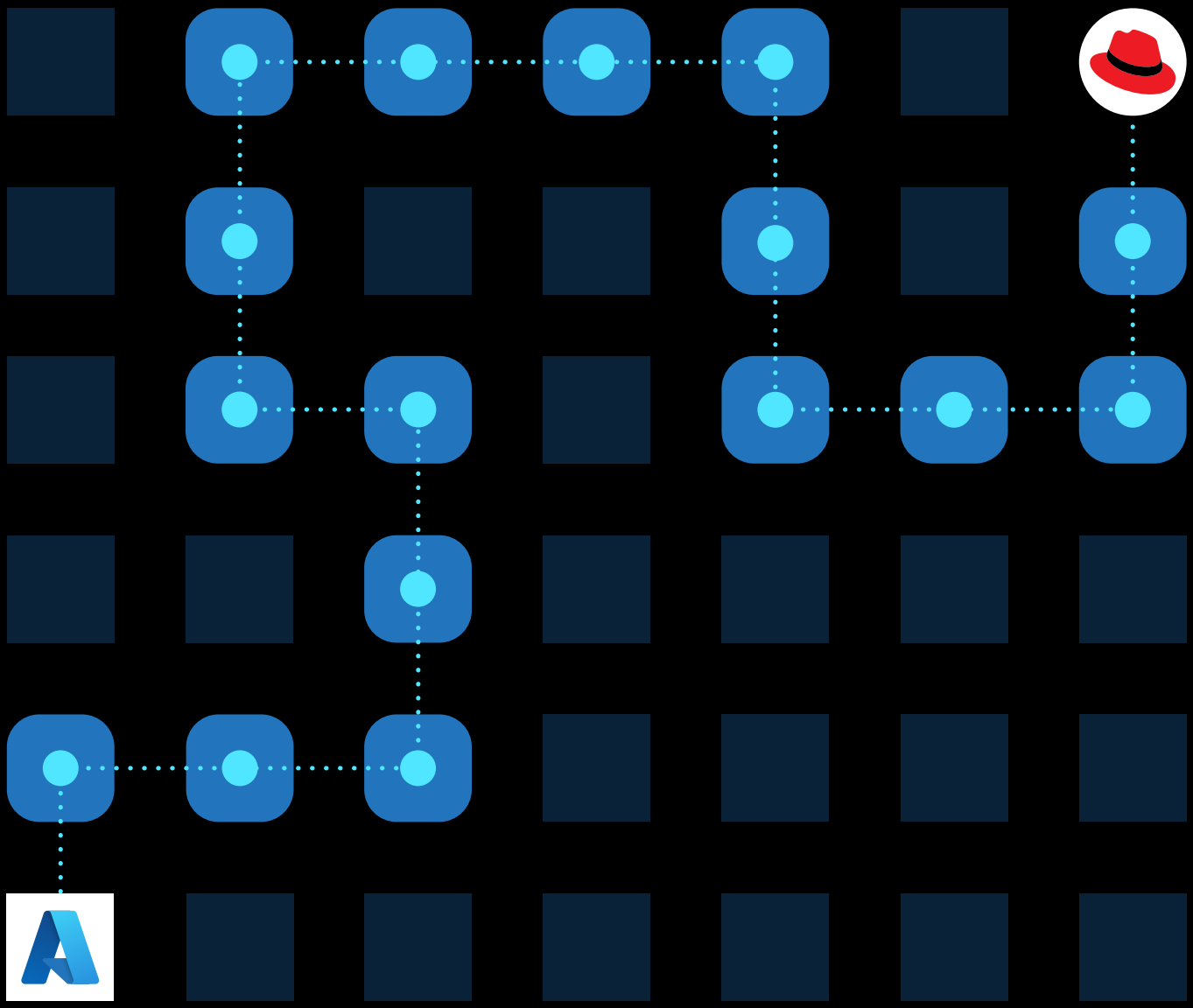
E-Book-Serie

Microsoft Azure

Erste Schritte mit Azure Red Hat OpenShift

Vom Proof of Concept zur Produktion



Erste Schritte mit Azure Red Hat OpenShift

Kapitel 1

Kontaktaufnahme mit Microsoft und Red Hat

Falls Sie Azure Red Hat OpenShift in Erwägung ziehen, möchten wir uns mit Ihnen in Verbindung setzen und uns darüber austauschen. Dieser Guide bietet ist zwar eine hilfreiche Unterstützung für die Verwendung der Plattform, jedoch können Red Hat und Microsoft viele Ressourcen bereitstellen, die über diesen Guide hinausgehen und Ihr Erlebnis weiter ergänzen. Wir möchten gemeinsam sicherstellen, dass Azure Red Hat OpenShift die Anwendungsplattform ist, die Ihre Innovationsanforderungen an Anwendungen erfüllt.

Azure Red Hat OpenShift wurde wegen einem einfachen Grund entwickelt: Kunden wie Sie haben danach gefragt. Unsere gemeinsamen Kunden – sowohl Unternehmen als auch kleine Organisationen – stellen das Produktportfolio von Red Hat immer auf Microsoft Azure bereit. Obwohl OpenShift aus Azure bereits seit mehreren Jahren als selbst gemanagtes Angebot unterstützt wird, erfordern die Einrichtung, das Deployment und die „Day 2 Operations" im Zusammenhang mit der Clusterverwaltung Expertise mit Kubernetes und Zeit. Dies kostet wichtige Zeit, die Sie für Ihre geschäftlichen Ziele benötigen.

Während mehr und mehr Kunden erfolgreiche Deployments mit dem selbst gemanagten Angebot Azure Red Hat OpenShift implementieren, merken wir, dass sie weitaus weniger Zeit für die Einrichtung und für die „Day 2 Operations" und mehr für ihre Anwendungen aufwenden.

Wir haben diesen Guide auf Grundlage unserer praktischen Erfahrung verfasst, um unseren Kunden Best Practices für die Anwendungsentwicklung in Azure Red Hat OpenShift bereitzustellen. Dieser Guide soll lediglich zur Unterstützung dienen. Sie können die Kapitel in Reihenfolge lesen – von der Einführung bis zum Fazit – oder einfach nach den von Ihnen benötigten spezifischen Informationen suchen.

Zielgruppe für diesen Guide

Dieser Guide ist für eine Zielgruppe mit technischer Expertise gedacht wie Entwickler, Operatoren und Plattformarchitekten, die lernen möchten, wie sie ihre Fertigkeiten in Bezug auf Erstellung und Deployment von Anwendungen durch Nutzung von Azure und Red Hat OpenShift für das Full-Service-Deployment vollständig gemanagter Red Hat OpenShift-Cluster stärken.

Inhalt dieses Guides

In diesem Guide stellen wir Ihnen die wichtigsten Themen vor, die Sie für das Verständnis und für die Einführung von Azure Red Hat OpenShift benötigen – vom Proof of Concept zum Produktiv-Deployment.

Kapitel 2, *Einführung in Red Hat OpenShift*, beginnen wir mit einer Einführung in Red Hat OpenShift und die Gründe, warum so viele Entwickler, Operatoren und Plattformarchitekten es als ihre Anwendungsplattform verwenden, und wie sie einen großen Nutzen daraus ziehen. Anschließend erfahren Sie, wieso Red Hat OpenShift für viele Kunden die bevorzugte Plattform ist.

Kapitel 3, *Azure Red Hat OpenShift,* erläutert den gemanagten Service und wie dieser für Sie, dem Kunden, bereitgestellt wird.

Kapitel 4, *Vor der Provisionierung – Fragen zur Unternehmensarchitektur*, behandelt wichtige Fragen, die Sie sich stellen sollten, bevor Sie mit dem Bereitstellen mit Azure Red Hat OpenShift beginnen. Dazu gehören einige unterstützende Inhalte zu Best Practices, die sich aus den Erfahrungen von Kunden ableiten, die sich mit der Unternehmensarchitektur auseinandergesetzt haben.

Kapitel 5, *Provisionierung eines Clusters von Azure Red Hat OpenShift,* führt die wichtigsten Ressourcen in der offiziellen Dokumentation auf, die Sie für das Bereitstellen eines Clusters von Azure Red Hat OpenShift benötigen.

Kapitel 6, *Nach der Provisionierung – Day 2,* behandelt die Aufgaben nach der Provisionierung, oft auch „Day 2" genannt. Dieser Guide erklärt jeweils, wie der gemanagte Service Azure Red Hat OpenShift solche Aufgaben einfacher macht, als wenn Sie sich selber mit diesen Themen auseinandersetzen.

Kapitel 7, *Anwendungen auf Red Hat OpenShift bereitstellen,* ist eine kurze Anleitung zum Bereitstellen von Anwendungen auf der Plattform. Es ist aber erwähnenswert, dass es hier keinen Unterschied zur Ausführung von Red Hat OpenShift in anderen Umgebungen gibt.

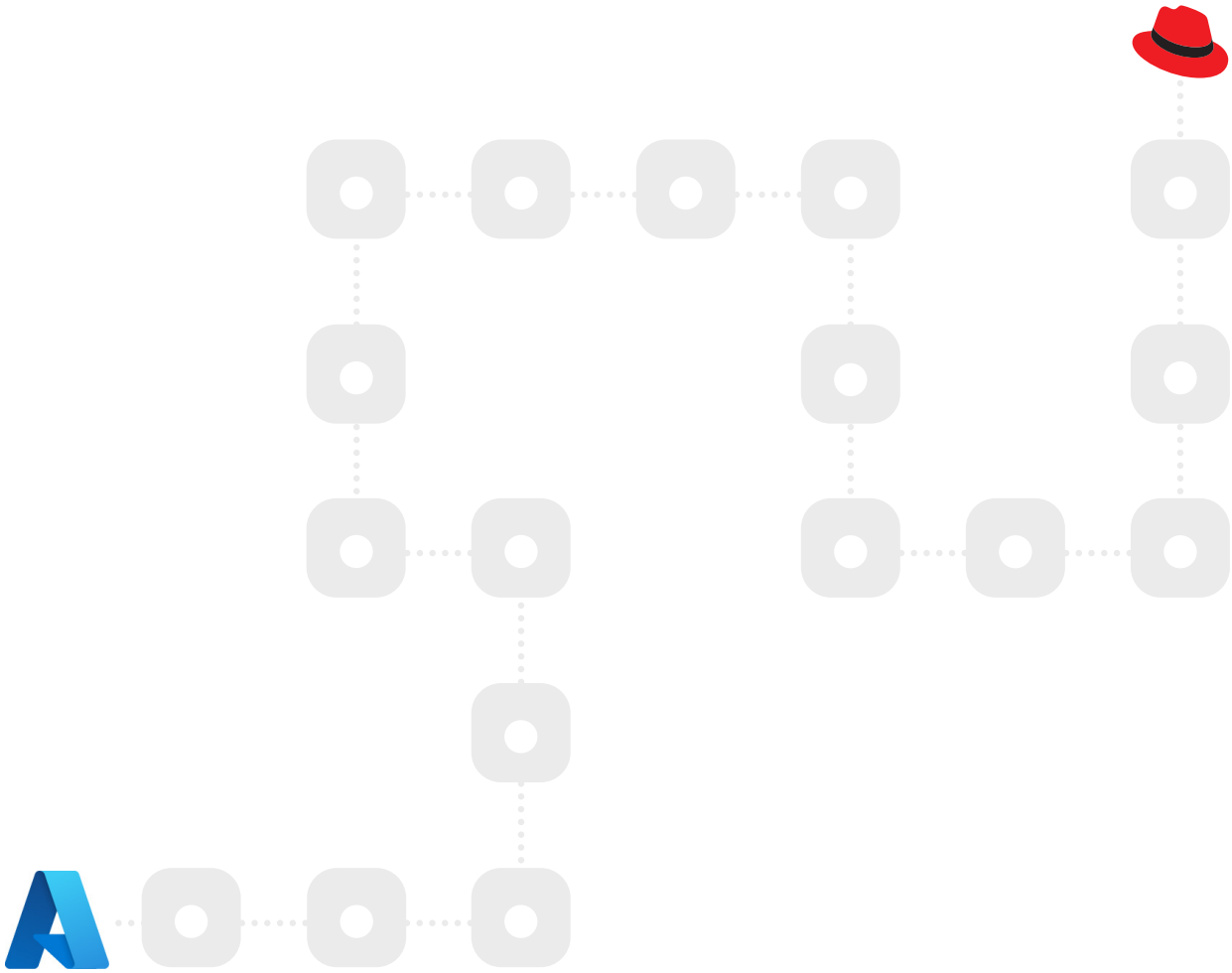
Kapitel 8, *Erkunden der Anwendungsplattform,* bietet einen Überblick über einige wichtigen Funktionen der Anwendungsplattform – Container-Registry, Pipelines, Serverless und ähnliches.

Kapitel 9, *Integration mit anderen Services,* behandelt die Plattformintegration mithilfe von Azure Service Operator, Azure DevOps und ähnlichen Services.

Kapitel 10, *Onboarding von Workloads und Teams,* bietet abschließend einige Best Practices und Empfehlungen zum Onboarding von Anwendungsteams und dazu, wie Sie die Einführung des Services innerhalb Ihrer Organisation vorantreiben.

Kapitel 11, *Fazit,* enthält das Fazit.

Kapitel 12, *Glossar,* dient als Glossar.



Kapitel 2

Einführung in Red Hat OpenShift

Dieses Kapitel enthält eine kurze Einführung in Red Hat OpenShift, die Verwendung und die Vorteile des Services, von denen Kunden von Red Hat üblicherweise profitieren. Somit erhalten Sie einen nützlichen Überblick, falls Sie zum ersten Mal mit OpenShift zu tun haben, oder eine Auffrischung, falls Sie weiterführende Information zur Plattform erhalten möchten.

Red Hat OpenShift – Überblick

Red Hat OpenShift ist eine unternehmensgerechte Anwendungsplattform, auf der Operationen für den gesamten Stack automatisiert werden, um Deployments in Hybrid Clouds und Multi-Clouds noch einfacher verwalten zu können. Sie ist daraufhin ausgelegt, die Entwicklerproduktivität zu steigern und Innovationen zu fördern. Dank automatisierten Abläufen und optimiertem Lifecycle-Management unterstützt Red Hat OpenShift Entwickler-Teams beim Erstellen und dem Bereitstellen neuer Anwendungen und hilft Operations-Teams beim Provisionieren, Managen und Skalieren von Kubernetes-Plattformen.

Entwickler-Teams haben Zugriff auf validierte Bilder und Lösungen von Hunderten von Partnern mit Security Scanning und kryptographischer Signierung während des Bereitstellungsprozesses. Sie können auf On-Demand-Images zugreifen und nativen Zugriff auf eine breite Palette an Cloud-Services von Drittanbietern erhalten – alles über eine einzige Plattform.

Operations-Teams erhalten Transparenz in Deployments überall und teamübergreifend, mit integrierter Protokollierung und Überwachung. Red Hat Kubernetes Operators betten die besondere Anwendungslogik ein, durch die der Service funktional ist. Sie ist nicht einfach konfiguriert, sondern auch für Performance optimiert sowie vom BS per Touch aktualisiert und gepatcht. Kurz gesagt, bietet Red Hat OpenShift alles, damit Organisationen das Potenzial der IT- und Entwickler-Teams voll ausschöpfen können.

OpenShift Cloud Services – Kostenersparnisse und Unternehmensvorteile

Tausende von Kunden vertrauen Red Hat OpenShift, um die Art und Weise zu ändern, wie sie Anwendungen bereitstellen, ihre Kundenbeziehungen verbessern und einen Wettbewerbsvorteil erzielen, um in ihren Branchen führend zu sein. Die Studie von IDC, die in Der Business Value von Red Hat OpenShift, März 2021 beschrieben wird, zeigt den erheblichen Nutzen, den die befragten Organisationen mit der Red Hat OpenShift Platform erhalten, indem sie hochwertigere und zeitnähere Anwendungen und Funktionen für ihre Unternehmen bereitstellen und gleichzeitig ihre Entwicklungs- und IT-bezogene Kosten und Anforderungen an den Zeitaufwand optimieren können.

Dies sind die wichtigsten Metriken:

- 636 % Return on Investment innerhalb von fünf Jahren
- Amortisierung in zehn Monaten
- 54 % weniger Betriebskosten über fünf Jahre
- Dreimal so viele neue Funktionen pro Jahr
- 20 % höhere Produktivität der Anwendungsentwickler
- 71 % weniger ungeplante Ausfallzeiten
- 21 % mehr Effizienz bei IT-Infrastruktur-Teams

Quelle: Von Red Hat gesponsertes IDC Whitepaper. *Der Business Value von Red Hat OpenShift*. Dok.-Nr. US47539121, März 2021.

OpenShift Cloud Services, inklusive Azure Red Hat OpenShift, bieten zusätzlich zum Mehrwert von Red Hat OpenShift weitere geschäftliche Vorteile. In der Studie von Forrester mit dem Namen Total Economic Impact™ von Red Hat OpenShift Cloud Services – Kosteneinsparungen und Unternehmensvorteile werden mehrere wichtige finanzielle Vorteile hervorgehoben.

Dies sind die wichtigsten finanziellen Vorteile der OpenShift Cloud Services:

- 468 % Return on Investment (ROI)
- 4,08 Mio. USD Kapitalwert (Net Present Value, NPV)
- Amortisierung in sechs Monaten

Über diese finanziellen Vorteile hinaus, gehören die folgenden wichtigen Erkenntnisse des Forrester-Berichts zu quantifizierten Vorteilen:

- **Beschleunigung der Entwicklung**: Die Verwendung von Red Hat OpenShift Cloud Services ermöglicht Organisationen die Verkürzung ihrer Entwicklungszyklen von bis zu 70 %.
- **20 % Zurückgewinnung von Arbeitszeit von Entwicklern, die zuvor aus Wartungsarbeiten für die Infrastruktur bestand**: Die Befragten merkten an, dass Red Hat OpenShift Cloud Services die Notwendigkeit der Wartung der Entwicklungsinfrastruktur für die Anwendung beseitigt haben, sodass sich die Entwickler voll und ganz der Entwicklung des Produkts oder der Lösung widmen können. Über drei Jahre hinweg bedeutet diese Zurückgewinnung der Entwicklerzeit eine Ersparnis von mehr als 2,3 Millionen USD.
- **Um 50 % verbesserte Betriebseffizienz**: Seitdem Red Hat OpenShift Cloud Services gemanagte Services sind, merkten die Befragten an, dass sie durch die Verwendung der Lösung 50 % der DevOps-Mitarbeiter, die zuvor für die Verwaltung der Infrastruktur verantwortlich waren, produktivere Arbeiten zuweisen können. Über drei Jahre hinweg bedeutet diese höhere Betriebseffizienz eine Ersparnis von mehr als 1,3 Millionen USD.*

Der Bericht enthält auch die folgenden nicht quantifizierten Vorteile:

- **Entwicklerzufriedenheit und -bindung**: Die Befragten betonten, dass Entwickler von Red Hat OpenShift Cloud Services profitierten, da sie Updates in kleinere Teile aufteilen können, wodurch sich der Zeitdruck beim umfangreichen Testen reduziert und weniger Notfälle in der Produktivumgebung auftreten.
- **Sicherheit und reduziertes Risiko**: Die Befragten teilten mit, dass Red Hat OpenShift Cloud Services bestimmte Funktionen und Sicherheitsupdates automatisieren, sodass die Notwendigkeit der manuellen Wartung wegfällt und gleichzeitig sichergestellt wird, dass ihre Umgebung sicher bleibt.
- **Zuverlässigkeit**: Die Befragten merkten an, dass die Verwendung von Red Hat OpenShift Cloud Services ihre Anwendungsplattform langfristig zuverlässiger machten, da selbst mit einer expandierenden Umgebung weniger Systemausfälle auftreten.
- **Portierbarkeit und Geschäftskontinuität**: Die Befragten merkten außerdem an, dass Red Hat OpenShift Cloud Services die Geschäftskontinuität gewährleisteten und aufgrund der Portierbarkeit, Skalierbarkeit und Flexibilität bei ihrer Disaster-Recovery-Strategie eine unterstützende Rolle spielten.

Quelle: Forrester, *Der Total Economic Impact von Red Hat OpenShift Cloud Services*, Dezember 2021

***Weitere Informationen:** Unternehmen beschleunigen ihre Agilität mit Red Hat Cloud Services

„Red Hat OpenShift oder nur Kubernetes?" – Die Kosten für die Entwicklung der eigenen Kubernetes-Anwendungsplattform

Red Hat OpenShift wird oft als „Kubernetes für Unternehmen" bezeichnet, aber was heißt das eigentlich? Kunden fragen oft „OpenShift oder nur Kubernetes"? Es ist aber wichtig zu erwähnen, dass **OpenShift bereits Kubernetes verwendet**. Kubernetes stellt im Kontext der gesamten OpenShift-Architektur sowohl die Basis, auf der die OpenShift-Plattform entwickelt wurde, und viele Tools zur Ausführung von OpenShift bereit.

Kubernetes ist ein überaus wichtiges Open-Source-Projekt – eines der wichtigsten Projekte der Cloud Native Computing Foundation und eine essenzielle Technologie bei der Containerausführung.

Die eigentliche Frage, die potenzielle Benutzer von OpenShift jedoch möglicherweise stellen, ist: „**Kann ich meine Anwendungen einfach nur mit Kubernetes ausführen**?" Viele Organisationen beginnen mit dem Bereitstellen von Kubernetes und merken, dass sie innerhalb von ein paar Tagen Container oder sogar Unternehmensanwendungen ausführen können. Jedoch gehen mit den „Day 2 Operations" Sicherheitsanforderungen einher und weitere Anwendungen werden bereitgestellt, sodass einige Organisationen in die Falle tappen, ihre eigene **Platform as a Service (PaaS)** mit Kubernetes zu entwickeln. Sie fügen einen Open-Source-Ingress-Controller hinzu, schreiben ein paar Skripts zur Verbindung mit ihren **Continuous Integration/Continuous Deployment (CI/CD)**-Pipelines und versuchen anschließend eine komplexere Anwendung bereitzustellen… und dann fangen die Probleme an. Wenn das Bereitstellen von Kubernetes die Spitze des Eisbergs ist, dann ist die Komplexität des „Day 2"-Managements der versteckte, riesige, schiffeversenkende Rest desselben Eisbergs unter Wasser.

Es ist zwar möglich, mit der Lösung dieser Herausforderungen und Probleme zu beginnen, jedoch werden meist ein Operations-Team mit mehreren Personen und einige Wochen und Monate an Entwicklungs- und Wartungsaufwand für diese „benutzerdefinierte PaaS auf Kubernetes" benötigt. Dies führt zu ineffizienten Abläufen in der Organisation, Komplexität bei der Unterstützung im Hinblick auf die Sicherheit und Zertifizierung und es muss beim Onboarding von Entwickler-Teams alles von Grund auf neu entwickelt werden. Wenn eine Organisation alle Aufgaben im Zusammenhang mit der Entwicklung und Wartung dieser benutzerdefinierten Kubernetes-Plattform aufzählen würde, inklusive aller Extra-Komponenten, die für die erfolgreiche Ausführung von Containern erforderlich sind, könnte man sie in folgende Aufgabengruppen aufteilen:

- **Cluster-Management**: Dies beinhaltet das Installieren und Patchen von Betriebssystemen, das Installieren von Kubernetes, das Konfigurieren des CNI-Networkings, die Integration von Authentifizierung, das Einrichten von Ingress und Egress, persistentem Storage, Härtungs-Nodes, Sicherheits-Patching und das Konfigurieren der zugrundeliegenden Cloud/Multicloud.
- **Anwendungs-Services**: Hierzu gehören Log Aggregation, Zustandsprüfungen, Performance-Überwachung, Sicherheits-Patching, Container-Registry und die Einrichtung des Staging-Prozesses für Anwendungen.
- **Entwickler-Integration**: Dies beinhaltet die CI/CD-Integration, Entwicklertools/IDE-Integration, Framework-Integration, Middleware-Kompatibilität, die Bereitstellung von Performance-Dashboards für Anwendungen und RBAC.

Es gibt zwar einige Aktionen und Technologien, die man dieser Liste hinzufügen könnte – die meisten dieser Aktivitäten sind essenziell für alle Organisationen, die Container verwenden – jedoch ist die Komplexität, die Zeit und der Aufwand für die Einrichtung all dieser Dinge unerheblich im Vergleich mit der weiterführenden Wartung dieser einzelnen Elemente. Jede Integration erfordert umfassende Tests und jede Komponente und Aktivität hat voneinander abweichende Release-Zyklen, Sicherheitsrichtlinien und Patches.

Welche Vorteile von Red Hat OpenShift gibt es gegenüber Kubernetes?

Wenn eine Organisation damit beginnt Container, die nur auf Kubernetes entwickelt wurden, in einer Produktivumgebung auszuführen, werden die im vorherigen Abschnitt erwähnten Komponenten normalerweise zusammen installiert und integriert, sodass eine Art benutzerdefinierte Anwendungsplattform entsteht.

Azure Red Hat OpenShift kombiniert alle im vorherigen Abschnitt erwähnten Komponenten in einer einzigen Plattform. Abläufe für IT-Teams gestalten sich einfacher und Anwendungsteams können auf die für die Ausführung von Aufgaben benötigten Komponenten zugreifen. All diese Themen werden im Detail später im Guide behandelt. Sehen wir uns nun also einige der Hauptunterschiede zwischen beiden an:

- **Einfaches Deployment:** Das Deployment einer Anwendung in Kubernetes kann zeitaufwändig sein. Dies beinhaltet das Abrufen des GitHub-Codes auf einen Rechner, das Hochfahren eines Containers, das Hosten des Containers in einer Registry wie Docker Hub und schließlich das Verstehen Ihrer CI/CD-Pipeline, was sehr kompliziert sein kann. OpenShift, auf der anderen Seite, automatisiert die schwierigen Aufgaben und die Backend-Arbeit. Es erfordert nur die Erstellung eines Projekts und den Upload Ihres Codes.
- **Sicherheit:** Heute werden die meisten Kubernetes-Projekte von mehreren Entwicklern und Operatoren im Team bearbeitet. Auch wenn Kubernetes jetzt beispielsweise Kontrollen wie RBAC und IAM unterstützt, ist dennoch eine manuelle Einrichtung und Konfiguration erforderlich, die Zeit in Anspruch nimmt. Red Hat und OpenShift haben bei der Identifizierung von Best Practices für Sicherheit durch jahrelange Erfahrung einen großartigen Job gemacht. Diese stehen Kunden standardmäßig zur Verfügung. Sie fügen einfach neue Nutzer hinzu und OpenShift kümmert sich um Dinge wie Namespacing und Erstellung unterschiedlicher Sicherheitsrichtlinien.
- **Flexibilität**: Durch die Verwendung von Azure Red Hat OpenShift können Sie von bekannten Best Practices für Deployment, Management und Aktualisierung profitieren. Alle schwierigen Aufgaben im Backend werden für Sie durchgeführt, sodass Sie Ihre Apps schneller bereitstellen können. Für Teams, die Anweisungen erhalten möchten, wie sie Dinge erledigen und von einem optimierten Ansatz profitieren, ist dies vorteilhaft. Die Kubernetes-Plattform ermöglicht jedoch die manuelle Anpassung Ihrer CI/CD-DevOps-Pipeline, die mehr Raum für Flexibilität und Kreativität bei der Entwicklung Ihrer Prozesse bietet.

- **Tägliche Abläufe:** Cluster bestehen aus einer Gruppe mehrerer VMs und Ihre Operations-Teams müssen zwangsläufig neue VMs hochfahren, die zu einem Cluster hinzugefügt werden. Der Konfigurationsprozess über Kubernetes kann zeitaufwändig und komplex sein. Skripts müssen für die Einrichtung von Selbstregistrierung oder Cloud-Automatisierung entwickelt werden. Mit Azure Red Hat OpenShift werden Cluster-Provisionierung, -Skalierung und -Upgrade automatisiert und von der Plattform gemanagt.
- **Management**: Sie können zwar die Standardtools und -dashboards von Kubernetes verwenden, die in allen Distributionen enthalten sind, jedoch benötigen die meisten Entwickler eine umfassendere und robustere Plattform. Azure Red Hat OpenShift bietet eine hervorragende Webkonsole, die auf Kubernetes APIs und den dazugehörigen Funktionen basiert, mit denen Operations-Teams ihre Workloads managen können.

Kapitel 8, Erkunden der Anwendungsplattform, enthält eine anleitende Beschreibung vieler wichtiger Funktionen von Azure Red Hat OpenShift, die einen Mehrwert bieten.

Zusammenfassung

Azure Red Hat OpenShift ist für viele gemeinsame Kunden von Red Hat und Microsoft die bevorzugte Anwendungsplattform zur Einführung von containerisierten Anwendungen.

Im nächsten Kapitel erkunden wir Red Hat OpenShift als Cloud-Service und gehen auf die Architektur, die Integration und das Management ein.

Kapitel 3

Azure Red Hat OpenShift

Mit **Azure Red Hat OpenShift** können Sie vollständig gemanagte Red Hat OpenShift-Cluster bereitstellen, ohne sich Sorgen über die Erstellung und das Management der Infrastruktur für die Ausführung zu machen.

Azure Red Hat OpenShift ist eine cloudnative On-Demand-Anwendungsplattform und wird gemeinsam von Red Hat und Microsoft entwickelt, betrieben und unterstützt. Ein spezialisiertes **Site Reliability Engineering** (**SRE**)-Team automatisiert, skaliert und sichert die OpenShift-Cluster und arbeitet gemeinsam, um eine integrierte Supporterfahrung zu liefern. Mit Azure Red Hat OpenShift müssen keine virtuelle Maschinen betrieben werden und Patching ist nicht erforderlich. Control Plane-Nodes und Worker-Nodes werden von Red Hat und Microsoft für Sie gepatcht, aktualisiert und überwacht. Ihre Azure Red Hat OpenShift-Cluster werden in Ihrer Azure-Subskription bereitgestellt und sind auf Ihrer Azure-Rechnung enthalten.

Stellen Sie Anwendungen mit Azure Red Hat OpenShift schneller bereit:

- Unterstützen Sie Entwickler bei Innovationen über integrierte CI/CD-Pipelines und der einfachen Verbindung Ihrer Anwendungen mit Hunderten von Azure-Services wie MySQL, PostgreSQL, Redis und Azure Cosmos DB.
- Entfernen Sie mit der automatisierten Provisionierung, Konfiguration und Abläufen die Komplexität und Produktivitätsbarrieren.
- Skalieren Sie nach Ihren Anforderungen und starten Sie ein hoch verfügbares Cluster mit drei Worker-Nodes in wenigen Minuten, skalieren Sie bei Änderungen des Anwendungsbedarfs. Zusätzlich haben Sie eine Auswahl an standardmäßigen Worker-Nodes mit viel Arbeitsspeicher oder großer CPU.
- Abläufe, Sicherheit und Compliance der Enterprise-Klasse mit einem integrierten Support-Erlebnis

Als Nächstes gehen wir auf die technischen Details der Architektur und Funktionsweise von Red Hat OpenShift ein.

Architektur

Azure Red Hat OpenShift nutzt Azure-Infrastruktur-Services wie virtuelle Maschinen, Netzwerk-Sicherheitsgruppen, Storage-Konten und andere Azure-Services als Basis für die Red Hat OpenShift-Lösung. Die Architektur von Azure wird außerdem in Ihrer Azure-Subskription vollständig bereitgestellt. Somit ist es einfach, mit anderen Services zu integrieren, die bereits in Ihrem Konto ausgeführt werden.

OpenShift baut auf dem Red Hat Enterprise Linux CoreOS auf, welches die Microservices-basierte Architektur von kleineren, entkoppelten, zusammenarbeitenden Einheiten hostet. Auf jeder virtuellen Maschine wird der `kubelet`-Service ausgeführt. Dieser ist die Basis des Kubernetes-Clusters. Die Datenbank hinter dieser Architektur, die in der Control Plane ausgeführt wird, ist **etcd**, ein zuverlässiger geclusterter Schlüssel-Wert-Speicher.

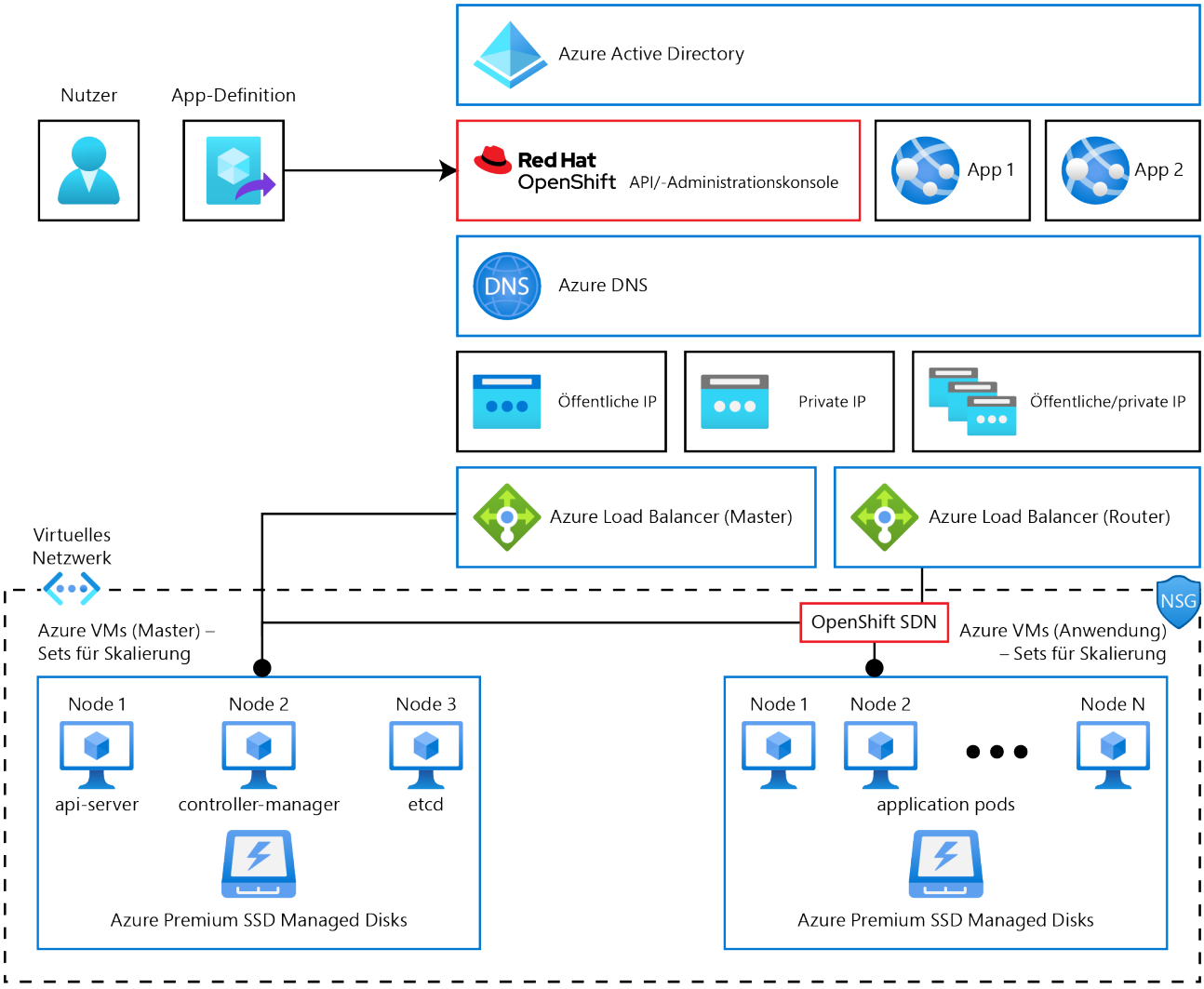


Abbildung 3.1: Architektur von Azure Red Hat OpenShift

Die folgenden Abschnitte, *Computing – Kontroll- und Worker-Nodes sowie Networking* führen die Inhalte dieses Diagramms weiter aus.

Computing – Kontroll- und Worker-Nodes

Die Architektur von Red Hat OpenShift wird auf virtuellen Maschinen (Nodes) ausgeführt, die eine von drei spezifischen Rollen innehaben: Kontrolle, Infrastruktur oder Anwendung. Version 4 von Azure Red Hat OpenShift unterstützt jedoch zum Zeitpunkt der Veröffentlichung noch keine Infrastruktur-Nodes. Daher geht dieses Buch nur auf Kontroll- und Worker-Nodes ein.

Die **Kontroll**-Nodes (früher **Master**-Nodes genannt) sind virtuelle Maschinen, die essenzielle Komponenten für den Kubernetes-Cluster enthalten. Dazu gehören der API-Server, der Controller Manager Server, der Scheduler und etcd. Die Kontroll-Nodes managen die Kubernetes-Cluster und planen die Pod-Ausführung auf den Worker-Nodes.

- **Der Kubernetes-API-Server** validiert und konfiguriert die Daten für Pods, Services und Replication Controller. Zudem ist er der zentrale Punkt für den gemeinsamen Zustand des Clusters.
- **Der Kubernetes Controller Manager** überprüft etcd auf Änderungen an Objekten wie Replication-, Namespace- und Service Account Controller-Objekte und verwendet die API, um den angegebenen Zustand zu erzwingen. Einige dieser Prozesse erstellen einen Cluster mit einem aktiven Leader nach dem anderen.
- **Der Kubernetes Scheduler** hält nach neu erstellten Pods ohne zugewiesene Nodes Ausschau und wählt den bestmöglichen Node für das Hosten des Pods aus.
- **etcd** speichert den persistenten Masterzustand, während andere Komponenten auf Änderungen in etcd warten, um sich dann selbst in den definierten Zustand zu bringen.

Die **Anwendungs**-Nodes sind die eigentlichen Ausführungsorte Ihrer Anwendungen.

Jeder Node im Kubernetes-Cluster führt einen Service namens `kubelet` aus, welcher das **Container Runtime Interface** (**cri-o**) einen Service-Proxy und einige andere essenzielle Services auf jedem Node aufrechterhält. Alle Nodes sind mit der Software-Defined-Networking-Technologie von OpenShift verbunden. Diese ist in Azure Red Hat OpenShift ein **Open Virtual Network (OVN)**, welches auf einem virtuellen Netzwerk von Azure ausgeführt wird.

Azure Red Hat OpenShift erstellt Nodes, die auf virtuellen mit Azure-Datenträgern für Storage verbundenen Azure-Maschinen ausgeführt werden. Sie werden möglicherweise feststellen, dass sie ziemlich große Datenträger haben – 1 TB. Das kommt daher, dass die IOPS-Garantie für die Storage-Performance in Azure mit der Größe des Datenträgers zusammenhängt. 1 TB garantiert eine ausreichende Bandbreite für den von der etcd-Datenbank zugrundeliegende Datenträger.

Networking

Azure Red Hat OpenShift benötigt ein einzelnes virtuelles Netzwerk von Azure mit zwei konfigurierten Subnetzen – eines für die Control Plane-Nodes und eines für die Worker-Nodes. Die Mindestgröße beider Netzwerke ist /27 (32 Adressen). Achten Sie jedoch darauf, die Größe nicht zu klein einzustellen, falls Sie vorhaben, den Cluster später zu skalieren.

Die zwei Azure Load Balancer (im Diagramm als **Master** und **Router** aufgeführt) leiten den Datenverkehr wie folgt:

- Master-/Control Plane Load Balancer: Sendet den Ingress-Datenverkehr für Nutzer der OpenShift API. Er stellt außerdem die ausgehende Konnektivität für die Control Plane-Nodes bereit.
- Router-/Application Load Balancer: Sendet Ingress-Datenverkehr über einen OpenShift-„Router" oder einen Ingress-Controller an die in OpenShift ausgeführten Anwendungen. Er stellt außerdem die ausgehende Konnektivität für die Worker-Nodes bereit.

Einen hervorragenden Artikel zur Networking-Konfiguration, den Anforderungen und Einschränkungen finden Sie in der **Networking-Dokumentation**.

Integration mit anderen Azure-Services

Azure Red Hat OpenShift kann als nativer Service auf Azure zusammen und integriert mit vielen Services von Azure, die Sie bereits verwenden, bereitgestellt werden. Nachfolgend finden Sie eine allgemeine Übersicht einiger Infrastruktur-Services von Azure, für die gängige Integrationen existieren.

Abbildung 3.2 zeigt viele der gängigen Azure-Service-Integrationspunkte für OpenShift auf Azure:

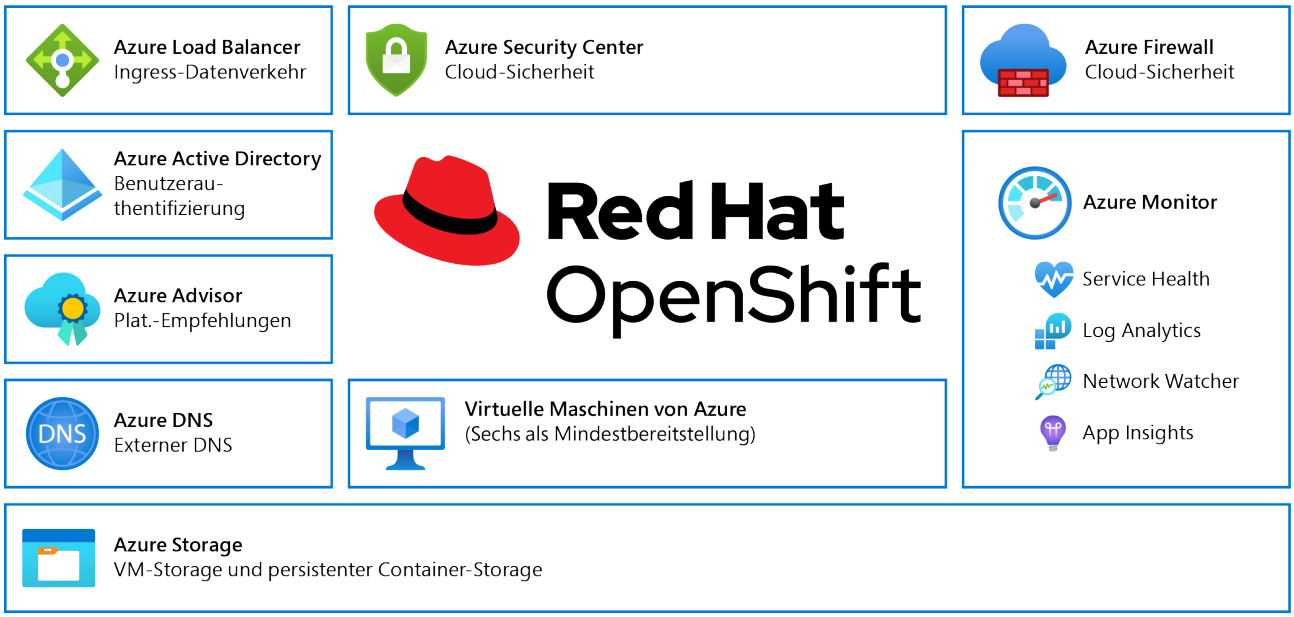


Abbildung 3.2: Gängige Azure-Service-Integrationspunkte

Außerdem können auf OpenShift ausgeführte Anwendungen mit Azure Service Operator sehr nah mit Azure-Services integriert werden. Sie finden hierzu später mehr Details unter *Kapitel 9, Integration mit anderen Services*.

Management

Um zu verstehen, was genau als Teil des Azure Red Hat OpenShift-Service gemanagt wird, hilft es, alle Elemente vom Rechenzentrum bis zu den Cluster-Operatoren als gemanagt zu betrachten. Cluster-Operatoren sind Services, die auf den Control Plane-Nodes ausgeführt werden und für die Überwachung, Updates und für den Zustand des Clusters verantwortlich sind. Dies bedeutet, dass Microsoft und Red Hat diese Komponenten überwachen, warten und managen, um den Cluster online zu halten. Alles was sich über den Cluster-Operatoren befindet, liegt in der Verantwortung des Kunden.

Als Kunde von Azure Red Hat OpenShift erhalten Sie vollständigen **cluster-admin**-Zugriff auf den Cluster, was bedeutet, dass Sie teilweise die Verantwortung übernehmen, keine Teile Ihres Clusters kaputtzumachen. Es ist wichtig die Support-Richtlinie zu verstehen, um zu wissen, was Sie mit dem Cluster machen dürfen und was nicht.

Eine detaillierte Erklärung dazu, für was genau Microsoft und Red Hat als Teil des Cloud-Services zuständig sind, finden Sie in der Verantwortungsmatrix von Azure Red Hat OpenShift.

Authentifizierung und Autorisierung

Azure Active Directory bietet eine gängige Authentifizierungsmethode für Azure Red Hat OpenShift-Cluster. Dies ist jedoch nicht obligatorisch. Andere Authentifizierungsmechanismen wie die Anmeldung mit GitHub oder einfache „Passwortdatei" können als Alternativen genutzt werden.

Wenn Azure Active Directory verwendet wird, leiten Azure Red Hat OpenShift und die Kubernetes APIs die Authentifizierungsanfragen weiter. Nutzer müssen Ihre Zugangsdaten angeben und werden anhand Ihrer Rollen autorisiert.

Die Layer für die Authentifizierung identifiziert den Nutzer dieser Anfragen bei der Azure Red Hat OpenShift-API. Sie verwendet Informationen zu diesem Nutzer, um zu bestimmen, ob die Anfrage zugelassen werden soll.

Die Konfigurationsanweisungen für Azure Active Directory finden Sie im Abschnitt *Authentifizierung – Azure Active Directory*. Wenn Sie einen anderen Authentifizierungsanbieter anstatt Azure Active Directory verwenden, ist dieser Vorgang in seiner Funktionsweise sehr ähnlich.

Die Autorisierung erfolgt in der Azure Red Hat OpenShift-Policy Engine, die Aktionen wie „create pod" oder „list services" definiert und sie in Rollen in einem Richtliniendokument gruppiert. Rollen sind an Nutzer oder Gruppen durch Nutzer- oder Gruppen-ID gebunden. Wenn ein Nutzer oder Service einen Aktionsversuch startet, prüft die Policy Engine auf eine oder mehrere der Rollen, die dem Nutzer zugewiesen sind (z. B. Kundenadministrator oder Administrator des aktuellen Projekts), bevor der Vorgang fortgesetzt werden kann.

Die Beziehungen zwischen Cluster-Rollen, lokalen Rollen, Bindings von Cluster-Rollen, Bindings von lokalen Rollen, Nutzern, Gruppen und Service Accounts sind in *Abbildung 3.3* dargestellt:

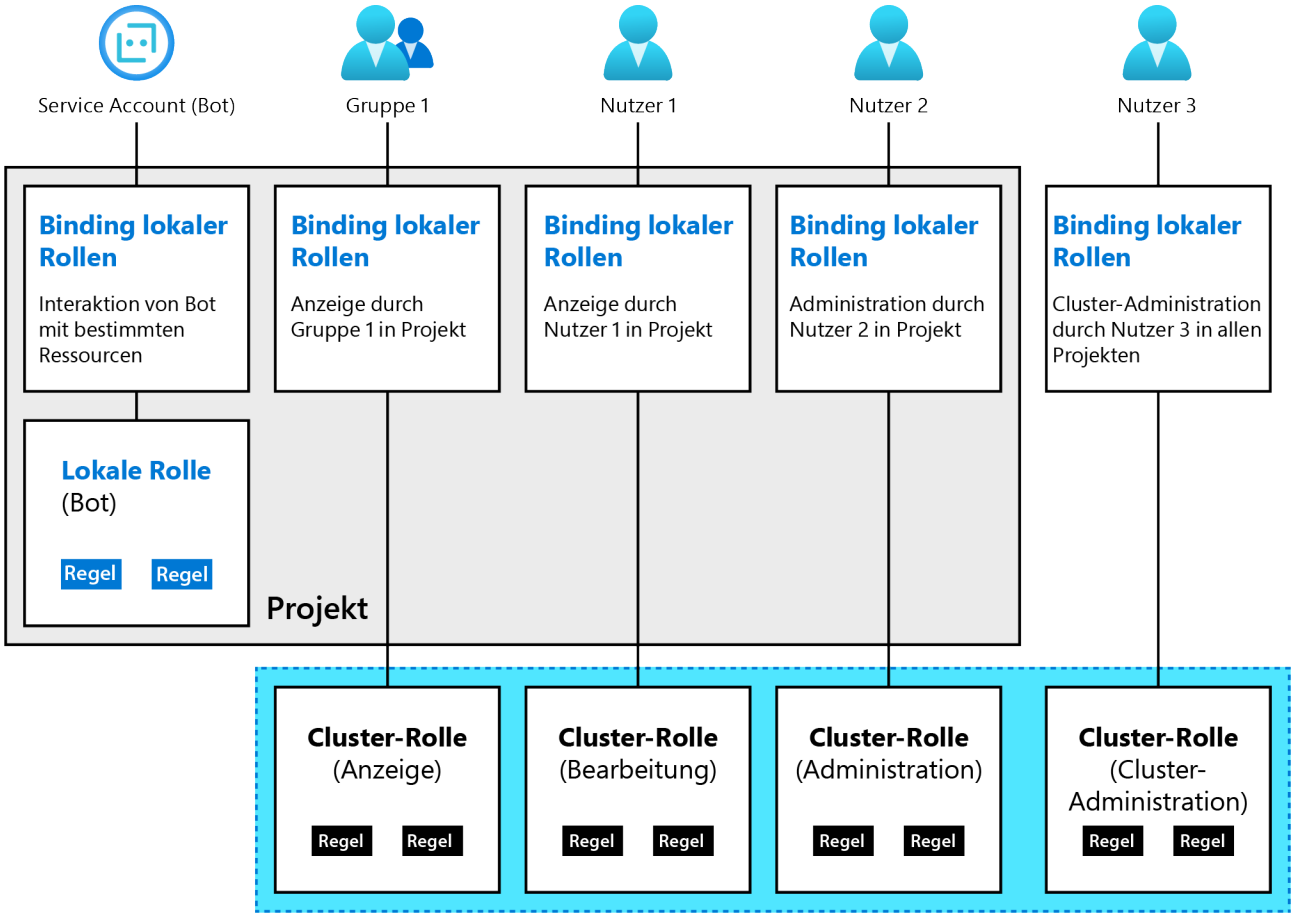


Abbildung 3.3: Beziehungen zwischen Rollen

Die Dokumentation für Red Hat OpenShift hat eine vollständige Liste von Authentifizierungsanbietern.

Support

Azure Red Hat OpenShift ist besonders, was das Support Management angeht. Teams bei Microsoft und Red Hat arbeiten zusammen mit dem globalen Site Reliability Engineering (SRE)-Team, um den Servicebetrieb zu ermöglichen.

Kunden fordern Support im Azure-Portal an. Die Anfragen werden von Ingenieuren bei Microsoft und Red Hat gesichtet und behandelt, um diese schnell abzuarbeiten – ob auf Ebene der Azure-Plattform oder von OpenShift.

Hier finden Sie eine Beschreibung des integrierten Supportvorgangs:

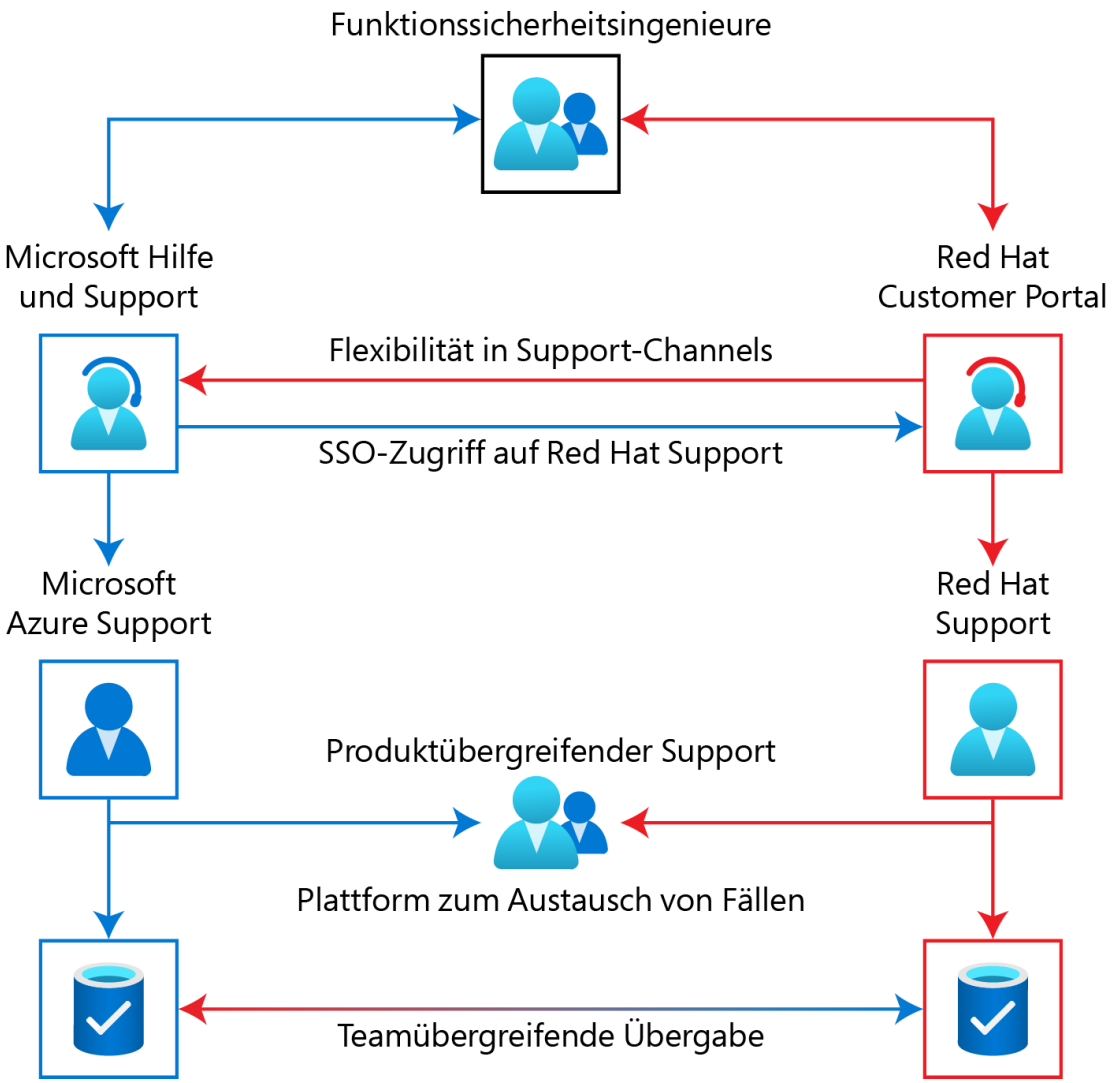


Abbildung 3.4: Der integrierte Supportvorgang

Abbildung 3.4 zeigt, dass Kunden ein Supportticket sowohl im Supportportal von Microsoft als auch von Red Hat erstellen können. Beachten Sie, dass der Cluster beim OpenShift Cluster Manager registriert sein sollte, um Zugriff auf das Supportportal von Red Hat zu erhalten.

Support Engineers von Microsoft können mit erteilter Zustimmung des Kunden über die Plattform zum Austausch von Fällen in jeder Phase nahtlos mit Red Hat zusammenarbeiten. Das gleiche gilt für Support Engineers von Red Hat, wenn eine Zusammenarbeit mit Microsoft angefragt wird. Support Engineers beider Unternehmen haben Zugriff auf das SRE-Team, welches bei Bedarf Behebungsmaßnahmen zur Reparatur von Clustern vornehmen kann.

Preisgestaltung und Subskriptionen

Einer der Hauptvorteile von Azure Red Hat OpenShift im Vergleich zu Do-it-yourself (DIY)-Installationen ist der Umstand, dass sowohl die Computing-, Netzwerk-, Storage- und Azure-Infrastruktur als auch die OpenShift-Subskriptionen nicht einzeln sondern alle über Ihre Azure-Subskription abgerechnet werden. Um eine Vorstellung der Kosten der Lösung zu bekommen, fügen Sie einfach Azure Red Hat OpenShift dem Angebotsersteller im [Azure-Preisrechner](#) hinzu.

Dies ist ein Guide zur Preisgestaltungsseite:

1. Geben Sie *openshift* im Suchfeld ein und fügen Sie es als neue Schätzung hinzu.

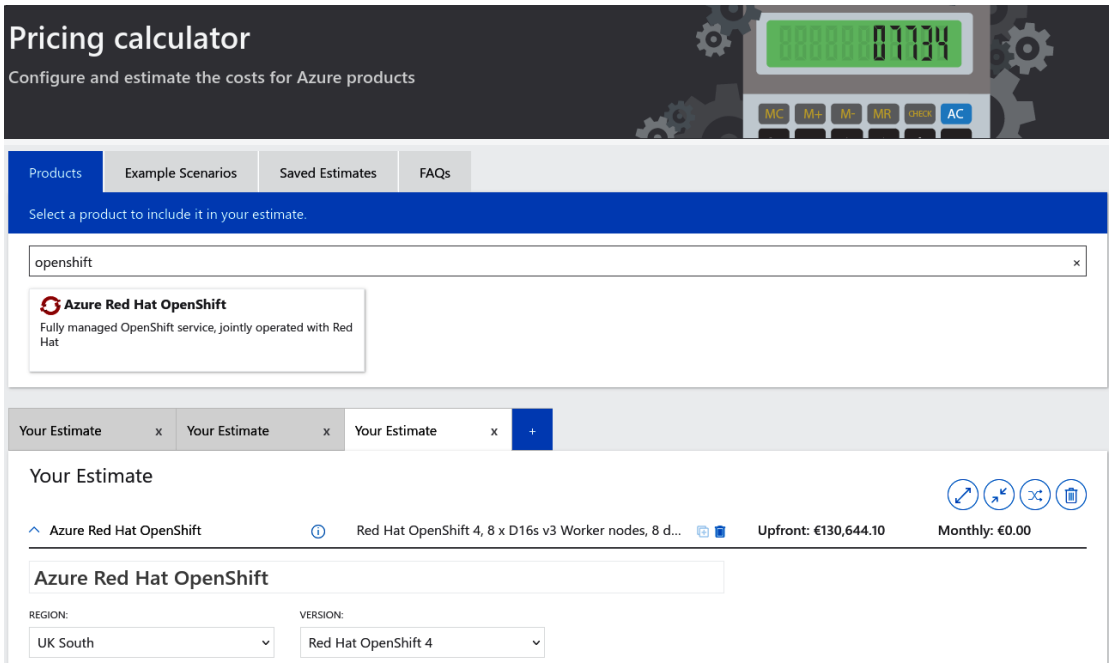


Abbildung 3.5: Der Preisrechnerbereich

2. Im unteren Bereich der Seite finden Sie ein Dropdown-Kombofeld, mit dem Sie die Preiseinheit in Ihre lokale Währung ändern können. In diesem Beispiel wird GBP (£) verwendet.
3. Legen Sie Ihre Azure-Region für das Bereitstellen von Azure Red Hat OpenShift fest. Der Preis variiert je nach Region ein wenig, um die unterschiedlichen Computing-Kosten in den jeweiligen Azure-Regionen zu berücksichtigen.

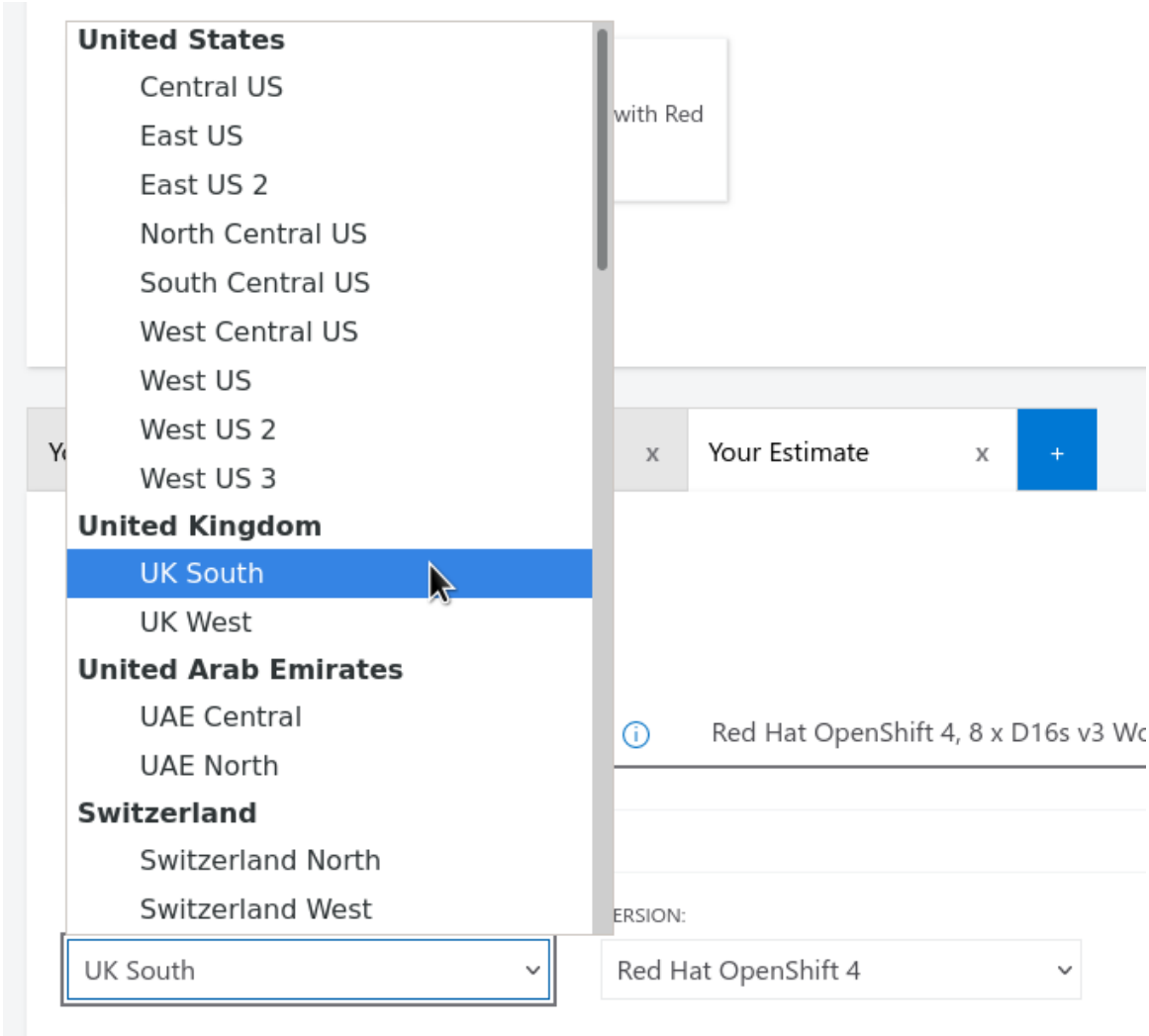


Abbildung 3.6: Auswahl der Azure-Region

4. **Die Worker-Nodes** sind die eigentlichen Ausführungsorte Ihrer Anwendungen. Unter **Einsparungsmöglichkeiten** befinden sich zwei Abschnitte. **Lizenz** bezieht sich auf die Kosten der OpenShift-Subskriptionen und der Abschnitt **Virtueller Computer** auf die Kosten der für die Ausführung der Cluster erforderlichen Computing-Services. Die maximale Anzahl unterstützter Worker-Nodes in einem Cluster ist 100.

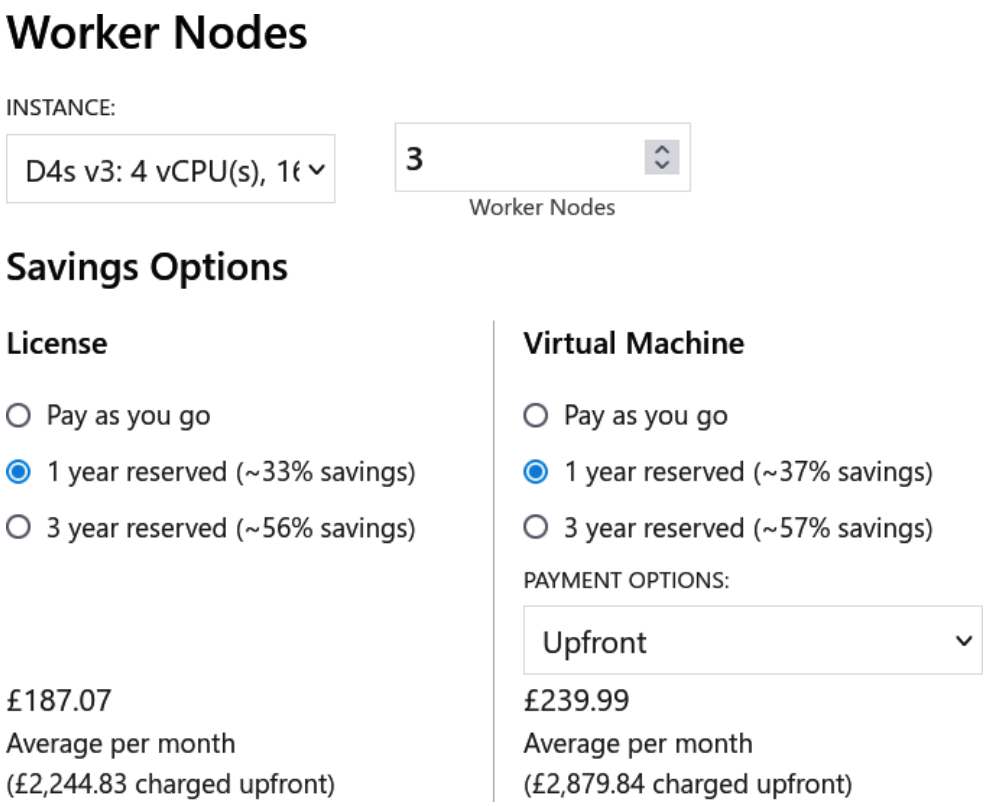


Abbildung 3.7: Details der Worker-Nodes

5. **Verwaltete Betriebssystem-Disks** bezieht sich auf die Größe der Disks, die von Red Hat CoreOS für die Betriebssystempartitionen verwendet wird. Dazu gehört nicht der Storage für persistente Anwendungsvolumes. Diese werden separat provisioniert.

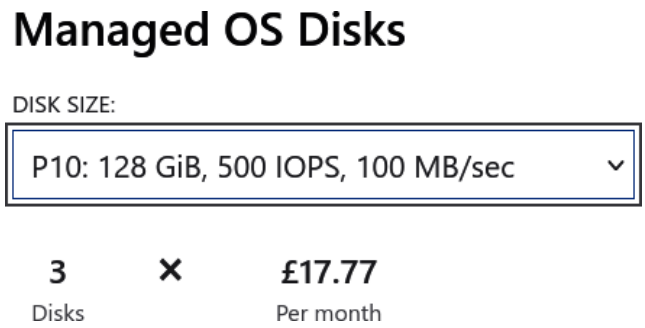


Abbildung 3.8: Verwaltete Betriebssystem-Disks

6. Es gibt einen ähnlichen Abschnitt für die **Master-Nodes**, die heutzutage üblicherweise Control Plane-Nodes genannt werden. Beachten Sie, dass die Control-Nodes im Vergleich zu den Worker-Nodes eine weitaus größere zugewiesene Disk-Größe haben. Dies kommt daher, dass sie einen höheren IOPS oder Durchsatz benötigen, wodurch eine größere Disk-Größe auf Azure erforderlich ist. Die genaue Anzahl der Control Plane-Nodes muss stet 3 sein, um die Cluster-Stabilität zu gewährleisten.

Beachten Sie, dass der Rechner dazu da ist, Preisschätzungen anzuzeigen. Die tatsächlichen Kosten variieren natürlich je nach Nutzung.

Azure Reserved Virtual Machine Instances

Eine **reservierte Instanz** bezieht sich auf die Verpflichtung, diese Ressource für einen bestimmten Zeitraum zu verwenden – ein oder drei Jahre. Es gibt Optionen für reservierte Instanzen für virtuelle Maschinen von Azure Red Hat OpenShift.

Organisationen entscheiden sich normalerweise für reservierte Instanzen, um einen erheblichen Preisnachlass für die IaaS zu erhalten, wenn sie wissen, dass ein Cluster für einen längeren Zeitraum ausgeführt werden wird. Produktivumgebungen werden beispielsweise oft mit reservierten Instanzen ausgeführt. Es ist aber erwähnenswert, dass reservierte Instanzen keinen Einfluss auf das Service-Level oder die Cluster-Architektur haben.

[Weitere Informationen zu Azure Reserved Instances]

Zusammenfassung

Diese Kapitel beschrieb die Einzelheiten des gemanagten Cloud-Services von Azure Red Hat OpenShift. Es wurde eine allgemeine Übersicht der Architektur gegeben, die Integration mit anderen Azure-Services wurde angeführt (weitere Details dazu finden Sie in *Kapitel 9, Integration mit anderen Services*) und Erwägungen zum Management, der Authentifizierung, dem Support und der Preisgestaltung wurden behandelt.

Das nächste Kapitel konzentriert sich auf die Fragen und Entscheidungen, die eine Organisation in der Phase vor der Provisionierung bei der Bereitstellung von Azure Red Hat OpenShift beantworten bzw. treffen muss.

Kapitel 4

Vor der Provisionierung – Fragen zur Unternehmensarchitektur

Viele Organisationen sehen Azure Red Hat OpenShift im Azure-Portal und können, nachdem sie die Dokumentation einmal durchgelesen haben, einen Cluster unter wenig Aufwand mit Red Hat OpenShift bereitstellen. Wenn jedoch etwas mehr Zeit für die Planung von Deployments und für das Stellen einiger Fragen im Voraus aufgewendet wird, ist es möglich, viel Zeit beim zukünftigen Löschen und erneuten Provisionieren von Clustern einzusparen.

Dieses Kapitel basiert auf echter, praktischer Arbeitserfahrung mit vielen Kunden. Es behandelt viele der typischen Fragen zur Zeit vor der Provisionierung, die vorab beantwortet werden sollten. Dieses Kapitel beinhaltet Folgendes:

- Wie viele Cluster sind benötigt, einschließlich Staging, Produktion und Ähnliches
- Öffentliche und private Netzwerksichtbarkeit
- Hybride Konnektivität, wie etwa das Verbinden mit On-Premises-Lösungen

Wir fangen damit an, herauszufinden, wie viele Cluster benötigt werden.

Wie viele Cluster werden benötigt?

Es gibt viele Deployment-Muster für OpenShift, jedoch lautet eine häufig gestellte Frage „Wie viele Cluster benötigt meine Organisation?“. Dies ist natürlich eine Entscheidung, die von der Organisation abhängt, die folgenden Abschnitte geben Ihnen jedoch einige Informationen als Entscheidungshilfe.

Lifecycle-Phasen: Entwicklung, Tests, Produktion

Die meisten Organisationen stellen unabhängig ihrer Größe ihre IT-Systeme mit Lifecycle-Phasen bereit. Dieser Ansatz wird auch Staging-Muster genannt. Die drei am häufigsten verwendeten Phasen sind Entwicklung, Tests und Produktion. Mit mehreren Phasen können Anwendungen und Deployments von Anwendungen in einer sicheren Umgebung getestet werden, bevor diese Änderungen eine Produktivumgebung erreichen. Das am häufigsten verwendete und empfohlene Staging-Muster nutzt mindestens drei separate Azure Red Hat OpenShift-Cluster:

- **Entwicklung:** In diesem Cluster können Entwickler und Operatoren alles testen. Dies kann ein großer „Sandbox"-Cluster sein, jedoch ist es üblicher und oft auch sicherer, kleine kurzlebige Cluster zu haben, in denen Tests häufig erstellt und gelöscht werden können.
- **Tests:** In diesem Cluster werden anstehende Cluster-Änderungen wie Patches oder Konfigurationsänderungen getestet und validiert, bevor sie für die Produktion freigegeben werden. Dieser Vorgang wird in manchen Organisationen auch „Vorproduktion" genannt, jedoch kann die Vorproduktion auch in einer völlig anderen Umgebung stattfinden.
- **Produktion:** In diesem Cluster werden Ihre richtigen Anwendungen ausgeführt.

Einige Organisationen haben zusätzlich zu den vorangehenden Clustern weitere Umgebungen wie Umgebungen für Integrationstests, jedoch können nur Sie wissen, wie viele Staging-Umgebungen Ihre Organisation am ehesten benötigt. Betrachten Sie andere ähnliche Unternehmensanwendungen, um gängige Deployment-Muster zu sehen, falls Sie sich nicht sicher sind.

In Situationen, in denen Azure Red Hat OpenShift für nicht kritische Anwendungen verwendet wird, könnte es für Ihre Organisation Sinn ergeben, nur zwei Cluster (Zusammenführung von Entwicklung und Tests) oder sogar einen einzigen Cluster zu verwenden, der die Entwicklung, Tests und Produktion beinhaltet. Dies hat den Vorteil, dass die Cloud-Kosten gering gehalten und die Gesamtanzahl der Cluster, die gemanagt werden müssen, reduziert werden. Wenn nur ein einzelner Cluster betrieben wird, entscheiden sich Administratoren möglicherweise dafür, separate Namespaces für Entwicklung, Tests und Produktion zu verwenden. Jedoch gibt es Nachteile, wenn nur ein Cluster betrieben wird:

- Änderungen, die den gesamten Cluster betreffen (wie Software-Patches) können Probleme in der Produktion verursachen, die andernfalls in einer Testumgebung erkannt und vorgebeugt hätten werden können.
- Wenn sich eine Anwendung in einer Test- oder Entwicklungsumgebung auf eine unerwartete Art und Weise verhält, wie etwa das Erzeugen von vielen Containern oder das Vereinnahmen des gesamten verfügbaren Disk-Speichers, führt dies zu Problemen in der Produktionsumgebung.

Dieses Buch kann Ihnen keine genaue Anzahl an Clustern nennen, die sich perfekt für Ihre Situation eignet. Wie vorher erwähnt, kann es hilfreich sein, andere Unternehmensanwendungen in Ihrer Organisation zu betrachten, um zu sehen, wie viele Lifecycle-Phasen verwendet werden.

Pläne für Business Continuity, Disaster Recovery und Failover

Zusätzlich zu den standardmäßigen Staging-Umgebungen, möchten viele Organisationen außerdem eine Failover-Produktivumgebung erstellen. Eine Failover-Produktivumgebung wird verwendet, wenn ein katastrophaler Fehler einen Ausfall des gesamten Clusters oder der Azure-Region zur Folge hat. Diese Maßnahme wird oft **Disaster Recovery (DR)** genannt. Üblicherweise wird der Cluster in einer anderen Azure-Region als die des Produktions-Clusters bereitgestellt.

Der gemanagte Cloud-Service beinhaltet ein **Service-Level Agreement (SLA)** von 99,95 %. Dies ist für viele geschäftskritische Anwendungen akzeptabel. Für einige Anwendungen wird jedoch möglicherweise ein besseres SLA benötigt. Es ist wichtig zu erwähnen, dass mit einem einzigen Cluster ein höheres SLA nicht möglich ist. Überlegen Sie, ob Ihre Anwendung ein höheres Service Level als 99,95 % benötigt, oder ob dies für Ihre Anwendung ausreicht.

Falls nicht, können Sie eine bessere Service-Level-Verfügbarkeit (wie etwa 99,999 %) erzielen, indem Sie mehrere Cluster parallel ausführen und ein kombiniertes SLA berechnen. Die Cluster können sich alle in derselben Region (z. B. `westeurope`) oder in mehreren (z. B. `westeurope` und `northeurope`) befinden, um ein noch höheres Verfügbarkeitslevel zu erreichen. Die folgende Azure-Dokumentation beschreibt die Berechnung von kombinierten SLAs bei der Ausführung von mehreren Clustern.

[Azure-Dokumentation zu kombinierten SLAs]

Multicluster- und Multiregions-Deployments von Azure Red Hat OpenShift sind nicht im Umfang dieses Buchs enthalten. Dies liegt daran, dass es bei der Entwicklung dieser komplexeren Architekturen einige Herausforderungen damit gibt, den Datenverkehr in den Cluster zu leiten, und es bei der Entscheidung, wie und welche Daten zwischen Clustern geteilt werden sollten, einiges zu berücksichtigen gibt.

Regionen und Verfügbarkeitszonen

Azure Red Hat OpenShift wurde dafür entwickelt, drei Verfügbarkeitszonen pro Region, in der die Lösung bereitgestellt wird, zu verwenden. Eine [Verfügbarkeitszone] ist in Azure ein autonomes Rechenzentrum mit eigener Stromversorgung, Kühlung und Netzwerkkonnektivität innerhalb einer Region. Wenn Sie ein Deployment von Azure Red Hat OpenShift betrachten, sehen Sie jeweils einen einzelnen Control-Node (virtuelle Maschine) und einen Anwendungs-Node pro Verfügbarkeitszone.

Abbildung 4.1 zeigt die Verteilung von Control-Nodes und Anwendungs-Nodes (Worker) in allen drei Verfügbarkeitszonen innerhalb einer einzelnen Azure-Region:

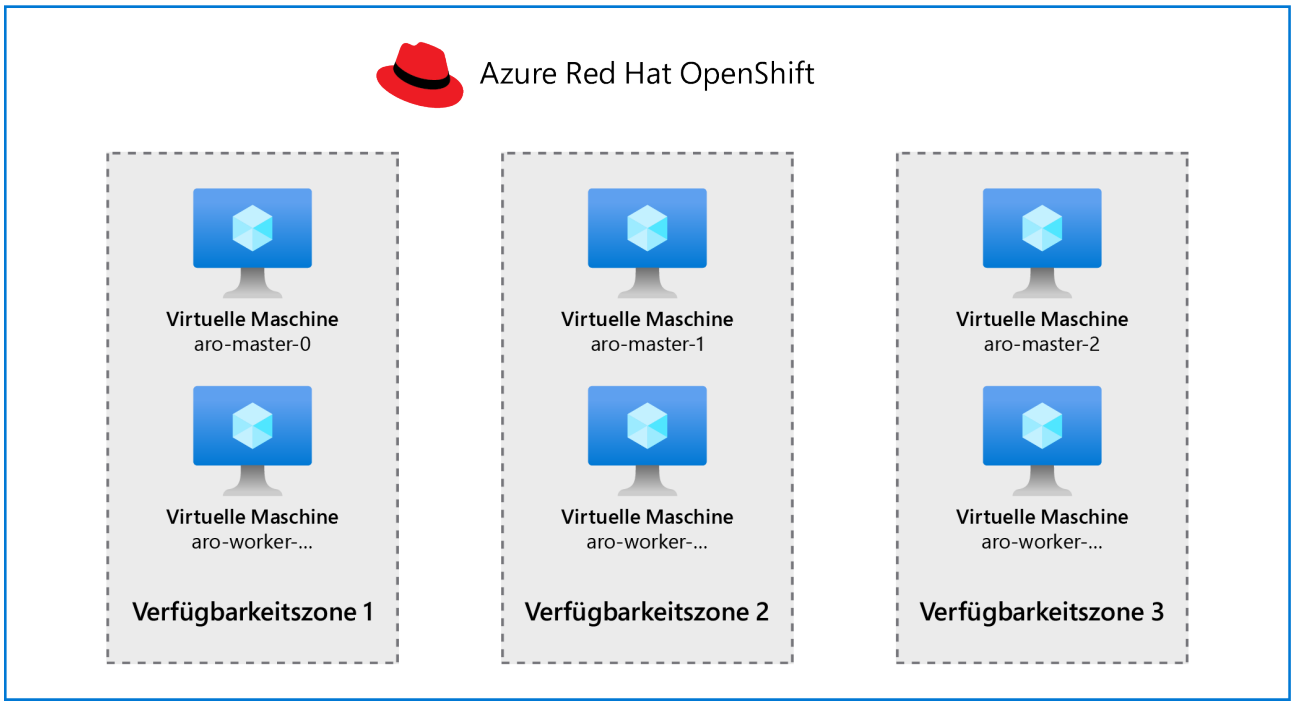


Abbildung 4.1: Verteilung von Control- und Anwendungs-Nodes in allen drei Verfügbarkeitszonen innerhalb einer einzelnen Azure-Region

Azure Red Hat OpenShift wurde dafür entwickelt, um als vollständig gemanagter, hochverfügbarer Service angeboten zu werden. Daher ist es nicht möglich, weniger als drei Control-Nodes und drei Worker-Nodes bereitzustellen.

Netzwerkkonzepte

In der Einführung zu Azure Red Hat OpenShift wurde zuvor eine hervorragende Dokumentationsseite zu Networking-Konzepten erwähnt, die auf der Microsoft Dokumentationswebsite zu finden ist:

- Netzwerkkonzepte für Azure Red Hat OpenShift

Diese Seite enthält eine detaillierte Beschreibung von allen Networking-Komponenten in Azure Red Hat OpenShift – die Load Balancer, die öffentlichen und privaten IP-Adressen, die Netzwerk-Sicherheitsgruppen und mehr.

Nachfolgend sind einige wichtige Punkte für das Verständnis:

- Azure Red Hat OpenShift wird in einem bestehenden oder neuen virtuellen Netzwerk bereitgestellt. Es wird außerdem nur ein einzelnes virtuelles Netzwerk unterstützt – jedoch würden mehrere Netzwerke keine zusätzlichen Vorteile bringen, da OpenShift sein eigenes **Software-Defined Network (SDN)** namens OVS als oberste Schicht verwendet.
- Die Mindestgröße der Master- und Anwendungs-Node-Subnetze ist `/27`.
- Die standardmäßige Pod-CIDR ist `10.128.0.0/14`.
- Die standardmäßige Service-CIDR ist `172.30.0.0/16`.
- Jedem Node wird für seine Pods ein `/23`-Subnetz (512 IP-Adressen) zugeordnet. Dieser Wert kann nicht geändert werden.
- Egress-IPs werden derzeit nicht unterstützt.
- Es ist möglich, das Routing des Egress-Datenverkehrs zu steuern. Genauer gesagt, um ihn über Azure Firewall zu senden. Zum Zeitpunkt der Veröffentlichung ist diese Funktion in der öffentlichen Vorschau und ist hier dokumentiert: [Steuern des Egress-Datenverkehrs]().

Öffentliche oder private Netzwerksichtbarkeit

Sie hören möglicherweise öfters, dass Azure Red Hat OpenShift sowohl als öffentliches als auch privates Deployment bereitgestellt werden kann. Dies stimmt zwar, jedoch ist es wichtig zu wissen, worin der Unterschied besteht, wenn die Control Plane öffentlich/privat gemacht wird und die Anwendungen auf Ihrem Cluster öffentlich/privat gemacht werden.

In Bezug auf die Sichtbarkeit des API-Servers ist dies eine Entscheidung, die Sie zum Zeitpunkt der Provisionierung treffen müssen. Es ist nicht möglich, die öffentliche/private Sichtbarkeit zu ändern, nachdem der Cluster provisioniert wurde.

Wenn Sie später in diesem Kapitel mit dem Erstellen eines Azure Red Hat OpenShift-Clusters beginnen, werden Sie den Befehl `az aro create` ausführen, welcher Argumente zur Sichtbarkeit des Clusters unterstützt. Das folgende Beispiel zeigt, wie Sie die Sichtbarkeit einstellen können:

Beides privat

```
az aro create .... --apiserver-visibility Private --ingress-visibility Private
```

Privater API-Server und öffentlicher Worker-Ingress

```
az aro create .... --apiserver-visibility Private --ingress-visibility Public
```

Die Sichtbarkeit von `apiserver` und des Anwendungs-Ingress werden im Azure-Architekturdiagramm in *Abbildung 4.2* dargestellt. Dieses Diagramm zeigt auf, wie Azure Red Hat OpenShift interne und öffentliche Azure-Load-Balancer an den Orten verwendet, an denen die API und der Router abhängig von den ausgewählten Sichtbarkeitsoptionen bereitgestellt werden.

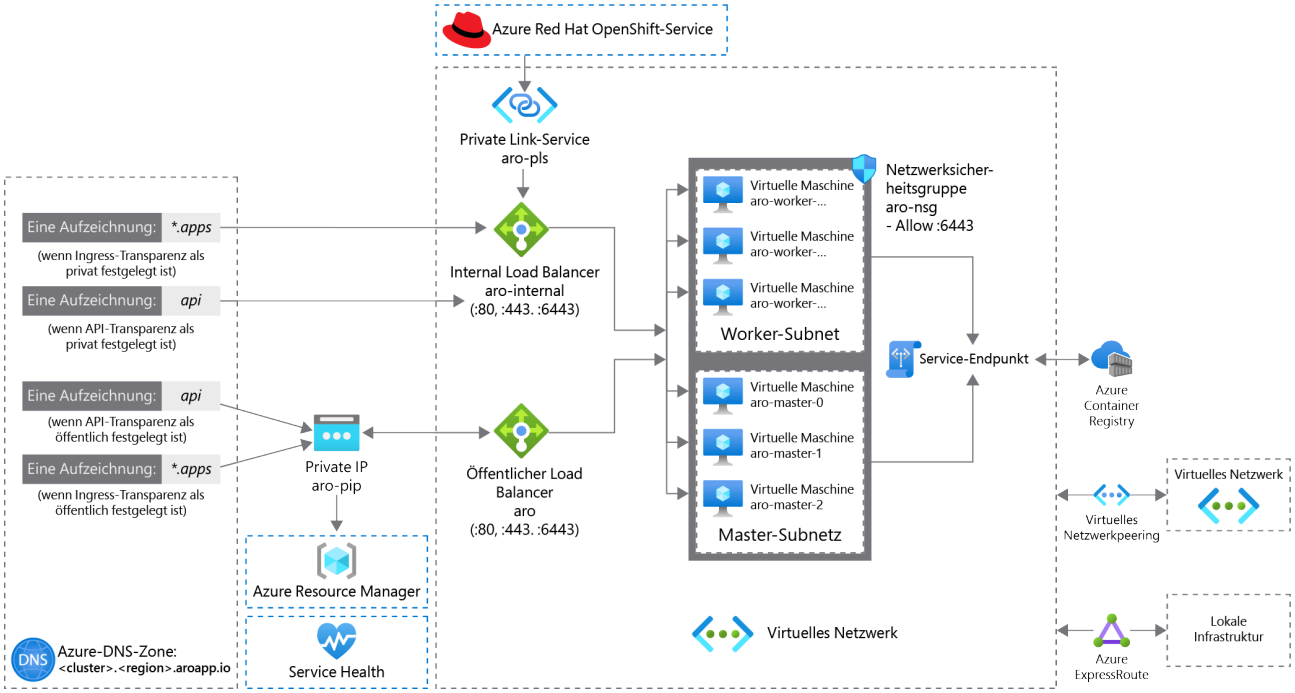


Abbildung 4.2: Azure Red Hat OpenShift verwendet interne und öffentliche Azure-Load-Balancer

Detailliertere Beschreibungen zur Sichtbarkeit des API-Servers und des Ingress sind nachfolgend aufgeführt.

Sichtbarkeit des API-Servers (Control Plane)

`--apiserver-visibility` kann **public** oder **private** sein:

- **Private** bedeutet, dass die Control Plane von Red Hat OpenShift (ursprünglich „Master-Nodes" genannt), in der die Kubernetes API ausgeführt wird, nicht über das öffentliche Internet zugänglich ist. Diese Control Plane ist für Entwickler und Operatoren bestimmt, um den Cluster zu kontrollieren und Anwendungen oder den Cluster selbst bereitzustellen, zu löschen und zu skalieren. Unternehmen die mit Express-Route-Verbindungen für Azure oder mit einem **Virtual Private Network (VPN)** auf Computing-Ressourcen zugreifen, sollten sich in den meisten Fällen für die private Option entscheiden.
- **Public** bedeutet, dass die Control Plane des Clusters über das öffentliche Internet zugänglich ist. Dies setzt den Cluster dem Risiko aus, aus dem Internet angegriffen zu werden – selbst wenn der Zugriff nur mit Authentifizierung erfolgt. Die Option `public` ist aber für Lab- und Testumgebungen nützlich, in denen Sie nicht das Quellnetzwerk Ihrer Nutzer kontrollieren können. In Umgebungen, in denen echte Daten gespeichert werden, oder in Produktivumgebungen wird es dringend empfohlen, die Sichtbarkeit des API-Servers auf `private` zu setzen.

Die nachfolgende Liste führt einige Beispiele auf, in denen die API-Serverkonnektivität erforderlich ist. Diese Anwendungsfälle sollten bei der Entscheidung für die private oder öffentliche Option berücksichtigt werden:

- Entwickler, die Skripts und Tools aus ihrer IDE verwenden (z. B. `kubectl rollout`)
- Operatoren, die den Zustand ihrer Cluster überprüfen (z. B. `kubectl get nodes`)
- CI/CD-Server, die den Zustand von Deployments überprüfen oder ändern müssen (z. B. Azure DevOps)
- Cluster-Sicherheitstools, die sich mit dem Cluster verbinden, um den Zustand zu überprüfen
- Überwachungstools, die von der Kubernetes API abhängig sind

In den meisten Fällen können **private** Verbindungen für das Netzwerk konfiguriert werden, jedoch kann die oben stehende Liste durch zusätzliche Beispiele aus Ihrer Organisation ergänzt werden und Sie finden möglicherweise einen unerwarteten Anwendungsfall, für den eine **öffentliche** Sichtbarkeit des API-Servers notwendig ist.

Ingress-Sichtbarkeit (Anwendungen)

`--ingress-visibility` kann sowohl „public" als auch „private" sein:

- **Private** bedeutet, dass zugängliche Services (im Zusammenhang mit Anwendungen auf dem Cluster) keine direkten Verbindungen aus dem öffentlichen Internet zulassen. Es ist weiterhin möglich, das Netzwerk-Routing so zu konfigurieren, dass Nutzeranfragen durch Azure Firewall oder eine Firewall einer Webanwendung gehen, die optional in einem separaten Azure-Netzwerk ausgeführt werden, bevor die Nutzer sich mit Ihren Anwendungen verbinden. **Private** ist auch dafür geeignet, wenn Anwendungen auf dem Cluster nur intern in Ihrer Organisation verwendet werden sollen, wie etwa die Verarbeitung von Gehaltsabrechnungen, Datenanalysen oder andere interne Webanwendungen.
- **Public** bedeutet, dass die Anwendungen auf Ihrem Cluster über das öffentliche Internet zugänglich sind. Es ist jedoch trotzdem notwendig, eine Ingress- oder Routing-Ressource zu konfigurieren, um auf diese Anwendungen zugreifen zu können. Dies wäre der Fall, wenn Sie eine öffentliche E-Commerce-Shoppingwebsite oder eine andere öffentliche Anwendung auf Red Hat OpenShift hosten.

Ingress wird manchmal als OpenShift-„Router" bezeichnet.

Empfehlungen für die Produktion

Es wird Folgendes empfohlen:

- Greifen Sie über eine private Verbindung auf den Cluster zu – nicht über das öffentliche Internet. Azure ExpressRoute ist die beste Option, wenn der Cluster permanente Konnektivität mit einem On-Premises-Netzwerk oder einem Unternehmensbüro benötigt. Andernfalls kann auch eine VPN zu Azure eine private Verbindung ermöglichen. Weitere Details hierzu finden Sie im Abschnitt **Hybride Konnektivität**.
- Legen Sie die Sichtbarkeit des API-Servers (Control Plane) auf `private` fest und schränken Sie optional den Zugriff weiter mithilfe von Firewalls ein.
- Legen Sie die Ingress-Sichtbarkeit (Anwendungen) für Anwendungen, die innerhalb Ihrer Organisation ausgeführt werden, auf `private` fest. Wenn es für Anwendungen auf Azure Red Hat OpenShift einen Use Case gibt, für den die Anwendung im öffentlichen Internet gehostet werden muss, richten Sie einen separaten Azure Red Hat OpenShift-Cluster ein – mit mindestens einem Cluster für interne Apps und einem weiteren für externe Apps mit einer öffentlichen Ingress-Sichtbarkeit. Es ist aber auch möglich, `öffentlichen` und `privaten` Ingress im selben Cluster zu mischen.

Die meisten Organisationen lassen sich von ihren Azure-Networking- und Sicherheitsteams beraten, um zusätzliche Kontrollen zu ermitteln, die möglicherweise erforderlich sind. Für manche Organisationen ist es beispielsweise notwendig, eine Webanwendungs-Firewall vor eine Webanwendung zu schalten. Dies ist auch ein gängiges Deployment-Muster bei der Bereitstellung von Anwendungen auf Azure Red Hat OpenShift.

Hybride Konnektivität

Die meisten Organisationen, die Anwendungen auf Azure Red Hat OpenShift bereitstellen, führen Anwendungen aus, die eine Verbindung mit unterstützenden Services benötigen, die in On-Premises-Umgebungen ausgeführt werden. Die Verbindung von Azure mit einer On-Premises-Umgebung wird häufig hybride Konnektivität oder Hybrid Cloud-Infrastruktur genannt. Es gibt einige Methoden für die Einrichtung von Verbindungen mit On-Premises-Umgebungen. Dies sind die zwei nennenswertesten Optionen:

- **VPN**: geeignet für Konnektivität zu Azure mit geringer Komplexität VPNs werden normalerweise über Azure VPN Gateway verbunden. Weitere Informationen finden Sie unter Was ist VPN Gateway?
- **Azure ExpressRoute-Verbindungen**: geeignet für eine dedizierte, robuste, permanente Verbindung mit Azure. Weitere Informationen finden Sie unter Was ist Azure ExpressRoute?

Beide Lösungen können es unabhängig von der verwendeten Verbindungsmethode Anwendungen ermöglichen, von On-Premises-Umgebungen sich mit Anwendungen auf Azure Red Hat OpenShift zu verbinden und umgekehrt.

Die Verbindung von einer On-Premises-Umgebung mit Azure Red Hat OpenShift geschieht häufig über ein virtuelles Hub-Netzwerk. In *Abbildung 4.3* wird ein Diagramm gezeigt, das die Funktionsweise der Konnektivität mir Azure ExpressRoute erklärt:

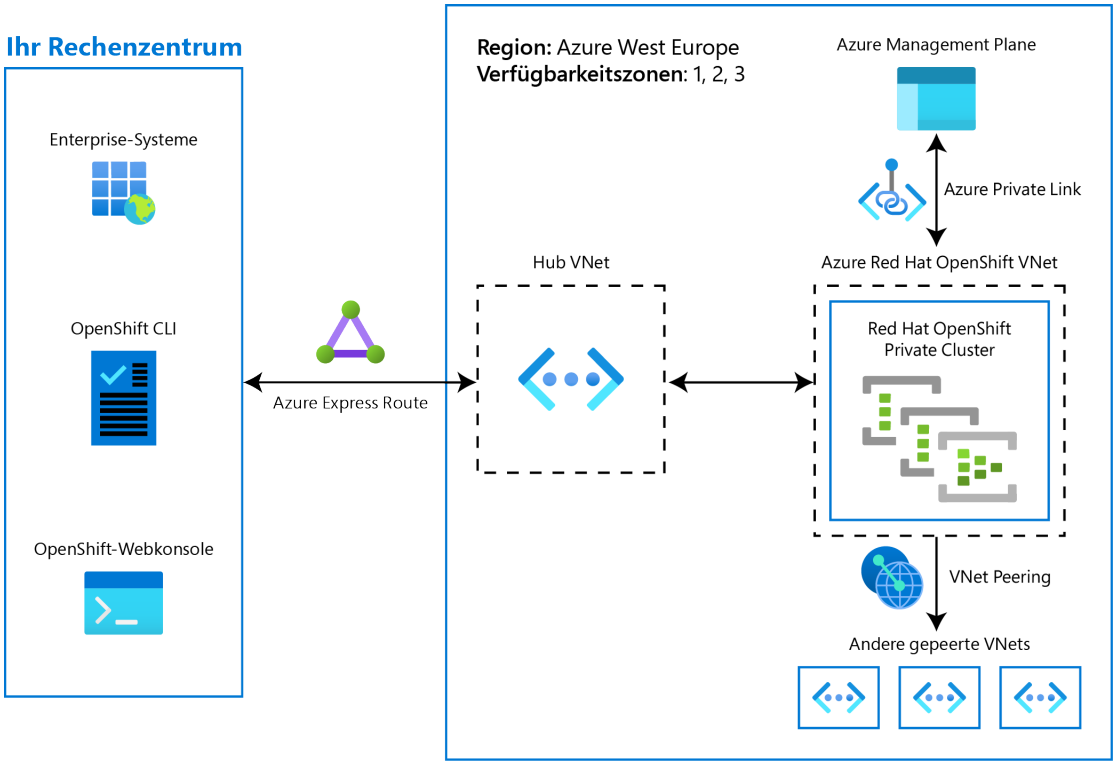


Abbildung 4.3: Konnektivität mit Azure ExpressRoute

Das oben stehende Diagramm bildet eine allgemeine Architektur der Verbindungsmethode von Azure ExpressRoute mit Azure Red Hat OpenShift ab. Dieses Diagramm zeigt zwar Azure ExpressRoute an, die Konnektivität über VPN ist aber hinsichtlich des Konzepts sehr ähnlich.

Überlegungen zu Anwendungen, die in einer hybriden Architektur ausgeführt werden

Wenn sich Anwendungen auf Azure Red Hat OpenShift mit On-Premises-Umgebungen verbinden, gibt es einige Faktoren, die Anwendungseigentümern möglicherweise auffallen:

- **Verbindungslatenz:** Azure ExpressRoute bietet eine Verbindung zu On-Premises-Umgebungen mit geringer Latenz an, während VPNs, deren Verbindungen über das öffentliche Internet laufen, eine wesentlich höhere Latenz aufweisen können. Dies bereitet manchen Anwendungen wie Webservern, bei denen die Verbindung lediglich HTTP-Datenverkehr überträgt, wahrscheinlich keine Probleme. Wenn sich jedoch eine serverseitige Anwendung auf diesem Webserver mit einer Datenbank in einer On-Premises-Umgebung verbinden muss, kann dies zu erheblichen Performance-Einbußen führen.
- **Kosten für Ingress-/Egress-Datenverkehr:** Bei einigen Verbindungstypen entstehen Kosten für den Ingress- und Egress-Datenverkehr – entweder bei Azure oder beim Verbindungsanbieter. Es wird empfohlen, vorsichtshalber die Kosten in einer Testumgebung zu messen und davon abzuleiten, welche Kosten bei einer Spitzenbelastung entstehen, um sicherzugehen, dass keine Überraschungen auf Sie zukommen.
- **Fehlerszenario:** Azure ExpressRoute-Verbindungen sind permanent und robust, während VPN-Verbindungen oft unterbrochen oder verbunden/getrennt werden. Da VPN-Verbindungen über das öffentliche Internet laufen, kann die Verbindungslatenz und -qualität des Services im Tagesverlauf häufig variieren. Es ist wichtig, die Anwendungsperformance sowohl unter schlechten hybriden Verbindungsbedingungen als auch bei optimaler Verbindungsperformance zu testen. Ist die Anwendung auf Azure Red Hat OpenShift in der Lage, weiterhin – auch wenn beeinträchtigt – ausgeführt zu werden, falls die Verbindung für mehrere Stunden unterbrochen sein sollte, oder ist ihre Verfügbarkeit komplett von der hybriden Verbindung abhängig?

Ihre Organisation wird höchstwahrscheinlich noch einige zusätzliche Punkte in Ihrer eigenen internen Checkliste durchgehen müssen. Die oben stehenden Punkte sollten aber ausreichen, um sich ein Bild davon zu machen, wie die Funktionsweise Ihrer hybriden Architektur aussehen könnte.

Zusammenfassung

In diesem Kapitel haben wir viele der gängigen Fragen zur Zeit vor der Provisionierung behandelt, die Sie sich selber stellen und recherchieren sollten, bevor Sie Azure Red Hat OpenShift bereitstellen. Das nächste Kapitel enthält einige nützlichen Tipps für Ressourcen im Zusammenhang mit dem tatsächlichen Bereitstellen des Clusters.

Kapitel 5

Provisionierung eines Azure Red Hat OpenShift-Clusters

Lassen Sie uns etwas zurückblicken, um zu sehen, was wir bis jetzt behandelt haben. Sie haben in diesem Buch bereits Folgendes gelernt:

- In *Kapitel 2*, *Einführung in Red Hat OpenShift*, haben wir Ihnen eine kurze Einführung in Red Hat OpenShift gegeben und die Vorteile erläutert, die OpenShift im Vergleich zu Kubernetes hat.
- In *Kapitel 3*, *Azure Red Hat OpenShift*, haben wir für Sie die Einzelheiten des gemanagten Cloud-Services von Azure Red Hat OpenShift beschrieben und wichtige Konzepte wie Überlegungen zur Architektur, zum Management, zur Authentifizierung, zum Support und zur Preisgestaltung behandelt.
- In *Kapitel 4*, *Vor der Provisionierung – Fragen zur Unternehmensarchitektur*, haben wir die gängigen Fragen behandelt, die sich Organisationen stellen sollten, bevor sie Azure Red Hat OpenShift bereitstellen.

Nun sind wir bereit, einen Cluster zu provisionieren und bereitzustellen. Sie können die offizielle Deployment-Anleitung auf der Dokumentationsseite einsehen (Provisionieren ist ein Teil des Deployments. Die Provisionierung eines Deployments bedeutet, dass sichergestellt wird, dass die erforderliche IT-Infrastruktur zur Unterstützung des Deployments vorhanden ist).

[Tutorial – Erstellen eines Azure Red Hat OpenShift 4 Clusters]

Dieses Kapitel behandelt nicht individuellen Befehle, die Sie eingeben müssen, um einen Cluster zu erstellen, da die sich die Befehle im Laufe der Zeit ändern können und die Dokumentation sie bereits detailliert behandelt.

Dieses Kapitel wird jedoch Anweisungen zum allgemeinen Prozess geben und aufführen, was Sie beim Bereitstellen eines Clusters berücksichtigen sollten.

Manuelle Deployments – Erwartungen an das Timing

Einige Organisationen müssen wissen, wie lange die Provisionierung eines Clusters dauert. Lassen Sie uns dies mit einer detaillierten Beschreibung erörtern.

Dies ist ein allgemeiner Überblick über die Deployment-Voraussetzungen:

- Eine az-Befehlszeile, die auf einer Workstation eines Systemadministrators bereitgestellt wird
- Die zur Verwendung mit Ihrer Azure-Subskription registrierten Ressourcenanbieter
- Ein „Pull"-Secret von Red Hat (optional)
- Eine Domain für Ihren Cluster (optional)
- Ein virtuelles Netzwerk mit zwei leeren Subnetzen – eins für die Control Plane-Nodes und eins für die Anwendungs-Nodes

Um diese Voraussetzungen zu erfüllen, bräuchte ein erfahrener Administrator für Azure und Red Hat OpenShift wahrscheinlich 30 Minuten für die Ersteinrichtung und 10 Minuten, wenn diese Schritte als Teil eines manuellen Vorgangs wiederholt ausgeführt werden.

Sobald diese Voraussetzungen erfüllt sind, benötigt der automatisierte Provisionierungsvorgang abhängig von der Auslastung in der Azure-Region üblicherweise zwischen 25 und 40 Minuten.

Deployment-Automatisierung

Da der Provisionierungsvorgang von Azure Red Hat OpenShift automatisiert ist, versuchen Organisationen häufig auch die Voraussetzungsschritte (Erstellen von Netzwerken, Subnetzen, Service-Prinzipien usw.) mit Tools zu automatisieren.

Es gibt viele Tools, die diese Schritte automatisieren. Dies sind einige der empfohlenen Tools:

- Das az-Befehlszeilen-Tool: Wenn automatisiert, wird dieses Tool normalerweise in einem Container oder als Teil eines CI/CD-Prozesses installiert. Hierbei werden meistens Tools wie Jenkins, Azure DevOps oder möglicherweise Ansible verwendet. Beachten sie, dass das az-Befehlszeilen-Tool nur einmal bereitgestellt werden muss, jedoch müssen für weitere Cluster die Azure-Subskriptions-ID angegeben werden, um unterschiedliche Teile der Organisation zu repräsentieren.
- Die für die Verwendung mit Ihrer Azure-Subskription registrierten Ressourcenanbieter: Wie bereits erwähnt, ist dies Teil der Einrichtung des az-Befehlszeilen-Tools.
- Ein „Pull"-Secret von Red Hat (optional): Red Hat hat eine dokumentierte, unterstützte REST API für das Abrufen eines „Pull"-Secrets. Weitere Informationen hierzu finden Sie in diesem **Artikel**.
- Eine Domain für Ihren Cluster (optional): Dies hängt davon ab, wie Sie DNS-Datensätze erstellen, wenn Sie aber Azure DNS verwenden, dann können Terraform, Ansible oder andere bekannte Azure-Automatisierungstools dies für Sie übernehmen.
- Ein virtuelles Netzwerk mit zwei leeren Subnetzen – eines für die Control Plane-Nodes (Master) und eines für die Anwendungs-Nodes (Worker): Dies wird üblicherweise durch bekannte Azure-Automatisierungstools automatisiert. Terraform, Ansible oder ähnliche Tools können dieses Netzwerk und die Subnetze für Sie erstellen.

Wenn Sie die Voraussetzungsschritte automatisieren, können Sie die Einrichtungszeit von 30 oder 10 Minuten (bei einem manuellen Prozess) auf nur eine oder zwei Minuten reduzieren. Das Deployment des Clusters kann nicht beschleunigt werden (dies dauert immer zwischen 25 und 40 Minuten), jedoch kann es im Vergleich zu einer On-Premises-Umgebung ein ziemlich schneller Deployment-Prozess sein.

Die Deployment-Automatisierung hat viele Vorteile, die über die Prozessbeschleunigung hinausgehen. Kunden, die Ihr Deployment automatisieren, erhalten die Vorteile eines robusten, wiederholbaren Prozesses, der einfach protokolliert und geprüft werden kann. Es kommt außerdem sehr oft vor, dass Kunden Self-Service-Katalogelemente in ihre eigenen Portale integrieren, sodass es Teams ermöglicht wird, Azure Red Hat-OpenShift-Cluster ohne weitere Interaktionen mit dem Cloudplattform-Team zu provisionieren und deprovisionieren.

Zugreifen auf den Cluster

Dieser Abschnitt enthält eine einfache Referenz für den Zugriff auf Ihr Azure Red Hat OpenShift-Cluster nach der Provisionierung.

Über die Web UI

Rufen Sie in einer Bash-Shell, falls Sie die CLI installiert haben, oder in einer Azure Cloud Shell (Bash)-Sitzung in Ihrem Azure-Portal die Anmelde-URL Ihres Clusters ab, indem Sie den folgenden Befehl ausführen:

```
az aro show -n $CLUSTER_NAME -g $RG_NAME --query "consoleProfile.url" -o tsv
```

Die Ausgabe sollte der folgenden ähneln: `openshift.xxxxxxxxxxxxxxxxxxxx.eastus.aroapp.io`. Die Anmeldungs-URL für Ihr Cluster ist `https://` und anschließend der Wert von `consoleProfile.url`, wie beispielsweise `https://openshift.xxxxxxxxxxxxxxxxxxxx.eastus.aroapp.io`.

Öffnen Sie diese URL in Ihrem Browser. Sie werden aufgefordert, sich über den Nutzer `kubeadmin` anzumelden. Verwenden Sie den Benutzernamen und das Passwort, die das Installationsprogramm Ihnen während dem Installationsvorgang bereitgestellt hat.

Nach der Anmeldung sollten Sie die Azure Red Hat OpenShift-Webkonsole sehen.

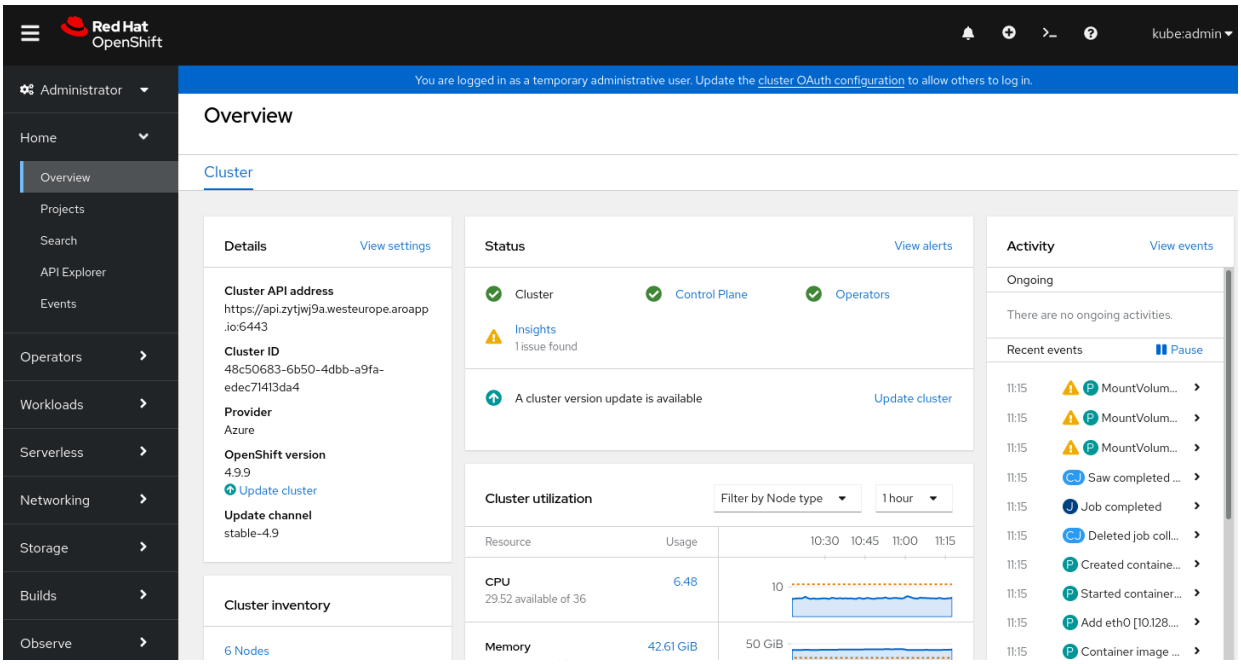


Abbildung 5.1: Azure Red Hat OpenShift-Webkonsole

Nehmen Sie sich einen Moment Zeit, um die Webkonsole zu erkunden. Die Konsole sollte eine aktuelle Version OpenShift ausführen und alle Komponenten sollten in einem fehlerfreien Zustand vorliegen, oder nach der Installation bald in einen fehlerfreien Zustand übergehen.

Über die OpenShift-CLI (oc)

Sie werden die aktuelle Version der OpenShift-CLI (oc) herunterladen müssen. Melden Sie sich hierfür einfach an und rufen Sie die folgende Seite auf: https://console.redhat.com/openshift/downloads.

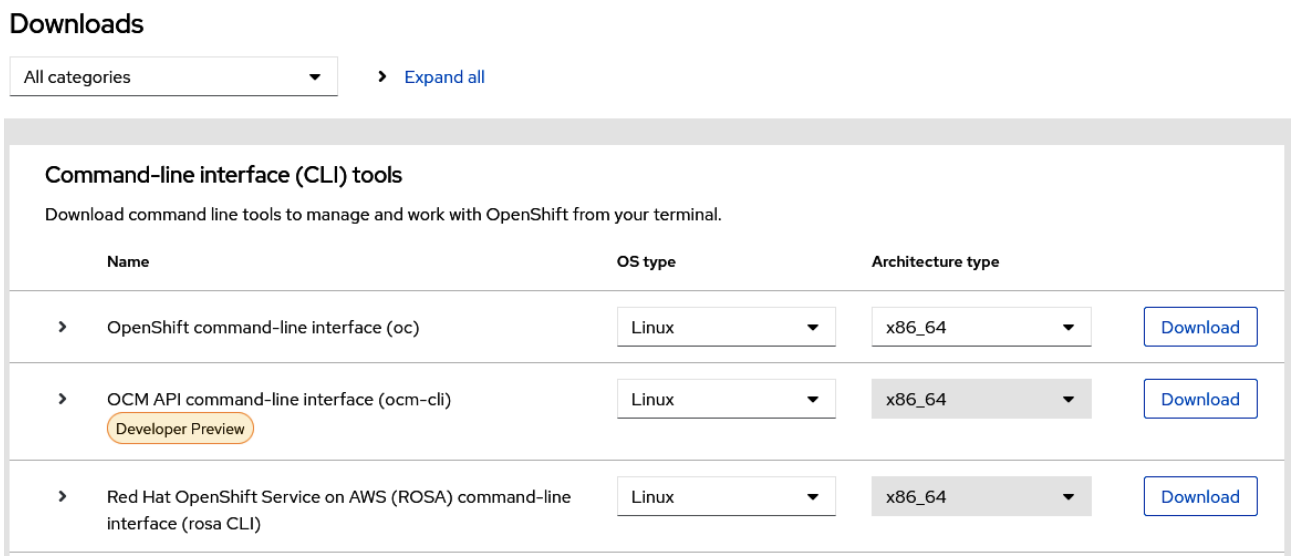


Abbildung 5.2: Herunterladen der OpenShift-Befehlszeile

Extrahieren Sie das Archiv (`.tar.gz` oder `.zip`) auf Ihrem System und geben Sie anschließend den Befehl `oc` in Ihrem Pfad ein. Auf Linux wird der Befehl `oc` normalerweise in `usr/local/sbin/` eingegeben.

Ausführen der OpenShift-CLI und Anmelden beim Cluster

Für die Authentifizierung bei Ihrem Cluster über die Befehlszeile müssen Sie den login-Befehl und das Token über die Webkonsole abrufen. Melden Sie sich mit dem Browser in der Webkonsole an, klicken Sie auf den Benutzernamen oben rechts und anschließend auf **Copy login command**.

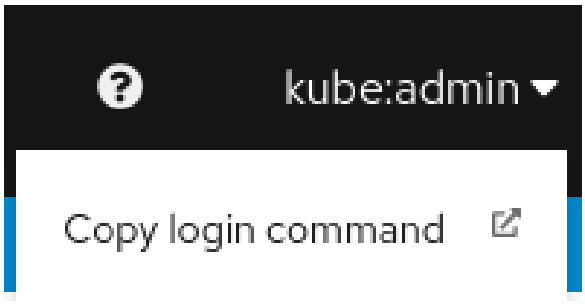


Abbildung 5.3: Copy login command

Es wird eine neue Seite geöffnet, die wie folgt aussieht:

Abbildung 5.4: Anmeldung mit dem API-Token

Sie können anschließend diesen Befehl kopieren und ihn im Terminal einfügen, um sich bei Ihrem Azure Red Hat OpenShift-Cluster anzumelden.

Wenn Sie beispielsweise die Azure Cloud Shell (Bash)-Session in Ihrem Azure-Portal verwenden, fügen Sie diesen login-Befehl ein. Anschließend sollte der Befehl `oc status` in etwa wie folgt aussehen:

```
user@Azure: oc status
In project default on server https://api.cyki1k6g.westeurope.aroapp.io:6443

svc/openshift - kubernetes.default.svc.cluster.local
svc/kubernetes - 172.30.0.1:443 -> 6443

View details with 'oc describe <resource>/<name>' or list everything with 'oc get all'.
```

Sie können sich jetzt einen Moment Zeit nehmen, um die `oc`-Befehlszeile zu erkunden und sich mit Ihrer neuen Azure Red Hat OpenShift-Umgebung vertraut zu machen. Wenn Sie bereits ein erfahrener OpenShift 4-Nutzer sind, sollte Ihnen dieser Cloud-Service von Azure Red Hat OpenShift bekannt vorkommen und er sollte sich genau auf dieselbe Weise verhalten, wie andere OpenShift 4-Umgebungen, die Sie möglicherweise in der Vergangenheit verwendet haben.

Zusammenfassung

Dies war ein sehr kurzes Kapitel, da die offiziellen Provisionierungsanweisungen sehr umfassend sind und als maßgeblicher Guide zur Einrichtung von Azure Red Hat OpenShift mit Ihrer Azure-Subskription dienen sollen. Sie werden bemerken, dass die Provisionierungsanweisungen weitaus einfacher als die Anweisungen für das Provisionieren einer selbst gemanagten OpenShift-Umgebung sind. Dies liegt daran, dass ein erheblicher Engineering-Aufwand hinter dem Azure Red Hat OpenShift-Service steckt, womit sowohl eine enge Integration mit Azure als auch eine Bereitstellung einer vorgegebenen, umfassend getesteten und gut unterstützen Universalarchitektur gewährleistet werden kann. Letztendlich ist dies ein Vorteil für Organisationen, da weniger Zeit für das Provisionieren von Azure Red Hat OpenShift aufgewendet werden muss und mehr Zeit für das Bereitstellen von Anwendungen übrig bleibt.

Kapitel 6

Nach der Provisionierung – Day 2

Nachdem Azure Red Hat OpenShift bereitgestellt wurde, gibt es einige Aktivitäten, die nach der Provisionierung erforderlich sind, bevor ein Cluster bereit für die Produktion ist. Dieses Kapitel behandelt viele der Aktivitäten, darunter:

- Authentifizierung – Azure Active Directory – einschließlich der Methode zur Synchronisierung von Benutzergruppen mit einem Community-Operator
- Operatoren – einschließlich OperatorHub
- Erläuterungen zur Protokollierung – einschließlich Protokollweiterleitung
- Erläuterungen zur Überwachung – einschließlich der Überwachung von OpenShift-Containern
- Upgrades und Patches – einschließlich des unterstützten Version-Lifecycles
- Cluster-Skalierung – einschließlich der manuellen und automatisierten Skalierung von Clustern
- Anwendungsskalierung – eine kurze Anmerkung zur Anwendungsskalierung
- Einrichtung eines „Pull"-Secrets – Registrierung mit OpenShift Cluster Manager
- Limitbereiche – einschließlich der Anwendungsfälle von Limitbereichen
- Persistenter Storage – einschließlich der verfügbaren und unterstützten Storage-Klassen
- Sicherheit und Compliance – eine Anmerkung zur Sicherheitskontrollen

Diese Abschnitte, die jeweils eine andere Aufgabe nach der Provisionierung beschreiben, helfen Ihnen beim Verständnis dessen, was alles benötigt wird, um einen kürzlich bereitgestellten Cluster für Produktionsanwendungen vorzubereiten.

Authentifizierung – Azure Active Directory

Azure Red Hat OpenShift unterstützt alle Authentifizierungsanbieter, die in der OpenShift-Dokumentation aufgeführt sind. Sehen Sie sich die vollständige Liste der Authentifizierungsanbieter an. Der Großteil der Kunden verwendet jedoch mit hoher Wahrscheinlichkeit Azure Active Directory zur Authentifizierung und für das Single Sign-On in Azure Red Hat OpenShift, da dieser Anbieter bereits auf Azure verfügbar ist.

Die Konfiguration für die Integration von Azure Active Directory ist bewusst ein „Day 2"-Ablauf, da es Fälle gibt, in denen Organisationen einen alternativen Authentifizierungsanbieter verwenden möchten. Der Einrichtungs- und Installationsvorgang für Azure Active Directory dauert in etwa 15 Minuten bei der Ersteinrichtung. Wenn Sie sich anschließend mit dem Prozess weiter vertraut gemacht haben, werden Sie in der Lage sein, ihn in ungefähr fünf Minuten zu durchlaufen. Viele Organisationen entscheiden sich dafür, Teile dieser Provisionierung mithilfe von Tools wie ARM-Vorlagen oder Ansible Playbooks zu automatisieren.

- Konfigurieren von Azure Active Directory für Azure Red Hat OpenShift (grafisches Portal)
- Konfigurieren von Azure Active Directory für Azure Red Hat OpenShift (Befehlszeile)

Verwenden von Active Directory-Benutzergruppen

Die Konfiguration der Azure Active Directory-Authentifizierung ermöglicht es Nutzern nur, sich mit bestehenden Anmeldedaten anzumelden. Es werden nicht die bestehenden Benutzergruppen für einen Nutzer in Red Hat OpenShift übernommen. Es kommt häufig vor, dass eine Organisation Benutzergruppen von Active Directory importieren möchte, sodass sie für die Konfiguration von Benutzerberechtigungen innerhalb von OpenShift verwendet werden können. Es gibt einen separaten, von der Community unterstützten Operator, der hierfür verfügbar ist – der Group Sync Operator.

- Group Sync Operator auf GitHub

Beachten Sie, dass der Group Sync Operator von der Community unterstützt wird. Dies bedeutet, dass er zum Zeitpunkt der Veröffentlichung nicht offiziell von Red Hat oder Microsoft unterstützt wird. Dieses Buch empfiehlt, den detaillierten Konfigurations- und Einrichtungsanweisungen für den Operator in der README-Datei des Projekts zu folgen.

Operatoren

Operatoren in OpenShift 4 gehören zu den fundamentalen Entwicklungskomponenten der Plattform und bieten einen erheblichen Mehrwert für Kunden von Red Hat OpenShift. Operatoren bestehen aus Code und führen einen Service in einem Cluster-Container aus. Einige Operatoren sind unter anderem für die Wartung von Cluster-Networking, der Maschinenkonfiguration und der Cluster-Upgrades verantwortlich. Solche Operatoren werden oft Cluster-Operatoren genannt. Sie finden eine Liste dieser Operatoren im Abschnitt **Administration** der OpenShift-Konsole.

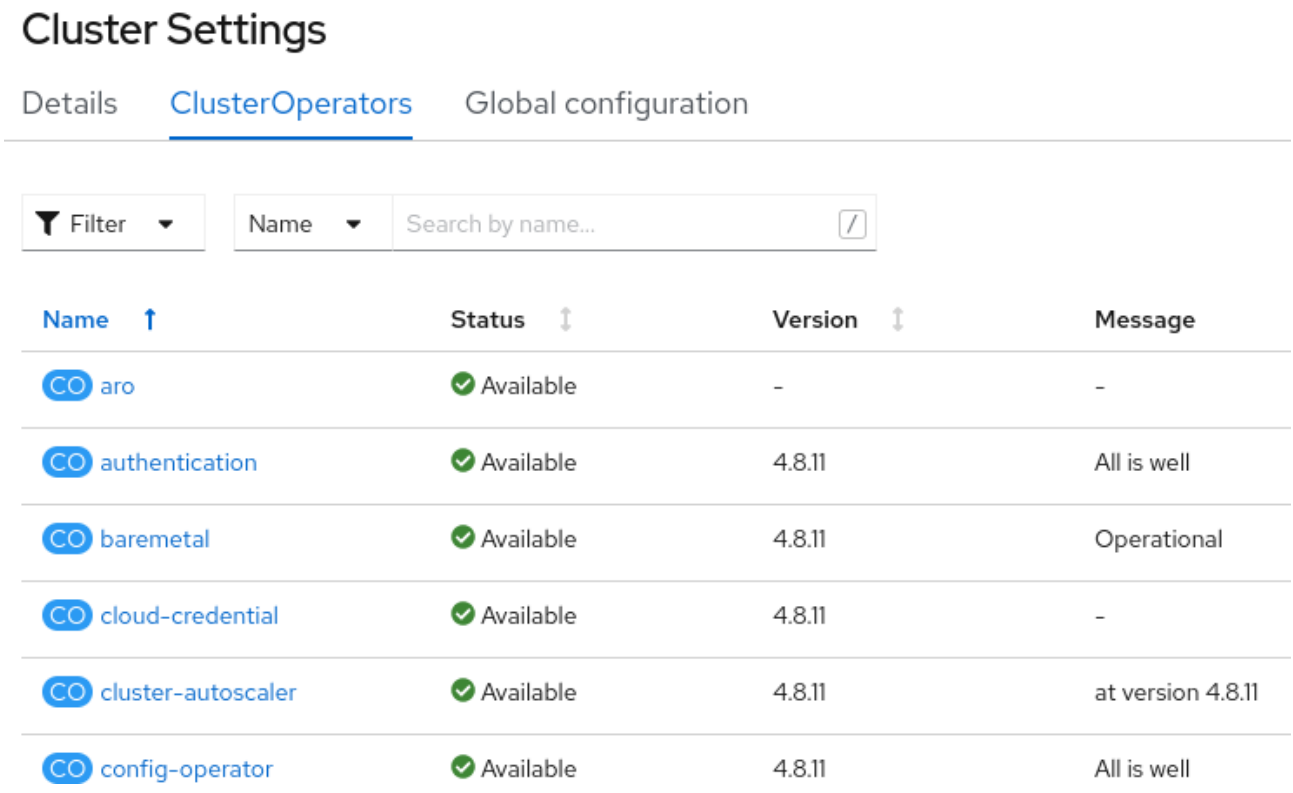


Abbildung 6.1: Cluster-Einstellungen

Sie können anhand des oben abgebildeten Screenshots sehen, dass es einen Operator gibt, der nur für Azure Red Hat OpenShift da ist, um Teile des Services zu warten und sicherzustellen, dass der Cluster einen unterstützten Konfigurationszustand beibehält.

Operatoren können viele Dinge warten, wie etwa den Zustand eines Services, Updates, Patches, Skalierungen und weitere Funktionen.

OperatorHub

Über OperatorHub haben Administratoren auf viele weitere Operatoren Zugriff, die über den standardmäßigen Cluster-Operator, der im Hintergrund auf jedem Cluster ausgeführt wird, hinausgehen. OperatorHub vereinfacht die Suche nach beliebten Community- und Unternehmens-Operatoren, die ein Administrator möglicherweise mit der Red Hat OpenShift-Installation zur Verfügung stellen möchte. OperatorHub befindet sich in der Seitenleiste von jedem OpenShift-Cluster.

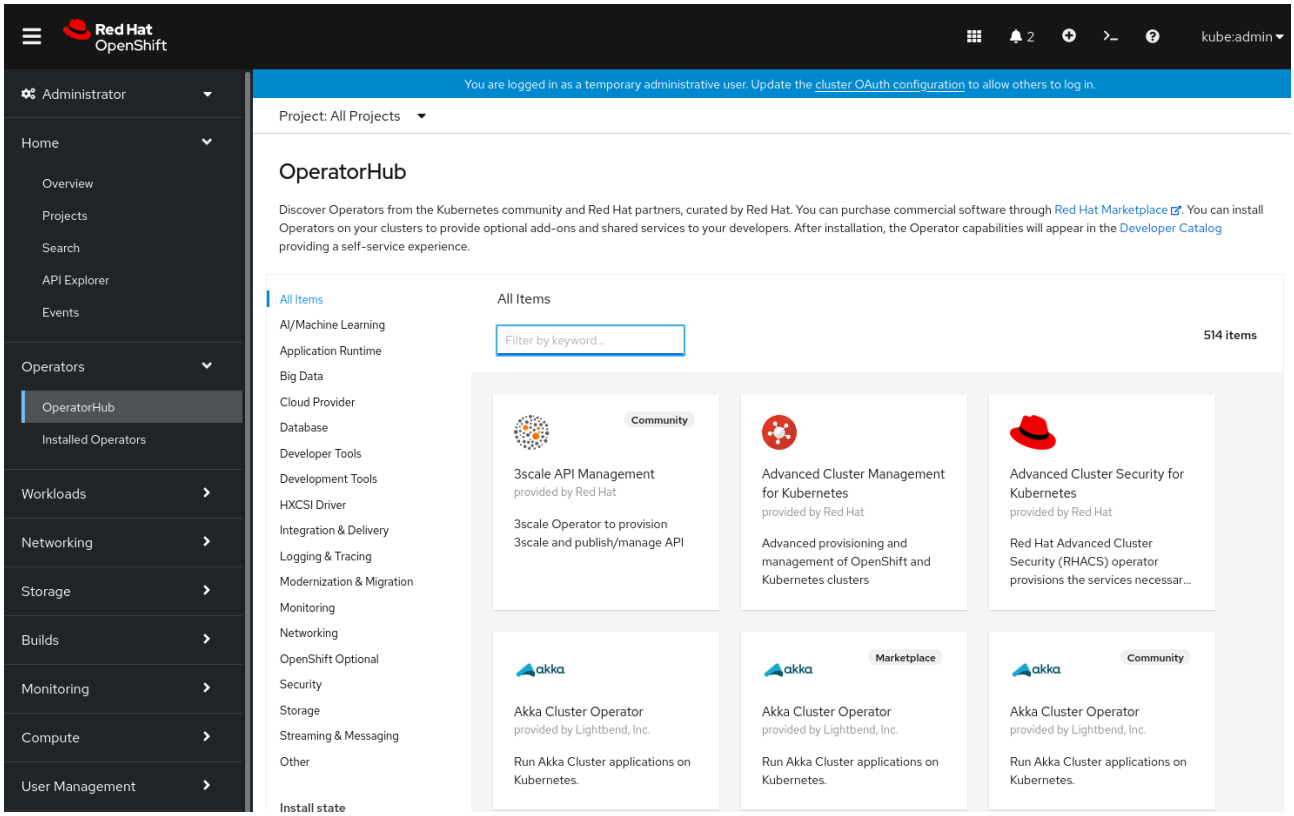


Abbildung 6.2: OperatorHub

Administrator können unter der Verwendung von Operatoren beliebte Software schnell und einfach bereitstellen und warten, ohne sich um die Konfiguration und den Code von Kubernetes kümmern zu müssen. Viele Produkte von Red Hat werden mittlerweile als Operatoren angeboten, wie etwa Advanced Cluster Management, Red Hat OpenShiftService Mesh und Red Hat OpenShift Pipelines. Es gibt jedoch auch eine umfassende Bibliothek mit Operatoren für Software, die nicht von Red Hat bereitgestellt wird, wie etwa MariaDB-Datenbanken, Couchbase oder IBM Block Storage-Treiber.

Weitere Informationen zu Operatoren finden Sie unter den folgenden empfohlenen Links:

- Operatorhub.io
- Operatoren in OpenShift

Erläuterungen zur Protokollierung

Azure Red Hat OpenShift verwendet dieselbe Protokollierungs- und Überwachungsarchitektur wie Red Hat OpenShift. Beide Angebote nutzen dieselben Operatoren für die Protokollierung.

Protokolle können in drei Kategorien aufgeteilt werden:

- **Anwendung:** Containerprotokolle, die von Benutzeranwendungen generiert werden, die im Cluster ausgeführt werden, mit Ausnahme von Infrastruktur-Containeranwendungen.
- **Infrastruktur:** Protokolle, die von Infrastrukturkomponenten generiert werden, die im Cluster ausgeführt werden, und OpenShift Container Platform-Nodes wie Journal-Protokolle. Infrastrukturkomponenten sind Pods, die in `openshift*`-, `kube*`- oder standardmäßigen Projekten ausgeführt werden.
- **Audit:** Protokolle, die von auditd, dem Node-Auditsystem, generiert werden und in der Datei `/var/log/audit/audit.log` gespeichert werden, und Protokolle vom Kubernetes API- und OpenShift API-Server.

Eine detailliertere Beschreibung der OpenShift-Protokollierung finden Sie unter Understanding Red Hat OpenShift Logging in der Produktdokumentation.

Verwenden der Weiterleitung von Cluster-Protokollen an Azure Monitor

Das Senden von Azure Red Hat OpenShift-Protokollen an den Container Insights-Service von Azure Monitor ist eine gängige Integration. Dieser Vorgang wird manchmal Azure Log Analytics genannt. Dies ermöglicht eine persistente und kostengünstige Protokollaufbewahrung in Azure.

Die Protokollierungsarchitektur von OpenShift führt auf jedem Node Fluent Bit aus. Ein Operator würde zunächst die Konfiguration dieses Fluent Bit-Services so ändern, dass die Protokolle direkt gesendet werden. Das Ändern der Protokollierungskonfiguration in Azure Red Hat OpenShift ist jedoch nicht Teil der Unterstützungsrichtlinie. OpenShift hat einen integrierten, unterstützen Mechanismus für die Weiterleitung von Protokollen mit dem Namen `ClusterLogForwarder`.

Wenn Sie `ClusterLogForwarder` verwenden, ist es möglich, Protokolle an Elasticsearch, Fluentd und syslog weiterzuleiten. Azure Monitor unterstützt diese Protokolle jedoch nicht direkt. Ein Workaround hierfür besteht darin, eine zusätzliche Fluent Bit-Instanz zwischenzuschalten, welche Protokolle von OpenShift empfängt und sie an Azure Monitor weiterleitet. Dieser Ansatz wird in der Dokumentation von Microsoft und auch in der Community-Dokumentation beschrieben.

Erläuterungen zur Überwachung

Es ist zwar nicht erforderlich, eine komplexe, benutzerdefinierte Überwachung für Azure Red Hat OpenShift zu implementieren, da dies Teil des gemanagten Service ist, für den Sie bezahlen.

Jedoch möchten viele Kunden OpenShift-Container mit Azure Container Insights überwachen. Dies ist eine Funktion, die sich derzeit in der öffentlichen Vorschau befindet und in dieser Dokumentation beschrieben wird.

Upgrades und Patches

Wenn Sie einen Azure Red Hat OpenShift-Cluster provisionieren, wird kein Update-Kanal ausgewählt. Dies bedeutet, dass Ihr Cluster nicht von Anfang an standardmäßig Updates erhält.

Um Ihren Update-Kanal zu sehen, navigieren Sie in der Navigationsseitenleiste zu **Administration**→ **Cluster Settings**. So sehen die Cluster-Einstellungen aus, kurz nachdem ein Azure Red Hat-OpenShift-Cluster provisioniert wurde:

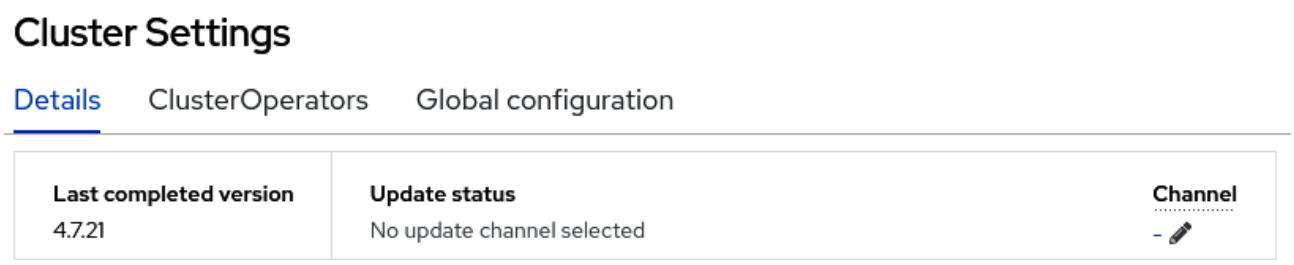


Abbildung 6.3: Cluster-Einstellungen nachdem ein Cluster provisioniert wurde

Um einen Update-Kanal auszuwählen, wählen Sie die Verknüpfung „-“ aus und sehen Sie sich die verfügbaren Optionen an:

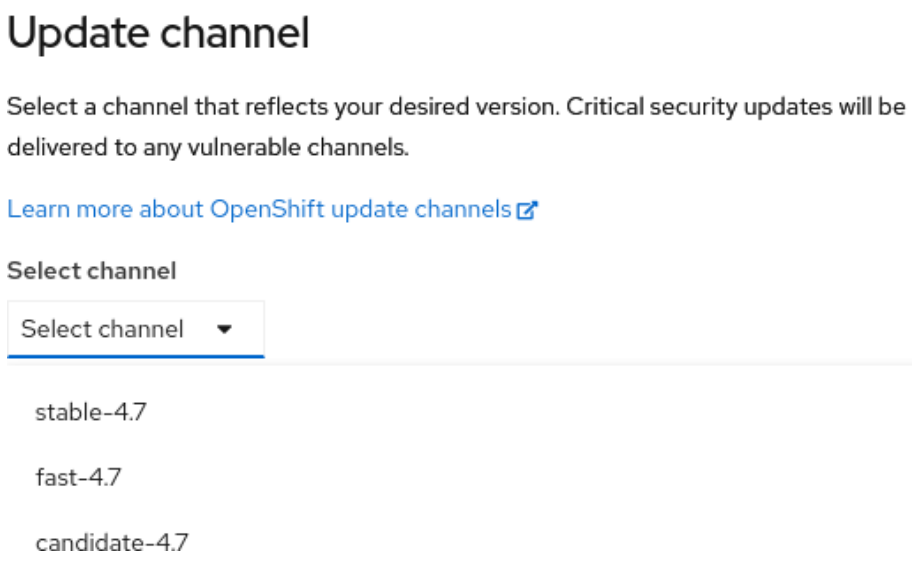


Abbildung 6.4: Auswahl eines Update-Kanals

Um die Unterschiede zwischen `stable`, `fast` und `candidate` zu verstehen, sehen Sie sich die Dokumentationsseite [Understanding upgrade channels and releases] an.

Es wird dringend empfohlen, für alle Cluster in der Produktion den Kanal **stable** auszuwählen.

Unterstützter Versions-Lifecycle

Es ist wichtig, zu wissen, welche Versionen von Azure Red Hat OpenShift unterstützt werden, wenn Sie Ihr Produktions-Deployment planen. Auf der formalen Lifecycle-Seite gibt es Informationen, die Organisationen dabei helfen zu verstehen, welche Versionen unterstützt werden und welche nicht.

[Seite zum Support-Lifecycle für Azure Red Hat OpenShift]

Um einen Teil der Informationen auf dieser Seite vereinfacht zusammenzufassen:

Azure Red Hat OpenShift unterstützt zwei Nebenversionen der Red Hat OpenShift Container Platform, die **Generally Available (GA)** sind:

- Die aktuelle GA-Nebenversion, welche in Azure Red Hat OpenShift veröffentlicht wird (im Weiteren als „N“ bezeichnet)
- Eine vorherige Nebenversion („N-1“)

Die Red Hat OpenShift Container Platform verwendet die semantische Versionierung. Die Semantische Versionierung verwendet unterschiedliche Ebenen von Versionsnummern, um verschiedene Versionierungsebenen zu unterscheiden. Die folgende Tabelle veranschaulicht die verschiedenen Teile einer semantischen Versionierungsnumnmer in diesem Beispiel anhand der Versionsnummer 4.9.3:

Hauptversion (x)	Nebenversion (y)	Patch (z)
4	9	3

Jede Nummer in der Version weist auf eine allgemeine Kompatibilität mit der vorherigen Version hin:

- **Hauptversion:** Es wird derzeit keine Veröffentlichung einer Hauptversion geplant. Hauptversionen ändern sich, wenn beispielsweise wenn eine API durch eine Änderung inkompatibel wird oder die Abwärtskompatibilität defekt ist.
- **Nebenversionen:** werden ungefähr alle drei Monate veröffentlicht. Nebenversions-Upgrades können Funktionserweiterungen, Verbesserungen, Einstellungen veralteter Elemente, Entfernungen, Bug Fixes und Sicherheitsverbesserungen enthalten.
- **Patches:** werden üblicherweise jede Woche oder nach Bedarf veröffentlicht. Patchversions-Upgrades können Bug Fixes und Sicherheitsverbesserungen enthalten.

Für einen ausführlicheren Blick auf die unterstützten Versionen, lesen Sie die Seite Support-Lifecycle für Azure Red Hat OpenShift.

Skalierung von Clustern

Red Hat OpenShift Container Platform und somit auch Azure Red Hat OpenShift wurden mit einer skalierbaren Architektur entwickelt. Im Zusammenhang mit der Skalierung von OpenShift, müssen Administratoren und Anwendungseigentümer üblicherweise die Cluster-Skalierung und die Anwendungsskalierung voneinander unterscheiden.

Dieser Abschnitt behandelt die Cluster-Skalierung und anschließend gibt es eine kurze Anmerkung zur Anwendungsskalierung als separaten Abschnitt.

Unterstützte maximale Anzahl

Cluster-Skalierung bedeutet, weitere Worker-Nodes zum Cluster hinzuzufügen. Somit werden zusätzliche Computing-Ressourcen für Anwendungen im Cluster geschaffen. Es stellt sich natürlich die Frage, wann es notwendig ist, auch die Control Plane zu skalieren. Azure Red Hat OpenShift stellt schon drei Control Plane-Nodes bereit, welche prinzipiell genug Kapazität haben, um auf die maximal unterstützte Anzahl (derzeit 60) an Worker-Nodes in einem Azure Red Hat OpenShift-Cluster zu skalieren.

Zum Zeitpunkt der Veröffentlichung gibt es drei unterstützte Instanzgrößen für Control Plane-Nodes: `Standard_D8s_v3`, `Standard_D16s_v3`, and `Standard_D16s_v3`.

Für die Worker-Nodes werden mehr Azure-Instanztypen unterstützt: rechenoptimiert (F-Serie), speicheroptimiert (E-Serie) und universell (D-Serie).

Auf der **Seite zur Unterstützungsrichtlinie** finden Sie eine Liste der derzeit unterstützten Azure-Instanztypen für Azure Red Hat OpenShift.

Minimales Deployment und Scale-to-Zero

Organisationen möchten hin und wieder kleinere Cluster für Test- und Entwicklungszwecke oder andere Use Cases, für die eine reduzierte Verfügbarkeit akzeptabel ist, bereitstellen. Derzeit ist die minimale Cluster-Größe drei Control Plane-Nodes und drei Anwendungs-Nodes. Eine noch kleinere Skalierung wird nicht unterstützt.

Manuelles Skalieren des Clusters

Operatoren, die sich das Azure-Portal ansehen, sind möglicherweise versucht, zusätzliche Worker-Nodes zum Azure Red Hat OpenShift-Cluster hinzuzufügen, indem Sie direkt eine virtuelle Maschine in Azure erstellen. Dies ist jedoch nicht möglich, da die Ressourcengruppe, die Azure Red Hat OpenShift beinhaltet, aus der Sicht eines Azure-Administrators „gesperrt" ist. Dies ist unabhängig von den Berechtigungen. Selbst Nutzer mit der **owner**-Berechtigung können nicht die Inhalte einer Azure Red Hat OpenShift-Ressourcengruppe ändern. Daher werden Ihnen Fehlermeldungen zur Ablehnung der Berechtigung angezeigt, wenn Sie versuchen, innerhalb der Azure Red Hat OpenShift-Ressourcengruppe eine virtuelle Maschine zu erstellen.

Die vorgesehene Methode zur Skalierung in Azure Red Hat OpenShift ist die Verwendung der OpenShift-Funktion `MachineSets`, mit der OpenShift, wenn aufgefordert, einen neuen Anwendungs-Node erstellt. Nutzer mit Cluster-Admin-Berechtigungen finden **MachineSets** unter **Compute**.

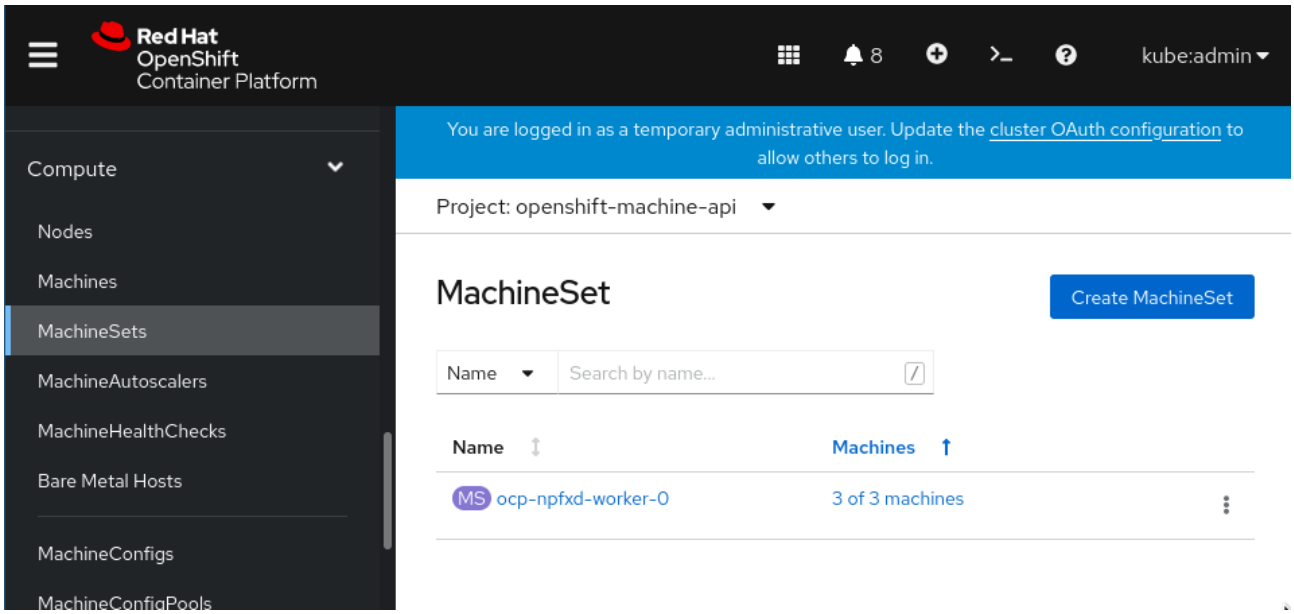


Abbildung 6.5: Admin-Zugriff auf MachineSets

Wenn Sie auf das Menü „Edit" eines `MachineSet`-Anwendungs-Nodes klicken, sehen Sie, dass Sie einfach eine neue virtuelle Maschine für den Anwendungs-Node provisionieren können, indem Sie **Edit Machine Count** auswählen.

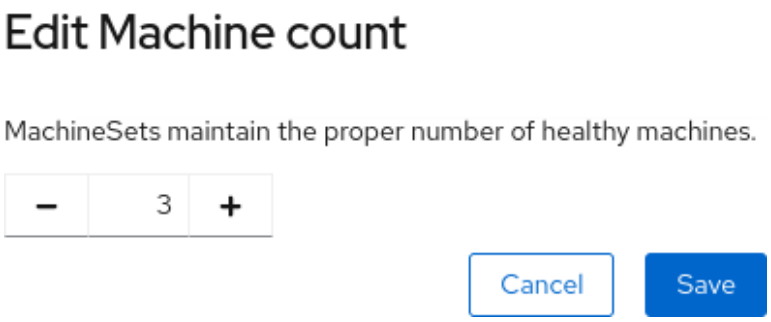


Abbildung 6.6: Bearbeiten der Maschinenanzahl

Sobald die Anzahl aktualisiert wurde, können Administratoren entweder in das Azure-Portal oder zur Ansicht **Compute** → **Machines** wechseln, um die Provisionierung einer neuen virtuellen Maschine für den Anwendungs-Node zu sehen.

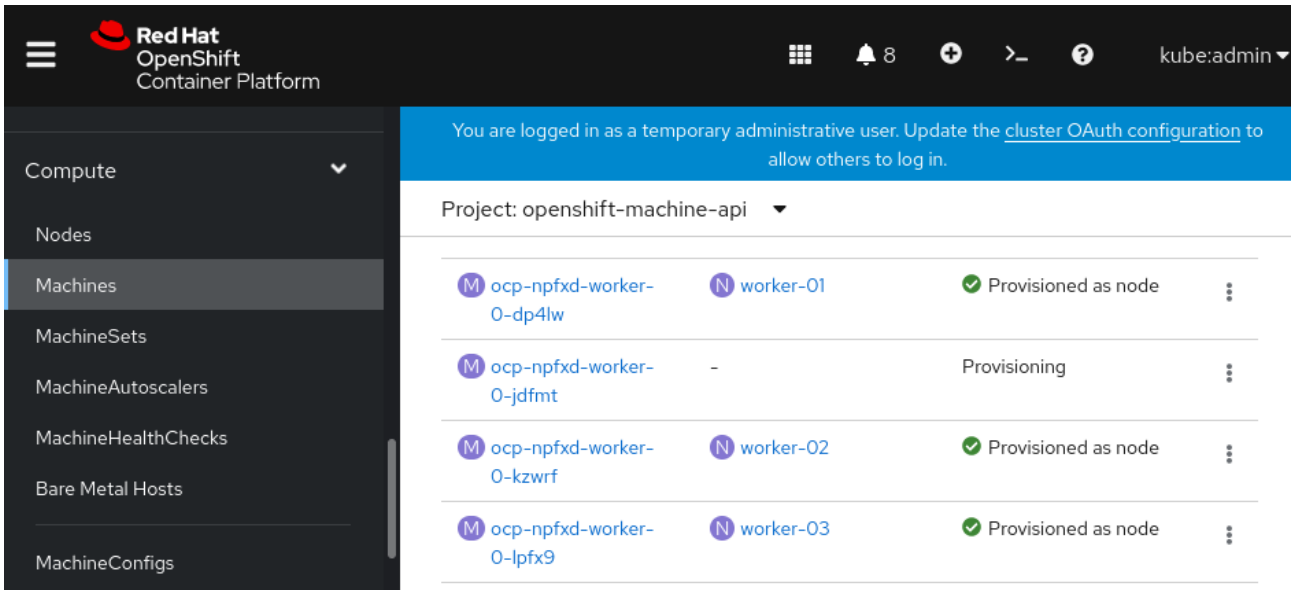


Abbildung 6.7: Die Ansicht „Machines“

Das Provisionieren einer zusätzlichen virtuellen Maschine in den meisten Azure-Regionen dauert normalerweise bis zu fünf Minuten.

Administratoren müssen keine Aktivitäten nach der Provisionierung durchführen, um diese neue Kapazität online zu bringen. OpenShift stellt sie dem Cluster automatisch zur Verfügung und Anwendungen nehmen sie bei Bedarf in Anspruch.

Automatische Skalierung von Clustern

Für das standardmäßige Deployment von Azure Red Hat OpenShift ist die Funktion zur automatischen Skalierung nicht vorab aktiviert. Die Aktivierung dieser Funktionalität ist jedoch unkompliziert. Administratoren müssen einfach eine `MachineAutoscaler`-Ressource erstellen, die sich im Seitenleistenmenü **Compute** von Azure Red Hat OpenShift befindet.

`MachineAutoscaler`-Ressourcen steuern ein `MachineSet`, welches wiederum Kapazitäten der virtuellen Maschine für den Anwendungs-Node erstellt oder löscht. Das folgende Beispiel zeigt, dass es möglich ist, eine minimale oder maximale Anzahl an Maschinen in einem `MachineSet` festzulegen:

```
apiVersion: autoscaling.openshift.io/v1beta1
kind: MachineAutoscaler
metadata:
    name: worker-us-east-1a
    namespace: openshift-machine-api
spec:
    minReplicas: 1
    maxReplicas: 12
    scaleTargetRef:
        apiVersion: machine.openshift.io/v1beta1
        kind: MachineSet
        name: worker
```

OpenShift verfügt außerdem über das Konzept von Cluster-Autoscalern, mit denen Sie so skalieren können, dass immer eine bestimmte Menge an RAM, CPU oder Ähnliches verfügbar bleibt. Die Entscheidung zwischen dem `MachineAutoscaler` oder dem `ClusterAutoscaler` hängt davon ab, auf welche Weise Sie Kapazitäten in Ihrem Cluster hinzufügen und entfernen möchten.

Red Hat OpenShift-Dokumentation zum MachineAutoscaler und ClusterAutoscaler

Anwendungsskalierung

Die Skalierung von Microservice-Anwendungen ist ein komplexes Thema, welches nicht im Umfang dieses Buchs enthalten ist. Es gibt jedoch zwei Tipps für nützliche Seiten zum Einstieg in dieses Thema:

- HorizontalPodAutoscaler: Geben Sie die minimale und maximale Anzahl an Pods an, die Sie ausführen möchten, sowie die CPU- oder Arbeitsspeicher-Nutzung, die Ihre Pods als Ziel haben sollten.
- Serverless und Scale-to-Zero mit Knative.

Der Cluster überwacht die ausgeführten Container und die verbleibende Kapazität auf dem Cluster. Wenn Knative oder der `HorizontalPodAutoscaler` mehr Ressourcen anfordern, als im Cluster verfügbar sind, muss eine Skalierung von `ClusterAutoscaler` oder `MachineSet` stattfinden, um die angeforderten Ressourcen dem Cluster hinzuzufügen.

Mit diesem Ansatz können Anwendungen und der Cluster selbst je nach Bedarf gleichzeitig hoch- und herunterskalieren.

Einrichtung eines „Pull"-Secrets (Registrierung mit OpenShift Cluster Manager)

Bei einem neuen Deployment von Azure Red Hat OpenShift ist für `cloud.redhat.com` kein „Pull"-Secret konfiguriert. Dies bedeutet, dass Ihre Cluster nicht standardmäßig in der Red Hat Hybrid Cloud-Konsole (http://console.redhat.com) angezeigt werden – diese Konsole wird OpenShift Cluster Manager genannt.

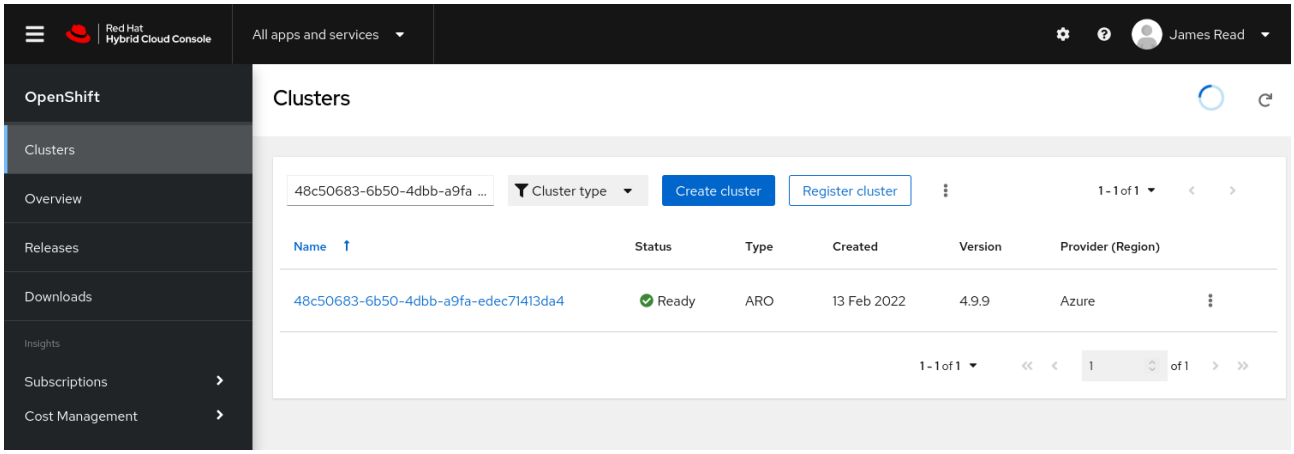


Abbildung 6.8: OpenShift Cluster Manager mit einem angezeigten Azure Red Hat OpenShift-Cluster

Das Konfigurieren eines „Pull"-Secrets für `cloud.redhat.com` ist sehr unkompliziert. Nach der Konfiguration wird es im OpenShift Cluster Manager-Portal unter http://console.redhat.com angezeigt und Sie können dann Supporttickets über das Standard Support-Ticketsystem an Red Hat senden.

Hier finden Sie die Anweisungen zum Einrichten eines „Pull"-Secrets:

- Hinzufügen oder Aktualisieren eines „Pull"-Secrets

Ein zusätzlicher Vorteil eines eingerichteten „Pull"-Secrets ist, dass Kunden anschließend Supporttickets direkt an Red Hat senden können. Das „Pull"-Secret erteilt Ihrem Konto bei Red Hat eine Berechtigung, die Supportmitarbeitern ermöglicht, Ihren Cluster anzuzeigen. Das Erstellen eines Supporttickets für Azure Red Hat OpenShift funktioniert somit auf dieselbe Weise, wie für jedes andere Produkt von Red Hat.

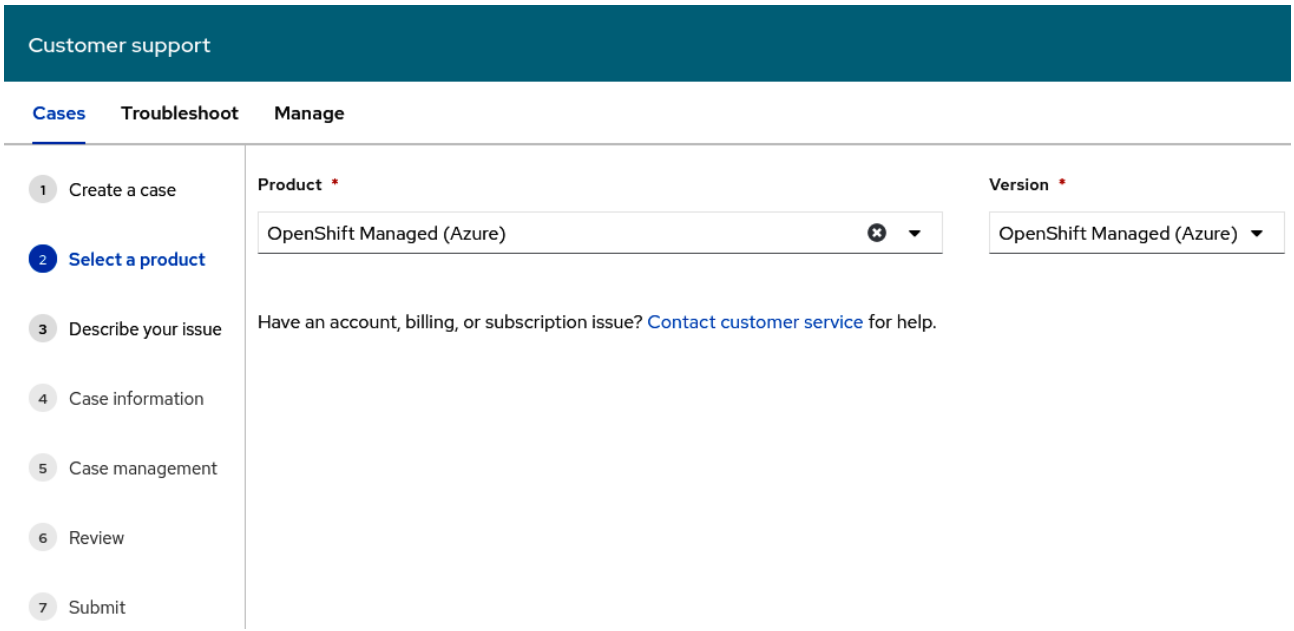


Abbildung 6.9: Erstellen eines Supportfalls

Limitbereiche

Wenn ein Entwicklungs- oder Anwendungsteam mit dem Bereitstellen von containerisierten Anwendungen beginnt, ist es nur eine Frage der Zeit, bis sich eine Anwendung falsch verhält und unnötig Ressourcen im Cluster verbraucht. Eine Ursache kann eine fehlerhafte Anwendung mit einem Speicherleck sein, oder eine Anwendung, für die versehentlich eingestellt wurde, dass sie skalieren soll, sobald sie nur 10 % der CPU verwendet, anstatt 100 %. Um Szenarien vorzubeugen, in denen sich diese falsch verhaltenden Anwendungen unkontrolliert horizontal skalieren und zu viele Ressourcen verbrauchen, wird die Verwendung von `LimitRange` empfohlen.

`LimitRange` ermöglicht es Ihnen, den Ressourcenverbrauch für bestimmte Objekte in einem Projekt einzuschränken. `LimitRange` für Folgendes angewendet werden:

- Pods und Container: Sie können die minimalen und maximalen Voraussetzungen für CPU- und Arbeitsspeicher für Pods und Container festlegen.
- Image Streams: Sie können Einschränkungen für die Anzahl an Images und Tags in einem `ImageStream`-Objekt festlegen.
- Images: Sie können die Größe von Images einschränken, die in eine interne Registry gepusht werden.
- **Persistent Volume Claims (PVCs):** Sie können die Größe von den anfragbaren PVCs einschränken.

Ein Beispiel für einen Limitbereich für Container – das Festlegen der minimal, maximal und standardmäßig erlaubten CPU- und Arbeitsspeicheranfragen – ist hier aufgeführt:

```
apiVersion: "v1"
kind: "LimitRange"
metadata:
  name: "resource-limits"
spec:
  limits:
    - type: "Container"
      max:
        cpu: "2"
        memory: "1Gi"
      min:
        cpu: "100m"
        memory: "4Mi"
      default:
        cpu: "300m"
        memory: "200Mi"
      defaultRequest:
        cpu: "200m"
        memory: "100Mi"
      maxLimitRequestRatio:
        cpu: "10"
```

Dieser `LimitRange`-Codeausschnitt kann angewendet werden, indem Sie die YAML direkt in den Editor der Red Hat OpenShift-Konsole einfügen oder dies über eine Datei mit `oc apply -f limitrange.yaml` tun.

[Dokumentation zu Limitbereichen]

Persistenter Storage

Die von Azure Red Hat OpenShift bereitgestellten virtuellen Maschinen enthalten Azure-Disks, mit denen Sie Red Hat CoreOS installieren und den Azure Red Hat OpenShift-Service ausführen können. Diese Disks sollten nur für den OpenShift-Cluster und nicht von Anwendungen verwendet werden.

Anwendungen, die persistenten Storage benötigen, sollten die Kubernetes-Funktion `PersistentVolume` verwenden. Dieses Konzept wird ausführlich in der OpenShift-Dokumentationsseite Understanding persistent storage beschrieben. Azure Red Hat OpenShift unterstützt zwar alle `PersistentVolume`-Anbieter als selbst gemanagte OpenShift-Installationen, auf Azure sind jedoch die am meisten verwendeten persistenten Storages die Folgenden:

Name	Schrift	Zugriffsmodi	Storage-Klasse
Azure Files	Dateisystem, keine Compliance mit POSIX	ReadWriteOnce	Azure File
Azure Disk	Block	ReadWriteOnce	Azure Disk
Red Hat OpenShift Data Foundation	Dateisystem, Block, Objekt	(Mehrere)	OCS/ODF docs

Sicherheit und Compliance

Die Red Hat OpenShift-Dokumentation enthält jetzt viele einige Inhalte zur Implementierung vieler Sicherheitskontrollen und zur Beibehaltung der Compliance.

OpenShift-Dokumentation zu Sicherheit und Compliance

Dieser Abschnitt der Dokumentation enthält Folgendes:

- Eine detaillierte Beschreibung der Funktionsweise der Containersicherheit in OpenShift. Sie sollten wissen, dass OpenShift viele Out-of-the-Box-Sicherheitskontrollen für Container anbietet, die es nicht in anderen Kubernetes-basierten Services gibt. Diese Kontrollen unterstützen Sie dabei, Ihre Organisation und Ihre Anwendungen zu schützen.
- Auf Pod-Schwachstellen scannen
- Protokollzugriff prüfen
- Zertifikate konfigurieren

Es sind auch einige hilfreiche sicherheitsbezogene Operatoren enthalten, wie etwa:

- **Compliance Operator:** führt Scans aus und bietet Empfehlungen zur Behebung gängiger Sicherheitsprobleme an.
- **File Integrity Checking:** überprüft kontinuierlich, ob sich Dateien (insbesondere sensible Sicherheitskonfigurationsdateien) ändern.

Zusammenfassung

In diesem Kapitel haben wir die meisten Aufgaben und Themen behandelt, die Organisationen üblicherweise berücksichtigen, wenn sie einen neu bereitgestellten Cluster für Produktionsanwendungen vorbereiten. Es ist zwar nicht notwendig, sich bei jedem nachfolgenden Deployment diese Themen nochmal durchzulesen, jedoch sollten Sie beim Bereitstellen Ihres ersten Clusters Persistent Storage, Limits, Authentifizierung und die anderen Themen in diesem Kapitel berücksichtigen.

Organisationen versuchen meistens so viele Aufgaben nach der Provisionierung wie möglich zu automatisieren. Die Einrichtung und Konfiguration von Azure Directory kann beispielsweise größtenteils mit einem PowerShell-Skript oder einigen wenigen ARM-Vorlagen durchgeführt werden. Dies hilft Zeit bei repetitiven Aufgaben zu sparen, wenn Sie viele Cluster bereitstellen.

Das nächste Kapitel in diesem Buch behandelt das Bereitstellen einer Beispielanwendung in Ihrem produktionsfähigen Azure Red Hat OpenShift-Cluster.

Kapitel 7

Bereitstellen einer Beispielanwendung

Der Inhalt dieses Guides dient hauptsächlich der Unterstützung von technischen Experten (Entwickler und Operatoren), die erfahren möchten, was die Anforderungen für die produktive Verwendung von Azure Red Hat OpenShift sind. Dieser Guide setzt voraus, dass der Leser über ein grundlegendes Verständnis von Red Hat OpenShift verfügt, da ein großer Teil der Architektur, der Managementportale und des Benutzererlebnisses identisch mit Azure Red Hat OpenShift ist.

Dieses Kapitel dient als erklärende Einführung zum Bereitstellen einer einfachen Beispielanwendung mit dem Namen „Fruit Smoothies“. Diese Anwendung soll als schneller Einstieg und als Veranschaulichung dienen, sodass Sie besser nachvollziehen können, wie Sie Azure Red Hat OpenShift verwenden. Beachten Sie, dass dies eine Beispielanwendung für Azure Kubernetes Service ist und das Bereitstellen auf Azure Red Hat OpenShift soll aufzeigen, dass OpenShift zu 100 % kompatibel mit Kubernetes ist.

Dieses Kapitel basiert größtenteils auf http://aroworkshop.io und es wurden zusätzliche Beschreibungen und Kontext hinzugefügt.

Eine Übersicht der Anwendung für Bewertungen

Die Anwendungsarchitektur ist simpel und sieht folgendermaßen aus:

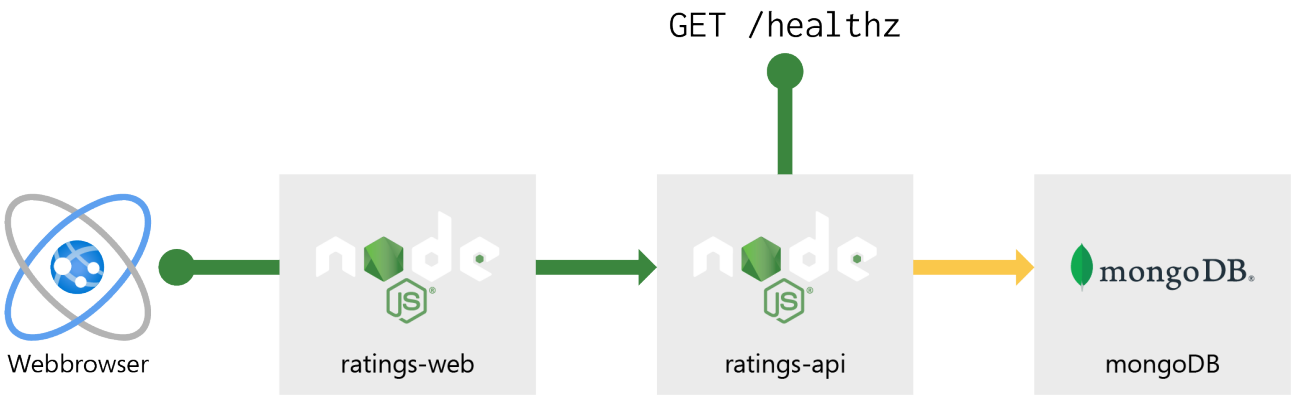


Abbildung 7.1: Architektur der Anwendung „Fruit Smoothies“

Im oben stehenden Diagramm können Sie sehen, dass die Anwendung aus drei Services und zwei öffentlichen Endpunkten besteht, die in der Tabelle unten aufgeführt sind. Der erste Endpunkt auf der linken Seite des Diagramms steht für den Webbrowser eines Nutzers, der die öffentliche HTML-Webanwendung anzeigt. Der zweite Endpunkt, `healthz`, ist eine Zustandsprüfung im Service `ratings-api`. In der folgenden Tabelle befinden sich Beschreibungen der individuellen Services und Links zu deren Repositories:

Komponente	Beschreibung	GitHub-Repository
`rating-web`	Ein öffentliches Web-Frontend – die „Website“.	GitHub-Repo
`rating-api`	Dieser Service nimmt den Input der Web-Benutzeroberfläche und speichert sie in der Datenbank. Er gibt außerdem Ergebnisse aus der Datenbank über Port 3000 an die Webanwendung zurück.	GitHub-Repo
`mongodb`	Eine NoSQL-Datenbank mit vorgeladenen Daten.	Daten

Sie können die GitHub-Repository-Links aufrufen, um diese Anwendungen weiter zu erkunden und zu verstehen. Die folgende Anleitung enthält Schritt-für-Schritt-Anweisungen zum Bereitstellen der jeweiligen Anwendung.

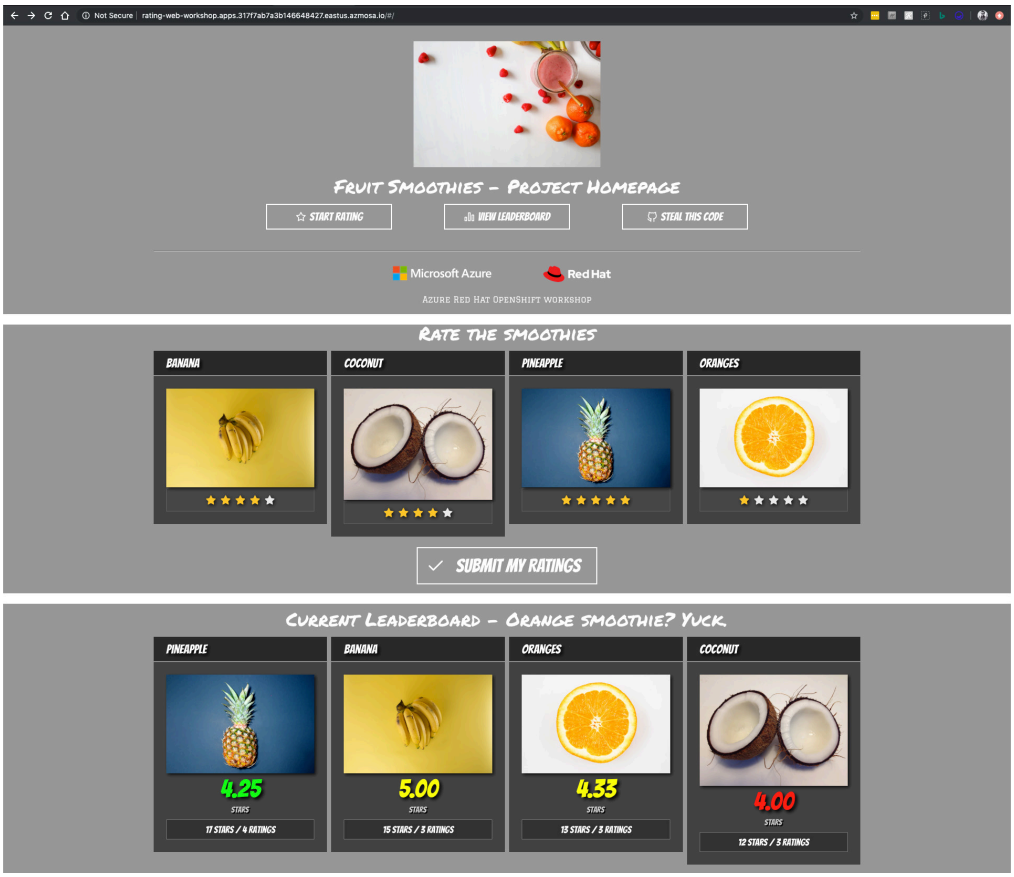


Abbildung 7.2: Screenshots zur Übersicht über die Anwendung „Fruit Smoothies“

Sobald Sie fertig sind, haben Sie eine laufende Webanwendung, die ähnlich wie die oben stehenden Screenshots aussieht. Sie werden außerdem besser verstehen, wie Entwickler- und Operations-Teams Anwendungen in Azure Red Hat OpenShift bereitstellen, was Sie wiederum bei Ihrer eigenen Entscheidungsfindung hilft, wenn Sie selber Anwendungen in Azure Red Hat OpenShift bereitstellen.

Erstellen des Clusters und Verbinden

Der Rest dieses Kapitels setzt voraus, dass Sie eine funktionierende Azure Red Hat OpenShift-Umgebung haben, mit der Sie arbeiten können. Es gibt keine speziellen Voraussetzungen für die Bewertungsanwendung und sie wird in einer leeren, neuen Azure Red Hat OpenShift-Umgebung bereitgestellt. Falls Sie bis jetzt noch keinen Cluster provisioniert haben, lesen Sie die folgenden Kapitel:

- *Kapitel 4, Vor der Provisionierung – Fragen zur Unternehmensarchitektur*
- *Kapitel 5, Provisionierung eines Azure Red Hat OpenShift-Clusters*

Der Abschnitt *Zugreifen auf den Cluster* in *Kapitel 5, Provisionierung eines Azure Red Hat OpenShift-Clusters,* ist eine hilfreiche Erinnerung daran, wie Sie auf einen Cluster zugreifen, den Sie möglicherweise zuvor provisioniert haben.

Anmelden bei der Webkonsole

Jeder Azure Red Hat OpenShift-Cluster hat eine DNS-Adresse für die OpenShift-Webkonsole. Sie können den Befehl `az aro list` verwenden, um die Cluster in Ihrer derzeitigen Azure-Subskription aufzulisten:

```
az aro list -o table
```

Die URL der Cluster-Webkonsole wird angezeigt. Öffnen Sie den Link in einem neuen Browser-Tab und melden Sie sich mit dem Nutzer `kubeadmin` oder einem anderen Benutzerkonto an, der Berechtigungen für das Erstellen von Projekten hat.

Nach der Anmeldung sollten Sie die Azure Red Hat OpenShift-Webkonsole sehen.

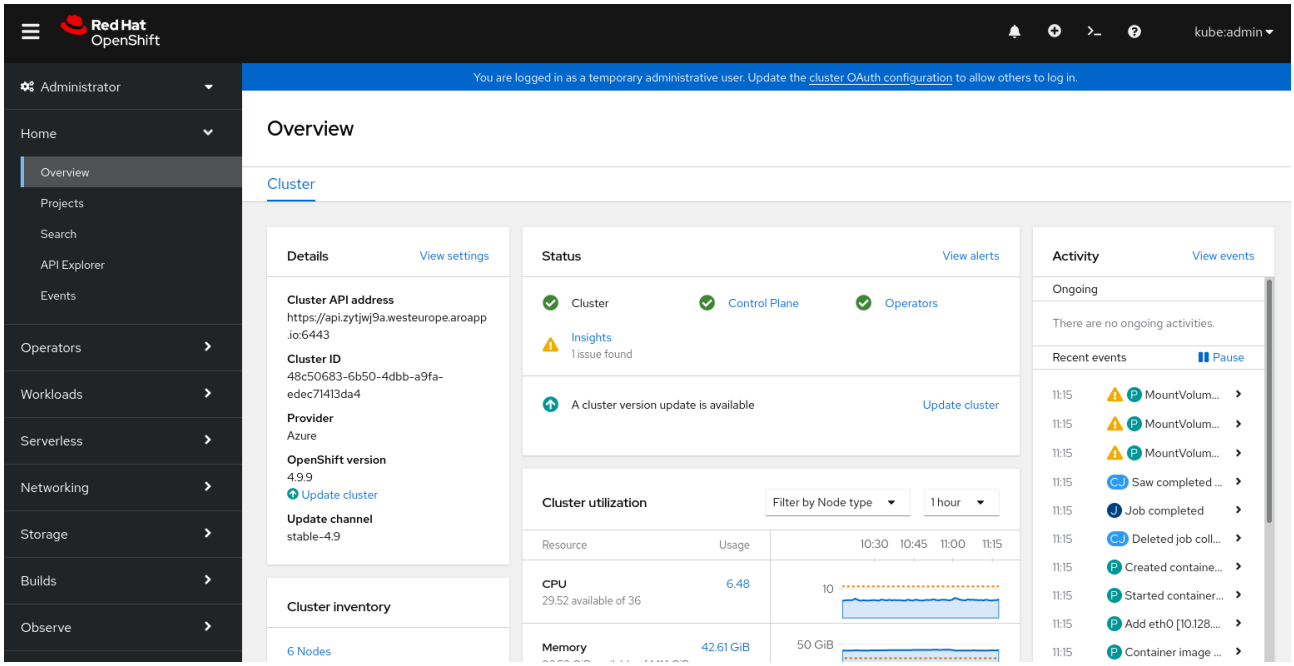


Abbildung 7.3: Azure Red Hat OpenShift-Webkonsole

Installieren des OpenShift-Clients

Öffnen Sie die Azure Cloud Shell oder verwenden Sie Ihr lokales Linux-Terminal und installieren Sie den OpenShift-Befehlszeilen-Client. Dies ist erforderlich, um über die Befehlszeile auf den Cluster zuzugreifen. Die Anweisungen hierfür sind wie folgt:

```
cd ~
curl https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-linux.tar.gz > openshift-client-linux.tar.gz

mkdir openshift

tar -zxvf openshift-client-linux.tar.gz -C openshift

echo 'export PATH=$PATH:~/openshift' >> ~/.bashrc && source ~/.bashrc
```

Dies sind die alternativen Download-Links für Windows und Mac:

- https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-windows.zip
- https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-mac.tar.gz

Sie sollten in der Lage sein, den Befehl `oc` in einem dieser heruntergeladenen Pakete auszuführen.

Abrufen des login-Befehls und -Tokens

Nachdem der Client installiert wurde, ist es notwendig, einen Token abzurufen, um sich beim Cluster anzumelden. Melden Sie sich in der OpenShift-Webkonsole an und klicken Sie auf den Benutzernamen oben rechts und klicken Sie dann auf **Copy Login Command**.

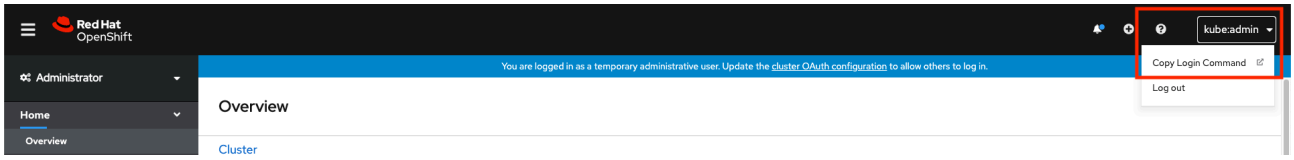


Abbildung 7.4: „Copy Login Command" zur Anmeldung beim Cluster

Fügen Sie den login-Befehl in Ihre Shell ein (lokales Linux-Terminal oder Azure Cloud Shell). Sie sollten sich nun mit dem Cluster verbinden können.

Abbildung 7.5: Verwendung des login-Befehls, um sich beim Cluster anzumelden

Sobald Sie sich verbunden haben, können wir mit dem Erstellen eines Projekts fortfahren.

Erstellen eines Projekts

Ein Projekt in OpenShift ist wie ein logischer Ordner, der die Anwendung für Bewertungen enthält. Alle Container und Anwendungen in Red Hat OpenShift müssen sich in einem Projekt befinden. Sie können Projekte verwenden, um Anwendungen oder Abteilungen voneinander abzugrenzen. Die nächsten Anweisungen erklären, wie Sie ein Projekt erstellen.

Sie können zwar ein Projekt in der Web-Schnittstelle erstellen, aber diese Anweisungen beziehen sich auf die Verwendung der Befehlszeile:

```
oc new-project workshop
```

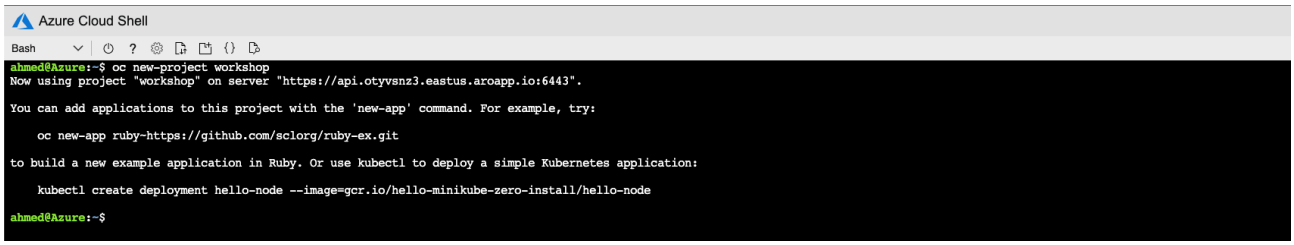


Abbildung 7.6: Erstellen eines neuen Workshops in Azure Cloud Shell

Sobald das Projekt erstellt wurde, können Sie zu diesem wechseln, indem Sie `oc project workshop` verwenden. Der nächste Schritt ist das Bereitstellen des ersten von drei Microservices, die zum Beginn dieses Kapitel vorgestellt wurden – MongoDB. Der Service wird im soeben erstellten Projekt bereitgestellt.

Ressourcen

- Azure Red Hat OpenShift-Dokumentation – Getting started with the CLI
- Azure Red Hat OpenShift documentation – Projects

Bereitstellen von MongoDB

Azure Red Hat OpenShift bietet ein Container-Image und eine Vorlage für eine einfache Erstellung eines neuen MongoDB-Datenbankservices. Die Vorlage bietet Parameterfelder zum Definieren aller obligatorischen Felder der Umgebung (Nutzer, Passwort, Datenbankname usw.) mit vordefinierten Standardwerten sowie automatisch generierten Passwortwerten. Sie definiert sowohl eine Deployment-Konfiguration als auch einen Service.

Die MongoDB-Instanz wird direkt als Container-Image von Docker Hub aus bereitgestellt. OpenShift ermöglicht es Ihnen, all dies über die Webkonsole durchzuführen. Wechseln Sie oben im Menü zur „Developer"-Ansicht, navigieren Sie zu Seite **Add** und wählen Sie **Container images** aus.

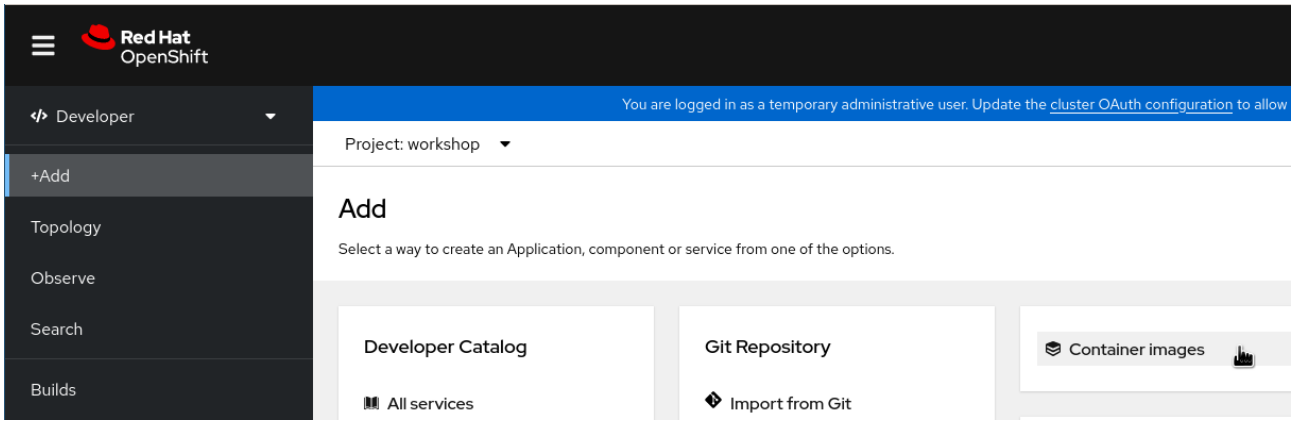


Abbildung 7.7: Hinzufügen eines Container Images

Sie werden zur Seite **Deploy Image** weitergeleitet. Füllen Sie das Formular wie folgt aus:

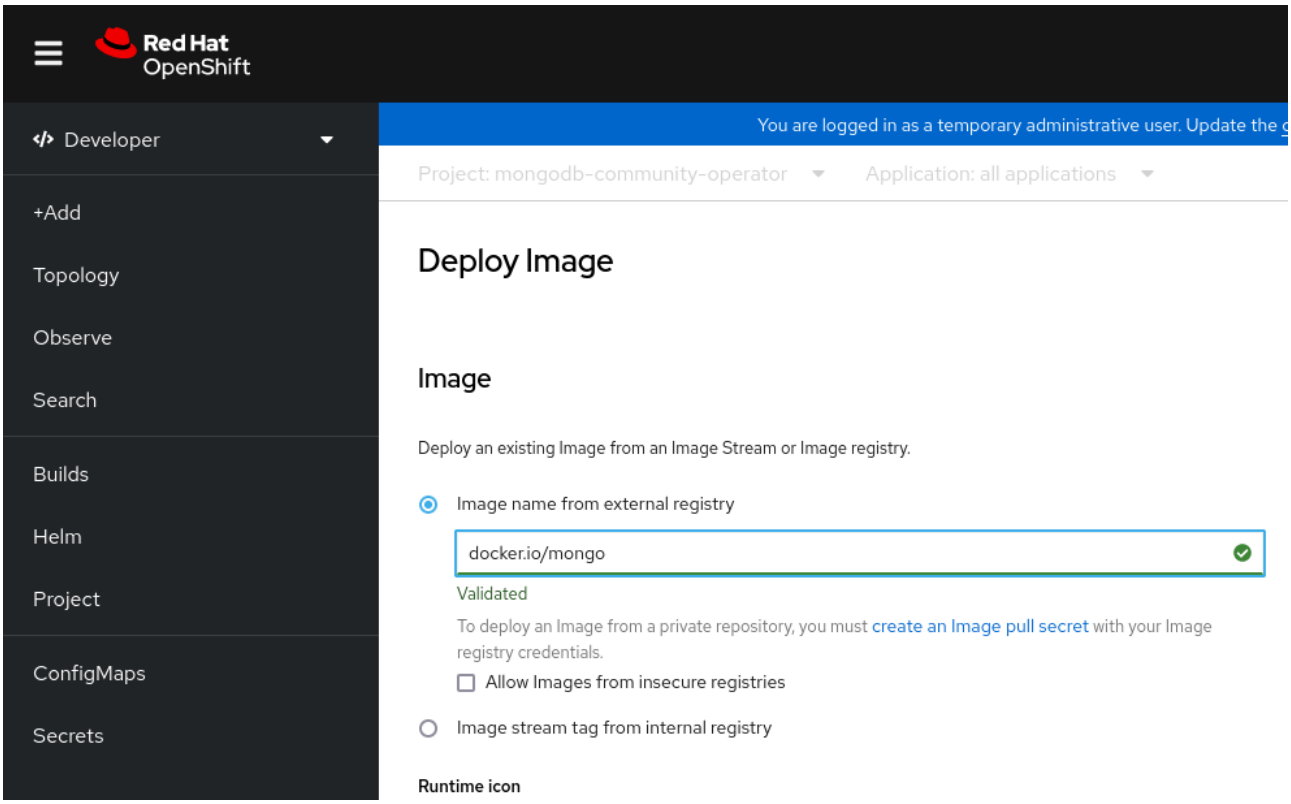


Abbildung 7.8: Ausfüllen des Formulars, um ein Image bereitzustellen

Stellen Sie sicher, die Werte wie folgt festzulegen:

Feld	Wert
Image name from external registry	`docker.io/mongo`
Runtime icon	`mongodb`
Application name	`ratings`
Name	`mongodb`
Create route to application	Nicht markiert – ein Routing gewährt den externen Zugang zur Datenbank, was bei dieser Anwendung nicht erforderlich ist
Resource type	Deployment

Wenn Sie am unteren Ende des Formulars angekommen sind, wählen Sie den Deployment-Link, um das Formular zu erweitern, sodass Sie die Umgebungsvariablen eingeben können.

Die folgenden Umgebungsvariablen dienen als Voreinstellung, um die Datenbank zu initialisieren, wenn sie das erste Mal hochfährt:

```
Name                    Wert
MONGO_INITDB_DATABASE   ratingsdb
```

Für die MongoDB-Datenbank werden kein Benutzername und kein Passwort festgelegt und die Authentifizierung ist deaktiviert – dies ist die Standardkonfiguration.

Überprüfen Sie die Umgebungsvariable erneut und klicken Sie dann auf die Schaltfläche **Create**, um fortzufahren und den MongoDB-Container bereitzustellen.

Die MongoDB-Instanz sollte eine kurze Zeit später im Projekt `workspace` ausgeführt werden. Sie können dieses Deployment anzeigen, indem Sie zur Ansicht **Topology** wechseln.

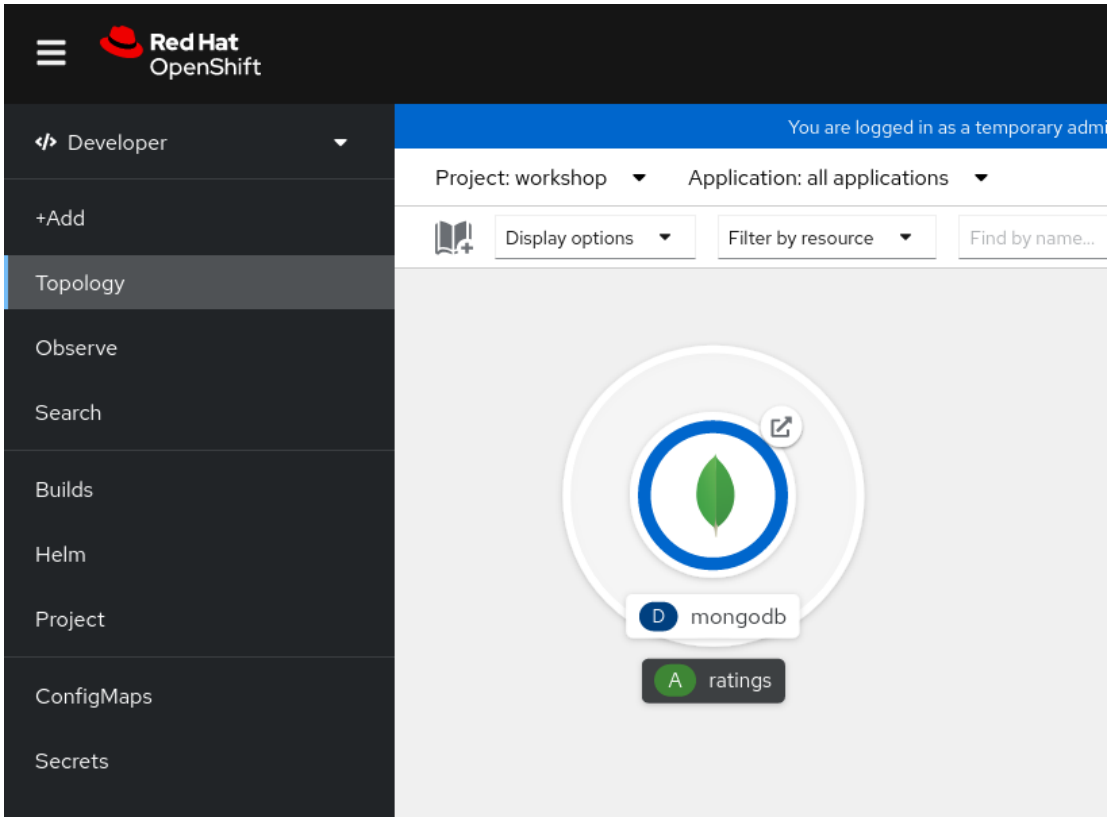


Abbildung 7.9: Die ausgeführte MongoDB-Instanz

Führen Sie den Befehl `oc get all` aus, um den Status der neuen Anwendung anzuzeigen und zu verifizieren, dass das Deployment der MongoDB-Vorlage erfolgreich war. Eine Beispielausgabe für `oc get all` sieht folgendermaßen aus;

```
user@host: oc get all
NAME                              READY    STATUS     RESTARTS    AGE
pod/mongo-6c6fcb45b8-8wpdm    1/1      Running    0           29s

NAME             TYPE         CLUSTER-IP       EXTERNAL-IP    PORT(S)      AGE
service/mongo    ClusterIP    172.30.88.119    <none>         27017/TCP    30s

NAME                       READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/mongo    1/1      1             1            30s

NAME                                    DESIRED    CURRENT    READY    AGE
replicaset.apps/mongo-6c6fcb45b8    1          1          1        30s

NAME                                        IMAGE REPOSITORY
TAGS      UPDATED
imagestream.image.openshift.io/mongo    image-registry.openshift-image-registry.svc:5000/workshop/mongo
latest    30 seconds ago
```

Wenn alles ordnungsgemäß funktioniert, sollte die Spalte `STATUS` anzeigen, dass der Container ausgeführt wird.

Abrufen des MongoDB-Service-Hostnamens

Sobald das Deployment fertig ist, müssen wir den Service finden, der dazu erstellt wurde, um uns zu ermöglichen, aus dem Cluster auf die Datenbank zuzugreifen. `svc` steht für `Services`:

```
user@host: oc get svc mongodb
NAME     TYPE         CLUSTER-IP       EXTERNAL-IP    PORT(S)      AGE
mongo    ClusterIP    172.30.88.119    <none>         27017/TCP    77s
```

Der Service ist über den folgenden DNS-Namen zugänglich: `mongodb.workshop.svc.cluster.local`, der das folgende Format aufweist: `[Servicename].[Projektname].svc.cluster.local`. Dieser wird nur innerhalb des Clusters aufgelöst.

Bereitstellen der Ratings API

Es ist nun an der Zeit, die zweite Anwendung bereitzustellen, nämlich die `rating-api`. Es handelt sich um eine Node.js-Anwendung, die eine Verbindung zu einer MongoDB-Instanz herstellt, um Elemente abzurufen und zu bewerten. Hier finden Sie einige Details, die Sie zum Bereitstellen der Anwendung benötigen:

- `rating-api` in GitHub
- Der Container macht Port `3000` zugänglich.
- Eine MongoDB-Verbindung ist mithilfe einer Umgebungsvariablen namens `MONGODB_URI` definiert.

Notieren Sie sich diese Details, da Sie sie in den nächsten Abschnitten brauchen werden.

Forken der Anwendung in Ihr eigenes GitHub-Repository

Um Änderungen vorzunehmen, wie etwa das Hinzufügen von CI/CD-Webhooks, benötigen Sie Ihre eigene Kopie des Codes der `ratings-api`. In Git wird dies ein „Fork" genannt. Sie können die Anwendung in Ihr persönliches GitHub-Repository forken. Gehen Sie zum oben genannten GitHub-Repository und klicken Sie auf die Schaltfläche **Fork**.

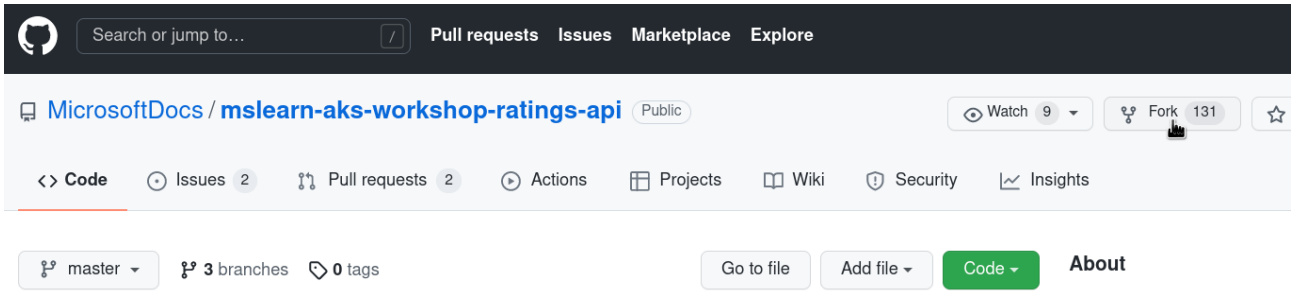


Abbildung 7.10: Forken der Anwendung im GitHub-Repository

Notieren Sie sich die neue Repository-Adresse. Sie werden sie in den folgenden Anweisungen benötigen.

Verwenden der OpenShift CLI zum Bereitstellen von rating-api

OpenShift kann Code direkt aus einem Git Repository bereitstellen, indem es sich den Inhalt anschaut und auf Basis des Inhalts ein „Builder Image" auswählt – Java, PHP, Perl, Python, oder Ähnliches. In diesem Fall ist `rating-api` eine JavaScript-Anwendung. Das Builder Image lädt die JavaScript-Abhängigkeiten mithilfe von npm herunter. Als Ergebnis haben Sie dann ein neues Container-Image, das in der internen OpenShift-Container-Registry gespeichert ist. Diese Build-Strategie wird **Source 2 Image (S2I)** genannt und wird im Glossar detaillierter beschrieben.

Sie können ein neues S2I-Build starten, indem Sie `oc new-app` verwenden:

```
user@host: oc new-app https://github.com/<your GitHub username>/mslearn-aks-workshop-ratings-api
--strategy=source --name=rating-api

--> Found image 0aea15f (3 weeks old) in image stream "openshift/nodejs" under tag "14-ubi8" for
"nodejs"

    Node.js 14
    ----------
     Node.js 14 available as container is a base platform for building and running various Node.
js 14 applications and frameworks. Node.js is a platform built on Chrome's JavaScript runtime
for easily building fast, scalable network applications. Node.js uses an event-driven, non-
blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time
applications that run across distributed devices.

    Tags: builder, nodejs, nodejs14

    * The source repository appears to match: nodejs
    * A source build using source code from https://github.com/MicrosoftDocs/mslearn-aks-
workshop-ratings-api will be created
      * The resulting image will be pushed to image stream tag "rating-api:latest"
      * Use 'oc start-build' to trigger a new build
```

Wechseln Sie zur Ansicht **Topology** innerhalb der Webkonsole und Sie sollten sehen, dass der Build-Vorgang der Anwendung startet und wie das Deployment nach ein paar Minuten fertiggestellt wird.

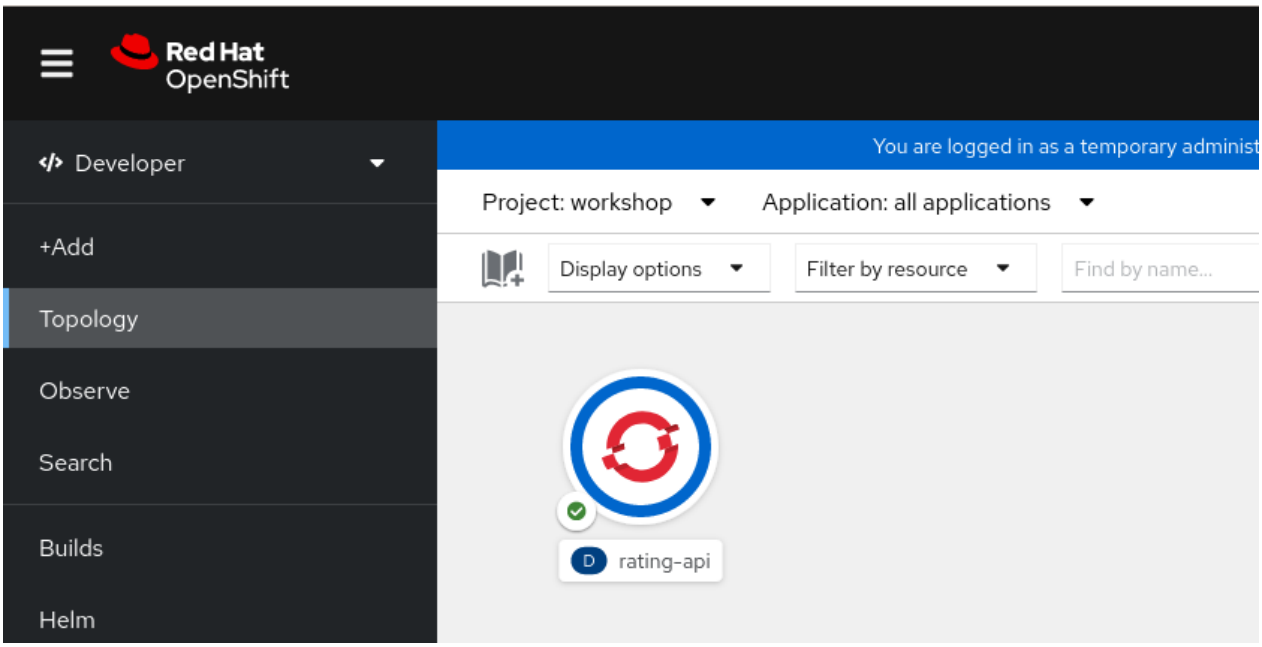


Abbildung 7.11: Die Ansicht „Topology“

Der Build und Startvorgang des Pods sollte nur eine oder zwei Minuten dauern.

Konfigurieren der erforderlichen Umgebungsvariablen

An diesem Punkt sind sowohl die Datenbank als auch die Ratings API bereitgestellt. Sie müssen für die Ratings API jedoch festlegen, wie sie sich mit der Datenbank verbindet. Die Konfiguration von containerbasierten Anwendungen wird meistens mithilfe von Umgebungsvariablen durchgeführt.

Klicken Sie auf das Deployment und bearbeiten Sie es. Erstellen Sie die folgende Umgebungsvariable:

```
Name          Wert
MONGODB_URI   mongodb://mongodb.workshop.svc.cluster.local:27017/ratingsdb
```

Sobald Sie die neue Umgebungsvariable gespeichert haben, löst sie ein Redeployment des `ratings-api`-Services aus und die neue Umgebungsvariable wird verwendet.

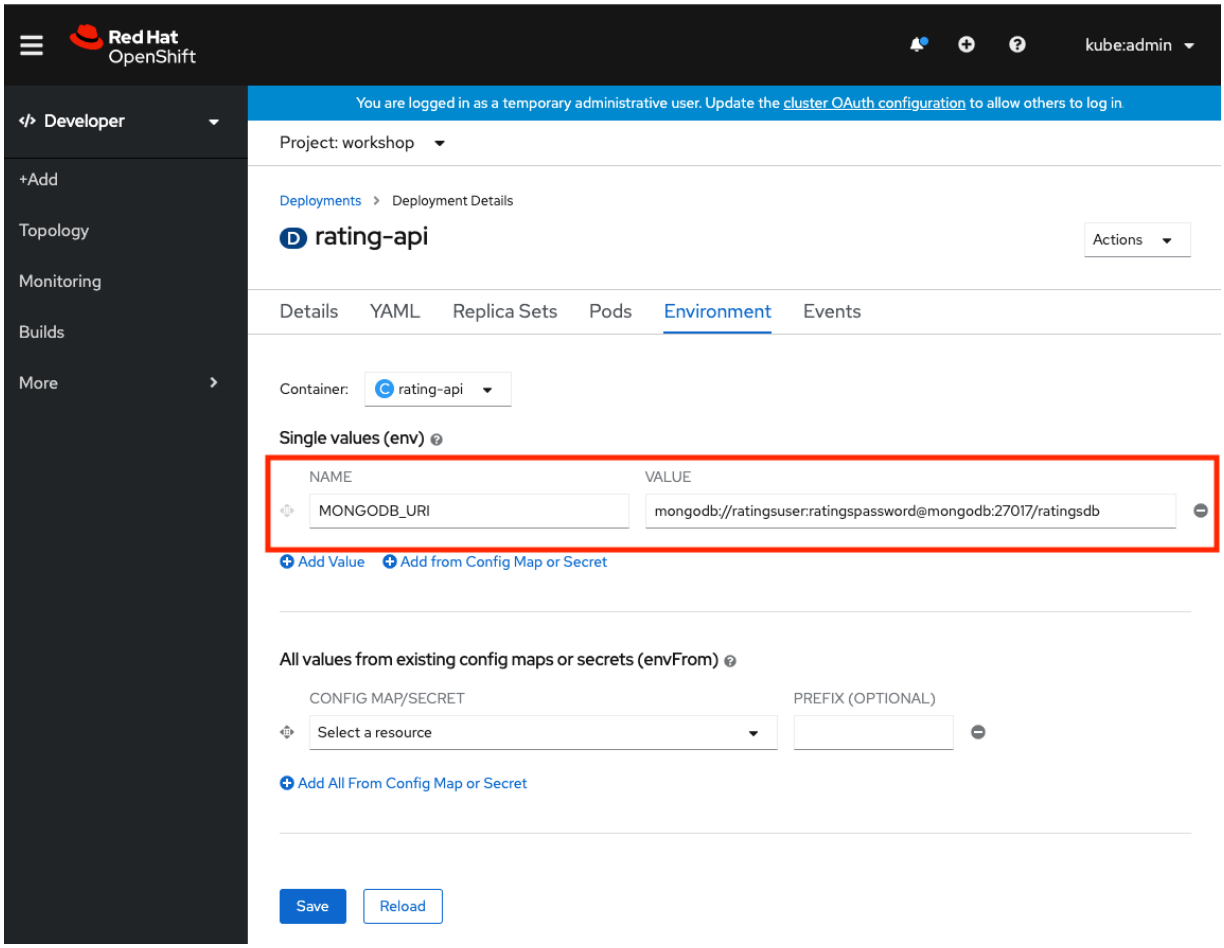


Abbildung 7.12: Festlegen der MONGODB_URI-Umgebungsvariable über die Webkonsole

Die Umgebungsvariable kann auch folgendermaßen über die Befehlszeile festgelegt werden:

```
oc set env deploy/rating-api MONGODB_URI=mongodb://mongodb.workshop.svc.cluster.local:27017/ratingsdb
```

OpenShift wird unabhängig von der verwendeten Methode den Container erneut starten müssen, um die neuen Umgebungsvariablen verwenden zu können.

Verifizieren der Ausführung des Services

Wenn Sie zu den Protokollen des `rating-api`-Deployments navigieren, sollten Sie eine Protokollnachricht sehen, die bestätigt, dass sich der Code erfolgreich mit MongoDB verbinden kann. Um dies zu tun, klicken Sie auf dem Detailbildschirm des Deployments auf den Tab **Pods** und anschließend auf einen der Pods.

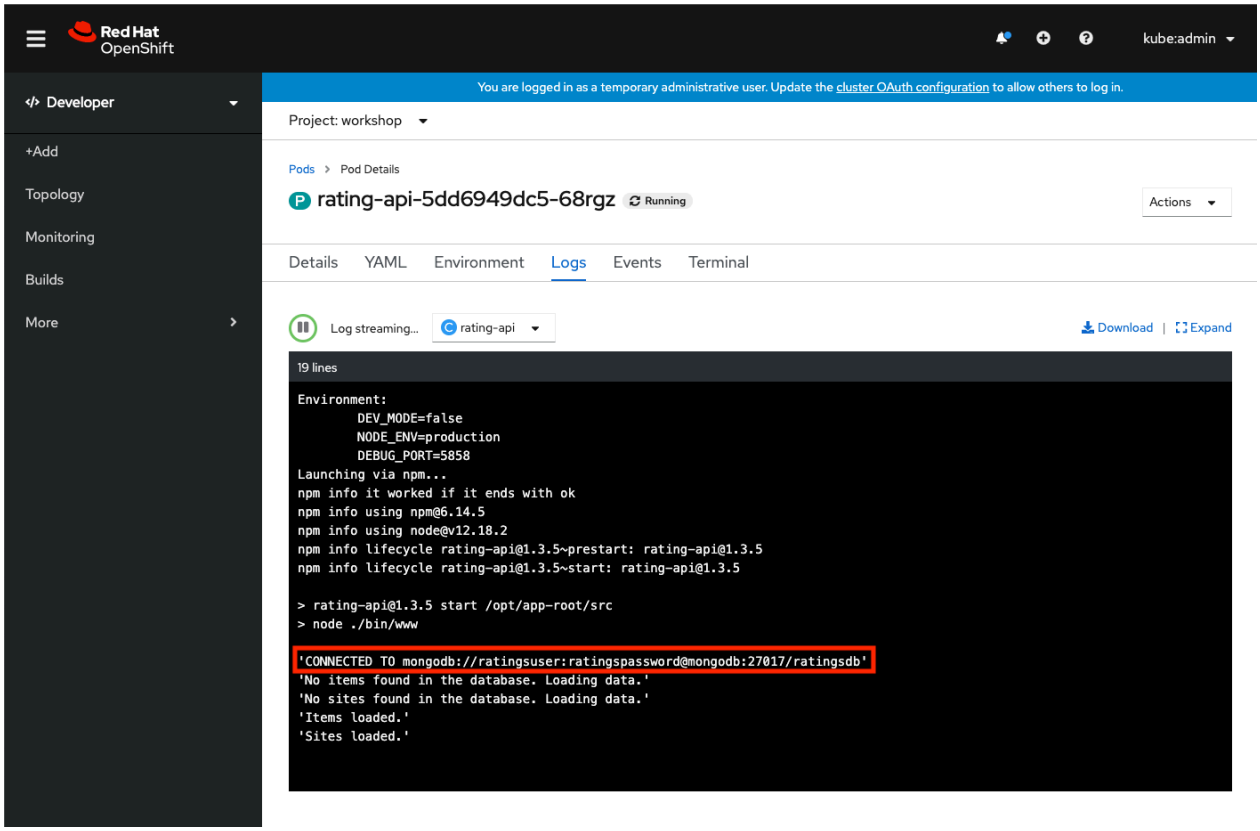


Abbildung 7.13: Protokollnachricht, die bestätigt, dass sich der Code erfolgreich mit MongoDB verbinden kann

Ändern des Service-Ports der rating-api

OpenShift erstellt einen Service mit dem Port `8080`. Jedoch wird der Service aufgrund einer Library-Aktualisierung über Port `3000` ausgeführt. Es ist notwendig, den Standardservice zu bearbeiten.

Navigieren Sie zum Menü unter **Networking** → **Services**, wählen Sie den Service `rating-api` aus der Liste und bearbeiten Sie ihn wie folgt (ersetzen Sie einfach `8080` mit `3000`):

```
ports:
  - name: 3000-tcp
    protocol: TCP
    port: 3000
    targetPort: 3000
```

Starten Sie den Service neu, damit er den neuen Port verwendet:

```
user@host: oc rollout restart deploy/rating-api
```

Jetzt wird der Service an den richtigen Port gebunden – `3000` anstatt `8080`.

Abrufen des Service-Hostnamens der rating-api

Wir müssen nun validieren, dass es einen `rating-api`-Service gibt, da er im nächsten Abschnitt verwendet wird, wenn die Anwendung `rating-web` bereitgestellt wird:

```
oc get service rating-api
```

Der Service ist über den DNS-Namen `rating-api.workshop.svc.cluster.local:3000` über Port `3000` zugänglich, der das folgende Format aufweist: `[Servicename].[Projektname].svc.cluster.local`. Dieser wird nur innerhalb des Clusters aufgelöst.

Bereitstellen des Bewertungs-Frontends

Bei `rating-web` handelt es sich um eine Node.js-Anwendung, die eine Verbindung zu `rating-api` herstellt. Nachfolgend finden Sie einige Details, die Sie zum Bereitstellen der Anwendung benötigen:

- `rating-web` in GitHub
- Der Container macht Port `8080` zugänglich.
- Die Web-App stellt über das interne Cluster-DNS mittels eines Proxys über eine Umgebungsvariable namens API eine Verbindung zur `API` her.

Verwenden der OpenShift CLI zum Bereitstellen von rating-web

Diese Anwendung kann genauso, wie die Anwendung `rating-api`, mithilfe von `oc new-app` über S2I bereitgestellt werden. Diesmal wird die App jedoch unter Verwendung einer Dockerfile, die sich bereits im Git-Repository befindet, mit einer Dockerfile-Strategie erstellt. Deshalb sollten Sie kein `--strategy`-Argument angeben:

```
user@host: oc new-app https://github.com/<your GitHub username>/mslearn-aks-workshop-ratings-web
--name rating-web

--> Found container image e1495e4 (2 years old) from Docker Hub for "node:13.5-alpine"

    * An image stream tag will be created as "node:13.5-alpine" that will track the source image
    * A Docker build using source code from https://github.com/MicrosoftDocs/mslearn-aks-
workshop-ratings-web will be created
      * The resulting image will be pushed to image stream tag "rating-web:latest"
      * Every time "node:13.5-alpine" changes a new build will be triggered

--> Creating resources ...
    imagestream.image.openshift.io "node" created
    imagestream.image.openshift.io "rating-web" created
    buildconfig.build.openshift.io "rating-web" created
    deployment.apps "rating-web" created
    service "rating-web" created
--> Success
```

Der Build der Abhängigkeiten in einem Container dauert eine Weile. Warten Sie zwei bis drei Minuten bis zur Fertigstellung und kehren Sie zur Ansicht **Topology** zurück, in der Sie sehen können, wie der Service online gebracht wird.

Konfigurieren der erforderlichen Umgebungsvariablen

Erstellen Sie die `API`-Umgebungsvariablen für die `rating-web`-Deployment-Konfiguration. Der Wert dieser Variable wird der Hostname/Port des `rating-api`-Services sein.

Sie können die Umgebungsvariable über die OpenShift-Befehlszeile festlegen, anstatt die Azure Red Hat OpenShift-Webkonsole zu verwenden:

```
oc set env deploy rating-web API=http://rating-api:3000
```

Zugänglichmachen des rating-web-Services mithilfe von Routing

Das Zugänglichmachen des Services bedeutet, dass eine öffentlich zugängliche URL für Nutzer bereitgestellt wird, mit der sie auf den Service zugreifen können. Wenn der Service nicht zugänglich gemacht wird, kann auf ihn nur innerhalb des Clusters zugegriffen werden:

```
user@host: oc expose svc/rating-web
route.route.openshift.io/rating-web exposed
```

Rufen Sie abschließend die URL des zugänglich gemachten Services ab:

```
user@host: oc get route rating-web
NAME          HOST/PORT                                                PATH    SERVICES      PORT
TERMINATION    WILDCARD
rating-web    rating-web-workshop.apps.zytjwj9a.westeurope.aroapp.io        rating-web    8080-tcp
None
```

Beachten Sie, dass der **vollqualifizierte Domainname (FQDN)** standardmäßig aus dem Anwendungsnamen und dem Projektnamen besteht. Der Rest des FQDN, die Subdomain, ist die Azure Red Hat OpenShift-Cluster-spezifische Apps-Subdomain.

Testen des Services

Öffnen Sie den Hostnamen in Ihrem Browser. Die Bewertungs-App-Seite sollte sich öffnen. Testen Sie die Seite, indem Sie ein paar Bewertungen einreichen und sich das Leaderboard anschauen.

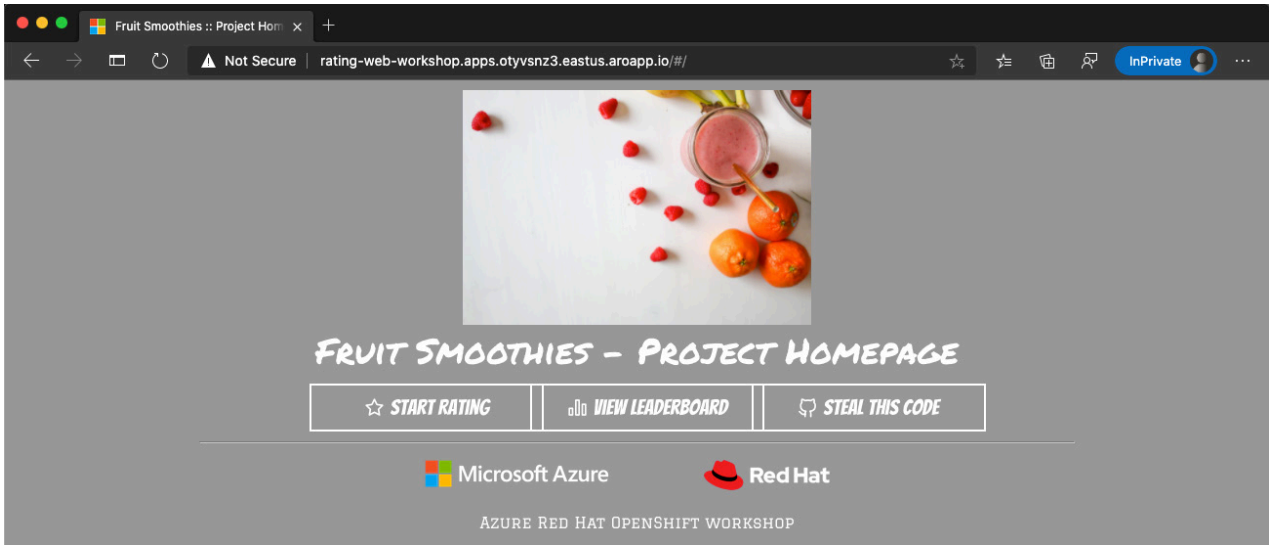


Abbildung 7.14: Testen des Bewertungs-App-Services

Einrichten des GitHub-Webhooks

Um S2I-Builds auszulösen, sobald Sie Code in Ihr GitHub-Repository pushen, müssen Sie den GitHub-Webhook einrichten:

1. Rufen Sie das Trigger-Secret des GitHub-Webhooks ab. Sie werden dieses Secret in der URL des GitHub-Webhooks benötigen:

```
user@host: oc get bc/rating-web -o=jsonpath='{.spec.triggers..github.secret}'
3ffcc8d5-a243
```

   Notieren Sie sich den Secret-Schlüssel, da Sie ihn in ein paar Schritten brauchen werden.

2. Rufen Sie die Trigger-URL des GitHub-Webhooks von der Build-Konfiguration ab:

```
user@host: oc describe bc/rating-web
...
Webhook GitHub:
        URL:    https://api.quwhfg7o.westeurope.aroapp.io:6443/apis/build.openshift.io/v1/
namespaces/workshop/buildconfigs/rating-web/webhooks/<secret>/github
...
```

3. Ersetzen Sie den Platzhalter `<secret>` mit dem Secret, das Sie im ersten Schritt abgerufen haben. In diesem Fall ist das Secret `3ffcc8d5-a243`. Somit sieht die endgültige URL folgendermaßen aus:

```
https://api.quwhfg7o.westeurope.aroapp.io:6443/apis/build.openshift.io/v1/namespaces/workshop/
buildconfigs/rating-web/webhooks/3ffcc8d5-a243/github
```

   Sie werden diese URL verwenden, um den Webhook für Ihr GitHub-Repository einzurichten.

4. Navigieren Sie zu Ihrem GitHub-Repository. Wählen Sie **Add Webhook** unter **Settings** → **Webhooks**.
5. Fügen Sie im Feld **Payload URL** Ihre GitHub-URL ein, in der Sie den Wert `<secret>` mit Ihrem Secret ersetzt haben.
6. Ändern Sie die Standardangabe im Feld **Content type** von `application/x-www-form-urlencoded` zu `application/json`.

7. Klicken Sie auf **Add webhook**.

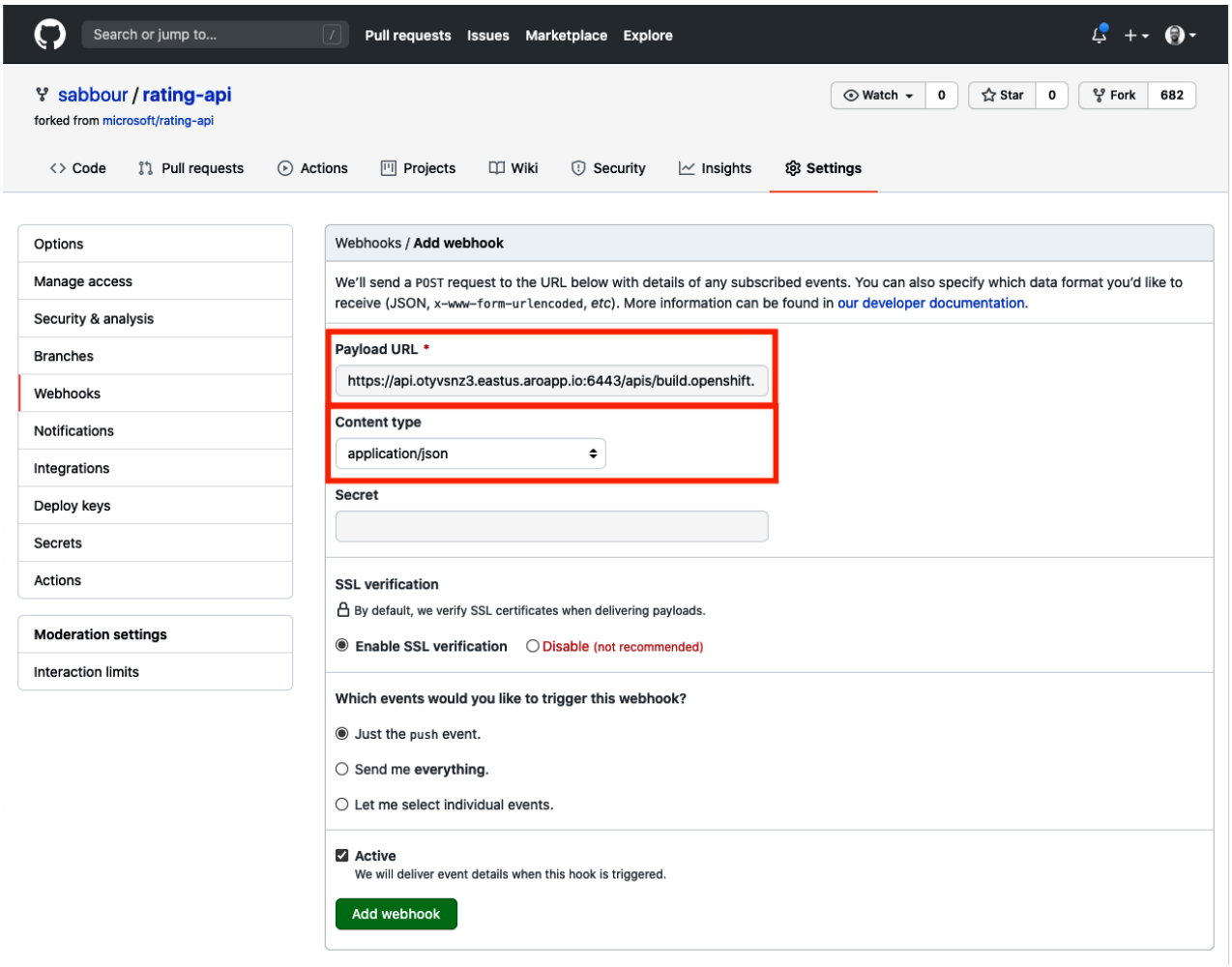


Abbildung 7.15: Hinzufügen eines Webhooks

Sie sollten eine Nachricht von GitHub sehen, die Sie darüber informiert, dass Ihr Webhook erfolgreich konfiguriert wurde.

Wenn Sie nun eine Änderung in Ihr GitHub-Repository pushen, wird jeweils immer ein neues Build gestartet und nachdem dieses fertiggestellt wurde, wird ein neues Deployment gestartet.

Vornehmen einer Änderung an der Website-App und Anzeigen des rollenden Updates

Gehen Sie zur Datei `https://github.com/<Ihr GitHub-Benutzername>/rating-web/blob/master/src/App.vue` in Ihrem GitHub-Repository.

Bearbeiten Sie die Datei, indem Sie die Zeile `background-color: #999;` zu `background-color: #0071c5` ändern.

Schreiben Sie die Änderungen an der Datei in den `master`-Zweig.

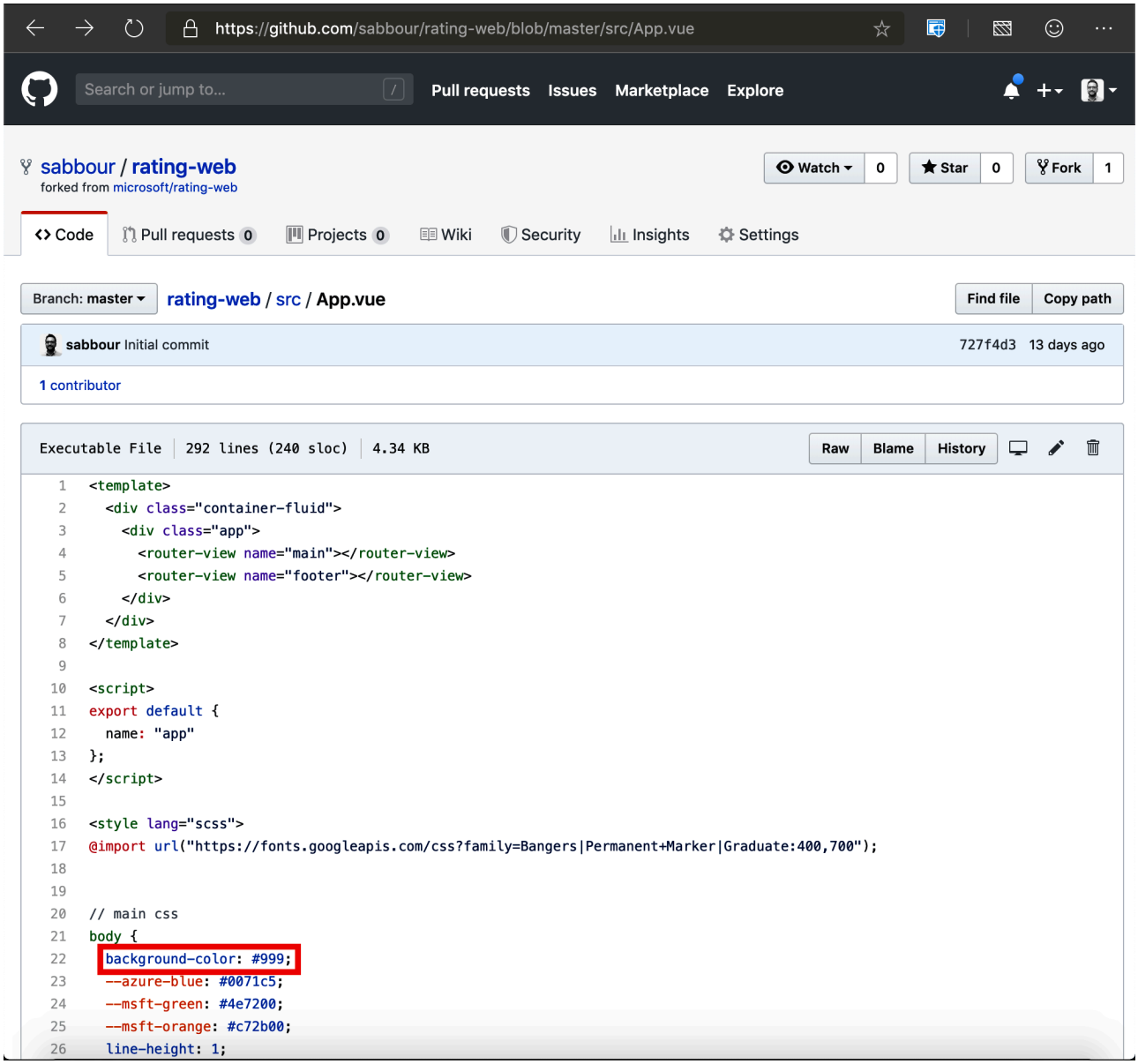


Abbildung 7.16: Schreiben der Änderungen an der Datei in den master-Zweig

Wechseln Sie direkt zum Tab **Builds** in der OpenShift-Webkonsole. Sie werden sehen, dass ein neuer Build in der Warteschlange ist, der durch den Push ausgelöst wurde. Wenn der Build fertiggestellt ist, wird ein neues Deployment ausgelöst und die Farbe der Website aktualisiert.

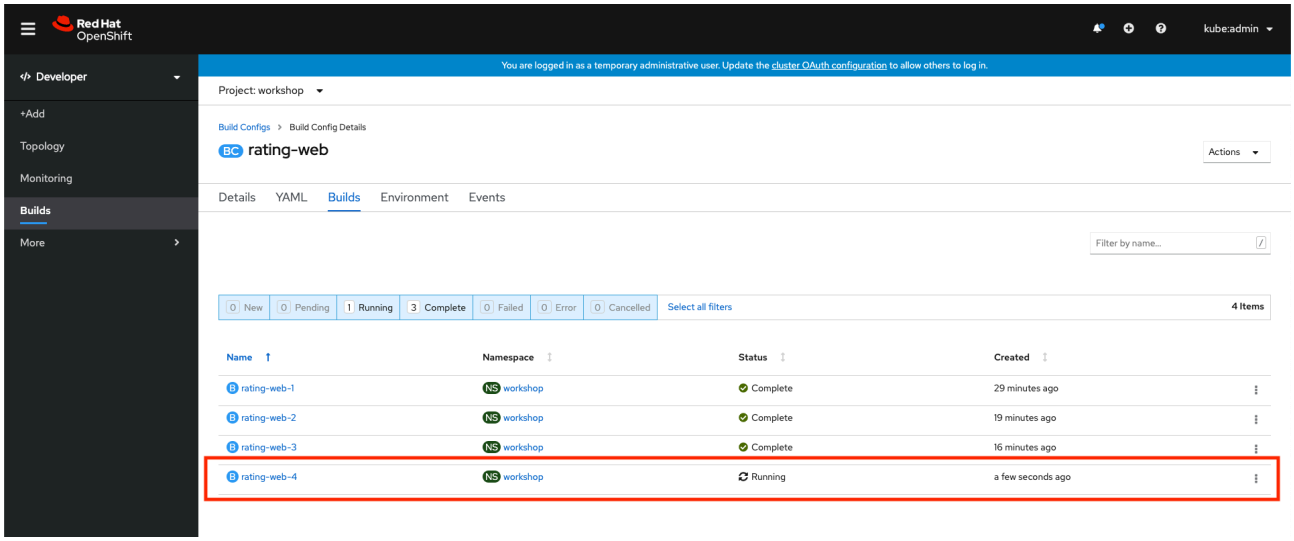


Abbildung 7.17: Der Tab „Builds“ zeigt die Ausführung eines neuen Builds an

Kehren Sie nun zur Seite `ratings-web` zurück. Wenn der Vorgang erfolgreich war, werden Sie sehen, dass sich die Farbe der Website geändert hat.

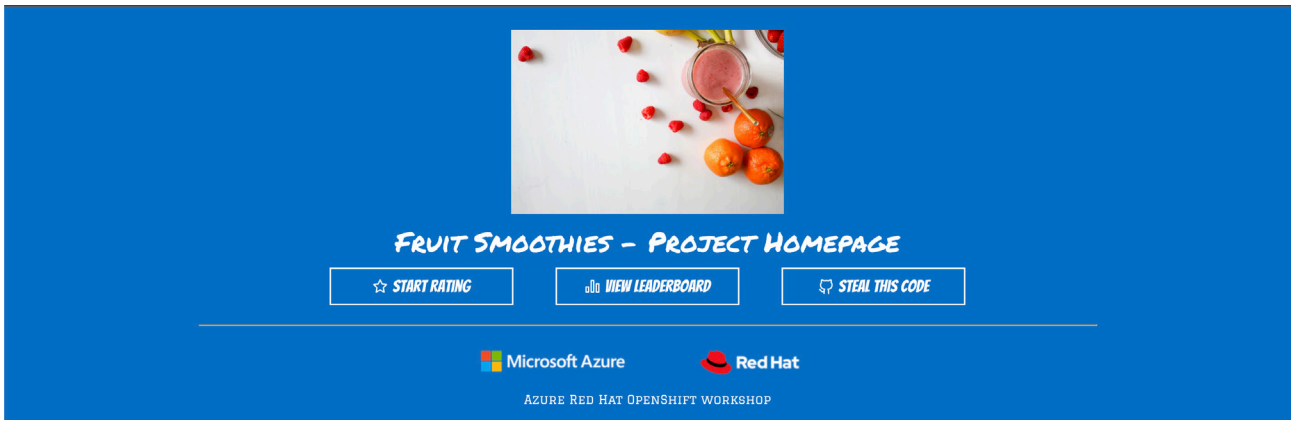


Abbildung 7.18: Die Homepage von „Fruit Smoothies“ mit der neuen Farbe

Zusammenfassung

In diesem Kapitel haben wir eine simple Anwendung bereitgestellt, die aus drei kleineren Microservice-Anwendungen besteht – einer MongoDB-Datenbank, der `ratings-api` und dem Code `ratings-web`. Die Anwendung hat zwar nur wenig Ähnlichkeit mit einer Produktions-Anwendung, jedoch dient sie als kurze Veranschaulichung der Konzepte im Zusammenhang mit dem Bereitstellen von Anwendungen in Azure Red Hat OpenShift. Sie können die Anweisungen als Grundlage für das Bereitstellen Ihrer eigenen Anwendungen verwenden.

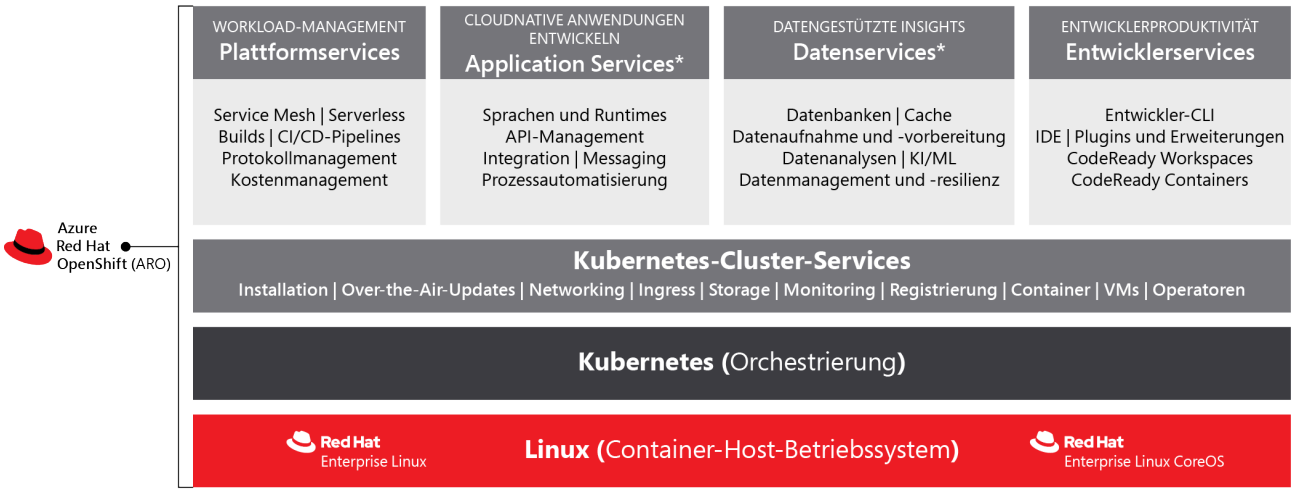
Red Hat OpenShift unterstützt außerdem das Bereitstellen von Anwendungen über OperatorHub, Helm Charts und über externe CI/CD-Systeme wie Azure DevOps. Die beste Deployment-Strategie für Ihre Organisation hängt von den Tools und Technologien ab, die Sie intern verwenden.

Im nächsten Kapitel schauen wir uns einige Aspekte des Mehrwerts von Azure Red Hat OpenShift an, der OpenShift als Kubernetes-Lösung von anderen Anwendungsplattformen abhebt. Wir gehen auf die Features und Funktionen ein, die für die komplexen Anforderungen von Unternehmen entwickelt wurden.

Kapitel 8

Erkunden der Anwendungsplattform

In den vorangehenden Kapiteln haben wir erfahren, wie Red Hat OpenShift eine Vielzahl von Services anbietet, die Kubernetes als Grundlage haben. Die Kombination dieser Services bildet eine richtige Anwendungsplattform. Die Services können in fünf Kategorien aufgeteilt werden: Plattformservices, Anwendungsservices, Datenservices, Entwicklerservices und Kubernetes-Cluster-Services.



*Red Hat OpenShift® enthält unterstützte Runtimes für beliebte Programmiersprachen/Frameworks/Datenbanken.
Die aufgeführten zusätzlichen Funktionen stammen aus den Portfolios von Red Hat Application und Data Services.

Abbildung 8.1: Die in Azure Red Hat OpenShift enthaltenen Services

Die in **OpenShift Platform Plus—Multicluster Management**, **Cluster Security** und **Global Registry** gruppierten Komponenten sind zusätzliche Produkte, für die eine Subskription erforderlich ist. Sie sind mit Azure Red Hat OpenShift kompatibel, aber nicht im Azure Red Hat OpenShift-Angebot enthalten.

Die folgenden Abschnitte beinhalten einige der wichtigsten Vorteile, die diese Services bieten, und Links weiteren Informationen.

Cluster-Services – integrierte Container-Registry

Red Hat OpenShift stellt eine integrierte, interne Container-Registry nach dem Bereitstellen des Clusters zur Verfügung. Diese Registry wird sowohl für Cluster-interne Services wie Cluster-Operatoren verwendet als auch als Standard-Anwendungscontainer für Kunden. Dies ist eine Standard-Registry und erfordert keine zusätzliche Einrichtung. Sie wird sogar von einem Infrastruktur-Operator gewartet.

Alle Kunden, die einen Kubernetes-Containerservice bereitstellen, müssen meistens zusätzlich ihre eigene Container-Registry bereitstellen, um ihre Container-Images privat zu halten. Die damit verbundene Day-2-Installation und -Einrichtung ist mit OpenShift nicht notwendig, da bereits die integrierte Container-Registry im Cluster zur Verfügung steht. Dies ist ein Beispiel einer einfachen und zeitsparenden Funktion von OpenShift.

Integrated OpenShift Container Platform Registry overview

Ein Cluster-Administrator entscheiden sich oft dafür, diese Container-Registry extern zugänglich zu machen, sodass Nutzer außerhalb von OpenShift Container-Images in die Registry zu pushen. Dies wird in Azure Red Hat OpenShift vollständig unterstützt und die Dokumentation zur Vorgehensweise finden Sie in der standardmäßigen OpenShift-Dokumentation zum Zugänglichmachen der Registry.

Plattformservices – OpenShift Pipelines

Kunden von Red Hat OpenShift haben verschiedene Möglichkeiten zur Erstellung ihrer Anwendungen und viele bekannten CI/CD-Tools wie Jenkins, CircleCI und GitHub Actions verfügen über Plugins, die OpenShift unterstützen. OpenShift bietet jedoch auch über den OpenShift Pipelines-Operator Plattformfunktionen zur Erstellung von Anwendungen mit cloudnativen Container-Pipelines an.

OpenShift Pipelines basiert auf einem Community-Projekt mit dem Namen Tekton. Jede Phase der Pipeline, wie etwa Code-Pulls aus einem Git Repository, das Ausführen eines Java-Compilers oder das Zusammenstellen eines RPM-Pakets, wird innerhalb eines Containers ausgeführt. Dies bedeutet, dass Entwickler und Operatoren komplexe und ausgeklügelte Pipelines erstellen können, indem sie sich die umfassenden Softwareentwicklungsvorteile von Containern zu Nutzen machen.

Die Installation von OpenShift Pipelines ist über OperatorHub möglich. Navigieren Sie einfach zu **OperatorHub** und wählen Sie den Operator aus, um die Installation zu starten. Es wird keine Konfiguration benötigt und die Operator-Installation dauert normalerweise weniger als eine Minute.

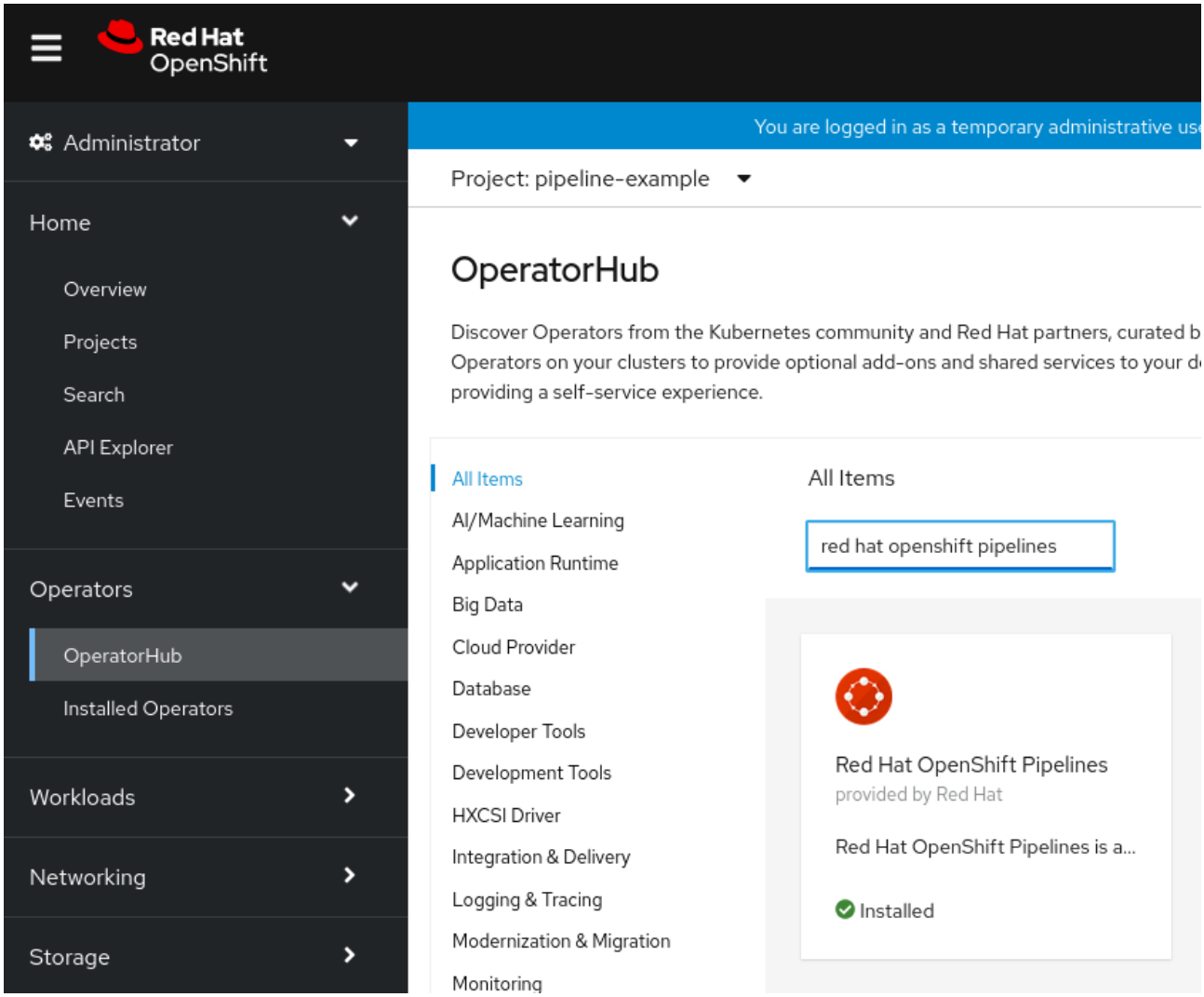


Abbildung 8.2: Installation von OpenShift Pipelines über OperatorHub

Sobald der OpenShift Pipelines-Operator installiert ist, wird der neue Abschnitt **Pipelines** in der Seitenleiste und eine neue Option zum Hinzufügen von Pipelines zu Katalogelementen angezeigt (wie etwa beim Erstellen des Node.js-Beispiels).

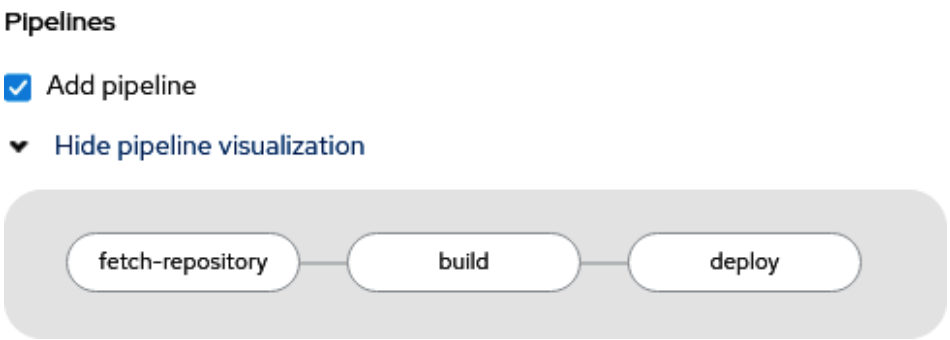


Abbildung 8.3: Hinzufügen von Pipelines

Wenn Sie OpenShift Pipelines verwenden, ist es möglich, den visuellen Builder zu verwenden, um komplexe, verzweigte Pipelines einzurichten und zu erstellen. Hier ist ein Beispiel-Screenshot einer etwas komplexeren Pipeline:

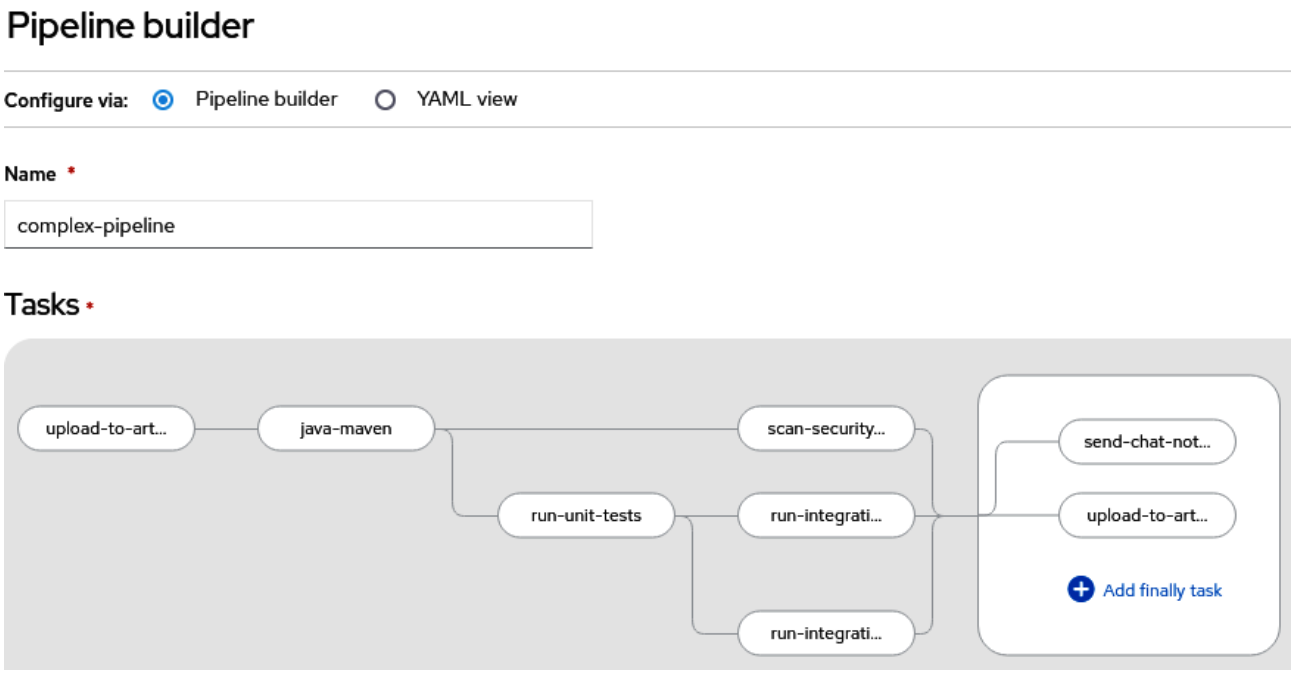


Abbildung 8.4: Eine komplexe Pipeline

Viele Organisationen verwenden eine Vielzahl von Tools und Technologien zur Softwareentwicklung. OpenShift Pipelines macht es einfach, den Build direkt in OpenShift zu integrieren, indem es sich alle Vorteile von Containern zu Nutze macht. Somit wird unabhängig von der zugrunde liegenden Infrastruktur eine konsistente, benutzerfreundliche CI/CD-Lösung mit einem sehr geringen Einrichtungsaufwand ermöglicht.

Weitere Informationen

- Understanding OpenShift Pipelines in the OpenShift
- Community-Seite von Tekton

Plattformservices – OpenShift Serverless

Es ist ein verbreiteter Irrtum, dass die Container-Technologie lediglich für Services mit langer Ausführungszeit nützlich ist. Tatsächlich werden viele kurzlebige Aufgaben und Serverless-Funktionen mit kurzlebigen Containern ausgeführt. Aufgrund der Vorteile von Containern bezüglich Startgeschwindigkeit, Konsistenz und des einfachen Herunterfahrens, eignen sie sich gut für Serverless-Workloads. Natürlich benötigen Serverless-Services zugrunde liegende Server zur Code-Ausführung. Daher wird Serverless gelegentlich als „Function as a Service" bezeichnet.

Red Hat OpenShift ermöglicht Serverless- bzw. „Function as a Service"-Workloads über den OpenShift Serverless-Operator. Dieser Operator basiert auf dem bekannten Open-Source-Projekt Knative.

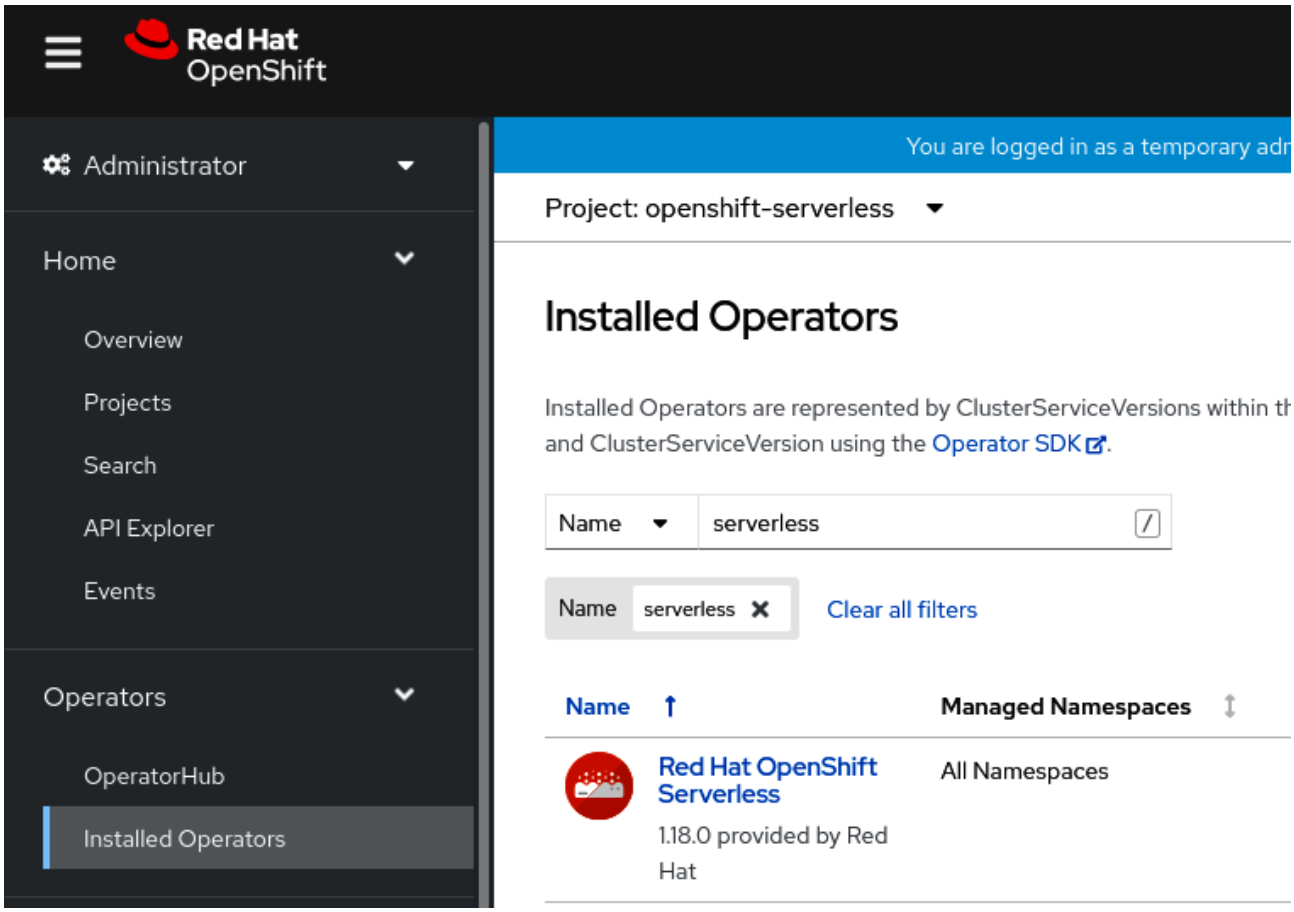


Abbildung 8.5: Ansicht „Installed Operators"

Nach der Bereitstellung im Cluster (was in der Regel wiederum ein oder zwei Minuten dauert) müssen für den Operator einige Einstellungen vorgenommen werden. Es gibt zwei **Custom Resource Definitions (CRDs)** die besonders beachtet werden müssen: **Knative Serving** und **Knative Eventing**.

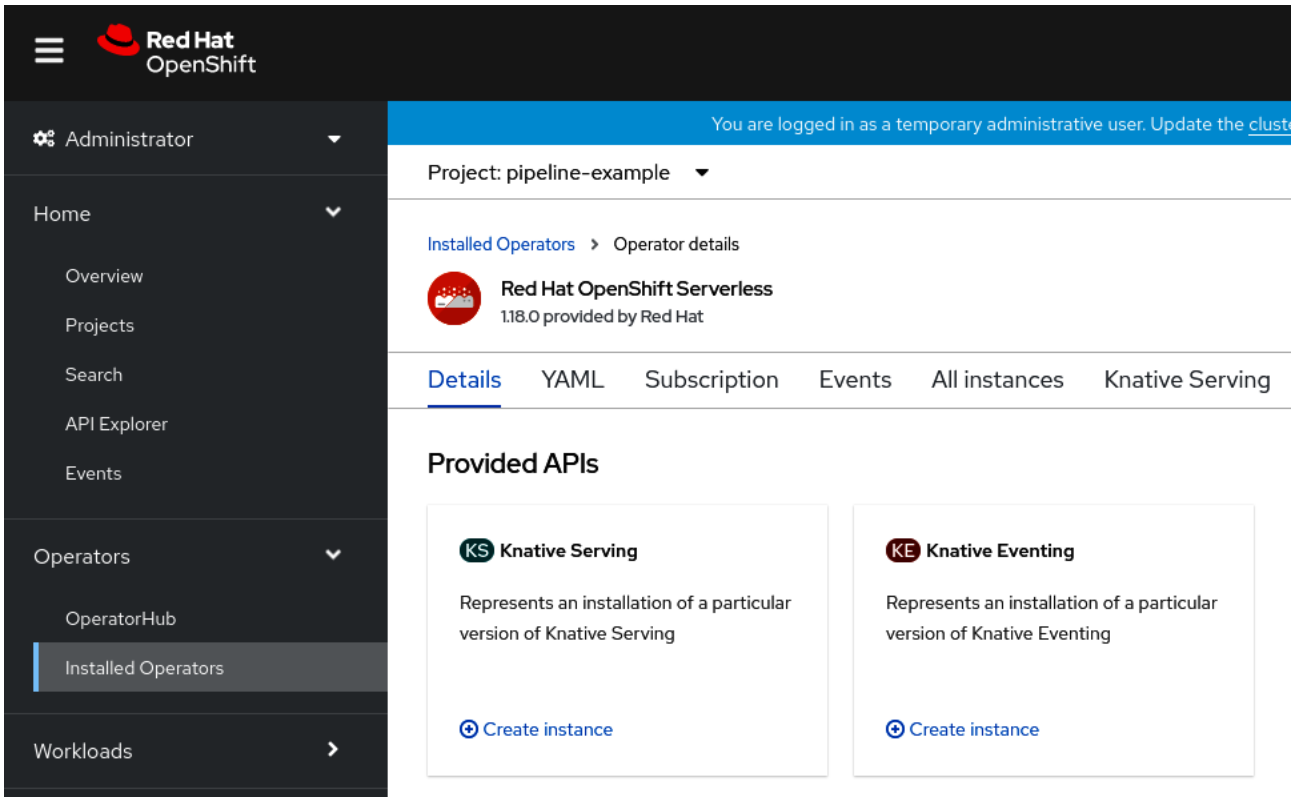


Abbildung 8.6: „Knative Serving“ und „Knative Eventing“ im Menü „Operators“

- **Knative Serving** vereinfacht das Deployment von Anwendungen, skaliert auf Basis des eingehenden Datenverkehrs dynamisch und unterstützt benutzerdefinierte Rollout-Strategien mit Datenverkehrsaufteilung, z. B. eine Funktion, die ein Image in einen Storage Bucket hochlädt und diesen Upload in einer NoSQL-Datenbank protokolliert.
- **Knative Eventing** ermöglicht die „späte Bindung“ von Event-Quellen zur Laufzeit der Anwendung (im Gegensatz zur Build-Zeit), beispielsweise eine Anwendung, die auf neue Image-Uploads in einen Storage Bucket antwortet – diese Anwendung muss zur Build- oder Event-Breitstellungszeit keine Kenntnis von dem Storage Bucket haben, aber mit Knative Eventing kann diese Event-Quelle einfach zur Laufzeit an die Anwendung „gebunden“ werden.

Die beiden folgenden Abschnitte enthalten Beispiele für jede Art dieser Serverless-Anwendungen unter OpenShift.

Plattformservices – OpenShift Serverless – Beispiel für Knative Serving

Im Folgenden wird gezeigt, wie Knative Serving die Autoskalierung einer Anwendung ermöglicht, insbesondere die Skalierung auf null, wenn es keine Anfragen gibt. Eine sehr einfache Anwendung, die wir verwenden können, ist eine Webseite mit ASCII-Bildern von Tieren:

- php-ascii-pets GitHub repository

Das Hinzufügen dieses Repositorys von Git aus ist ganz einfach – Red Hat OpenShift erkennt automatisch ein kompatibles Builder Image von PHP.

OpenShift erkennt automatisch, wie dieses Projekt erzeugt wird.

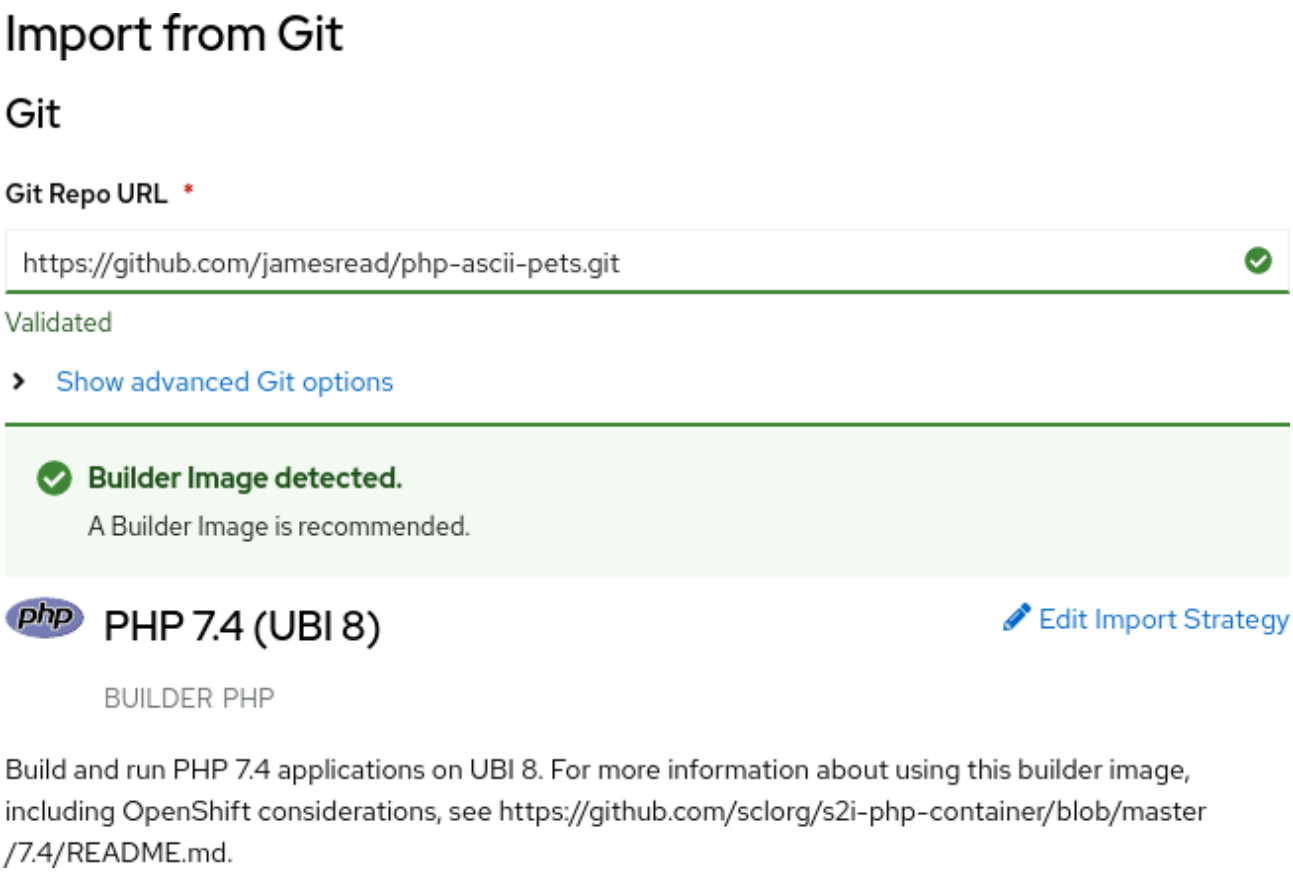


Abbildung 8.7: Ein Builder Image wird automatisch erkannt und empfohlen

Der wichtige Teil, der dieses „Serverless" Deployment ausmacht, betrifft die Auswahl des Deployment-Typs. Beachten Sie, dass ein „Serverless" Deployment verfügbar ist, wenn OpenShift Serverless installiert ist.

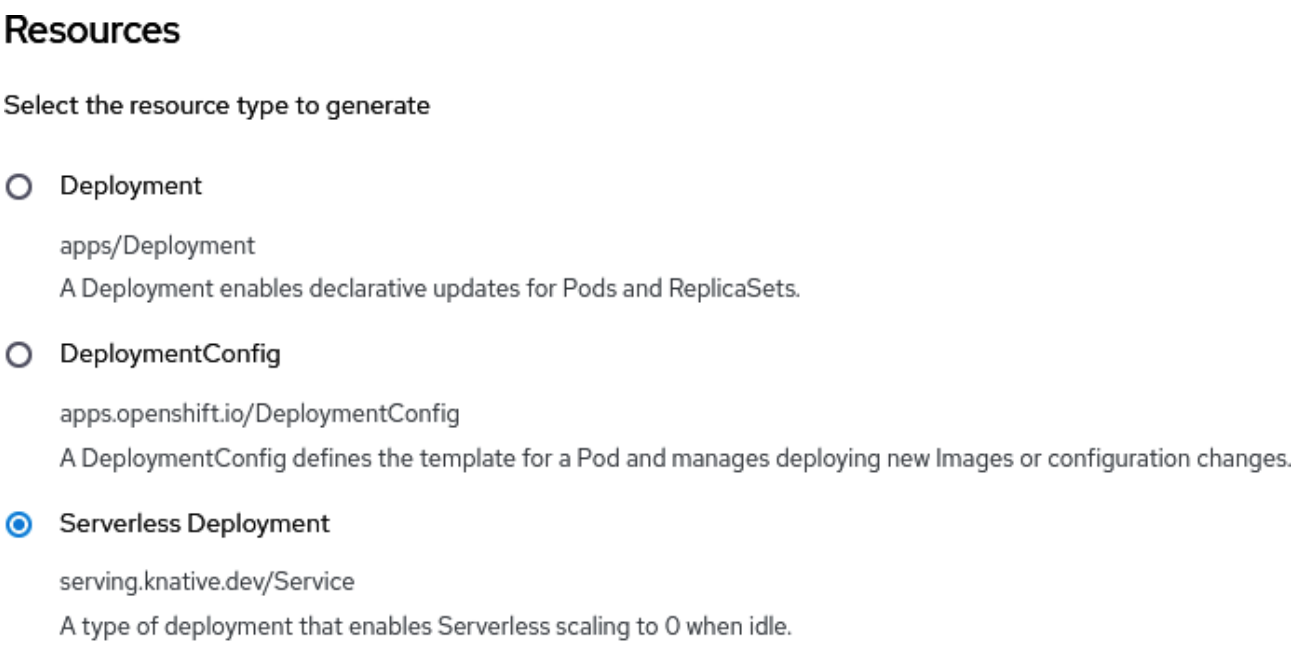


Abbildung 8.8: Deployment-Typ „Serverless"

Es dauert einen Moment, bis der erste Build abgeschlossen ist. Wenn der Build fertiggestellt ist, sollte ein Pod bereitgestellt sein. Die Topologie-Ansicht sollte in etwa so aussehen:

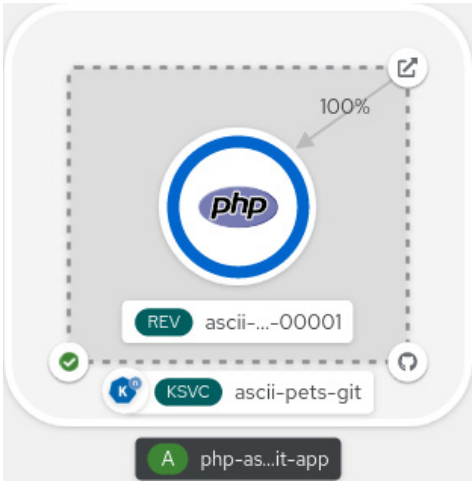


Abbildung 8.9: Eine Knative Serving-App

Die eigentliche Anwendungsseite sieht wie in *Abbildung 8.9* aus. Es handelt sich um eine sehr einfache Anwendung, aber das „Tier", die ASCII-Grafik, ist an den Pod-Hostnamen gebunden. Wenn wir in unserer einfachen Demo-Anwendung viel zusätzliche Last hinzufügen, erzeugt Knative Serving weitere Pods, und Sie können sehen, dass verschiedene „Tiere" angezeigt werden.

Wenn die Anwendung jedoch eine Minute lang keine Anfragen erhält, skaliert Knative Serving diese Anwendung auf null herunter. In der Topologie-Ansicht wird gezeigt, dass die Anwendung nicht mehr ausgeführt wird.

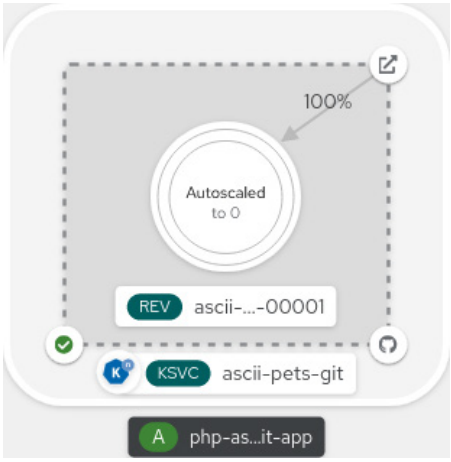


Abbildung 8.10: Ein Serverless Deployment mit Knative Serving und einer auf null skalierten Anwendung

Wenn schließlich jemand die Seite besucht, wird die Anwendung automatisch wieder auf 1 Replikat oder 2 oder 3 oder mehr Replikate hochskaliert, je nachdem, wie viele Instanzen nach Ansicht von OpenShift erforderlich sind, um die Nachfrage zu bedienen.

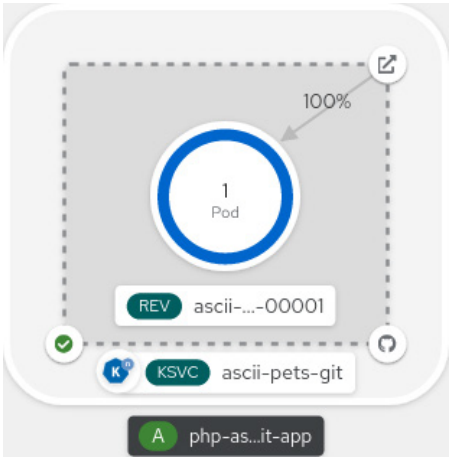


Abbildung 8.11: Automatische Skalierung als Reaktion auf Datenverkehr

Mit Red Hat OpenShift Serverless und Knative Serving können Organisationen dynamisch mit sehr wenig zusätzlicher Konfiguration und ohne Änderungen an Anwendungen vertikal hoch- und herunterskalieren. Dies kann ungemein nützlich sein, um die richtige Größe von Deployments zu erhalten und zu verhindern, dass Ressourcen unnötig genutzt und bezahlt werden.

Weitere Informationen

- About OpenShift Serverless
- learn.openshift.com, enthält einen Kurs zu OpenShift Serverless
- Knative Community Site

Plattformservices – OpenShift Serverless – Beispiel für Knative Eventing

Aufbauend auf der Beispielanwendung und den Grundlagen aus dem vorherigen Kapitel kann OpenShift Serverless eine Anwendung neben dem eingehenden Datenverkehr auch auf Basis von Metriken skalieren. Insbesondere in Szenarien wie der event-gesteuerten Architektur ist es wünschenswert, Instanzen einer Anwendung vertikal skalieren zu können, um Events aus einer Nachrichten-Queue zu verarbeiten oder gemäß einem Timer „aufzuwecken".

Mit dem gleichen Deployment ermöglicht die OpenShift-Konsole die einfache Konfiguration verschiedener Event-Quellen.

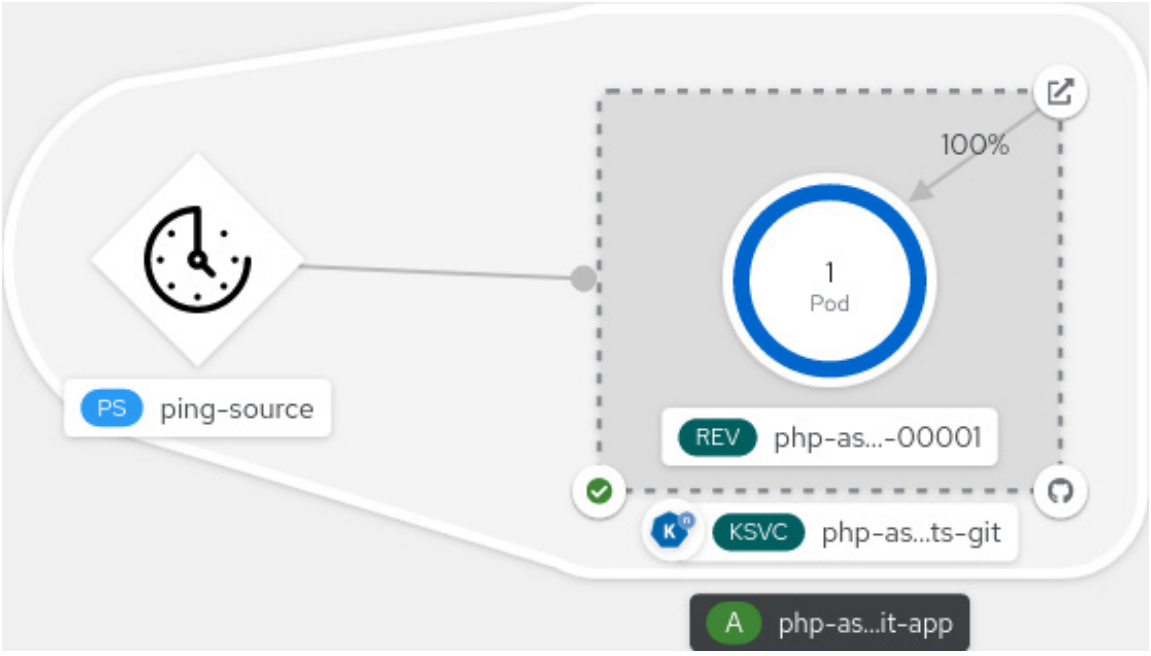


Abbildung 8.12: Eine „Ping"-Event-Quelle, die die Anwendung gemäß einem Timer anpingt

Ein einfacher Anwendungsfall für eine Ping-Event-Quelle wäre das „Aufwecken" eines Backup-Job-Containers jede Nacht um Mitternacht. Ein weiterer Verwendungszweck dieses Ping-Timers könnte ein Überwachungscontainer sein, der in regelmäßigen Abständen ausgeführt wird.

Weitere Informationen

- OpenShift Serverless Eventing explained in 5 minutes
- About OpenShift Serverless

Plattformservices – OpenShift Service Mesh

Red Hat OpenShift Service Mesh, das auf dem Open-Source-Projekt Istio basiert, fügt bestehenden verteilten Anwendungen eine transparente Schicht hinzu, ohne dass Änderungen am Service-Code erforderlich sind. Die Unterstützung für Red Hat OpenShift Service Mesh wird Services durch Bereitstellen eines speziellen Sidecar-Proxys in Ihrer Umgebung hinzugefügt, der die gesamte Netzwerkkommunikation zwischen Microservices abfängt. Sie konfigurieren und verwalten das Service Mesh mit den Control-Plane-Funktionen.

Zu den Anwendungsfällen, die mit OpenShift Service Mesh ermöglicht werden können, gehören:

- Transparente Verschlüsselung für serviceübergreifende Kommunikation mit automatischer mTLS.
- Mehrere Versionen eines Service bereitstellen und z. B. A/B-Tests ermöglichen.
- Observability der Kommunikation zwischen Microservices mit Kiali.
- Tracing von Service-zu-Service-Aufrufen mit Jaeger.

Wie alle anderen Plattformfunktionen von OpenShift wird Service Mesh mit einem Operator von Red Hat installiert.

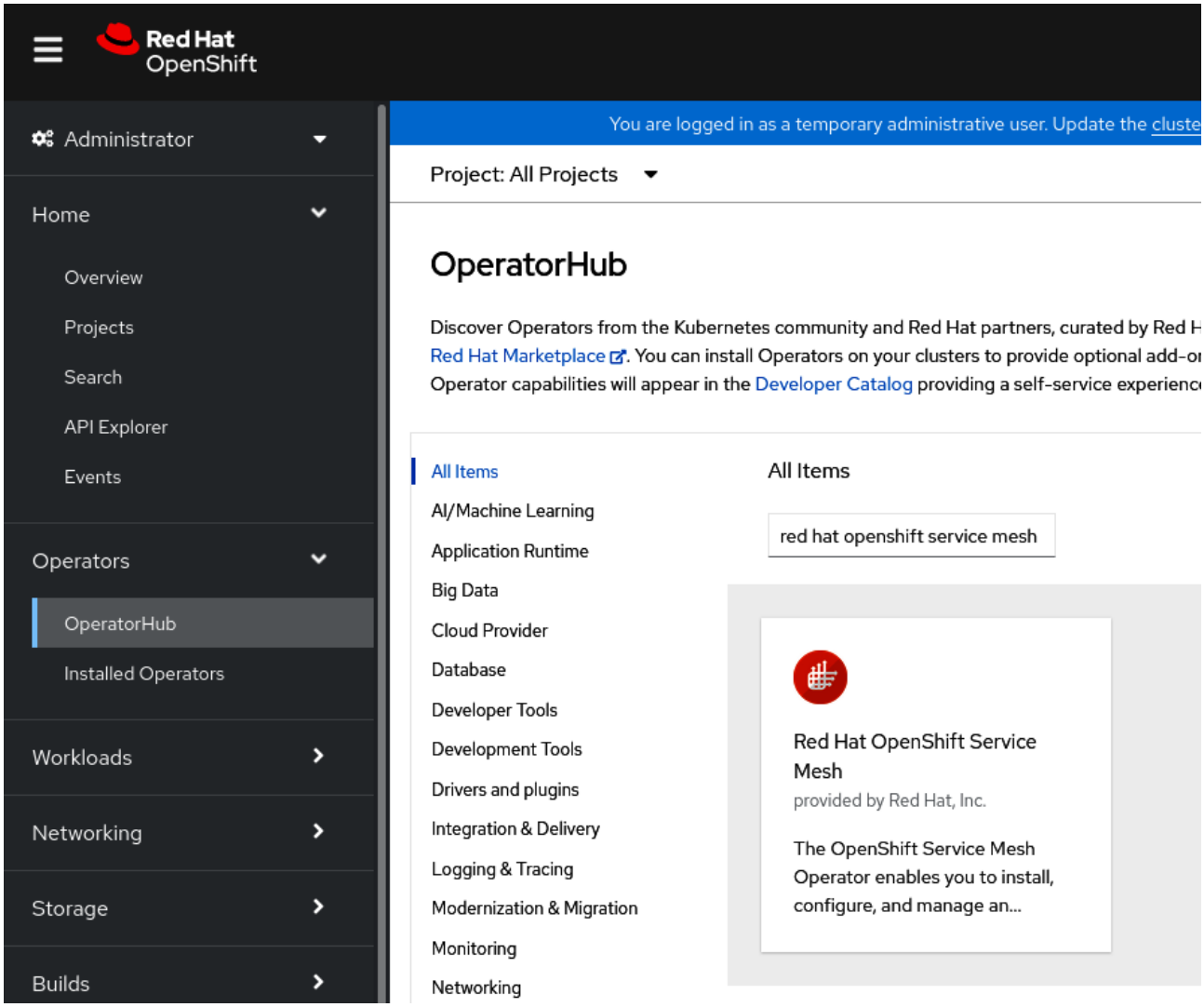


Abbildung 8.13: Installation von Service Mesh über OperatorHub

Die Dokumentation zu OpenShift Service Mesh enthält eine hervorragende Beispielanwendung namens BookInfo-Demo. Das ist eine einfache Anwendung, die einen Buchladen nachbildet, der auf OpenShift ausgeführt wird.

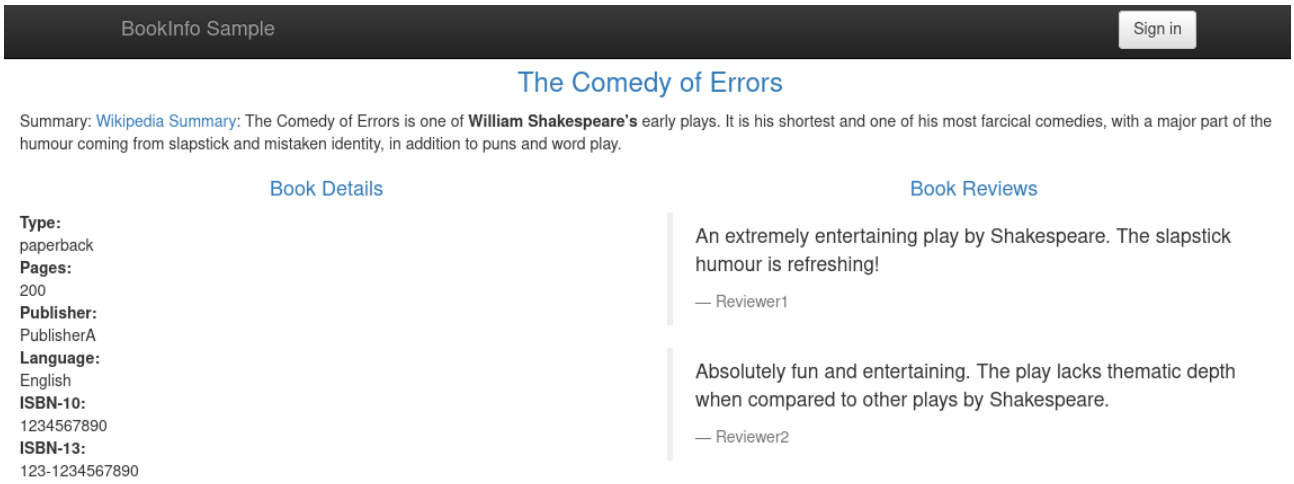


Abbildung 8.14: Die Anwendung BookInfo

Damit die Anwendung BookInfo mit Service Mesh arbeitet, muss eine Service Mesh Control Plane erstellt werden. Dies ist über die grafische Oberfläche von OpenShift einfach zu erreichen, wie in *Abbildung 8.15* gezeigt:

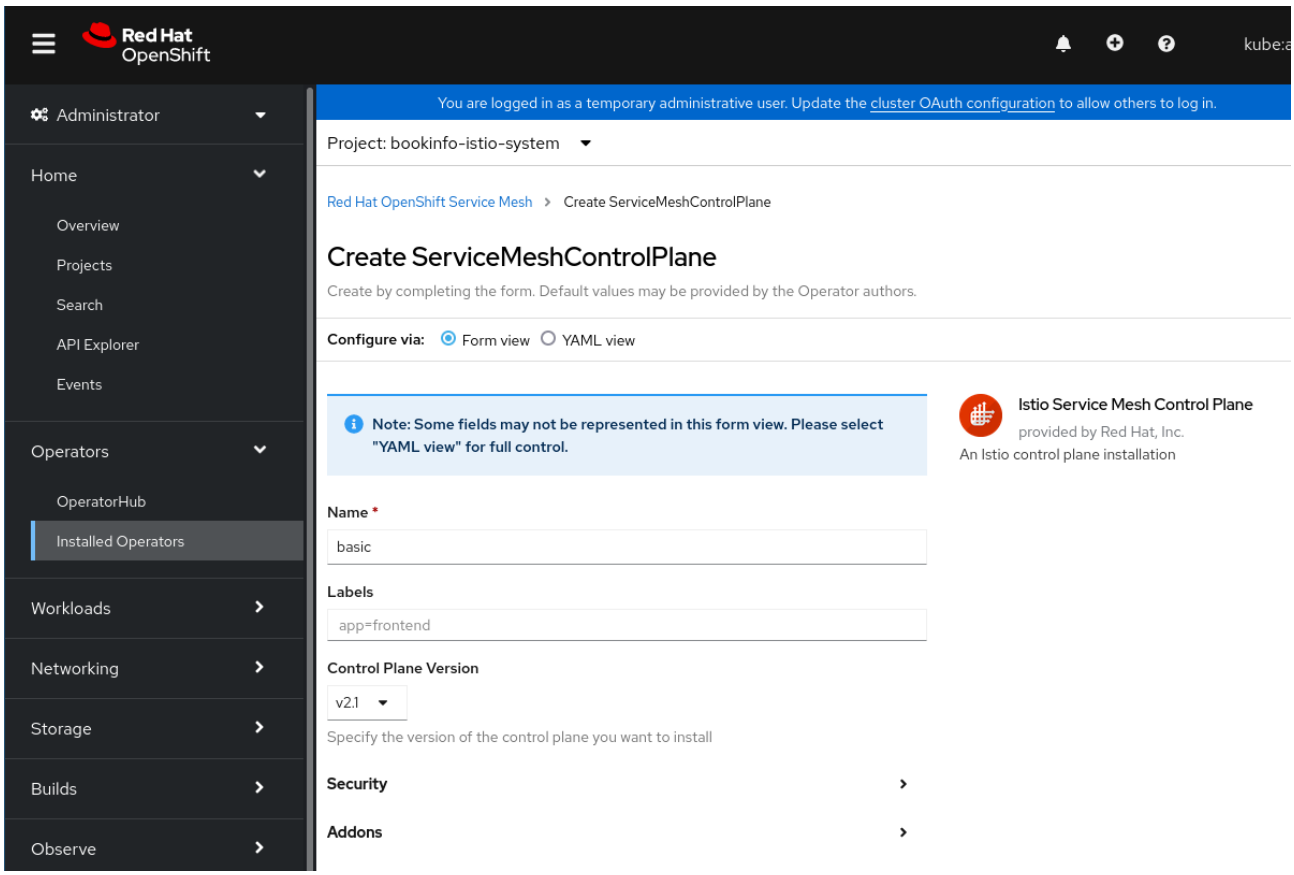


Abbildung 8.15: Einrichten einer Control Plane im Projekt „BookInfo bookinfo-istio-system“

Das BookInfo-Projekt akzeptiert eingehenden Datenverkehr über die Control Plane. In diesem Fall wurde ein separates Projekt mit dem Namen `bookinfo-istio-system` erstellt, das die Control Plane enthält.

Es ist nicht erforderlich die Service Mesh Control Plane und Ihr Projekt zu trennen, und es ist sogar möglich, eine Service Mesh Control Plane über mehrere Projekte hinweg zu teilen.

In den nächsten beiden Abschnitten werden zwei Anwendungsfälle für Service Mesh im Detail behandelt: Observability und verteiltes Tracing.

Plattformservices – OpenShift Service Mesh – Observability mit Kiali

Wenn Sie sich die zugrunde liegenden Pods betrachten, aus denen diese Anwendung besteht, können Sie in der Ansicht „Red Hat OpenShift Topology" sehen, dass sie sechs Microservices umfasst.

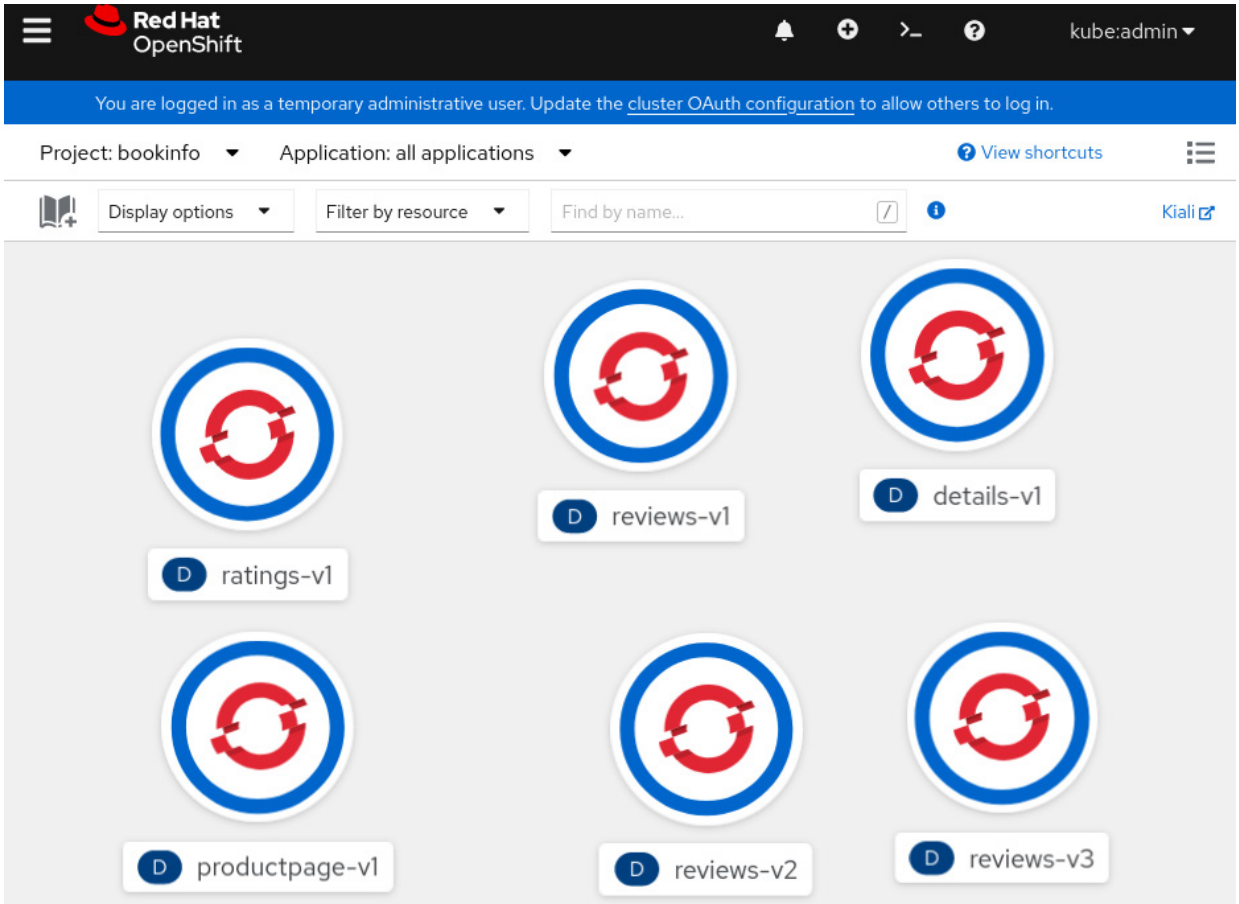


Abbildung 8.16: Die sechs Services, aus denen die Anwendung BookInfo besteht

Obwohl diese Ansicht nützlich ist, vermittelt sie nicht wirklich, wie diese Services verbunden sind. Dies ist der erste Vorteil von Service Mesh – Observability. In der geringfügig anderen, mit Service Mesh gelieferten Konsole Kiali wird eine weitaus verfeinerte Architektur der gleichen sechs Services dargestellt.

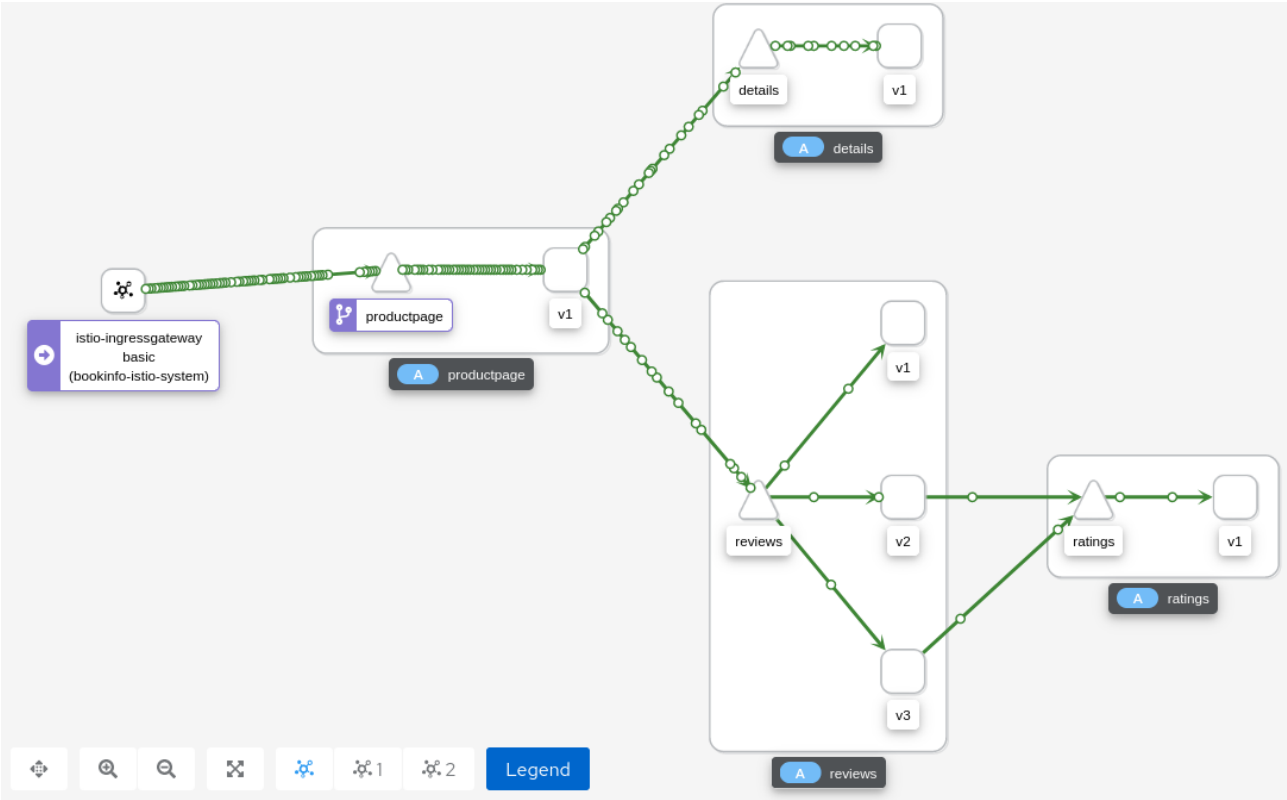


Abbildung 8.17: Das automatisch generierte Diagramm der Anwendungsarchitektur, aktiviert durch Service Mesh

Diese Ansicht zeigt nicht nur die wirkliche Konnektivität zwischen den Services, sondern kann auch ein Echtzeit-Diagramm des Datenverkehrs sein. In diesem Fall erhält die Anwendung Datenverkehr in beträchtlichem Umfang, wobei jede Anfrage durch eine Kreisanimation auf der Verbindung dargestellt ist.

Um diese Observability zu erweitern, kann ein Administrator auf eine der Serviceverbindungen klicken und das Datenverkehrsprofil anzeigen. In diesem Fall wird angezeigt, dass der Datenverkehr überwiegend den Status **HTTP OK** aufweist.

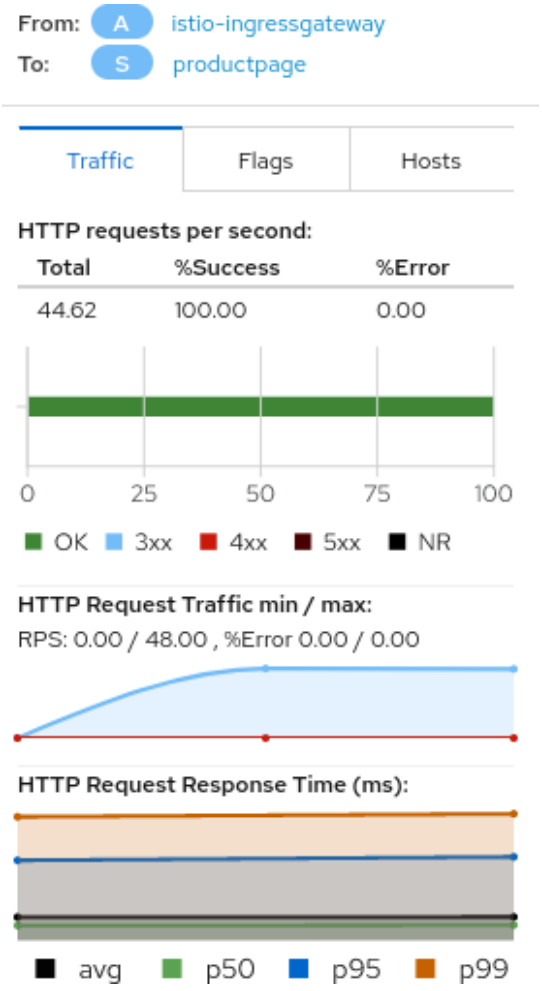


Abbildung 8.18: Verschiedene HTTP-Status

Wir können einen künstlichen Fehler in diese Anwendung einfügen, indem wir den Microservice `details` herunterfahren und ihn auf null Replikate skalieren:

Abbildung 8.19: Null Replikate des Service „details", um einen Fehler zu simulieren

Wenn wir jetzt zur Kiali-Ansicht zurückkehren, werden Sie bemerken, dass dem Diagramm einige zusätzliche Komponenten hinzugefügt wurden, aber vor allem wurde die Verbindung zum Service `details` in Rot hervorgehoben, um auf einen Fehler hinzuweisen.

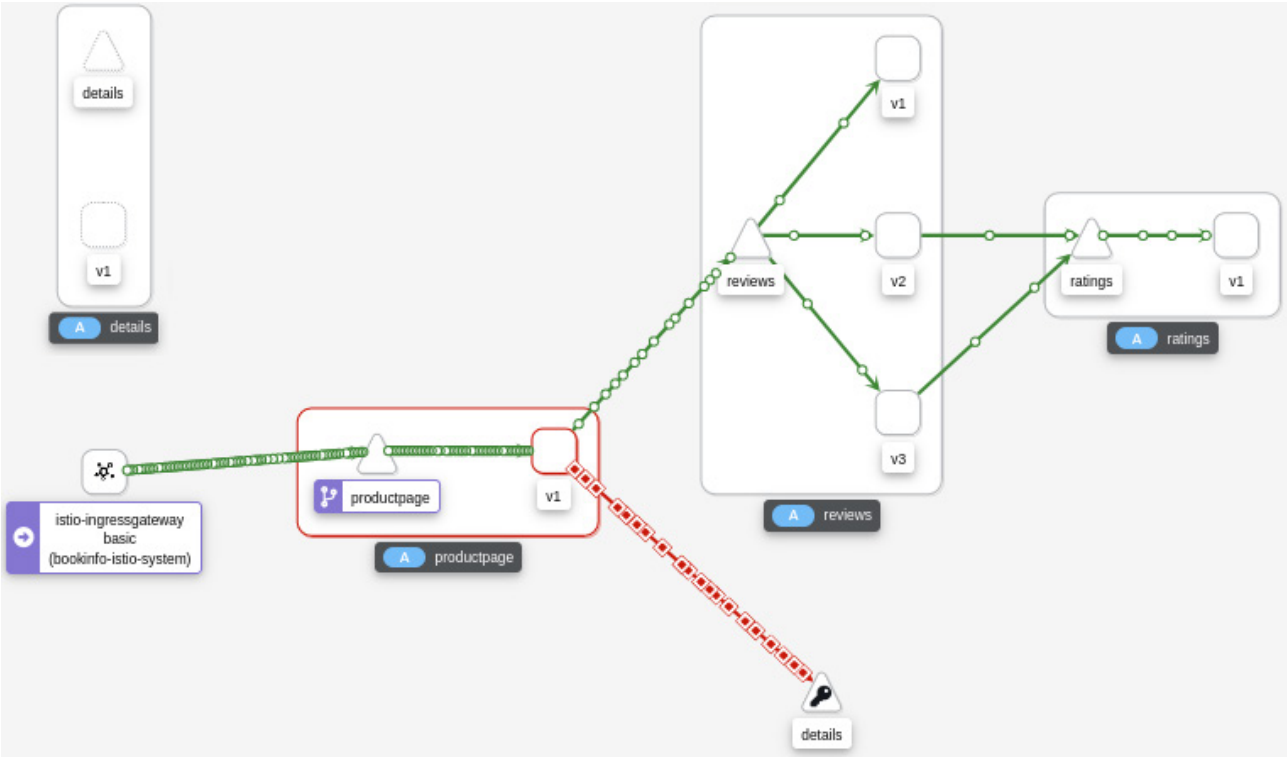


Abbildung 8.20: Die rote Verbindung weist auf ein Problem mit diesem Service hin

Wir haben nun erfahren, dass Kiali von Service Mesh für die Observability sehr hilfreich ist. In kleineren Anwendungen wie BookInfo ist Service Mesh nützlich, wenn Anwendungen aber größer und komplexer werden, mit manchmal 20 oder mehr Microservices und Unmengen verschiedener Interaktionstypen, sind Tools wie Service Mesh und Kiali enorm wertvoll, ja fast unverzichtbar.

Plattformservices – Red Hat OpenShift Service Mesh – verteiltes Tracing mit Jaeger

Ein weiterer äußerst wertvoller Service, der von Service Mesh bereitgestellt wird, ist Jaeger. Dieser Service ermöglicht verteiltes Tracing für komplexe Microservices-Anwendungen. Im vorherigen Abschnitt haben wir bei der Anwendung BookInfo festgestellt, dass Anfragen fehlschlagen, wenn der Service `details` offline geschaltet wird.

Die Ursache des Fehlers konnte in diesem Fall einfach festgestellt werden: Der Service `details` war nicht erreichbar, und wir haben das verursacht. Wenn die Ursache jedoch unklarer wäre und weitere Untersuchungen erforderte, dann wäre das verteilte Tracing von Jaeger hilfreich gewesen.

Jaeger erfolgt Anfragen vom ersten Service über nachfolgende Verbindungen durch das System. Für die Aktivierung von Jaeger ist zwar etwas zusätzlicher Anwendungscode erforderlich, aber die Client-Librarys und minimale Codeänderungen machen dies für Entwickler relativ einfach.

Im Service `productpage` (der in Python geschrieben ist) können wir den relevanten Code suchen, der das verteilte Tracing von Jaeger in diesem Service aktiviert. Dieser Code importiert die Jaeger-Client-Library in den Service `productpage`:

```
from jaeger_client import Tracer, ConstSampler
from jaeger_client.reporter import NullReporter
from jaeger_client.codecs import B3Codec
```

Nach dem Import der Client-Library muss „productpage“ einen neuen Tracer einrichten. Dieser Code initialisiert einen Jaeger-`Tracer` im Service `productpage`:

```
tracer = Tracer(
    one_span_per_rpc=True,
    service_name='productpage',
    reporter=NullReporter(),
    sampler=ConstSampler(decision=True),
    extra_codecs={Format.HTTP_HEADERS: B3Codec()}
)
```

Zuletzt teilen wir Jaeger immer noch im Service `productpage` mit, dass wir einen neuen Span – eine neue Anfrage an mehrere Services – erstellen möchten:

```
span = tracer.start_span(
    operation_name='op', child_of=span_ctx, tags=rpc_tag
)
```

Jaeger verfolgt dann diesen Tracer durch mehrere Services. Diese müssen ebenfalls angepasst werden, damit sie mit Jaeger arbeiten, aber es gibt Client-Librarys für die meisten gängigen Programmiersprachen.

Den vollständigen Quellcode für den Service `productpage` finden Sie auf GitHub.

Es gibt zwei Optionen für die Instrumentierung von Code – die Verwendung von Jaeger-Client-Librarys, die jetzt als veraltet gelten, oder die Verwendung der Optionen der offenen Standards aus dem OpenTelemetry-Projekt, die bevorzugt werden:

- OpenTelemetry libraries

Nach Abschluss der Instrumentierung einer Anwendung werden Traces wie diese angezeigt:

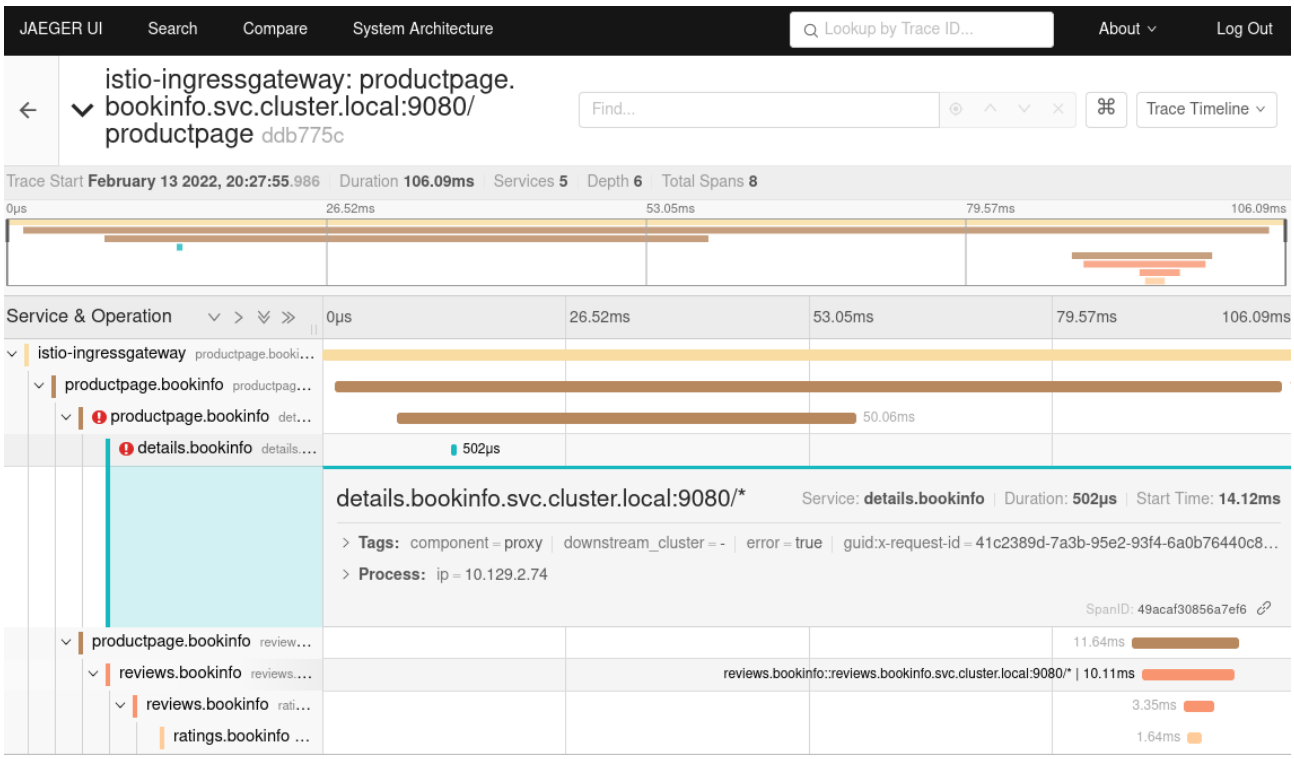


Abbildung 8.21: Ein Aufruf-Trace

Diese Ansicht zeigt einen Aufruf-Trace, ähnlich wie bei Kiali, aber hier wird er in einer hierarchischen Ansicht dargestellt. Jede der Zeilen kann erweitert und weitere Details über Dauer, Startzeit, Quell-IP-Adresse und einige andere Details der Anfrage angezeigt werden. Nochmal: Diese Informationsebene ist bei komplexen Microservices-Anwendungen nahezu unerlässlich.

In Bezug auf den Fehler, den wir zu diagnostizieren versuchten, zeigt die Trace-Ansicht deutlich, dass beim Microservice `details` Probleme auftreten. In diesem Fall ist der Fehler ganz einfach, aber die Erweiterung der Detailansicht zeigt, dass der Service die Fehlercodes `HTTP 503` ausgibt.

Abbildung 8.22: Fehlerdetails

HTTP 503 ist der Standardfehlercode für „Service nicht verfügbar“. In diesem Fall liegt es einfach daran, dass der Service auf null Replikate skaliert ist. Durch erneutes Hochskalieren des Service wird der Service wiederhergestellt.

Zusammen sind Jaeger und Kiali leistungsstarke Tools, da Microservices-Anwendungen immer komplexer werden. Sie werden als Teil des Azure Red Hat OpenShift-Service bereitgestellt und erfordern keine zusätzlichen Ausgaben oder Subskriptionen.

Zusammenfassung

In diesem Kapitel wurden einige der wichtigen Mehrwertservices behandelt, die von der Anwendungsplattform – Red Hat OpenShift – im Gegensatz zu einer „leeren“ Kubernetes-Umgebung bereitgestellt werden. Es ist zwar möglich, ein ähnliches Maß an Funktionalität von OpenShift zu replizieren, indem alle entsprechenden Upstream-Open-Source-Projekte auf OpenShift installiert werden, aber Integration, Lifecycle und Support sind der Umfang, der mit Azure Red Hat OpenShift geliefert wird.

Diese verschiedenen Plattformservices, Anwendungsservices, Datenservices, Entwicklerservices und Clusterservices führen letztlich zu einer größeren Produktivität Ihrer Entwickler und Operators bei der Arbeit mit Kubernetes und der Bereitstellung mehrerer komplexer Unternehmensanwendungen.

Kapitel 9

Integration mit anderen Services

Es ist ganz normal, dass eine Anwendung nicht vollständig in Red Hat OpenShift existieren kann – die meisten Anwendungen basieren in irgendeiner Form auf externen Datenbanken oder Services auf Azure.

Azure Red Hat OpenShift „unterbricht" hier keine Art der Konnektivität. Wenn eine Anwendung beispielsweise auf Azure CosmosDB basiert, sollte sie weiterhin in der Lage sein, eine Verbindung von Azure Red Hat OpenShift zu Azure CosmosDB herzustellen, ohne dass Änderungen an der Anwendung erforderlich sind. Je nach Ihrer Anwendung und Organisation können Sie einige dieser externen Abhängigkeiten getrennt von Azure Red Hat OpenShift oder gleichzeitig bereitstellen.

Wenn Sie der Meinung sind, dass die Bereitstellung dieser externen Abhängigkeiten mit Ihrer Anwendung von Vorteil wäre, dann kann Azure Service Operator diesen Prozess erheblich vereinfachen. Anstatt die Azure CLI, die Azure Cloud Shell oder ARM-Vorlagen verwenden zu müssen, können Sie diese Ressourcen mit Azure Service Operator so bereitstellen, als ob sie nativ in Kubernetes vorhanden wären.

Azure Service Operator

Azure Service Operator ist ein weiterer Operator, der Ihr Leben als Entwickler oder Administrator, der Anwendungen auf Azure Red Hat OpenShift bereitstellt, erleichtert. Ein Vorteil ist, dass es die Provisionierung von Azure-Services ermöglicht, ohne dass die OpenShift-Konsole verlassen werden muss. Dadurch können Anwendungen mit potenziellen Abhängigkeiten von Azure-Services, wie Azure CosmosDB, schneller und einfacher bereitgestellt werden.

Ein weiterer, möglicherweise noch bedeutender Vorteil der Verwendung von Azure Service Operator ist, dass diese Abhängigkeiten von Azure-Services mit der Definition der OpenShift-Anwendung in einem Infrastructure-as-Code-Ansatz mit Kubernetes-YAML-Standarddateien gespeichert werden können.

Das funktioniert denkbar einfach: Azure Service Operator sucht nach neuen, in YAML definierten **CRDs**, die einer Definition für eine Ressource auf Azure entsprechen. Azure Service Operator übersetzt dann dieses YAML in die erforderlichen Azure API-Aufrufe, um das zu erstellen, was auch immer auf Azure angefordert wird. Nachdem der Operator installiert ist, können Nutzer im OpenShift Developer Catalog den Erstellungsbildschirm für viele gängige Azure-Ressourcen, wie öffentliche Azure-IP-Adressen, Azure SQL-Datenbanken oder Azure Firewall, anzeigen.

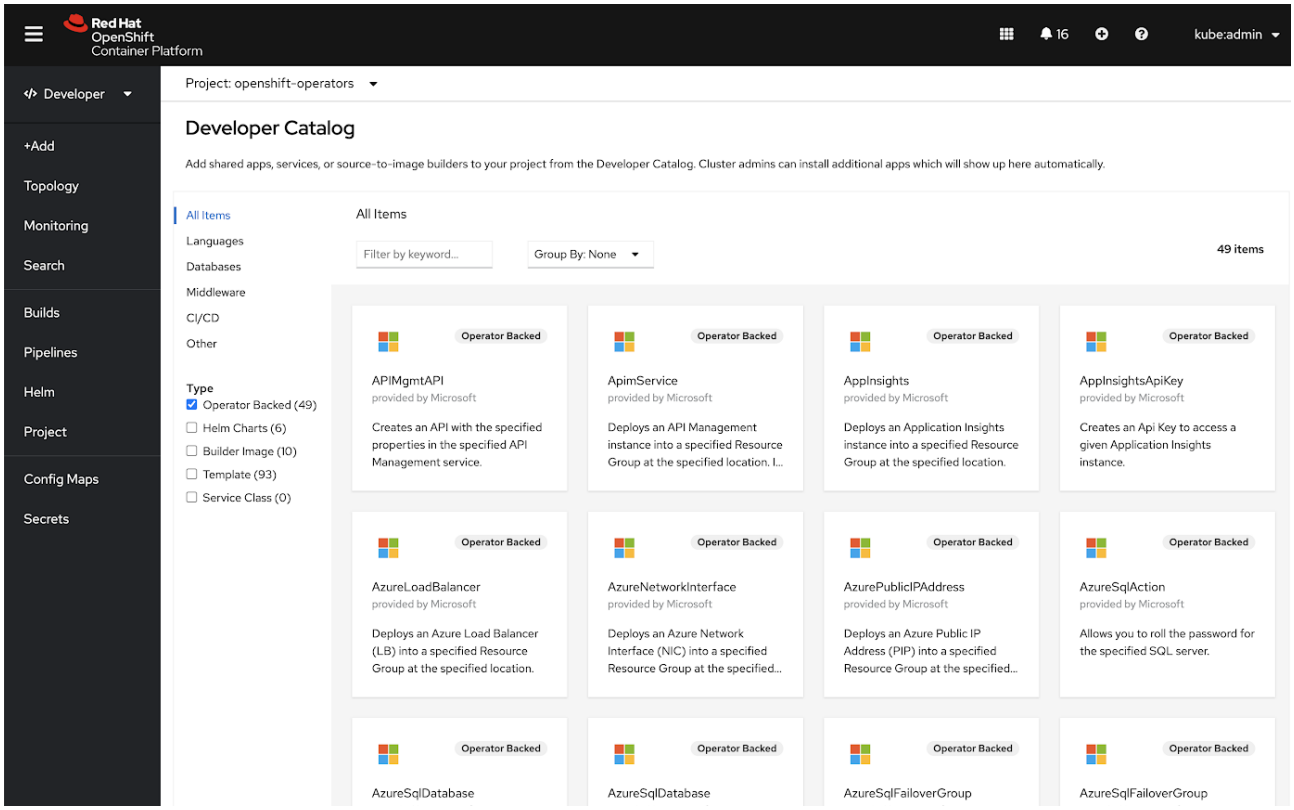


Abbildung 9.1: Einige Azure-Services, die von Azure Service Operator zugänglich gemacht werden

Azure Service Operator kann über OperatorHub installiert und für alle Nutzer von OpenShift verfügbar gemacht werden.

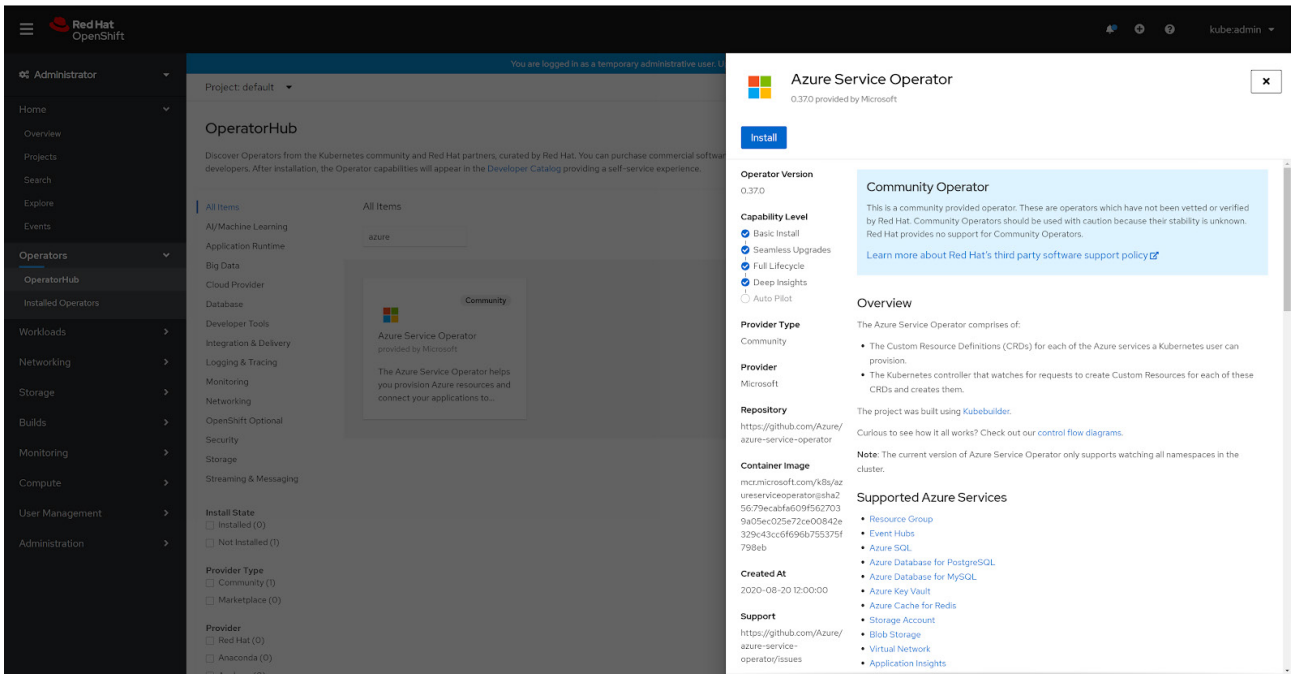


Abbildung 9.2: Der Installationsbildschirm für Azure Service Operator mit der OperatorHub-Galerie im Hintergrund

Die Verwendung von Azure Service Operator ist nicht obligatorisch für Services, die auf Azure ausgeführt werden, und es ist wichtig zu verstehen, dass die zugrunde liegenden Azure-Services nativ auf Azure bereitgestellt werden – nicht auf OpenShift. Azure Service Operator macht dies jedoch schneller und einfacher für Anwendungen mit Abhängigkeiten von Azure-Services.

Ein häufiger Anwendungsfall für Azure Service Operator ist die Bereitstellung abhängiger Datenbanken zusammen mit der Anwendung. Stellen Sie beispielsweise für eine Webanwendung, die eine Instanz von Azure CosmosDB verwendet, Azure CosmosDB mit Azure Service Operator im Rahmen der Bereitstellung der Anwendung auf OpenShift bereit. Azure Service Operator umfasst die Unterstützung für Azure SQL Database, Azure Database for MySQL, Azure PostgreSQL sowie für einige andere.

Vorausgesetzt, dass Azure Service Operator installiert ist, könnte das folgende Kubernetes-Manifest mit der OpenShift-Anwendung gespeichert und zur Provisionierung einer MySQL-Datenbank verwendet werden:

Quelle: Übernommen aus dem Azure Service Operator Samples Repository

```
apiVersion: azure.microsoft.com/v1alpha1
kind: MySQLServer
metadata:
  name: aso-wpdemo-mysqlserver
spec:
  location: eastus2
  resourceGroup: aso-wpdemo-rg
  serverVersion: "8.0"
  sslEnforcement: Disabled
  minimalTLSVersion: TLS10 # Possible values include: 'TLS10', 'TLS11', 'TLS12', 'Disabled'
  infrastructureEncryption: Enabled # Possible values include: Enabled, Disabled
  createMode: Default # Possible values include: Default, Replica, PointInTimeRestore (not
implemented), GeoRestore (not implemented)
  sku:
    name: GP_Gen5_4 # tier + family + cores eg. - B_Gen4_1, GP_Gen5_4
    tier: GeneralPurpose # possible values - 'Basic', 'GeneralPurpose', 'MemoryOptimized'
    family: Gen5
    size: "51200"
    capacity: 4
```

Es wäre ungewöhnlich, Azure Service Operator für Services zu verwenden, die viele Azure-Services mit komplexen Abhängigkeiten umfassen. In solchen Fällen sind Tools wie Azure ARM-Vorlagen oder Bicep möglicherweise besser geeignet.

Eine ausführliche Einführung in Azure Service Operator finden Sie in den folgenden Blog-Beiträgen:

- Blog post – September 2020
- Azure Service Operator on OperatorHub.io
- Azure Service Operator on GitHub

Integration in Azure DevOps

Zusammen mit OpenShift Pipelines möchten viele Nutzer Azure Red Hat OpenShift in Azure DevOps integrieren. Diese Lösungen können problemlos nebeneinander bestehen, wobei einige Projekte die eine Lösung verwenden und andere die andere – oder manchmal verwenden die Projekte auch beide Lösungen. Die Entscheidung darüber, wann die jeweilige Lösung verwendet wird, hängt von einigen Faktoren ab.

Im Allgemeinen bietet Azure DevOps ein hohes Maß an Integration von anderen Azure-Tools, kann aber auch problemlos auf OpenShift und anderen Compute-Ressourcen bereitgestellt werden.

Andererseits ist OpenShift Pipelines ein gut integriertes Angebot, das mit OpenShift geliefert wird und ein konsistentes Multi-Cloud-Erlebnis bietet.

Azure DevOps behandelt OpenShift einfach wie jeden anderen Kubernetes-Cluster. Daher arbeiten alle Standard-Kubernetes-Schnittstellen und -APIs wie erwartet.

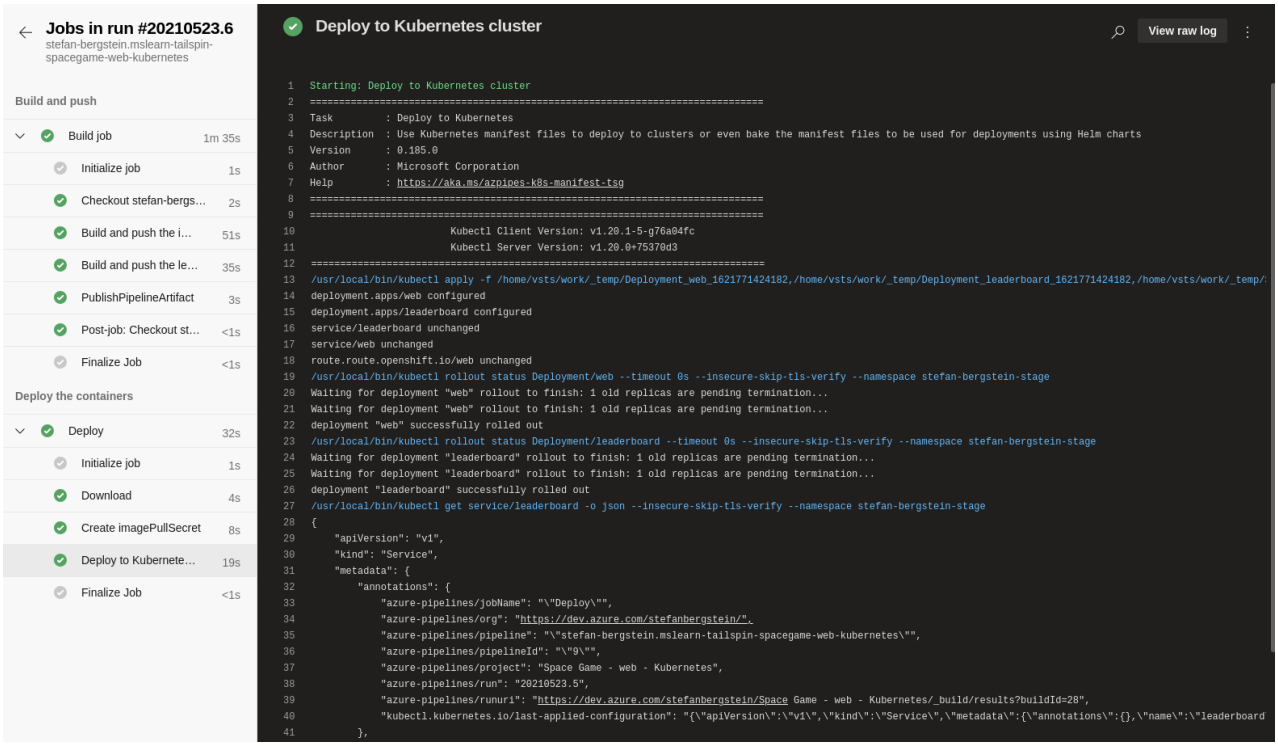


Abbildung 9.3: Eine Azure DevOps-Pipeline, die Inhalt auf OpenShift verschiebt

Weitere Informationen

Der folgende Community-Blog-Beitrag enthält eine gutes Tutorial über die ersten Schritte mit Azure Red Hat OpenShift und Azure DevOps:

- Blog post – May 2021

Integration in Visual Studio Code

Als Teil der OpenShift-Entwicklerintegration gibt es Plugins für viele gängige IDEs, darunter auch Visual Studio Code. Dadurch können Entwickler und Administratoren schnell und einfach auf Kubernetes- und OpenShift-Ressourcen zugreifen, ohne ihre IDE verlassen zu müssen.

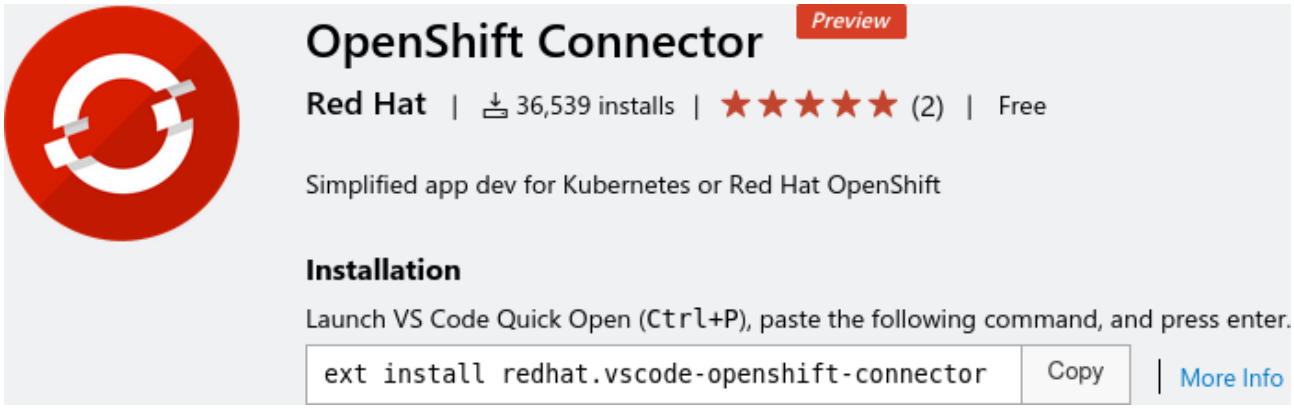


Abbildung 9.4: Installation von OpenShift Connector

Wenn der Connector heruntergeladen und in Visual Studio Code installiert wurde, werden Sie aufgefordert, sich bei Ihrem OpenShift-Cluster anzumelden. Das folgende Beispiel ist ein einfaches Projekt im Stil von „Hello World“ mit einer Verbindung zu Azure Red Hat OpenShift:

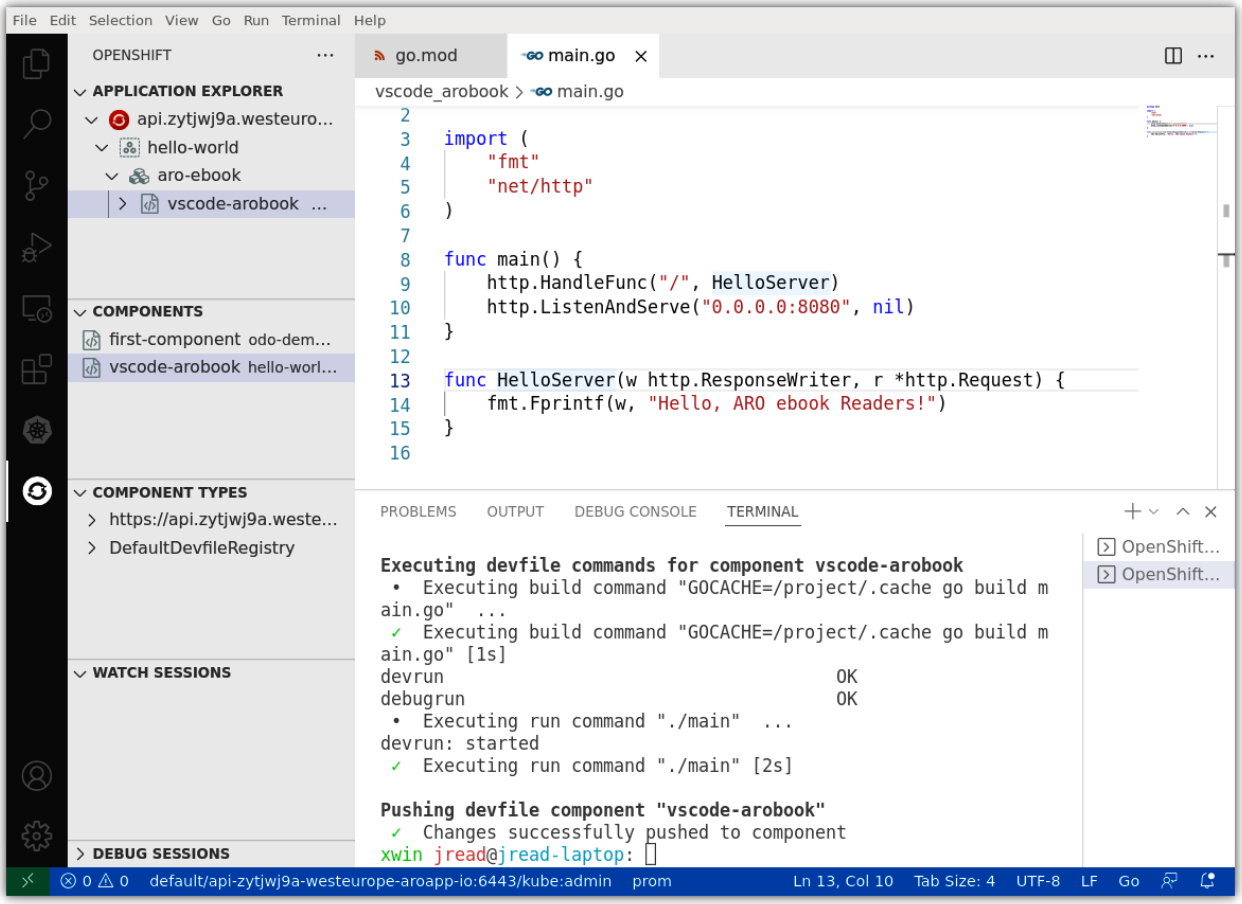


Abbildung 9.5: Ein Golang-Projekt, das mit dem OpenShift-Connector für Visual Studio Code bereitgestellt wurde

Ein Vorteil bei der Verwendung des OpenShift-Connector mit Visual Studio Code ist die Verfügbarkeit eines grafischen Frontends zu dem beliebten OpenShift-Tool „OpenShift Do“ – normalerweise einfach „odo“ genannt. Dies ermöglicht es Entwicklern, in übergeordneten Konzepten zu denken und nicht nur in reinen Kubernetes-Komponenten. Entwickler müssen sich nicht so viele Gedanken über Details von Deployments, ReplicaSets usw. machen. Im vorherigen Screenshot sehen Sie, dass dies ein einfaches Projekt mit einem verfügbaren HTTP-Service ist.

Darüber hinaus können mit dem Connector auch Codeänderungen direkt nach OpenShift verschoben werden, ohne notwendigerweise zuerst die Versionskontrolle nutzen zu müssen. Dies ist für eine schnelle Entwicklung sehr nützlich, da der Code nicht jedes Mal in die Versionskontrolle verschoben, ein neuer Container erstellt und bereitgestellt werden muss.

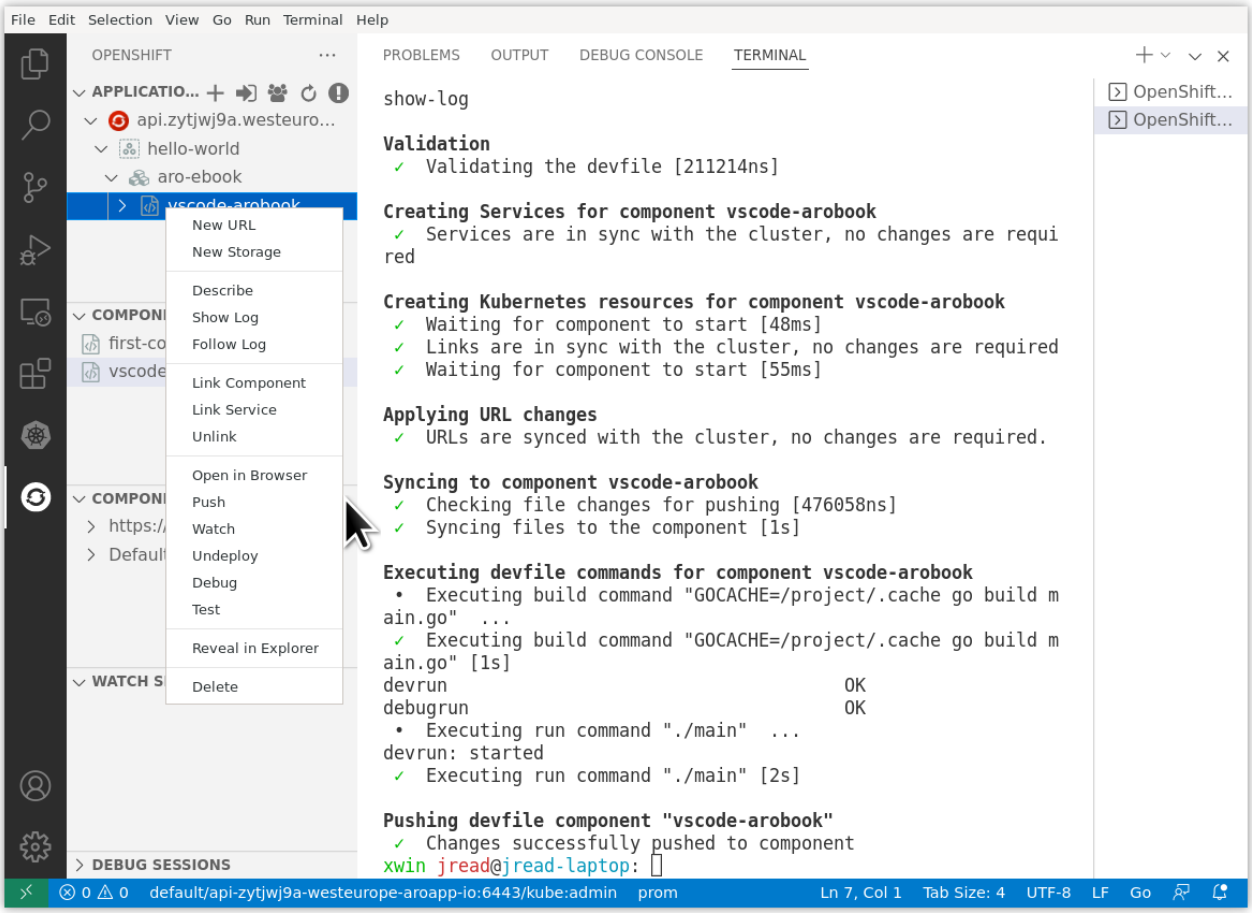


Abbildung 9.6: Das „Push"-Menü für ein Projekt und ein aktueller Push im Terminalfenster

Dieser Connector beschleunigt einfach die Entwicklungs- und Testzyklen, für Entwickler, die bevorzugt mit Visual Studio Code arbeiten.

Abbildung 9.7: Die grundlegende Golang-Anwendung, die auf OpenShift ausgeführt wird

Selbstverständlich sind Entwickler nicht nur auf Visual Studio Code beschränkt – viele Entwickler arbeiten mit Vim, Eclipse und anderen Editoren –, aber Visual Studio Code ist sehr beliebt bei denjenigen Entwicklern, die „schlanke IDEs" bevorzugen.

Weitere Informationen

- Demo video – using Visual Studio Code with OpenShift
- Visual Studio Code OpenShift Connector

Integration in GitHub Actions

Jetzt kann GitHub Actions für die Bereitstellung in jeder beliebigen Red Hat OpenShift-Umgebung, einschließlich Azure Red Hat OpenShift, verwendet werden. Navigieren Sie aus einem GitHub-Repository zu **Actions** → **New Workflow**, und wählen Sie aus der Liste der verfügbaren Aktionen **OpenShift** aus.

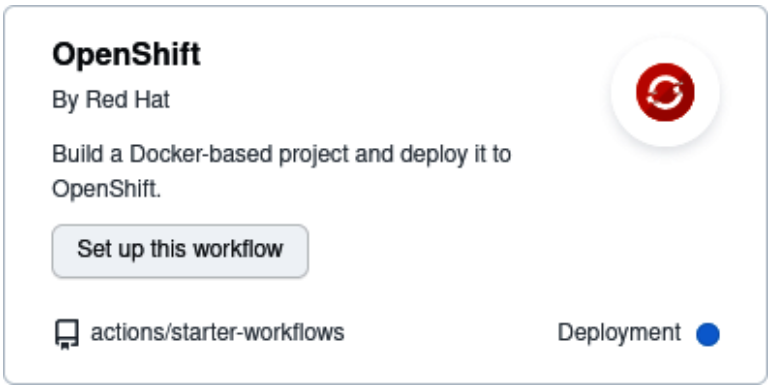


Abbildung 9.8: Bereitstellen auf OpenShift von GitHub aus

Die Standardvorlage enthält einen Satz ausgezeichneter Beispielaktionen, für die nur einige Umgebungsvariablen für die Verbindung zu einem OpenShift-Cluster festgelegt werden müssen:

```
name: OpenShift

env:
  # ✏️ EDIT your repository secrets to log into your OpenShift cluster and set up the context.
  # See https://github.com/redhat-actions/oc-login#readme for how to retrieve these values.
  # To get a permanent token, refer to https://github.com/redhat-actions/oc-login/wiki/Using-a-
Service-Account-for-GitHub-Actions
  OPENSHIFT_SERVER: ${{ secrets.OPENSHIFT_SERVER }}
  OPENSHIFT_TOKEN: ${{ secrets.OPENSHIFT_TOKEN }}
  # ✏️ EDIT to set the kube context's namespace after login. Leave blank to use your user's
default namespace.
  OPENSHIFT_NAMESPACE: ""
  ...
```

Einen hervorragenden Blog-Beitrag, in dem beschrieben wird, wie dies ausgeführt wird, finden Sie zusammen mit einem Demo-Video hier:

- Blog-Beitrag
- GitHub Actions with OpenShift demo video

Zusammenfassung

In diesem Kapitel wurden viele der gängigen Integrationen behandelt, die Kunden mit Azure Red Hat OpenShift nutzen. Wie in der Kapiteleinführung erwähnt, ermöglichen die OpenShift-Standard-APIs die Integration von vielen weiteren Services, die in diesem Kapitel nicht aufgeführt sind. Eine Vielzahl von Services kann auch problemlos mit den in OperatorHub aufgeführten Operatoren integriert werden.

Im nächsten Kapitel stellen wir einige Best Practices und Empfehlungen dazu vor, wie Sie Anwendungsteams einbinden und die Einführung des Service in Ihrer Organisation vorantreiben können.

Kapitel 10

Onboarding von Workloads und Teams

Wenn Sie die Schritte *vor der Provisionierung* durchlaufen, Azure Red Hat OpenShift provisioniert und die notwendigen Schritte nach der Provisionierung ausgeführt haben, müssen Sie nur noch Entwickler und Anwendungen auf dem Cluster unterstützen und einbinden. Wie Sie dabei vorgehen, hängt letztlich von der Kultur Ihrer Organisation ab – einige Entwickler- und Anwendungsteams sind darauf erpicht, „ins kalte Wasser zu springen" und loszulegen, während andere Teams vielleicht mehr Anleitung schätzen.

Dieses Kapitel dient als Satz von Checklisten, die eine gute Basis bieten, um sicherzustellen, dass Ihre Entwickler und Anwendungsteams bei der Verwendung von Azure Red Hat OpenShift wahrscheinlich schneller einsatzbereit sind.

Erweitern Sie diese Checklisten, wenn Sie sie an ihre Organisationsstruktur und -kultur anpassen.

Checkliste: Vor dem Onboarding

Vor dem Onboarding einer neuen Workload ist es wichtig, dem Anwendungsteam oder den Verantwortlichen für diese Workload zu vermitteln, dass Azure Red Hat OpenShift so bereitgestellt und konzipiert wurde, dass es bereits den Best Practices Ihrer Organisation entspricht:

- Azure Red Hat OpenShift wurde in einer Azure-Zielzone oder in einer anderen Azure-Umgebung bereitgestellt, die für die Anwendungen der Organisation zugelassen ist.
- Der Cluster verfügt über alle erforderlichen „Day-2"-Konfigurationen, wie Storage-Klassen und Authentifizierung (in *Kapitel 6, Nach der Provisionierung – Day 2* beschrieben).
- Der Cluster ist bereits vollständig konfiguriert, wird von Ihrem Plattformteam unterstützt und wird auch durch den integrierten Support von Red Hat und Microsoft unterstützt.
- Der Cluster wurde auf die letzte stabile Version aktualisiert und verfügt über alle erforderlichen Patches. Er wird auch in Zukunft gut gepatcht werden.

- Der Azure Red Hat OpenShift-Cluster verfügt über eine Verbindung zu den wichtigsten erforderlichen Ressourcen:

  - Konnektivität zu der lokalen Umgebung über Azure ExpressRoute oder ein VPN
  - Konnektivität zu einem Unternehmens-Artefakt-Repository (wie einem Java-Maven-Repository)
  - Konnektivität zum öffentlichen Internet zum Herunterladen von Abhängigkeiten während des Anwendungs-Builds

Onboarding von Pilot-Workloads

Ein gängiges Pattern für die Einführung von Azure Red Hat OpenShift in einer Organisation besteht darin, mindestens zwei Pilot-Workloads einzubinden:

- **Die minimal sinnvolle Workload** – Idealerweise ist die erste Pilot-Workload für ein SRE-Team eine kleine, gut bekannte Anwendung mit sehr einfachen technischen Anforderungen. Dies geht etwas über eine einfache „Hello World"-Anwendung hinaus, da die Anwendung normalerweise einen echten Verantwortlichen innerhalb der Organisation haben sollte. Bei der ersten Workload liegt der Schwerpunkt jedoch darauf zu überprüfen, ob der Azure Red Hat OpenShift-Cluster die geschäftlichen Anforderungen auf Clusterebene erfüllen kann – Azure-Konnektivität, Azure Active Directory-Anmeldung – und um einfach allgemeine Probleme zu identifizieren, die bei jeder Anwendung auftreten können.
- **Eine sinnvolle Referenz-Workload** – Nach der Bereitstellung einer minimalen, einfachen Workload ist es für die weitere Einführung sehr hilfreich, wenn die zweite Workload ausreichend komplex und sinnvoll (wichtig für das Unternehmen oder sogar geschäftskritisch) ist. Anhand dieser Workload können Probleme, wie Konnektivität zu wichtigen Datenbanken, Performance-Tests und Protokollierung/Metriken, im großen Umfang identifiziert und behoben werden. Wichtig ist, dass diese sinnvolle Referenz-Workload dann als **interne Referenz** in Ihrer Organisation verwendet werden kann. Dadurch können zukünftige Entwicklungs- und Anwendungsteams Vertrauen in die Azure Red Hat OpenShift-Plattform gewinnen.

Es gibt selbstverständlich auch andere Pattern für das Onboarding der ersten Workloads, die für Ihre Organisation sinnvoll sein können. Es könnte nötig sein, Anwendungen in einem Rechenzentrum, das demnächst außer Betrieb genommen wird, ins Blickfeld zu nehmen oder Anwendungen, die auf einem alten Java-Anwendungsserver ausgeführt werden.

Checkliste: Onboarding-Meeting mit weiteren Teams

Beim Onboarding eines neuen Entwicklungs- oder Anwendungsteams für den Cluster ist ein gutes Vorgehen, ein Onboarding-Meeting abzuhalten:

- Erstellen Sie einen Namespace oder eine Reihe von Namespaces, die das Team verwenden kann.
- Zeigen Sie dem Team eine kurze Demo von Red Hat OpenShift, und weisen Sie auf die wichtigsten Funktionen wie „Deploy from GitHub" (oder Ähnliches) hin.
- Überprüfen Sie, ob der Cluster über genügend freie Kapazität zum Bereitstellen der Ziel-Workload (CPU, RAM, Storage usw.) verfügt.
- Überprüfen Sie, ob sich die Teammitglieder beim Cluster anmelden können – mit Konnektivität zu Azure Active Directory kann dies vereinfacht werden.
- Stellen Sie für das SRE-Team (oder ein vergleichbares Team), das den Cluster managt, Kontaktdaten bereit.
- Halten Sie eine Liste mit Links für die ersten Schritte in OpenShift bereit:
    - http://learn.openshift.com
    - https://github.com/openshift-labs/starter-guides
    - http://docs.openshift.com

Checkliste: Regelmäßige Gespräche zur Zustandsprüfung

Nachdem ein Entwicklungs- oder Anwendungsteam mit der Bereitstellung des Clusters begonnen hat, ist es häufig sinnvoll, regelmäßige Meetings zur Zustandsprüfung abzuhalten. Diese könnten Teil eines Daily Stand-up sein, oder einfach wöchentlich 30 Minuten könnten ausreichen. Folgende Punkte sollten Sie besprechen:

- Wie ist das Team im Allgemeinen vorangekommen – wurden Anwendungen bereitgestellt?
- Gab es kürzlich Probleme bei der Verwendung des Clusters (Red Hat OpenShift-Wissens- oder Konnektivitätsprobleme)?
- Gab es Ausfälle in Azure oder beim Cluster, die sich negativ ausgewirkt haben?
- Erinnerung an die Verfügbarkeit des SRE-Teams und die Kontaktdaten für die Beantwortung von Fragen.

Überlegen Sie, was Sie sonst noch bei regelmäßigen Gesprächen zur Zustandsprüfung für ähnliche Projekte bereits genutzt haben könnten. Ergänzen Sie diese Checkliste um Punkte, die Sie für sinnvoll halten.

Anti-Pattern: Developer Playgrounds/Sandboxen

Ein häufiges „Anti-Pattern" zur Förderung der Akzeptanz von Entwicklern in einer Organisation ist die Bereitstellung einer Sandbox-Umgebung, die gemeinhin als „Developer Playground" bezeichnet wird, und die Erwartung, dass Entwickler und Anwendungen Azure Red Hat OpenShift einführen werden. Ohne zusätzliche Unterstützung oder Ressourcen kann Azure Red Hat OpenShift eine einschüchternde Umgebung sein mit einem hohen Komplexitätsgrad, der überfordern kann. Häufig kann die Einführung eines „Sandbox"-Pattern zu vergeudeten Cloud-Computing-Ausgaben und leeren Clustern führen.

Wenn Ihre Organisation jedoch in der Vergangenheit bereits „Sandboxing" mit Entwicklungsteams durchgeführt hat, finden Sie hier einige Tipps, um dies so erfolgreich wie möglich zu gestalten:

- Führen Sie zumindest eine kurze Demo dazu durch, was Azure Red Hat OpenShift kann.
- Stellen Sie Dokumentation und mindestens die Links für die ersten Schritte bereit, wie zuvor erwähnt.
- Organisieren Sie einen „Hackathon"-Event, bei dem Entwickler etwas mit OpenShift in einem kleinen Wettbewerbsformat erstellen sollen.
- Bieten Sie optional erweiterten Support, eine Einführung für das SRE-Team und ein stärker geführtes Onboarding-Erlebnis an.

Die Befolgung einer Checkliste mit Pattern für die geführte Einführung, wie zuvor beschrieben, führt mit größerer Wahrscheinlichkeit zu einer Akzeptanz der Plattform.

Azure Red Hat OpenShift Developer Workshop

Der Azure Red Hat OpenShift Developer Workshop (https://aroworkshop.io) ist ein nützlicher vorgefertigter Workshop, den Sie in Ihrer Organisation einsetzen können, um ein geführtes, praktisches Erlebnis mit Azure Red Hat OpenShift zu ermöglichen. Der Workshop ist in zwei Lab-Übungen unterteilt, die beide darauf ausgelegt sind, einen Entwickler oder einen Operator, der neu in OpenShift ist, durch ein angeleitetes Tutorial zu führen. Beide Labs heben die wichtigsten Funktionen von OpenShift hervor und zeigen, wie sie genutzt werden können. Lab 1 ist eine Einführung in modernes Microservices-Design, und in Lab 2 geht es um die Interna des Service.

Die Verwendung des Inhalts von aroworkshop.io für Ihren eigenen internen Azure Red Hat OpenShift-Cluster kann für die Schaffung eines Bewusstseins für Azure Red Hat OpenShift in Ihrer Organisation hilfreich sein. In einer Besprechung vor dem Workshop können Sie erklären, wie Azure Red Hat OpenShift in der bestehenden Azure Red Hat OpenShift-Subskription Ihrer Organisation bereitgestellt wurde und dass die Nutzung des Service in dem Workshop ein ähnliches Erlebnis sein könnte wie die Nutzung von Azure Red Hat OpenShift für das Hosting ihrer Produktionsanwendungen.

Zusammenfassung

In diesem Kapitel wurden einige hilfreiche Checklisten und Anleitungen für das erfolgreiche Onboarding von Workloads und Teams für Azure Red Hat OpenShift vorgestellt. Es wurde ein häufiges Anti-Pattern beschrieben – „Sandbox"-Cluster ohne Support. Es ist schwierig, eine vollständige Liste von Ressourcen und Checklistenpunkten zu erläutern, die für alle Organisationen gelten, daher ist es wichtig, dass Sie den Inhalt dieses Kapitels an Ihre Anforderungen anpassen.

Die Cloud bietet eine ansprechende Palette von Services. Viele werden jedoch übersehen oder schlecht eingeführt, weil Organisationen sich stark auf die Technologie und deren Bereitstellung konzentrieren. Das Verständnis dieser Themen ist wichtig, aber sie sind nur ein kleiner Teil der Gleichung, wenn es darum geht, diesen Service in die Produktion zu übernehmen und effizient einzuführen. In diesem Kapitel wurde versucht hervorzuheben, dass Training, regelmäßiges Einchecken und vergleichbare Aktivitäten notwendig sind, um Ihre Einführung von Azure Red Hat OpenShift erfolgreicher zu gestalten.

Kapitel 11

Fazit

Ihre Entwicklungs- und Operations-Teams können einen Großteil ihrer Arbeitszeit mit der Provisionierung, Einrichtung, Wartung und Überwachung Ihrer Cluster und der CI/CD-Pipeline verbringen. Wenn sie das tun, fehlt ihnen wertvolle Zeit für das, was sie am besten können – Ihre Anwendungen auf dem neuesten Stand halten.

Wie Sie in diesem Guide erfahren haben, können Sie mit Azure Red Hat OpenShift vollständig gemanagte Red Hat OpenShift-Cluster bereitstellen, ohne sich Sorgen über die Erstellung oder das Management der benötigten Infrastruktur für die Ausführung zu machen. Sie haben gesehen, dass die alleinige Ausführung von Kubernetes einige Einschränkungen mit sich bringt, hauptsächlich in Bezug auf zusätzliche Aufmerksamkeit bei Aufgaben, die mit Azure Red Hat OpenShift automatisiert werden könnten.

Beachten Sie bei der Wahl einer Cluster-Management-Strategie für Ihre Organisation die Vor- und Nachteile, die Sie bei einer Kubernetes-Plattform im Vergleich zum Azure Red Hat OpenShift-Service erhalten, der auf dem Kubernetes-Framework basiert und zusätzliche Out-of-the-Box-Vorteile bietet.

Weitere Informationen zu Azure Red Hat OpenShift finden Sie auf der Produktseite oder im Abschnitt „Dokumentation“. Sie können auch einen praktischen Workshop absolvieren und sich registrieren, um sich zu einem beliebigen Zeitpunkt ein Webcast anzusehen. Vor allem hoffen wir, dass Sie sich an Microsoft und Red Hat wenden, damit Sie für Ihre Testversion von Azure Red Hat OpenShift Support erhalten.

Autoren und Versionen

James Read <james@redhat.com>
Principal Solution Architect bei Red Hat,
für Microsoft – aktualisiert und überarbeitet für AROv4

Ahmed Sabbour <asabbour@microsoft.com>
Senior Product Marketing Manager bei Microsoft,
für Azure Red Hat OpenShift – erste Version für AROv3

Autoren von vorherigen Editionen

Oren Kashi <okashi@redhat.com>
Senior Principal Technical Product Marketing Manager bei Red Hat

Dank gilt Brooke Jackson, Nermina Miller, Jose Moreno, Ahmed Sabbour, Aditya Datar, Vince Power, Alex Patterson und anderen, die freundlicherweise ihre Zeit und ihr Feedback für die Überarbeitung dieses Guide angeboten haben.

Kapitel 12

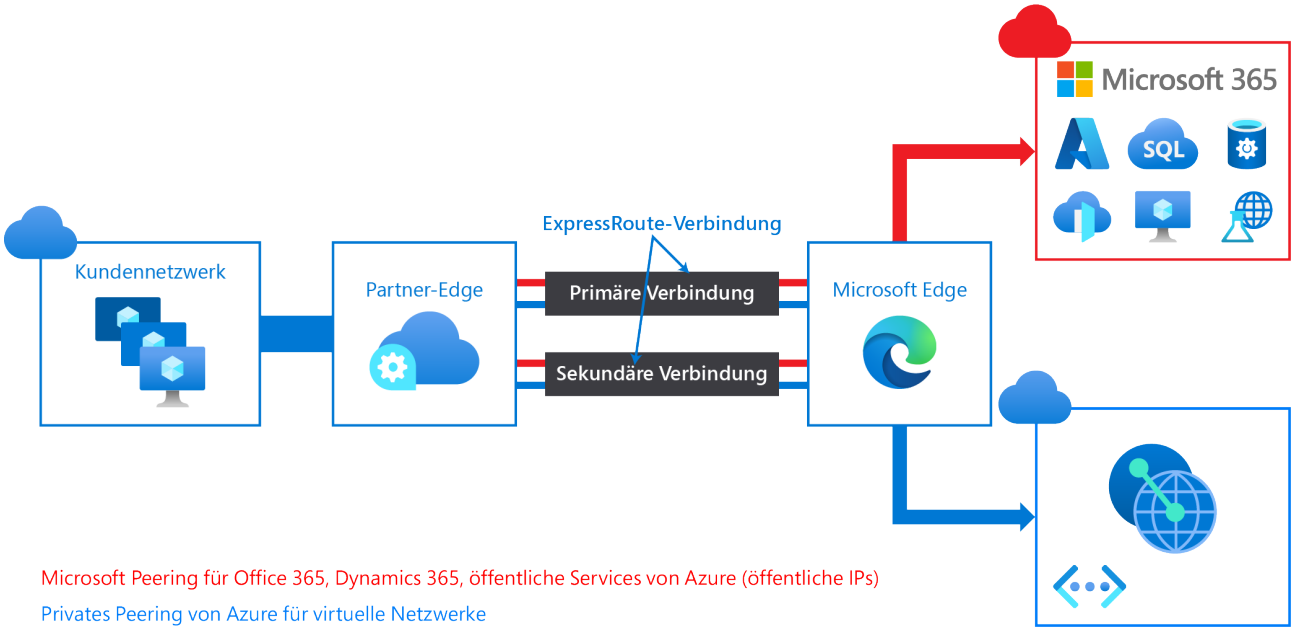
Glossar

Dieses Glossar stellt eine hilfreiche Schnellreferenz für einige der in diesem Guide und im Azure Red Hat OpenShift-IT-Ökosystem verwendete Begriffe dar. Die Begriffe sind alphabetisch geordnet.

Azure ExpressRoute

ExpressRoute ermöglicht die Erweiterung Ihrer lokalen Netzwerke in die Microsoft-Cloud über eine private Verbindung und der Hilfe eines Konnektivitätsanbieters. Mit ExpressRoute können Sie Verbindungen zu Microsoft-Cloud-Services, wie Microsoft Azure und Microsoft 365, einrichten.

Das folgende ExpressRoute-Netzwerkdiagramm ist aus der Microsoft Azure-Dokumentation entnommen:



Weitere Informationen zu ExpressRoute finden Sie auf der Seite Was ist Azure ExpressRoute? der Microsoft Azure-Dokumentation.

Eine Erklärung der Funktionsweise von ExpressRoute mit Azure finden Sie in *Kapitel 4, Vor der Provisionierung – Fragen zur Unternehmensarchitektur.*

Azure-Zielzonen

Azure-Zielzonen sind ein populäres Deployment Pattern für Organisationen, die ein umfangreiches Deployment mit Azure und unter Berücksichtigung von Aspekten zu Skalierbarkeit, Sicherheit, Governance, Netzwerk und Identität planen.

Was ist eine Azure-Zielzone?

Zum Zeitpunkt der Erstellung dieses Guide befand sich ein Projekt auf GitHub in der Entwicklung, das Empfehlungen zum Deployment von Azure Red Hat OpenShift in einer Azure-Zielzonenarchitektur gibt: https://github.com/Azure/Enterprise-Scale/tree/main/workloads/ARO

Builds und Image Streams

Ein Build ist die Transformation von Eingabeparametern in ein resultierendes Objekt. Am häufigsten dient der Prozess zum Transformieren von Eingabeparametern oder Quellcode in ein ausführbares Image. Ein BuildConfig-Objekt ist die Definition des gesamten Build-Prozesses.

Azure Red Hat OpenShift nutzt Kubernetes, indem Docker-formatierte Container aus Build Images erstellt und in eine Container Image Registry übergeben werden.

Build-Objekte haben gemeinsame Merkmale: Eingaben für einen Build, erforderlicher Abschluss eines Build-Prozesses, Protokollierung des Build-Prozesses, Veröffentlichung von Ressourcen aus erfolgreichen Builds und Veröffentlichung des finalen Status des Build. Builds nutzen Ressourceneinschränkungen, indem sie Einschränkungen für Ressourcen wie CPU- und Arbeitsspeichernutzung sowie Build- oder Pod-Ausführungszeit angeben.

Das Azure Red Hat OpenShift-Build-System bietet erweiterbare Unterstützung für Build-Strategien, die auf auswählbaren, in der Build-API angegebenen Typen basieren. Es gibt drei primäre Build-Strategien:

- **Docker Build** – es wird ein Dockerfile verwendet, das hochgeladen oder aus einem Quell-Repository abgerufen werden kann
- **Source-to-Image (S2I) Build** – verwendet ein Quell-Repository (wie Git) und legt fest, wie die Anwendung aus bekannten Sprach-Build-Dateien (z. B. Maven `.pom`-Dateien für Java-Projekte) erzeugt werden sollen
- **Benutzerdefinierter Build**

Standardmäßig werden Docker-Builds und S2I-Builds unterstützt.

Das resultierende Objekt eines Build hängt vom Builder für die Erstellung ab. Für Docker- und S2I-Builds sind die resultierenden Objekte ausführbare Images. Für benutzerdefinierte Builds sind die resultierenden Objekte, was der Autor des Builder Image angegeben hat.

Container

Die grundlegenden Einheiten von Azure Red Hat OpenShift-Anwendungen heißen Container. Bei Linux-Containertechnologien handelt es sich um schlanke Mechanismen zur Isolierung ausgeführter Prozesse, sodass sie auf die reine Interaktion mit ihren designierten Ressourcen beschränkt sind.

Viele Anwendungsinstanzen können in Containern auf einem einzelnen Host ohne gegenseitige Transparenz der Prozesse, Dateien, Netzwerke usw. ausgeführt werden. In der Regel bietet jeder Container einen einzigen Service (oft als „Microservice“ bezeichnet), wie einen Webserver oder eine Datenbank, obwohl Container für beliebige Workloads genutzt werden können.

Container Images

Container in Azure Red Hat OpenShift basieren auf Docker-formatierten Container Images. Ein Image ist eine Binärdatei, die alle Anforderungen zur Ausführung eines einzelnen Containers sowie Metadaten mit einer Beschreibung der Anforderungen und Fähigkeiten beinhaltet.

Es handelt sich um eine Art Packaging-Technologie. Container haben nur Zugriff auf im Image definierte Ressourcen, es sei denn, Sie gewähren dem Container bei der Erstellung zusätzlichen Zugriff. Durch Deployment desselben Image in mehreren Containern auf mehreren Hosts und Load Balancing zwischen ihnen kann Azure Red Hat OpenShift Redundanz und horizontale Skalierung für einen Service in einem Image bieten.

Container Registry

Azure Red Hat OpenShift bietet eine integrierte Container Image Registry namens **OpenShift Container Registry (OCR)**, die die automatische Provisionierung neuer Image Repositorys nach Bedarf ermöglicht. Dies stellt Nutzern einen integrierten Ort für ihre Anwendungsbuilds zur Übergabe der resultierenden Images bereit.

Wenn ein neues Image an die OCR übergeben wird, benachrichtigt die Registry Azure Red Hat OpenShift über das neue Image. Dabei werden alle Informationen über es weitergeleitet, wie Namespace, Name und Image-Metadaten. Verschiedene Teile von Azure Red Hat OpenShift reagieren auf neue Images, wobei neue Builds und Deployments erstellt werden.

Azure Red Hat OpenShift kann auch jeden beliebigen Server nutzen, der die Container Image Registry-API als Quelle von Images implementiert, darunter Docker Hub und Azure Container Registry.

Deployments und Deployment-Konfigurationen

Azure Red Hat OpenShift setzt auf Replication Controllern auf und fügt erweiterte Unterstützung für Softwareentwicklung und Deployment Lifecycle durch das Konzept der Deployments hinzu. Im einfachsten Fall erstellt ein Deployment nur einen neuen ReplicationController und lässt ihn Pods hochfahren. Azure Red Hat OpenShift-Deployments ermöglichen auch die Transition von einem vorhandenen Deployment eines Image zu einem neuen und definieren zudem Hooks, die vor oder nach der Erstellung des ReplicationController ausgeführt werden.

Das Azure Red Hat OpenShift DeploymentConfig-Objekt definiert die folgenden Details eines Deployment:

1. Elemente einer ReplicationController-Definition
2. Trigger zur automatischen Erstellung eines neuen Deployment
3. Strategie für die Transition zwischen Deployments
4. Lifecycle Hooks

Jedes Mal, wenn ein Deployment ausgelöst wird, sei es manuell oder automatisch, managt ein Deployer Pod das Deployment (einschließlich Skalierung des alten ReplicationController nach unten, Skalieren des neuen ReplicationController nach oben und Ausführen von Hooks). Der Deployment Pod bleibt für einen unbestimmten Zeitraum nach Abschluss des Deployment bestehen, um die Deployment-Protokolle beizubehalten. Wird ein Deployment durch ein anderes ersetzt, wird der vorherige ReplicationController beibehalten, um im Bedarfsfall ein einfaches Rollback zu ermöglichen.

Detaillierte Anweisungen zum Erstellen von und Interagieren mit Deployments finden Sie unter Deployments and DeploymentConfigs.

Jobs

Ein Job ähnelt einem ReplicationController dahingehend, dass sein Zweck, die Erstellung von Pods aus bestimmten Gründen ist. Der Unterschied liegt darin, dass Replication Controller für Pods designt sind, die fortlaufend ausgeführt werden, wohingegen Jobs für einmalige Pods konzipiert wurden. Ein Job verfolgt alle erfolgreichen Abschlüsse nach. Wenn die angegebene Menge an Abschlüssen erreicht wurde, ist der Job selbst abgeschlossen.

Im Thema *Jobs* finden Sie weitere Informationen zur Verwendung von Jobs.

Pods und Services

Azure Red Hat OpenShift nutzt das Kubernetes-Konzept eines Pod. Hier handelt es sich um einen oder mehrere Container, die auf einem Host gemeinsam bereitgestellt werden, und die kleinste Recheneinheit, die definiert, bereitgestellt und gemanagt werden kann.

Pods sind das ungefähre Äquivalent einer Rechnerinstanz (physisch oder virtuell) zu einem Container. Jedem Pod ist eine eigene interne IP-Adresse zugeordnet und er ist daher im Besitz des gesamten Port-Platzes. Container in Pods können ihren lokalen Storage und Netzwerke teilen.

Pods haben einen Lifecycle. Sie werden definiert, für die Ausführung auf einem Node zugewiesen, ausgeführt, bis ihre Container beendet oder aus einem anderen Grund entfernt werden. Pods können, abhängig von der Richtlinie und dem Exit-Code, nach dem Beenden entfernt werden. Oder sie werden beibehalten, um Zugriff auf die Protokolle ihrer Container zu ermöglichen.

Azure Red Hat OpenShift behandelt Pods weitgehend als unveränderbar. Änderungen an einer Pod-Definition sind während der Ausführung nicht möglich. Azure Red Hat OpenShift implementiert Änderungen durch Beenden und Neuerstellen eines vorhandenen Pod mit einer modifizierten Konfiguration, modifizierten Basis-Images oder beidem. Die Runtime-Komponenten eines Pod werden als erweiterbar behandelt und aus dem definierten Container Image neu erstellt. Daher sollten Pods in der Regel von Kontrollpersonen auf höherer Ebene gemanagt werden, statt direkt von Nutzern.

Projekte und Nutzer

Ein Projekt ist ein Kubernetes-Namespace mit zusätzlichen Anmerkungen und das zentrale Mittel, mit dem der Zugriff auf Ressourcen für normale Nutzer gemanagt wird. Ein Projekt ermöglicht es einer Community aus Nutzern, Inhalte von anderen Communitys isoliert zu organisieren und zu managen. Nutzer müssen Zugriff auf Projekte durch Administratoren oder, wenn sie Projekte erstellen dürfen, automatisch Zugriff auf ihre eigenen Projekte erhalten. „name“, `displayName` und „description“ von Projekten können sich unterscheiden.

„name“ ist obligatorisch, eine eindeutige ID für das Projekt und am häufigsten bei der Verwendung der CLI-Tools oder API präsent. Der Name darf maximal 63 Zeichen lang sein. `displayName` ist optional und der Anzeigename in der Webkonsole (standardmäßig `name`). „description“ ist optional, kann eine detailliertere Beschreibung des Projekts sein und ist auch in der Webkonsole sichtbar.

Entwickler und Administratoren können mit Projekten unter Verwendung der CLI oder Webkonsole interagieren.

ReplicaSets

Ähnlich wie ein ReplicationController stellt ein ReplicaSet sicher, dass eine angegebene Anzahl von Pod-Replikaten jederzeit ausgeführt wird. Der Unterschied zwischen einem ReplicaSet und einem ReplicationController ist, dass ein ReplicaSet auf dem Set basierende Selector-Anforderungen unterstützt, wohingegen ein ReplicationController nur auf Gleichheit basierende Selector-Anforderungen unterstützt.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
    name: frontend-1
    labels:
        tier: frontend
spec:
    replicas: 3
    selector:
        matchLabels:
            tier: frontend
        matchExpressions:
        - {key: tier, operator: In, values: [frontend]}
    template:
        metadata:
            labels:
                tier: frontend
        spec:
            containers:
                - image: openshift/hello-openshift
                  name : helloworld
                  ports:
                  - containerPort: 8080
                    protocol: TCP
            restartPolicy: Always
```

Der obige Code zeigt Folgendes:

- Label-Abfrage für ein Set von Ressourcen. Die Ergebnisse von `matchLabels` und `matchExpressions` sind logisch verbunden.
- Auf Gleichheit basierender Selector zum Angeben von Ressourcen mit Labels, die dem Selector entsprechen.
- Set-basierter Selector zum Filtern von Schlüsseln. Dadurch werden alle Ressourcen mit `key = tier` und `value = frontend` ausgewählt.

ReplicationController

Ein ReplicationController stellt sicher, dass eine angegebene Anzahl von Replikaten eines Pod jederzeit ausgeführt wird. Wenn Pods beendet oder gelöscht werden, agiert der ReplicationController, um weitere zu instanziieren (bis zur definierten Anzahl). Wenn mehr als gewünscht ausgeführt werden, löscht er gleichermaßen so viele wie nötig, um der definierten Menge zu entsprechen.

Eine ReplicationController-Konfiguration besteht aus:

- Anzahl der gewünschten Replikate (Anpassung zur Laufzeit möglich)
- Pod-Definition, die bei Erstellung eines replizierten Pod verwendet wird
- Selector zur Identifizierung gemanagter Pods
- Ein Selector ist ein Set mit zu Pods zugewiesenen Labels, die vom ReplicationController gemanagt werden. Diese Labels sind in der Pod-Definition enthalten, die der ReplicationController instanziiert. Der ReplicationController verwendet den Selector, um zu bestimmen, wie viele Instanzen des Pod bereits ausgeführt werden, um nach Bedarf Anpassungen vorzunehmen.

Der ReplicationController führt keine automatische Skalierung basierend auf Last oder Datenverkehr durch, da er beides nicht nachverfolgt. Stattdessen würde dies eine Anpassung der Replikatanzahl durch einen externen Auto-Scaler erfordern.

Ein ReplicationController ist ein zentrales Kubernetes-Objekt. Nachfolgend finden Sie ein Beispiel für eine ReplicationController-Definition:

```
apiVersion: v1
kind: ReplicationController
metadata:
    name: frontend-1
spec:
    replicas: 1
    selector:
        name: frontend
    template:
        metadata:
            labels:
                name: frontend
        spec:
            containers:
                - image: openshift/hello-openshift
                  name: helloworld
                  ports:
                - containerPort: 8080
                  protocol: TCP
            restartPolicy: Always
```

Der obige Code zeigt Folgendes:

- Anzahl der auszuführenden Kopien des Pod.
- Auszuführender Label Selector des Pod.
- Vorlage für den Pod, den der Controller erstellt.
- Labels auf dem Pod sollten die vom Label Selector umfassen.
- Die maximale Namenslänge nach Erweiterung beliebiger Parameter sind 63 Zeichen.

Routen und Ingresses

Azure Red Hat OpenShift unterstützt Routen und Ingresses. Beide werden verwendet, um einen Service über einen DNS-Namen, wie www.example.com, verfügbar zu machen, damit er für externe Clients zugänglich ist.

Routen wurden ursprünglich in Red Hat OpenShift Version 3 konzipiert. Seit diesem Release hat Red Hat mit der Kubernetes-Community an der Standardisierung dieser Funktionalität zu dem, was jetzt **Ingress** genannt wird, gearbeitet.

Die Ressource Kubernetes Ingress in OpenShift Container Platform implementiert den Ingress Controller mit einem geteilten Router-Service, der als Pod im Cluster ausgeführt wird. Der übliche Weg für das Management von Ingress-Datenverkehr erfolgt mit dem Ingress Controller. Sie können diesen Pod wie alle anderen regulären Pods skalieren und replizieren. Die Router-Service basiert auf HAProxy, einer Open-Source-Load-Balancer-Lösung.

Die OpenShift Container Platform-Route stellt Ingress-Datenverkehr für Services im Cluster bereit. Routen bieten erweiterte Funktionen, die möglicherweise von standardmäßigen Kubernetes Ingress Controller nicht unterstützt werden, wie TLS-Neuverschlüsselung, TLS Passthrough und aufgeteilter Datenverkehr für Blue/Green Deployments.

Ingress-Datenverkehr greift auf Services im Cluster über eine Route zu. Routen und Ingresses sind die Hautressourcen bei der Behandlung von Ingress-Datenverkehr. Ein Ingress bietet ähnliche Funktionen wie eine Route, wie z. B. externe Anfragen akzeptieren und sie basierend auf der Route löschen. Mit einem Ingress können Sie allerdings nur bestimmte Verbindungstypen zulassen: HTTP/2, HTTPS und **Server Name Identification (SNI)** sowie TLS mit einem Zertifikat. In OpenShift Container Platform werden Routen generiert, um die von der Ingress-Ressource festgelegten Bedingungen zu erfüllen.

Source-to-Image (S2I)

S2I ist ein Toolkit und Workflow zum Erzeugen reproduzierbarer Container Images aus Quellcode. S2I erzeugt ausführungsbereite Images. Dabei wird Quellcode in ein Container Image injiziert. Der Container bereitet dann den Quellcode zur Ausführung vor. Durch Erstellung selbstorganisierender Builder Images können Sie Ihre Build-Umgebungen versionieren und kontrollieren – genau wie bei der Verwendung von Container Images zur Versionierung Ihrer Runtime-Umgebungen.

Bei einer dynamischen Sprache wie Ruby sind die Build-Zeit- und Runtime-Umgebungen in der Regel gleich. Beginnend mit einem Builder Image, das diese Umgebung beschreibt – mit Ruby, Bundler, Rake, Apache, GCC und anderen Paketen zum Einrichten und Ausführen einer installierten Ruby-Anwendung –, führt S2I die folgenden Schritte durch:

1. Ein Container wird aus dem Builder Image gestartet, wobei die Anwendungsquelle in ein bekanntes Verzeichnis injiziert ist.
2. Der Container-Prozess transformiert diesen Quellcode in das entsprechende ausführbare Setup – in diesem Fall durch Installieren von Abhängigkeiten mit Bundler und Verschieben des Quellcodes in ein Verzeichnis, in dem Apache zum Suchen nach der Ruby-Datei `config.ru` vorkonfiguriert wurde.
3. Der neue Container wird commitet und der Image-Einstiegspunkt auf ein (vom Builder Image bereitgestelltes) Skript festgelegt, das Apache zum Hosten der Ruby-Anwendung startet.

Für kompilierte Sprachen wie C, Go oder Java gleichen die Abhängigkeiten für die Kompilierung evtl. erheblich die Größe der tatsächlichen Runtime-Artefakte aus. Um Runtime Images schlank zu halten, ermöglicht S2I Build-Prozesse mit mehreren Schritten. Bei diesen wird ein binäres Artefakt wie eine ausführbare oder Java-WAR-Datei im ersten Builder Image erstellt, extrahiert und in ein zweites Runtime Image injiziert, das die ausführbare Datei einfach am richtigen Ausführungsort platziert.

Um beispielsweise eine reproduzierbare Build-Pipeline für Tomcat (beliebter Java-Webserver) und Maven zu erstellen, führen Sie die folgenden Schritte aus:

1. Erstellen Sie ein Builder Image mit OpenJDK und Tomcat, in das eine WAR-Datei injiziert werden muss.
2. Erstellen Sie ein zweites Image auf dem ersten Image (Maven und beliebige andere Standardabhängigkeiten), in das ein Maven-Projekt injiziert werden muss.
3. Rufen Sie S2I unter Verwendung der Java-Anwendungsquelle und des Maven-Image auf, um die gewünschte Anwendungs-WAR zu erstellen.
4. Rufen Sie S2I erneut unter Verwendung der WAR-Datei aus dem vorherigen Schritt und des anfänglichen Tomcat-Image zur Erstellung des Runtime Image auf.

Durch Platzierung unserer Build-Logik in Images und Kombination der Images in mehreren Schritten bleiben Runtime- und Build-Umgebung eng beieinander (selbes JDK, selbe Tomcat-JARs) ohne Deployment von Build-Tools in der Produktion.

Die Ziele und Vorteile der Verwendung von S2I als Build-Strategie sind:

- **Reproduzierbarkeit**: Ermöglichen Sie, dass Build-Umgebungen eng versioniert werden. Kapseln Sie sie dazu in einem Container Image, und definieren Sie eine einfache Schnittstelle (injizierter Quellcode) für Aufrufer. Reproduzierbare Builds sind eine zentrale Anforderung für Sicherheitsupdates und Continuous Integration in einer containerisierten Infrastruktur. Builder Images stellen Wiederholbarkeit sowie das Wechseln zwischen Runtimes sicher.
- **Flexibilität**: Jedes vorhandene Build-System, das auf Linux ausgeführt werden kann, kann in einem Container ausgeführt werden. Jeder individuelle Builder kann auch Teil einer größeren Pipeline sein. Zudem lassen sich die Skripts, die den Quellcode der Anwendung verarbeiten, in das Builder Image injizieren. Dadurch können Autoren vorhandene Images anpassen, um Quellbearbeitung zu ermöglichen.
- **Geschwindigkeit**: Statt mehrere Layer in einem einzigen Dockerfile zu erzeugen, ermutigt S2I Autoren, eine Anwendung in einem einzigen Image Layer zu repräsentieren. Dies spart Zeit bei Erstellung und Deployment und ermöglicht eine bessere Kontrolle über die Ausgabe des finalen Image.
- **Sicherheit**: Builds mit Dockerfiles werden ohne viele der normalen operativen Kontrollelemente von Containern ausgeführt, in der Regel als Root und mit Zugriff auf das Container-Netzwerk. Mit S2I kann kontrolliert werden, welche Rechte und Berechtigungen für das Builder Image verfügbar sind, da der Build in einem einzelnen Container gestartet wird. Gemeinsam mit Plattformen wie OpenShift kann S2I Administratoren die strenge Kontrolle der Entwicklerberechtigungen zur Build-Zeit ermöglichen.

Vorlagen

Eine Vorlage beschreibt ein Set von Objekten, die parametrisiert und verarbeitet werden können, um eine Liste von Objekten zur Erstellung durch Azure Red Hat OpenShift zu erzeugen. Eine Vorlage kann verarbeitet werden, um alle Elemente zu erstellen, zu deren Erstellung Sie in einem Projekt berechtigt sind, beispielsweise Services sowie Build- und Deployment-Konfigurationen. Eine Vorlage kann auch ein Set von Labels definieren, die auf jedes in der Vorlage definierte Objekt angewendet werden können.

Sie können eine Liste von Objekten aus einer Vorlage mithilfe der CLI oder, wenn eine Vorlage in Ihr Projekt oder die globale Vorlagen-Library hochgeladen wurde, mithilfe der Webkonsole erstellen. Ein kuratiertes Set von Vorlagen finden Sie in der OpenShift-Library für Image Streams und Vorlagen.