WHITEPAPER

# GESTALTUNG EINER ERFOLGREICHEN MICROSERVICES-ARCHITEKTUR

## VORTEILE

- Einfachere Entwicklung und Wartung
- Schnellere, unabhängige Bereitstellung
- Unabhängige Skalierbarkeit von Anwendungskomponenten
- Keine langfristige technologische Bindung

facebook.com/redhatinc
@redhatnews
linkedin.com/company/red-hat

de.redhat.com

## ZUSAMMENFASSUNG

Bei einer Microservices-Architektur handelt es sich um einen neuen Architekturstil zur Erstellung lose gekoppelter, aber autonomer Services. Neue Technologietrends – etwa DevOps, Platform-as-a-Service (PaaS), Container sowie Continuous Integration und Continuous Delivery (CI/CD) – ermöglichen Unternehmen die Entwicklung und das Management dieser modularen Systeme in einem noch nie dagewesenen Ausmaß, das frühere Ansätze wie SOA (serviceorientierte Architektur) in den Schatten stellt. Doch der Erfolg von Unternehmen, die ein Refactoring monolithischer Anwendungen in Microservices vornehmen, variiert sehr stark. Grundvoraussetzung für eine effektive Nutzung von Microservices ist ein solides Verständnis dessen, wie und aus welchem Grund Unternehmen Microservices zur Erstellung von Anwendungen einsetzen sollten.

## VERBESSERUNG DER SERVICEORIENTIERTEN ARCHITEKTUR

Eine serviceorientierte Architektur (SOA) wird üblicherweise als eine Sammlung von Anwendungskomponenten definiert, die miteinander kommunizieren, um über ein Netzwerk Dienste für andere Komponenten bereitzustellen. Das Ziel einer SOA bestand darin, ohne komplexe zentralisierte Komponenten robuste verteilte Anwendungen zu entwickeln.

Außerdem war eine SOA eng mit den Organisationsstrukturen verknüpft und wurde zur Unterstützung neuer interner Strukturen verwendet. Aus diesem Grund hing ihr Erfolg stark von den neu strukturierten organisatorischen Kapazitäten und dem Aufbau der Teams ab, die die Architektur entwarfen. Anstatt lose gekoppelte, aber autonome Systeme zu schaffen, führte eine SOA zu hochgradig fragilen Systemen, die auf eine komplexe Infrastruktur angewiesen waren. Darüber hinaus gingen frühe SOA-Implementierungen mit einem Vendor Lock-in einher, da die proprietäre Middleware oftmals auf eine zentralisierte Logik, Persistenz, Governance und Verwaltung ausgerichtet war.

Die Versprechen einer SOA werden zunehmend von Microservices-Architekturen erfüllt – und zwar in jeder Phase der Anwendungserstellung, angefangen bei der Entwicklung über die Bereitstellung bis hin zum Betrieb. Eine Microservices-Architektur ist darauf ausgerichtet, die Technologie zu vereinfachen, um mit optimierten Komponenten komplexe Systeme zu schaffen. Eine zentralisierte Logik- und Integrationsinfrastruktur, die auf schwergewichtigen, nicht standardisierten Plattformen basiert, wird durch eine Kommunikation über einfache, standardisierte Verbindungen ersetzt, die auf asynchronen HTTP- oder Messaging-Protokollen beruhen. SOAP, XML und andere schwergewichtige Protokolle und Datenformate werden durch das schlanke JSON über eine HTTP-basierte REST-Schnittstelle ersetzt. Jeder Microservice verfügt über seinen eigenen Datenspeicher; zentralisierte Governance und Persistenz sind nicht erforderlich.

Microservices verwenden Methoden und Praktiken aus den Bereichen Continuous Integration (CI) und Continuous Delivery (CD) sowie einige wichtige Komponenten, die bei einer SOA weniger üblich waren. Hierzu zählen:

- Mehrsprachige Programmierung und Persistenz
- Container oder unveränderliche virtuelle Maschinen (VM)
- Elastische, programmierbare Infrastructure-as-a-Service (IaaS) und Platform-as-a-Service (PaaS)

### SIND SIE BEREIT FÜR MICROSERVICES?

Folgende Bedingungen müssen erfüllt sein:

- Wurde eine gut strukturierte monolithische Anwendung entwickelt?
- Wurde ermittelt, welche Bedürfnisse durch Microservices erfüllt werden?
- Wurden die Teams im Hinblick auf die Microservices neu ausgerichtet?
- Werden im DevOps- und CI/CD-Bereich Best Practices umgesetzt?
- Wurden geschäftliche Grenzen innerhalb der Anwendung ermittelt?
- Wurden Tools und Prozesse für Orchestrierung und Management von Microservices implementiert?

## INNOVATIVE LÖSUNG FÜR EINE FLEXIBLE, REAKTIONSSCHNELLE IT

### SCHNELLERE BEREITSTELLUNG

Microservices haben einen kleineren Umfang, da der Fokus auf Domain-Grenzen und einer einheitlichen Domain-Modellierung liegt, und erfordern weniger Code. Einsatzstrategien wie fokussierte, eigenständige Archive – oftmals verpackt als Linux-Container – und CI/CD führen zu einer schnelleren, zuverlässigeren Bereitstellung. Dies hat zur Folge, dass der Softwareentwicklungszyklus im Allgemeinen beschleunigt wird. Neue Features und Fehlerbehebungen sowie umfassend getestete Sicherheitspatches werden zügiger bereitgestellt.

### MODULARE STEUERUNG

Bei der Nutzung von Microservices kann jeder Dienst individuell skaliert werden, um an zeitlich begrenzte oder saisonbedingte Traffic-Steigerungen angepasst zu werden, eine Batch-Verarbeitung durchzuführen oder andere betriebliche Anforderungen zu erfüllen. Eine bessere Fehlerisolation sorgt dafür, dass sich Serviceprobleme wie Speicherlecks oder offene Datenbankverbindungen jeweils nur auf den betreffenden Dienst auswirken. Die Skalierbarkeit von Microservices ergänzt die Flexibilität von Cloud Services und ermöglicht es, den Service zu verbessern und ohne Dienstunterbrechungen mehr Kunden simultan zu bedienen.

## MEHR AUSWAHL

Der Markt für Microservices wird von Open Source Technologielösungen und Organisationspraktiken bestimmt. Folglich reduzieren Microservices den Vendor Lock-in und verhindern eine langfristige technologische Bindung. Somit können Sie die Tools, die Sie zum Erreichen Ihrer IT- und Geschäftsziele benötigen, selbst auswählen.

## SCHAFFUNG EINER SOLIDEN GRUNDLAGE FÜR MICROSERVICES

Für einen erfolgreichen Einsatz von Microservices müssen Unternehmen zunächst eine solide Grundlage für ihre monolithische Architektur schaffen. Um die Vorteile von Microservices voll ausschöpfen zu können, müssen Modularität, Domain-Grenzen und die Grundlagen der Theorie verteilter Systeme berücksichtigt werden.

Darüber hinaus profitieren komplexere Systeme am meisten von Microservices. Auch wenn jeder Service vollkommen unabhängig ist, müssen bestimmte betriebliche Anforderungen erfüllt werden. Dazu gehören:

- DevOps
- PaaS
- Container oder unveränderliche VM
- Replikation, Registrierung und Erkennung von Services
- Proaktives Monitoring und Warnmeldungen

Da die Erfüllung dieser Anforderungen eine erhebliche Investition ohne unmittelbare Rendite bedeuten kann, stellen Microservices möglicherweise nicht für jedes Team oder Projekt eine kosteneffiziente Lösung dar. Durch einen „Monolith first"-Ansatz ist sichergestellt, dass die Anwendungsentwicklung auf soliden Designprinzipien beruht und Domain-Grenzen richtig definiert werden. Auf dieser Grundlage können Sie schrittweise auf eine Microservices-Architektur umstellen, wenn dies aus Gründen der Skalierbarkeit erforderlich ist. Eine einfache Warenkorb-Anwendung sollte zum Beispiel folgende Merkmale aufweisen:

- Separation of Concerns
- Starke Kohäsion und geringe Verknüpfung durch gut definierte Programmierschnittstellen (APIs)
- Separate Schnittstellen, APIs und Implementierungen gemäß dem Gesetz von Demeter, auch bekannt als Law of Demeter (LoD).
- Domain-driven Design zur Gruppierung verwandter Objekte

Der Erfolg einer Microservices-Architektur hängt nicht von der Technologie, sondern von der Organisationsstruktur eines Unternehmens ab. Voraussetzung sind flexible und autonome Teams mit einer flachen Organisationsstruktur und funktionsübergreifenden Kompetenzen.

Nachdem eine zu skalierende monolithische Anwendung den Prinzipien der Softwarearchitektur entsprechend entwickelt wurde, kann ein Refactoring in Microservices erfolgen. Die effektivste Refactoring-Methode besteht aus folgenden Schritten:

1. Ermitteln Sie die geschäftlichen Grenzen und semantischen Unterschiede in der Anwendungsdomain und zerlegen Sie jede Domain in ihre eigenen Microservices.
2. Suchen Sie die Komponente, die am öftesten auf Anfrage geändert wird – z. B. Updates von Business-Rules im Zusammenhang mit Preisberechnungen oder regulatorischen Änderungen – oder für die zur Schließung von Sicherheitslücken oft Patches bereitgestellt werden.
3. Nachdem die grundlegenden domainbasierten Microservices definiert wurden, sind die APIs anzupassen, die für die Interaktion von Services verwendet werden. Zur Gestaltung dieser APIs können Sie Aggregat-, Proxy-, Chain-, Event- oder andere Designmuster verwenden.

## AUFEINANDER ABGESTIMMTE, KOMPETENTE TEAMS

Der Erfolg einer Microservices-Architektur hängt nicht von der Technologie, sondern von der Organisationsstruktur eines Unternehmens ab. Voraussetzung sind flexible und autonome Teams mit einer flachen Organisationsstruktur und funktionsübergreifenden Kompetenzen.

Um ein effektives, kompetentes Team zu schaffen, muss eine Neuausrichtung der Mitarbeiter auf die Funktionalität anstatt auf die Architektur vorgenommen werden. Ein Beispiel sind hier die kleinen beweglichen Teams von Amazon mit 8 bis 10 Mitarbeitern (auch bekannt als „Two-Pizza-Teams“), die für die Entwicklung und Wartung des Service zuständig sind. Das Gesetz von Conway besagt, dass Unternehmen nur solche Designs erzeugen können, die ihre Organisationsstruktur abbilden. Sind Teams in verschiedene Aufgabenbereiche unterteilt, etwa Entwicklung, Betrieb, Qualitätssicherung und Sicherheit, kommt es zu einer beschränkten Flexibilität und zu Verzögerungen.

Durch die Einführung von DevOps und die Festlegung von Kommunikationsstrategien vor der Umstellung auf Microservices können diese Probleme vermieden oder abgeschwächt werden. Zudem lässt sich so verhindern, dass eine misslungene SOA anstelle einer effektiven Microservices-Architektur geschaffen wird.

## EFFEKTIVE MANAGEMENTTOOLS

Neben einer gut entworfenen Software und einem effektiven, organisierten Team erfordert eine hochgradig skalierbare Architektur Tools, die Sie beim Management zusätzlicher Services und Anwendungskomponenten unterstützen. Hierzu gehört u. a. Folgendes:

- Tools für die Registrierung und Erkennung von Services, z. B. Kubernetes
- Verpackungsstandards für die Containerisierung von Anwendungen (z. B. Docker) und Orchestrierungstools für die Container-Replikation in großem Maßstab (z. B. Kubernetes). Diese beiden bewährten Open Source-Technologien sind in OpenShift by Red Hat enthalten.
- Entwicklungstools für CI-Umgebungen, z. B. Jenkins oder Shippable für Docker und Kubernetes
- Tools zum Auflösen von Abhängigkeiten, z. B. Nexus
- Failover- und Resilienz-Tools, z. B. Bibliotheken wie Hystrix und Ribbon
- Tools für Service Monitoring, Warnmeldungen und Ereignisse, z. B. der ELK-Stack (ElasticSearch, LogStash und Kibana)

## DATENMANAGEMENT

Ein weiterer wichtiger Aspekt, der bei der Umstellung auf Microservices zu berücksichtigen ist, ist das Datenmanagement. Anders als bei einer SOA findet bei Microservices keine gemeinsame Nutzung von Daten statt. Vielmehr verfügt jeder Microservice über einen separaten physischen Datenspeicher und eine mehrsprachige Persistenz, die es ermöglicht, für jeden Microservice verschiedene Datenbank-Engines auszuführen. Auf diese Weise ist es möglich, für jeden Service einen individuellen Datenspeicher auszuwählen, statt alle Daten in einem unternehmensweiten relationalen Datenbankmanagementsystem (RDBMS) zu speichern.

Die Unterhaltung mehrerer Versionen einer Unternehmensdatenbank kann jedoch Lizenzkosten und Komplexität in die Höhe treiben. Außerdem müssen die Datenspeicher oft aus Konsistenzgründen aufeinander abgestimmt werden. Generische ETL- (Extract, Transform, Load) oder Datenvirtualisierungstools können die Datennormalisierung unterstützen. Das Event Sourcing wiederum ist ein verbreitetes Designmuster für die Anpassung von Datenspeichern an nachträgliche Änderungen.

## FAZIT

Eine Microservices-Architektur kann Unternehmen viele Vorteile bieten - von der unabhängigen Skalierbarkeit der einzelnen Anwendungskomponenten bis hin zur schnelleren, einfacheren Softwareentwicklung und -wartung. Doch Microservices sind nicht unbedingt für jedes Team oder Projekt von Nutzen und können eine erhebliche Investition ohne unmittelbare Rendite bedeuten. Die Umstellung auf Microservices sollte schrittweise vollzogen werden. Anstelle einer vollständigen Umstellung kann auch ein Refactoring für Teile bestehender Anwendungen von Vorteil sein. Für den erfolgreichen Einsatz von Microservices müssen Unternehmen zuerst eine gut durchdachte Anwendung erstellen, die vorhandenen Plattformstandards entspricht. Danach sollten sie ein Refactoring der Anwendung in eine Reihe von Microservices im Hinblick auf Geschäftsanforderungen durchführen. Mit den richtigen Mitarbeitern, Prozessen und Tools können Microservices die Entwicklung und Bereitstellung beschleunigen, die Wartung vereinfachen, die Skalierbarkeit verbessern und eine langfristige technologische Bindung verhindern.

### ÜBER RED HAT

Red Hat, der weltweit führende Anbieter von Open Source-Lösungen, folgt einem Community-basierten Ansatz um verlässliche und leistungsstrake Technologien in den Bereichen Cloud, Linux, Middleware, Storage und Virtualisierung bereitzustellen. Darüber hinaus bietet Red Hat einen vielfach ausgezeichnete Support-, Training- und Consulting-Services. Red Hat ist ein S&P 500-Unternehmen mit über 70 Niederlassungen weltweit, das seine Kunden und Partner mithilfe hochwertiger Services und Technolgien dabei unterstützt, Ihr Geschäft voranzutreiben.

EUROPA, NAHOST UND AFRIKA (EMEA)
00800 7334 2835
de.redhat.com
europe@redhat.com

TÜRKEI
00800-448820640

ISRAEL
1-809 449548

VAE
8000-4449549



facebook.com/redhatinc
@redhatnews
linkedin.com/company/red-hat

de.redhat.com
INC0336100_0216