der Performance und der Latenzen.

Welche (technischen) Entwicklungen gab es in den letzten Jahren beim Thema Hochverfügbarkeit?

Daniel Hafner: Da gab es, wenn man es jetzt mal technisch aus Oracle-Sicht betrachtet, eine Menge. Oracle hat sehr viel am Cluster erweitert und an neuen Funktionalitäten hinzugefügt. Kernfunktionalitäten davon sind die Datenbank als RAC, RAC One Node oder ein einfacher Aktiv/Passiv Cluster.. Dies gibt es zwar schon sehr lange, aber es wurden zum Beispiel Flex-Cluster-Technologien eingeführt. Es wurde einfach viel an kleinen Stellen daran gearbeitet, die Gesamt-Verfügbarkeit zu erhöhen und ein bisschen die Flexibilität zu verbessern. Das ist meiner Meinung nach momentan der Trend, denn hochverfügbar sind die Systeme zum Teil schon. Jetzt müssen sie wieder flexibler werden. Das sieht man etwa bei dem Gedanken, dass man Datenbanken auch in die Cloud repliziert und das als zweiten Ausfallstandort eventuell in Betracht zieht. In der Gesamtbetrachtung ist es in Bezug auf die Verfügbarkeit ein sehr positiver Trend, die hochverfügbar zu haltenden Systeme zu erkennen. Viele Kunden kommen jetzt zu der Feststellung, dass ihre Systeme nicht mehr ausreichend dimensioniert sind und mehr Verfügbarkeitsanspruch haben müssen.

Klaus-Michael Hatzinger: Hochverfügbarkeit bedeutet für mich nicht nur, dass ein System technisch permanent verfügbar ist, sondern dass man auch die geplanten Downtimes minimieren kann. Diesbezüglich hat Oracle speziell die Enterprise-Edition-Datenbank so gebaut, dass man Wartungstätigkeiten durchführen kann, ohne das System stoppen zu müssen, was natürlich wieder die Verfügbarkeit verringern würde.

Daniel Hafner: Genau. Es geht immer mehr in den ununterbrochenen 24/7-Betrieb mit geplanten Wartungen und nur kurzen Unterbrechungen.

Wie viel Hochverfügbarkeit sollte ein Betrieb anstreben? Gibt es Best Practises?

Daniel Hafner: Das kommt immer darauf an. Es ist eine Kosten-Nutzen-Rechnung. Ich sage es mal so: Man kann Hochverfügbarkeit mittlerweile eigentlich günstig herstellen. Zwei Server und eine kleine Cluster-Lösung sind für 20.000 bis 30.000 Euro zu realisieren. Das geht auch relativ zügig. Wenn allerdings Hochverfügbarkeit heißt, dass ein System nicht nur hochverfügbar, sondern auch funktional bleiben soll, dann gehört da mehr dazu, wie beispielsweise applikatorisch und performancetechnisch verfügbar zu bleiben. Die Frage lautet: Wie lange ist die Meantime To Recover, wenn es wirklich mal einen Ausfall gibt? Wie lange dauert es, dass sich das System automatisch wiederherstellt? Dies muss man in Betracht ziehen und anschließend kann man eine Abschätzung vornehmen. Eine richtige Best Practise gibt es dafür nicht! Was es hingegen gibt, wenn man Oracle ins Blickfeld nimmt, ist der Maximum Availability Architecture Guide. Darin wird im Kern auf RAC und Data Guard fokussiert. Die Oracle Best Practise besteht darin, zwei Cluster-Systeme und dazwischen einen Data Guard einzusetzen, um Desaster-tolerant zu sein. Dazu gehören die Fragen: „Wie hoch ist die Fehlertoleranz?", „Wie schnell soll das System wieder verfügbar sein?", „Soll es automatisch wieder verfügbar sein?", „Soll es manuell umgeschaltet werden können?" und „Gibt es dabei Entscheidungspunkte?". Wichtigstes Thema dabei ist zum Beispiel auch, ob es einen dritten Standort geben soll. Wenn man sich ein bisschen mit der Split-Brain-Theorie auseinandersetzt, bemerkt man schnell, dass man mit zwei Standorten nicht auskommt. Ein dritter Standort ist die Minimum-Anforderung dafür, dass man wirklich alles automatisch umschalten kann und eine qualifizierte Entscheidung dahintersteht. Was die Verfügbarkeit für die Betriebe selbst betrifft, muss diese auf Basis der Kosten getroffen werden, die entstehen, wenn das System zum Stillstand kommt. Man sollte eine einfache Kosten-Nutzen-Rechnung aufstellen, eine Risiko-Abschätzung vornehmen, herausfinden, was es bedeutet, wenn das System eine Stunde, mehrere Stunden oder einen Tag lang steht, und diesem die Kosten für die Hochverfügbarkeit gegenüberstellen. So stellt sich meist relativ schnell heraus, dass eine Summe von 20.000 bis 30.000 Euro schnell erreicht wird. Eine Ausgabe für die nächsten drei bis fünf Jahre ist dann eine gut angelegte Investition. Dies ist das Minimum! Ausbaustufen gibt es darüber hinaus in großer Anzahl. Man sieht ja auch, wenn man Oracle mal aus dem Fokus herausnimmt, dass die Verfügbarkeit schon massiv aufgrund der Virtualisierungswelle gestiegen ist. Man verbindet die Server zu einem VM-Cluster und kann dann die Maschinen hin- und herschieben, wenn der Server ausfällt

oder in die Wartung geht. Damit hat sich die Hochverfügbarkeit bereits sehr schnell etabliert. Was Oracle und VMware betrifft, ist das eine komplett eigene Thematik aufgrund der Lizenzierungssituation. Daher setzen wir bei vielen Kunden auf Oracle VM und können so schnell und sehr günstig Hochverfügbarkeit herstellen. Im einfachsten Fall sind das Ausfallsysteme mit zwei Servern, wobei der zweite Server im Prinzip nichts oder nur geringfügig etwas tut. Dies ist sozusagen die Minimal-Konfiguration und ein mögliches Konzept.

Was ist bei Datenbanken hinsichtlich der Hochverfügbarkeit zu beachten?

Daniel Hafner: Das ist ein ganz wichtiges Thema. Die Datenbank stellt ja meistens ein Kernstück dar. Das heißt, dass viele Faktoren stimmen müssen, damit die Datenbank überhaupt funktional für die Applikationen sinnvoll einen Service bieten kann.

Natürlich muss die Datenbank funktionieren, sie muss Datenspeicherung zulassen, die Performance muss dementsprechend in Ordnung sein und die Geschwindigkeit zur Datenbank muss schnell genug sein. Im Bereich Cloud ist es ein großes Thema, dass die Daten innerhalb kurzer Zeit schnell wiederhergestellt werden können. Die besten Systeme im Umfeld rundherum helfen nichts, wenn beispielsweise ein User versehentlich eine Tabelle löscht oder Daten unbrauchbar macht. Es muss Konzepte geben, die die Integrität der Daten gewährleisten beziehungsweise die einen schnellen Wiederherstellungszeitpunkt haben – teilweise automatisiert und teilweise nicht automatisiert, je nachdem, wie es sich ein Kunde wünscht. Dies geht weit hinaus über das alleinige Zurverfügungstellen der Datenbank selbst.

Klaus-Michael Hatzinger: Für mich ist noch die Wiederanlauffähigkeit der Applikation im Falle eines Ausfalls ein wesentlicher Aspekt. Das heißt, dass man vielleicht einen Stillstand der Datenbank für nur zwei Minuten hat, die Applikationen damit jedoch nicht zurechtkommen und beispielsweise Batch-Jobs zurückfahren und wieder von vorne anfangen. Am Ende hat man tatsächlich einen Ausfall des Systems für drei bis vier Stunden, obwohl die Datenbank selbst nur zwei Minuten stillstand.

Daniel Hafner: Das ist ein ganz wichtiger Punkt! Hochverfügbarkeit auf der Datenbank heißt nicht unbedingt Fehlertoleranz der Applikationen. Es muss eine Kombination aus beidem sein. Es kommt sehr oft bei Kunden vor, dass die vorhandenen Oracle-Systeme eigentlich wieder sehr schnell verfügbar sind, aber die Auswirkungen eines kurzen Ausfalls unter Umständen bei den Applikationen zu langen Anlaufzeiten führen. Man sieht sehr oft, dass viele Applikationen nicht einmal HA-ready sind. Bei Web-Applikationen macht man sich weniger Sorgen, weil diese meis-

tens mit Connection-Pools arbeiten, die den Wiederanlauf von Hause aus managen können. Fat-Client-Applikationen tun sich hingegen jedoch meistens sehr schwer. Es gibt Technologien wie Transparent Application Failover (TAF) und Fast Application Notification (FAN) von Oracle, die versuchen, dieser Tatsache irgendwie zu begegnen. Dies ist allerdings bei den wenigsten Applikationen implementiert. Allein die Session-Konsistenz an sich muss wiederhergestellt werden. Das ist gar nicht so trivial, wie es klingt, und deswegen umgehen es die meisten Applikationen und lassen die Implementierung gänzlich weg.

Was sind konkrete Maßnahmen zur Realisierung von hochverfügbaren Datenbanken?

Daniel Hafner: Als konkrete Maßnahmen betrachte ich folgendes Vorgehen. Als ersten Schritt sehe ich die Frage danach, wie viel Hochverfügbarkeit ich überhaupt benötige. In einem zweiten Schritt sollte ein Konzept erstellt werden. Dies können unterschiedliche Konzepte sein, etwa Aktiv-Passiv-Systeme (eine Datenbank schwenkt einfach zwischen Servern hin und her), Aktiv-Aktiv-Systeme (in diesem Fall RAC) oder Desaster-tolerante Systeme wie Oracle Data Guard beziehungsweise eine Kombination aus allem. Die Mutter des Ganzen wäre ein hochverfügbares, Desaster-tolerantes System eventuell durch eine logische Replikation, gelöst mit Oracle Golden Gate, einer Drittanbieterlösung oder einer Kombination aus allem, je nach Ansprüchen des Kunden. Die größte anzunehmende Konfiguration ist RAC, Data Guard auf einem Standort und dann nochmaliges Replizieren in einen komplett anderen Standort oder über Data Center – vielleicht auch interkontinental – mit einer logischen Replikation. Das ist zwar mit entsprechend hohen Kosten verbunden, aber so kann man sehr hohe Verfügbarkeiten wirklich garantieren. Dies gilt auch für die Architektur darunter sowie für die Integrität der Daten und ebenso für die Verfügbarkeit bei geplanten Wartungen. Hier verbinden sich maximale Flexibilität mit hoher Fehlertoleranz und System-Verfügbarkeit.

Oft implementieren wir (zwei) Minimal-Konstrukte. Diese bestehen aus einem Aktiv-Passiv-System, Oracle RAC und einem RAC mit einem Data Guard. Das sind zwei Klassiker, die man oft vorfindet. Die etwas größeren Kunden haben Oracle RAC und Data Guard kombiniert und über mehrere Data Center gespannt. Viel Größeres sieht man schon sehr selten.

Ist spezielle Software zur Sicherstellung von Hochverfügbarkeit notwendig?

Daniel Hafner: Dies hängt natürlich vom Konzept ab und davon, auf welcher Ebene ich die Hochverfügbarkeit zur Verfügung stelle.

Was ist bei der Hardware-Architektur zu beachten?

Daniel Hafner: Bei der Hardware-Architektur ist erst einmal zu beachten, welche Server man einsetzt. Wie sind diese intern aufgebaut? Sind alle Peripherie-Komponenten bereits irgendwie redundant ausgelegt? Bei den namhaften Herstellern weiß man, dass alles intern schon über mehrere Lanes verbaut ist. Dies bekommt man relativ schnell mit, da die Systeme von Hause aus schon sehr fehlertolerant gebaut sind. Dann hängt es davon ab, wie wichtig es ist, dass ein Server am Leben bleibt. Im Storage-Bereich beispielsweise kann man auf die entsprechenden RAID-Konfigurationen eingehen. Bei den Netzwerk-Komponenten sollte man darauf achten, dass vielleicht nicht alles auf einer Netzwerkkarte läuft. Es kann auch passieren, dass Memory-Bereiche fehlerhaft werden. Man bekommt es oft gar nicht mit, wenn Datensätze komplett falsch heruntergeschrieben werden. Dies kann zu Daten-Inkonsistenzen in der Datenbank führen. Der etwas teurere Einsatz von eventuell doppelt gepufferten und checksummengeprüften DIMMs kann vor solchen Fehlern bewahren. Sonst kann man sich bei der Wahl der Hardware je nach Implementierung selbst überlegen, wie man vorgehen will.

Ebenfalls wichtig bei der Architektur ist, die Frage miteinzukalkulieren, ob die Performance leiden darf. Es ist oft so, dass man mit einer gewissen Backend-Performance kalkuliert. Wenn ein Pfad ausfällt, ist die Performance so stark beeinflusst, dass mit dem System eigentlich nicht mehr zu arbeiten ist. Bei Aktiv-Aktiv-Systemen sollte man immer auch betrachten, dass beim Ausfall einer Seite die verbleibende Seite Lasten bis zur doppelten Last aushalten muss. Die Architektur und die ganze Hardware-Größe sollten dementsprechend angepasst sein, sodass ein System zumindest das geplante Minimum bewältigen kann. Unsere Empfehlung ist immer, dass die Hardware so ausfallsicher sein sollte, dass sie auf jeden Fall maximal bis zu 50% ausgelastet ist, damit es nicht zu Performance-Problemen kommt. Daneben achten wir darauf, dass das Backend entsprechend gut ausgestattet ist. Wenn wir Desaster-Toleranz zum Backend herstellen, dann machen wir das über zwei, vier oder mehr Verbindungen, damit der potenzielle Impact möglichst gering ist.

Ist Hochverfügbarkeit teuer?

Daniel Hafner: Das ist von der Implementierung abhängig. Bisher war Oracle-Hochverfügbarkeit nicht unbedingt teuer. Wenn man einfach nur einen RAC haben wollte, brauchte man bis inklusive Oracle 18c nur eine Oracle Standard Edition mit 2 Lizenzen, zwei Server und ein externes Storage, und dann hatte man auch schon einen RAC dabei. Somit hielt sich das in einem akzeptablen Kostenrahmen. Aktuell hat sich das leider geändert, weil Oracle die RAC-Funktionalität ab Oracle 19c in der Standard Edition 2 zum Ärger aller Kunden nicht mehr unterstützt. Gehe ich aufgrund von höheren Verfügbarkeitsansprüchen in den Enterprise-Bereich, wird es wirklich teuer. Dann landet man schnell im sechsstelligen Euro-Bereich.

Klaus-Michael Hatzinger: Ich denke, man muss die Relation zu den Kosten sehen, die entstehen, wenn das System steht. Wenn es in einer Stunde Ausfallzeit zu einem Schaden von einer Million Euro kommt, ist das in Relation zur Investition nicht mehr so teuer.

Wie wird Hochverfügbarkeit in der Cloud sichergestellt? Was sollte man wissen?

Daniel Hafner: Ich konzentriere mich mal auf die Oracle Cloud. Oracle hat 3 Ebenen der Datenverfügbarkeit oder Sicherheit bereitgestellt. Zuerst die regionale Ebene — auch genannt „region domain" —, die eine Sammlung von Data Centers darstellt. Die nächste Stufe, die „availability domain", stellt dann das Data Center selbst dar. Und zum Schluss die „faulty domain". Dies ist eine Hardware-Gruppierung innerhalb eines Data Centers. Man kann sich dabei für eine individuelle Kombination aller drei Varianten entscheiden. Das bedeutet, dass wenn ich einen RAC aufbauen möchte, ich dies einfach nach den eigenen Wünschen

in unterschiedlichen Data Centern, in unterschiedlichen Brandabschnitten oder einfach nur auf unterschiedlichen Servern konfigurieren kann. Bezüglich der Datensicherheit kann ich dadurch insbesondere die Desaster-Toleranz beeinflussen. Hier möchte ich beispielsweise sicherstellen, dass die Maschine sich nicht im selben Brandabschnitt oder am selben Standort befindet, damit ich die Daten in irgendeiner Form auf jeden Fall gespiegelt habe. Auf alle diese Wünsche kann man in der Oracle Cloud Rücksicht nehmen und es lässt sich alles gut konfigurieren. So kann man seine Systeme dementsprechend aufbauen wie auch On-Premises. Ein Thema ist natürlich darüber hinaus die Hybrid-Beziehung, also eine Cloud- und On-Premises-Lösung und überhaupt der Weg zur Cloud mit all seinen Verbindungen. Die Frage ist: Was hoste ich alles in der Cloud? Ist es nur die Datenbank, muss ich mich ganz massiv mit den Latenzen auseinandersetzen. Ich muss erheben, dass die Leitungen dahinter passen und die entsprechende Redundanz bieten. Der Trend im Cloud-Geschäft geht dahin, dass die komplette Applikation in der Cloud liegt und man von außen darauf zugreift. Dies ist für sich eigentlich schon ein isoliertes System, eventuell mit einem Ausfallstandort On-Premises oder umgekehrt, sodass man primär einen On-Premises-Standort hat und einen Ausfallstandort in der Cloud. Dabei hängt es davon ab, um welche Art von Applikation es sich handelt. Ist es eine Web-Applikation, wird es wahrscheinlich komplett egal sein, ob diese lokal oder irgendwo in der Cloud liegt. US-Webseiten öffnen sich in der Regel in Europa fast genauso schnell. Ist es hingegen eine Fat-Client-Applikation, die sich im Hintergrund mit dem Web-Server austauscht, ist die Latenz wieder ein Thema. Da muss man wirklich darauf achten, dass sich die Data Center in der Nähe befinden. Die magische Distanz liegt immer bei etwa 100 - 150 Kilometern. Darüber hinaus leidet die Latenz aus physikalischen Gründen meistens deutlich. Man sieht dann sehr deutlich, dass bei allen Applikationen die Performance leidet. Als dritte Komponente hat man den Internet- oder Leitungsprovider. Bei den Business-Leitungen ist dies meistens weniger ein Thema, bei den Privatleitungen sind jedoch Ausfälle und Fehler eher zu erwarten. Hier könnte man zur Vermeidung parallel unterschiedliche Provider und unterschiedliche Leitungen nutzen, sofern das möglich ist.

Was erwarten Sie in der Zukunft von Oracle?

Daniel Hafner: Ich glaube, dass die große Erwartung von Oracle ist, dass man immer mehr in die Cloud geht. Oracle hat allerdings mittlerweile erkannt, dass das nicht ganz so einfach ist. Ich erwarte, dass wir in Zukunft sehr viele Hybrid-Lösungen implementieren können. Man hat einerseits On-Demand-Systeme in der Cloud, aber die wirklich wichtigen Produktionssysteme meistens trotzdem vor Ort.

Klaus-Michael Hatzinger: Vielleicht werden in Zukunft noch kleinere Cloud-at-Customer-Maschinen entwickelt.

Daniel Hafner: Dann würde Oracle den Markt wahrscheinlich schnell durchdringen können. Im Moment ist es noch viel zu groß skaliert. In Österreich kann man die Kunden an einer Hand abzählen, bei denen Cloud at Customer ein Thema sein könnte.

Klaus-Michael Hatzinger: Oracle hat auch schon viele Kunden verärgert oder verunsichert. Ich nenne als Beispiel die Themen VMware-Lizenzierung oder die Standard Edition. Mit Version 12.1.0.1 war plötzlich Endstation für die Standard Edition One und Standard Edition. Die SE-Kunden mussten für das technisch gleiche oder etwas abgespeckte Produkt SE2 (ab 12.1.0.2) neue Lizenzbestimmungen akzeptieren, was dazu führte, dass sie zwar weiterhin das Gleiche bezahlten, aber dafür in Zukunft technische Einschränkungen hinnehmen mussten (Reduzierung Sockellimit, Thread Cap Limit, RAC nur auf 2 Sockel). Jetzt mit Oracle 19c haben wir brandaktuell das Thema, dass Oracle den RAC gänzlich aus der SE2 einfach gestrichen hat. Solche Maßnahmen seitens Oracle machen Oracle sicher nicht zum Anbieter des Vertrauens. Die Unberechenbarkeit von Oracle führt dazu, dass die Kunden mehr und mehr das Vertrauen in die Kontinuität der Lizenzgewährung des Herstellers verlieren. Ich hoffe, dass Oracle das Vertrauen der Kunden wiedererlangen kann, denn wer will sich schon in die Cloud eines Herstellers begeben, der immer wieder die Grundlagen der von Kunden bereits getroffenen Investitionen verändert.

Zur Person: **Ing. Daniel Hafner**

Ing. Daniel Hafner startete seine Karriere nach Abschluss der HTL Matura im Jahr 2008 bei DBConcepts GmbH im Oracle Datenbank-Team. Er machte die Ausbildung zum Junior DBA und war schon innerhalb eines halben Jahres Oracle Certified Professional. Er führte seither zahlreiche Projekte im Oracle Infrastruktur-Umfeld durch und spezialisierte sich im Oracle Tuning-Umfeld sowie für Oracle Engineered Systems wie Exadata, Private Cloud Appliance und ODA. Seit Juni 2013 ist er Oracle Certified Master DBA und zählt heute zu den Besten seines Faches. Daniel ist Mastermind der DBConcepts internen Datenbank Monitoring Infrastruktur, mit der heute täglich über 1000 Oracle-Instanzen permanent überwacht werden. Mitte 2018 übernahm er die Abteilungsleitung und Koordination der gesamten Datenbanktruppe bei DBConcepts.

Infrastructure as Code: Mit Terraform Infrastructure as a Service in der Oracle Cloud nutzen

Jan Brosowski, Oracle

Automatisierung von System-Management-Prozessen ist bei Cloud-Infrastruktur-Projekten ein zentraler Baustein effizienten Managements. Eine große Anzahl von Tools buhlt in diesem Kontext um die Gunst der System-Administratoren. Ein „Platzhirsch“ im Bereich des Deployens von Infrastrukturen ist Terraform. Als ein typischer Vertreter der Tools im Cloud Native Development ist es auf eine Aufgabe fokussiert und nutzt sogenannte „Provider“, um diese Aufgabe in möglichst vielen Cloud- und Virtualisierungsumgebungen bereitzustellen.

Um den generellen Umgang mit Terraform im Zusammenspiel mit der Oracle Cloud Infrastructure (OCI) zu erklären, wird im Folgenden ein durchgängiges, einfaches Beispiel genutzt: Zwei virtuelle Maschinen sollen ausgerollt werden, die nach außen einen Webservice über einen Load Balancer anbieten. Zusätzlich sollen sie über einen Bastions-Host per SSH erreichbar sein *(siehe Abbildung 1)*.

Das Beispiel ist so zugeschnitten, dass Sie es mit einem Test-Account *[1]* in der Oracle-Cloud nachspielen können, ohne dass Ihnen weitere Kosten außer dem Zeiteinsatz entstehen. Für umfangreichere Tests wenden Sie sich bitte an Ihren Oracle Solution Engineer und fragen Sie ihn nach einem „SC Assisted Trial".

Bestandteile der Terraform-Umgebung

Terraform ist erstaunlich einfach aufgebaut: Alle notwendigen Komponenten laufen auf einem Administrationsrechner außerhalb der zu steuernden Cloud. Es benötigt keinerlei Agenten oder Ähnliches, es interagiert vollständig über REST APIs mit dem jeweiligen Cloud Service *[2]*.

In diesem Beispiel nutzen wir als Administrationsrechner eine aktuelle Oracle-Linux-7-Umgebung. Terraform ist aber auch für andere Betriebssysteme erhältlich, eine Liste findet sich auf der Herstellerseite.

Terraform nutzt zum Zugriff auf OCI den zugehörigen OCI-Provider, der beim ersten Zugriff auf die Oracle Cloud automatisch heruntergeladen und installiert wird. Somit bleibt nur die Installation von Terraform als Aufgabe des Administrators, die unter den gängigen Linux-Betriebssystemen mit dem jeweiligen Paketmanager erledigt wird.

Der Deploy-Vorgang wird über einen Satz von Konfigurations-Dateien gesteuert. Die in diesem Beispiel genutzten Dateien werden hier nur auszugsweise zitiert, sie können auf der Homepage der DOAG heruntergeladen werden. Zum Nachvollziehen der Erläuterungen in diesem Artikel empfiehlt es sich, dieses ZIP-File herunterzuladen, das enthaltene Verzeichnis zu entpacken und die jeweils angesprochene Datei zu öffnen.

Variablen und Daten in Terraform-Konfigurationsdateien

Typischerweise umfasst die Darstellung einer Infrastruktur aus mehreren Servern, Netzwerken, Regeln und Betriebssystemen sowohl eher statische als auch dynamische Elemente. Statische Elemente sind beispielsweise

- die Netzwerktopologie,
- die verwendeten Server oder auch
- die Firewall-Regeln zwischen den Komponenten.

Diese Elemente werden üblicherweise statisch, als während der Ausführung nicht veränderlicher Quellcode kodiert.

Dynamische Elemente kann man wiederum in zwei Klassen unterscheiden. Manche beschreiben Details der Umsetzung in einer bestimmten Cloud-Umgebung. Dies sind beispielsweise

- die Bezeichnung eines Compartments,
- die Zugangsdaten (Credentials), um die Umgebung einzurichten,
- die zu vergebenden Passworte oder SSH-Keys, um den Zugang zur Umgebung zu gewährleisten.

Diese Elemente werden üblicherweise als „Variablen" bezeichnet und können bei jeder Ausführung entweder als Variablen-Datei, als Environment-Variable oder als Kommandozeilen-Parameter übergeben

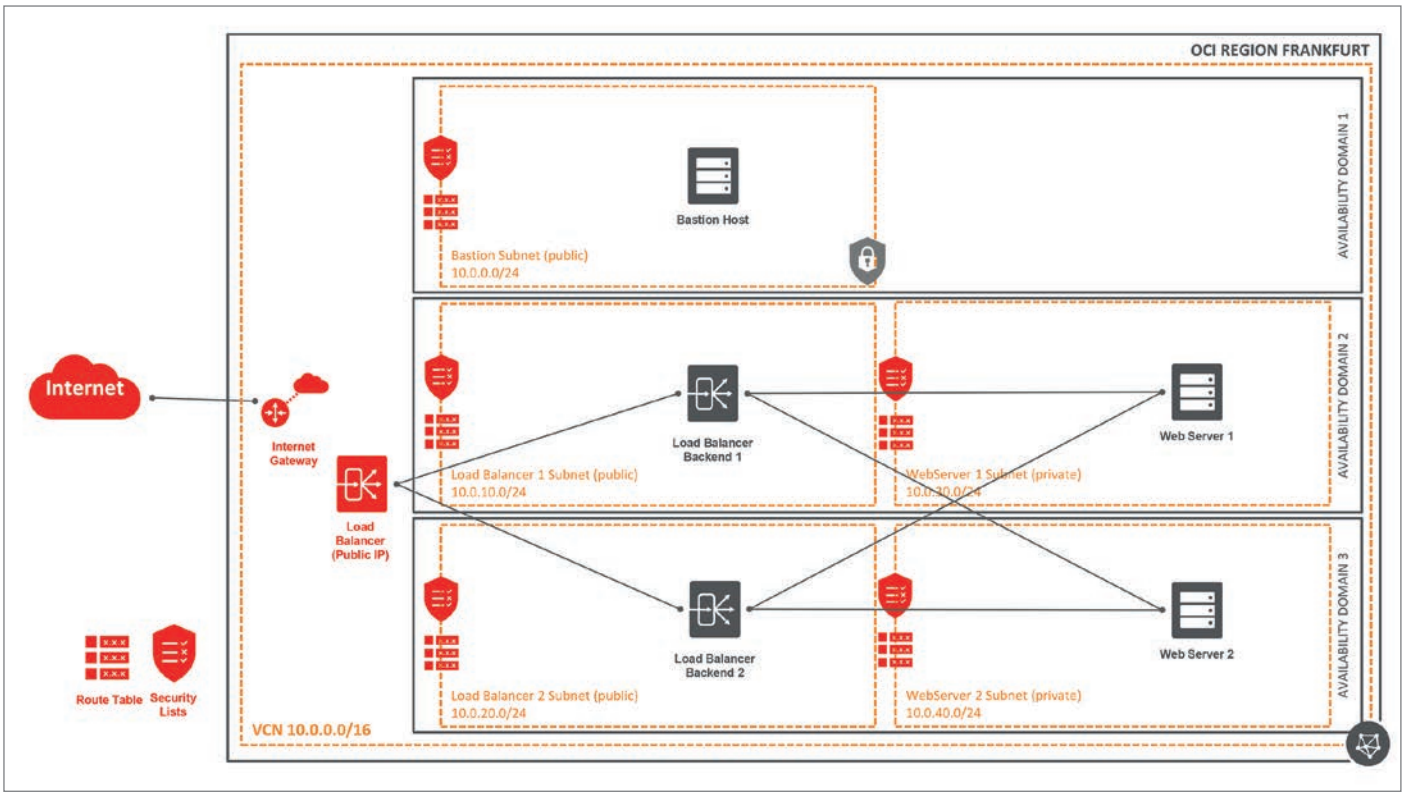


Abbildung 1: Infrastruktur-Übersicht des Beispiels (Quelle: Jan Brosowski)

werden. Im Beispiel sind Variablen noch mit Werten zu belegen. Sie sind in der Datei terraform.tfvars.TOBEMODIFIED zu finden.

Die zweite Klasse dynamischer Elemente sind solche, die bei jeder Umsetzung der im Code beschriebenen Infrastruktur unterschiedlich sein können und von der Cloud-Umgebung vorgegeben werden. Dies sind beispielsweise

- die OCID eines Image,
- die verwendeten öffentlichen IP-Adressen.

Solche Elemente werden üblicherweise als „Daten" bezeichnet, sie werden dynamisch während der Realisierung erzeugt und genutzt. Sie werden entweder in eigenen .tf-Dateien ermittelt oder dynamisch bei der Erzeugung von Ressourcen.

Im Folgenden werden die .tf-Dateien des Beispiels durchgegangen, um die verschiedenen Bestandteile zu besprechen.

Teil 1: Authentifizierung gegenüber der Cloud

Bei allen Cloud-Anbietern hat sich für den Zugriff mit Werkzeugen wie Terraform der Zugriff über APIs durchgesetzt. Um hier eine Authentifizierung zu ermöglichen, nutzt man ein Paar kryptographischer Schlüssel, die als „API-Key" bezeichnet werden. Der Code in der Datei 01auth.tf ist universell, alle für die Authentifizierung notwendigen Inhalte liegen in der Variablen-Datei vor.

Die eigentliche Authentifizierung ist, wie in *Listing 1* ersichtlich, relativ kurz.

Die zugehörigen Variablen *(siehe Listing 2)* finden sich in der Variablen-Datei terraform.tfvars. Die Variablen beinhalten eine UserID (user_ocid), den Pfad zu einem privaten Schlüssel, der zur Authentifizierung genutzt wird, und den Ort (region und compartment_ocid), an dem die Installation stattfinden soll. Wenn Sie das Beispiel nachvollziehen möchten, müssen diese Daten in der Oracle-Cloud-Management-Oberfläche zusammengestellt werden. Der Prozess, wie man diese Daten und Schlüssel erzeugt, ist unter dem Link *https://docs.cloud.oracle.com/iaas/Content/API/Concepts/apisigningkey.htm* dokumentiert *[3]*. Die so erhaltenen Daten trägt man in terraform.tfvars.ToBeModified ein und nennt die Datei dann um in terraform.tfvars. Somit ist der Zugriff auf die Oracle-Cloud möglich.

```
# ---- provider
provider "oci" {
   region             = "${var.region}"
   tenancy_ocid       = "${var.tenancy_ocid}"
   user_ocid          = "${var.user_ocid}"
   fingerprint        = "${var.fingerprint}"
   private_key_path   = "${var.private_key_path}"
}
```

Listing 1

```
tenancy_ocid     = "[...]"
user_ocid        = "[...]"
fingerprint      = "[...]"
private_key_path = "/path/to/private/key.pem"
compartment_ocid = "[...]"
region           = "eu-frankfurt-1"
```

Listing 2

```
resource "oci_core_virtual_network" "tf-redstack-vcn" { … }
```

Listing 3

```
resource "oci_core_virtual_network" "tf-redstack-vcn" {
   cidr_block      = "${var.cidr_vcn}"
   compartment_id  = "${var.compartment_ocid}"
   display_name    = "TF Redstack Demo VCN"
   dns_label       = "tfdemovcn"
}
```

Listing 4

Teil 2: Definition der notwendigen Netzwerke

Abbildung 1 zeigt die notwendigen Netzwerkobjekte:

- Zunächst ist ein VCN (Virtual Cloud Network) notwendig, in dem alle Komponenten verbunden sind, wobei sie sich in unterschiedlichen Subnetzen befinden. Man sieht, dass sich das VCN über alle Availabililty Domains (AD) erstreckt.
- Dann gibt es fünf Subnetze, die jeweils in einer AD präsent sind. Zwischen diesen Subnetzen gibt es Routing-Regeln und Security Lists.
- Es existiert ein Gateway ins öffentliche Internet, hinter dem zum einen der Bastions-Host erreichbar ist, aber auch die redundanten Load Balancer, die die Arbeitslast auf die Webserver verteilen.

Die Definitionen all dieser Ressourcen sind in der Datei 02_vcn.tf zu finden. Die Definitionen laufen alle nach einem Muster ab, das auch in den nächsten Teilen des Deployments wiederkehrt.

Ein Block startet mit dem Wort resource, gefolgt vom Typ der Ressource sowie einem Namen für die Ressource *(siehe Listing 3)*.

In diesem Beispiel wird also das virtuelle Core-Netzwerk angelegt, in dem alle Subnets des Beispiels beheimatet sein werden. Es trägt den Namen tf-redstack-vcn *(siehe Listing 4)*. Innerhalb der geschweiften Klammern werden dann Argumente dieses Netzwerks definiert.

Zwei der Argumente werden mit Variablen aus der tfvars-Datei gefüllt, während zwei andere mit Strings belegt sind. Welche Ressource über welche Argumente verfügt, kann man der Dokumentation des OCI-Providers *[4]* entnehmen.

Bestimmte Ressourcen hängen auch von anderen Ressourcen ab. So wird die Ressource oci_core_subnet einem VCN zugeordnet. In dem Beispiel sind abhängige Ressourcen nacheinander angeordnet, was für bessere Lesbarkeit und Wartbarkeit des Codes sorgt *(siehe Listing 5)*.

Terraform kommt aber auch damit klar, wenn diese Ressourcen durcheinander definiert würden.

```
resource "oci_core_route_table" "tf-redstack-bastion-rt" {…}
resource "oci_core_security_list" "tf-redstack-bastion-sl" {…}

# ------ Create a public subnet for bastion
resource "oci_core_subnet" "tf-redstack-bastion-sn" { …
   compartment_id    = "${var.compartment_ocid}"
   vcn_id            = "${oci_core_virtual_network.tf-redstack-vcn.id}"
   route_table_id    = "${oci_core_route_table.tf-redstack-bastion-rt.id}"
   security_list_ids = ["${oci_core_security_list.tf-redstack-bastion-sl.id}"]
   dhcp_options_id   = "${oci_core_virtual_network.tf-redstack-vcn.default_dhcp_options_id}"
}
```

Listing 5

In den Beispieldateien sind neben diesen Ausschnitten alle übrigen Komponenten der Netzwerkumgebung beschrieben. Gerade die Security-Lists ermöglichen sehr fein-granulare Regeln. Diese würden ein eigenes Beispiel erfordern, daher werden sie hier ausgeklammert.

Teile 3-6: Webserver und Bastionsserver installieren

Teil 3: Ein geeignetes Linux-Image auswählen

In der Oracle-Cloud ist eine große Anzahl vorgefertigter Betriebssystem-Images verfügbar, die über OCIDs identifiziert werden. In der Dokumentation gibt es eine Übersicht über die Images und deren OCIDs an den verschiedenen Standorten *[5]*.

Die Images werden gepflegt, regelmäßig gibt es aktualisierte Versionen. Daher nutzt das Beispiel *(siehe Listing 6)* ein kurzes Skript, um das aktuellste Oracle Linux 7.6 Image zu suchen.

Die Abfrage ist zweigeteilt: Über die Compartment-ID sowie Strings für OS und Version wird die Suche auf Oracle Linux, die Version 7.6 sowie den verwendeten Standort eingeschränkt. Aus dieser Liste werden dann Images für GPU-unterstützte Bare-Metal-Shapes mithilfe des regulären Ausdrucks herausgefiltert. So bleibt eine Liste der Oracle Linux 7.6 Images übrig, aus denen im nächsten Schritt die ID des neusten Image ausgewählt wird.

```
data "oci_core_images" "ImageOCID-ol7" {
   compartment_id           = "${var.compartment_ocid}"
   operating_system         = "Oracle Linux"
   operating_system_version  = "7.6"

   # filter to avoid Oracle Linux 7.6 images for GPU
   filter {
       name    = "display_name"
       values  = ["^.*Oracle-Linux-7.6-[^G].*$"]
       regex   = true
   }
}
```

Listing 6

```
resource "oci_core_instance" "tf-redstack-websrv1" {
   availability_domain   = "${lookup(data.oci_identity_availability_domains.ADs.availability_domains[var.AD_
websrv1],"name")}"
   compartment_id        = "${var.compartment_ocid}"
   display_name          = "TF Redstack Demo web server #1"
   shape                 = "${var.websrv_instance_shape}"
   preserve_boot_volume  = "false"

   source_details {
       source_type    = "image"
       source_id      = "${lookup(data.oci_core_images.ImageOCID-ol7.images[0], "id")}"
   }

   create_vnic_details {
       subnet_id         = "${oci_core_subnet.tf-redstack-websrv1-sn.id}"
       hostname_label    = "websrv1"
       assign_public_ip  = "false"
   }

   metadata {
       ssh_authorized_keys   = "${file(var.websrv_ssh_public_key_file)}"
       user_data             = "${base64encode(file(var.websrv_bootstrap))}"
   }
}
```

Listing 7

Teil 4, 5 und 6: VMs erzeugen und installieren

Das Erzeugen und die Installation der VMs läuft immer nach dem gleichen Muster ab. Eine Shape-Größe wird definiert, in einer Availability-Domain instanziiert und dann mit einem Image betankt. Die Maschine erhält auch eine virtuelle Netzwerkkarte (VNIC), die mit einem zuvor erzeugten Subnetz verbunden wird. Exemplarisch hier das Erzeugen eines der Webserver *(siehe Listing 7)*. Wieder werden einige Argumente aus der Variablendatei übernommen und das herausgesuchte Image aus Teil 3 wird eingesetzt.

Der Abschnitt metadata individualisiert die Installation: Es wird ein SSH-Key übergeben, um den User opc zu identifizieren. Das verwendete SSH-Key-Paar muss vor dem Terraform-Lauf erzeugt werden, ein beigefügtes Skript namens „generate_ssh_keys.sh" unterstützt hierbei, indem es zwei Passwort-lose Key-Paare lokal erstellt.

Der Abschnitt user_data übergibt dann ein Bootstrap-File nach Cloud-Init-Spezifikation. Diese sind ebenfalls im Beispiel hinterlegt. Im Beispiel wird bei den Webservern Apache installiert sowie eine einfache Webseite angelegt, um später zu sehen, welchem der Webserver man vom Load Balancer zugewiesen wurde. Alternativ könnte man hier auch mit Tools wie Ansible oder Puppet arbeiten.

```
resource "oci_load_balancer" "tf-redstack-loadbal" {
   shape          = "${var.load_balancer_shape}"
   compartment_id  = "${var.compartment_ocid}"
   subnet_ids      = [
       "${oci_core_subnet.tf-redstack-loadbal-sn1.id}",
       "${oci_core_subnet.tf-redstack-loadbal-sn2.id}",
   ]
   display_name    = "TF Redstack Web LoadBal"
}
```

Listing 8

```
resource "oci_load_balancer_backendset" "tf-redstack-loadbal-bes1" {
   name              = "TF_RedStack_LoadBal_backendset"
   load_balancer_id  = "${oci_load_balancer.tf-redstack-loadbal.id}"
   policy            = "ROUND_ROBIN"
[…]
   }
}
```

Listing 9

```
resource "oci_load_balancer_backend" "tf-redstack-loadbal-be01" {
   load_balancer_id  = "${oci_load_balancer.tf-redstack-loadbal.id}"
   backendset_name   = "${oci_load_balancer_backendset.tf-redstack-
loadbal-bes1.name}"
   ip_address        = "${oci_core_instance.tf-redstack-websrv1.pri-
vate_ip}"
   port              = 80
[…]
}
```

Listing 10

```
resource "oci_load_balancer_listener" "tf-redstack-loadbal-listener1" {
   load_balancer_id         = "${oci_load_balancer.tf-redstack- load-
bal.id}"
   name                     = "tf_redstack_LoadBal_listener"
   default_backend_set_name  = "${oci_load_balancer_backendset.tf-red-
stack-loadbal-bes1.name}"
   port                     = 80
   protocol                 = "HTTP"
}
```

Listing 11

```
$ terraform init

Initializing provider plugins...
- Checking for available provider plugins on https://releases.
hashicorp.com...
- Downloading plugin for provider "oci" (3.19.0)...
- Downloading plugin for provider "local" (1.2.0)...
[...]
Terraform has been successfully initialized!
[...]
```

Listing 12

Teil 7: Load Balancer

Die Konfiguration des Load-Balancing-Service gestaltet sich einfach. Er besteht aus drei Ressourcen, die auf bereits definierte Netzwerke zugreifen.

Zunächst werden der Load Balancer definiert und seine Subnetze zugewiesen *(siehe Listing 8)*. Anschließend wird dem Load Balancer ein sogenanntes „Backendset" zugewiesen. Dieses besteht aus den beiden Backends, die den Zugriff auf die beiden Webserver erlauben. Im Backendset werden die Verteilregel (hier Round Robin) sowie eine Regel zur Überprüfung der angebundenen Server hinterlegt. *(siehe Listing 9)*. So wird sichergestellt, dass nur aktive Webserver auch Anfragen erhalten.

Im Folgenden dann die beiden Backend-Komponenten, deren Argumente die privaten IPs der Webserver sind *(siehe Listing 10)*.

Zuletzt wird noch definiert, auf welchem Port und mit welchem Protokoll der Load Balancer auf seiner Public IP lauschen soll *(siehe Listing 11)*:

Durchführung des Deploys und abschließende Tests

Die letzte .tf-Datei in diesem Beispiel, 08_output.tf, stellt die notwendigen Informa-

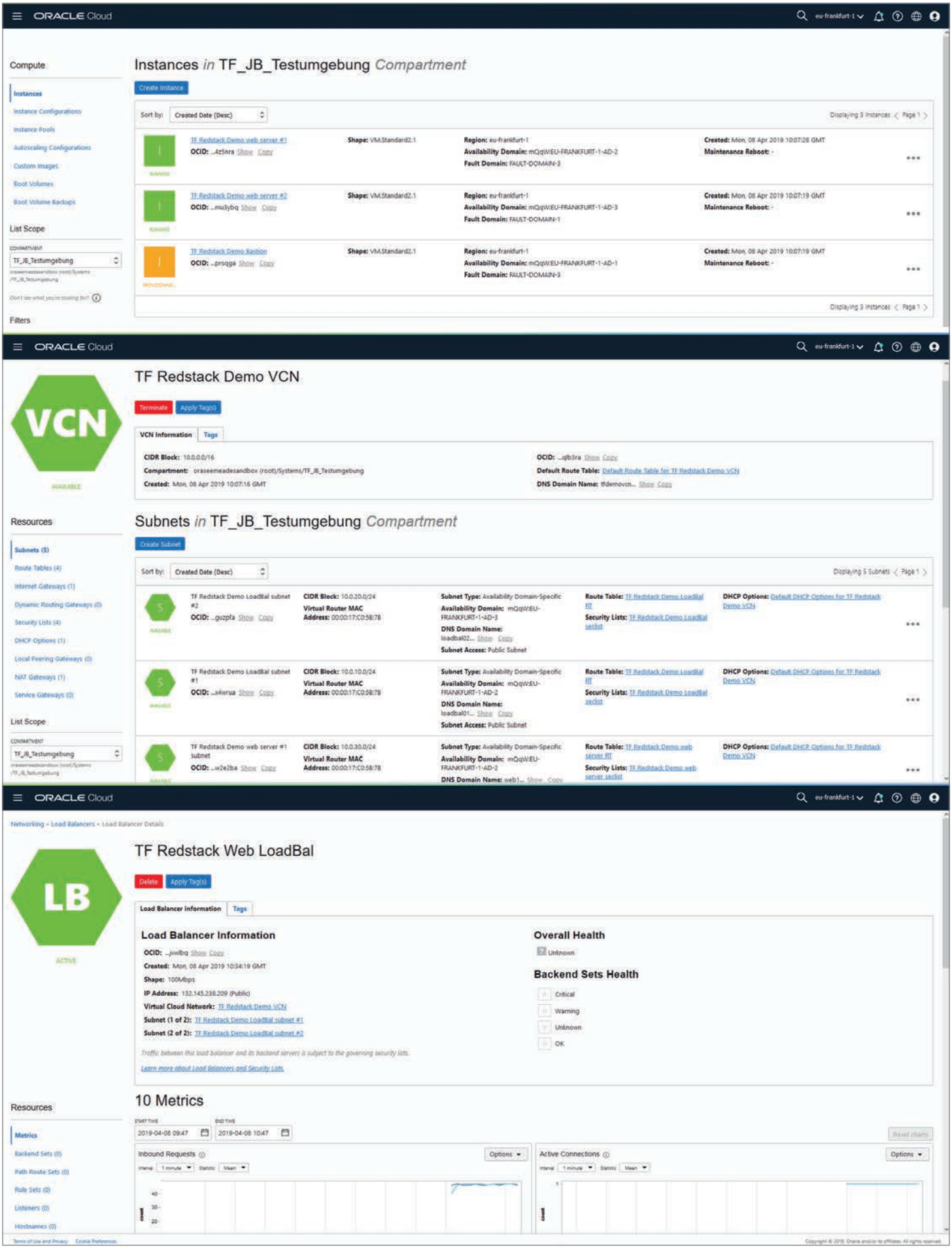


Abbildung 2: Exemplarische Screenshots während des Terraform apply. (Quelle: Jan Brosowski)

tionen bereit, um die Verbindungen zum Bastions-Host für den SSH-Zugriff sowie den Zugriff auf die Webserver zu testen. Dazu ermittelt sie die IP-Adressen der VMs und des Internet-Gateways und erzeugt eine entsprechende SSH-Konfig-Datei. Das wird der letzte Schritt des Deployments werden.

Um nun das Deployment durchzuführen, muss Terraform die Konfigurationsdateien einmal einlesen und das Verzeichnis initialisieren. Dazu dient der Befehl terraform init. Dabei wird ein Verzeichnis namens „.terraform" angelegt, in dem Informationen über die Umsetzung der .tf-Dateien abgelegt werden *(siehe Listing 12)*.

Anschließend kann man mit terraform plan die Umsetzung simulieren. Hier erhält man eine Übersicht darüber, welche Schritte Terraform umsetzen wird und wie viele Änderungen dazu notwendig sein werden. Bei der ersten Durchführung werden dies sehr viele sein. Ändert man später Details in den .tf-Dateien, werden entsprechend geringe Änderungen gefunden werden.

Das Deployen startet man mit terraform apply. Terraform sucht sich die notwendigen Daten zusammen und wird nachfragen, ob es wirklich die Änderungen durchführen soll (man bestätigt mit „yes"). Viele Operationen werden dabei parallelisiert; in der Cloud-Management-Oberfläche kann man zusehen, wie Netzwerke, Load Balancer und virtuelle Maschinen installiert werden (ähnlich *Abbildung 2*, dort wird eine der Instanzen noch installiert, während VCN, Load Balancer und 2 Instanzen schon fertig sind).

Einen exemplarischen, stark gekürzten Output des Prozesses finden Sie im folgenden *Listing 13*. Der Output schließt die Installation mit der Ausgabe der IP-Adressen und dem Erzeugen der SSH-Konfig-Datei (sshcfg) ab. Anschließend kann man die SSH-Verbindungen über den Bastions-Host sowie den Zugriff über den Load Balancer auf die Webserver testen *(siehe Abbildung 3)*. Dabei werden abwechselnd die Seiten des websrv1 und des websrv2 angezeigt

```
$ terraform apply
data.oci_core_images.ImageOCID-ol7: Refreshing state...
data.oci_identity_availability_domains.ADs: Refreshing state...

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:
[...]
Plan: 23 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Enter a value: yes

oci_core_virtual_network.tf-redstack-vcn: Creating...
[...]

================ You can SSH to the compute instances =

===== Using ssh commands with all parameters
bastion   : [...]
websrv1   : [...]
websrv2   : [...]

===== OR using the ssh aliases contained in the sshcfg file
ssh -F sshcfg bastion
ssh -F sshcfg websrv1
ssh -F sshcfg websrv2

================ Load Balancer URL for Web Servers: http://132.145.250.201
```

Listing 13

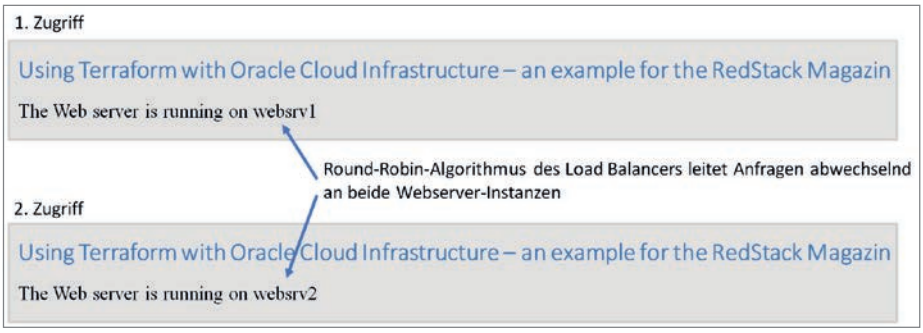


Abbildung 3: Zugriff auf die Webserver (Quelle: Jan Brosowski)

Zusammenfassung

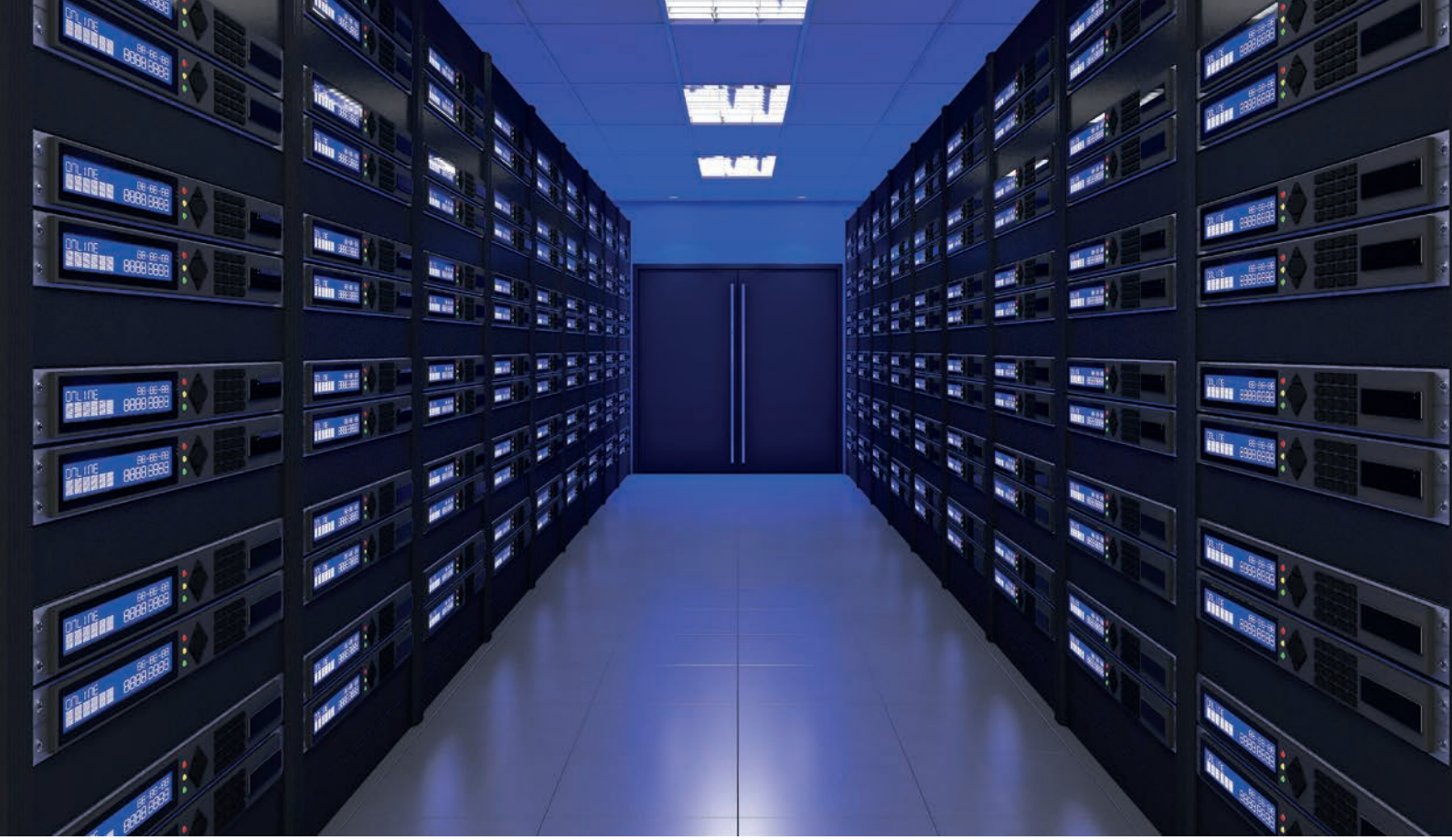
Das Beispiel zeigt, wie einfach mit Terraform eine Umgebung in der OCI ausgerollt werden kann. Es ist ein Einstieg, der einen ersten Blick auf die Möglichkeiten von Terraform erlaubt. Mit wenigen Ergänzungen könnte man beispielsweise noch ein Datenbank-Backend für die Webserver aufbauen oder die Installation mit Ansible und Puppet weiter automatisieren.

Genauso schnell, wie Sie die Umgebung aufbauen konnten, können Sie sie auch wieder einreißen: Der Befehl terraform destroy baut die gesamte Umgebung in etwa einer Minute wieder ab.

Quellen

[1] *https://cloud.oracle.com/tryit*
[2] *https://www.terraform.io/intro/index.html*
[3] *https://docs.cloud.oracle.com/iaas/Content/API/Concepts/apisigningkey.htm*
[4] *https://www.terraform.io/docs/providers/oci/index.html*
[5] *https://docs.cloud.oracle.com/iaas/images*

Jan Brosowski
jan.brosowski@oracle.com



Angewandtes maschinelles Lernen zum automatisierten Management von Oracle-Datenbanken

Mark V. Scardina, Oracle

Rechenzentren wachsen seit jeher, wenn auch nicht mehr im physischen Sinne, doch nach wie vor in ihrer Dichte. Verbesserungen in der Virtualisierung und Hardware haben dazu geführt, dass heute deutlich mehr Systeme betrieben und gemanagt werden als vor wenigen Jahren. Daher werden heute wesentlich mehr Administratoren benötigt.

Im Falle von Datenbanken haben neue Technologien wie die Oracle-Multitenant-Architektur dazu geführt, dass sich die Dichte verzehnfacht hat. Dies ging allerdings nicht mit einem vergleichbaren Anstieg bei der Anzahl der Administratoren einher. Stattdessen müssen sie heute erheblich mehr Datenbanken betreuen, die Effizenz muss also zunehmen.

Das führt zu modernen Strategien auf Basis von Machine Learning, einer Disziplin der Künstlichen Intelligenz, die Gartner als „AIOps“ bezeichnet. Auch wenn diese Abkürzung „Artificial Intelligence for Operations“ bedeutet, definieren die Autoren es mehr als „augmented intelligence“, als „erweiterte“ oder „verbesserte“ Intelligenz: Es geht darum, die Fähigkeiten und Effizienz von Administratoren zu erweitern und zu verbessern, sodass sie mehr Datenbanken bei strengeren SLAs betreiben können.

Oracle hat diesen Bedarf nach mehr Automation und effizientem Management großer Flotten von Datenbanken antizipiert und führte 2015 das Autonomous Health Framework ein. Seitdem

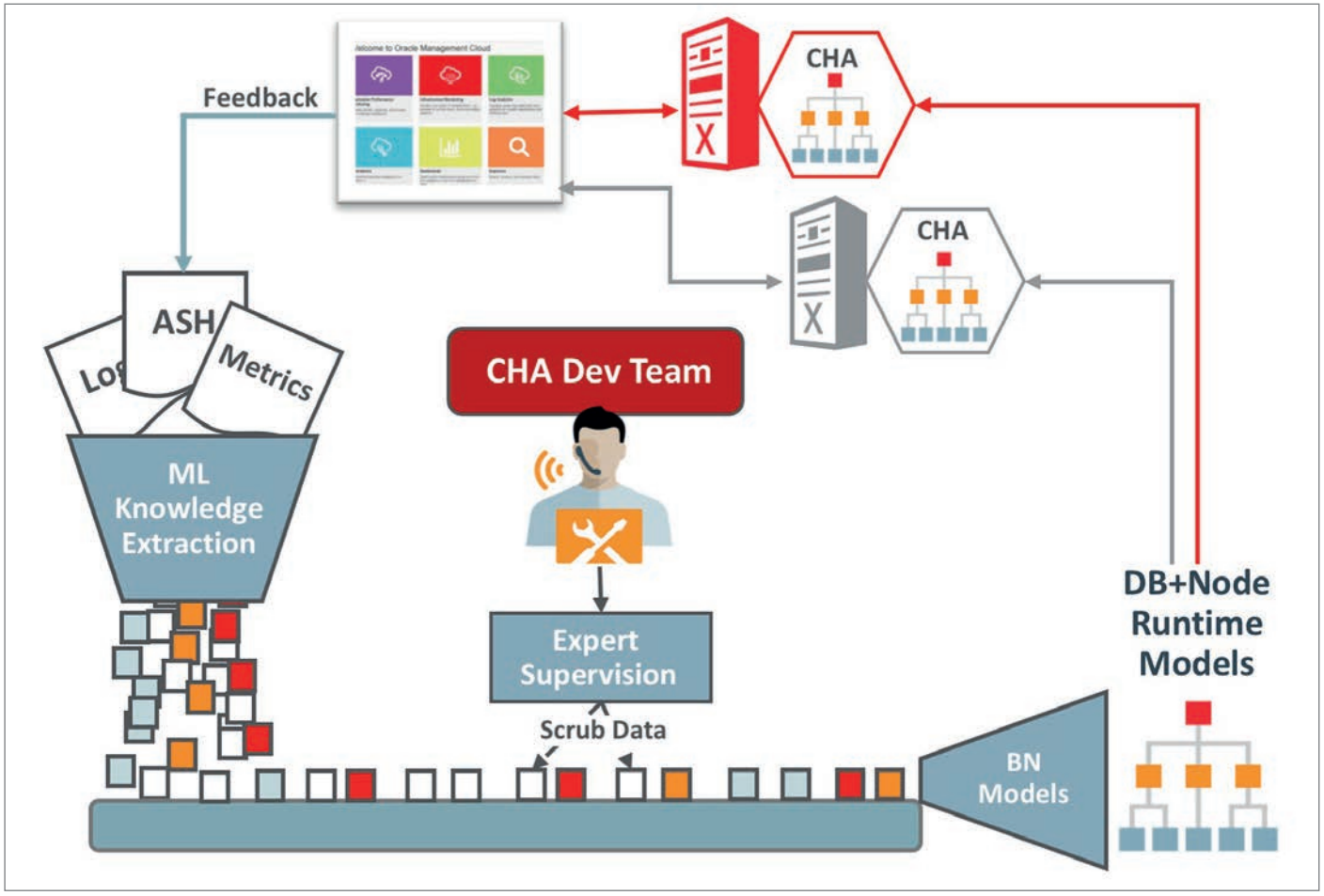


Abbildung 1: Applied Machine Learning Pipeline (Quelle: Mark Scardina)

wurde es kontinuierlich mit Methoden des angewandten maschinellen Lernens erweitert, um Performance zu sichern und schnelle Wiederherstellung im Fehlerfall zu gewährleisten.

Effizienzsteigerungen wurden traditionell durch Zentralisierung und Automatisierung erreicht. Doch gibt es Effekte, die einfache Tasks in großen Umgebungen schwieriger machen, gerade wenn Wartungsfenster immer kleiner werden. Außerdem ist gerade die Netzwerkbandbreite begrenzt, doch viele konventionelle Zentralisierungs-Ansätze führen eine große Menge an Daten, Log-Files und Trace-Files an einem zentralen Ort für Monitoring und Analysen zusammen. Dies ist weder zielführend noch effizient. Zudem sollten in sehr großen Umgebungen Log-Files und Trace-Files nicht aus ihren eigenen Umgebungen entnommen werden, da dies zwangsläufig zu Problemen beim Datenschutz führen kann.

Diese Überlegungen führten zu einem vollständig neuen Ansatz, der zunächst auf die Vermeidung von Problemen und Downtimes (Prävention) fokussiert war. Da es aber auch bei Nutzung von MAA (Maximum Availability Architecture) zu Fehlern kommen kann, wurde die schnelle Wiederherstellung („Rapid Recovery“) als ebenso wichtiges Feld erkannt. Der neue Ansatz muss sowohl umfassende Analysen als auch automatisierte Reaktionen umfassen; nicht zuletzt sind manuelle Tätigkeiten zu vermeiden.

Angewandtes maschinelles Lernen

Aus diesen Gründen hat man sich bei der Entwicklung entschlossen, angewandtes maschinelles Lernen zu verwenden, um sowohl proaktive als auch reaktive Operationen effzient durchzuführen.

Abbildung 1 zeigt Oracles Prozess zur Generierung der notwendigen KI-Modelle. Generell benötigen Diagnosen in IT-Umgebungen eine Root-Cause-Analyse, um den korrekten Lösungsweg zu empfehlen. Einfach nur Cluster in Daten zu identifizieren, reicht vielleicht bei Systemen zur Produktempfehlung in Webshops, aber bei der Unterstützung von operationalen Tätigkeiten bedeutet eine Korrelation nicht unbedingt auch einen Kausalzusammenhang. Deswegen bewerten Experten in einem überwachten Lernprozess die Ergebnisse und überwachen sowohl das Training-Set als auch die resultierenden Modelle. Das maschinelle Lernen findet dabei in der Entwicklung statt, es basiert auf Daten von Oracles SaaS und Cloud-Implementierungen, ebenso wie auf Daten ausgewählter Kunden. Das bedeutet, dass die aufwendige Modellierung bei Oracle stattfindet, während nur fertige Modelle und Wissensbanken im späteren praktischen Einsatz verwendet werden. Dies stellt schnelle Reaktionen mit minimalem Overhead im alltäglichen Einsatz sicher. Alle so erstellten Modelle verfügen über entsprechende Feedback-Mechanismen, um die Qualität der Modelle regelmäßig zu verbessen.

Es ist wichtig zu verstehen, dass die beiden Anwendungsfelder „Prävention“ und „Rapid Recovery“ unterschiedliche Ausgangsdaten verwenden und dass dies zum Einsatz unterschiedlicher Algorith-

men innerhalb des maschinellen Lernens führt. Es gibt keinen „universellen Algorithmus“, der beide Bereiche abdecken kann.

Im Bereich Prävention sind vor allem Echtzeitdaten und -metriken die Ausgangsdaten, beispielsweise aus dem Betriebssystem, der Datenbank, der zugrunde liegenden Hardware.

Es geht darum, in diesen Daten in Echtzeit Auffälligkeiten zu indentifizieren und darauf zu reagieren, um die Verfügbarkeit der Datenbank sicherzustellen. Dabei sind Verfahren wie Mustererkennung, autoassoziative multivariate Regression, Filter unter Berücksichtigung bedingter Wahrscheinlichkeiten und ähnliche Verfahren für Diagnosen und Prognosen im Einsatz.

Für Rapid Recovery sind Log- und Trace-Files wichtige Daten für die Analyse, da man in ihnen anormale Events und Muster von Ereignissen (sogenannte Signaturen) finden kann. Auf Basis dieser Signaturen kann man Incidents und Probleme erkennen und entsprechende Gegenmaßnahmen ableiten, um schnell die Verfügbarkeit wiederherzustellen. Typische Methoden sind hierbei K-nearest Neighbor, TF-IDF (term frequency-inverse document frequency) zur Vorhersage und Mustererkennung, sequenzielles Muster-Mining und Entscheidungsbäume zur Abweichungserkennung sowie LSTM und RNN für Prognose und Diagnose.

Anwendung in der Real-time Prevention

Wen man nach einer ganzheitlichen Lösung sucht, um proaktiv Performance und Verfügbarkeit sicherzustellen, denken Benutzer üblicherweise an Aspekte, mit denen man die eigene Session bewerten könnte: Laufen alle Transaktionen so wie gewünscht? Allerdings ist dieser Ansatz kurzsichtig: Um die Benutzer-Sessions zu sichern, muss die Lösung die zugrunde liegende Datenbank-Instanz sowie zugehörige Infrastruktur überwachen und schützen: Server, Netzwerk und Storage. Daher betrachten die nachfolgenden Use Cases diese drei Elemente.

Zunächst ein einfacher Use Case: Eine Datenbank-Session hängt und blockiert kritische Ressourcen, sodass andere Sessions ebenfalls blockiert werden.

Die Maßnahmen zur Vermeidung solcher Szenarien sind direkt in den Code der Datenbank integriert worden: Erster Bestandteil ist ein Modell normal ablaufender Sessions und darauf basierend Modelle zur Erkennung hängender Sessions. Diese Modelle wurden im Labor entwickelt, mit dem Ziel einer Runtime Engine, die effizent und effektiv hängende Sessions erkennen kann, ohne dass ein Datenbankadministrator benachrichtigt werden muss, der dann die Session aufwendig händisch erkennen und beenden muss.

Diese Engine ist als Code implementiert, der „Hang Manager“ genannt wird und im DAIO-Hintergrundprozess mitläuft. Er macht Snapshots des Zustandes der verschiedenen Sessions und vergleicht über die Zeit, wie sich diese weiter entwickeln. Wenn das Verhalten auf eine hängende Session hindeutet, wird ein Abhängigkeits-Baum aufgebaut, um die ursprünglich blockierende Session zu identifizieren. Wenn diese Root Session erkannt wurde, wird mit Modellen geprüft, ob auch diese hängt. Ist das der Fall, wird sie beendet. Schlägt das fehl, wird der zugehörige Schattenprozess ebenfalls beendet. Dies passiert alles vollautomatisch in weniger als zwei Minuten, ohne dass sein Administrator eingreifen muss. Die Tätigkeit des Hang Manager wird im Alert Log dokumentiert, ein Trace-File wird angelegt. Diese Funktionalität ist seit Version 11gR2 standardmäßig aktiv in der Datenbank und wird seitdem kontinuierlich weiterentwickelt.

Im zweiten Beispiel gehen wir von einer Datenbank-Instanz aus, deren Performance wegen ausgelasteter Ressourcen nicht zufriedenstellend ist.

Für dieses Szenario wurde mittels angewandten maschinellen Lernens ein Modell gängiger RAC-Datenbank-Probleme in konsolidierten Umgebungen erstellt. Weitere Modelle, die auf normale Betriebszustände kalibriert wurden, wer-

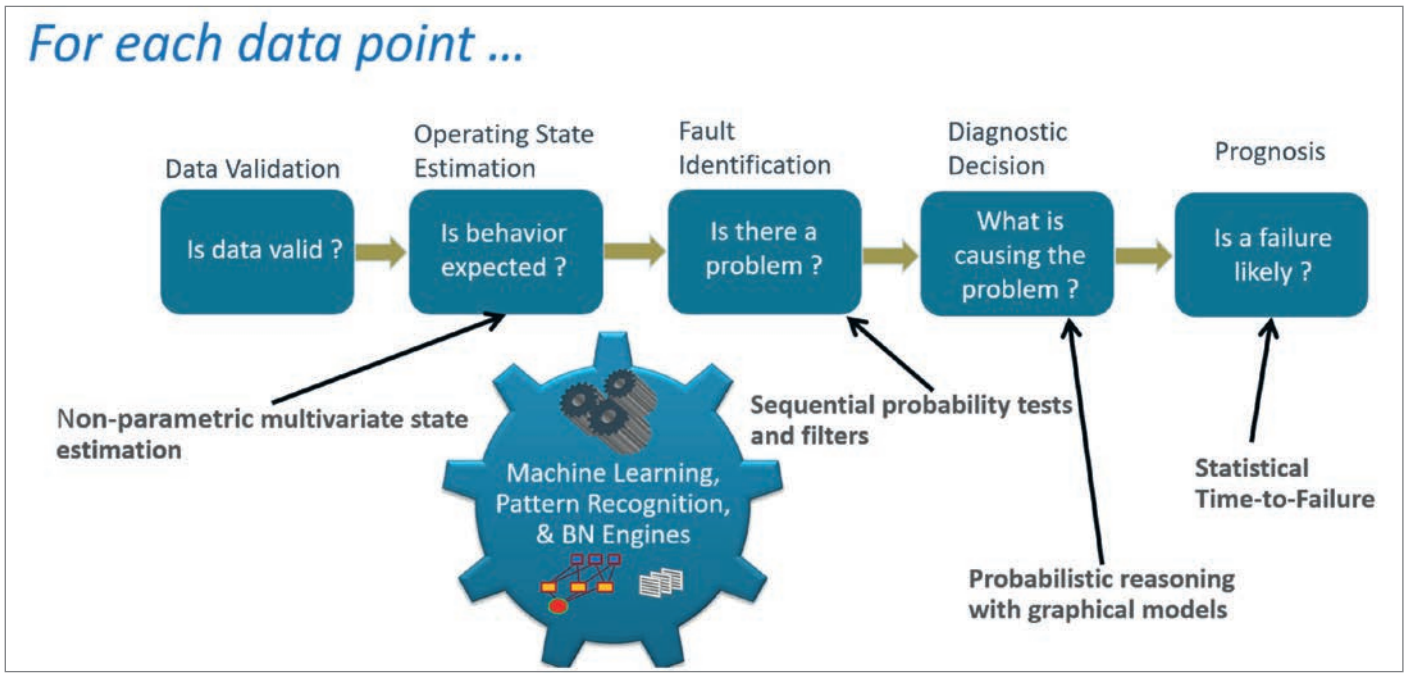


Abbildung 2: Cluster Health Advisor Runtime ML Pipeline (Quelle: Mark Scardina)

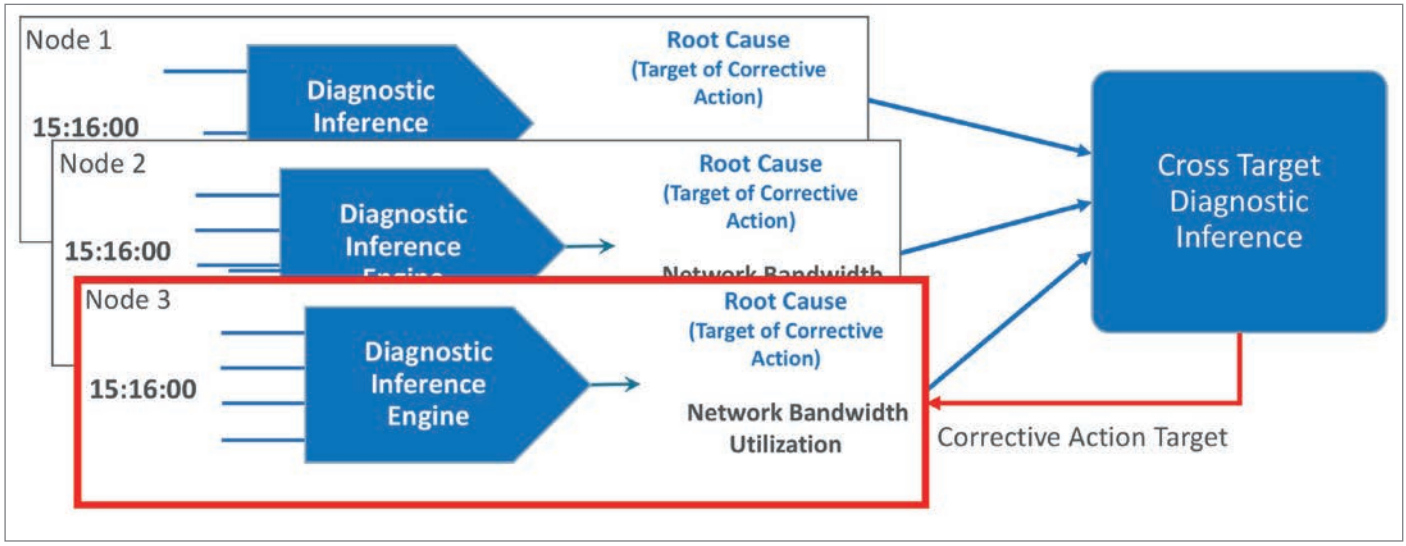


Abbildung 3: Cross Target Diagnostic Inference (Quelle: Mark Scardina)

den genutzt, um früh entstehende Abweichungen zu erkennen. Auf Basis dieser Abweichungen werden dann Situationsspezifische Bayes'sche Netze verwendet, um eine Root-Cause-Analyse durchzuführen und notwendige präventive Maßnahmen zu ergreifen. Das Training dieser Netze fand mit Daten der Oracle Cloud, Oracle SaaS und ausgewählter Kundenumgebungen statt. Die trainierten Modelle werden dann kompiliert eingesetzt für die Echtzeit-Prognose und -Diagnose.

Im Einsatz wurden die Umgebungsdaten von einem schlanken Prozess gesammelt, der auf jedem Knoten eines RAC-Clusters Datenbank- und Betriebssystemdaten aus dem Hauptspeicher ausliest. Im Falle des Betriebssystems wurden die Daten aus dem Cluster-Health-Monitor-Prozess ausgelesen, der diese Daten sowieso bereits erfasst. Im Falle der Datenbank wurden die Monitoring-Daten aus einem Bereich des Shared Memory ausgelesen, in dem Hintergrund-Prozesse bereits Metriken, Wait Events etc. ablegen. Die Daten werden in Intervallen von 5 Sekunden gelesen und ausgewertet, um frühzeitig Signale entstehender Probleme zu erkennen. Sollten diese auftreten, wird der oben beschriebene Prozess zur Root-Cause-Analyse gestartet, Gegenmaßnahmen eingeleitet und die Administratoren alarmiert.

Die dafür notwendige Prozess-Pipeline *(siehe Abbildung 2)* startet mit einem Vektor von 150 Betriebssystem- und Datenbank-Metriken, die in einem 5-Sekunden-Raster mit einer Auflösung von einer Sekunde synchron erhoben werden. Jeder der Vektoren wird zunächst validiert, dann wird geprüft, ob die Werte in einem zu erwartenden Verhältnis zueinanderstehen oder ob es erste Signale gibt, die auf ein Problem hindeuten. Werden Abweichungen erkannt, werden eine weitere Analyse angestoßen und auf Basis der Bayes'schen Entscheidungsnetze Maßnahmen getroffen. Bei bestimmten Problemen, beispielsweise wenn der Ressourcen-Verbrauch relativ hoch ist, kann eine Prognose über die zu erwartende Time-to-Failure abgegeben werden.

Zusätzlich kann es im Umfeld von RAC-Umgebungen dazu kommen, dass die Ursache eines Problems eines Node auf einem anderen Node liegt. Mit der Version 18c wurde die Analyse um eine „Cross Node"-Komponente ausgedehnt, die in *Abbildung 3* dargestellt wird. Diese Komponente erlaubt es Administratoren, die erwähnten möglichen Abbhängigkeiten bei einer Analyse zu berücksichtigen. Die zugrunde liegenden Technologien sind im Cluster Health Advisor bereits seit 12cR2 enthalten.

Als nächstes Beispiel betrachten wir ein Performance-Problem, das von der Infrastruktur ausgeht: Eine ASM-Instanz sei hängen geblieben und blockiere den Datenbank-IO.

Auch hier greift der oben erwähnte Hang Manager, der auch für ASM-Instanzen entsprechend angepasst implementiert wurde. Dieser Hang Manager spricht allerdings zusätzlich permanent mit den angebundenen Datenbank-Instanzen. Sollte ein Prozess nun hängen bleiben, kann entweder die hängende Session oder der gesamte ASM-Prozess terminiert werden. Dies bedeutet nicht, dass die Datenbanken terminiert werden, da sie sich dank FlexASM zu anderen ASM-Instanzen verbinden können.

Ein weiteres häufiges Infrastruktur-Problem im Umfeld von Cloud und konsolidierten Umgebungen ist ein Performance-Einbruch wegen zu hoher Nutzung physischer Ressourcen.

Auch hier wird mit den oben beschriebenen Mechanismen der Mustererkennung gearbeitet: Modelle, die normale Betriebszustände abbilden, überwachen permanent die Metriken von Betriebssystem, Netzwerken und Storage. So werden entstehende Lastspitzen rechtzeitig erkannt und Administratoren informiert. Diese Features sind seit 12cR2 vorhanden und wurden mit 18c deutlich erweitert.

Rapid Recovery Use Cases

Nun ein Blick auf den zweiten Anwendungsbereich: Rapid Recovery, wenn es nach einem Event oder einem Incident einer schnellen Wiederherstellung bedarf. Der erste Use Case betrachtet hier den Fall, dass man schnell wichtige Events in Gigabytes von Log- und Trace-Files finden muss.

Um dieses Problem anzugehen, werden Methoden des maschinellen Lernens angewandt, um aus Log-Files ungewöhnliche Ereignisse herauszufiltern und diese in einer Timeline zu korrelieren und anzuordnen. Die ersten sieben Schritte (Training) der Pipeline in *Abbildung 4* werden

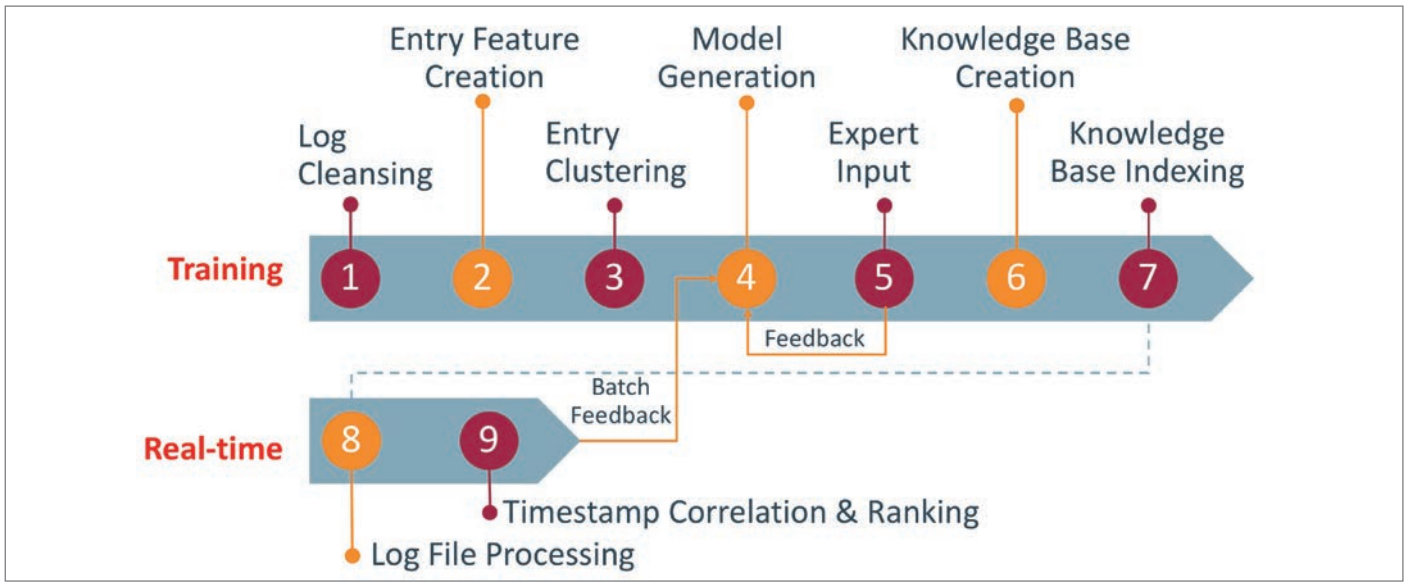


Abbildung 4: Trace File Analyzer ML Pipeline (Quelle: Mark Scardina)

auf Basis der Files durchgeführt, die sich über Jahre des Betriebs angesammelt haben. Als Resultat existieren eine Wissensbasis und ein Index, der dafür genutzt werden kann, in Echtzeit Anomalien und außergewöhnliche Ereignisse aus Log-Files herauszufiltern *(siehe Schritte 8 und 9 in Abbildung 4)*.

Zunächst werden dazu die Log-Files gesäubert und in ihre variablen und konstanten Anteile aufgespalten. Darauf aufbauend werden Merkmale identifiziert und extrahiert, die bei bestimmten Ereignissen auftreten. Dieses Auftreten wird dann geclustert, um die Bedeutung und Wichtigkeit zu bewerten und um letztlich Signaturen bestimmter Ereignisse abzuleiten. Diese Signaturen werden dann von Experten bewertet mit dem Ziel, Entscheidungsbäume zu erzeugen, die Anomalien über die Signaturen erkennen und klassifizieren können. Die Erkennung und Klassifizierung wird durch Feedback-Zyklen mit Trainingsdaten weiter verbessert. Erst dann werden Indizes der Muster für die spätere Erkennung generiert.

Im Echtzeit-Betrieb werden dann Log-Files permanent gegen diese Muster geprüft, um so anormale Ereignisse zu erkennen und diese schnell für weitere Diagnosen und die Lösung der Probleme zu klassifizieren.

Diese Methoden stehen im Trace File Analyzer 18c oder im Oracle Cloud Admin Dashboard für Analysen und Diagnose-Unterstützung zur Verfügung. Die

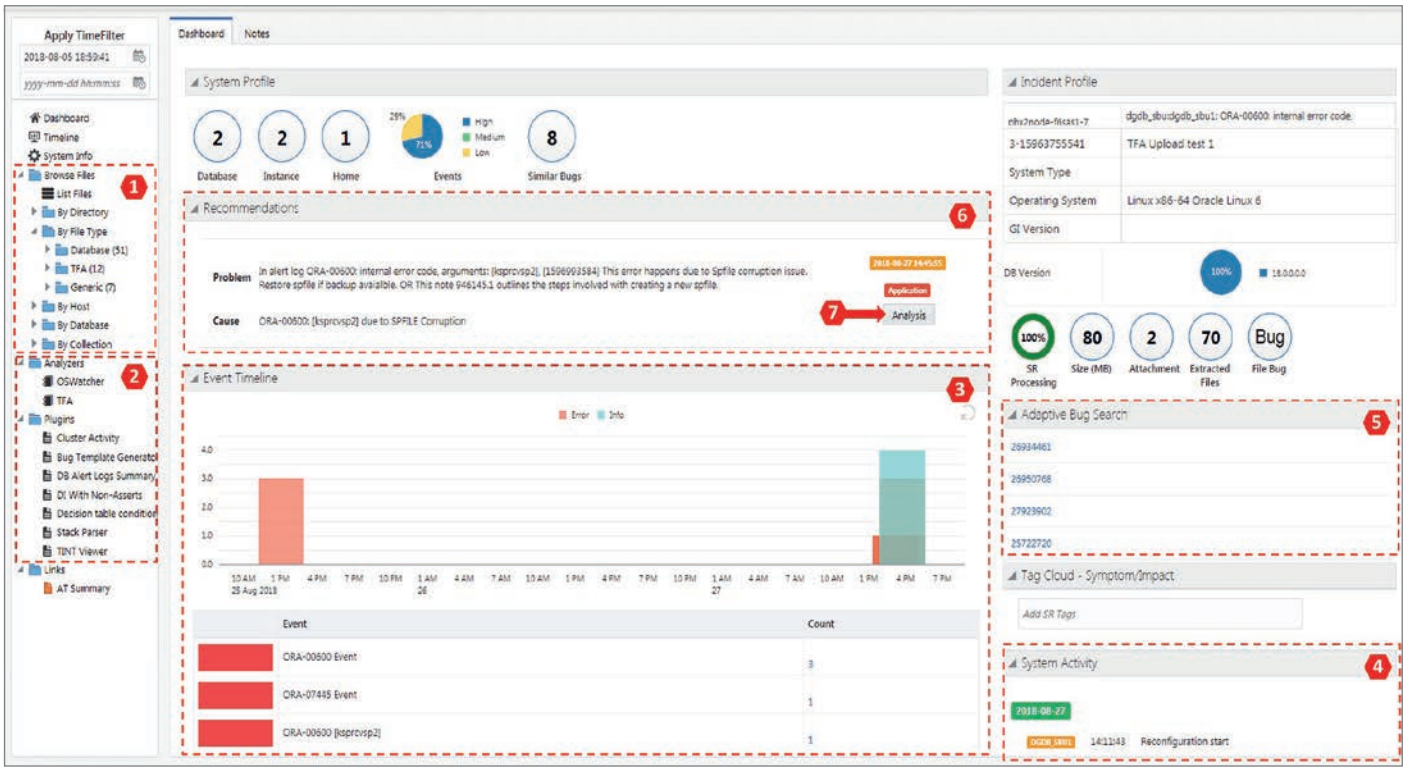


Abbildung 5: Cloud Administrator Incident Dashboard (Quelle: Mark Scardina)

Algorithmen laufen vollständig autonom, sobald der Smart Collector auf dem zu überwachenden System eine Alarmierung auslöst.

Viele Administratoren entwickeln im Laufe der Zeit eine Art Gespür, mit dem sie aufgrund von Einträgen in Log-Files Probleme vorhersehen können. Dies ist mehr als ein Gespür, es basiert auf Fakten, es handelt sich um typische Muster in Log-Files, die auf ein Problem hindeuten. Solche Muster aufzuspüren, ist der zweite Ansatzpunkt für maschinelles Lernen im Kontext von Log- und Trace-Files.

Indem viele Bugs und Service Requests mit Methoden des Data Mining untersucht wurden, konnten wiederholende Muster erkannt und so Zeitreihen sowie zeitliche Signaturen identifiziert werden, die auf das Vorhandensein bekannter Probleme hindeuten oder hinweisen. Diese Muster werden im Trace File Analyzer hinterlegt, sodass dort diese Muster gesucht und erkannt werden können. Dies erlaubt es, unter Umständen vor dem eigentlichen Auftreten eines Problems dieses bereits zu erkennen und korrigierende Maßnahmen zu empfehlen.

Der letzte Use Case zeigt Ihnen einen Anwendungsfall hinter den Kulissen beim Oracle Support: Es geht um die Effizienz, mit der innerhalb der Service-Request- und Bug-DB-Infrastruktur nach relevanten Bugs und Patches gesucht wird. Es kommt häufig vor, dass bestimmte Bugs oder Issues bei Kunden bereits in anderen Fällen gelöst wurden oder Patches schon bereitstehen. In diesen Fällen kommt es darauf an, schnell den jeweiligen Patch oder Workaround zu identifizieren. Liefert man ausreichende Input-Daten, beispielsweise aus speziellen Kollektoren, können schneller relevante Bugs oder ähnliche Service Requests identifiziert werden. Hier greifen die Methoden des Machine Learning, die kontinuierlich SRs sowie die BugDB durchforsten, um identische oder sehr ähnliche Bugs und Incidents zu identifizieren.

Dies wurde für mehr als 400 Oracle-Produkte implementiert, um mithilfe logistischer Regressionen potenzielle Duplikate zu identifizieren und den Support Engineers oder Cloud Admins direkten Zugriff auf diese zu geben. Der Prozess wird kontinuierlich durch das Feedback der Nutzer verbessert.

Die Ähnlichkeit von Bugs und Incidents wird dabei mit Merkmalen aus Error Stacks, Trace-Files, den Beschreibungen etc. abgeleitet. Aus diesen werden sogenannte „Issue Signatures" generiert, welche die Grundlage für Regressionsmodelle zur Bewertung unterschiedlicher Merkmale sind. Auf Basis des so gebildeten gewichteten Modells können dann die Ähnlichkeiten bewertet werden. Das Modell wird kontinuierlich auf Basis des Feedbacks der Benutzer weiterentwickelt.

Momentan sehen nicht nur Administratoren diese Funktionalitäten. Sie wurde auch Entwicklern mit Zugriff auf die Bug-DB zur Verfügung gestellt, um schneller geeignete Lösungen für Bugs zu finden und die Bearbeitungszeiten durch eine kollaborative Bearbeitung zu verkürzen. Ein Einsatz in weiteren Bereichen ist geplant.

Alles aus einem Guss

Abbildung 5 zeigt das Dashboard eines Cloud Admins, auf dem dieser über einen Incident informiert wird. Es wurde so entworfen, dass die Problemlösung schnell und sicher erfolgen kann. Dazu dienen verschiedene Sektionen auf dem Dashboard. In der Sektion 1 werden zunächst die Daten zusammengestellt, die analysiert wurden. Diese sind bereits indiziert und klassifiziert und werden in einer hierarchischen Ansicht mit Drill-Down-Funktionalität bereitgestellt.

Unterhalb davon, in der 2. Sektion, finden sich Diagnose- und Analyse-Plug-ins, die auf Basis der oben beschriebenen KI-Modelle auf die Dokumente in der Sektion 1 angewandt werden. Diese Plug-ins stellen gefundene, wichtige Ereignisse in Form einer Timeline in Sektion 3 dar, von wo man schell und einfach weitere Analysen und Aktionen starten kann.

Die gesammelten und als relevant eingestuften Log-File- und Trace-Informationen werden in Sektion 4 dargestellt, sodass hier nur Auffälligkeiten zu finden sind (und kein Grundrauschen). In Sektion 5 werden relevante, bekannte Bugs dargestellt, die zu den gefundenen Abnormalitäten passen.

Abschließend wird zentral in Sektion 6 das Ergebnis der Root-Cause-Analyse aufbereitet und dargestellt, inklusive der detaillierten, empfohlenen Lösungsschritte, die mit einem Klick durchgeführt werden können (7).

Zusammenfassend bleibt zu erwähnen, dass Oracles Anwendung maschinellen Lernens im Bereich des Operatings und der Administration von Analysten als sinnvoll und zielführend angesehen wird. Gartner *[1]* veröffentlichte beispielsweise eine klare Handlungsempfehlung für CIOs, Methoden der KI in allen Bereich zu nutzen.

Noch spezifischer sind die Empfehlungen der IDC *[2]:* Sie sehen die Anwendung von KI in Produkten wie der Autonomous Database als eine außerordentliche Verbesserung an, die Hunderte, wenn nicht gar Tausende von Stunden an administrativem Aufwand pro Jahr und Datenbank einsparen wird.

Mehr Informationen über das Oracle Autonomous Health Framework finden Sie auf der Webseite und im User's Guide: www.oracle.com/goto/AHF.

Quellen

[1] Predicts 2019: Artificial Intelligence Core Technologies - Chirag Dekate, et al, Gartner, 2019

[2] IDC PERSPECTIVE: Oracle's Autonomous Database: AI-Based Automation for Database Management and Operations - Carl W. Olofson & David Schubmehl, IDC

Mark V. Scardina
mark.scardina@oracle.com

APEX 19.1 ist da.

Was ist neu?

Carsten Czarski, Oracle

Mit dem Release 18, das letztes Jahr freigegeben wurde, hat auch Application Express den neuen Release-Zyklus und das neue Versions-Schema für Oracle-Produkte übernommen. Die Release-Nummer gibt nun das Erscheinungsjahr an – und das Ziel sind ein bis zwei Releases pro Jahr. Künftig sind also mehr Releases, als bisher zu erwarten (es gab etwa alle 18 Monate ein neues APEX Release) – dafür sind die einzelnen Releases kleiner und enthalten nicht mehr so viele neue Features.

Folgerichtig steht nun das Release 19.1 bereit. Neben einer modernisierten Oberfläche inklusive „Dark Mode" gibt es Neuerungen für das Data Loading, JET Charts und Interactive Grid. Weiterhin wird die in 18.1 eingeführte Unterstützung von externen Datenquellen (REST Enabled SQL, REST-Services) für Formulare ausgebaut, sodass Updates (DML) auf solche Datenquellen möglich werden.

Dark Mode

Schon beim ersten Anmelden am APEX Workspace erkennt man, dass man auf einer neuen Version arbeitet: Wie bei fast jeder APEX-Version wurde das Look & Feel modernisiert. Neu ist der Dark Mode, der aktiviert wird, wenn man zunächst auf das „User-Symbol" ganz oben rechts klickt *(siehe Abbildung 1)* und den neuen Dark Mode einschaltet. Die Seite lädt neu und sieht wie in *Abbildung 2* ersichtlich aus.

Insbesondere beim Arbeiten in dunklen Umgebungen hat sich der Dark Mode für viele Anwender als weniger anstrengend für die Augen erwiesen – zahlreiche Anwendungen bieten bereits einen Dark Mode an – und auch auf Betriebssystemen wie MacOS ist dieser verfügbar.

Natürlich betrifft der Dark Mode ausschließlich den Application Builder, also die Oberfläche, mit der man als Entwickler arbeitet. Endbenutzer merken nichts davon – das Look & Feel bestehender APEX-Anwendungen bleibt unverändert.

Für Universal-Theme-Anwendungen können Entwickler jedoch selbstverständlich einen neuen Theme Style anlegen, der einen Dark Mode implementiert.

Data Loading

Möchte man, als Entwickler, Daten in den APEX Workspace laden, so bietet sich der Bereich SQL Workshop > Data Workshop

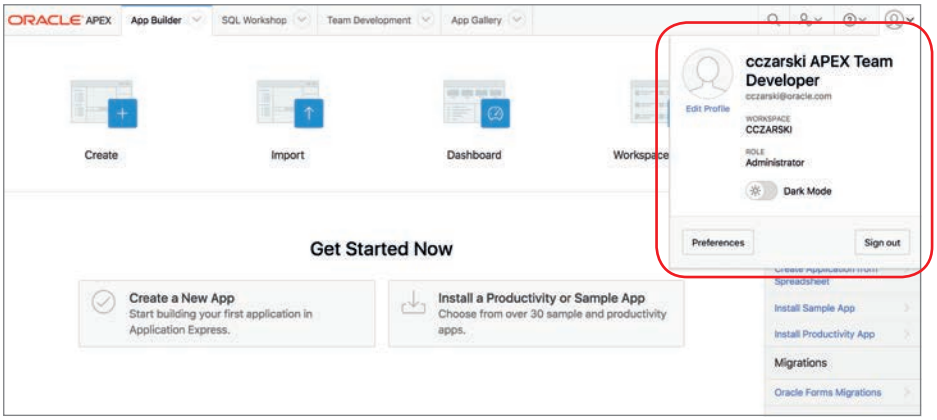


Abbildung 1: Application Express 19.1 (Quelle: Carsten Czarski)

an. Hier steht Funktionalität zum Upload und Download von Daten bereit.

Bislang unterstützte APEX hier nur CSV- und XML-Dateien – und XML-Dateien auch nur dann, wenn die XML-Tags ganz genau den Tabellenspalten einer existierenden Tabelle entsprechen. Dieser Bereich wurde in APEX 19.1 erheblich modernisiert: Die Oberfläche ist nicht nur wesentlich benutzerfreundlicher (Dateien können per Drag & Drop hochgeladen werden, Datentypen und Spalten werden automatisch erkannt), es werden nun neben XML- und CSV- auch JSON-Dateien und XLSX-Dateien (Spreadsheets) unterstützt *(siehe Abbildung 3)*.

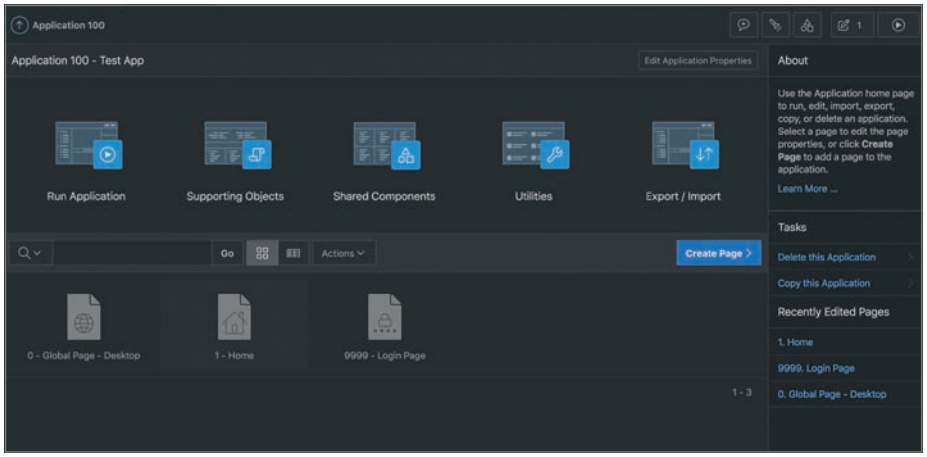


Abbildung 2: Dark Mode im APEX Application Builder (Quelle: Carsten Czarski)

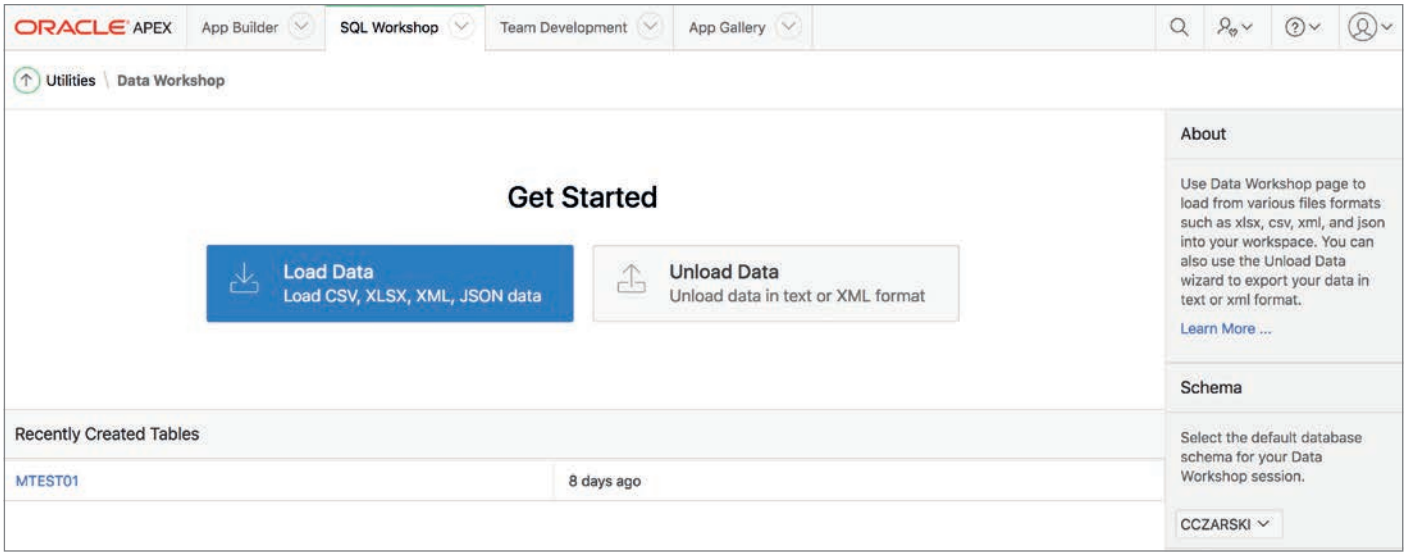


Abbildung 3: Upload von CSV-, XLSX-, JSON- oder XML-Dateien in APEX 19.1 (Quelle: Carsten Czarski)

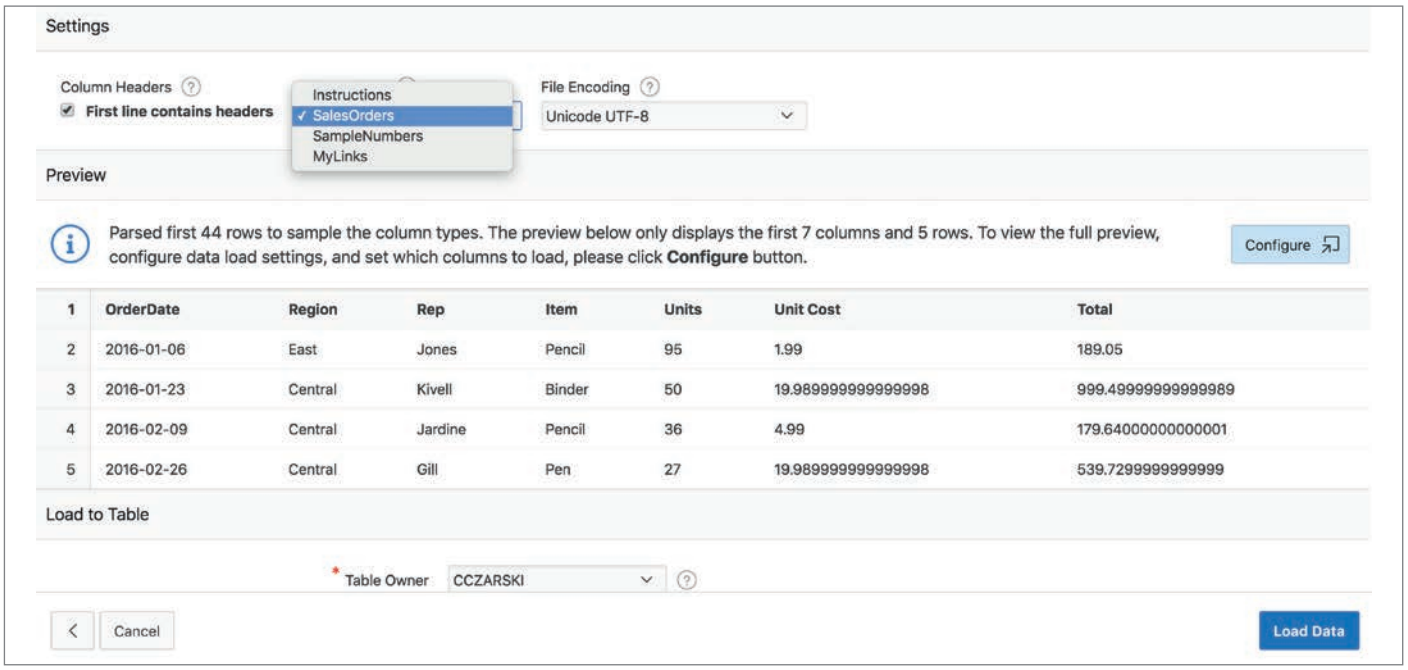


Abbildung 4: Nach dem Upload zeigt APEX eine Vorschau auf die Inhalte an. (Quelle: Carsten Czarski)

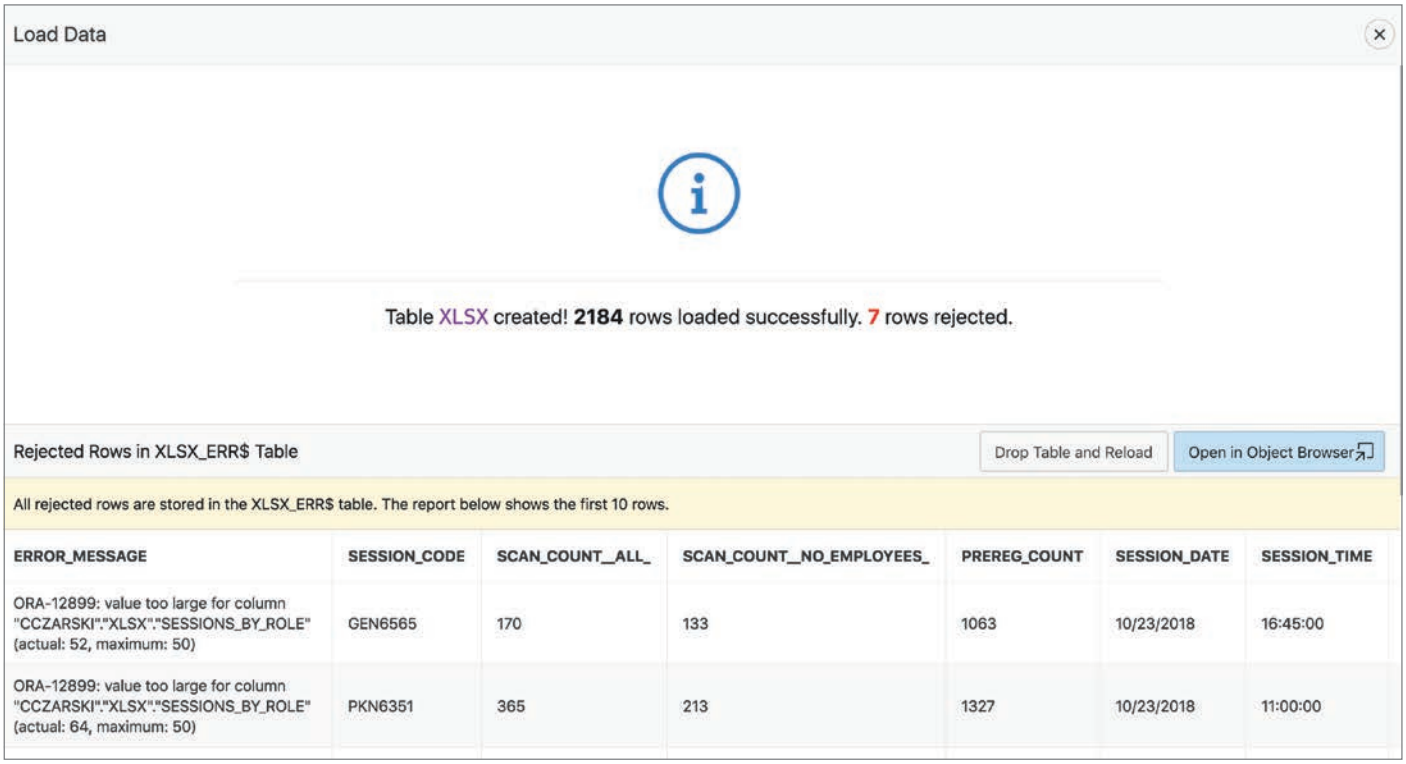


Abbildung 5: Zusammenfassung nach Abschluss des Ladevorgangs (Quelle: Carsten Czarski)

Nachdem eine Datei per Drag & Drop hochgeladen wurde, analysiert APEX diese automatisch und präsentiert eine erste Vorschau auf die Inhalte *(siehe Abbildung 4)*.

Tabellenspalten und Datentypen werden vom Parser automatisch erkannt. Für XLSX- Dateien mit mehreren Worksheets kann das Gewünschte in einer Auswahlliste gewählt werden (APEX nimmt standardmäßig das erste, das es findet). Die Schaltfläche Configure verzweigt zu einem Dialog, in dem unter anderem festgelegt werden kann, wie viele Zeilen zum Erkennen der Datentypen analysiert und welche Spalten überhaupt in die Tabelle geladen werden sollen.

Nachdem Namen für die neue Tabelle und für die Error-Log-Tabelle (diese nimmt Zeilen auf, die nicht in die Zieltabelle geladen werden können) angegeben wurden, wird der Ladevorgang mit der Schaltfläche Load Data gestartet.

APEX führt den Ladevorgang im Hintergrund aus und zeigt den Fortschritt an – dieser Dialog kann auch geschlossen werden, ohne dass der Ladevorgang unterbrochen wird. Wartet man bis zum Ende, so gibt es eine kurze Zusammenfassung. Zurückgewiesene Zeilen sind übrigens nicht verloren – sie stehen nach wie vor in der Error Table bereit (diese enthält alle Spalten als VARCHAR2-Datentyp) und können anschließend manuell weiterverarbeitet werden *(siehe Abbildung 5)*.

APEX_DATA_PARSER PL/SQL-Package

Der Parser, den das SQL Workshop Data Loading verwendet, steht auch dem APEX-Entwickler zur Verfügung. Mit dem APEX_DATA_PARSER PL/SQL-Paket kann der Entwickler CSV-, XLSX-, JSON- oder XML-Dateien auch selbst parsen und verarbeiten. Die Tabellenfunktion APEX_DATA_PARSER.PARSE erwartet den Dateiinhalt als BLOB und gibt die Ergebnisse den Spalten COL001 bis COL300 zurück (es werden also bis zu 300 Spalten unterstützt). Der BLOB mit dem Dateiinhalt kann aus einer Tabelle, aber auch aus anderen Quellen (Dateisystem, Netzwerk) kommen. Das Syntaxbeispiel in *Listing 1* holt eine XLSX-Datei aus dem Internet, übergibt diese an APEX_DATA_PARSER und gibt die Spalten 1 bis 10 zurück.

Da APEX_DATA_PARSER.PARSE wie eine Tabelle verwendet werden kann, sind die Möglichkeiten zur Weiterverarbeitung vielfältig: Der Entwickler kann eine Subquery verwenden, einen Join der Ergebnisse zu einer anderen Tabelle durchführen, ein INSERT as SELECT oder eine ganz einfache PL/SQL Cursor Loop implementieren. Letztlich kann man sich auf die eigentliche Geschäftslogik konzentrieren, das sehr technische Geschäft des File-Parsing übernimmt APEX.

Neben PARSE stellt APEX_DATA_PARSER auch weitere Funktionen bereit: So gibt es eine Funktion, die die in einer XLSX-Datei vorhandenen Worksheets zurückgibt, sowie Funktionen zum „Discovery" von Spalten und Datentypen.

Neue Region „Formular"

Die Architektur eines APEX-Formulars ändert sich mit APEX 19.1. Bislang gab es im APEX-Datenmodell eigentlich keine Entsprechung für ein Formular, denn bis einschließlich APEX 18.1 besteht ein „Formular" aus …

- einer APEX-Seite
- den Formularelementen (Page Items)
- dem „Automated Row Fetch"-Prozess im Bereich Pre-Rendering
- dem „Automated Row Process"-Prozess im Bereich Processing

Die Datenquelle für das Formular (Tabelle oder View) sowie die Einstellungen zum Primärschlüssel sind bei den Prozessen hinterlegt. Und zwar doppelt: sowohl beim Row Fetch als auch beim Row Pro-

```
with content as (
   select apex_web_service.make_rest_request_b(
       p_url          => 'http://.../sample_1000.xlsx',
       p_http_method  => 'GET' ) as data
   from dual
)
select col001, col002, col003, col004,
       col005, col006, col007, col008
   from content c,
       apex_data_parser.parse(
           p_content   => c.data,
           p_file_name => 'sample_1000.xlsx' ) p;

COL001    COL002         COL003         COL004     COL005             COL006   COL007         COL008
          First Name     Last Name      Gender     Country            Age      Date           Id
1         Dulce          Abril          Female     United States      32       15/10/2017     1562
2         Mara           Hashimoto      Female     Great Britain      25       16/08/2016     1582
3         Philip         Gent           Male       France             36       21/05/2015     2587
4         Kathleen       Hanner         Female     United States      25       15/10/2017     3549
5         Nereida        Magwood        Female     United States      58       16/08/2016     2468
10        Fallon         Winward        Female     Great Britain      28       16/08/2016     5486
:         :              :              :          :                  :        :              :
```

Listing 1: Einfaches File-Parsing mit PL/SQL und dem APEX_DATA_PARSER-Paket

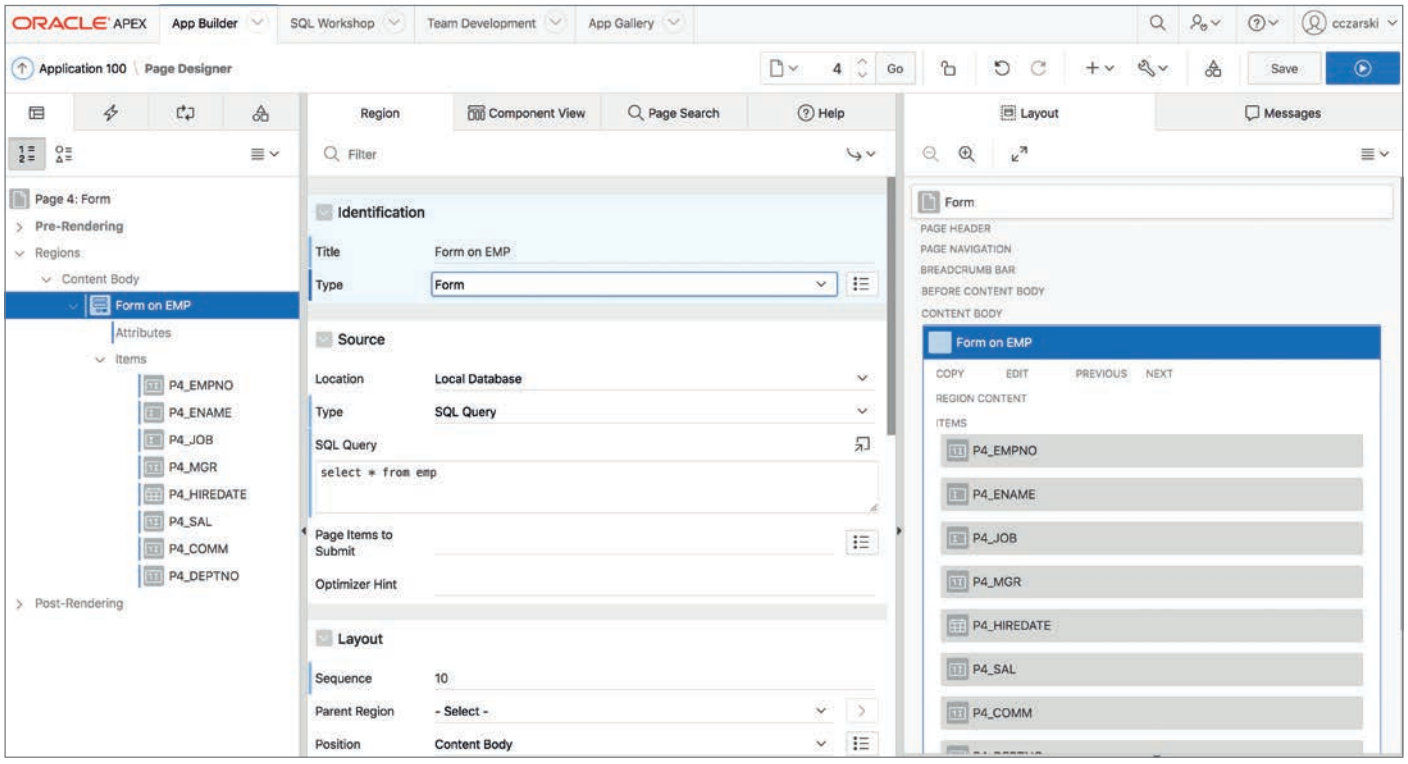


Abbildung 6: Form Region im Page Designer (Quelle: Carsten Czarski)

cessing. Seitenelemente, die Teil des Formulars sind, müssen im Bereich Source als Datenbankspalte definiert werden. Die Datenbankspalte bezieht sich dann auf die in den Prozessen hinterlegte Tabelle. Pro Seite war bislang nur ein Formular möglich.

Letztlich fehlte den Formularen eine zentrale Entität, die alle Definitionen enthält und auf die sich alle beteiligten Elemente (Page Items, Prozesse) beziehen.

APEX 19.1 löst dies mit der Region vom Typ Form. Eine solche kann nun auch im Page Designer per Drag & Drop auf die Seite gezogen werden. *Abbildung 6* zeigt die neue Form Region im Page Designer. Sobald die Datenquelle angegeben wurde (Tabelle, SQL-Abfrage oder Web-Source-Module), kann der Page Designer die Formularelemente automatisch erzeugen, denn es gibt mit der Form Region eine zentrale Stelle, die die Eigenschaften des Formulars definiert.

Wie für die „alten" Formulare gibt es auch für die neue Form Region einen Row-Fetch-und einen Row-DML-Prozess *(siehe Abbildung 7)*. Beide enthalten nun

jedoch keine Informationen über die Tabelle mehr – vielmehr beziehen sie sich auf die Form Region und bekommen ihre Informationen zur Datenquelle von dort.

Gleiches gilt für Formularelemente (Page Items): Diese sind – nach wie vor – einer Region zugeordnet, in der sie dargestellt werden. Zusätzlich kommt mit APEX 19.1 die optionale Zuordnung zu einer Form Region hinzu; damit, und mit den Einstellungen im Bereich Source, wird bestimmt, woher die Daten für dieses Formularelement kommen *(siehe Abbildung 8)*.

Zur Darstellung können Seitenelemente nach wie vor jeder beliebigen Region zugeordnet werden: Die Zuordnung zu einer Form Region ist völlig unabhängig vom Layout. Auch ermöglicht diese neue Architektur mehrere Formulare auf ein- und derselben APEX-Seite, was häufig nachgefragt wurde.

Auch das konkrete Row Processing bietet dem Entwickler weitaus mehr Möglichkeiten als früher: So kann nun eigener PL/SQL-Code als Teil des „Automatic Row

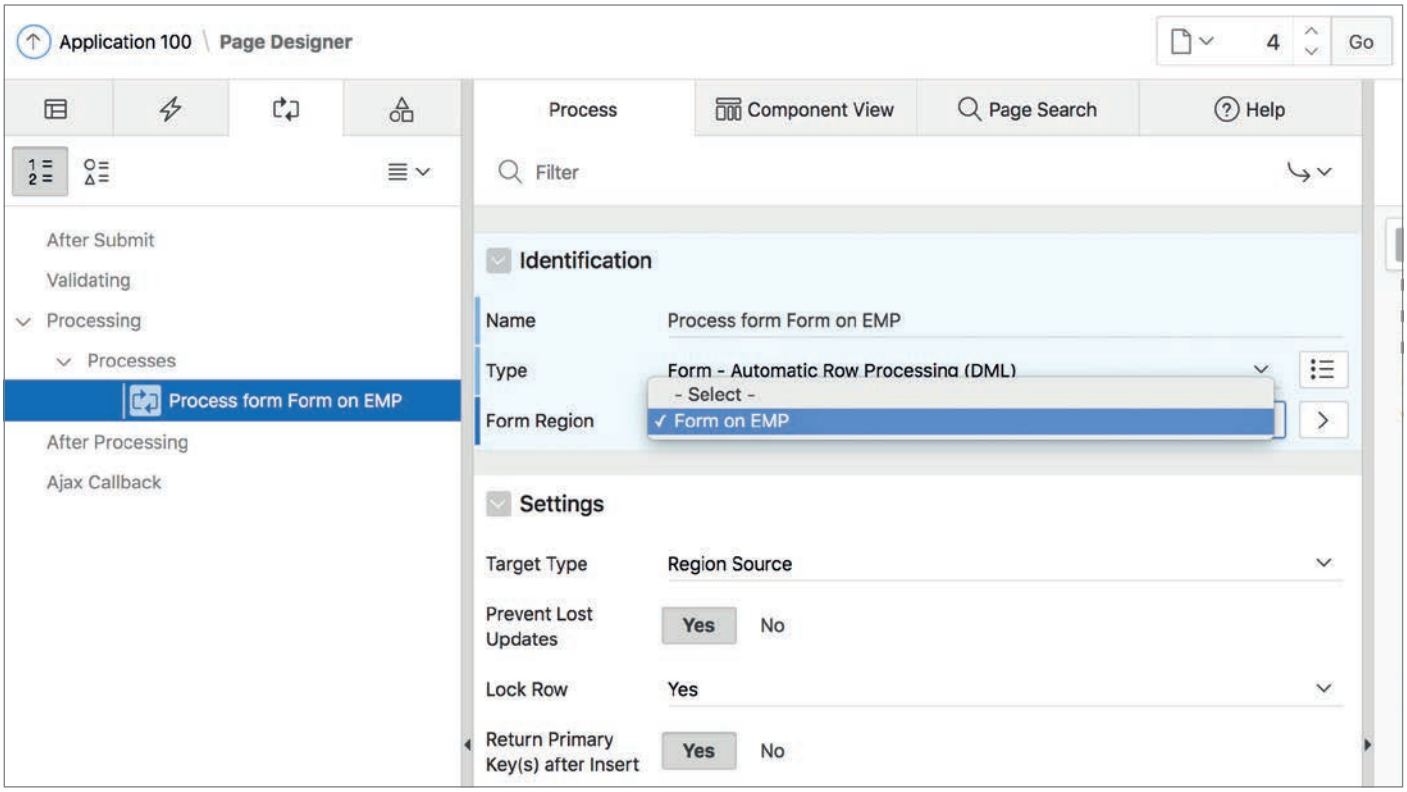


Abbildung 7: Row-Fetch- und Row-DML-Prozesse beziehen sich auf die Form Region (Quelle: Carsten Czarski)

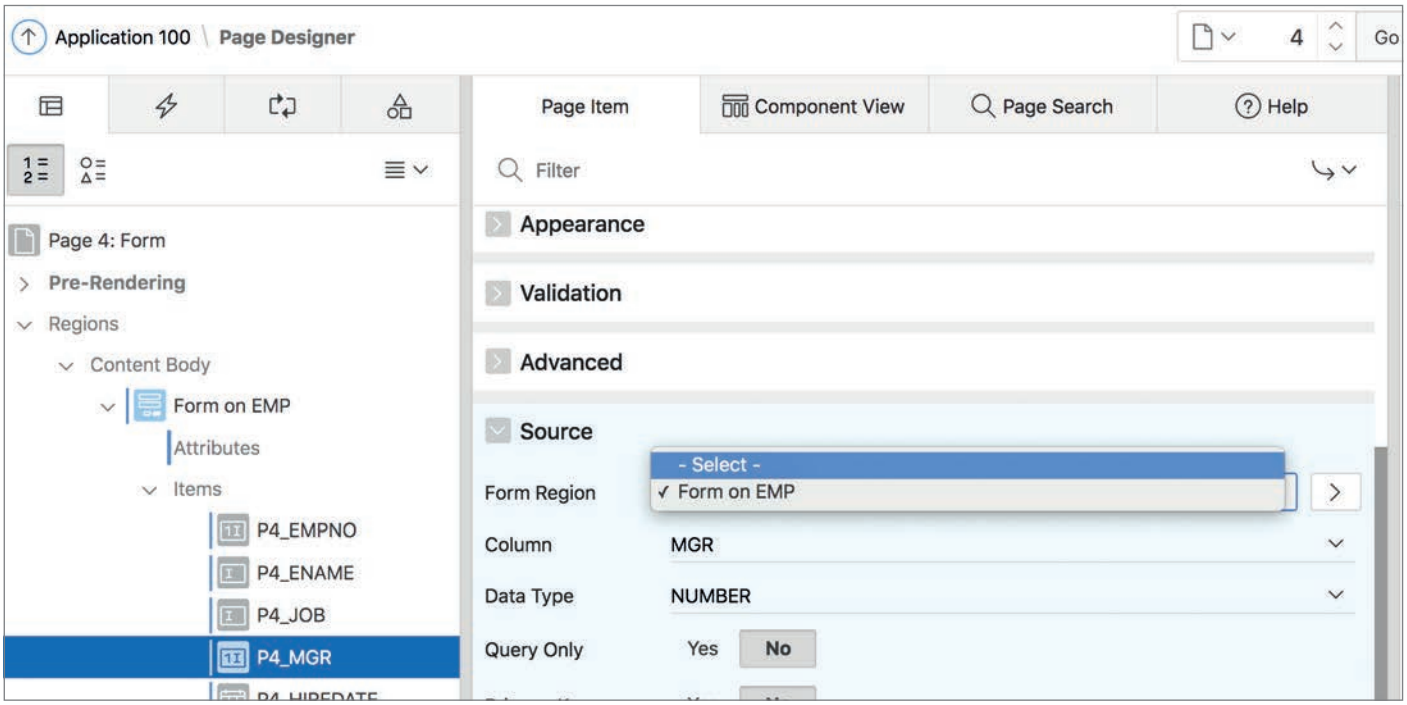


Abbildung 8: Zuordnung eines Seitenelemente zu einer Form Region (Quelle: Carsten Czarski)

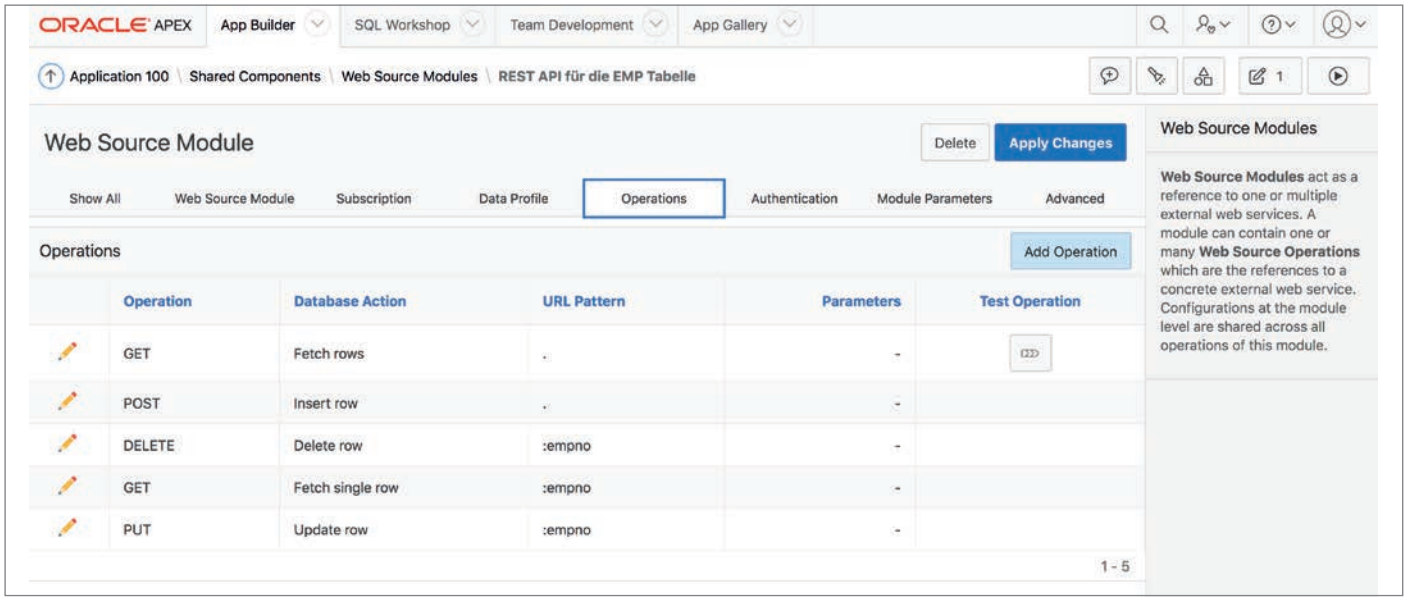


Abbildung 9: Ein Web Source Module speichert Metadaten zu einem REST-API (Quelle: Carsten Czarski)

Processing"-Prozesses hinterlegt werden. Der Vorteil im Vergleich zu einem „normalen" PL/SQL-Page-Prozess ist, dass APEX immer noch die Lost-Update- Erkennung durchführt. Das ist ein großer Vorteil im Vergleich zu früheren Versionen, in denen eigener PL/SQL-Code stets dazu führte, dass man auch die Lost-Update-Erkennung selbst machen musste.

Alles in allem erlaubt die neue Form Region wesentlich flexiblere und besser strukturierte Implementierungen von Formularen. Allerdings werden „alte" Formulare mit dem Upgrade auf APEX 19.1 nicht automatisch migriert. Es sind einfach zu viele Fälle denkbar, in denen eine solche Migration nicht möglich ist, man denke nur an unterschiedliche Tabellen in den Row-Fetch- und Row-DML-Prozessen.

Im Bereich Application Utilities > Upgrade Application gibt es daher einen Migrations-Assistenten. Wenn eine „alte" Formularseite für ein Upgrade zur neuen Form Region infrage kommt, kann das hier per Mausklick gemacht werden. Es empfiehlt sich allerdings, dies vorher auf einer Kopie der Anwendung zu testen.

Formulare und externe Datenquellen

Die Unterstützung externer Datenquellen wie REST-Services und REST Enabled SQL wurde bereits mit APEX 18.1 eingeführt – zu dieser Zeit allerdings beschränkt auf Read-Only-Komponenten wie Reports, Charts oder Kalender.

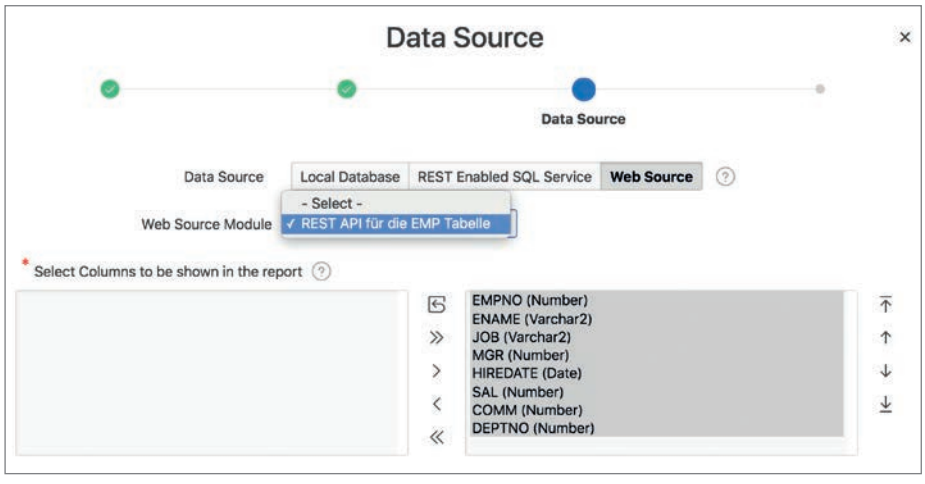


Abbildung 10: APEX 19.1 bietet Formulare für externe und interne Datenquellen (Quelle: Carsten Czarski)

Zusammen mit der neuen Form Region baut APEX 19.1 diese Unterstützung nun weiter aus, indem externe Datenquellen nun für Formulare (Forms) unterstützt werden – und zwar deklarativ: Der REST-Service lässt sich in APEX nutzen wie eine Tabelle.

Damit Formulare zum Erstellen, Ändern oder Löschen mit einem REST API überhaupt arbeiten können, muss dieses zunächst die entsprechenden Methoden und Endpunkte (POST, PUT, DELETE) anbieten. Ist das der Fall, so wird in APEX zunächst ein Web Source Module erzeugt. Web Source Modules speichern alle von APEX benötigten Metadaten zu einem externen REST-Service – dazu gehören URL-Endpunkte, HTTP-Methoden, aber auch technische Dinge wie HTTP-Header-Variablen oder Informationen zur Struktur der JSON-Antwort des REST-Service.

Abbildung 9 zeigt ein bereits angelegtes Web Source Module für eine externe REST- Schnittstelle: Metadaten für die HTTP-Methoden GET, POST, PUT und DELETE sind vorhanden; APEX „weiß" nun also, wie die REST-Schnittstelle aufgerufen werden muss, um Daten zu holen beziehungsweise einen Datensatz zu erzeugen, zu ändern oder zu löschen.

Ein neues Formular kann mit APEX 19.1 direkt auf diesem Web Source Module erstellt werden. Am einfachsten ist der Assistent Create Form and Report Pages – denn dieser erstellt einen Bericht mit passender Verknüpfung zu einem Formular. *Abbildung 10* zeigt den Schritt zur Auswahl der Datenquelle – während frühere APEX-Ver-

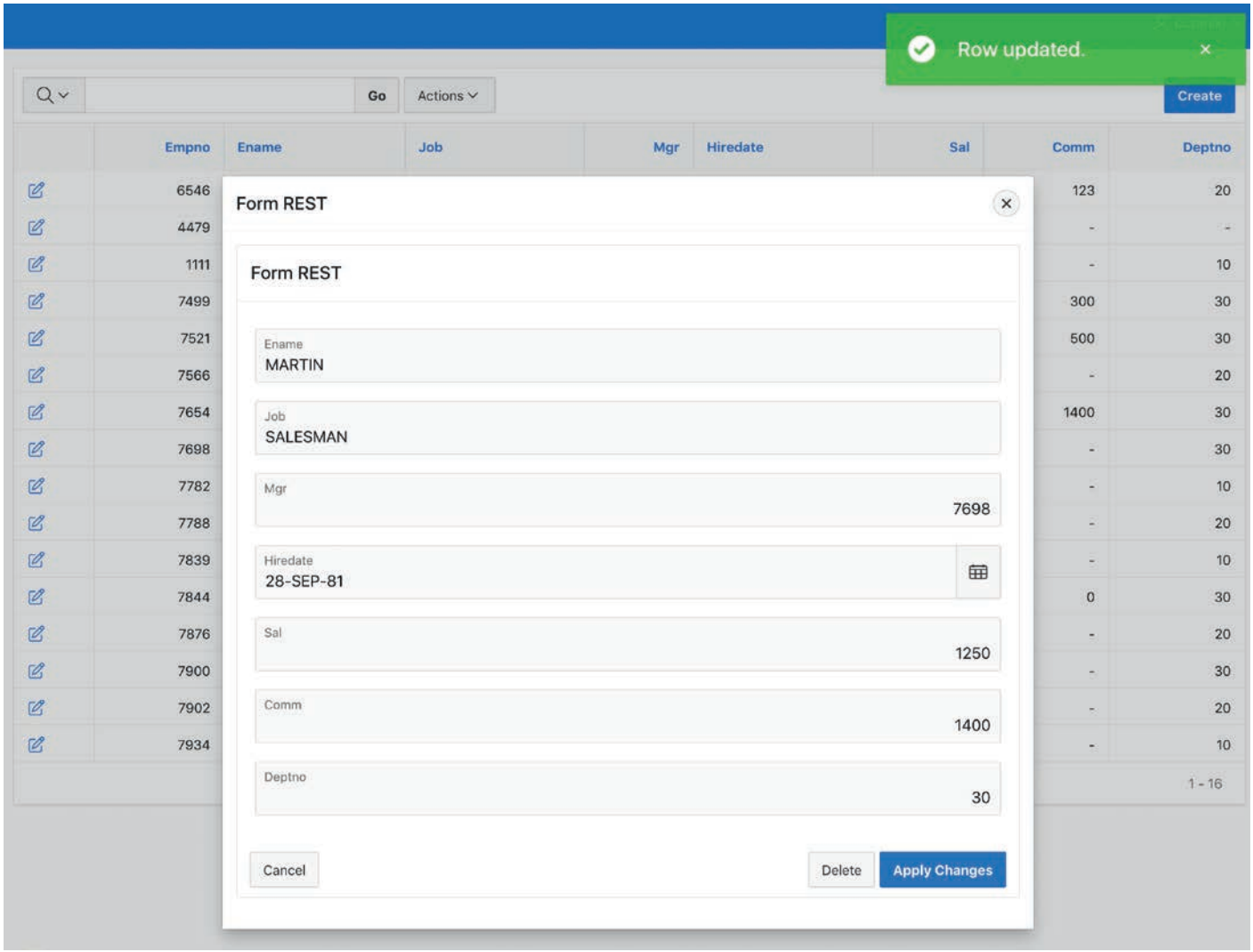


Abbildung 11: Bericht und Formular mit einem REST-Service (Quelle: Carsten Czarski)

sionen hier nur die Auswahl einer Tabelle oder View erlaubten, bietet APEX 19.1 sowohl externe und lokale Datenquellen als auch Tabellen oder SQL-Abfragen an.

Es wird deutlich, dass APEX das externe REST API betrachtet wie eine Tabelle. Es werden Spalten zurückgeliefert; diese haben Datentypen und zur Erstellung des Formulars muss die Primärschlüsselspalte angegeben werden. Nach Abschluss des Wizard enthält die APEX-Anwendung eine Berichts- und eine Formularseite *(siehe Abbildung 11)*. Beide arbeiten statt mit einer Tabelle mit der externen REST-Schnittstelle: alles deklarativ und ohne manuelles PL/SQL-Coding.

Weitere neue Features

Neben diesen größeren neuen Features bietet APEX 19.1 noch eine Menge kleinerer neuer Funktionen.

- Das Interactive Grid wurde mit einigen neuen deklarativen Eigenschaften ausgestattet. So können Eigenschaften der Toolbar nun im Page Designer eingerichtet werden – JavaScript Code ist nicht mehr nötig. Details zu den verfügbaren neuen Attributen finden sich auf dem Oracle Application Express Blog (siehe unter „Weitere Informationen").
- Der in Oracle JET neue Diagrammtyp Status Meter Gauge wird von APEX unterstützt. Im Gegenzug fällt der von Oracle JET nicht mehr unterstützte Dial Gauge weg. Für Gantt Charts gibt es neue deklarative Attribute für die Darstellung von Tooltips. Auch Stack Charts bekommen neue deklarative Attribute.
- Die von APEX verwendeten jQuery- und Oracle-JET-Bibliotheken erfahren ein Upgrade: APEX 19.1 verwendet Oracle JET 6.1.0, jQuery 3.3.1 und jQuery UI 1.12.1.
- Es gibt ein neues Regions-Template Inline Popup. Eine Region mit diesem Template wird auf der Seite zunächst nicht dargestellt. Mit den Dynamic Actions Open Region und Close Region wird sie als Dialog geöffnet beziehungsweise geschlossen. Damit werden einfache modale Dialoge möglich, ohne dass eigens eine Seite angelegt werden muss. Im Vergleich zu früheren APEX-Versionen bietet APEX 19.1 dies komplett deklarativ an.

Zusammenfassung

APEX 19.1 reiht sich nahtlos in die Reihe von APEX-Versionen seit dem ersten Release HTML DB 1.5 im Jahr 2004 ein. Es gibt einige „größere" neue Funktionen wie das neue Data Loading oder die Unterstützung von REST-Services in Formularen. Daneben gibt es eine Reihe kleinerer Features wie die Verbesserungen im

Interactive Grid oder für die JET Charts. Es lohnt sich, das Release auf apex.oracle.com genauer anzusehen oder es für eine eigene Installation herunterzuladen.

Zu allen hier vorgestellten neuen Features sind detaillierte Blog Postings auf dem Oracle Application Express Blog (siehe unter „Weitere Informationen") verfügbar.

Weitere Informationen

- Informationen zu Application Express und Demo-Umgebung *http://apex.oracle.com*
- Oracle Application Express Blog – mit vielen Postings zu APEX 19.1 *http://blogs.oracle.com/apex*

Carsten Czarski
carsten.czarski@oracle.com

Schnellstart – Versionskontrolle für existierende Oracle-(APEX-)Projekte

Ottmar Gobrecht, Linde

Viele Oracle-(APEX-)Projekte setzen bis heute keine Versionskontrolle ein. Die Gründe dafür sind vielfältig. Meist geht man wohl davon aus, dass die Datenbank ein sicherer Ort für den Quellcode ist. Mit einem funktionierenden Backup ist das auch richtig, man verliert aber auf jeden Fall die komplette Historie der Änderungen. Oft fehlt in laufenden Projekten auch einfach die Zeit, sich zusätzlich noch mit der Einführung einer Quellcode-Versionierung zu beschäftigen, weil auf den ersten Blick kein direkter Nutzen zu sehen ist. Wer wagt da unter Zeitdruck den zweiten Blick? Dieser Artikel will die Hürde für die Einführung einer Versionsverwaltung ein wenig tiefer legen.

Einleitung

Der erste Schritt ist der wichtigste, das wissen alle, die sich schon einmal nach langem Zögern endlich auf den Weg gemacht haben – worum auch immer es ging. Das Gleiche gilt für die Quellcode-Versionierung. Wenn man erst einmal auf dem Weg ist, fragt man sich, wie man so lange ohne auskommen konnte. Eine versionierte Quellcode-Basis vergrößert die Komfortzone bei der Entwicklung und die investierte Zeit zahlt sich mehrfach wieder aus. Doch wie geht man möglichst einfach und schnell den ersten Schritt? Wie gestaltet man die Struktur eines Repository?

Workflow-Änderung: dateibasiertes Arbeiten

Eine der wichtigsten Fragen lautet: Kann ich weiterarbeiten wie bisher? Die Antwort hängt ganz davon ab, wie man bisher gearbeitet hat. Grundsätzlich bedeutet das Arbeiten mit einer Quellcode-Versionierung auch einen Wechsel weg vom Entwickeln in der Datenbank. Das direkte Ändern und Kompilieren von Code in der Datenbank oder das Ändern von Tabellen mit dem Tool seiner Wahl muss sich hin zu einem dateibasierten Arbeiten entwickeln. Das bedeutet, mit einer Datei im lokalen Quellcode-Repository mit dem Tool der Wahl zu arbeiten. Die Veränderungen werden dann aus der Datei heraus kompiliert. Für Schema-Änderungen müssen Skripte den notwendigen DDL-Code enthalten. Je nach bisher verwendeten Tools ist das beim Bearbeiten von Logik eher nur ein subtiler, beim Verändern von Datenbankobjekten wie Tabellen dann doch schon ein großer Unterschied. Wurde bisher ein Schema-Diff-Tool verwendet, um

Änderungen der Entwicklungsumgebung in die Produktion zu transportieren, dann sieht das vielleicht auf den ersten Blick sogar wie ein Rückschritt aus.

Der Punkt ist hier, dass man nur dann eine verwertbare Quellcode-Historie bekommt, wenn man konsequent auf das dateibasierte Arbeiten setzt. Es kann auch einen Zwischenschritt hin zu diesem Ansatz geben. Man arbeitet weiter wie bisher und exportiert regelmäßig alle Objekte des Schemas in die Quellcode-Verwaltung. Auf diesem Weg erhält man wenigstens eine gewisse geordnete Transparenz der Änderungen. Werden dann im Laufe des Projektes die Vorteile dieses Vorgehens ersichtlich, fehlt einem meist noch die Information, wer denn die jeweiligen Änderungen ausgeführt hat. Außerdem werden durch die gewonnene Transparenz schnell Wünsche geweckt wie „Wenn ich den alten Code im Repository habe, dann darf ich ja mal etwas probieren, ich kann ja notfalls zurück". Schon ist die erste Hürde genommen und die Offenheit, vielleicht doch einmal richtig damit loszulegen, da.

Dass so etwas noch viel besser funktionieren kann, darauf kommen die meisten Entwickler dann selbst. Zum Beispiel das Anlegen von einem Branch (Entwicklungszweig) im lokalen Repository. Damit kann man etwas auszuprobieren, ohne Gefahr zu laufen, aufgrund eines unerwartet zu lösenden Bugs die gemachten Änderungen wieder verwerfen zu müssen. Man wechselt bequem in den (parallelen) Hauptzweig, arbeitet am Bugfix, aktualisiert seinen Branch danach auf den geänderten Hauptzweig und kann dann einfach weiter ausprobieren. Das geschieht alles ohne die Gefahr, Dinge zu verlieren oder andere Kollegen zu beeinflussen. Habe ich jetzt irgendjemanden abgehängt? Kein Problem, einfach loslegen – hier hilft nur Praxis.

Toolvergleich: DDL-Export

Wie bewerkstelligt man den ersten Schritt? Wie exportiert man die Schema-Objekte seiner Anwendung in eine wohlsortierte Verzeichnisstruktur? Die meisten der verwendeten grafischen Entwicklungstools bringen dafür eine Export-Funktion mit. Diese ist je nach Tool mehr oder weniger gut dafür geeignet, ein Quellcode-Repository aufzubauen. Deshalb schauen wir uns an, wie die am häufigsten verwendeten Tools SQL-Developer, PL/SQL-Developer und Toad das können. Ein Wort vorweg: Die Intention des Schema-Exports dieser Tools ist eigentlich, die Objekte in ein anderes Schema, vielleicht sogar in eine andere Datenbank zu bringen, und nicht, ein Quellcode-Repository zu erstellen. Folgende Fragen bewegen uns, wenn es darum geht, ein solches aufzubauen:

- Ist eine Skript-Datei pro Objekt möglich?

- Sind Unterverzeichnisse pro Objekttyp möglich?
- Sind eigene Dateien für Fremdschlüssel möglich? (Nützlich für einfachere Master-Skripte)
- Können „Object already exists“-Fehler verhindert werden? (Oder anders gefragt: Sind die Skripte wiederanlauffähig?)
- Können Daten exportiert werden? (Möglichst im CSV-Format zur Verfolgung von Stammdatenänderungen)
- Ist eine APEX App exportierbar? (Womöglich auch zerlegt in die Einzelteile wie Pages, Shared Components usw.)

Kriterium	SQL-Dev.	PL/SQL-Dev.	Toad
Datei pro Objekt	Ja	Ja	Ja
Unterverz. pro Typ	Ja	Nein	Ja
FK Constr. extra	Ja	Nein	Ja
Verhi. „object exist”	Nein	Nein	Nein
Export Daten	Ja	Nein	Jein
Export APEX App	Jein	Nein	Nein

Abbildung 1: Toolvergleich DDL-Export (Quelle: Ottmar Gobrecht)

Abbildung 1 zeigt das Ergebnis bezogen auf die Fragestellung.

Anmerkungen zum SQL-Developer:

- Ist am übersichtlichsten
- Viele Formate für Datenexport (auch CSV)
- Umfangreich konfigurierbar
- Export APEX App nur mit Command-Line-Version SQLcl

Anmerkungen zum PL/SQL-Developer:

- Wenig konfigurierbar
- Enttäuscht für den Aufbau eines Quellcode-Repository

Anmerkungen zu Toad:

- Zwei Exportmöglichkeiten (mindestens)
  - Entweder Unterverzeichnisse pro Objekttyp…
  - … oder Daten
- Daten nur als Insert Statements
- Umfangreich konfigurierbar, relativ unübersichtlich

Keines der Tools liefert uns ein fertiges, gut strukturiertes Quellcode-Repository. Man muss mehr oder weniger viel nacharbeiten. Das heißt nicht, dass es nicht gehen würde, zumal das Einsortieren der Objekte in eine vernünftige Struktur eine einmalige Arbeit sein sollte. Lediglich beim Export der APEX-Anwendung wünscht man sich eine automatische Lösung, weil dieser Export in Zukunft häufiger gemacht werden muss. Das liegt in der Natur von APEX als deklarativem „Low Code Framework“.

Wir entwickeln die APEX-Anwendung im Browser und die Meta-Daten der Anwendung liegen im APEX-Repository der Datenbank. Ein Export der Anwendung kann auch mit dem Browser gemacht werden, dabei erhält man jedoch nur eine einzige große SQL-Datei mit der gesamten Applikation. Wünschenswert für ein Quellcode-Repository wäre ein Export der Einzelteile wie zum Beispiel Pages, Shared Components, Plug-ins usw. Damit lässt sich dann für die Anwendung verfolgen, wie sie sich entwickelt hat, und auch eine Suche im Repository etwa nach einem bestimmten Package-Aufruf ergibt bei einer zerlegten APEX-Anwendung wesentlich mehr Sinn. In der Vergangenheit konnte man dafür den APEX-Export-Splitter verwenden, im Grunde eine Java-Klasse, die im Lieferumfang jeder APEX-Version bis heute enthalten ist. Der Nachteil ist, dass man die erstellte Verzeichnisstruktur so nehmen muss, wie sie vom Splitter erstellt wurde, oder man verändert sie im Nachgang mit lokalen Skripten auf dem PC. Schöner wäre allerdings, die Verzeichnisstruktur schon während des Exports anpassen zu können, um alle Master-Skripte in einem zentralen Ordner des Repository speichern zu können.

Seit APEX Version 5.1.4 gibt es zu diesem Zweck das APEX_EXPORT Package. Mit diesem kann man entweder das große Gesamtskript oder eben die zerlegte Variante exportieren. Das Rückgabefor-

```
WITH
   FUNCTION backapp RETURN BLOB IS
   BEGIN
       RETURN plex.to_zip(plex.backapp(
           p_app_id                  => 100,
               /* If null, we simply skip the APEX app export. */
           p_include_object_ddl      => true,
               /* If true, include DDL of current user/schema and
               all its objects. */
           p_include_templates       => true,
               /* If true, include templates for README.md, export
               and install scripts. */
           p_include_runtime_log     => true,
               /* If true, generate file plex_backapp_log.md with
               runtime statistics. */
           p_include_data            => true,
               /* If true, include CSV data of each table. */
           p_data_max_rows           => 1000,
               /* Maximum number of rows per table. */
           p_data_table_name_like => ,DEMO_PRODUCT_INFO,DEMO_STATES‘
               /* A comma separated list of like expressions to filter
               the tables. Example: 'EMP%,DEPT%' will be translated to
               "where ... and (table_name like 'EMP%' escape '\' or
               table_name like 'DEPT%' escape '\')". */
       ));
   END backapp;

SELECT backapp FROM dual;
```

Listing 1: Möglicher Erst-Export

mat ist in beiden Fällen eine Collection – jede Zeile der Collection beinhaltet eine Pfadangabe für die jeweilige Exportdatei und ein CLOB-Feld mit dem Inhalt. Man könnte also in einem Export-Skript diese Collection bearbeiten und die Pfadangaben an die eigenen Bedürfnisse anpassen. Genau das macht das vom Autor des Artikels als Open Source veröffentlichte Tool-Package PLEX – der Name steht für PL/SQL Export Utilities. Des Weiteren kann PLEX jede der genannten Fragestellungen zum DDL-Export mit Ja beantworten – kein Wunder, es ist dafür entwickelt worden.

Schnellstart: Package PLEX

PLEX hat zum Ziel, die erste Version des Repository mit einer einfachen Query als BLOB selektieren zu können. Nach Speichern des BLOB in das lokale Verzeichnissystem auf dem PC mit der Endung „.zip" kann man es entpacken und findet in der generierten Verzeichnisstruktur dann auch Skript-Beispiele für zukünftige halb- oder vollautomatische Exporte und Deployments per Shell-Befehl beziehungsweise SQL*Plus. Ein möglicher Erst-Export könnte so aussehen wie in *Listing 1* dargestellt.

Da PLEX boolesche Parameter besitzt, benutzen wir eine Inline-Funktion in der With-Klausel. Wer eine Datenbankversion kleiner als 12c verwendet, erstellt sich eine Hilfsfunktion analog dem Beispiel. PLEX hat noch einige Parameter mehr, um zum Beispiel den APEX App Export zu konfigurieren. Weitere Details sind auf der offiziellen Projektseite *[1]* zu finden. Je nach Größe des Schemas und der APEX App kann dieser erste Aufruf zwischen wenigen Sekunden und mehreren Minuten dauern. Das liegt daran, dass im Hintergrund für jedes Objekt das Package DBMS_METADATA bemüht wird, um das DDL des Objektes zu generieren. Wer an Laufzeit-Infos interessiert ist, findet im Hauptverzeichnis des Exports eine Logdatei mit Informationen darüber, was PLEX so alles gemacht hat und wie lange die jeweiligen Schritte gedauert haben. PLEX selbst existiert jetzt in einer ersten Version – Verbesserungsvorschläge oder Fehlermeldungen sind willkommen und können über die Projektseite als Issue gemeldet werden.

```
BEGIN
   FOR i IN (
       SELECT 'DEMO_STATES' AS object_name
              FROM dual
           MINUS
       SELECT object_name
              FROM user_objects
   ) LOOP
       EXECUTE IMMEDIATE q'[
----------------------------------------
CREATE TABLE "DEMO_STATES" (
   "ST"           VARCHAR2(30),
   "STATE_NAME"   VARCHAR2(30)
)
----------------------------------------
       ]';
   END LOOP;
END;
/

BEGIN
   FOR i IN (
       SELECT ,STATE_DESCRIPTION‘ AS column_name
              FROM dual
           MINUS
       SELECT column_name
       FROM user_tab_columns
           WHERE table_name = ,DEMO_STATES‘
               AND column_name = ,STATE_DESCRIPTION‘
   ) LOOP
   EXECUTE IMMEDIATE q‘[
----------------------------------------
ALTER TABLE demo_states ADD (
   state_description VARCHAR2(255)
)
----------------------------------------
       ]‘;
   END LOOP;
END;
/
```

Listing 2: Wiederanlauffähiges Skript

Ab hier hängt es stark von den Bedürfnissen des jeweiligen Projektes ab, ob und wie häufig man einen DDL-Export ausführt. Der Normalweg sollte ein regelmäßiges Exportieren der APEX-Anwendung sein. Alles andere sollte ab jetzt lokal bearbeitet werden und es infolgedessen kein Bedürfnis für einen Export geben. Wie schon eingangs erwähnt, mag es aber auch Situationen geben, in denen man regelmäßig alle Schema-Objekte exportiert, um die Anwendung zu dokumentieren. Dies sollte allerdings nur als Zwischenschritt hin zu einer dateibasierten Arbeitsweise verstanden werden.

Einen Sonderfall stellt generierter Code dar: Also zum Beispiel das Nutzen von Quick-SQL in APEX oder das Generieren von Tabellen-APIs. Hier kann man überlegen, den generierten Code mit entsprechend konfigurierten Objekt-Filtern zu exportieren. Bei APIs sollte man darüber nachdenken, ob man nicht lieber den Generator anstelle des generierten Codes versioniert – das hängt jedoch ganz davon ab, ob der generierte Code noch manuell verändert wird oder nicht. Jeder manuell erstellte Code sollte versioniert werden.

Geschwindigkeit: Immer Skripte

Wie man schon an den von PLEX mitgelieferten Skript-Beispielen erkennen kann, sollte es das Ziel sein, jedwedes Deployment per Skript auszuführen. Dazu gehört dann auch wieder die Überlegung, das Objekt-DDL nur beim allerersten Mal zu exportieren – denn was passiert zum

Beispiel beim Export von Tabellen-DDL? Hat man eine weitere Spalte per Alter-Table-Anweisung in das Tabellenskript geschrieben und dafür gesorgt, dass es wiederanlauffähig ist *(siehe Listing 2)*, dann überschreibt ein weiterer DDL-Export die Alter-Anweisung und unser Tabellenskript enthält einfach die neue Spalte, gelistet in der Create-Table-Anweisung. Diese wird jedoch aufgrund der Wiederanlauffähigkeit niemals mehr ausgeführt, da unsere Tabelle ja schon existiert.

Somit haben wir jetzt eine Quellcode-Datei, die für das Deployment nicht mehr geeignet ist. Schade eigentlich, denn bis eben hätten wir mit unserem Master-Skript einfach immer alle Objekt-Skripte aufrufen können und nur die Änderungen (wie z.B. unsere neue Spalte) wären wirklich ausgeführt worden – ein einfaches Deployment und eine übersichtliche Historie im Repository. Demgegenüber steht das automatische Erstellen von Diff-Skripten, die mitunter nicht wiederanlauffähig sind und auch bei etwas komplizierteren Schema-Änderungen die Gefahr eines Datenverlustes beinhalten. Man kommt auch bei sehr teuren Tools dieser Klasse nicht umhin, die generierten Skripte auf Richtigkeit zu überprüfen und manuelle Anpassungen für etwaige Datenmigrationen auszuführen. Die Entscheidung, welchen Weg man geht, nimmt einem niemand ab. Jetzt sind wir mittendrin in der Diskussion darüber, wie man ein Deployment in das Zielsystem durchführt. Da kann dann auch PLEX nicht weiterhelfen – außer, dass das Package per se versucht, alle notwendigen Objekt-DDLs wiederanlauffähig zu extrahieren. Hier sind wir im weiten Feld der individuellen Bedürfnisse eines Projektes angekommen. Die von PLEX gelieferten Skriptbeispiele müssen auf jeden Fall an die Bedürfnisse des jeweiligen Projektes angepasst werden. Der Autor beispielsweise unterscheidet fast immer zwischen Skripten für das Backend und solchen für das Frontend – sowohl für den Export als auch für das Deployment.

Überlegung: Verzeichnisstruktur im Repository

Wie soll man sein Repository strukturieren? Hier ein Vorschlag, der sich im Laufe der Zeit für mich persönlich als vorteilhaft herausgestellt hat. Jedes Projekt mag da andere Vorstellungen haben. Ich verwende hier eine einfache Auflistung – die ersten Wörter jedes Listenpunktes stellen den Verzeichnisnamen dar und etwaige Erklärungen finden sich in Klammern dahinter:

- app_backend (unser Schema DDL)
  - constraints (ein Ordner pro Objekttyp)
  - package_bodies
  - packages
  - ref_constraints
  - sequences
  - tables
  - …
- app_data (Verfolgung von Master-Daten, optional)
- app_frontend (unsere APEX app)
  - pages
  - shared_components
  - …
- docs (die Dokumentation)
- reports
- scripts (alle Master-Skripte vereint)
  - logs (Export- und Installations-Logs)
    - export_app_100_from_MYSCHEMA_at_DEV_20190315_132542
    - install_app_100_into_MYSCHEMA_at_TEST_20190318_083756
    - …
  - misc (häufig genutzte Skripte, etwa zum Rekompilieren des Schemas)
  - 1_export_app_from_DEV.bat (Shellskript mit Parametern für das eigentliche SQL-Exportskript)
  - 2_install_app_into_TEST.bat (Shellskript mit Parametern für das eigentliche SQL-Deploymentskript)
  - 3_install_app_into_PROD.bat
  - export_app_custom_code.sql (Export-Master-Skript mit individuellen Anpassungen)
  - install_app_custom_code.sql (Deployment-Master-Skript mit individuellen Anpassungen je Release)
  - install_backend_generated_by_plex.sql (wird aufgerufen vom Master-Skript)
  - install_frontend_generated_by_apex.sql (wird aufgerufen vom Master-Skript)
- tests (Unit- und Frontend-Tests)
- README.md (generelle Informationen zum Projekt und Links in die Dokumentation)

Das Repository sollte logisch aufgebaut sein und keine extrem tiefen Verzeichnisstrukturen beinhalten, um den Umgang damit zu erleichtern. Alle Master-Skripte sind vereint in einem Skript-Ordner, auch das von APEX generierte Installationsskript für das Frontend. Alle Master-Skripte sollten während der Verwendung die jeweiligen Rückmeldungen in entsprechende Log-Dateien im Unterordner „logs" ablegen – damit sind Exporte von Quellcode und Deployments in Zielsysteme nachvollziehbar.

Ausblick: CI/CD & Schema-Migration

Die skizzierte Diskussion um das Deployment lässt es schon erahnen: Einfach ein Quellcode-Repository aufsetzen und man ist fertig, ist nicht die Lösung. Das war nur der erste Schritt zum Aufwärmen. Denn hat man erst einmal alles wohl strukturiert und per Skript lauffähig gemacht, dann kann man das Deployment auch im zweiten Schritt automatisieren. Der Inhalt dieser Diskussion sprengt allerdings bei Weitem den Rahmen dieses Artikels, der ja den ersten Schritt im Fokus hat. Daher legt der Autor jedem nahe, nach diesem ersten Schritt nicht stehen zu bleiben und weitere Schritte ins Auge zu fassen. Einen sehr guten Grundlagenartikel hat beispielsweise Martin Fowler zum Thema verfasst – er betrachtet jede Änderung an der Datenbank als eine Migration. Zum weiteren Studium seien daher folgende Einstiegspunkte genannt:

- Antti Kirmanen: Git vs. Subversion (SVN): Welches Versionskontrollsystem sollten Sie nutzen? *[2]*
- Martin Fowler: Evolutionary Database Design *[3]*
- Samuel Nitsche:
  - There is no clean (database) development without Version Control *[4]*
  - „One does not simply update a database" – migration based database development *[5]*
- Blain Carter:
  - Tips to help PL/SQL developers get started with CI/CD *[6]*
  - CI/CD for Database Developers – Export Database Objects into Version Control *[7]*
- Jeff Smith: 19.X SQLcl Teaser: LIQUIBASE *[8]*

Quellen

[1] *https://github.com/ogobrecht/plex*
[2] *https://entwickler.de/online/development/git-subversion-svn-versionskontrollsystem-579792227.html*
[3] *https://www.martinfowler.com/articles/evodb.html*
[4] *https://cleandatabase.wordpress.com/2017/09/22/there-is-no-clean-database-development-without-version-control*
[5] *https://cleandatabase.wordpress.com/2017/11/28/one-does-not-simply-update-a-database-migration-based-database-development*
[6] *https://learncodeshare.net/2018/04/30/tips-to-help-pl-sql-developers-get-started-with-ci-cd*
[7] *https://learncodeshare.net/2018/07/16/ci-cd-for-database-developers-export-data-base-objects-into-version-control*
[8] *https://www.thatjeffsmith.com/archive/2019/01/19-x-sqlcl-teaser-liquibase/*

Ottmar Gobrecht
ottmar.gobrecht@linde.com

Ansible oder Puppet: Which way to go?

Raphael Daum (DBConcepts) und Florin-Catalin Enache (T-Systems Austria)

Was verbindet DevOps, Cloud, Serverless Computing und Continous Delivery? Hinter all diesen Begriffen verbirgt sich die Notwendigkeit der Automatisierung, die Notwendigkeit, Abläufe deterministisch und reproduzierbar zu machen. Wir stellen zwei prominente Vertreter dieses Automatisierungsansatzes vor: Der Newcomer Ansible trifft auf Puppet, den Darling der letzten Dekade. Was verbindet sie, was trennt sie und welchen Nutzen bieten sie?

Alt gegen Jung, so könnte ein Vergleich zwischen den Tools Puppet und Ansible zusammengefasst werden. Während der frühere Platzhirsch scheinbar schon mit dabei war, als das Internet gerade anfing, ein Massenphänomen zu werden, betritt mit Ansible ein Werkzeug zu einem Zeitpunkt die Bühne des Geschehens, an dem nur mehr von Cloud und Terraforming die Rede ist. Somit sollte auch schon klar sein, welchem der beiden Konfigurationsmanagement-Tools die Zukunft gehört.

Werkzeugkoffer: Meiner oder deiner?

Damit würden wir es uns allerdings zu einfach machen, denn beide trennt nicht nur zeitlich die eine oder andere Dekade, sondern auch architektonisch unterscheiden sie sich in manchen Bereichen grundlegend. Und somit kann die Frage, ob sich der Einsatz von Puppet oder Ansible eher lohnt, auch nicht beantwortet werden. Obwohl eine Konfigurationsmanagement-Software im Grunde nichts anderes als ein Werkzeugkoffer ist, der einen in die Lage versetzen soll, spannende IT-Dinge zu bauen, und somit eigentlich in den Hintergrund treten soll, ist es dann doch der Werkzeugkoffer der Wahl. Und niemand will, gerade in den Zeiten von „Agile Development“ und „Continous Delivery“, seinen Werkzeugkoffer im Quartalsrhythmus austauschen, denn das verursacht unnötige Kosten und Reibungsverluste. Deshalb sollte die Entscheidung mit Bedacht getroffen und hierbei insbesondere herausgearbeitet werden, welche Probleme die Konfigurationsmanagement-Software lösen soll (und welche nicht gelöst werden können).

Ansible: Speed Transformation

Ansible ist eine Open-Source-Automatisierungsplattform. Mithilfe der einfachen Automatisierungssprache lässt sich eine IT-Anwendungsinfrastruktur in Ansible-Playbooks perfekt beschreiben. Ansible ist auch eine Automatisierungs-Engine zur Ausführung von Ansible-Playbooks *[5]*.

Die jetzt Red Hat gehörende Software kann leistungsfähige Automatisierungsaufgaben verwalten und lässt sich an viele verschiedene Workflows und Umgebungen anpassen. Gleichzeitig kann

Ansible von neuen Benutzern schnell genutzt werden, um produktiver zu werden.

Ansible: Konzepte und Architektur [6]

In der Ansible-Architektur gibt es zwei Arten von Rechnern: Kontrollknoten und verwaltete Hosts. Ansible wird auf einem Kontrollknoten installiert und ausgeführt, und dieser Rechner enthält auch Kopien der Ansible-Projektdateien. Bei einem Kontrollknoten kann es sich um den Laptop eines Administrators, um ein von einer Reihe von Administratoren gemeinsam genutztes System oder um einen Server handeln, auf dem Ansible Tower ausgeführt wird.

Verwaltete Hosts sind in einem Inventar aufgeführt, in dem diese Systeme außerdem zur einfacheren kollektiven Verwaltung in Gruppen organisiert werden. Das Inventar kann in einer statischen Textdatei definiert oder dynamisch mithilfe von Skripten ermittelt werden, die Informationen aus externen Quellen abrufen.

Statt komplexe Skripte zu schreiben, erstellen Ansible-Benutzer Plays auf hoher Ebene, um sicherzustellen, dass ein Host oder eine Gruppe von Hosts einen bestimmten Status aufweist.

Ein Play führt eine Reihe von Aufgaben auf dem oder den Hosts in der Reihenfolge aus, die in dem Play angegeben ist. Diese Plays werden im YAML-Format in einer Textdatei angegeben. Eine Datei, die ein oder mehrere Plays enthält, wird als Playbook bezeichnet.

Mit jeder Aufgabe wird ein Modul *[7]* ausgeführt, ein kurzer Code-Abschnitt (geschrieben in Python, PowerShell oder einer anderen Sprache) mit bestimmten Argumenten. Jedes Modul ist im Grunde ein Tool in einem Toolkit. Im Lieferumfang von Ansible sind mehrere Tausend nützlicher Module enthalten, die ein breites Spektrum von Automatisierungsaufgaben ausführen können. Sie können auf Systemdateien agieren, Software installieren oder API-Aufrufe vornehmen.

Wenn ein Modul in einer Aufgabe verwendet wird, stellt es im Allgemeinen sicher, dass ein bestimmtes Element im Zusammenhang mit dem Rechner einen bestimmten Status aufweist. Eine Aufgabe mit einem Modul *[8]* kann zum Beispiel sicherstellen, dass eine Datei vorhanden ist und über bestimmte Berechtigungen und Inhalte verfügt, während eine Aufgabe mit einem anderen Modul vielleicht gewährleistet, dass ein bestimmtes Dateisystem gemountet ist. Wenn das System nicht diesen Status aufweist, sollte es von der Aufgabe auf diesen Status gesetzt werden. Wenn das System bereits diesen Status aufweist, sollte die Aufgabe nichts unternehmen. Schlägt eine Aufgabe fehl, besteht das Standardverhalten von Ansible darin, das restliche Playbook für die Hosts abzubrechen, auf denen die Auf-

```
tasks:
  - name: install httpd
    yum:
        name: httpd
        state: installed
  - name: web server is enabled
    service:
        name: httpd
        enabled: true
  - name: install mysql
    yum:
        name: mysql-server
        state: installed
  - name: mysql is enabled
    service:
        name: mysql
        enabled: true
  - name: ensure index.html file exists
    lineinfile:
    path: /var/www/html/index.html
    line: ,'Ansible is fun!'
    state: present
```

Listing 1: Ansible in Aktion

```
# install apache2 package
package { 'apache2':
  ensure => installed,
}

# ensure apache2 service is running
service { 'apache2':
  ensure => running,
}

# install mysql-server package
package { 'mysql-server':
  ensure => installed,
}

# ensure mysql service is running
service { 'mysql':
  ensure => running,
}

# ensure index.html file exists
file { '/var/www/html/index.html':
  ensure => file,
  content => 'Puppet is fun!',
  require => Package['apache2'],
}
```

Listing 2: P uppet-DSL in Aktion

gabe fehlgeschlagen ist. Aufgaben, Plays und Playbooks sollten idempotent sein. Dies bedeutet, dass man in der Lage sein sollte, ein Playbook mehrere Male sicher auf denselben Hosts auszuführen. Außerdem sollte das Playbook bei der Ausführung keine Änderungen vornehmen, wenn die Systeme den korrekten Status aufweisen *(siehe Listing 1)*.

Puppet: Neuer Wein in alten Schläuchen

Puppet *[1]* gibt es seit dem Jahr 2005 und erfreut sich immer noch großer Beliebtheit. Ein Blick auf Puppetforge *[2]* macht schnell deutlich, dass Puppet schon zum Lösen verschiedenster Probleme Anwendung fand und es hier kaum etwas gibt, das noch nicht als fertiges Modul zum Download zur Verfügung steht. Das ist sicherlich die Stärke von Puppet, zumal viele der Module von Puppetlabs – das ist die Firma hinter Puppet – selbst gewartet werden und somit den hohen Qualitätsansprüchen entsprechen.

DSL: Schnörkellose Abstraktion

Puppet unterscheidet sich in zwei wesentlichen Punkten von Ansible. Erstens verfügt Puppet über eine eigene Sprache (DSL, domain specific language) – das darf auch nicht weiter verwundern, denn Puppet wurde vor fast 15 Jahren ersonnen und zu dieser Zeit waren DSLs sehr modern. Auch wenn die Lernkurve dadurch etwas steiler ist und das erste Deployment mit Puppet vielleicht nicht in einem Nachmittag zu bewerkstelligen ist, wird man mit einer sehr ausgereiften Sprache belohnt. Ein Vorteil der Puppet DSL ist sicherlich, dass der Fokus vom imperativen und sequenziellen Programmieren (zuerst das, dann jenes, ...) auf ein deklaratives Arbeiten verlegt wird. Nicht die Reihenfolge und die spezifische Ausformulierung der Schritte sollen im Vordergrund stehen, sondern die Beschreibung des gewünschten Endzustands. Ich deklariere, dass ein Paket x auf Host y installiert sein soll, und Puppet kümmert sich um die Umsetzung meines Wunsches, egal ob implizit vorrangige Schritte notwendig sind oder „yum/dnf/apt-get/pacman“ aufgerufen werden muss. Puppet weiß, was zu tun ist; die Puppet DSL steht sozusagen über den Niederungen der OS-Spezifika. Der „Visual Index“ *[3]* bietet einen schnellen Einstieg in die Sprache. In *Listing 2* wird die Sprache kurz anschaulich gemacht.

Master and Agent

Der zweite Unterschied besteht darin, dass Puppet von Beginn an eine Mas-

ter-Agent-Architektur verfolgt hat, was Ad-hoc-Deployments etwas schwieriger macht. Die Agents können nur vom Master aus gesteuert werden und akzeptieren nicht beliebige Befehle. Die Kommunikation zwischen Agent und Master ist verschlüsselt und die Konfiguration der Agents erfolgt zentral vom Master aus. Das bringt initial einen etwas höheren Installations- und Konfigurationsaufwand mit sich, zahlt sich dann jedoch recht schnell aus, weil sichergestellt ist, dass die Agents auf den zu verwaltenden Hosts nur Code ausführen, der auf dem Master hinterlegt worden ist. Und das ist gerade ab einer gewissen Teamgröße nur mit Vorteilen verbunden, weil über einfache Mechanismen sichergestellt werden kann, wer was wann am Puppet-Master verändert hat. In Kombination mit Framework r10k *[4]* können Git-Repositories dazu verwendet werden, Puppet-Manifeste (Puppet DSL mit der Datei-Endung .pp) wie Source Code zu verwalten („Configuration as Code"). Das Branchen innerhalb von Git findet seine Entsprechung im Konfigurationsmanagement, wenn der Puppet-Code im Branch „dev" nur auf jenen Hosts ausgeführt wird, die in der Gruppe „dev" stecken. Nichts steht hier dem Mantra „Release early, release often" im Wege. Sobald ein Push im Branch „dev" registriert wird, kann dadurch ein Puppet-Run auf allen Hosts der Gruppe „dev" getriggert werden, um zu sehen, ob der „Change" richtig „deployed" wurde.

Fazit: Puppet & Ansible

Es gibt sicherlich Platz für beide und die Überlappungen von Puppet und Ansible sind beträchtlich, beackern sie doch das gleiche Problemfeld. Tendenziell hat Puppet dort sein Stärken, wenn es darum geht, IT-Infrastruktur mittel- und langfristig über den gesamten Lifecycle hinweg zu betreuen. Das hierarchische Prinzip, die solide Codebasis und die zahlreichen zusätzlichen Management-Features in der Puppet Enterprise Edition *[9]* machen Puppet immer noch zu einem sehr attraktiven Produkt, auch wenn der Hype gerade anderswo stattfindet.

Mit seinem Designziel ist Ansible sehr leicht erlernbar und zuverlässig. Ansible ist sehr leicht auch mit Puppet integrierbar, aber auch mit zahlreichen anderen Tools, die seine Funktionalitäten erweitern, zum Beispiel Ansible Tower (AWX), Jenkins etc.

Ob jetzt Ansible oder Puppet oder ein anderes Automatisierungstool zum Einsatz kommt, ist letztlich nicht so wichtig. Wichtig ist, dass automatisiert wird. Über das „Wie", also das Tool, darf dann diskutiert werden. Die Notwendigkeit ist jedoch nicht bestreitbar...

Quellen

[1] *https://puppet.com*
[2] *https://forge.puppet.com*
[3] *https://puppet.com/docs/puppet/5.3/lang_visual_index.html*
[4] *https://puppet.com/blog/git-workflows-puppet-and-r10k*
[5] *https://www.ansible.com*
[6] *https://www.ansible.com/how-ansible-works*
[7] Manpage (1) ansible
[8] *http://docs.ansible.com/ansible/intro_installation.html*
[9] *https://puppet.com/products/puppet-enterprise*

Über die Autoren

Raphael Daum ist DBA, Automatisierungsgezeichneter und Anhänger des relationalen Datenmodells sowie endloser Rekursion sowie endloser ...

Florin-Catalin Enache ist DBA und System Administrator, der auf Oracle und Red-Hat-Technologien fokussiert ist. Er ist Oracle Certified Master und Red Hat Certified Engineer. In seiner Freizeit fährt er gerne Rad und hört Hörbücher.

Raphael Daum
raphael.daum@dbconcepts.at

Florin-Catalin Enache
catalin@enache.at

Petition der DOAG zur Modernisierung von Forms an Oracle

Oracle Forms wird dieses Jahr 40 Jahre alt. In den letzten Jahren sind bei vielen Oracle-Kunden erhebliche Investitionen in Ihre Forms-Entwicklungen geflossen. Mit den zukünftig geplanten Forms-Versionen ist ein langfristiger Support gesichert und Oracle bietet Forms weiterhin als mögliches PL/SQL-4GL-Werkzeug an.

Frank Hoffmann und die DOAG starteten als Initiative zur Modernisierung der Oracle-Forms-Technologie eine Petition. Die Befürworter dieser Petition fordern von Oracle eine zeitgemäße Modernisierung der Laufzeitumgebung und die Möglichkeit einer Cloud- und mobilfreundlichen Deploymentoption (JavaScript).

Weiterführende Informationen

Zum Hintergrund: *https://backoffice.doag.org/formes/pubfiles/11309466/docs/Presse/2019/2019-05-22_DOAG_Oracle_Forms_Petition_Info.pdf*

Die Petition: *https://www.doag.org/go/petition*

Automatisierung von Datenbank-Backups mit der Zero Data Loss Recovery Appliance

Ralf Zenses und Jan Brosowski, Oracle

Backups sind traditionell ein hochautomatisierter Bereich des Rechenzentrums-Betriebs. Im folgenden Artikel werden Backup-Ansätze mit ihren Vor- und Nachteilen vorgestellt sowie neue Herausforderungen bei der Backup-Automatisierung beleuchtet. Insbesondere wird auf die neue Technologie der Recovery Appliance eingegangen.

Seit vielen Jahren schon werden hierfür Automatismen eingesetzt:

- Scheduler starten automatisch Backup-Vorgänge einzelner Systeme oder Datenbanken.
- Backup-Clients erzeugen Snapshots auf Filesystemen, sie versetzen Datenbanken in Backup-Modes und deaktivieren diese wieder.
- Lifecycle-Management-Systeme sorgen dafür, dass nur notwendige Backups aufgehoben werden und effizient mit Speicherplatz umgegangen wird.
- Band-Überwachungssysteme prüfen die Qualität der eingesetzten Medien und veranlassen bei Bedarf deren Kopie oder sortieren sie aus.
- Nicht zuletzt fahren weltweit Millionen von Robotern automatisch die richtigen Medien zu Bandlaufwerken und sorgen so für den wohl auffälligsten – weil physischen – Teil der Automatisierung von Backup-Tätigkeiten.

All diese klassischen Methoden der Automatisierung sind wohlbekannt und werden immer wieder überarbeitet und verbessert.

Neue Herausforderungen an die Backup-Automatisierung

Diese klassischen Methoden der Backup-Automatisierung sind allerdings vor dem permanenten Wandel, den wir im Betrieb von Datenbanken beobachten, nicht

mehr ausreichend, sie müssen um weitere Automatismen ergänzt werden:

- Die Zeitfenster für die Wiederherstellung und der zulässige Datenverlust (RTO und RPO) werden kleiner: Die Folge sind häufigere Backups und strengere SLAs für die Wiederherstellung. Häufig wird sogar ein maximaler Datenverlust von nahe Null verlangt, was klassische Backup-Umgebungen an den Rand ihrer technischen Möglichkeiten bringt.
- Zudem werden die Fenster für Backup-Aktivitäten geringer. Gerade Netzwerk-Bandbreite wird zu einem knappen Gut in Rechenzentren. Während sich die Datenmengen in den letzten 20 Jahren mehr als vertausendfacht haben (statt weniger Gigabyte pro Datenbank spricht man von wenigen Terabyte pro Datenbank), hat sich die Netzwerkbandbreite gerade mal verzehnfacht (von Gigabit als Quasi-Standard im Jahr 2000 zu 10 Gigabit als Quasi-Standard heute). Das geht mit gewaltigen Bewegungen im Netzwerk einher, insbesondere wenn Full Backups der Datenbanken angelegt werden müssen.

Doch rückt auch die eigentlich wichtigste Funktion des Backups wieder in das Zentrum der Betrachtungen: der Restore. Was nutzt ein angelegtes Backup, wenn man sich nicht sicher sein kann, im Falle der Fälle darauf zurückgreifen zu können, und man statt der angestrebten Stunden Tage und Wochen auf die Wiederherstellung wartet?

Daher werden Funktionen wie eine permanente Prüfung der Backups, Methoden zur Sicherstellung der Restaurierbarkeit und Methoden zur Pflege der gespeicherten Daten zentral. Außerdem rücken Methoden zur Sicherstellung von garantierten Wiederherstellungszeiten in den Fokus.

Datenbanken nehmen dabei eine Sonderstellung ein, die sie von Filesystemen unterscheidet: Sie unterliegen heute meist einem 24/7-Betrieb, in dem es keine lastarmen Zeiten mehr gibt. Zudem speichern sie in vielen Fällen besonders kritische Daten, auf die mit einer Vielzahl von Systemen zugegriffen wird.

Full Backups, Deduplication und Incremental Backups

Im Laufe der Zeit wurden verschiedene Maßnahmen, die sich bei Filesystemen bewährt haben, auf Datenbanken übertragen. Um diese und ihre Auswirkungen auf Datenbank-Backups zu verstehen, sollen sie hier kurz beleuchtet werden.

Der klassische Weg des Backups ist der sogenannte Full Backup: Alle Dateien eines File-Systems oder alle Blöcke einer Datenbank werden auf ein Backup-System übertragen. Dies hat einen klaren Vorteil: Das Backup ist vollständig und konsistent. Auch kann es in vorhersagbarer Zeit vollständig zurückgesichert werden. Es hat jedoch zugleich einen großen Nachteil: Es ist sehr groß, insbesondere wenn mehrere Backup-Generationen aufgehoben werden sollen, und seine Übertragung in das Backup-System dauert lange.

Dieser Schwäche begegnete man mit dem Ansatz der Deduplizierung: Auf dem Backup-System nahm man das neue Full Backup entgegen und verglich die Inhalte mit den schon vorhandenen Backups. Waren zwei Blöcke oder Dateien identisch, wurde dieser Block oder diese Datei nur einmal abgelegt und aus mehreren Backups verlinkt. Somit sank zumindest der Speicherplatzbedarf, die Daten mussten dennoch übertragen werden. Heute verliert Deduplizierung immer mehr an Bedeutung, da sie nicht in der Lage ist, mit verschlüsselten Backups umzugehen. Dies ist allerdings gerade bei Cloud-Umgebungen, hochgradig konsolidierten Umgebungen oder bei der Verarbeitung personenbezogener Daten der Quasi-Standard für den Datenbankbetrieb geworden.

Um der Schwäche der zu übertragenden großen Datenmenge zu begegnen, erkannte man früh die Möglichkeit des sogenannten inkrementellen Backups: Statt aller Daten werden nur die Daten übertragen, die sich verändert haben. Bei File-Systemen ist dies ein weitverbreiteter Ansatz, und auch bei Datenbanken hat er sich bewährt. Leider geht er im Falle des Restore bei Datenbanken mit einem sehr unangenehmen Effekt einher. Statt nur eines einzigen Backups müssen mehrere in der richtigen Reihenfolge verwendet werden: Zunächst muss der letzte Full Backup zurückgespielt werden; anschließend müssen alle inkrementellen Backups zwischen dem letzten Full Backup und dem Zeitpunkt der Wiederherstellung angewendet werden. Dies kann eine Wiederherstellung unberechenbar langsam machen.

Daher ist in der Praxis eine Mischform aus regelmäßigen Full Backups (beispielsweise wöchentlich) mit täglichen inkrementellen Backups üblich. Sie kombiniert die Platzersparnis und kurze Laufzeit der inkrementellen Backups mit noch erträglichen Wiederherstellungszeiten, da maximal 6 inkrementelle Backups nach der Wiederherstellung des Full Backups angewandt werden müssen.

In vielen Fällen erreicht diese Strategie schon heute ihre Grenzen, gerade bei steigenden Anforderungen an Verfügbarkeit, bei weiterhin wachsenden Datenbanken und der nicht mitwachsenden Netzwerkbandbreite. Daher ist ein neuer Ansatz notwendig.

Incremental Forever mit Virtual Full Backups

Die Lösung hier ist eine andere Herangehensweise, die Oracle in der Zero Data Loss Recovery Appliance (kurz „Recovery Appliance“) umgesetzt hat. Wichtigstes Element ist hierbei eine gewisse Intelligenz der Recovery Appliance: Sie versteht den Aufbau einer Oracle-Datenbank aus ihren Datenblöcken.

Der erste Teil ist hierbei eine Strategie, die als Incremental Forever bekannt ist. Es wird einmalig ein Full Backup angefertigt und anschließend nur noch Incremental Backups. Somit sind die einzelnen Backup-Fenster kurz, die übertragene Datenmenge gering. Zwei wichtige Ziele sind also erreicht.

Um nun zu vermeiden, im Falle eines Restore alle Incremental Backups nacheinander anwenden zu müssen, ist die Recovery Appliance in der Lage, den Inhalt der Backups zu verstehen und auszuwerten. Dadurch kann sie auf Datenbank-Block-Ebene entscheiden, aus welchen Blöcken ein Full Backup zu einem bestimmten Zeitpunkt bestehen muss. Sie erzeugt dieses sogenannte Virtual Full Backup im Moment des Abrufes aus den Blöcken der vorliegenden Incremental Backups und des initialen Full Backups. Es wird also kein zusätzlicher Speicherplatz belegt, es werden lediglich Pointer angelegt. *Abbildung 1* gibt einen Überblick am Beispiel von 4 nacheinander durchgeführten Incremental Backups einer Datenbank, die aus 5 Blöcken besteht.

An Tag 0 wurde ein Full Backup angelegt, bestehend aus den fünf Blöcken A-E,

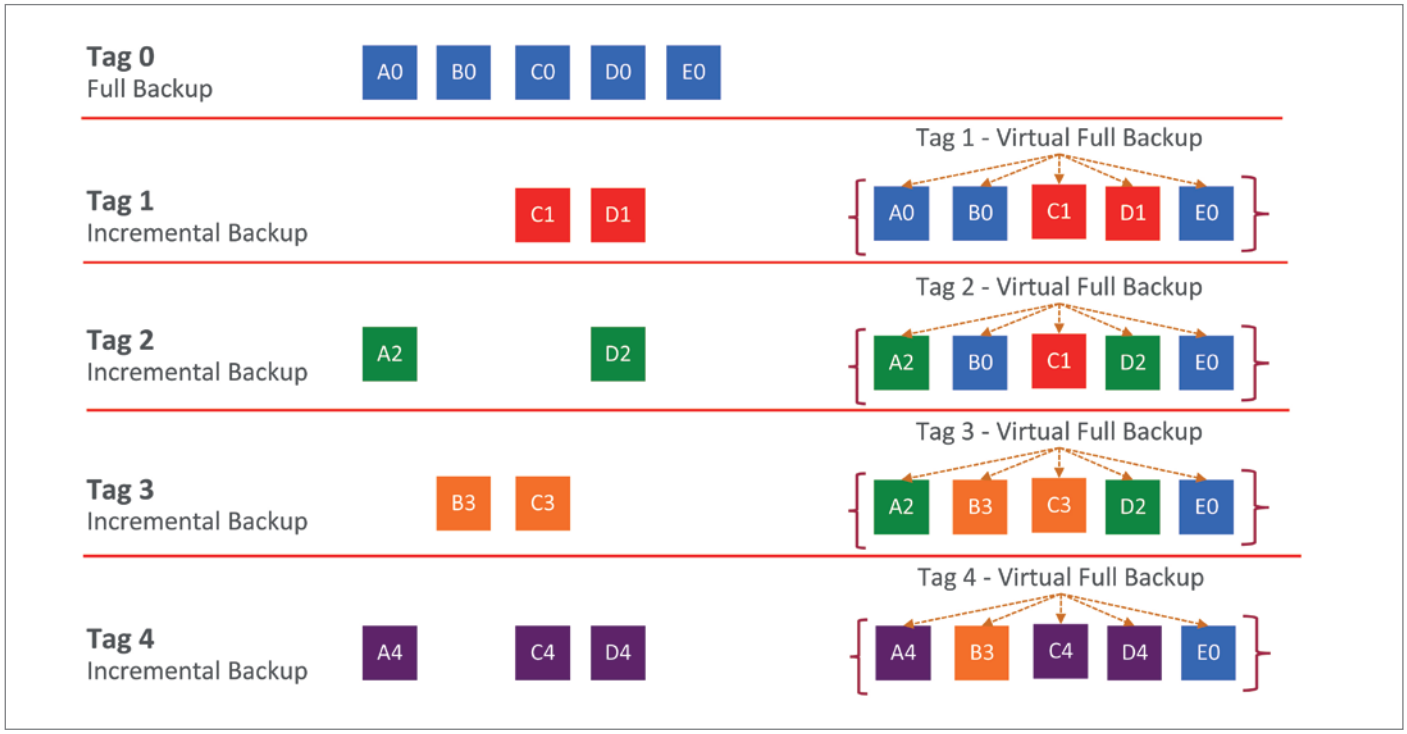


Abbildung 1: Strategie des Virtual Full Backups (Quelle: Ralf Zenses)

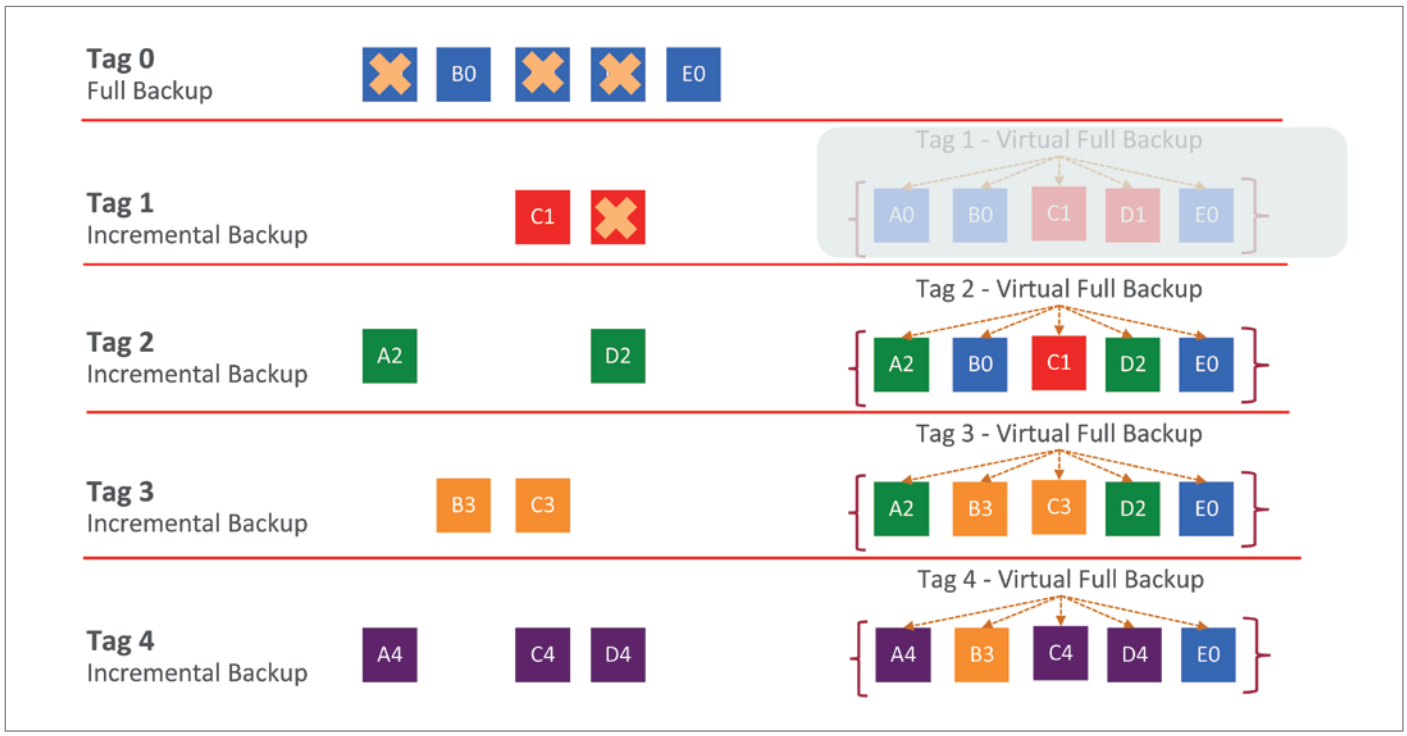


Abbildung 2: Gezieltes Löschen nicht mehr benötigter Datenblöcke (Quelle: Ralf Zenses)

jeweils mit dem Index 0 für Tag 0. An den folgenden Tagen wurden nur Incremental Backups angelegt, die jeweils aus den veränderten Blöcken bestehen (beispielsweise an Tag 3 die Blöcke B und C, jeweils mit Index 3 für den Tag 3).

Aus diesen Blöcken kombiniert die Recovery Appliance dann Virtual Full Backups. Man sieht beispielsweise, dass der Virtual Full Backup am Tag 3 aus den Blöcken A und D von Tag 2, den Blöcken B und C von Tag 3 sowie dem Block E von Tag 0 besteht.

Durch diese Logik wird auch vermieden, dass man alle Incremental Backups aufbewahren muss. Besteht die Notwendigkeit, den Zustand der drei letzten Tage wiederherstellen zu können, müssen nur die Blöcke aufbewahrt werden, die an einem dieser drei Tage Bestandteil eines Full Backups wären. Alle übrigen Blöcke können gelöscht werden. In *Abbildung 2* wird dieser Effekt dargestellt.

Im Beispiel sieht man, dass nur die Virtual Full Backups der Tage 2 bis 4 benötigt werden. Folglich sind nur die Blöcke, die in diesen Virtual Full Backups enthalten

sind, aufzubewahren. Dies sind die Blöcke A2 und A4, B0 und B3, C1, C3 und C4 und so weiter. Nicht mehr notwendig sind die Blöcke A0, C0, D0 und D1. Sie können gezielt aus den Backups entfernt werden.

Ein wichtiger Aspekt ist hierbei, dass die Blöcke so aufbewahrt werden, dass der jeweils letzte Stand schnell abrufbar ist – mit hoher Wahrscheinlichkeit wird er bei einem Restore benötigt. Daher werden die Blöcke so angeordnet, dass der jeweils aktuelle Zustand der Datenbank repräsentiert wird.

All diese Operationen geschehen automatisch im Hintergrund, die Administratoren brauchen sich um nichts zu kümmern.

Permanente Konsistenz-Überprüfung

Die Recovery Appliance muss durch diese Strategie Datenbankblöcke regelmäßig „anfassen“: Beim Entgegennehmen im Incremental Backup, beim Umsortieren zur Sicherstellung schneller Restore-Zeiten und bei der Analyse, ob Blöcke noch benötigt werden. Bei jedem dieser Schritte überprüft die Recovery Appliance die Konsistenz des Blockes durch Check-Summen. Somit können eventuell später auftretende Restore-Probleme frühzeitig erkannt und auch vermieden werden. Zum einen speichert die Recovery Appliance jeden Block mehrfach ab und kann so den Großteil der Probleme direkt selbst lösen. Sollten alle Kopien des Blocks defekt sein – beispielsweise durch Netzwerkfehler während des Incremental Backups – kann sie den Block erneut bei der Datenbank anfordern.

Somit werden automatisch die größten Probleme und größten Risiken eines fehlerhaften Restore ausgeschlossen und die Qualität des gesamten Prozesses deutlich erhöht.

Zero Data Loss

Der vollständige Name der Recovery Appliance besagt, dass es einen weiteren, ebenfalls automatisierten Baustein gibt, der die Zeit zwischen dem letzten Incremental Backup und dem Moment des Zusammenbruchs einer Datenbank abdeckt – nur dann kann man von einem Restore ohne Datenverlust sprechen.

Bei klassischen Verfahren sichert man zu diesem Zweck die sogenannten Archive Logs der Datenbank gesondert ab, da man mit diesen die Änderungen im genannten Zeitraum „nachziehen“ kann. Durch permanentes Sichern der Archive Logs sind so kürzere RPOs möglich, die sich aber immer noch weit von Null entfernt befinden.

Die Recovery Appliance nutzt hier eine Technologie namens Delta, über die Datenbanken permanent alle Schreib-Tätigkeiten an die Recovery Appliance replizieren. Dieser Datenstrom von Änderungen wird innerhalb der Recovery Appliance aufgezeichnet und es werden virtuelle Archive Logs daraus generiert. Diese können – ebenso wie die echten Archive Logs – für ein Nachziehen der Änderungen seit dem letzten Incremental Backup genutzt werden. Sie sind jedoch deutlich aktueller als die Archive Logs einer Datenbank, da sie annähernd synchron zum Datenbank-Betrieb mitgeführt werden.

Auch dieser Prozess ist vollständig automatisiert. Administratoren schalten ihn in der sogenannten Backup-Policy ein, die gesamte weitere Konfiguration wird automatisch durchgeführt.

Steuerung und Überwachung mittels Policies

Dies leitet zu einem wichtigen Bestandteil der Recovery Appliance über: der Steuerung und Überwachung der beschriebenen Mechanismen. Heute sind in Unternehmen Hunderte, wenn nicht Tausende von Oracle-Datenbanken im Einsatz; daher müssen nicht nur die Mechanismen selbst, sondern auch deren Einsatz und die Überwachung hochgradig automatisiert sein.

Daher nutzt die Recovery Appliance hier einen Policy-basierten Ansatz. Administratoren definieren einmal einen Satz von Regeln (Policies) dafür, welche Parameter bei Backup und Restore eingehalten werden sollen (RPO und RTO), sowie daraufhin, ob ein kontinuierliches Backup erfolgen soll. Diese Regeln bekommen eigene Namen (beliebt: Platin, Gold, Silber, Bronze. Andere Namen sind natürlich möglich).

Dann werden Datenbanken diesen Policies zugewiesen. Alle weiteren Konfigu-

	Klassische Full Backups in Standard-Backup-Umgebungen	Wöchentliche Full- + tägliche Incremental-Strategien in Standard-Backup Umgebungen	Virtual Full + Incremental Forever auf Recovery Appliance
Sicherungsvorgang	Sehr langsam: gesamte Datenbank wird bei jedem Backup übertragen	Überwiegend schnell: Beim wöchentlichen Full Backup langsam, bei Incrementals schnell	Schnell: nur beim ersten Backup wird ein Full Backup angefertigt, dann nur noch schnelle Incremental Backups
Last auf dem Netzwerk beim Sicherungsvorgang	Hoch	Beim wöchentlichen Full Backup hoch	Niedrig, außer beim initialen Backup
Recovery-Vorgang	Schnell: ein Full Backup wird zurückgesichert	Langsam, da alle Incrementals nacheinander übertragen und angewendet werden müssen	Schnell: ein Virtual Full Backup wird zurückgesichert.
Speicherplatzbedarf	Sehr hoch, insb. bei verschlüsselten Daten	Gering	Sehr gering
Recovery-Garantie	Keine Prüfung der Konsistenz durch die Backup-Umgebung	Permanente Prüfung des Backups	
Zero-Data-Loss-Strategie	Ansatzweise möglich durch regelmäßige Sicherung von Archive Logs. Datenverlust reduziert in den Bereich von Minuten-Stunden	Teil des Konzepts durch Delta-Push. Datenverlust im Bereich Null bis wenige Sekunden	

Tabelle 1

rationen übernimmt dann die Recovery Appliance, sodass ein Administrator nur wenig machen muss.

In der Folge sieht der Administrator dann, wie die Einhaltung der Policies funktioniert: Er kann so frühzeitig erkennen, ob unter Umständen zusätzlicher Storage-Bedarf besteht oder ob es technische Probleme bei einzelnen Backups gibt.

Zusammenfassung

Ein neues Backup-Konzept wie das der Recovery Appliance mag auf den ersten Blick ungewohnt sein, doch bietet es durch seine hohe Automation deutliche Vorteile gegenüber traditionellen Ansätzen. Durch das Incremental Forever Backup werden wertvolle Zeit und Netzwerk-Ressourcen im Alltag gespart und durch die innovativen Virtual Full Backups wird das Rücksichern einer Datenbank deutlich beschleunigt und vereinfacht. Vielfältige Automatisierungen helfen dem Administrator, das Backup einfacher, aber auch sicherer zu gestalten. *Tabelle 1* fasst die wichtigsten Merkmale zusammen.

Eine letzte Besonderheit sei zum Abschluss noch genannt: die Integration mit RMAN. Datenbankadministratoren sind es gewohnt, mit RMAN zu arbeiten. Gerade in stressigen Situationen – wie dem Zurücksichern einer Datenbank aus dem Backup – ist es zentral, durch eine neue Technologie wie die Recovery Appliance keine zusätzlichen Stress-Auslöser einzuführen. Daher wurde die Recovery-Appliance so in RMAN integriert, dass sich die Befehle für die Administratoren nicht verändern. Die gesamte „Magie" der Recovery Appliance, die das Backup auf eine qualitativ neue Stufe hebt, wird in solch stressigen Situationen vollständig verborgen und läuft automatisiert ab. So können sich die beteiligten Menschen auf ihre Tätigkeiten konzentrieren.

Ralf Zenses
Ralf.Zenses@Oracle.com

Jan Brosowski
jan.brosowski@oracle.com

Datenbank in der Wolke – Teil 2: Infrastruktur-Management

Borys Neselovskyi, Opitz Consulting Deutschland

Die Cloud-Technologie ist heute nicht mehr wegzudenken. Viele Unternehmen nutzen bereits Public Cloud Services oder betreiben einige Teile ihrer Infrastruktur in der Cloud. Auch der Datenbankbetrieb wird immer häufiger von eigenen Rechenzentren in die Cloud verlagert. Die Vorteile der Cloud sind dabei vielfältig. Sie reichen von konkreten Kosteneinsparungen über die Entlastung von Fachkräften bis hin zu einem Mehr an Übersichtlichkeit und Flexibilität. Im ersten Teil dieser Artikelserie wurden bereits die Vorteile der Cloud diskutiert und danach Abrechnungsmodelle sowie verschiedene Dienste von Oracle im Detail vorgestellt (siehe Red Stack Magazin 2/19). In diesem Teil befasst sich die Serie ausführlich mit den Infrastrukturen in der Cloud. Teil 3 wird in Ausgabe 4/19 erscheinen und konkrete Kundenszenarien vorstellen.

Zwei Infrastrukturen in der Oracle Cloud

Die erste Generation der Cloud-Infrastruktur von Oracle trug den Namen „Oracle Public Cloud“. Bereits 2012 stellte Oracle einen Dienst namens „Schema Cloud Service“ bereit. Mit diesem Dienst war es möglich, eine Datenbank in der Cloud zu nutzen. Dabei handelte es sich um eine PaaS-Anwendung, das heißt, die Nutzer besaßen keinen Zugriff auf den Server und griffen lediglich auf die Datenbank zu. Zwei Jahre später wurde der „Database Cloud Service“ ins Leben gerufen. Ab diesem Zeitpunkt war es möglich, virtualisierte Datenbankserver in der Cloud zu erstellen und zu nutzen. Die Public-Cloud-Infrastruktur entsprach allerdings nicht allen modernen Sicherheits- und Verfügbarkeitsanforderungen. Im Jahr 2016 präsentierte Oracle eine neue Infrastruktur namens „Bare Metal Cloud“. Und im September 2017 benannte man dann beide Infrastrukturen um:

- „Oracle Public Cloud“ wurde zu „Oracle Cloud Infrastructure Classic“ (Abkürzung: OCI Classic)
- „Oracle Bare Metal Cloud“ wurde zur „Oracle Cloud Infrastructure“ (Abkürzung: OCI)

Die klassische Infrastruktur bietet eine ältere Generation von Hardware und Storage. Die Server werden in der virtualisierten Variante provisioniert. Als Datenbank-Storage fungiert ein sogenannter „Block Storage“, der per Netzwerk an den Datenbankserver angebunden ist und in zwei Varianten ausgeliefert wird: einer Variante für die Datenbanken mit „normalem“ I/O-Aufkommen und einer Variante für hohes I/O-Aufkommen.

Die Größe einer Datenbank kann bei der Provisionierung initial maximal zwei Terabyte betragen. Wenn der Platz nicht ausreicht, kann der Datenbank-Storage um bis zu zehn Terabyte vergrößert werden.

Die zweite Generation der Oracle Cloud bietet eine Infrastruktur mit moderner Hardwareausstattung und Netzwerklandschaft. In der OCI können Hosts virtualisiert oder physikalisch betrieben werden. Sie bietet ebenfalls die neuen Storage-Varianten. Der neue, NVME-basierte Flash Storage passt sowohl für die OLTP- als auch für die Data-Warehouse-Datenbanken. Der NVME Storage wird direkt an den Server angebunden, was die Netzwerklatenz zwischen Server und Storage minimiert. Neben dem NVME Storage steht auch der Block Storage zur Verfügung. Dieser ist in drei Varianten verfügbar, die jeweils für Datenbanken mit unterschiedlichen I/O-Anforderungen geeignet sind:

- Standard: für Datenbanken mit normalem I/O-Aufkommen
- High: für Datenbanken mit mittlerem I/O-Aufkommen
- Dense: für I/O-intensive Datenbanken

Mit dem Framework Virtual Cloud Network (VCN) haben Cloud-Nutzer die Möglichkeit, eigene Netzwerksegmente in der Cloud zu erstellen und diese in privaten Firmennetzwerken zu integrieren. So werden die in der Cloud und On-Premises betriebenen Netzwerkabschnitte als eine Einheit konfiguriert. Mithilfe von Cloud Tools können Netzwerkeinstellungen, Sicherheitsregeln und Firewall-Freigaben sehr präzise eingestellt werden.

Darüber hinaus ist die Gruppierung und Bündelung von Ressourcen mithilfe von virtuellen Abteilungen (Begriff: Compartment) möglich. So können separate Compartments für verschiedene Firmen oder Abteilungen erstellt und gepflegt werden. Zu einem Compartment gehören virtuelle Netzwerke, Server, Datenbanken und viele andere Ressourcen. Man definiert die Netzwerk- und Sicherheitsregeln, die in einem Compartment zur Geltung kommen.

Beide Cloud-Infrastrukturen bieten darüber hinaus Storage-Kapazitäten für die Speicherung von Daten:

- Der Object Storage ist für die Speicherung von Dateien jeder Art gut geeignet.
- Der Archive Object Storage ist für Datenbanksicherungen konzipiert. Der Storage kann an die Server via NFS-Freigabe eingebunden werden.

Da alte und neue Cloud-Infrastruktur in Aufbau und Architektur sehr unterschiedlich sind, unterscheiden sich auch die Server-Vorlagen der beiden Infrastrukturen. Im weiteren Verlauf des Artikels gehe ich genauer auf den Aufbau und den Betrieb der Datenbanken in beiden Cloud-Infrastrukturen ein.

Datenbank-Provisionierung in der Oracle Cloud Infrastructure Classic

In der klassischen Infrastruktur können Datenbanken in allen Variationen provisioniert werden, von Standard über Enterprise und High Performance bis Extreme Performance.

Shape	Anzahl Prozessoren (OCPU*)	RAM (GB)	Datenbank-Speicher (GB)
General Purpose Shapes			
OC3	1	7,5	Bis 1200
OC4	2	15	Bis 1200
OC7	16	120	Bis 1200
OC9	32	240	Bis 1200
High Memory Shapes			
OC1M	1	15	Bis 1200
OC3M	4	60	Bis 1200
OC5M	16	240	Bis 1200
OC9M	32	480	Bis 1200
High IO Shapes			
OCIO1M	1	15	400
OCIO3M	4	60	1600
OCIO5M	16	240	400

Tabelle 1: Shapes der Oracle Cloud Infrastructure Classic im Überblick (Achtung: Nicht alle Shapes stehen in allen Regionen oder Verfügbarkeitszonen bereit.)

Möglich ist der Aufbau sowohl von Single-Instance- als auch von RAC-Datenbanken. Auch die Replikation mittels Data Guard ist in der Classic-Infrastruktur verfügbar.

Die Datenspeicherung passiert in einem Cloud Block Storage, der per Netzwerk an die Datenbankserver angebunden ist. Man unterscheidet zwischen zwei Storage-Varianten: einer Variante für Datenbanken mit „normalem" I/O-Aufkommen und einer Variante für Datenbanken mit hohem I/O-Aufkommen.

Die virtuellen Server können auf zweierlei Weise erstellt werden:

- Als Database Cloud Service: In diesem Modell werden sowohl der Server als auch die Datenbank per Cloud-Oberfläche erstellt.
- Als Virtual Image: Der Server wird per Cloud-Mechanismen erstellt. Die Datenbank wird jedoch vom Nutzer mithilfe des Tools „dbca" oder händisch mithilfe von Skripten angelegt.

In *Tabelle 1* sind die wichtigsten Fakten über die ausgewählten Shapes-Varianten zusammengefasst. Eine vollständige Liste finden sie am Ende des Textes *(siehe unter [1])*.

Bei der Abrechnung von Datenbanken in der OCI Classic sind die Kosten für zwei Bereiche relevant:

- Kosten für die Infrastruktur: Server, Storage und Netzwerk
- Kosten für ein Datenbank-Paket: Eine ausgewählte Datenbank-Edition wird pro Prozessor (OCPU) auf Stundenbasis abgerechnet.

Die *Tabelle 2* bietet einen Überblick über die Kosten für die Infrastruktur in der OCI Classic:

Alle drei Finanzierungsmodelle, also PAYG, monatlicher Flex-Preis oder BYOL, können dabei in Anspruch genommen werden.

Datenbank-Provisionierung in der Oracle Cloud Infrastructure

In der neuen Cloud-Infrastruktur können alle Datenbanktypen mit allen Optionen provisioniert werden. Der große Unterschied zur OCI Classic besteht darin, dass in der OCI alle physikalischen Maschinen in der Cloud gemietet werden können. So können Ressourcen-Engpässe vermieden und die gesamte Leistung des Servers in Anspruch genommen werden. Die Shapes für dedizierte Server in der Cloud haben die Bezeichnung „Bare Metal Instances" und bestehen aus einer Kombination von Prozessoren, RAM und teilweise lokalem NVME (SSD) Storage. Wichtig ist dabei zu wissen, dass der gesamte physikalische Server in der Cloud gemietet werden kann. Da die Abrechnung von Datenbank-Leistungen auf Basis von verwendeten CPUs erfolgt, kann der Be-

Produkt	Pay as You Go (PAYG)
Compute Classic (1 OCPU pro Stunde)	0,0738 €
Compute Classic – High I/O (1 OCPU pro Stunde)	0,1107 €
Block Storage Classic (1 GB pro Monat)	0,0369 €
Block Storage Classic – High I/O (1 GB pro Monat)	0,0738 €
Erste statische IP-Adresse	Frei
Weitere statische IP-Adresse	0,0037 €

Tabelle 2: Kosten für die Infrastruktur in der OCI Classic (Stand: August 2018)

Shape	Anzahl Prozessoren (OCPU)	RAM (GB)	DB Storage – Type	DB Storage Gesamt (TB)	2/3-fache Spiegelung (TB)
BM.DenseIO1.36	2- 36	512	NVME	28,8	9,4/5,4
BM.DenseIO2.52	2- 52	768	NVME	51,2	16/9

Tabelle 3: Bare Metal Shapes der Oracle Cloud Infrastructure im Überblick

Shape	Datenbanklizenz	Pay as You Go (PAYG): Gehostete Umgebung pro Stunde	Monthly Flex: Gehostete Umgebung pro Stunde
BM.DenseIO1.36	Database Standard Edition	5,4263 €	3,6175 €
BM.DenseIO1.36	Database Enterprise Edition	6,1265 €	4,0843 €
BM.DenseIO1.36	Database Enterprise Edition High Performance	7,6143 €	5,0762 €
BM.DenseIO1.36	Database Enterprise Edition Extreme Performance	9,102 €	6,068 €
BM.DenseIO2.52	Database Standard Edition	9,3297 €	6,2198 €
BM.DenseIO2.52	Database Enterprise Edition	10,0298 €	6,6865 €
BM.DenseIO2.52	Database Enterprise Edition High Performance	11,5176 €	7,6784 €
BM.DenseIO2.52	Database Enterprise Edition Extreme Performance	13,0055 €	8,6703 €

Tabelle 4: Kostenüberblick Bare Metal Shapes. Achtung: Das Betreiben von Oracle Application Clusters (RAC) wird in den Bare-Metal-Cloud-Varianten nicht mehr unterstützt (Stand: September 2018).

Shape	Gesamter Storage	Nutzbarer Storage mit „Normal Redundancy“	Nutzbarer Storage mit „High Redundancy“
BM.DenseIO1.36	28.8 TB NVMe	DATA 9.4 TB RECO 1.7 TB	DATA 5.4 TB RECO 1 TB
BM.DenseIO2.52	51.2 TB NVMe	DATA 16 TB RECO 4 TB	DATA 9 TB RECO 2.3 TB

Tabelle 5: Nutzbarer Speicher für Bare-Metal-Datenbank-Systeme

Shape	Anzahl Prozessoren (OCPU)	RAM (GB)	DB Storage – Block	DB Storage – local – NVME (TB)	PAYG pro OCPU – Ein Monat
Standard Shapes	1 – 24	7 - 320	Bis 1 PB	-	41,22 €
Dense IO Shapes	4 – 24	60 - 320	Bis 1 PB	3,2 – 25,6	82,36 €

Tabelle 6: Virtual-Machine-Instanzen in der Oracle Cloud Infrastructure (Quelle [3] - Stand: August 2018)

Modell (X7)	DB Server	Prozessoren (OCPU)	RAM (GB) pro DB Server	Storage (TB) gesamt/ nutzbar	Kosten in Euro (PAYG): Stunden/Monat
Viertel-Rack	2	2 - 92	720	360/106	26,046/17363,98
Halbes Rack	4	4 - 184	720	720/212	52,092/38756
Volles Rack	8	8 - 368	720	1440/414	104,184/77512,9

Tabelle 7: Exadata Cloud Service im Überblick (Stand: August 2018)

trieb mit einer kleinen Anzahl von Prozessoren gestartet werden, aber alles innerhalb eines dedizierten Servers. Bei Performance-Engpässen können weitere Kerne aktiviert werden. *Tabelle 3* bietet einen Überblick über die Bare Metal Shapes *(siehe auch [2] - Stand: Oktober 2018)*.

Die Kosten für die Bare Metal Shapes beinhalten

- zwei aktivierte OCPUs
- und die Datenbank-Lizenz für zwei OCPUs.

Zusätzliche OCPUs können separat erworben werden. *Tabelle 4* zeigt einen Überblick über die Preise für die Bare Metal Shapes:

Was Sie über den lokalen Speicher in der OCI wissen müssen

In *Tabelle 5* finden sich Daten zur Kapazität des NVME Storage. Beachtenswert ist, dass der nutzbare Platz durch den Aufbau der Storage-Redundanz geringer wird, da die Daten in ASM zwei- oder dreifach gespiegelt werden. Die Tabelle zeigt, wie sich verschiedene Konfigurationen auf den nutzbaren Speicher für Bare-Metal-Datenbank-Systeme auswirken.

Wie auch in der klassischen Infrastruktur können in der OCI-Landschaft virtuelle Server provisioniert werden. Als Speicher für Datendateien steht je nach Shape-Variante entweder ein Block Storage oder der lokale NVME Storage zur Verfügung. In *Tabelle 6* sind Informationen über die virtuellen Shapes zu finden.

Wie administriere ich meine Umgebung in der Cloud?

Beim Database Cloud Service handelt es sich um ein klassisches Platform-as-a-Service-Angebot (PaaS). Bei diesem Modell kümmert sich der Cloud-Betreiber, in diesem Fall also Oracle, um die gesamte Cloud-Infrastruktur und stellt den Betrieb des Rechenzentrums sicher. Der Kunde selbst ist für die Datenbankserver inklusive aller Komponenten wie etwa Grid-Infrastruktur und Datenbank verantwortlich.

Standardaufgaben wie Datenbank-Backup, -Recovery oder -Patching können in der Cloud über die webbasierte Cloud-Konsole erledigt werden. Diese Aufgaben können auch auf Betriebssystem-Ebene mithilfe der folgenden Utilitys umgesetzt werden:

- Datenbank-Backup: Mit dem Tool „bkup_api“ können Datenbanksicherungen konfiguriert beziehungsweise automatisiert werden.
- Datenbank-Restore: Mit dem Unterbefehl „orec“ des Dienstprogramms „dbaascli“ können Cloud-Datenbanken aus der Sicherung wiederhergestellt werden.
- Patchen der Datenbank-Umgebung: Der Unterbefehl „patch“ des Dienstprogramms „dbaascli“ kann verwendet werden, um Patches anzuwenden.
- In einer Cloud-Umgebung, die eine RAC-Installation beinhaltet, kümmert sich die Utility „raccli“ um Backups, Restore und Patching der Datenbank.

Diese Cloud-spezifischen Aufgaben können in der Cloud-Konsole erledigt werden:

- Verwaltung von Cloud Accounts und Identity Management
- Verwaltung von kundenspezifischen Netzwerken
- Verwaltung von Sicherheitszertifikaten

Exadata Cloud Service

Die Exadata Machine wurde von Oracle für den Betrieb von sehr großen und un-

ternehmenskritischen Datenbanken entwickelt. Sie ist mit sehr modernen Hardware-Komponenten ausgestattet: Leistungsfähige Datenbank- und Storageserver verfügen über moderne Prozessoren und RAM. Im Storage-Bereich stehen sowohl herkömmliche Festplatten als auch SSD- bzw. Flash-Speicherkarten zur Verfügung. Die Kommunikation zwischen einzelnen Datenbank- und Storage-Servern erfolgt über das performante InfiniBand-Netzwerkprotokoll. Alle Exadata-Komponenten sind zudem redundant ausgelegt. Was Exadata besonders macht, ist eine Hardware-Optimierung für lange und komplexe Abfragen sowie für die Datenladeprozesse im OLTP- und Data-Warehouse-Umfeld. Die Intelligenz auf der Storage-Ebene ermöglicht das Aufbereiten von Daten direkt im Storage. Zum Datenbankserver werden nur die Ergebnisse geliefert. Aufgrund dieser Eigenschaften halte ich die Exadata für eine der besten Maschinen für den Betrieb von großen, unternehmenskritischen Oracle-Datenbanken.

Seit dem Jahr 2017 wird Exadata von Oracle in der Cloud angeboten. Anwender können seitdem die ganze Exadata Machine in der Cloud mieten, und zwar in drei Varianten: als Viertel-Rack, halbes oder volles Rack. Die Varianten unterscheiden sich in der Anzahl der Prozessoren und in der Größe des Storage. Analog zum Database Cloud Service können Datenbanken in den Versionen 11g, 12c und 18c betrieben werden. Als Software-Paket kann beim Exadata Cloud Service nur die Variante „Extreme Performance“ genutzt werden. Damit stehen alle Oracle- Eigenschaften und -Optionen inklusive In-Memory, HCC, Partitioning, Advanced Compression, RAC sowie Active Data Guard zur Verfügung.

Zuständigkeiten und Administration

Wie beim Database Cloud Service Bare Metall kümmert sich der Cloud-Nutzer lediglich um die Datenbank-Server, die virtualisiert sind. Alle anderen Komponenten werden vom Cloud-Betreiber administriert und gewartet.

Auch in einer Exadata-Cloud-Umgebung erfolgt die Administration der Datenbankumgebung über Cloud Tools. Darüber hinaus hat der Anwender den vollen Zugriff auf den Datenbank-Server und kann die herkömmlichen Tools einsetzen. Die Provisionierung von komplexen RAC- und Data-Guard-Umgebungen erfolgt mit wenigen Klicks über die Web-Oberfläche oder über das Kommandozeilen-Tool Cloud CLI.

Folgende Cloud Tools helfen beim Erledigen von administrativen Aufgaben auf Betriebssystem-Ebene:

- Für das Patchen des Exadata Cloud Service werden zwei Tools eingesetzt: „exadbcpatch“ und „exadbcpatchmulti“.
- Die Administration der Datenbankumgebung erfolgt über die Utility „dbaascli“.
- Die Verwaltung und Durchführung von Speicherungs- oder Restaurierungsaufgaben erfolgt mit den Tools „bkup“ und „bkupapi“.

Kosten

Die Abrechnung erfolgt auf Stunden- oder Monatsbasis. Man kann auch die bereits vorhandenen Lizenzen in der Cloud nutzen. Dabei handelt es sich um das Bring-Your-Own-Licence-Modell (BYOL).

(Mehr Informationen über den Exadata Cloud Service finden Sie in den Quellenangaben am Textende unter [4], [5] und [6]).

In *Tabelle 7* sind die wichtigsten Informationen über den Exadata Cloud Service zusammengefasst (Stand August 2018):
Wichtige Hinweise:

- Bei der Nutzung von Exadata Cloud Service Shapes werden für den ersten Monat 744 Stunden in Rechnung gestellt, unabhängig vom tatsächlichen Verbrauch. Für die laufende Nutzung derselben Instanz nach dem ersten Monat werden nur die genutzten Stunden berechnet. Für zusätzliche OCPUs werden für den ersten Monat und für die laufende Nutzung aktive Stunden in Rechnung gestellt.
- Die Kosten für Exadata Infrastructure sind für BYOL die gleichen wie für PAYG für die X7-Shapes.
- Monthly Flex: Für verschiedene Vertragslaufzeiten werden folgende Ermäßigungen seitens Oracle gestattet *(siehe Tabelle 8)*:

Laufzeit in Jahren	Ermäßigung auf den Listenpreis (%)
1	20
2	25
3	30
4	35

Tabelle 8: Ermäßigung bei der Variante Monthly Flex (Stand August 2018)

Die Spalte „Kosten“ in *Tabelle 9* zeigt die Kosten für die Exadata ohne Prozessoren. Jede CPU muss extra aktiviert werden. Die Kosten für die OCPU betragen 2,188 € pro Stunde und Prozessor (Stand August 2018).

Cloud@Customer

Die Angebote Cloud@Customer oder die Exadata Cloud Machine eignen sich ideal für Kunden, die Cloud-Vorteile wünschen, ihre Datenbanken aufgrund von Gesetzen oder Richtlinien jedoch nicht in die öffentliche Cloud verschieben können. Die Exadata Cloud Machine kombiniert eine leistungsfähige Datenbankplattform mit der Einfachheit, Agilität und Elastizität einer Cloud-basierten Bereitstellung. Der Service ist nahezu identisch mit einem öffentlichen Cloud-Dienst. Die Exadata befindet sich jedoch in den eigenen Rechenzentren der Kunden und wird von Spezialisten für Oracle Cloud verwaltet. *(Mehr Informationen dazu finden Sie unter [7], [8] und [9].)*

Autonome Datenbanken

Auf der Oracle Open World 2017 kündigte Larry Ellison die autonome Cloud-Datenbank an. Die autonome Datenbank soll Tätigkeiten wie Administration, Tuning, Patching, Backup und Fehlerbehebung ohne menschliches Eingreifen voll automatisiert durchführen und sie lernt dank Machine Learning aus Fehlern. Im Rahmen dieses Angebots verspricht Oracle eine sehr hohe Verfügbarkeit der Datenbank. Die geplante und ungeplan-

Produkt	Pay as You Go (PAYG)	Monatlicher Flexpreis	Metrik
Oracle Autonomous Transaction Processing	2,188 €	1,4587 €	OCPU pro Stunde
Oracle Autonomous Database, Exadata Storage	192,7404 €	128,4936 €	Terabytes-Storage-Kapazität pro Monat
Oracle Autonomous Data Warehouse Cloud, Extreme Performance Edition	2,188 €	1,4587 €	OCPU pro Stunde
Oracle Autonomous Data Warehouse Cloud, Exadata Storage	192,7404 €	128,4936 €	Terabytes-Storage-Kapazität pro Monat

Tabelle 9: Autonomous Database Service in Überblick – Stand: September 2018 (Weitere Informationen über die Kosten in [12] und [13])

te Downtime soll nicht mehr als 0,005 % der Laufzeit betragen. Da bei einer autonomen Datenbank viele Aufgaben automatisch und ohne menschlichen Aufwand betrieben werden, kann Oracle einen moderaten Preis für den Service veranschlagen.

Es existieren zurzeit zwei Varianten der autonomen Datenbank:

- Autonomous Transaction Processing *(Details siehe [10])*
- und Autonomous Data Warehouse *(Details siehe [11])*.

Die autonome Datenbank bietet sich für neue Datenbank-Anwendungen an. Bereits entwickelte große DWH- oder OLTP-Anwendungen, die sehr stark „customized" sind und eine Menge „selbst gestrickten" Code beinhalten, können von der automatisierten Datenbank so schnell nicht profitieren.

Die Preise für beide Dienste setzen sich aus zwei Metriken zusammen:

1. Extreme Performance Edition pro OCPU
2. Terabyte-Storage-Kapazität pro Monat

Tabelle 9 gibt einen Überblick über die Kosten für die autonome Datenbank.

Fazit

Für neue Datenbank-Landschaften in der Oracle Cloud steht die zweite Cloud-Generation namens OCI bereit. Diese bietet mehr Optionen als die erste Generation und erfüllt in fast allen Bereichen moderne Sicherheits- und Performance-Anforderungen. Die vielfältigen Netzwerkoptionen ermöglichen eine nahtlose Integration der Cloud-Datenbanken in interne Kunden-Netzwerke. Zahlreiche Cloud Tools unterstützen den Nutzer bei administrativen Tätigkeiten und sorgen so für die Verringerung an Aufwand.

Da sich der Betrieb einer Datenbank-Landschaft in der Cloud von Standard-DBA-Aufgaben unterscheidet, sollten sich Cloud-Anwender mit der neuen Architektur allerdings zunächst vertraut machen. Die Betriebs- bzw. Ausfallkonzepte müssen an die Cloud angepasst und erprobt werden. Dann wird der Umzug in die Cloud leicht ausfallen und der Datenbankbetrieb sichergestellt.

Quellen

[1] *https://cloud.oracle.com/de_DE/compute-classic/pricing*
[2] *https://cloud.oracle.com/de_DE/database/bare-metal/pricing*
[3] *https://cloud.oracle.com/compute/pricing*
[4] *https://www.oracle.com/technetwork/database/exadata/exadataservice-ds-2574134.pdf*
[5] *https://cloud.oracle.com/en_US/infrastructure/database/features/exadata*
[6] *https://cloud.oracle.com/en_US/infrastructure/database/exadata/pricing*
[7] *https://cloud.oracle.com/en_US/database/exadata-cloudmachine/features*
[8] *https://cloud.oracle.com/en_US/database/exadata-cloudmachine/pricing*
[9] *http://www.oracle.com/technetwork/database/exadata/exacc-x7-ds-4126773.pdf*
[10] *https://cloud.oracle.com/de_DE/atp*
[11] *https://cloud.oracle.com/de_DE/datawarehouse*
[12] *https://cloud.oracle.com/de_DE/datawarehouse/pricing*
[13] *https://cloud.oracle.com/de_DE/atp/pricing*

Anmerkung des Autors: Oracle Cloud ist im ständigen Wandel und so ist ein Dienst, den wir in Teil 1 dieser Artikelserie vorgestellt haben, aktuell nicht mehr verfügbar: Der Exadata Express Cloud Service wird nicht mehr angeboten.

Borys Neselovskyi
borys.neselovskyi@opitz-consulting.com

Kubernetes: Eine effiziente Automatisierungslösung für Container

Michael Schulze, Opitz Consulting Deutschland

Dass modulare Infrastrukturen herkömmliche Serversysteme, aber auch komplexe virtuelle Umgebungen ablösen, ist allgemein bekannt. Dabei ist ein Trend hin zu verteilter und skalierbarer Software mit integriertem Monitoring zu beobachten. Das schlägt sich konkret nieder in der Nutzung von Microservices und der Kapselung von Anwendungen zur Erfüllung von Mandantenfähigkeit und Security-Aspekten. Wir haben es hier mit einem Paradigmenwechsel zu tun, bei dem Container-Architekturen wie Docker eine zunehmende Rolle spielen und monolithische Systemlandschaften nach und nach ablösen. Denn für eine modulare Verteilung ist es notwendig, Anwendungen zustandslos und versioniert über Container zur Verfügung zu stellen. Ein weiterer wichtiger Punkt ist die Skalierbarkeit, um unterschiedlichen Ressourcenanforderungen gerecht zu werden.

Diese Entwicklung ist in allen Software-Lösungen zu beobachten, und auch im Cloud-Umfeld ist sie längst angekommen. Modulare Lösungen benötigen zudem Ausfallsicherheit, Lastverteilung und eine gezielte Verwaltung der Ressourcen. Vor diesem Hintergrund sind Orchestrierungslösungen für Container-Architekturen entstanden, wie die populäre Orchestrierungsplattform Kubernetes, mit der sich der Artikel im Detail beschäftigen wird.

Grundlagen der Container-Architektur

Im Gegensatz zur Virtualisierung mit Hypervisoren, wo eine VM ein komplettes Betriebssystem für die Anwendungen bereitstellt, teilen sich in der Containerarchitektur mehrere Container den OS-Kernel des darunterliegenden Betriebssystems und nutzen somit dessen Infrastruktur. Dabei werden zur Container-Laufzeit alle notwendigen Prozesse der Anwendung isoliert im OS gestartet und verwaltet. Der Container selbst entspricht nur einer Datei, in der Anwendungscode, die Konfiguration und die Abhängigkeiten (z. B. notwendige Bibliotheken) gebündelt provisioniert werden. Dabei verwaltet eine zum OS kompatible Container-Plattform die bereitgestellten Containerdateien *(siehe Abbildung 1)*. Das spart Platz und ermöglicht die modulare, zustandlose sowie versionierte Bereitstellung und Kapselung von Anwendungen.

Die Container Runtime selbst nutzt zur Laufzeit dann die Inhalte des Containers in Interaktion mit dem darunterliegenden Betriebssystem. Dabei können mithilfe implementierter Netzwerkmechanismen verschiedene Container Runtimes miteinander kommunizieren *[1]*.

Container-Lösungen gibt es in verschiedenen Ausprägungen schon länger, aber erst Docker als Open-Source-Container-Implementierung revolutionierte ab 2013 den Markt und entwickelte sich seither zu einem Quasi-Standard insbesondere in Development-Umgebungen.

Diese Entwicklung machte den Weg frei für eine flexible, serverübergreifende Bereitstellung von modularisierten Anwendungsstrukturen (Microservices) auf verschiedenen Plattformen und Umgebungen.

Insgesamt vereinfacht sich durch den Einsatz von Container-Technologie das Provisionieren, die Änderung und die Erweiterung komplexer Anwendungen erheblich.

Neben der Container-Technologie Docker gibt es weitere Container Engines wie rkt, LXC, LXD, die jedoch bis heute in ihrer Verbreitung nicht die Bedeutung von Docker erreichen konnten *[2]*.

Kubernetes: eine Einführung

Wie bereits erwähnt, nimmt im Entwicklungsbereich die Notwendigkeit, Anwendungen einer großen Anzahl von Benutzern zur Verfügung zu stellen und komplexere Strukturen abzubilden, stetig zu. Diesen erweiterten Anforderungen begegnet man schon seit Längerem mit Container-Orchestrierungswerkzeugen. Durch ihren Einsatz wird es möglich, Cluster von Containern sicher zu betreiben sowie Lastverteilung und Skalierungsoptionen zu realisieren. Lösungen wie Docker-Swarm etablierten sich, wurden allerdings von der schnellen Entwicklung des Container-Orchestrierungstools Kubernetes überholt.

Kubernetes verwendet Docker als Standard Container Engine. Mit Kubernetes können jedoch auch weitere Container Engines wie rkt verwendet werden. Die enthaltenen Komponenten bieten eine effiziente Verwaltung des automatisierten Betriebs, des Deployments, Skalierungsoptionen sowie Monitoring und Wartung von Container-basierten Anwendungen. Durch die Clusterfunktionalität und die

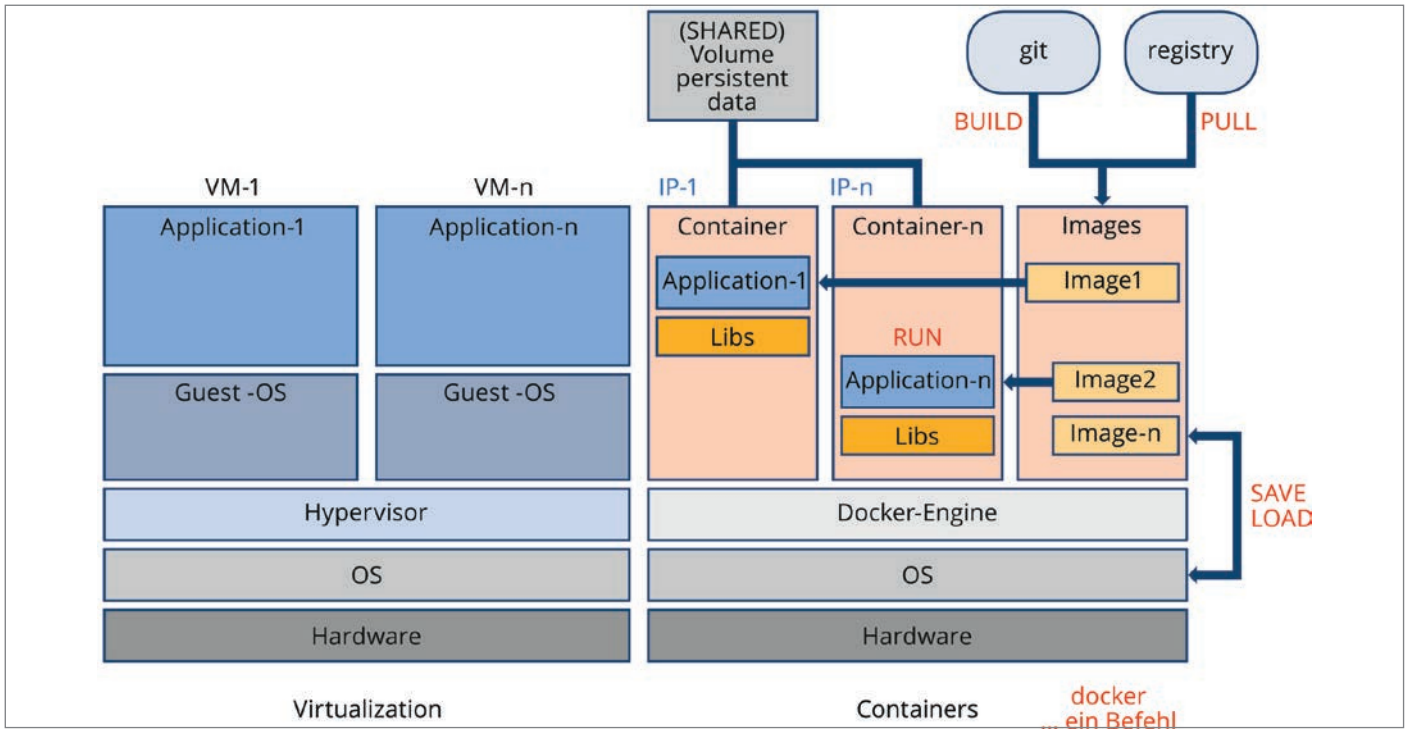


Abbildung 1: Aufbau einer Container-Architektur (Quelle: Michael Schulze)

Möglichkeit des Multi-Node-Betriebs können Container innerhalb des Clusters als eine Einheit verwaltet werden *[3] [4]*.

Die Geschichte von Kubernetes

Kubernetes wurde ursprünglich von Google entwickelt und intern eingesetzt (Projektname: Borg). Später stellte Google das Projekt in abgewandelter Form der Open-Source-Gemeinde zur Verfügung. Die Cloud Native Computing Foundation (CNCF), bei der Google Mitglied war, spielte für diese Entscheidung eine große Rolle. In dieser Stiftung sind mittlerweile so gut wie alle wichtigen Cloud Player vertreten *[5]*. Deshalb findet man Kubernetes-Lösungen heute auch in allen führenden Cloud-Plattformen wieder. Beispiele für den produktiven Einsatz von Kubernetes finden sich bei der New York Times, Philips, SAP und SaaS-Anbietern wie Box und GitHub sowie im Produkt Pokemon-Go *[6]*.

Kubernetes On-Premises

Die Core-Kubernetes-Pakete sind für die gängigsten Distributionen verfügbar. So lassen sich On-Premises komplexe Kubernetes-Cluster-Umgebungen (multinode) installieren. Neben der Core-Installation gibt es weitere auf Kubernetes basierende Management-Plattformen. OpenShift ist beispielsweise eine kommerzielle Lösung der Firma Red Hat, die (auch im DevOps-Kontext) um zusätzliche Komponenten erweitert wurde und als ganzheitliche Lösung zur Verfügung steht *[7]*. Ein Beispiel aus diesem Umfeld ist OpenDevStack, eine Open-Source-Plattform zur schnellen Bereitstellung von Microservices im DevOps-Kontext *[8]*. Für einen ersten Einstieg eignen sich One-Node-Kubernetes-Lösungen wie „minikube“ oder „minishift“, die eine auf bestimmte Ressourcen begrenzte, vollständige Kubernetes-Installation ermöglichen.

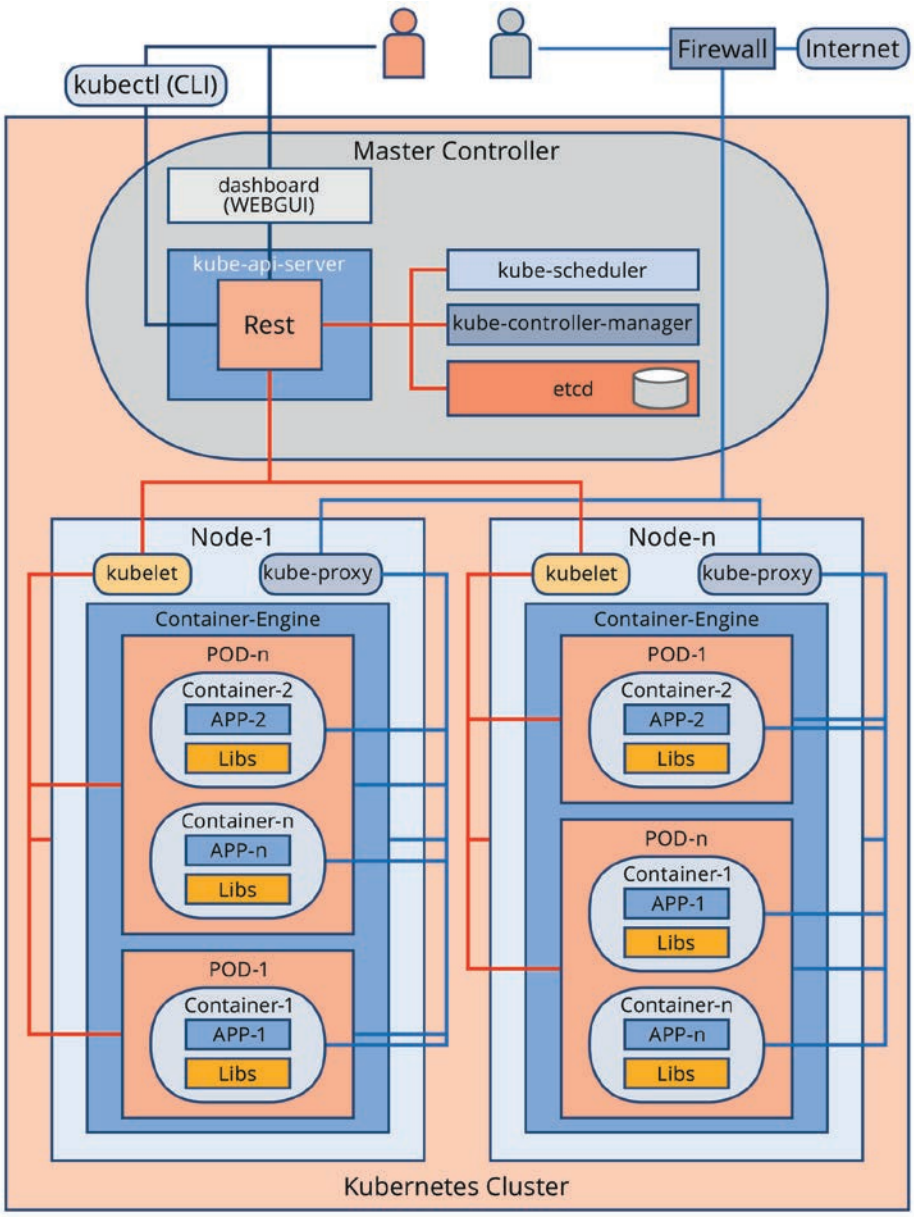


Abbildung 2: Kubernetes-Cluster (Quelle: Michael Schulze)

Kubernetes-Lösungen in der Cloud

Dass Kubernetes im Zeichen der Zeit steht, sieht man auch daran, dass aktuell führende Cloud-Anbieter unterschiedliche Lösungen in verschiedenen Ausprägungen für die Service-Bereitstellung von Kubernetes anbieten. Diese Cloud-Betreiber gehören der CNCF an, der Foundation, der das Kubernetes-Projekt ursprünglich von Google zur Verfügung gestellt wurde. Unter anderem die folgenden Kubernetes-Cloud-Lösungen stehen hier zur Verfügung *[9]*:

- Google: „GKE” (Google Kubernetes Engine)
  https://cloud.google.com/kubernetes-engine
- Amazon AWS: „EKS” (Elastic Kubernetes Service)
  https://aws.amazon.com/de/eks
- Microsoft Azure: „AKS“ (Azure Kubernetes Service)
  https://azure.microsoft.com/de-de/services/kubernetes-service
- Oracle Cloud: „OKE” (Oracle Kubernetes Engine)
  https://cloud.oracle.com/containers/kubernetes-engine
- IBM Cloud Kubernetes Service
  https://www.ibm.com/de-de/cloud/container-service

Kubernetes – Grundlagen der Architektur

Kubernetes besteht aus verschiedenen Core-Komponenten, die sich in Master- und Node-Komponenten sowie in verschiedene Add-ons aufteilen lassen *[10]*.

Diese werden im Folgenden näher erläutert.

Master Core Components

Der Master-Server besteht aus verschiedenen Komponenten:

- kube-controller-manager: Über den Controller Manager wird der Cluster gesteuert. Er kommuniziert mit den Node(s) und POD(s) und stellt die Ausfallsicherheit sowie die Skalierungseinstellungen (Replica) im Cluster sicher.
- kube-api-server: Über den im Master-Server integrierten API-Server einer weiteren Komponente wird eine REST-Schnittstelle zur Verfügung gestellt. Über diese Schnittstelle läuft die interne und externe Kommunikation des Kubernetes-Clusters. Beispiele wären hier die Status-Ermittlung von Ressourcen (GET) oder die Erstellung neuer Ressourcen (POST).
- etcd: Der API-Server kommuniziert mit dem etcd, dem Backend-Datenspeicher von Kubernetes. Diese Komponente besteht aus einem verteilten Open-Source-Schlüsselwertspeicher, in dem Kubernetes alle REST-Schnittstellen-Objekte und seine Zustände sichert. Der Schlüsselwertspeicher stellt das Speichern und das Replizieren von Daten sicher und bildet die Kernfunktion von Kubernetes.
- kube-scheduler: Der Scheduler entscheidet je nach Auslastungskriterien der ermittelten Statusinformationen, auf welchem Server-Knoten Ressourcen zur Verfügung stehen, und verteilt dann entsprechend *[11]*.

Node Core Components

Ein Node ist die Server-Komponente, die Ressourcen wie CPU, RAM, Netzwerk-Infrastruktur etc. für die dort provisionierten Container Runtimes zur Verfügung stellt. Nodes sind über ein Netzwerk miteinander verbunden und stellen so die Cluster-Kommunikation mit dem Master-Server sicher.

- kube-proxy: Für die Netzwerkkommunikation innerhalb des Clusters ist die Komponente kube-proxy zuständig. Sie enthält außerdem Routing-Funktionen, die Netzwerkkommunikation von und nach außen ermöglichen.
- kubelet: Auf jedem Node arbeitet ein Agent (kubelet) als Hauptprozess, der den Node und die dort laufenden Container-Runtime-Umgebungen hinsichtlich ihrer Ressourcen und Zustände überwacht und Indikatoren für den Master Controller zur Beurteilung des aktuellen Cluster-Status liefert. kubelet kommuniziert dabei mit dem kube-api-server auf dem Master-Server. Fällt eine Teilkomponente, etwa ein Container, aus, wird er nach den definierten Richtlinien auf einer anderen Ressource (Node/POD) wieder zur Verfügung gestellt.

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
      matchLabels:
          app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
      metadata:
          labels:
              app: nginx
      spec:
          containers:
          - name: nginx
              image: nginx:1.7.9
              ports:
              - containerPort: 80
```

Listing 1

POD

Ein POD (Deutsch: Hülse) ist die kleinste Einheit in einem Kubernetes-Cluster und kann als logische Host-Einheit verschiedene Anwendungscontainer enthalten. Diese Abstraktionsschicht ist auch notwendig, da mit Kubernetes verschiedene Container-Technologien, wie Docker und rkt, verwaltet werden können. Ein POD kann einen oder mehrere Container enthalten. Jeder POD besitzt eine eigene IP-Adresse, die enthaltenen Container nutzen diese gemeinsam *(siehe Abbildung 2)*.

Administrationstools CLI und GUI

kubectl ist die zentrale Commandline-Interface-(CLI)-Steuerung, um im Kubernetes- Cluster enthaltene Komponenten, zum Beispiel PODs, zu administrieren. Ebenso sind hier Möglichkeiten der Skalierung enthalten. kubectl-Befehle kommunizieren direkt mit der REST-Schnittstelle des Master-Servers (kube-api-server). Über diese Schnittstelle werden die Befehle dann direkt auf den Nodes ausgeführt.

Dashboard

Das in Kubernetes enthaltene Dashboard ist die Web-GUI-Schnittstelle, die einen großen Teil der Kubernetes-Funktionalität visualisiert. Auch hier können analog zu den verfügbaren CLI-Tools Parameter angepasst und neu definiert werden. Das Dashboard bietet als Steuerzentrale von Kubernetes umfangreiche Möglichkeiten sowie Monitoring-Funktionalitäten und ermöglicht so einen Überblick über die Kubernetes-Umgebung. Dabei nutzt es die REST-Schnittstelle auf dem Master Controller *[12]*.

YAML-Provisionierung am Beispiel von nginx

YAML („yet another markup language“) ist eine vereinfachte Beschreibungssprache und wird für die Definition von Deployment-Beschreibungen in Kubernetes verwendet. YAML bietet im Kubernetes-Kontext den Vorteil einer textbasierten,

versionierbaren Definition von Deployments und ihrer Parameter. Im YAML-File sind alle Informationen enthalten, die erforderlich sind, um den Container in der Kubernetes-Umgebung zu provisionieren und dort auszuführen. Da Kubernetes darauf basiert, komplexe Container-Applikationsumgebungen mit definierten Voreinstellungen in Kubernetes zu deployen, zu parametrisieren und zu ändern, vereinfacht das den Deployment-Prozess erheblich.

Das folgende Beispiel beschreibt die Bereitstellung einer Web-Tier-Umgebung auf einem nginx-Webserver und verdeutlicht die Funktionsweise der Mechanismen der Deployment-Bereitstellung in Kubernetes. Insbesondere das Zusammenspiel mit kubectl wird dadurch transparenter.

In der dargestellten Deployment-Beschreibung *(siehe Listing 1)* bestimmt das Image nginx:1.7.9, welches Container-Image aus dem Repository angewendet werden soll. Dieses wird dann in Kubernetes deployt und ausgeführt. Dabei sollen zwei Instanzen entstehen (Skalierungsoption) und der Webservice soll später auf Port 80 erreichbar sein.

Deployment-Vorgang

Die vorbereitete Deployment-Beschreibung durchläuft in der Anwendung für unser Beispiel die folgenden Schritte:

- Einlesen der Deployment-YAML-Beschreibung aus der Online-Ressource.
- Container-Image ermitteln (Repository).
- Container Runtime auf Node(s) und Pod(s) bereitstellen, entsprechend der definierten Skalierung, und starten.
- Anwendung steht nun zur Verfügung.

Die Umsetzung der genannten Schritte und die Kontrolle erfolgen durch das CLI-Interface kubectl *(siehe Listing 2)*.

Wie man sieht, wurde der Webserver, wie zuvor definiert, in zwei Instanzen bereitgestellt.

```
kubectl apply -f https://k8s.io/examples/application/deployment.yaml
kubectl get pods

Output:
deployment.apps "nginx-deployment" created

NAME                                READY     STATUS      RESTARTS   AGE
nginx-deployment-76bf4969df-279r4   1/1       Running         0       1h
nginx-deployment-76bf4969df-d796n   1/1       Running         0       1h
```

Listing 2

Zusammenfassung und Ausblick

Im Artikel wurde das Kubernetes Framework mit seinen Komponenten und Funktionalitäten vorgestellt; wir haben am Beispiel von nginx gesehen, wie einfach es ist, eine kleine Applikation im Cluster zur Verfügung zu stellen und Skalierungsoptionen anzuwenden. Mithilfe von YAML-Files als versionierbarer Grundlage für Deployment-Definitionen in Kubernetes können auch komplexe Container-Applikationsstrukturen gut verwaltet werden.

Die gestiegenen Anforderungen des Markts mit der Notwendigkeit einer schnellen, ausfallsicheren und flexibel skalierbaren Bereitstellung von Anwendungen über Container-Lösungen und deren Verwaltung hat dazu geführt, dass sich Kubernetes als produktionsreife Orchestrierungslösung bereits etabliert hat.

Kubernetes bietet Cluster- sowie Skalierungsfunktionalitäten und stellt eine effiziente Managementlösung für Container-Landschaften dar. Laut einer Erhebung der Crisp-Research AG planen 57% der deutschen Unternehmen im Jahr 2019 den Einsatz von Kubernetes *[13]*. Für Kubernetes stehen On-Premises-Installationsoptionen sowie diverse Cloud-Varianten aller namhaften Anbieter zur Verfügung.

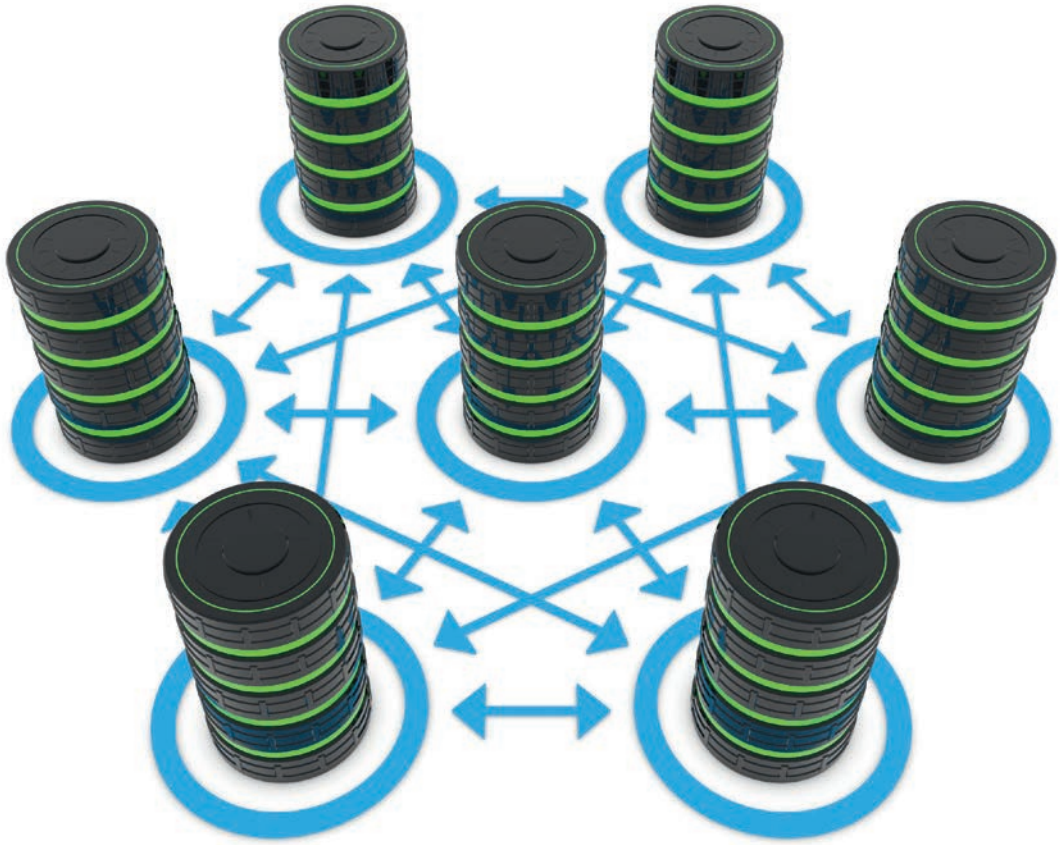
Damit wird Kubernetes in Zukunft sicher noch an Bedeutung gewinnen und sich langfristig weiter als Standard für Container-Cluster-Orchestrierung etablieren.

Orchestrierungswerkzeuge für Container-Lösungen einzusetzen, wäre der nächste logische Schritt, um Container-Umgebungen verwaltbar zu machen. Kubernetes spielt hier eine besondere Rolle. Als Open-Source-Projekt der CNCF steht das Projekt zudem unter dem Einfluss namhafter Cloud-Anbieter. Das garantiert eine permanente Weiterentwicklung.

Quellen

[1] *https://www.expertenderit.de/blog/7-linuxbasierte-container-plattformen-im-%C3%BCberblick*
[2] *https://www.docker.com/resources/what-container*
[3] *https://kubernetes.io/docs/reference/kubectl/overview*
[4] *https://kubernetes.io/docs/concepts*
[5] *https://www.cncf.io/blog/2017/12/06/cloud-native-technologies-scaling-production-applications*
[6] *https://blog.risingstack.com/the-history-of-kubernetes*
[7] *https://kubernetes.io/docs/concepts/overview/components*
[8] *https://www.redhat.com/de/technologies/cloud-computing/openshift*
[9] *https://kubernetes.io/docs/concepts/cluster-administration/cloud-providers*
[10] *https://coreos.com/etcd*
[11] OpenDevStack stellt Werkzeuge zur schnellen Provisionierung von Anwendungen bereit und wird bei Opitz Consulting bereits intern sowie auch bei einigen Kunden erfolgreich eingesetzt. *https://www.opitz-consulting.com/portfolio/software-development/devops-services/opendevstack.html*
[12] *https://github.com/kubernetes/dashboard*
[13] *https://www.crisp-research.com/kubernetes-und-cloud-native-trends-2019-das-jahr-de*

Michael Schulze
michael.schulze@opitz-consulting.com



Oracle Multitenant – was man wissen muss

Ulrike Schwinn, Oracle

Seit der Ankündigung mit Oracle Database 12.1, dass die Non-CDB-Architektur „deprecated“ ist, führt eigentlich kein Weg mehr an Oracle Multitenant vorbei. Oracle-Datenbank mit der Multitenant-Architektur sollte daher die bevorzugte Architektur sein, wenn es um die Datenbankkonfiguration ab Datenbank Release 12c geht. Was bedeutet das nun für den Betrieb? Was muss man wissen, wenn man mit der Multitenant-Architektur arbeitet?

Oracle Multitenant ist mit Oracle 12.1 – das heißt also vor zirka 6 Jahren – eingeführt worden. Viele Features und Erweiterungen wurden seit der initialen Einführung veröffentlicht. Anfängliche Feature-Restriktionen sind mittlerweile aufgehoben worden. Abgesehen von der Dokumentation – es gibt sogar ein Handbuch, das ausschließlich die Multitenant-Architektur beschreibt – findet man viele White Paper und sehr viele Blogeinträge zu den Funktionen und Features im Internet. Einige Beispiele sind auch im letzten Abschnitt des Artikels aufgelistet.

Aber vorab für alle diejenigen, die mit dem Multitenant-Thema starten wollen: Keine Angst — Oracle Multitenant nutzt Standard-Oracle-Technologien. Es gibt zwar einige hilfreiche Syntaxerweiterungen, aber vieles ist gleichgeblieben.

Was ist eine Pluggable Database eigentlich?

In einem Satz formuliert: Oracle Database 12c unterstützt eine neue Architektur, mit der viele „Subdatenbanken“ innerhalb einer einzigen „Superdatenbank“ verwaltet werden können. Die offizielle Terminologie für die Superdatenbank lautet „Container-Datenbank“ (kurz CDB), die für die Sub-Datenbank „Pluggable Database“

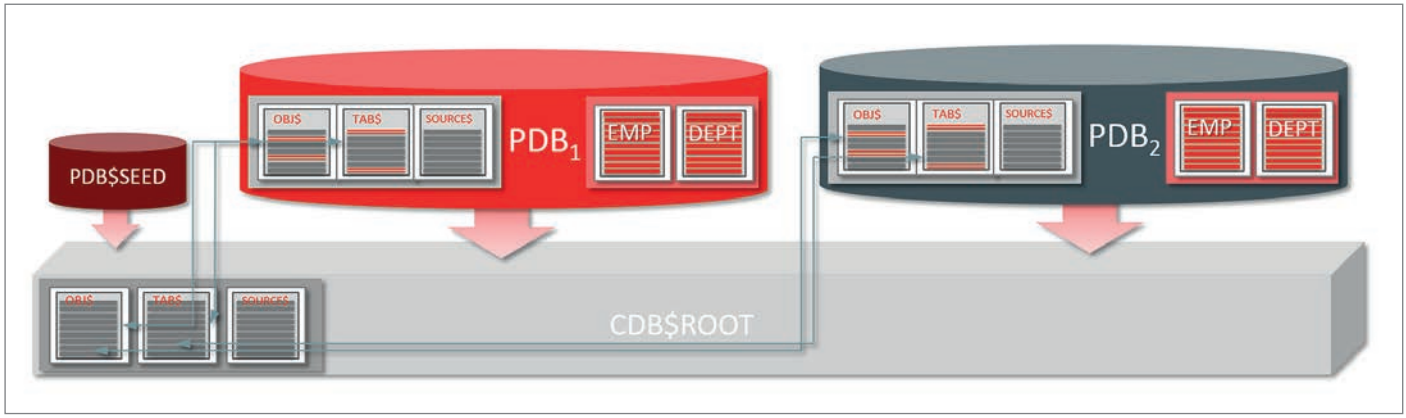


Abbildung 1: Data Dictionary und User-Daten (Quelle: Ulrike Schwinn)

(kurz PDB). Eine CDB ist eine Oracle-Database-Installation mit mindestens einer PDB. Jede Oracle-Datenbank ist entweder eine CDB oder eine NonCDB. Mit anderen Worten, mit der neuen Architektur können viele PDBs in einer einzigen CDB existieren. Um genauer zu sein: Bis zu 252 PDBs sind in einer CDB möglich. Verwendet man die Exadata-Plattform oder die Cloud-Editionen, sind sogar bis zu 4096 PDBs möglich.

Alle User-Daten werden dabei in einer „self-contained" Pluggable Database gespeichert, die die User-Tabellen mit Daten und ein Data Dictionary für die User-Schema-Informationen enthält. Ein ROOT-System (auch kurz ROOT) enthält dabei nur Oracle Data Dictionary-Informationen mit den Metadaten aller PDBs. Das Ganze wird durch eine horizontale Partitionierung im Data Dictionary realisiert. So kann beispielsweise auch die Portabilität der PDBs durch die Plug-in-Funktionen ermöglicht werden.

Die sogenannte „Seed PDB", eine Art Template für vom User erzeugte PDBs, ist read-only und steht in jedem Container zur Verfügung *(siehe Abbildung 1)*.

Der Begriff „Container" wird entweder für ein ROOT-System oder für eine PDB verwendet. Es gibt übrigens auch eine Single-Tenant-Ausprägung. Hier handelt es sich um ein ROOT-System mit genau einer userdefinierten PDB. Single Tenant kann ohne Lizenzierung der Multitenant-Option in der Enterprise- oder der Standard Edition verwendet werden. Möchte man die Anzahl der PDBs begrenzen, kann man den Initialisierungsparameter MAX_PDBS verwenden, der die Anzahl der userdefinierten PDBs begrenzen kann.

Wie sind solche PDBs nun aufgebaut? Wie steht es um den Speicherplatz? Jede PDB hat eigenen Satz an Tablespaces mit SYSTEM- und SYSAUX-Tablespaces. PDBs teilen sich REDO- und Control-Datei und (s)pfile. Standardmäßig hat die CDB einen einzigen TEMP-Tablespace. Allerdings kann man PDBs auch so konfigurieren, dass jede PDB ihren eigenen TEMP-Tablespace besitzt, was sicherlich auch zu empfehlen ist. UNDO-Tablespaces können entweder den CDB-UNDO-Tablespace gemeinsam (LOCAL UNDO OFF) verwenden, oder jede PDB hat ihren eigenen UNDO-Tablespace (LOCAL UNDO ON). LOCAL UNDO ON hat den Vorteil, dass man Features wie Flashback und Hot Clone verwenden kann. Dies ist die Standardeinstellung in allen Oracle-Cloud-Umgebungen. Die Festlegung erfolgt dabei beim CREATE DATABASE oder auch nachträglich mit einem „ALTER DATABASE"-Kommando. Letzteres bedarf allerdings eines vorangegangenen STARTUP MIGRATE. Möchte man überprüfen, wie diese Einstellung in der eigenen Umgebung lautet, kann man die nützliche Data Dictionary View DATABASE_PROPERTIES zurate ziehen *(siehe Listing 1)*.

Was bedeutet Multitenant eigentlich für das Oracle Data Dictionary? Wie man

```
SQL> SELECT property_name, property_value
  FROM database_properties
  WHERE property_name = 'LOCAL_UNDO_ENABLED';
PROPERTY_NAME         PROPERTY_VALUE
--------------------  ---------------
LOCAL_UNDO_ENABLED    TRUE
```

Listing 1: Undo-Mode-Einstellungen einer PDB

```
SQL> SELECT NAME
  FROM V$SYSTEM_PARAMETER
  WHERE ISPDB_MODIFIABLE='TRUE'
  ORDER BY NAME;

NAME
----------------------------------------
O7_DICTIONARY_ACCESSIBILITY
adg_account_info_tracking
approx_for_aggregation
approx_for_count_distinct
approx_for_percentile
aq_tm_processes
asm_diskstring
awr_pdb_autoflush_enabled
bitmap_merge_area_size
blank_trimming
…
```

Listing 2: Parameter, die auf PDB-Ebene änderbar sind

sich gut vorstellen kann, sind die „alten" Views nicht nur erweitert, sondern es sind neue Views eingeführt worden. Diese Views besitzen das Präfix CDB. Im Unterschied zu den herkömmlichen Views, die mit den Präfixen USER_, ALL_ oder DBA_ beginnen, geben sie einen Überblick über alle PDBs der Multitenant Container Database. So gibt CDB_TABLESPACES beispielsweise alle Tablespaces einer Multitenant Container Database aus; CDB_USERS gibt die User aller PDBs aus. Die neue Spalte CON_ID enthält dabei die eindeutige ID der PDBs: Der Wert 0 steht dabei für ROOT und 1 für das SEED-Template. Darüber hinaus gibt es auch noch neue Views wie zum Beispiel CDB_PDBS, die über alle PDBs in einer CDB Auskunft gibt, oder aber auch PDB_PLUG_IN_VIOLATIONS, die die Inkompatibilitäten zwischen einer PDB und der CDB ausgeben kann.

Wie steht es nun um die Initialisierungsparameter? Sind die Parameter auf PDB- Ebene änderbar? Nicht alle Parameter lassen sich auf PDB-Ebene mit einem ALTER-SYSTEM-Befehl ändern. Um schnell einen Überblick über die Untermenge von Parametern zu erhalten, eignet sich eine Abfrage auf V$SYSTEM_PARAMETER.

Typische Einsatzgebiete sind beispielsweise Memory-Einstellungen, die speziell für die PDBs gelten sollen *(siehe Listing 2)*.

Wenn der aktuelle Container eine PDB ist, wird das übliche Kommando ALTER SYSTEM SET <initialization_parameter> genutzt, um die Einstellung in der PDB zu ändern. Die Anweisung wirkt sich nicht auf ROOT oder die anderen PDBs aus. Auch hier kann man die Option SCOPE mit den üblichen Werten verwenden. Beachten sollte man nur, dass keine PDB-spezifischen Änderungen in den TEXT-Pfile geschrieben wird.

Die ersten Schritte

Nun stellt man sich natürlich die Frage, wie es mit der User-Verwaltung in einer solchen CDB aussieht. Wie kann man sich mit einer PDB verbinden? Um es gleich vorwegzunehmen: Eine Verbindung mit „AS SYSDBA" ist nicht mehr die bevorzugte Art, um sich mit der Datenbank als DBA zu verbinden. Stattdessen gibt es das neue Konzept des Common User. Ein Common User ist ein Datenbankbenutzer, der die gleiche Identität im ROOT-System und in jeder bestehenden und zukünftigen PDB hat. Jeder Common User kann sich mit ROOT und jeder PDB, in der er über Berechtigungen verfügt, verbinden und Operationen durchführen. Die User SYS und SYSTEM sind Oracle-definierte Common User. Möchte man weitere Common User anlegen, müssen diese mit dem Präfix c## oder C## beginnen. Die altbekannten Datenbank-User wie SCOTT usw. werden nun als Local User bezeichnet und in der jeweiligen PDB mit den üblichen Kommandos erzeugt.

```
SQL> connect system/"<password>"
Connected.
SQL> show con_name
CON_NAME
------------------------------
CDB$ROOT
SQL> alter session set container=pdb1;
Session altered.
SQL> select sys_context('USERENV', 'CON_NAME') from dual;
SYS_CONTEXT('USERENV','CON_NAME')
----------------------------------------------------------------
PDB1
SQL> connect system/"<password>"@pdb1
Connected.
SQL> show con_name
CON_NAME
------------------------------
PDB1
```

Listing 3: Verbindungsmöglichkeiten

```
SQL> CREATE LOCKDOWN PROFILE sec_profile;
SQL> ALTER LOCKDOWN PROFILE sec_profile DISABLE FEATURE=('XDB_PROTO-
COLS');
SQL> ALTER SYSTEM SET PDB_LOCKDOWN = sec_profile;
```

Listing 4: Lockdown-Profile zur Verhinderung von HTTP und FTP

Wie verbindet man sich nun mit einem Container – der PDB oder dem ROOT-System? Die einfachste Methode ist die Verwendung von Service-Namen der PDB oder die Verwendung des sogenannten „Easy Connect". Ganz klassisch ohne Nennung von Service-Namen oder Easy-Connect-Verbindung kann auch ein „ALTER SESSION"-Befehl verwendet werden. Eine Überprüfung kann über SQL*Plus- Variablen oder auch über den „SYS_CONTEXT"-Befehl erfolgen *(siehe Listing 3)*.

Ein ganz neues Sicherheitsfeature heißt PDB Lockdown Profile. Damit können potenziell gefährliche Datenbankbefehle und -funktionalitäten, die die Isolierung beeinträchtigen könnten, verhindert oder eingeschränkt werden. Gemeint sind hier Datenbankbefehle wie ALTER SYSTEM, ALTER PLUGGABLE DATABASE, ALTER SESSION und ALTER DATABASE sowie Datenbank-Features wie die XMLDB. Soll eine Datenbankfunktionalität innerhalb einer Pluggable Database komplett ausgeschaltet werden, um zum Beispiel den Zugang über HTTP/S oder FTP erst gar nicht zu ermöglichen, kann dies durch das in *Listing 4* dargestellte Lockdown-Profil geschehen. Die Aktivierung erfolgt je nach Gültigkeitsreichweite entweder als Default Profile im ROOT-Container oder aber in der entsprechenden PDB.

Arbeiten mit Oracle Multitenant

Da eine Verbindung zu einer PDB mithilfe des Servicenamens erfolgt, ist kein großes Umdenken, wenn überhaupt, beim Arbeiten mit SQL*PLUS, SQL*Developer, SQL*LOADER oder Data Pump erforderlich. Zusätzlich sind in den grafischen Werkzeugen wie Enterprise Manager und SQL Developer weitere Sichtweisen beziehungsweise Kommando-Wizards eingeführt worden, um das Arbeiten mit Containern zu erleichtern *(siehe Abbildung 2)*.

Für den Database Resource Manager bedeutet die Multitenant-Architektur,

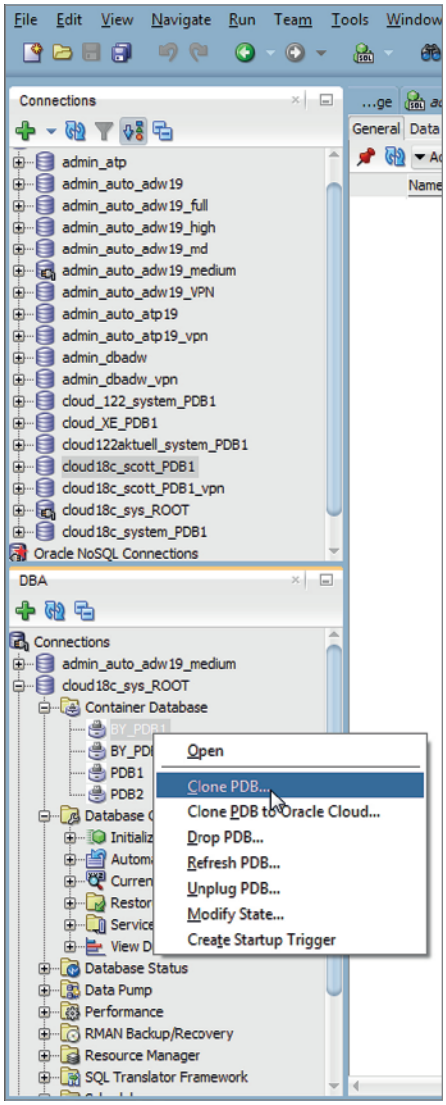


Abbildung 2: SQL-Developer-Menü bei ROOT-Verbindung (Quelle: Ulrike Schwinn)

dass nun Ressourcen-Pläne auf CDB- und PDB-Ebene erzeugt werden können. Auf CDB- Ebene können damit die Ressourcen wie CPUs und Parallel Execution Server pro PDB kontrolliert und damit PDBs über andere priorisiert werden. Auf PDB-Ebene selbst werden die Ressourcen auf Resource-Gruppenebene kontrolliert – ganz ähnlich wie mit einem Ressourcen-Plan in einer NonCDB. CDB-Ressourcen-Pläne werden dabei wie üblich über das Package DBMS_RESOURCE_MANAGER oder grafisch im Enterprise Manager erzeugt. Jede Plan-Direktive besteht aus dem Namen der PDB und sogenannten „Shares“, das sind Anteile der CDB-Ressourcen, die verfügbar für eine PDB sind. So kann beispielsweise festgelegt werden, dass einer PDB doppelt so viele Ressourcen zugewiesen werden wie einer zweiten PDB, indem der Wert für die Shares der ersten PDB doppelt so hoch wie für die zweite PDB ist. PDBs ohne einen spezifischen Ressourcen-Plan verwenden übrigens Default-Werte.

Im Backup- und Recovery-Umfeld hat sich nur wenig an den Kommandos verändert. Die Sicherung einer CDB ist im Wesentlichen identisch mit der einer NonCDB. Das Wichtigste, woran man denken sollte, ist, dass mit einer Vollsicherung der CDB auch eine Vollsicherung aller PDBs durchgeführt wird. Das Gleiche gilt dann natürlich auch für ein Recovery. Wie verbindet man sich mit RMAN? Eine Verwendung von RMAN TARGET sys stellt zum Beispiel eine ROOT-Verbindung her, ein RMAN TARGET sys@PDB1 die Verbindung mit der PDB mit Namen PDB1. Je nach der Art der Verbindung sind dann unterschiedliche Kommandos zum Erstellen von Backups möglich beziehungsweise erforderlich *(siehe Listing 5)*.

Es gibt natürlich auch die Möglichkeit, ein Flashback einer PDB durchzuführen. Dazu sind nur wenige Schritte erforderlich. Voraussetzung ist allerdings die Einstellung LOCAL UNDO ON. Übrigens Einstellungen wie ARCHIVELOG oder FLASHBACK sind Datenbankweite Operationen und können nur in der ROOT erfolgen. Überprüfen lassen sich diese Einstellungen wie immer über die View V$DATABASE.

Standardoperationen wie das Öffnen und Schließen einer PDB werden beispielsweise mit einem ALTER PLUGGABLE DATABASE OPEN /CLOSE durchgeführt. Damit nach dem Neustart der gesamten CDB die PDBs in ihrem aktuellen Zustand überführt werden, ist allerdings folgendes zusätzliche Kommando erforderlich *(siehe Listing 6)*.

Unerwähnt möchte ich auch nicht das neue Oracle Perl Skript catcon.pl lassen, das bei der Ausführung von Skripten in der gesamten CDB oder auch einzelnen PDBs hilfreich sein kann. Um die Funktionsweise zu erklären, soll im folgenden Beispiel das etwas angepasste Skript utlsampl.sql mit Namen utlsamplmult.sql in der PDB PDB1 und PDB2 durchgeführt werden *(siehe Listing 7)*. Beachten Sie bitte auch folgende Besonderheiten: Im Skript sollten keine expliziten EXIT Kommandos oder direkte Connections vorkommen und die PDBs PDB1 und PDB2 sollten geöffnet sein. Das Flag „-d“ gibt an, wo sich das Skript befindet. Das Flag „-l“ gibt das Verzeichnis an, in dem alle Logdateien erstellt werden sollen. Das Flag „-b“ gibt den Präfixnamen der zu erzeugenden Protokolldateien an.

```
RMAN> backup as compressed backupset database root; -- Nur Root Backup
RMAN> backup pluggable database PDB1,PDB2;          -- Backup PDB1, PDB2
RMAN> backup as compressed backupset database;      -- Gesamtbackup CDB
RMAN> BACKUP PLUGGABLE DATABASE pdb1 PLUS ARCHIVELOG;
```

Listing 5: BACKUP-Kommandos

```
alter pluggable database BY_PDB1 save state;
```

Listing 6

```
$ORACLE_HOME/perl/bin/perl5.22.0 catcon.pl -u sys/"Monchen__123456" -c
"PDB1 PDB2" -e -s -d $ORACLE_HOME/rdbms/admin -l /home/oracle -b utl-
sampl utlsamplmult.sql
```

Listing 7: Catcon.pl zur Ausführung von utlsampl.sql in PDB1 und PDB2

```
impdp system@PDB1
   NETWORK_LINK=DB1 VERSION=12 FULL=Y
   TRANSPORTABLE=ALWAYS METRICS=Y
   LOGFILE=log_dir:src112fullimp.log
   TRANSPORT_DATAFILES='/oradata/ts1.dbf' …
```

Listing 8

```
exec DBMS_PDB.DESCRIBE ('PDB1.xml');
```

Listing 9

```
create pluggable database PDB1
using ('PDB1.xml') …
```

Listing 10

```
start ?/rdbms/admin/noncdb_to_pdb.sql
```

Listing 11

Zusätzlich dazu existieren eine Reihe von neuen Operationen und damit auch SQL-Kommandos für das Anlegen, Löschen, Klonen, Refresh oder Relocate einer PDB. Die graphischen Werkzeuge wie SQL Developer oder Enterprise Manager können dabei eine gewisse Unterstützung bieten.

Von NonCDB nach CDB

Wie verwandelt man nun eine NonCDB in eine CDB? Die einfachste Methode ist sicherlich eine neue leere CDB zu erstellen und die Daten beispielsweise mit Data Pump Import zu importieren. Bei großen Datenmengen empfiehlt sich darüberhinaus die Transportable Tablespace Methode. Folgende Schritte sind erforderlich.

- Erzeugen einer neuen Datenbank/PDB
- Erzeugen eines Database Link zur Quell DB
- Read-Only Setzen der Tablespaces der Quell DB
- Kopie der Datendateien
- Import mit Data Pump *(siehe Listing 8)* beispielsweise mit

Eine weitere Möglichkeit ist, die Multitenant-Funktionalität Plugin zu verwenden. Ist die Datenbank zwar eine NonCDB aber auf dem Release-Stand 12c, könnte man dann folgendermaßen vorgehen.

- Setzen der Datenbank auf read-only
- XML-Datei für Metadaten erzeugen mit *(siehe Listing 9)*
- Shutdown der Datenbank
- Durchführung des Plug-ins *(siehe Listing 10)*
- Sanity Check *(siehe Listing 11)*

Fazit

Wenn es bisher auch noch nicht explizit erwähnt wurde: Weder das Anwendungs-Backend noch der Client-Code müssen für die Verwendung von Oracle Multitenant geändert werden. Das bedeutet auch, dass sich das Arbeiten mit einer einzelnen PDB nicht oder nur sehr wenig von dem mit einer herkömmlichen NonCDB unterscheidet. Hat man die Idee und das Konzept dahinter verstanden, ist es ein Einfaches, damit zu arbeiten und die Vorteile wie zum Beispiel das Konzept „Manage-as-one" zu nutzen. Darüber hinaus gibt es natürlich noch weitere interessante Funktionen wie etwa das Klonen, Refreshable PDB, Relocate, Plug-in und Plug-out, die in diesem Grundlagen-Artikel keine Erwähnung finden konnten. Möchte man mehr darüber erfahren, kann man einiges dazu im Handbuch oder in den White Papers finden. Folgende Liste zeigt eine kleine Auswahl an weiterführenden Links und Literatur zum Thema:

- Dojo zum Thema Multitenant: *https://blogs.oracle.com/coretec/dojo7*
- Deutschsprachiger Technologie-Blog: *https://blogs.oracle.com/coretec*
- Handbuch: Multitenant Administrator's Guide
- White Paper: Oracle Multitenant (Basics) *https://www.oracle.com/technetwork/database/multitenant-wp-12c-1949736.pdf*
- White Paper: Oracle Multitenant: New Features in Oracle Database 12.2 *https://www.oracle.com/technetwork/database/multitenant/overview/multitenant-wp-12c-2078248.pdf*
- White Paper: Oracle Multitenant: New Features in Oracle Database 18c *https://www.oracle.com/technetwork/database/multitenant/learn-more/multitenantwp18c-4396158.pdf*

Über die Autorin

Ulrike Schwinn studierte Mathematik und ist bei der Firma Oracle in München im Bereich Systemberatung tätig. In ihrer Funktion berät und schult sie Kunden in Fragen der neuesten Oracle-Datenbank und Cloud-Technologien wie zum Beispiel Oracle Autonomous Database. Sie wirkte an mehreren Oracle-Buchprojekten mit, veröffentlicht Blog-Postings im Internet und ist als Referentin auf IT-Kongressen vertreten.

Profile und Blog

- *https://www.linkedin.com/in/ulrikeschwinn*
- *https://twitter.com/uschwinn*
- *https://www.xing.com/profile/Ulrike_Schwinn*
- *https://blogs.oracle.com/coretec*

Ulrike Schwinn
Ulrike.Schwinn@oracle.com

Systematische Datenbank-Performance-optimierung als Projektaufgabe

Jörg Stahnke, Ppi

Performance wird oft nur als Projektrisiko betrachtet. Wenn Probleme auftreten, werden diese „ad hoc" gelöst. Der Artikel zeigt, wie man Performanceprobleme in einer Oracle-Datenbank durch ein systematisches Vorgehen von Anfang an vermeiden kann.

Sowohl für agile Projekte als auch bei Wasserfallprojekten gibt es häufig pro Sprint bzw. Release den in *Abbildung 1* dargestellten Ablauf:

Im Rahmen der Konzeption konzentriert man sich auf fachliche Anforderungen. Im Entwicklertest wird nur mit geringen Datenmengen gearbeitet. Erst im Abnahme-/ Integrationstest werden größere Datenmengen getestet und „ganz plötzlich" stellt man Performanceprobleme fest. Manchmal treten diese Probleme auch erst im Produktivbetrieb auf. Die Performanceprobleme werden erst sehr spät erkannt und man hat nur noch wenig Zeit. Deshalb bleiben oft nur zwei Varianten. Entweder man verschiebt den Termin der Produktivnahme oder man behebt in einer Nacht-und-Nebel-Aktion schnell die schlimmsten Symptome. Beide Alternativen sind nicht empfehlenswert und führen zu weiteren ungeplanten Kosten.

Schlechte Performance kann zu einem Hamsterradeffekt im Team führen.

Die Anwendung hat eine schlechte Performance. Dadurch sind Testzyklen sehr aufwendig. In der Konsequenz werden weniger Tests durchgeführt. Es dauert lange, bis der Entwickler weiß, ob sein Code das gewünschte Ergebnis liefert. Dadurch steigt die Fehlerrate. Durch weniger fachliche Testzyklen werden mehr Fehler übersehen und erst im Produktivbetrieb fallen diese auf. Dann müssen Hotfixes erstellt und im schlimmsten Fall zusätzliche Bereinigungsprogramme für fehlerhaft verarbeitete Daten geschrieben werden. Der Aufwand für den Tagesbetrieb steigt. In der Konsequenz hat das Team noch weniger Zeit, sich um Performancefragen zu kümmern, und es entsteht ein sich selbst verstärkender Hamsterradeffekt *(siehe Abbildung 2)*. Im Extremfall kann dies zur völligen Handlungsunfähigkeit führen.

Performance als Projektaufgabe

Den „Hamsterradeffekt" muss man auf jeden Fall vermeiden. Deshalb darf man

Performance nicht als Projektrisiko, um das man sich bei Bedarf kümmert, betrachten. Stattdessen muss Performance als Projektaufgabe von Anfang an geplant werden. Dies umfasst:

- Know-how
- Zeit
- Budget
- Mitarbeiter

Nach meiner Erfahrung ist die Suche nach der genauen Fehlerursache der schwierigste Schritt im Rahmen der Performanceoptimierung. Leider wird über Performance häufig in größeren Runden oder sogar speziellen Task-Force-Gruppen sehr ineffizient diskutiert *(siehe Abbildung 3)*.

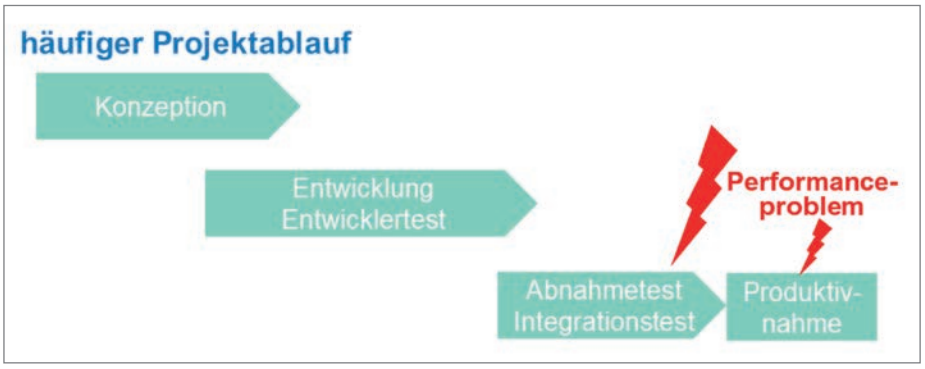


Abbildung 1: Häufiger Projektablauf (Quelle: Jörg Stahnke)

Eine auf Vermutungen basierende Diskussion ist ineffizient und kostet viel Zeit. Verantwortlichkeiten werden zwischen Entwicklern, Datenbankadministratoren, Betriebssystembetreuern, dem Netzwerk, der Hardware und anderen hin- und hergeschoben. Es gilt das Motto „Schuld sind immer die anderen". Meine Erfahrung ist außerdem, dass das Problem fast immer an einer völlig unerwarteten Stelle liegt. Dieser Effekt ist dadurch zu erklären, dass die Teammitglieder über erwartete performancekritische Stellen des Programms intensiv nachgedacht haben und deshalb hier eine sehr gute Lösung umgesetzt wurde. An Stellen, für die man keine Performanceprobleme erwartet, erfolgt dies nicht.

Deshalb hat sich für mich folgendes Vorgehen bewährt:

1. detaillierte Messungen durchführen
2. Fakten ermitteln
3. danach eine Diskussion führen

Diskussionen, die auf Messwerten und Fakten basieren, sind wesentlich zielgerichteter. Man kann über konkrete Tatsachen sprechen und findet schneller Lösungen *(siehe Abbildung 3)*.

Messung mithilfe eines Black-Box-Verfahrens

Die Oracle-Datenbank bietet die Möglichkeit, sich mithilfe der SQL-Tuning-Pa-

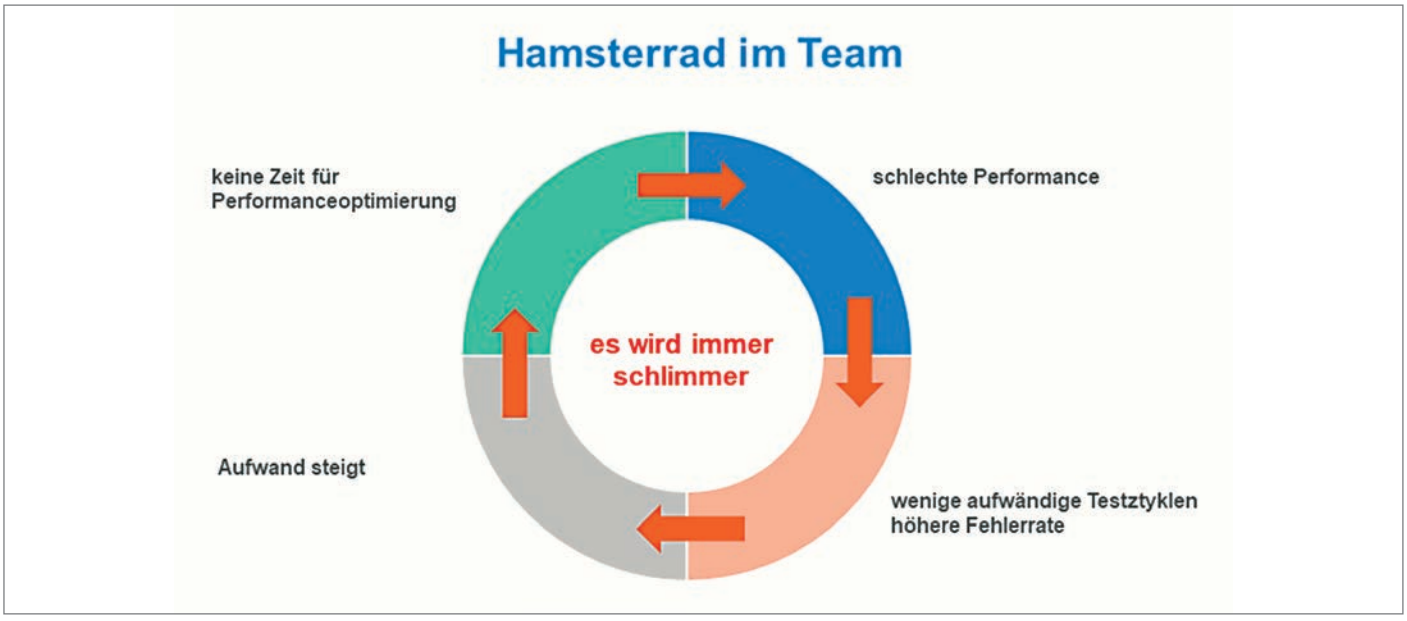


Abbildung 2: Hamsterrad im Team (Quelle: Jörg Stahnke)

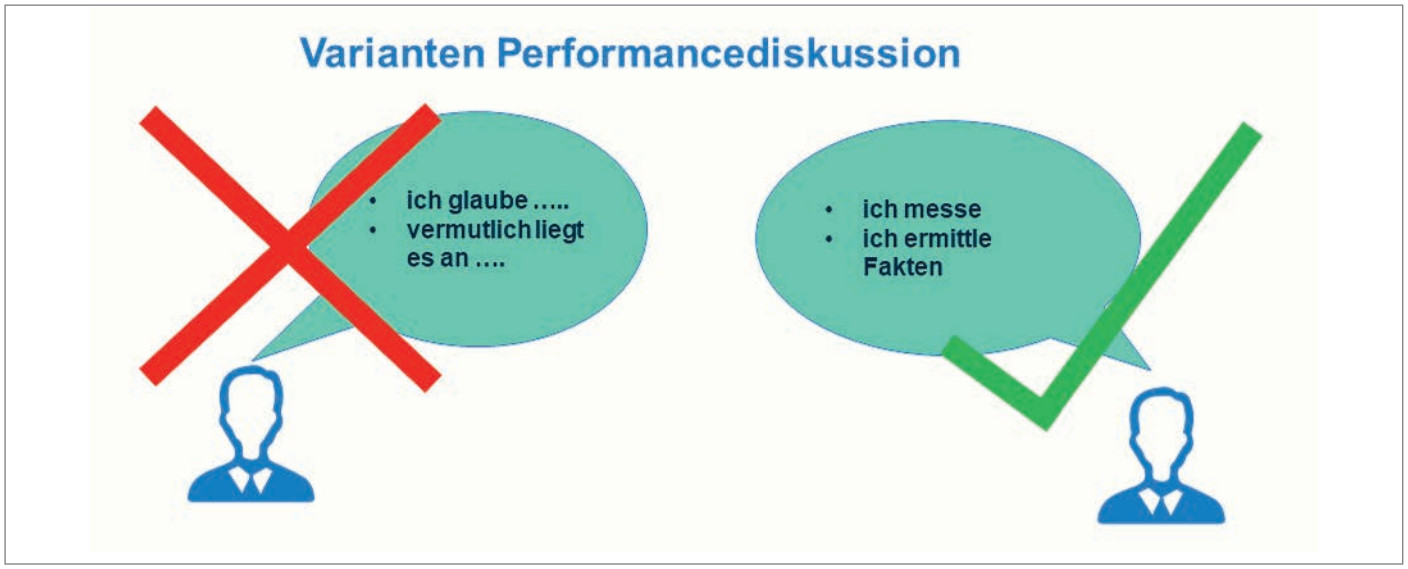


Abbildung 3: Varianten der Performancediskussion (Quelle: Jörg Stahnke)

ckages zwischen Applikation und Datenbank „einzuklinken" und den Workload dort detailliert zu erfassen. Dies ist in *Abbildung 4* dargestellt. Dazu muss man nur wissen, mit welchem Datenbankbenutzer sich die Anwendung an der Datenbank anmeldet. Der Ablauf ist wie folgt:

- Start der SQL-Tuning-Funktionalitäten
- Start der zu testenden Anwendung

Der Start des SQL-Tuning *[1]* erfolgt automatisiert per Skript.

Häufig sind für diese Messungen keine gesonderten Tests notwendig. Man misst einfach Tests, die gemäß Projektplanung durchgeführt werden. Geplante fachliche Tests oder regelmäßige Entwicklertests werden durch Performancemessungen begleitet. So entsteht kein Mehraufwand.

Die Performancemesswerte werden de facto als „Abfallprodukt" gewonnen. Die Messwerte stehen in Form von Datenbanktabellen zur Verfügung und können leicht ausgewertet werden.

Gewonnene Messwerte

Das erste wichtige Ergebnis liefert den Anteil der Datenbank an der Gesamtlaufzeit. Wenn von einer Stunde Laufzeit nur 5 Sekunden auf die Datenbank entfallen, kann man die Datenbankanalyse sofort beenden und sich auf andere Komponenten konzentrieren. Gerade in Data-Warehouse-Anwendungen hat die Datenbank häufig jedoch einen entscheidenden Anteil an der Gesamtlaufzeit. In den meisten Fällen sind es überraschenderweise nur 3 verschiedene SQL-Statements, die bis zu 90% der Gesamtlaufzeit ausmachen. Daher hat man sofort eine klare Aussage darüber, auf welche Bereiche sich die Optimierung konzentrieren muss.

Für jedes einzelne SQL-Statement liegen unter anderem folgende Messwerte vor:

- Wie oft wurde es ausgeführt?
- Wie viele Zeilen wurden geliefert?
- Wie lange hat es gedauert?
- Nach welchem Plan wurde es ausgeführt?
- Welche Teilschritte der Ausführung kosten die meiste Zeit?
- Gab es Wartezustände?
- Werden zu viele oder zu wenige Indizes verwendet?

Diese in *Abbildung 5* dargestellten Messwerte liefern ein klares Bild, wo das Problem ist und welche Ursache dahinter steckt.

Beispiele wären:

- ein SQL dauert nur 0,1 Millisekunden, wird aber 5 Millionen mal pro Stunde ausgeführt

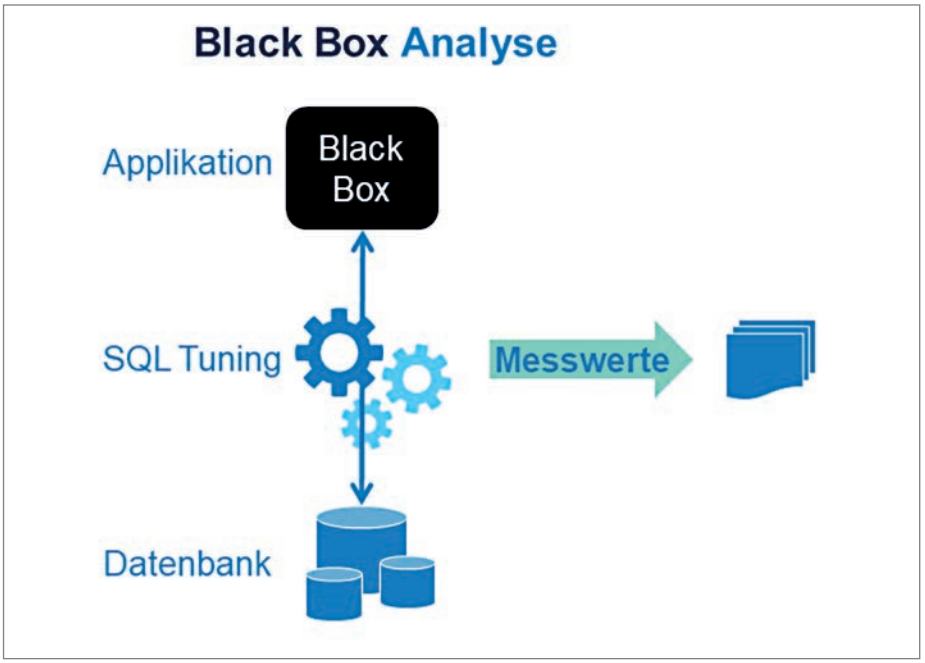


Abbildung 4: Ablauf der Blackbox-Messung (Quelle: Jörg Stahnke)

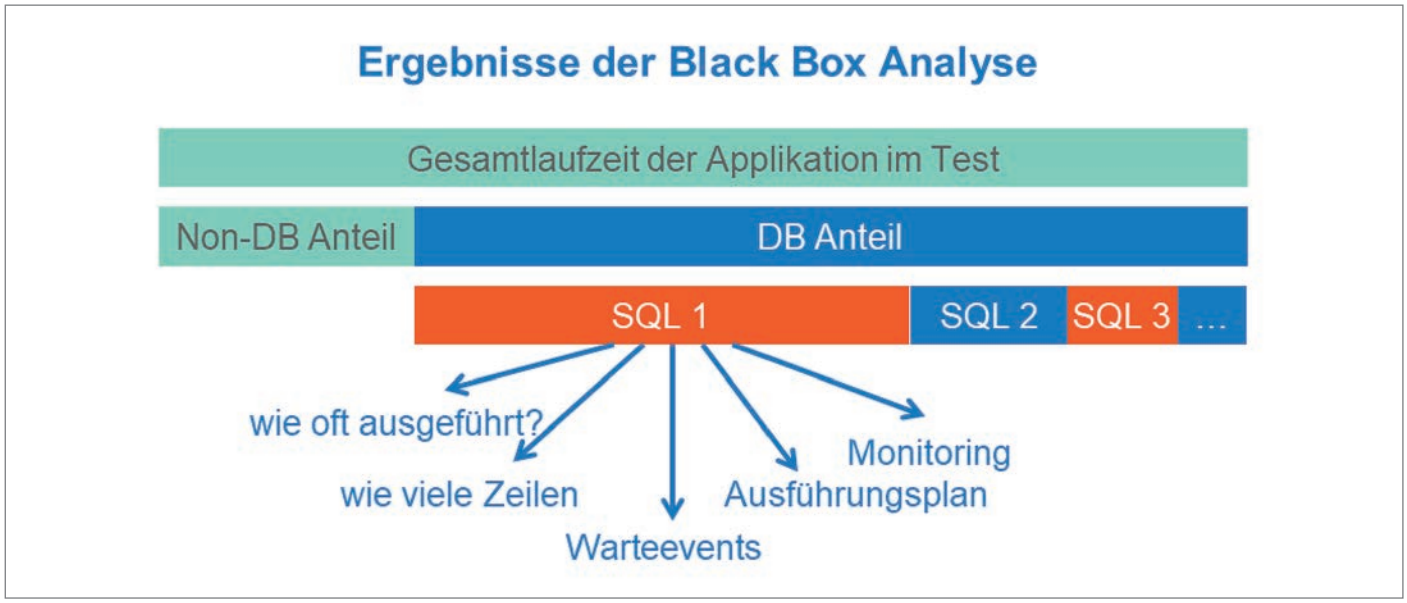


Abbildung 5: Übersicht über die Messwerte (Quelle: Jörg Stahnke)

- ein Index wird nicht verwendet, weil bei der Abfrage durch einen Tippfehler die Datentypen falsch übergeben wurden
- das Partition Pruning funktioniert nicht wie erwartet
- der Oracle Optimizer schätzt die Anzahl der Ergebniszeilen falsch ein und verwendet dadurch einen falschen Ausführungsplan
- wider Erwarten wird kein Parallel Query verwendet
- die Datenbank führt extrem viele I/O-Operationen auf den Plattensystemen aus
- umfangreiche (evtl. unnötige?) Sortiervorgänge
- häufige Wartezustände durch konkurrierende Datenzugriffe

Die Messwerte weisen also sehr detailliert auf die Ursachen der Performanceprobleme hin. Dadurch kann sofort eine Lösungssuche auf Detailebene erfolgen. Diese Arbeitsweise ist extrem zielgerichtet und effizient.

Der Aufwand für die Umsetzung der Lösungen kann dabei stark schwanken. Im einfachsten Fall wird das Problem durch eine Typ-inkonsistente Parameterübergabe aufgrund eines Tippfehlers verursacht. Dies kann innerhalb einer Minute korrigiert werden. Im schlimmsten Fall muss die Anwendungsarchitektur geändert werden, da es sonst zu massiven Wartezuständen auf der Datenbank kommt.

Kosten- und Nutzen-Betrachtung

Wenn Sie im Projekt systematisch Performance planen, müssen Sie dafür Zeit, Mitarbeiter und Budget vorsehen. Die Erarbeitung und Umsetzung von Lösungen erfordern Aufwand. Diesen Ausgaben stehen allerdings Einnahmen gegenüber. Ungeplante „Ad-hoc"-Optimierungen zum Projektende entfallen. Sie können effizienter entwickeln und testen, da alle Tests schneller sind. Entwickler arbeiten schneller und fehlerfreier. Die Frage „Hat mein Code das gewünschte Ergebnis?" wird in kürzerer Zeit beantwortet.

Netto entstehen daher keine Kosten. Dafür gibt es folgende Vorteile:

- Sie agieren, statt auf Probleme zu reagieren
- hohe Effizienz der Entwicklung
- zufriedenere Anwender
- entspannteres Arbeiten

In der Praxis fallen im Projekt die Kosten für die Performanceoptimierung immer an. Aus meiner Erfahrung hat man als Projektleiter im Prinzip nur zwei Wahlmöglichkeiten:

- Das Budget wird in einem geordneten Prozess geplant ausgegeben und die Applikation hat eine sehr gute Performance.
- Das Budget muss man zum Schluss in hektische Ad-hoc-Maßnahmen investieren und hat trotzdem nur eine teilweise optimierte Anwendung.

Empfehlung

Implementieren Sie für jedes Release den in *Abbildung 6* dargestellten Projektablauf für die Performance-Optimierung.

Mit Start des Release wird die beschriebene Black-Box-Analyse durchgeführt. Für die drei SQL-Statements mit der höchsten Gesamtlaufzeit wird eine detaillierte Analyse durchgeführt. Mithilfe von Modifizierungen des SQL-Statements wird losgelöst von der eigentlichen Applikation durch Datenbankspezialisten ein Lösungsvorschlag erarbeitet. Dabei wird zielgerichtet je nach Messwert/Ursache des Problems gearbeitet.

Sowohl die Messwerte als auch die Ursachen und damit die Lösungsansätze können sehr vielfältig sein. Immer wieder gibt es Überraschungen.

Mögliche Lösungsansätze sind zum Beispiel:

- geänderte DB-Parameter
- mehr/weniger Indizes
- Partitionierung
- Komprimierung
- Parallel Query
- technische Änderung des SQL-Statements
- minimale fachliche Änderung des SQL-Statements

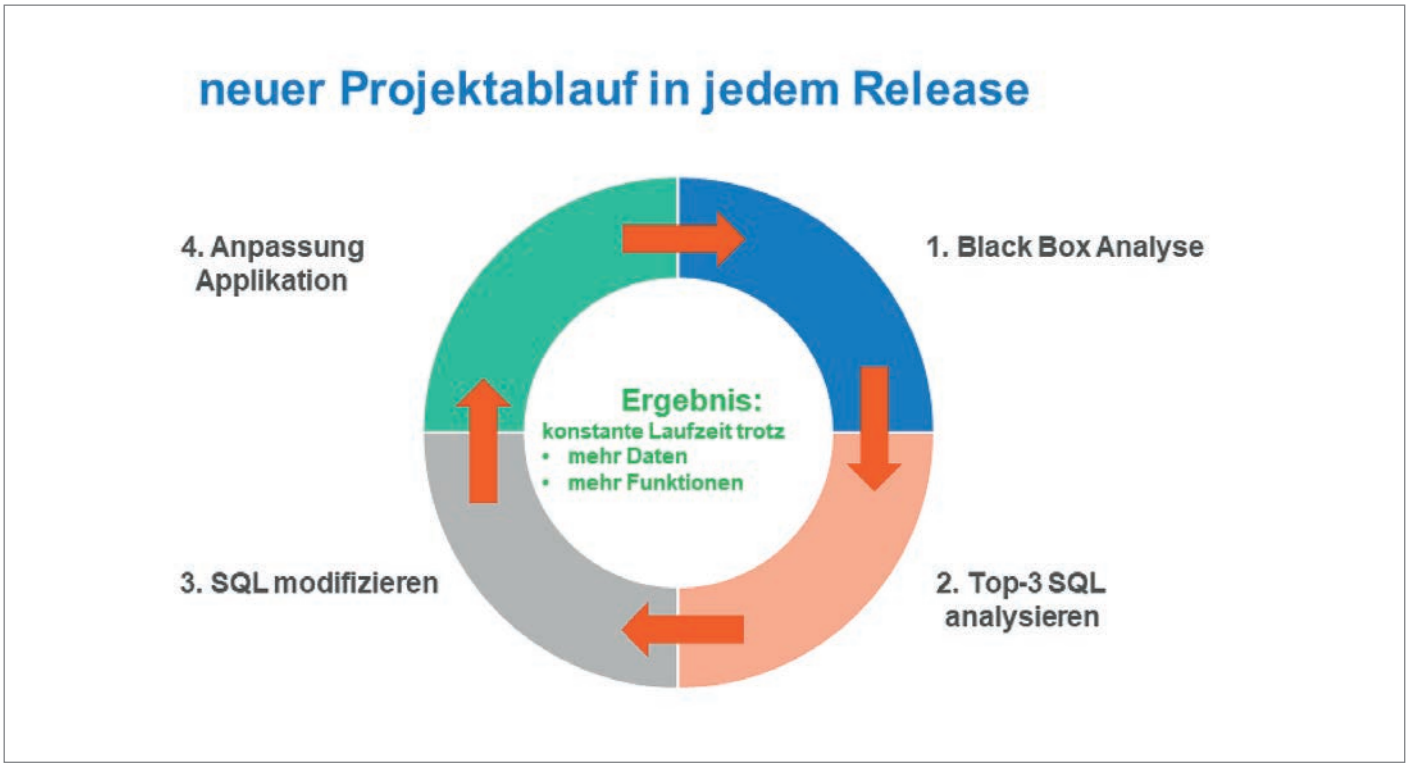


Abbildung 6: Neuer Projektablauf (Quelle: Jörg Stahnke)

Ziel ist es, eine Variante des SQL-Statements zu finden, die eine sehr gute Performance hat.

Danach wird mit den Anwendungsentwicklern diskutiert, wie dieses geänderte SQL-Statement in die Anwendung integriert werden kann. Von Antworten wie „Das geht nicht mit unserem Framework" darf man sich nicht entmutigen lassen. Häufig wäre die korrekte Antwort „Ich weiß noch nicht, wie dies mit unserem Framework geht" gewesen.

In Einzelfällen kann über Performance-kritische fachliche Anforderungen ohne großen Nutzen mit den Anwendern gesprochen werden. Beispielweise kann eine vollständige Protokollanzeige auf die Anzeige der letzten 2 Wochen reduziert werden, wenn nur die aktuellen Einträge fachlich interessant sind.

Im Ergebnis bleiben die Laufzeiten konstant, obwohl sich die Datenmenge im Data Warehouse durch Anbindung neuer Systeme erhöht hat und weitere fachliche Funktionalitäten ergänzt wurden.

Fazit

Unerwartete Performanceprobleme in Oracle-Datenbanken kann man durch folgendes Vorgehen ausschließen:

1. Performance muss geplant werden
2. systematisch mit Messwerten/Fakten arbeiten
3. keine Vermutungen aufstellen
4. gezielt Know-how im Team aufbauen
5. SQL-Tuning aktiv nutzen
6. regelmäßige Optimierungen in jedem Release

Mit diesen Empfehlungen werden Performanceprobleme von Anfang an vermieden und müssen nicht gelöst werden.

Quellen

[1] Der vollständige SQL Tuning Advisor erfordert eine zusätzliche Lizenzierung. Falls diese Lizenz nicht gekauft wurde, kann jedoch trotzdem eine Vielzahl von Messwerten lizenzfrei ermittelt werden.

Über den Autor

Jörg Stahnke verfügt über eine mehr als 20-jährige Erfahrung im Bereich Data Warehouse. Seine Spezialgebiete sind dabei die Erstellung von Data-Warehouse-Architekturen sowie die Performanceoptimierung. Dabei legt er sehr viel Wert auf Architekturen, die eine effiziente Entwicklung und schnelle Änderungszyklen ermöglichen sowie potenzielle Fehlerquellen vermeiden. Bereits seit Beginn seiner Tätigkeit setzt er Generatoren ein, um Entwicklungsprozesse zu automatisieren. Seit 7 Jahren arbeitet er daher mit Data Vault.

Jörg Stahnke
joerg.stahnke@ppi.de

Wie kann Ansible-Automatisierung das Leben von DBAs verbessern?

Nicolas Penot, dbi services

Ich liebe IT-Automatisierung. Meiner Ansicht nach ermöglicht die Automatisierung die Entwicklung von IT-Umgebungen. Automatisieren Sie Ihre sich wiederholenden Aufgaben, und Sie werden Zeit sparen, um an wichtigen Aufgaben arbeiten zu können: an Aufgaben, die den Wert Ihres IT-Ökosystems exponentiell steigern werden. Es macht außerdem Ihren Arbeitsalltag interessanter: Lernen, Programmieren, Testen, Automatisieren und etwas Neues lernen.

Welches Automatisierungswerkzeug passt also zu Ihrem Bedarf? Es gab eine Zeit, in der Bash-Skripte mein Lieblingswerkzeug für diese Aufgabe waren. Einfach zu programmieren und einfach zu implementieren. Bash ist auch in fast alle Umgebungen integriert. Damit ist es breit verfügbar und portabel. Für mich funktioniert es immer noch gut. Wenn jedoch verschiedene Leute an einer gemeinsamen Infrastruktur arbeiten, wird es nützlich, einige Standards zu haben. Eine vorhersagbare Codierungsstruktur, die leicht zu lesen und zu pflegen ist. Nachdem ich Ansible entdeckt habe, muss ich zugeben, dass es ein ernstzunehmendes Tool für die IT-Automatisierung ist.

Dieser Artikel stellt Ansible mit einigen Beispielen rund um GoldenGate vor. Wie können wir eine Automatisierung für ein Oracle-Produkt wie beispielsweise GoldenGate programmieren?

Anwendungsfall mit GoldenGate

In diesem Beispiel verfüge ich über drei Server. Zwei Server hosten eine Oracle-Datenbank und die GoldenGate-Binärdateien. Der dritte Server dient als Ansible-Host, von dem aus ich meine Ansible-Aufgaben ausführen werde *(siehe Abbildung 1)*:

Ansible

Aus meiner Sicht sind dies die drei Punkte, um Ansible und seinen Nutzen schnell zu verstehen:

- **Kein Agent:** Ansible erfordert keine Installation von Agenten auf Ihren Zielservern. Stattdessen verbindet es sich

über das gesicherte SSH-Protokoll, um seine Aufgaben auszuführen.

- **Fakten:** Fakten sind ein Satz von Variablen, den Ansible zur Laufzeit auf den Zielrechnern sammelt. Diese Variablen können dann in Ihren Skripten verwendet werden. Fakten sind fast alle Informationen, die Sie von einem Zielhost benötigen, wie beispielsweise IPs, NICs, Geräte usw. Sie können sogar Ihre Fakten hinzufügen, wie etwa die Liste der Oracle-Instanzen, die mit Ihrem Oracle Home laufen *(siehe Abbildung 2)*.

Zwei Stufen:

- Sie können das sogenannte Ad-hoc-Befehlszeilenwerkzeug verwenden. Mit diesem Tool können Sie mit einer Befehlszeile Aktionen wie zum Beispiel das Erstellen von Betriebssystembenutzern auf mehreren Servern ausführen.
- Dann können Sie eine Reihe von Operationen skripten. Zu diesem Zweck werden Sie ein Playbook verwenden. Ein Playbook ist eine Datei, die Ihre Abfolge von Operationen im YAML-Format enthält.

Installation

Sie kennen jetzt die Grundlagen und sind bereit, sie selbst zu testen. Zuerst müssen Sie vermutlich Ansible installieren. Zu dem Zeitpunkt, an dem ich diesen Artikel schreibe, enthält die Ansible-Dokumentation die folgenden Anweisungen *(siehe Listing 1)*:

Nun müssen Sie über eine passwortlose SSH-Konnektivität vom Ansible-Host zu den Oracle-Servern verfügen. Das nachstehende Snippet wird auf dem Ansible-Host ausgeführt *(siehe Listing 2)*. Der erste Befehl erstellt einen RSA-Authentifizierungsschlüssel. Die nächsten drei Befehle installieren den öffentlichen Schlüssel in der Datei „remote authorized_keys".

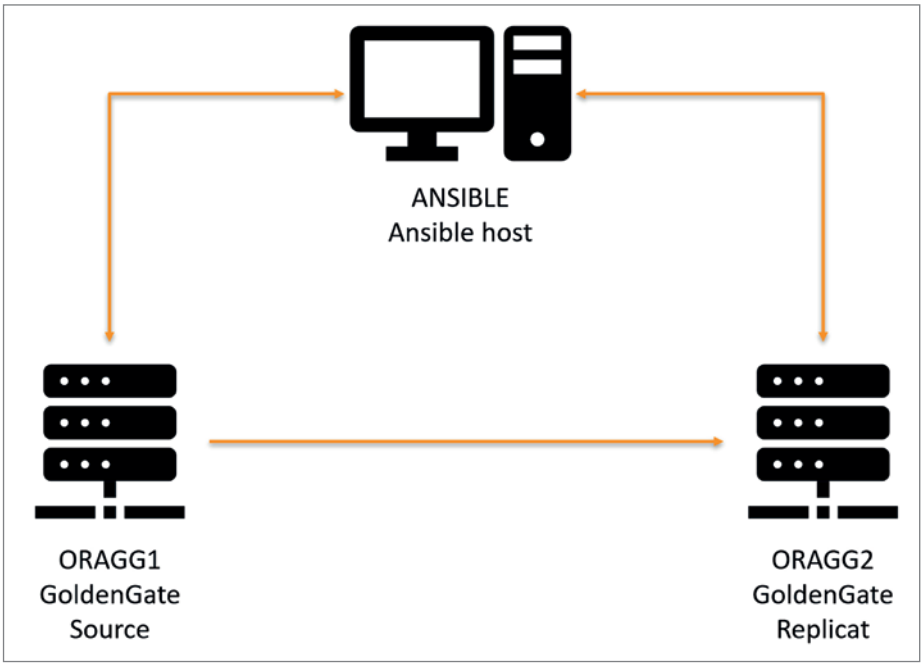


Abbildung 1 (Quelle: Nicolas Penot)

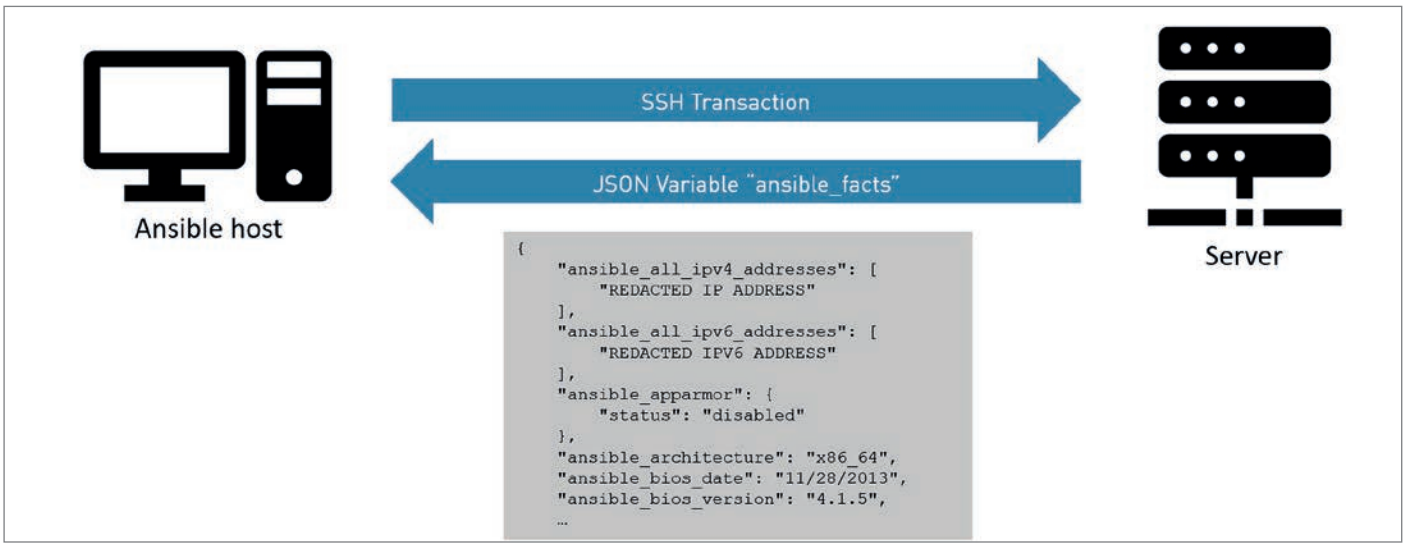


Abbildung 2 (Quelle: Nicolas Penot)

```
# CentOS / RHEL
$ yum install ansible
      #OR
# Ubuntu
$ sudo apt-add-repository --yes --update ppa:ansible/ansible
$ sudo apt-get install ansible
```

Listing 1

```
$ ssh-keygen -t rsa -C nicolas.penot@dbi-services.com
$ ssh-copy-id root@oragg1
$ ssh-copy-id root@oragg2
$ ssh-copy-id root@ansible
```

Listing 2

```
$ cat /etc/ansible/hosts    ## default location for the inventory file
[db_prod]
oragg[1:2]
[source]
oragg1
[replicat]
oragg2
[db_prod:children]
source
replicat
[db_prod:vars]
oracle_user=oracle
inventory_location=/u01/app/oraInventory
[jumphost]
ansible
```

Listing 3

```
$ ansible-inventory --graph
@all:
  |--@db_prod:
  |   |--@replicat:
  |   |   |--oragg2
  |   |--@source:
  |   |   |--oragg1
  |   |--oragg1
  |   |--oragg2
  |--@jumphost:
  |   |--ansible
  |--@ungrouped:
```

Listing 4

```
$ ansible db_prod --module-name command --args "hostname -f"
oragg2 | CHANGED | rc=0 >>
oragg2

oragg1 | CHANGED | rc=0 >>
oragg1
```

Listing 5

- copy - Copies files to remote locations
- cpanm - Manages Perl library dependencies.
- cpm_user - Get various status and parameters from WTI OOB and PDU devices
- cron - Manage cron.d and crontab entries

Abbildung 3 (Quelle: Nicolas Penot)

- azure - create or terminate a virtual machine in azure (D)

Abbildung 4 (Quelle: Nicolas Penot)

Ansible-Inventar

Das Inventar ist eine von Ansible verwendete Datei, um den Gruppennamen aufzulösen. Das ist ähnlich wie bei einer DNS, die aus einem Hostnamen eine IP-Adresse erhalten würde. Ansible verwendet das Inventar, um aus einer Gruppe eine Liste der Hosts zu erhalten. Dies liegt daran, dass Ansible die Hostgruppe als Parameter verwendet, um Aktionen auszuführen, statt die Hostnamen zu verwenden. Hier ist ein Beispiel *(siehe Listing 3)*.

Sie können den Befehl „ansible-inventory" verwenden, um zu sehen, was Ansible derzeit in seinem Inventar hat *(siehe Listing 4)*.

Es gibt zwei Gruppen: „db_prod" und „jumphost". Die Gruppe „db_prod" enthält zwei Untergruppen: eine mit den Servern, die als Extrahierung für GoldenGate verwendet werden, und eine mit den Hosts, die für die Replikation genutzt werden.

Das Ansible-Ad-hoc-Befehlszeilenwerkzeug

Der Ad-hoc-Befehl lautet „ansible". Im folgenden Beispiel verwende ich das Ansible-Modul „command", um auf jedem Server der Gruppe „db_prod" den Befehl „hostname -f" auszuführen *(siehe Listing 5)*.

Dann gibt Ansible einen Bericht über die Ausführung aus, einschließlich des Ergebnisses für den „hostname"-Befehl. Mit diesem einfachen Befehl können Sie sich bereits vorstellen, was Sie einfach und schnell tun können.

Jetzt lassen Sie vielleicht Ihre Fantasie spielen und Sie möchten eine komplexere Automatisierung erstellen. Gut, lassen Sie uns die Playbooks kennenlernen.

Ansible-Modul

Wie oben gesehen, habe ich die Option „--module-name" verwendet. Ansible funktioniert mit Modulen. Wir können Module mit Plug-ins vergleichen. Mit einem Modul kann Ansible Aktionen wie das Kopieren einer Datei, das Erstellen eines OS-Benutzers oder sogar das Erstellen einer Amazon-EC2-Instanz durchführen. Dies sind Beispiele für die mehr als 2000 Module, die es derzeit gibt *(siehe Abbildungen 3,4 und 5)*.

Hier ist zum Beispiel der Ansible-Befehl, mit dem ich auf allen Servern der Gruppe „db_prod“ einen neuen Benutzer namens „nicolas“ anlegen kann *(siehe Listing 6)*.

Module kombinieren, um komplexere Aktionen auszuführen: Anstatt alle Befehle nacheinander in ein Bash-Skript einfügen zu müssen, bietet Ansible das Konzept des Playbooks. Ein Playbook ist eine Datei im YAML-Format, in der Sie die Module auflisten, die auf einer Gruppe von Hosts nacheinander ausgeführt werden sollen. Dies ist ein Beispiel *(siehe Listing 7)*.

Dieses Playbook enthält vier Module. Das erste Modul „file“ wird verwendet, um ein Zuhause für unsere GoldenGate-Installation zu schaffen. Mit dem zweiten Modul „unarchive“ entpackt GoldenGate die Installationsdateien in das Verzeichnis „/u01/app/software/goldengate“. Das dritte Modul „fact“ erstellt eine Variable, die ich im nächsten Modul des aktuellen Playbooks verwenden kann. Und das letzte Modul – „command“ –, das wir bereits kennen, führt den sogenannten „Oracle runInstaller“ in der Kommandozeile aus.

Alle diese Module aus diesem Playbook werden in der Gruppe „db_prod“ ausgeführt. Dies ist in der dritten Zeile der YAML-Datei im Attribut „hosts“ zu sehen. Die „hosts“-Attribute beschreiben eine Gruppe von Hosts des Inventars und keine Hostnamenliste Ihrer Server.

Sie können das Playbook dann mit dem Befehl „ansible-playbook“ wie folgt ausführen *(siehe Listing 8)*.

Variablen

Im vorigen Snippet können wir auch Variablen in der YAML-Datei sehen. Die Variablen werden in doppelte geschweifte Klammern wie beispielsweise „{{ My_variable }}“ gesetzt. Diese Variablen werden während des Ausführens entsprechend der aktuellen Umgebung ersetzt. Daher kann das gleiche Playbook für jeden Server wiederverwendet werden. Die Werte, die sich zwischen den Servern unterscheiden – wie etwa die IP-Adressen –, können in Variablen umgewandelt werden, die sich entsprechend der Laufzeitumgebung ändern. Das erlaubt auch,

Abbildung 5 (Quelle: Nicolas Penot)

```
$ ansible db_prod --module-name user --args "name=nicolas group=dba"
oragg2 | CHANGED => {
   "changed": true,
   "comment": "",
   "create_home": true,
   "group": 54322,
   "home": "/home/nicolas",
   "name": "nicolas",
   "shell": "/bin/bash",
   "state": "present",
   "system": false,
   "uid": 54322
}
oragg1 | CHANGED => {
   "changed": true,
   "comment": "",
   "create_home": true,
   "group": 54322,
   "home": "/home/nicolas",
   "name": "nicolas",
   "shell": "/bin/bash",
   "state": "present",
   "system": false,
   "uid": 54322
}
```

Listing 6

```
$ cat p_goldengate_installation.yml
- name: "Play 1: GoldenGate installation"
   gather_facts: yes
   hosts: db_prod
   tasks:

   - name: "Creates GoldenGate home"
       file:
           path: "{{ goldengate_home }}"
           state: directory
           owner: oracle
           group: oinstall

   - name: "Unzip GoldenGate binaries"
       unarchive:
           src: "/u01/app/software/goldengate/fbo_ggs_Linux_x64_shipho-
me.zip"
           dest: "/u01/app/software/goldengate"
           owner: oracle
           group: oinstall

   - set_fact:
           installer_dir: "/u01/app/software/goldengate/fbo_ggs_Linux_
x64_shiphome/Disk1"

   - name: "Install GoldenGate"
       command: "{{ installer_dir }}/runInstaller -silent -waitforcom-
pletion INSTALL_OPTION=ORA12c SOFTWARE_LOCATION={{ goldengate_home }}
START_MANAGER=false INVENTORY_LOCATION={{ inventory_location }} UNIX_
GROUP_NAME=oinstall"
       become: yes
       become_user: oracle
```

Listing 7

```
$ ansible-playbook p_goldengate_installation.yml
```

Listing 8

```
$ cat p_goldengate_initial_load.yml
- name: "Play 1: Prepare Initial load on source"
  gather_facts: yes
  hosts: source
  tasks:

  - name: "Add extraction"
      include_role:
          name: goldengate
      vars:
          option: add_extraction

  - name: "Export schema {{ e_schema }} to {{ target_oracle_sid }}"
      include_role:
          name: goldengate
      vars:
          option: export_import_schema

- name: "Play 2: Configure replication on target"
  gather_facts: yes
  hosts: replicat
  tasks:

  - name: "Add replication to {{ target_oracle_sid }}"
      include_role:
          name: goldengate
      vars:
          option: add_replication
```

Listing 9

```
$ cat roles/goldengate/tasks/t_add_replication.yml
  - name: "Creates GG data directory"
      file:
          path: /u11/app/goldengate/data/{{ oracle_sid }}
          state: directory
          owner: oracle
          group: oinstall

  - name: "Copy repl{{ schema }} parameter file"
      template:
          src: "replicat.prm.j2"
          dest: "{{ goldengate_home }}/dirprm/repl{{ schema }}.prm"
          owner: oracle
          group: oinstall

  - name: "Copy GoldenGate script file"
      template:
          src: "add_replicat.j2"
          dest: "/tmp/add_replicat"
          owner: oracle
          group: oinstall

  - name: "Execute ggsci command"
      shell: "{{ goldengate_home }}/ggsci paramfile /tmp/add_replicat"
          become: true
          become_user: oracle
          environment:
              ORACLE_HOME: "{{ oracle_home }}"
              ORACLE_SID: "{{ oracle_sid }}"
              LD_LIBRARY_PATH: "{{ oracle_home }}/lib"
```

Listing 10

eine Variable anstelle aller Playbooks zu ändern, wenn Sie eine globale Variable ändern, wie beispielsweise das standardmäßige Oracle-Home.

Das für die Variablen verwendete Modell ist Jinja2. Jinja2 ist eine Bibliothek für Python, die von Ansible als Templating-Sprache genutzt wird. Das ermöglicht auch komplexere Strukturen wie bedingte Ersetzungen oder Schleifenersetzung *(mehr zu diesem Thema auf der Website http://jinja.pocoo.org/docs/2.10/)*.

Rollen

Stellen Sie sich vor, dass Sie ein Playbook erstellt hätten, das ein Home-Verzeichnis für Oracle sowie die OS-Benutzer und -Gruppen gemäß Ihren Standards erstellt, die OS-Kernelparameter konfiguriert und alle für Oracle-Software erforderlichen Oracle-Pakete installiert. Möchten Sie diesen Code vielleicht wiederverwenden oder wiederverwendbar machen?

Hierzu benutzt Ansible das Konzept der Rolle. Eine Rolle ist eine Reihe von Modulen, die Sie entwickelt haben und die in mehreren Playbooks wiederverwendet werden können. Wir können sie mit einer Funktion in einer Programmiersprache vergleichen.

Als Beispiel habe ich nachstehend ein Playbook, in dem nur Rollen aufgerufen werden, um GoldenGate erstmalig zu laden *(siehe Listing 9)*.

Dieses Playbook enthält drei Rollen. Zwei dieser Rollen werden in der Hostgruppe namens „source“ und die letzte Rolle in der Hostgruppe namens „replicat“ ausgeführt. Wie bereits gesagt: Rollen sind eine Reihe von Modulen, die im YAML-Format geschrieben sind. Dies ist beispielsweise die YAML-Datei der Rolle „GoldenGate“ mit der Option „add_replication“ *(siehe Listing 10)*.

Diese Rolle besteht aus 4 Ansible-Modulen: „file“, „template“, „template“ und „shell“.

Um dieses Playbook zu verwenden, muss ich zusätzliche Parameter hinzufügen, um die Datenbank anzugeben, auf der ich die Replikation installieren werde *(siehe Listing 11)*.

Dies ist das vollständige Ergebnis für unseren Anwendungsfall *(siehe Abbildung 6 und Abbildung 7)*.

```
$ ansible-playbook playbooks/p_goldengate_initial_load.yml \
-e "e_target_hostname=oragg2 e_target_oracle_sid=DB2 e_schema=soe"
```

Listing 11

```
PLAY [Play 1: Prepare Initial load on source] ************************************

TASK [Gathering Facts] ***********************************************************
ok: [oragg1]

TASK [Add extraction] ************************************************************

TASK [goldengate : include_tasks] ************************************************
included: /home/nico/ansible/roles/goldengate/tasks/./t_add_extraction.yml for oragg1

TASK [goldengate : Set facts] ****************************************************
ok: [oragg1]

TASK [goldengate : Creates GG data directory] ************************************
ok: [oragg1]

TASK [goldengate : Add supplemental log data] ************************************
changed: [oragg1]

TASK [goldengate : Copy extrsoe parameter file] **********************************
ok: [oragg1]

TASK [goldengate : Copy pumpsoe parameter file] **********************************
ok: [oragg1]

TASK [goldengate : Copy GoldenGate script file] **********************************
ok: [oragg1]

TASK [goldengate : Execute ggsci command] ****************************************
changed: [oragg1]

TASK [Export schema soe to DB2] **************************************************

TASK [goldengate : include_tasks] ************************************************
included: /home/nico/ansible/roles/goldengate/tasks/./t_export_import_schema.yml for orag

TASK [goldengate : Set facts] ****************************************************
ok: [oragg1]

TASK [goldengate : Get current SCN on source] ************************************
changed: [oragg1]
```

Abbildung 6 (Quelle: Nicolas Penot)

```
TASK [goldengate : Export schema soe] ********************************************
changed: [oragg1]

TASK [goldengate : Copy dumpfile soe.dmp to oragg2] ******************************
changed: [oragg1]

TASK [goldengate : Purge dumpfile /u01/app/oracle/admin/DB1/dpdump/soe.dmp] ******
changed: [oragg1]

TASK [goldengate : Check if schema soe exists] ***********************************
changed: [oragg1 -> oragg2]

TASK [goldengate : Create schema soe if not exists] ******************************
changed: [oragg1 -> oragg2]

TASK [goldengate : Check if tablespace for soe exists] ***************************
changed: [oragg1 -> oragg2]

TASK [goldengate : Create tablespace soe if not exists] **************************
changed: [oragg1 -> oragg2]

TASK [goldengate : Import schema soe] ********************************************
changed: [oragg1 -> oragg2]

TASK [goldengate : Purge dumpfile /u01/app/oracle/admin/DB2/dpdump/soe.dmp] ******
changed: [oragg1 -> oragg2]

PLAY [Play 2: Configure replication on target] ***********************************

TASK [Gathering Facts] ***********************************************************
ok: [oragg2]

TASK [Add replication to {{ target_oracle_sid }}] ********************************

TASK [goldengate : include_tasks] ************************************************
included: /home/nico/ansible/roles/goldengate/tasks/./t_add_replication.yml for oragg2

TASK [goldengate : Set facts] ****************************************************
ok: [oragg2]
```

Abbildung 7 (Quelle: Nicolas Penot)

Fazit

Auf einen Blick: Ansible verwendet Module, um Aufgaben über SSH auf Gruppen von Hosts auszuführen, die in einem Inventar definiert sind.

Wie wir in diesem Artikel gesehen haben, wird Ansible in wenigen Schritten Ihre Produktivität verbessern und Ihre Infrastruktur anpassen:

1. Ansible auf einem zentralen Host installieren
2. Erstellen eines Inventars Ihrer Server
3. Konvertieren sich wiederholender Aufgaben in Playbooks

Ich hoffe, dass Sie diesen Artikel nützlich fanden, um Ansible gut nachvollziehbar zu machen. Zögern Sie bitte nicht, uns zu kontaktieren, falls Sie weitere Informationen wünschen.

Nicolas Penot
penot@dbi-services.com

Wir begrüßen unsere neuen Mitglieder

Persönliche Mitglieder

- Rene Renk
- Zacharias Thiedtke
- Almir Avdic
- Bennet Schulz
- Christoph Pfyffer
- Waldemar Schlonsok
- Lars Schneider
- Richard Rieb
- Benedikt Kasparek
- Marc Neitzner
- Andrea Dengel
- Benjamin Nothdurft
- Andreas Wildmann

Firmenmitglieder DOAG

- EXXETA AG, Volkmar Kuhnle
- acarda GmbH, Johannes Waldheim

Termine

Juni

26.06.2019
DOAG Berliner Expertenseminar zum Thema Forms Report
Jan-Peter Timmermann
Berlin

27.06.2019
Regionaltreffen Hannover
Andreas Ellerhoff
Hannover

12.07.2019
DOAG Datenbank Webinar: Dank Swingbench die Oracle Datenbank besser verstehen
Martin Bach
online

16.07.2019
Regionaltreffen München/Südbayern
Andreas Ströbel

17.07.2019
DOAG User Day Hochverfügbarkeit
Martin Klier & Jürgen Vitek
Stuttgart

18.07.2019
Regionaltreffen Nürnberg
Martin Klier & Thomas Köppel

25.07.2019
Regionaltreffen Stuttgart
Jens-Uwe Petersen & Anja Stollberg

Juli

11.07.2019
Regionaltreffen Karlsruhe
Reiner Bünger & Jochen Kutscheruk

August

09.08.2019
DOAG Datenbank Webinar: Monitoring mit Check MK
Ernst Leber
online

September

02.09.2019
Go DevOps 2019
Christian Schwitalla
Berlin

05.09.2019
Regionaltreffen Karlsruhe
Reiner Bünger & Jochen Kutscheruk

Impressum

Red Stack Magazin wird gemeinsam herausgegeben von den Oracle-Anwendergruppen DOAG Deutsche ORACLE-Anwendergruppe 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 das User-Magazin rund um die Produkte der Oracle Corp., USA, im Raum Deutschland, Österreich und Schweiz. Es ist unabhängig von Oracle und vertritt weder direkt noch indirekt deren wirtschaftliche Interessen. Vielmehr vertritt es die Interessen der Anwender an den Themen rund um die Oracle-Produkte, fördert den Wissensaustausch zwischen den Lesern und informiert über neue Produkte und Technologien.

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.

Die DOAG Deutsche ORACLE-Anwendergruppe e.V. hält 100 Prozent der Stammeinlage der DOAG Dienstleistungen GmbH. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. wird gesetzlich durch den Vorstand vertreten; Vorsitzender: Stefan Kinnen. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. informiert kompetent über alle Oracle-Themen, setzt sich für die Interessen der Mitglieder ein und führen einen konstruktiv-kritischen Dialog mit Oracle.

Redaktion:
Sitz: DOAG Dienstleistungen GmbH
(Anschrift s.o.)
ViSdP: Mylène Diacquenod
Redaktionsleitung: Martin Meyer
Kontakt: redaktion@doag.org
Weitere Redakteure (in alphabetischer Reihenfolge): Lisa Damerow, Mylène Diacquenod, Marina Fischer, Sanela Lukavica, Robert Marz, Yann Neuhaus, Fried Saacke

Titel, Gestaltung und Satz:
Alexander Kermas
DOAG Dienstleistungen GmbH
(Anschrift s.o.)

Fotonachweis:
Titel: © vectorpouch | *www.freepik.com*
S. 5 links: © Fried Saacke | *www.doag.org*
S. 5 rechts: © Ronny Weiß | *www.doag.org*
S. 6: © Andrejs Vorobjovs/ Lettische ORACLE User Group
S. 7: © Martin Meyer | *www.doag.org*
S. 12: © jemastock | *https://de.123rf.com*
S. 19: © THEERAVAT BOONNUANG | *https://de.123rf.com*
S. 25 Logo: © Oracle Apex | *https://apex.oracle.com*
S. 41: © ontsunan | *https://de.fotolia.com*
S. 46: © scusi | *https://de.fotolia.com*
S. 52: © Artem Egorov | *https://de.123rf.com*
S. 57: © conceptw | *https://de.123rf.com*
S. 62: © everythingpossible | *https://de.123rf.com*
S. 67: © Poobest | *https://de.fotolia.com*
S. 73 Figuren ©: apinan | *https://de.123rf.com*
S. 73: Hintergrund: © Dzianis Kuryanovich | *https://de.123rf.com*

Anzeigen:
Simone Fischer,
DOAG Dienstleistungen GmbH
(verantwortlich, Anschrift s.o.)
Kontakt: anzeigen@doag.org
Mediadaten und Preise unter:
www.doag.org/go/mediadaten

Druck:
adame Advertising and Media GmbH,
www.adame.de

Inserentenverzeichnis

DOAG 2019 Logistik + IT

17. September 2019 in Frankfurt

DOAG

logistik.doag.org

WIR LEBEN DIGITALISIERUNG.

Die Digitale Transformation macht vor keiner Branche halt und verändert Wertschöpfungsketten und Strukturen auch Ihres Unternehmens. Sie müssen sich neuen Herausforderungen stellen. In andere Richtungen denken. Ihr Geschäftsmodell anpassen und weiterentwickeln, damit Sie auch in Zukunft wettbewerbsfähig bleiben. Wir sind mittendrin im Geschehen und gestalten partnerschaftlich Ihren Weg ins Zeitalter der Digitalisierung. Sprechen Sie mit uns.

m.trivadis.com/digitalisierung | info@trivadis.com

BASEL | BERN | BRUGG | BUKAREST | DÜSSELDORF | FRANKFURT A.M.
FREIBURG I. BR. | GENF | HAMBURG | KOPENHAGEN | LAUSANNE
MANNHEIM | MÜNCHEN | STUTTGART | WIEN | ZÜRICH

trivadis