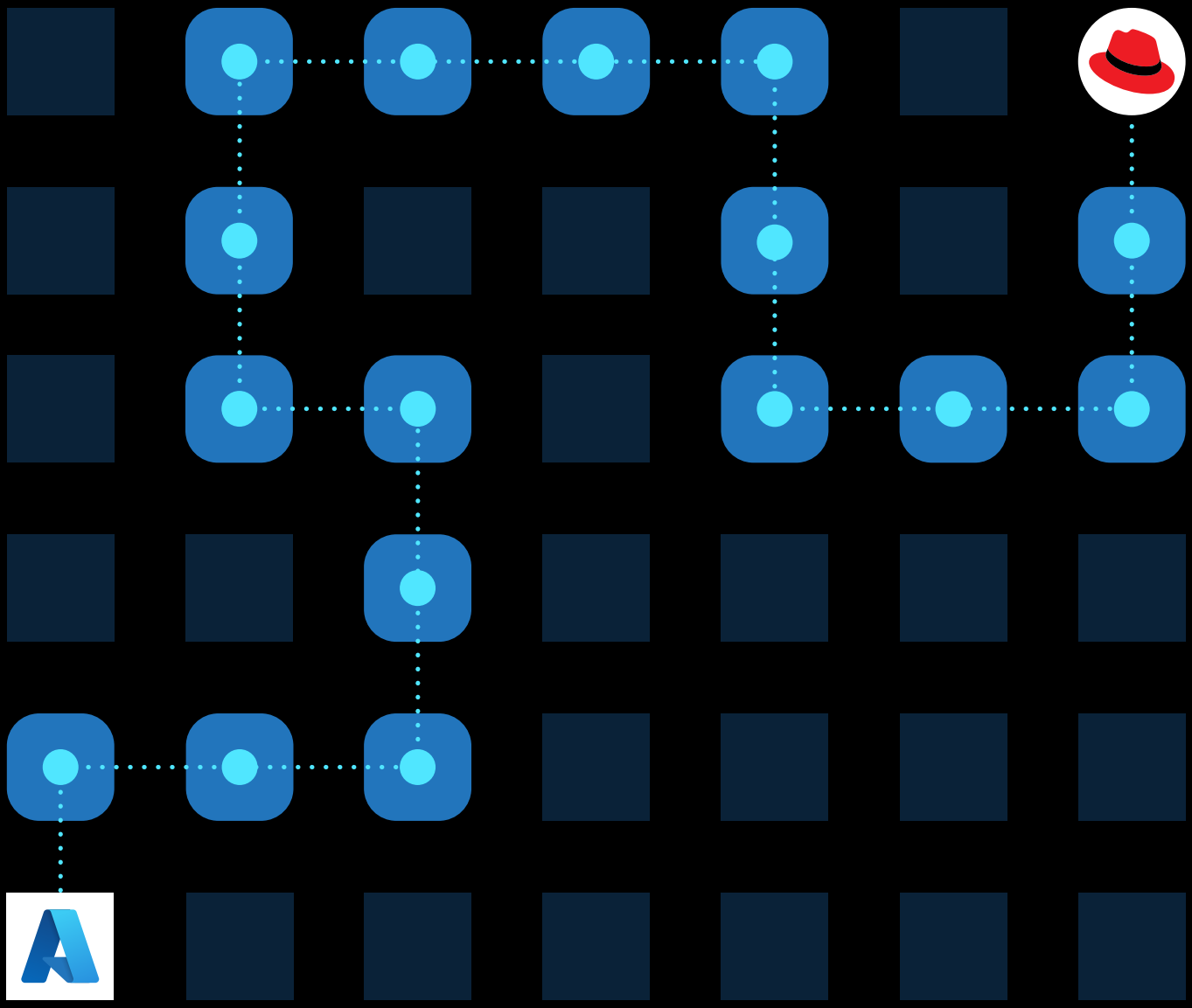
Série de ebooks

Microsoft Azure

Introdução ao Azure Red Hat OpenShift

Da prova de conceito à produção



Introdução ao Azure Red Hat OpenShift

Capítulo 1

Fale com a Microsoft e a Red Hat

Se tiver interesse no Azure Red Hat OpenShift, queremos conversar com você. Embora este guia dê um direcionamento útil no uso da plataforma, a Red Hat e a Microsoft podem oferecer diversos recursos que vão além do guia e ajudam você a superar os limites da experiência. Juntos, queremos assegurar que o Azure Red Hat OpenShift é a plataforma de aplicações que atenda às suas necessidades de inovação de aplicações.

O Azure Red Hat OpenShift foi criado por um único motivo: clientes como você pediram por isso. Mais do que nunca, nossos clientes em comum (empresas e pequenas organizações) estão implantando o portfólio da Red Hat no Microsoft Azure. Embora o OpenShift no Azure como uma oferta autogerenciada tenha tido suporte integral por muitos anos, a configuração, implantação e operações de segunda etapa no gerenciamento do cluster exigem experiência em Kubernetes e tempo para gerenciar. Isso ocupa um tempo importantíssimo que poderia ser empregado nas metas de negócios.

À medida que mais clientes realizam implantações bem-sucedidas com a oferta gerenciada, o Azure Red Hat OpenShift, vemos que estão passando muito menos tempo na configuração e operações de segunda etapa e mais tempo concentrados em suas aplicações.

Criamos este guia com base em nossa experiência real para oferecer aos nossos clientes as melhores práticas para criar aplicações no Azure Red Hat OpenShift. Ele é um guia de apoio individual. Você pode ler os capítulos em sequência (da introdução à conclusão) ou apenas buscar as informações específicas que precisa.

Para quem é este guia

Ele é destinado para profissionais técnicos: desenvolvedores, operadores e arquitetos de plataforma, que estão buscando melhorar os recursos de implantação e criação de aplicações usando o Azure e o Red Hat OpenShift para a implantação completa de serviços de cluster totalmente gerenciados do Red Hat OpenShift.

O que este guia aborda

Neste guia, falamos sobre os principais tópicos para entender e adotar o Azure Red Hat OpenShift, desde a prova de conceitos dentro da organização até a implantação da produção.

Capítulo 2: *Introdução ao Red Hat OpenShift* começa apresentando o Red Hat OpenShift e os motivos de tantos desenvolvedores, operadores e arquitetos de plataforma o escolherem como plataforma de aplicações, bem como suas muitas utilidades. Em seguida, você verá por que o Red Hat OpenShift é a plataforma mais escolhida.

Capítulo 3: *Azure Red Hat OpenShift* explica o serviço gerenciado e como ele é oferecido a você como cliente.

Capítulo 4: *Pré-provisionamento: questões sobre arquitetura empresarial* aborda pontos importantes que você deve considerar antes de implantar o Azure Red Hat OpenShift. Isso inclui muita orientação sobre as melhores práticas que aprendemos conversando com muitos clientes que já o implantaram.

Capítulo 5: *Provisionamento de um cluster do Azure Red Hat OpenShift* aponta os recursos essenciais para implantar um cluster do Azure Red Hat OpenShift na documentação oficial.

Capítulo 6: *Pós-provisionamento (segunda etapa)* aborda as tarefas de pós-provisionamento, normalmente chamadas de "segunda etapa". Sempre que possível, este guia explica como o serviço gerenciado, o Azure Red Hat OpenShift, torna isso mais fácil do que se você tivesse que abordar esses tópicos por conta própria.

Capítulo 7: *Implantação de aplicações no Red Hat OpenShift* é um breve guia para implantar aplicações na plataforma, mas é importante lembrar que isso não é diferente da execução do Red Hat OpenShift em qualquer outro ambiente.

Capítulo 8: *Explorando a plataforma de aplicações* oferece uma visão geral guiada de algumas funcionalidades importantes da plataforma de aplicações: Container Registry, Pipelines, serverless e outros.

Capítulo 9: *Integração com outros serviços* aborda a integração da plataforma usando o Azure Service Operator, Azure DevOps e serviços semelhantes.

Capítulo 10: *Processo de integração de cargas de trabalho e equipes* encerra com algumas das melhores práticas e recomendações sobre como integrar equipes de aplicações e ganhar força na adoção do serviço na sua organização.

Capítulo 11: *Conclusão* contém a conclusão.

Capítulo 12: *Glossário* é um glossário.

Capítulo 2

Introdução ao Red Hat OpenShift

Este capítulo contém uma breve introdução sobre o que é o Red Hat OpenShift, como ele é usado e quais são os benefícios do serviço geralmente relatados pelos clientes da Red Hat. Serve como um pontapé inicial para quem está conhecendo o OpenShift pela primeira vez ou como uma revisão rápida para quem quer saber mais sobre a plataforma.

Visão geral do Red Hat OpenShift

O Red Hat OpenShift é uma plataforma de aplicações pronta para uso empresarial com operações automatizadas de stack completo para gerenciamento de nuvem híbrida e implantações multicloud. Ele é otimizado para aumentar a produtividade do desenvolvedor e promover a inovação. Com o gerenciamento otimizado do ciclo de vida e operações automatizadas, o Red Hat OpenShift capacita as equipes de desenvolvimento a criar e implantar novas aplicações e ajuda as equipes de operações a provisionar, gerenciar e escalar plataformas do Kubernetes.

As equipes de desenvolvimento têm acesso a soluções e imagens validadas de centenas de parceiros com verificação de segurança e assinatura criptografada em todo o processo de entrega. Elas podem acessar imagens sob demanda e ter acesso nativo a uma grande variedade de serviços em nuvem de terceiros, tudo em uma única plataforma.

As equipes de operações dão visibilidade às implantações, não importa onde estejam, e entre equipes, com monitoramento e geração de logs integrados. Os operadores do Kubernetes da Red Hat integraram uma lógica de aplicações exclusiva que permite que o serviço seja funcional, não apenas configurado, mas também aprimorado para melhor desempenho, além de contar com atualização e aplicação de patches do sistema operacional com um único toque. Em resumo, o Red Hat OpenShift tem tudo o que uma organização precisa para liberar o potencial da TI e das equipes de desenvolvimento.

Serviços em nuvem do OpenShift: economias e benefícios para os negócios

Milhares de clientes confiam no Red Hat OpenShift para alterar o modo como entregam aplicações, melhorar o relacionamento com os clientes e ter vantagem competitiva para serem líderes em seus setores. Uma pesquisa da IDC, The Business Value of Red Hat OpenShift, março de 2021, demonstra o grande valor que as organizações entrevistadas alcançaram com a plataforma Red Hat OpenShift. Elas entregam aplicações e funcionalidades de maior qualidade e em tempo mais hábil aos seus negócios e otimizam os custos de desenvolvimento e relacionados a TI e os requisitos de tempo da equipe.

As métricas principais são as seguintes:

- 636% de retorno sobre o investimento em cinco anos
- Retorno financeiro em 10 meses
- Redução de 54% no custo operacional de cinco anos
- Três vezes mais funcionalidades por ano
- Aumento de 20% na produtividade do desenvolvedor de aplicações
- Redução de 71% no downtime não planejado
- 21% de aumento na eficiência para equipes de infraestrutura de TI

Fonte: IDC whitepaper, patrocinado pela Red Hat. *The Business Value of Red Hat OpenShift*, doc #US47539121, março de 2021

Baseado no valor do Red Hat OpenShift, os serviços de nuvem do Red Hat OpenShift, que incluem o Azure Red Hat OpenShift, entregam benefícios adicionais para os negócios. O estudo da Forrester intitulado The Total Economic Impact™ of Red Hat OpenShift Cloud Services - Cost Savings and Business Benefits destacou diversos benefícios financeiros importantes.

Os principais benefícios financeiros dos serviços em nuvem do OpenShift são os seguintes:

- 468% de retorno sobre o investimento (ROI)
- Valor presente líquido (VPL) de US$ 4,08 milhões
- Retorno financeiro em seis meses

Além desses benefícios financeiros, estas são algumas descobertas importantes sobre os benefícios quantificados no relatório da Forrester:

- **Aumento na velocidade do desenvolvimento**: o uso dos serviços em nuvem do Red Hat OpenShift permite que as organizações reduzam o ciclo de desenvolvimento em até 70%.
- **20% do tempo do desenvolvedor é recuperado do trabalho de manutenção da infraestrutura**: os entrevistados apontaram que os serviços em nuvem do Red Hat OpenShift eliminaram a necessidade de os desenvolvedores manterem a infraestrutura de desenvolvimento de aplicações, o que permitiu que eles se concentrassem em criar o produto ou a solução. Ao longo de três anos, a recuperação desse tempo do desenvolvedor representou mais de US$ 2,3 milhões.
- **50% de melhoria na eficiência operacional**: como os serviços em nuvem do Red Hat OpenShift são gerenciados, os entrevistados apontaram que usar a solução significa que eles podem redistribuir 50% dos funcionários de DevOps, que antes eram responsáveis por gerenciar a infraestrutura, para outro trabalho mais produtivo. Ao longo de três anos, esse aumento na eficiência operacional representou mais de US$ 1,3 milhões.*

O relatório também identificou os seguintes benefícios não quantificados:

- **Retenção e satisfação do desenvolvedor**: os entrevistados destacaram que os desenvolvedores se beneficiaram dos serviços em nuvem do Red Hat OpenShift. Isso permitiu que eles dividissem as atualizações em pedaços menores, reduzindo a pressão de testes excessivos em um período de tempo muito limitado, além de reduzir a necessidade de responder a simulações uma vez em produção.
- **Segurança e risco reduzido**: os entrevistados compartilharam como os serviços em nuvem do Red Hat OpenShift automatizaram algumas funcionalidades e atualizações de segurança, o que eliminou a necessidade de manutenção manual ao mesmo tempo em que continuou a garantir a segurança do ambiente.
- **Confiabilidade**: os entrevistados notaram que o uso dos serviços em nuvem do Red Hat OpenShift tornou a plataforma de aplicações mais confiável ao longo da execução, já que houve menos interrupções ou falhas no sistema, mesmo em um ambiente em expansão.
- **Portabilidade e continuidade dos negócios**: os entrevistados também apontaram que os serviços em nuvem do Red Hat OpenShift garantiu continuidade dos negócios e ajudou com a estratégia de recuperação de desastres, devido à portabilidade, escalabilidade e flexibilidade.

Fonte: Forrester, *The Total Economic Impact of Red Hat OpenShift Cloud Services*, dezembro de 2021

***Leitura adicional:** Empresas aumentam a agilidade com os serviços em nuvem

"Red Hat OpenShift ou simplesmente Kubernetes?": o custo de criar sua própria plataforma de aplicações Kubernetes

Normalmente, o Red Hat OpenShift é chamado de "Kubernetes empresarial", mas pode ser difícil entender o que isso significa de verdade. Os clientes geralmente perguntam: "OpenShift ou simplesmente Kubernetes?" No entanto, é importante entender que o **OpenShift já usa o Kubernetes**. Na arquitetura geral do OpenShift, o Kubernetes oferece a base para criar a plataforma OpenShift e boa parte das ferramentas necessárias para executá-la.

O Kubernetes é um projeto open source extremamente importante, um dos principais projetos da Cloud Native Computing Foundation e uma tecnologia essencial para a execução de containers.

No entanto, a dúvida real que os usuários em potencial do OpenShift podem ter é: "**Posso executar minhas aplicações somente com o Kubernetes**?" Muitas organizações começam implantando o Kubernetes e descobrem que um container, ou até mesmo uma aplicação empresarial, pode estar em execução em poucos dias. No entanto, à medida que as operações de segunda etapa começam, os requisitos de segurança aumentam e mais aplicações são implantadas, algumas organizações acabam caindo na armadilha de criar sua própria **plataforma como serviço (PaaS)** com a tecnologia Kubernetes. Adicionam um controlador de entrada open source, gravam alguns scripts para se conectar aos pipelines de **integração/implantação contínuas (CI/CD)** e tentam implantar uma aplicação mais complexa. É aí que começam os problemas. Se implantar Kubernetes fosse a ponta do iceberg, a complexidade do gerenciamento de segunda etapa seria o navio afundado e escondido abaixo do mesmo iceberg na água.

Embora seja possível começar a resolver esses desafios e problemas, normalmente é necessária uma equipe de operações com várias pessoas, além de várias semanas e meses de esforço para criar e manter esta "PaaS personalizada baseada no Kubernetes". Isso gera ineficiência na organização, complexidade de suporte do ponto de vista de segurança e certificação, bem como a necessidade de desenvolver tudo do zero ao integrar equipes de desenvolvedores. Se uma organização listasse todas as tarefas para criar e manter essa plataforma Kubernetes personalizada, ou seja, Kubernetes, com todos os componentes extras necessários para executar containers com sucesso, elas seriam agrupadas da seguinte maneira:

- **Gerenciamento de cluster**: inclui a instalação de sistemas operacionais, aplicação de patches no sistema operacional, instalação do Kubernetes, configuração de redes de CNI, integração de autenticação, configuração de entrada e saída, configuração de armazenamento persistente, reforço de nós, aplicação de patches de segurança e configuração da nuvem subjacente/multicloud.
- **Serviços de aplicações**: inclui agregação de logs, verificação de integridade, monitoramento de desempenho, aplicação de patches de segurança, registro de containers e configuração do processo de teste de aplicações.
- **Integração do desenvolvedor**: inclui integração CI/CD, integração de IDE/ferramentas do desenvolvedor, integração de framework, compatibilidade de middleware, fornecimento de painéis de desempenho das aplicações e RBAC.

Embora ainda tenha muitas ações e tecnologias que poderiam ser adicionadas à lista, já que a maioria dessas atividades é essencial para qualquer organização usar containers com seriedade, a complexidade, o tempo e o esforço em apenas configurar tudo isso é insignificante para a manutenção contínua dessas partes individuais. Cada integração precisa ser integralmente testada, e cada componente e atividade terá um ciclo de lançamento, uma política de segurança e patches diferentes.

O que você ganha com o Red Hat OpenShift em comparação com somente o Kubernetes?

Quando uma organização começa a executar containers em produção, baseado em apenas Kubernetes, os componentes mencionados na seção anterior são normalmente instalados e integrados em conjunto para criar um tipo de plataforma de aplicações com seu próprio design.

O Azure Red Hat OpenShift combina todos esses componentes mencionados em uma única plataforma, facilitando as operações para as equipes de TI ao mesmo tempo em que oferece às equipes de aplicações tudo o que precisam para executar as tarefas. Todos esses tópicos serão abordados em mais detalhes posteriormente no guia mas, pensando nisso, vejamos algumas das principais diferenças entre os dois:

- **Facilidade de implantação**: implantar uma aplicação no Kubernetes pode consumir muito tempo. Envolve puxar seu código GitHub em uma máquina, iniciar um container, hospedá-lo em um registro como Docker Hub e, por fim, entender seu pipeline de CI/CD, o que pode ser bem complicado. Por outro lado, o OpenShift automatiza o trabalho pesado e o de back-end, exigindo apenas que você crie um projeto e carregue seu código.
- **Segurança**: atualmente, vemos que a maioria dos projetos de Kubernetes é desenvolvida em equipes de diversos desenvolvedores e operadores. Embora agora o Kubernetes tenha suporte para controles como RBAC e IAM, ele ainda exige configuração e definição manual, e isso leva tempo. A Red Hat e o OpenShift fizeram um ótimo trabalho em identificar as melhores práticas de segurança após anos de experiência, disponíveis para clientes prontos para uso. Basta adicionar novos usuários, e o OpenShift cuidará de tudo, como o espaçamento entre nomes e a criação de diferentes políticas de segurança.
- **Flexibilidade**: ao usar o Azure Red Hat OpenShift, você aproveita as famosas melhores práticas de implantação, gerenciamento e atualização. Todo o trabalho pesado no back-end é feito por você sem a necessidade de muito esforço, o que permite que você entregue aplicações com mais rapidez. Além de ser uma boa opção para as equipes que gostam de receber instruções e de se beneficiar de uma abordagem otimizada, com a plataforma Kubernetes, você personaliza manualmente seu pipeline de DevOps CI/CD, o que dá mais espaço para a flexibilidade e a criatividade ao desenvolver seus processos.

- **Operações diárias**: os clusters são compostos por um grupo de várias máquinas virtuais e, inevitavelmente, suas equipes de operações precisarão criar novas máquinas virtuais que devem ser adicionadas a um cluster. O processo de configuração pelo Kubernetes pode ser demorado e complexo, exigindo que scripts sejam desenvolvidos para definir coisas como autorregistro ou automação na nuvem. Com o Azure Red Hat OpenShift, provisionamento de cluster, escala e operações de upgrade são automatizadas e gerenciadas pela plataforma.
- **Gerenciamento**: embora seja possível aproveitar os painéis e as ferramentas padrão do Kubernetes que estão inclusos em qualquer distribuição, a maioria dos desenvolvedores precisa de uma plataforma mais completa e robusta. O Azure Red Hat OpenShift oferece um ótimo console web baseado nas APIs do Kubernetes e recursos para que as equipes de operações gerenciem as cargas de trabalho.

No *capítulo 8, Explorando a plataforma de aplicações*, há uma descrição guiada de muitos dos importantes recursos de valor agregado do Azure Red Hat OpenShift.

Resumo

O Azure Red Hat OpenShift está sendo a escolha de muitos clientes conjuntos da Red Hat e da Microsoft como a plataforma de aplicações preferida para adotar aplicações em containers.

No próximo capítulo, vamos explorar o Red Hat OpenShift como um serviço em nuvem, explorando a arquitetura, a integração e o gerenciamento.

Capítulo 3

Azure Red Hat OpenShift

Com **Azure Red Hat OpenShift**, você pode implantar clusters totalmente gerenciados do Red Hat OpenShift sem se preocupar em criar e gerenciar a infraestrutura para executá-la.

O Azure Red Hat OpenShift é uma plataforma de aplicações sob demanda e nativa em nuvem projetada e operada em conjunto pela Microsoft e a Red Hat, que também oferecem suporte à solução. Uma equipe especializada de **engenharia de confiabilidade de sites** (**SRE**) automatiza, escala e protege clusters do OpenShift e trabalha lado a lado para oferecer uma experiência de suporte integrada. Com o Azure Red Hat OpenShift, não há máquinas virtuais para operar e nenhuma necessidade de aplicar patches. Patches são aplicados nos nós do plano de controle e de trabalho, além de serem atualizados e monitorados em seu nome pela Red Hat e pela Microsoft. Seus clusters do Azure Red Hat OpenShift são implantados na sua subscrição do Azure e inclusos na fatura.

Entregue aplicações com mais rapidez com o Azure Red Hat OpenShift:

- Capacite os desenvolvedores para que possam inovar por meio de pipelines CI/CD integrados e, em seguida, conecte suas aplicações facilmente a centenas de serviços do Azure, como MySQL, PostgreSQL, Redis e Azure Cosmos DB.
- Substitua a complexidade e remova barreiras à produtividade com operações, configuração e provisionamento automatizados.
- Escale da maneira que quiser, onde possa iniciar um cluster altamente disponível com três nós de trabalho em alguns minutos e escale à medida que a demanda de aplicações muda, com uma opção de nós de trabalho padrão de alta CPU e alta memória.
- Operações de nível empresarial, segurança e conformidade com uma experiência de suporte integrada.

Em seguida, entraremos nos detalhes técnicos por trás de como o Red Hat OpenShift é criado e operado.

Arquitetura

O Azure Red Hat OpenShift usa serviços de infraestrutura do Azure, como máquinas virtuais, grupos de segurança de rede, contas de armazenamento e outros serviços do Azure, como a base da instalação do Red Hat OpenShift. A arquitetura do Azure é totalmente implantada na subscrição do Azure, o que facilita a integração com outros serviços que já estão ativos na sua conta.

O próprio OpenShift é baseado no Red Hat Enterprise Linux CoreOS, que hospeda a arquitetura baseada em microsserviços de unidades menores e desacopladas que funcionam juntas. O serviço `kubelet` está em execução em cada máquina virtual, que é a base do cluster do Kubernetes. O banco de dados por trás dessa arquitetura, executada no plano de controle, é o **etcd**, um armazenamento de chave/valor clusterizado e confiável.

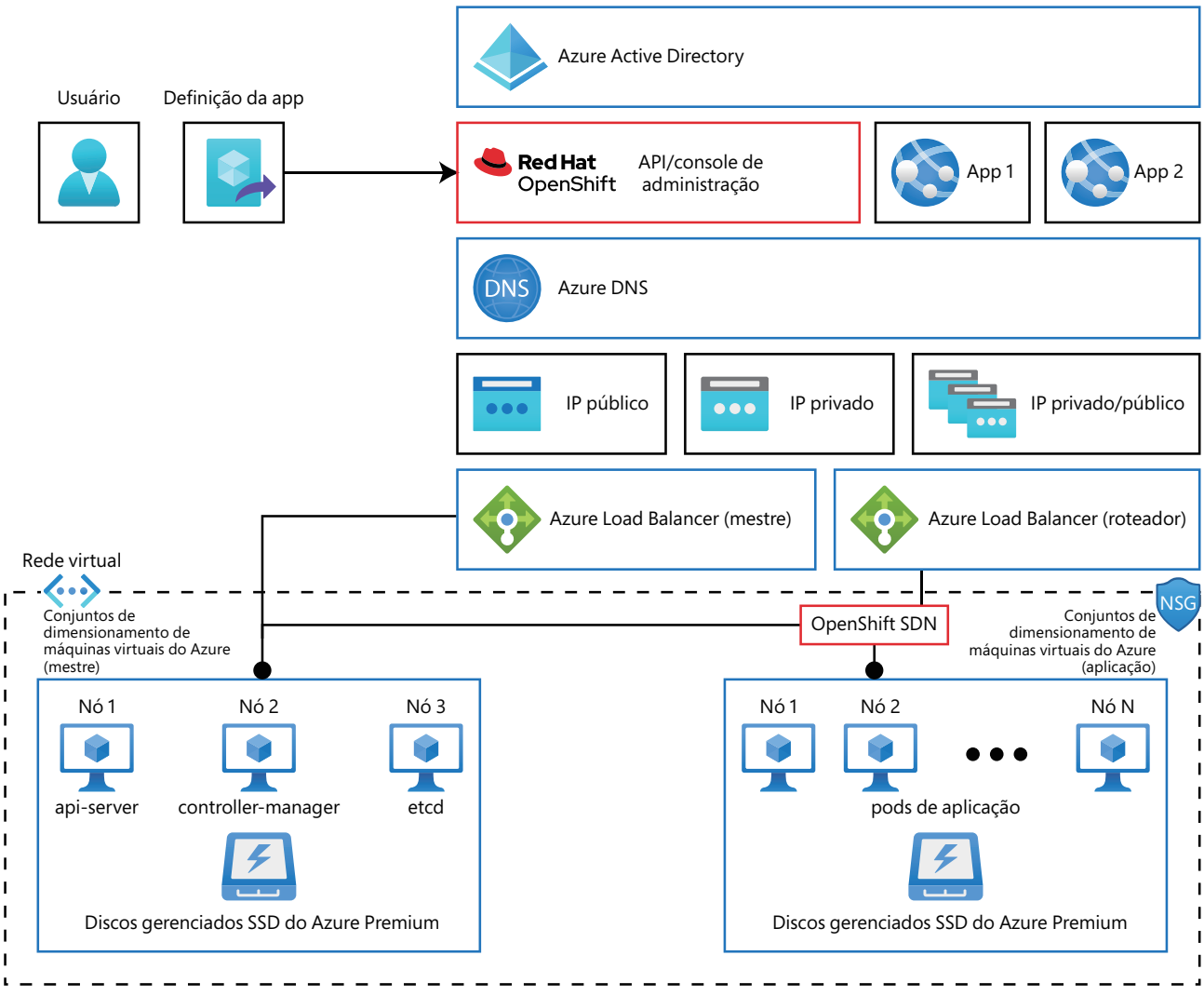


Figura 3.1: Arquitetura do Azure Red Hat OpenShift

As duas seções a seguir, *Computação: nós de controle e de trabalho* e *Redes*, dão mais detalhes sobre o que está incluso nesse diagrama.

Computação: nós de controle e de trabalho

A arquitetura do Red Hat OpenShift é executada em máquinas virtuais (nós) que têm uma das três funções: controle, infraestrutura ou aplicação. No entanto, até o momento em que este livro foi escrito, o Azure Red Hat OpenShift versão 4 não tem suporte para nós de infraestrutura. Por isso, o livro só abordará nós de controle e de trabalho.

Os nós de **controle** (historicamente chamados de nós **mestres**) são máquinas virtuais que contêm componentes essenciais para o cluster do Kubernetes. Ou seja, servidor de API, servidor gerenciador de controladores, programador e etcd. Eles gerenciam o cluster do Kubernetes e programam pods a serem executados nos nós de trabalho.

- **O servidor de API do Kubernetes** valida e configura os dados para pods, serviços e controladores de replicação. Ele também oferece um ponto de foco para o estado compartilhado do cluster.
- **O gerenciador de controladores do Kubernetes** busca mudanças do etcd de objetos, como replicação, namespace e objetos de controlador de conta de serviços, e usa a API para reforçar o estado especificado. Diversos processos como este criam um cluster com um líder ativo por vez.
- **O programador do Kubernetes** busca pods recém-criados sem um nó atribuído e seleciona o melhor nó para hospedar o pod.
- **O etcd** armazena o estado mestre persistente, enquanto os outros componentes vigiam as mudanças no etcd para entrar em um estado especificado.

Os nós de **aplicações** estão onde as aplicações serão executadas.

Cada nó em um cluster do Kubernetes executa um serviço chamado `kubelet`, que mantém a **interface do ambiente de execução de containers** (**cri-o**), um proxy de serviço e alguns outros serviços essenciais que são executados em cada nó. Todos os nós são conectados à tecnologia de rede definida por software do OpenShift. No Azure Red Hat OpenShift, ela é uma **Open Virtual Network (OVN)**, executada na rede virtual do Azure.

O Azure Red Hat OpenShift cria nós que são executados nas máquinas virtuais do Azure que, por sua vez, são conectadas aos discos do Azure para armazenamento. Você verá que elas têm discos bem grandes, 1 TB. Isso é porque, no Azure, o IOPS assegura que o desempenho do armazenamento esteja ligado ao tamanho do disco. O tamanho de 1 TB garante largura de banda suficiente para o disco subjacente usado pelo banco de dados do etcd.

Redes

O Azure Red Hat OpenShift exige uma única rede virtual do Azure, com duas sub-redes configuradas: uma para os nós de plano de controle e uma para os nós de trabalho. O tamanho mínimo das duas redes é de /27 (32 endereços). No entanto, é importante definir o tamanho para que não seja muito pequeno, caso você pretenda escalar o cluster mais tarde.

Os dois balanceadores de carga do Azure (marcados como **Mestre** e **Roteador** no diagrama) direcionam o tráfego da seguinte forma:

- Balanceador de carga de plano de controle/mestre: envia o tráfego de entrada para usuários da API do OpenShift. Ele também oferece conectividade de saída para os nós de plano de controle
- Balanceador de carga de aplicações/roteador: envia tráfego de entrada para as aplicações em execução no OpenShift, por meio de um "roteador" do OpenShift ou um controlador de entrada. Ele também oferece conectividade de saída para os nós de trabalho

Veja um excelente artigo sobre a configuração de redes, requisitos e limitações na documentação de redes.

Integração com outros serviços do Azure

Como um serviço nativo do Azure, o Azure Red Hat OpenShift pode ser implantado em conjunto e integrado com muitos dos serviços que você já usa no Azure. Veja a seguir uma visão geral de apenas alguns dos serviços de infraestrutura do Azure em que existem integrações comuns.

A *Figura 3.2* mostra muitos dos pontos de integração comuns dos serviços do Azure para o OpenShift no Azure:

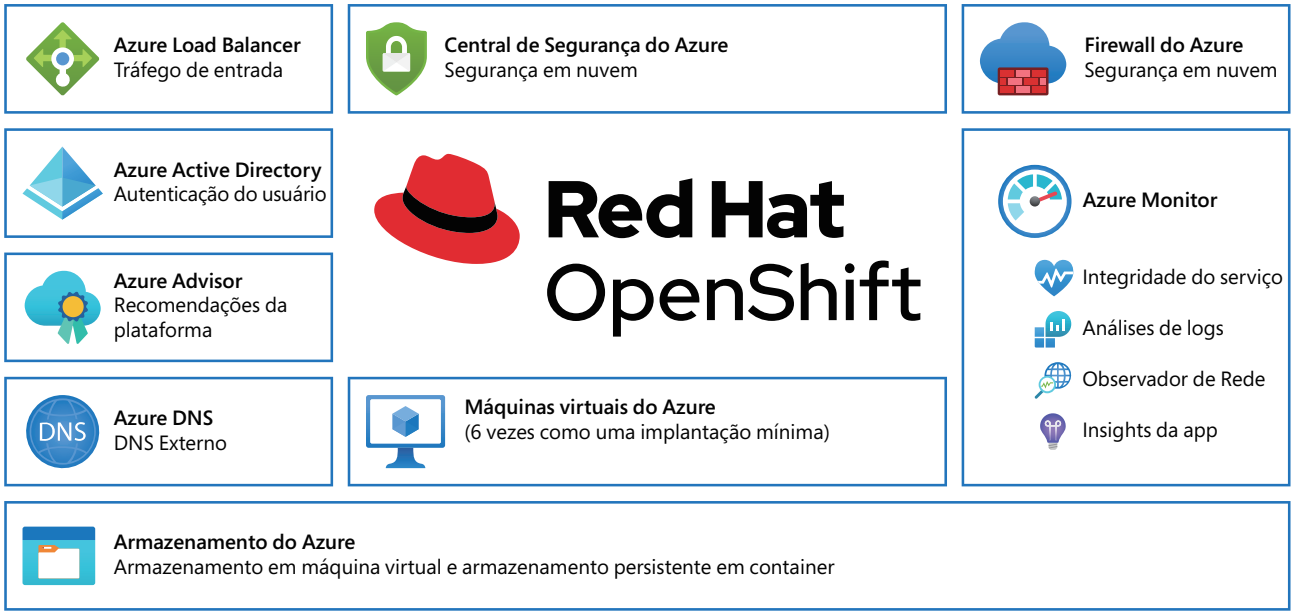


Figura 3.2: Pontos de integração comuns dos serviços do Azure

Além disso, as aplicações em execução no OpenShift podem se integrar intimamente aos serviços do Azure usando o Azure Service Operator. Este assunto é abordado em mais detalhes a seguir no *capítulo 9, Integração com outros serviços.*

Gerenciamento

Uma forma simples de entender o que é gerenciado como parte do serviço do Azure Red Hat OpenShift é pensar em tudo, desde o data center até os operadores de clusters, como sendo "gerenciado". Os operadores de clusters são serviços executados nos nós de plano de controle e cuidam do monitoramento, atualizações e a integridade do cluster. Isso significa que a Microsoft e a Red Hat, em conjunto, vão monitorar, manter e gerenciar esses componentes para manter o cluster online. Tudo que não seja os operadores de cluster continua sendo responsabilidade do cliente.

Como consumidor do Azure Red Hat OpenShift, você recebe acesso completo de **cluster-admin** ao cluster, o que significa que também compartilha a responsabilidade de não danificar partes do seu cluster. É importante entender a Política de suporte para entender o que você pode ou não fazer com o cluster.

Veja uma descrição detalhada de exatamente o que a Microsoft e a Red Hat farão como parte do serviço de nuvem em Matriz de responsabilidade do Azure Red Hat OpenShift.

Autenticação e autorização

O Azure Active Directory é uma forma comum de fornecer autenticação aos clusters do Azure Red Hat OpenShift. No entanto, não é obrigatório, e outros mecanismos de autenticação, como login com o GitHub ou simplesmente um "arquivo de senha", podem ser usados.

Quando o Azure Active Directory estiver sendo usado, o Azure Red Hat OpenShift e as APIs do Kubernetes encaminharão solicitações de autenticação. Os usuários apresentarão suas credenciais e serão autorizados com base em suas funções.

A camada de autenticação identifica o usuário associado às solicitações da API do Azure Red Hat OpenShift. Em seguida, a camada de autorização usará informações sobre o usuário solicitante para determinar se a solicitação deve ser autorizada.

Você encontra as instruções de configuração para o Azure Active Directory na seção *Autenticação*: *Azure Active Directory*. Este processo é funcionalmente muito semelhante caso outro provedor de autenticação esteja sendo usado em vez de o Azure Active Directory.

É o mecanismo da política do Azure Red Hat OpenShift que cuida da autorização, que define ações como "criar pod" ou "listar serviços" e as agrupa em funções em um documento da política. As funções são vinculadas a usuários ou grupos pelo identificador do usuário ou grupo. Quando uma conta de usuário ou serviço tenta realizar uma ação, o mecanismo de política busca uma ou mais das funções atribuídas ao usuário (como administrador do cliente ou administrador do projeto atual) antes de permitir que continue.

As relações entre funções de cluster, funções locais, vinculação de funções de cluster, vinculações de funções locais, usuários, grupos e contas de serviço são exibidas na *Figura 3.3*:

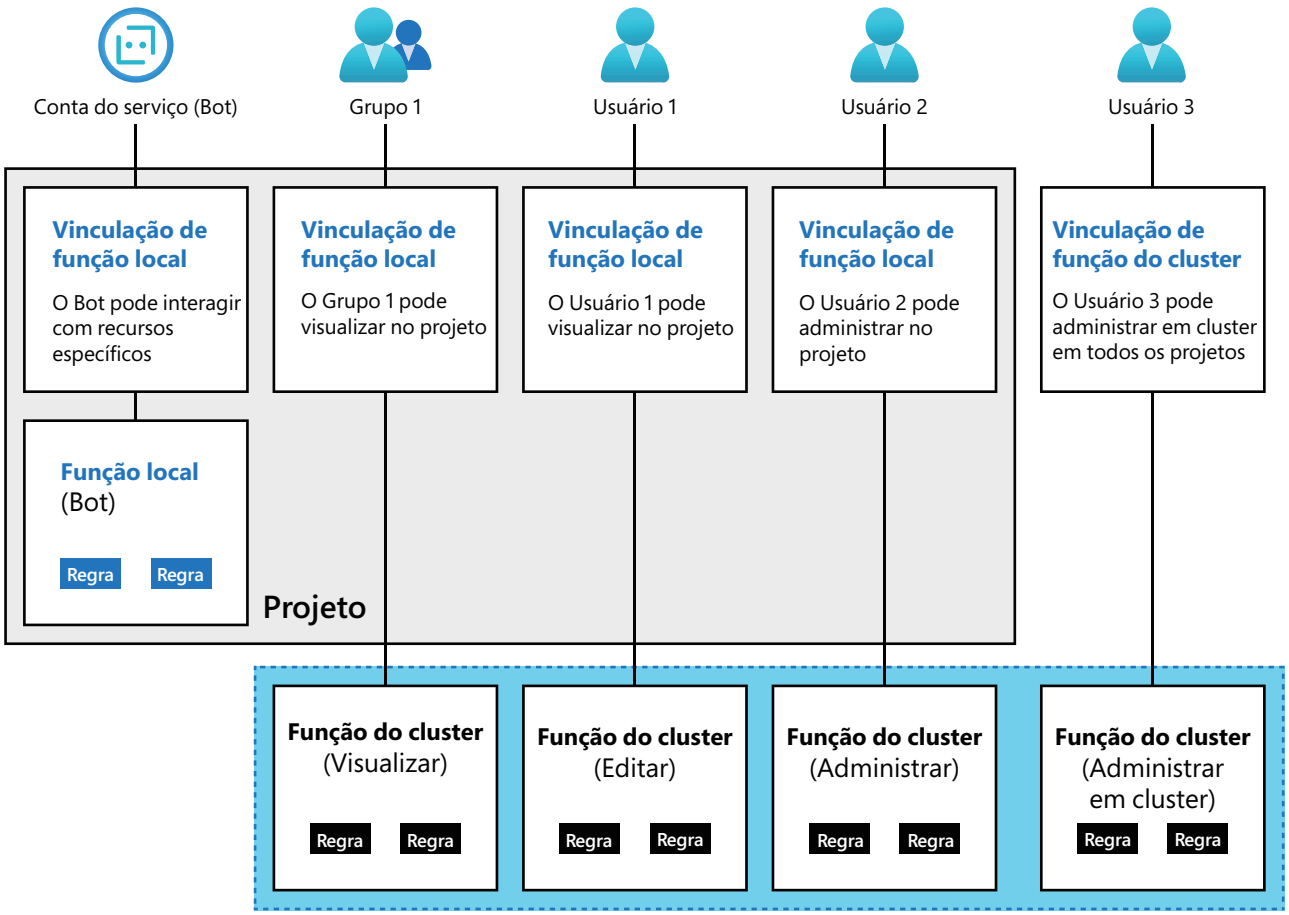


Figura 3.3: Relações entre funções

A documentação do Red Hat OpenShift tem uma lista completa de provedores de autenticação.

Suporte

O Azure Red Hat OpenShift é único na forma de gerenciamento do suporte. As equipes de suporte da Microsoft e da Red Hat trabalham juntas, ao lado da equipe global de Engenharia de confiabilidade de sites (SRE), para facilitar a operação do serviço.

O suporte de solicitações de clientes no portal do Azure e as solicitações passam por uma triagem, e são os engenheiros da Microsoft e da Red Hat que lidam com elas, seja no nível da plataforma do Azure ou do OpenShift.

Veja um resumo do processo de suporte integrado:

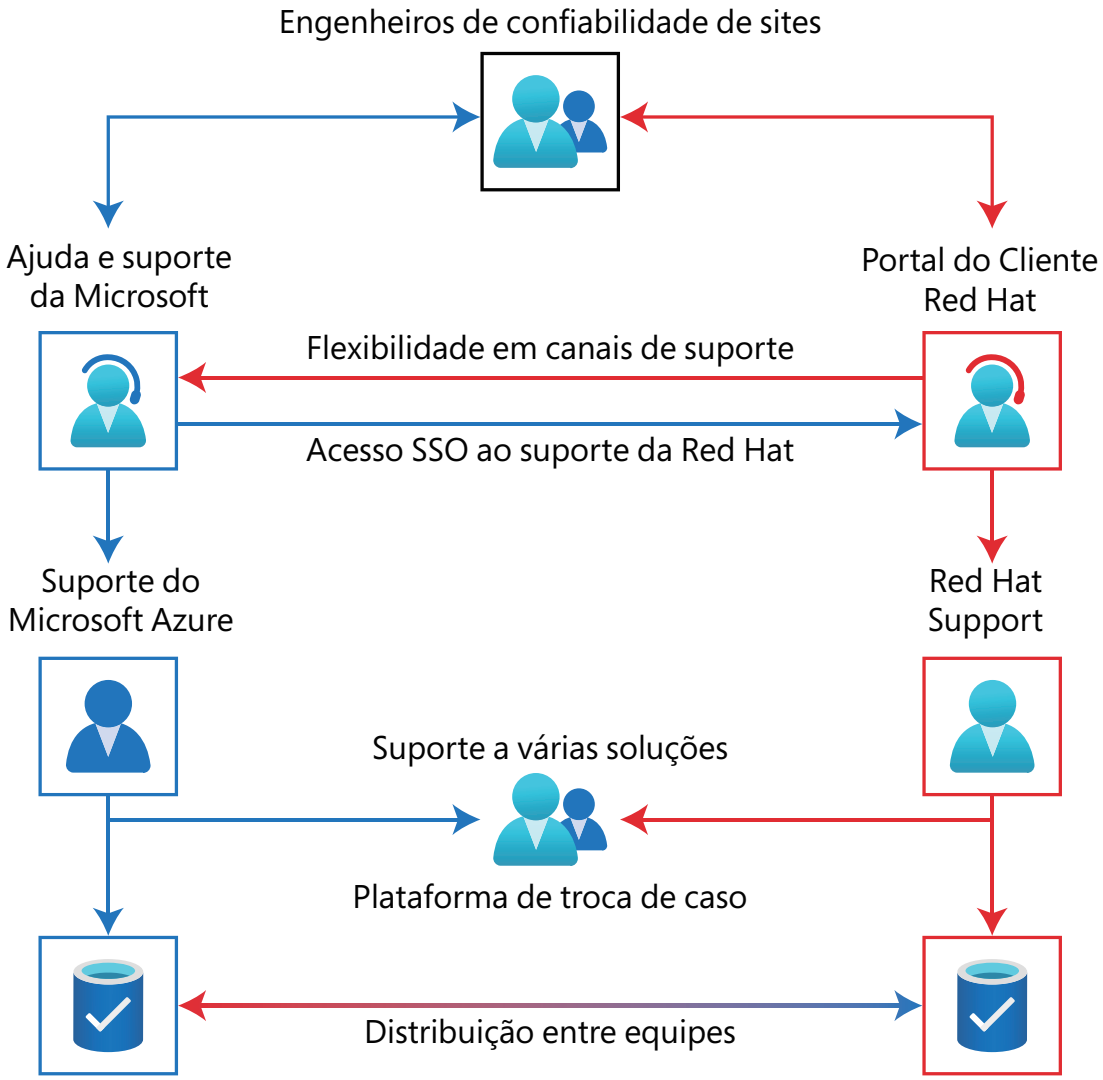


Figura 3.4: O processo de suporte integrado

A Figura 3.4 mostra que os clientes podem abrir um ticket de suporte no portal de suporte da Microsoft ou da Red Hat. Observe que, para acessar o portal de suporte da Red Hat, o cluster deve ser registrado no OpenShift Cluster Manager.

Os engenheiros de suporte da Microsoft podem colaborar sem problemas com a Red Hat em qualquer etapa, mediante autorização do cliente pela plataforma de troca do caso. Também é o caso para os engenheiros de suporte da Red Hat ao solicitar colaboração com a Microsoft. Os engenheiros de suporte das duas empresas têm acesso à equipe de SRE, que pode tomar medidas corretivas em clusters, se necessário.

Preços e subscrições

Um benefício importante do Azure Red Hat OpenShift em relação a uma instalação do tipo faça-você-mesmo é que a infraestrutura de computação, rede, armazenamento e do Azure, bem como as subscrições do OpenShift, são cobradas na sua subscrição do Azure, e não separadamente. Para ter uma ideia dos custos da solução, basta adicionar o Azure Red Hat OpenShift ao criador de cotações na Calculadora de preço do Azure.

Este é um guia para entender a página de preços:

1. Digite *openshift* na caixa de pesquisa e adicione a uma nova estimativa.

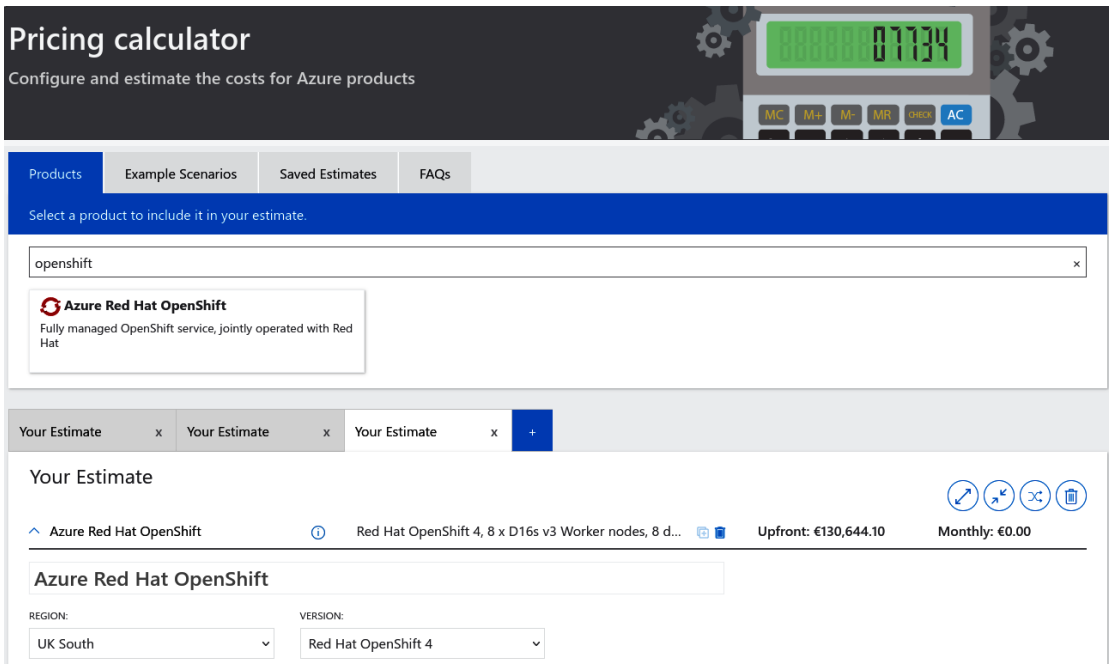


Figura 3.5: Painel da calculadora de preços

2. No fim da página, você verá uma caixa com menu suspenso para alterar a unidade de preços para sua moeda local. Neste exemplo, é usada libra esterlina (£).
3. Configure sua região do Azure para implantar o Azure Red Hat OpenShift. O preço variará um pouco entre as regiões para contabilizar diferentes custos de computação nas regiões do Azure.

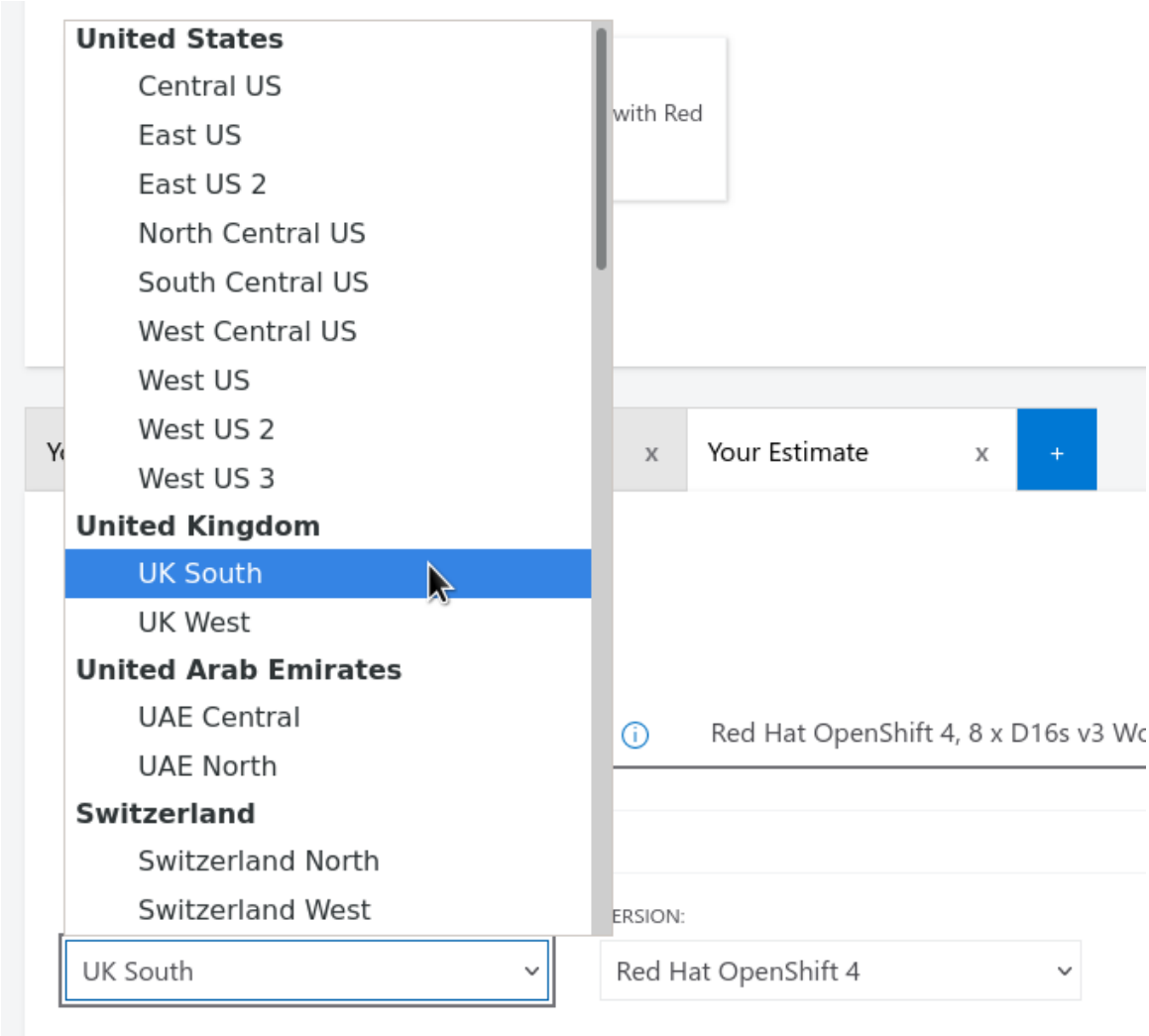


Figura 3.6: Selecionando a região do Azure

4. **Suas aplicações serão executadas nos nós de trabalho**. Em **Opções de economia**, você verá duas seções. A seção **Licença** se refere ao custo das subscrições do OpenShift, enquanto a seção **Máquina virtual** se refere ao custo dos serviços de computação necessários para a execução do cluster. O número máximo de nós de trabalho compatíveis em um cluster é 100.

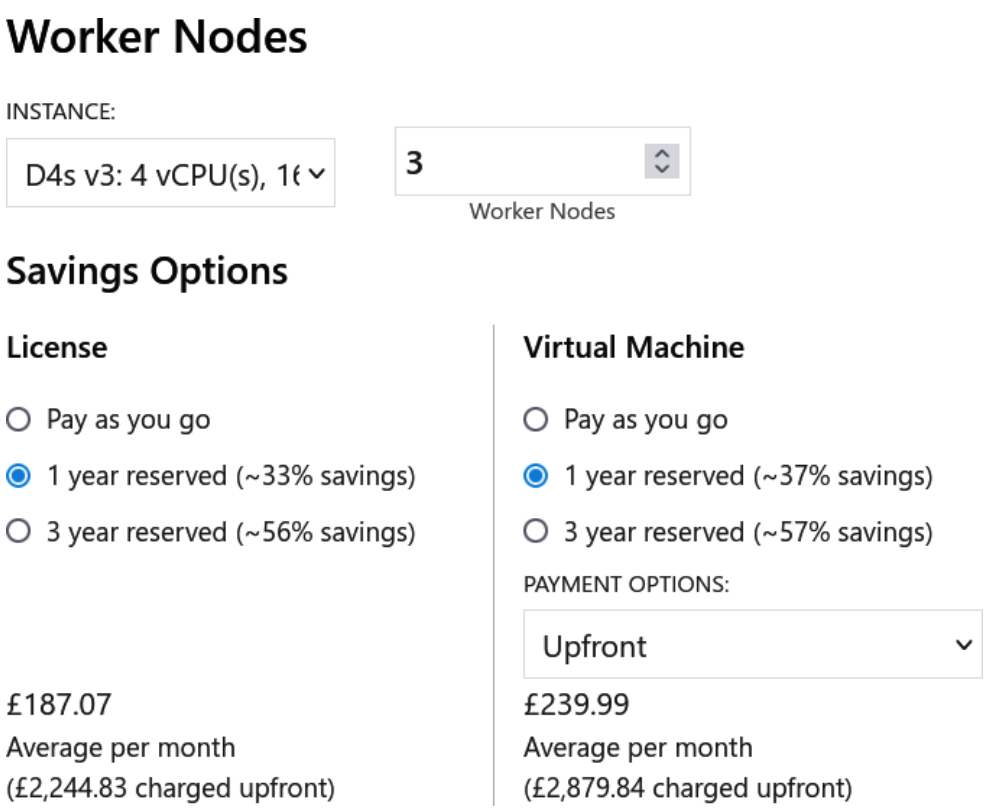


Figura 3.7: Detalhes dos nós de trabalho

5. **Discos de sistema operacional gerenciados** se refere ao tamanho dos discos usados pelo Red Hat CoreOS para as partições do sistema operacional. Isso não se refere ao armazenamento usado pelos volumes persistentes das aplicações, que são provisionados separadamente.

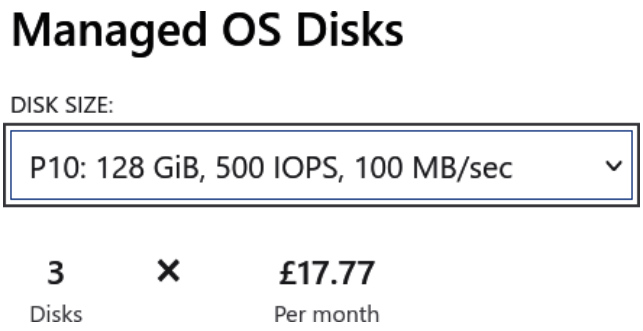


Figura 3.8: Discos de sistema operacional gerenciados

6. Há uma seção semelhante para **Nós mestres**, que normalmente são chamados de nós de controle na linguagem moderna. Os nós de controle contam com um disco muito maior associado a eles do que os nós de trabalho. Isso é porque eles exigem maior IOPS e capacidade, o que vem com tamanhos maiores do disco no Azure. Para a estabilidade do cluster, o número exato dos nós de plano de trabalho deve ser sempre três.

A calculadora foi criada para mostrar preços estimados. Os preços reais vão variar dependendo do uso.

Instâncias de máquina virtual reservadas do Azure

Uma **instância reservada** é um compromisso para consumir aquele recurso por um período, normalmente de 1 a 3 anos. Há opções de instâncias reservadas para as máquinas virtuais do Azure Red Hat OpenShift.

Quando sabem que um cluster será executado por mais tempo, as organizações normalmente escolhem instâncias reservadas para receber um desconto significativo na IaaS. Por exemplo, ambientes de produção são geralmente executados com instâncias reservadas. Vale observar que as instâncias reservadas não mudam o nível do serviço ou a arquitetura do cluster.

Mais informações sobre instâncias reservadas do Azure

Resumo

Este capítulo descreve as particularidades do serviço em nuvem gerenciado do Azure Red Hat OpenShift. Ele conta com uma visão geral da arquitetura, além de discutir brevemente a integração com outros serviços do Azure (que são abordados em mais detalhes no *capítulo 9: Integração com outros serviços*), junto com considerações sobre gerenciamento, autenticação, suporte e preços.

O próximo capítulo terá como foco dúvidas e decisões que uma organização terá que resolver na fase de Pré-provisionamento da implantação do Azure Red Hat OpenShift.

Capítulo 4

Pré-provisionamento: questões sobre arquitetura empresarial

Muitas organizações verão o Azure Red Hat OpenShift no portal do Azure e, lendo toda a documentação, é possível implantar com sucesso um cluster com o Red Hat OpenShift sem dificuldades. No entanto, se for possível dispor de um pouco mais de tempo para planejar implantações e tirar dúvidas com antecedência, muito mais tempo será economizado na futura exclusão e reprovisionamento de clusters.

Este capítulo se baseia na experiência prática e real do trabalho com muitos clientes. Ele aborda muitas das questões típicas sobre pré-provisionamento que devem ser tratadas com antecedência. O capítulo aborda:

- Quantos clusters são necessários, incluindo de teste, produção, entre outros
- Visibilidade de rede pública e privada
- Conectividade híbrida, como conexão a soluções on-premise

Começaremos falando sobre como definir a quantidade de clusters que você vai precisar.

Quantos clusters são necessários?

Existem muitos padrões de implantação para o OpenShift, mas uma dúvida comum é: "De quantos clusters minha organização precisa?". É claro que essa é uma decisão que depende da organização, mas nos parágrafos a seguir damos uma orientação para ajudar você a chegar a esse número.

Estágios do ciclo de vida: desenvolvimento, testes, produção

A maioria das organizações de qualquer escala implantará seus sistemas de TI empresarial com algum tipo de estágio de ciclo de vida. Essa abordagem às vezes também é chamada de padrão de fases. As três fases mais comuns são desenvolvimento, testes e produção. Ter várias fases permite que mudanças e implantações de aplicações sejam testadas em um ambiente seguro antes que elas cheguem a um ambiente de produção. O padrão de fases mais comum e recomendado é ter ao menos três clusters separados do Azure Red Hat OpenShift:

- **Desenvolvimento:** quando qualquer desenvolvedor e operador tem permissão para testar qualquer coisa. Este pode ser um cluster grande de "área restrita", mas é mais comum e, normalmente, mais seguro, ter clusters pequenos com tempo de vida curto em que testes são criados e destruídos frequentemente.
- **Testes:** quando as mudanças futuras em um cluster, como mudanças de patches ou de configuração, são testadas e validadas antes de serem lançadas para produção. Esta etapa também pode ser chamada de "pré-produção" em algumas organizações, mas a pré-produção também pode ser todo um outro ambiente.
- **Produção:** quando as aplicações realmente são executadas.

Além do mencionado anteriormente, algumas organizações terão ambientes adicionais, como ambientes de testes de integração, mas apenas você saberá quantos ambientes de estágios são necessários para sua organização. Se estiver com dúvidas, veja outras aplicações empresariais semelhantes para padrões de implantação comuns.

Em situações em que o Azure Red Hat OpenShift estiver sendo usado para aplicações que não sejam críticas, pode ser aceitável ter apenas dois clusters na sua organização (mesclando desenvolvimento e testes), ou até mesmo um único cluster que inclua desenvolvimento, testes e produção. A vantagem dessa alternativa é manter os custos baixos e reduzir o número geral de clusters que devem ser gerenciados. Ao operar de apenas um único cluster, os administradores podem escolher usar namespaces separados para desenvolvimento, testes e produção. No entanto, executar apenas um único cluster tem desvantagens:

- Mudanças que afetam todo o cluster, como patches de software, podem gerar problemas na produção que poderiam ser prevenidos e impedidos se elas fossem executadas em um ambiente de teste.
- Se uma aplicação em um ambiente de testes ou de desenvolvimento tiver um comportamento inesperado, como a indução de vários containers ou o uso de todo o espaço disponível do disco, o ambiente de produção terá problemas.

Este livro não pode passar um número exato de clusters que funcionará perfeitamente para o seu caso. Como mencionado, é recomendável analisar aplicações empresariais semelhantes na sua organização para ver quantos estágios de ciclo de vida estão sendo usados.

Continuidade de negócios, recuperação de desastres e failover

Junto aos ambientes de estágios padrão, muitas organizações também vão querer criar ao menos um ambiente de produção de failover. Um ambiente de produção de failover é usado no caso de uma falha catastrófica que derruba todo o cluster ou a região do Azure. Isso costuma ser chamado de **Recuperação de desastres (DR)**. Normalmente, ela seria implantada em uma região do Azure diferente do cluster de produção.

O serviço de nuvem gerenciado vem com um **Contrato de nível de serviço (SLA)** de 99,95%, o que, para muitas aplicações críticas para os negócios, é aceitável. No entanto, para algumas aplicações, pode ser necessário um SLA superior. É importante entender que não é possível ir além com este SLA com um único cluster. Veja se sua aplicação precisa de um nível de serviço maior que 99,95% ou se este valor é o suficiente.

Se não for, você pode conseguir níveis mais altos de disponibilidade de serviço (como 99,999%) calculando um SLA composto a partir da execução de vários clusters em paralelo. Os clusters podem estar todos na mesma região (por exemplo, `westeurope`) ou em várias regiões (por exemplo, `westeurope` e `northeurope`) para níveis ainda maiores de disponibilidade. A documentação do Azure a seguir descreve como calcular SLAs compostos ao executar vários clusters.

Documentação do Azure sobre SLAs compostos

Implantações de multicluster e multirregião do Azure Red Hat OpenShift estão fora do escopo deste livro. Isso ocorre porque, ao criar essas arquiteturas mais complexas, existem diversos desafios associados à obtenção de tráfego para o cluster. Além disso, escolher como e quais dados compartilhar entre os clusters exige muita atenção.

Regiões e zonas de disponibilidade

O Azure Red Hat OpenShift foi projetado para usar três zonas de disponibilidade em cada região em que é implantado. No Azure, uma zona de disponibilidade é um datacenter autônomo em uma região, com sua própria energia, resfriamento e conectividade de rede. Ao analisar uma implantação do Azure Red Hat OpenShift, você verá um único nó de controle (máquina virtual) e nó de aplicações em cada zona de disponibilidade.

A *Figura 4.1* mostra como os nós de controle e os nós de aplicações (de trabalho) são distribuídos entre as zonas de disponibilidade em uma única região do Azure:

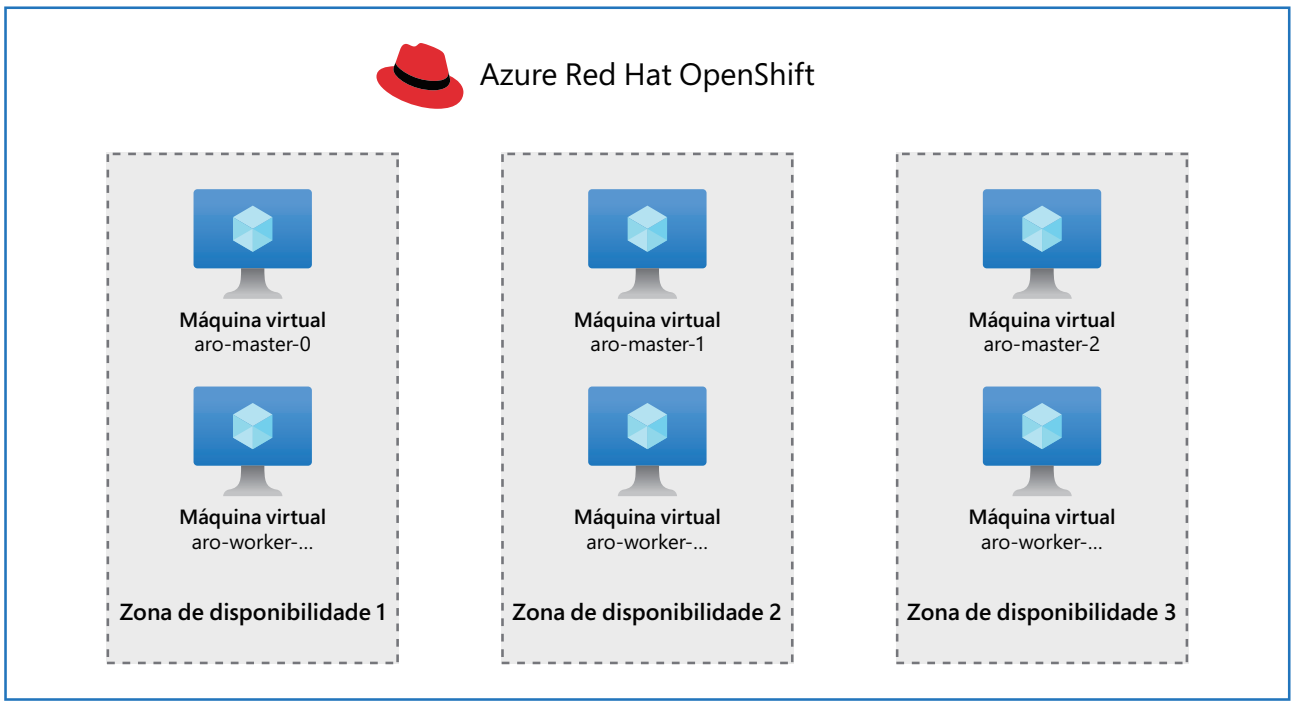


Figura 4.1: Distribuição dos nós de controle e de aplicações entre as três zonas de disponibilidade em uma única região do Azure

O Azure Red Hat OpenShift foi projetado para ser um serviço totalmente gerenciado e altamente disponível. Portanto, não é possível implantar menos que três nós de controle e três nós de trabalho.

Conceitos de rede

Esta excelente página de documentação, que foi mencionada anteriormente na introdução ao Azure Red Hat OpenShift e trata dos conceitos de rede, pode ser encontrada no site de documentação da Microsoft:

- Conceitos de rede do Azure Red Hat OpenShift

Essa página conta com uma descrição detalhada de todos os componentes de rede no Azure Red Hat OpenShift: balanceadores de carga, endereços IP públicos e privados, grupos de segurança de rede e muito mais.

Alguns pontos importantes que devem ser entendidos são:

- O Azure Red Hat OpenShift é implantado em uma rede virtual nova ou existente. Apenas uma única rede virtual é compatível, mas várias redes não trariam nenhum benefício adicional, já que o OpenShift coloca sua própria **rede definida por software (SDN)**, chamada OVS, no topo.
- O tamanho mínimo das redes do nó mestre e de aplicações é `/27`.
- O CIDR do pod padrão é `10.128.0.0/14`.
- O CIDR do serviço padrão é `172.30.0.0/16`.
- Em cada nó, é alocada uma sub-rede de `/23` (512 endereços IP) para seus pods. Esse valor não pode ser alterado.
- IPs de saída não são compatíveis no momento.
- É possível controlar o roteamento de tráfego de saída, especificamente para enviá-lo por meio do Azure Firewall. No momento em que o livro foi escrito, essa funcionalidade estava em pré-visualização pública e documentada aqui: [Controlar tráfego de saída].

Visibilidade de rede pública ou privada

Você deve ouvir com frequência que o Azure Red Hat OpenShift pode ter uma implantação pública ou privada. Embora isso seja verdade, é importante entender a diferença entre tornar o plano de controle público/privado e tornar as aplicações no seu cluster públicas/privadas.

Para a visibilidade do servidor da API, é uma decisão que você precisa fazer no tempo de provisionamento. Não é possível ajustar a visibilidade pública/privada depois que o cluster já foi provisionado.

Para criar um cluster do Azure Red Hat OpenShift mais adiante neste capítulo, você executará o comando `az aro create`, que aceita argumentos sobre a visibilidade do cluster. O exemplo a seguir mostra como ajustar a visibilidade:

Ambas privadas

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

Servidor da API privado e entrada de trabalho pública

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

Veja a visibilidade da `apiserver` e o mapa de entrada da aplicação no diagrama da arquitetura do Azure na *Figura 4.2*. Este diagrama mostra como o Azure Red Hat OpenShift usa balanceadores de carga internos e públicos do Azure, onde a API e o roteador são implantados dependendo das opções de visibilidade selecionadas.

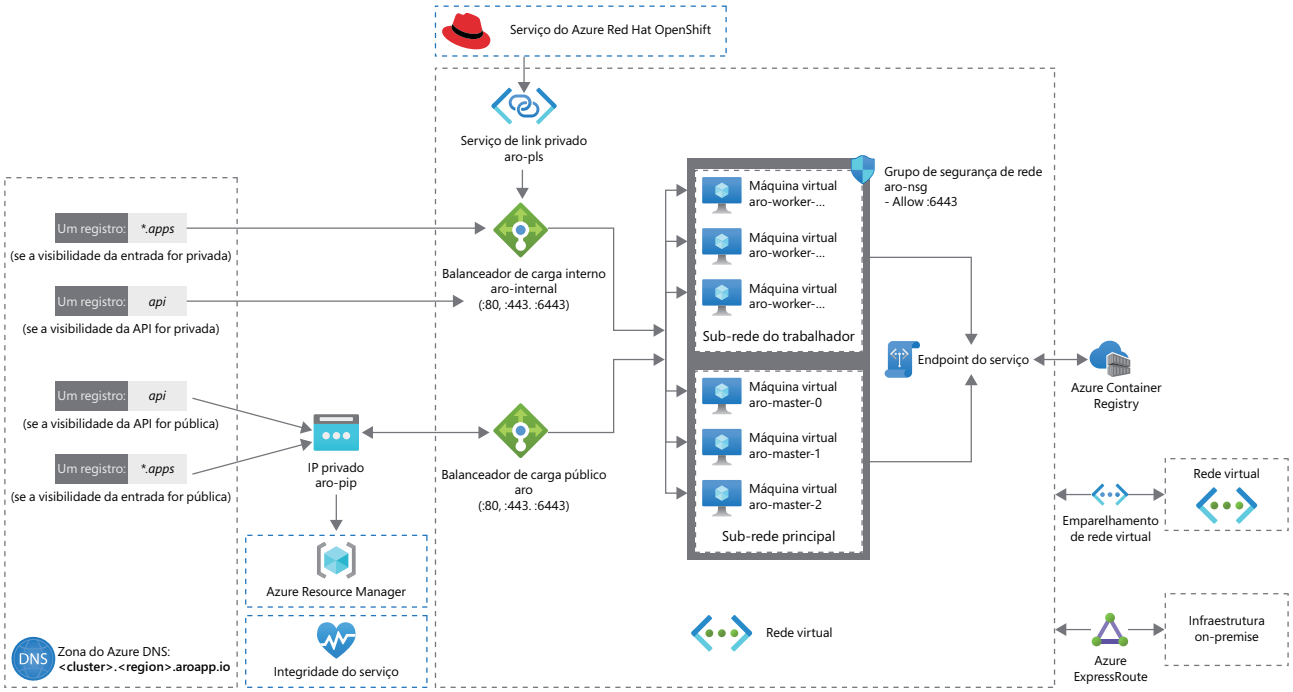


Figura 4.2: Azure Red Hat OpenShift usando balanceadores de carga internos e públicos do Azure

A seguir é possível encontrar descrições mais detalhadas sobre a visibilidade do servidor da API e a visibilidade de entrada.

Visibilidade do servidor da API (plano de controle)

`--apiserver-visibility` pode ser **pública** ou **privada**:

- **Privada** significa que o plano de controle do Red Hat OpenShift (historicamente chamado de "nós mestres"), onde a API Kubernetes é executada, não é acessível a partir de uma internet pública. Esse plano de controle é direcionado a desenvolvedores e operadores para que eles controlem o cluster e pode ser usado para implantar, excluir ou escalar aplicações, além do próprio cluster. As empresas que têm conexões de rota expressa para o Azure ou usam uma **rede privada virtual (VPN)** para acessar recursos computacionais devem escolher a opção privada na maioria dos casos.
- **Pública** significa que o plano de controle do cluster é acessível a partir de uma internet pública. Isso expõe o cluster ao risco de ataques de qualquer pessoa na internet, mesmo que o acesso seja autenticado. No entanto, defini-la como `pública` é útil para ambientes de teste ou de laboratório, onde não é possível controlar a rede de origem de seus usuários. Para ambientes em que dados reais são armazenados, ou qualquer tipo de ambiente de produção, é altamente recomendável definir a visibilidade do servidor da API como `privada`.

A lista a seguir apresenta alguns exemplos de onde a conectividade do servidor da API é necessária. Isso deve ser levado em consideração ao selecionar a opção pública ou privada:

- Desenvolvedores usando scripts e ferramentas de seu IDE (por exemplo, `kubectl rollout`)
- Operadores inspecionando o status de seu cluster (por exemplo, `kubectl get nodes`)
- Servidores de CI/CD que precisam inspecionar ou ajustar o estado das implantações (por exemplo, Azure DevOps)
- Ferramentas de segurança do cluster que se conectam ao cluster para examinar o estado
- Monitoramento de ferramentas que dependem da API do Kubernetes

Na maioria do casos, as redes podem ser configuradas para virem de conexões **privadas**, mas a lista anterior pode incluir outros exemplos de dentro da sua organização, e você pode encontrar uma exigência inesperada por visibilidade **pública** do servidor da API.

Visibilidade de entrada (aplicações)

`--ingress-visibility` pode ser pública ou privada:

- **Privada** significa que serviços expostos (que se relacionam às aplicações em execução no cluster) não permitem conexões diretamente da internet pública. Ainda é possível configurar o roteamento de redes para que os usuários passem primeiro pelo Azure Firewall, ou um firewall de aplicações web, em execução opcionalmente em uma rede do Azure separada antes de se conectar às suas aplicações. **Privada** também funciona bem se as aplicações no cluster forem usadas somente internamente na sua organização, por exemplo, para processamento de pagamentos, análise de dados ou outras aplicações web internas.
- **Pública** significa que as aplicações no seu cluster são alcançáveis pela internet pública. No entanto, ainda seria necessário configurar um recurso de entrada ou de rota para conseguir acessar essas aplicações. Este seria o caso se você estivesse hospedando um site de compras de comércio eletrônico público ou outra aplicação pública no Red Hat OpenShift.

Entrada às vezes é chamada de "roteador" do OpenShift.

Recomendações para produção

É altamente recomendável:

- Acessar o cluster por uma conexão privada, e não por uma internet pública. O Azure ExpressRoute é a melhor opção quando o cluster precisar de conectividade permanente a partir de uma rede on-premise ou escritório corporativo. Caso contrário, uma VPN no Azure também pode oferecer conexão privada. Veja mais detalhes sobre isso na seção **Conectividade híbrida**.
- Defina a visibilidade do servidor da API (plano de controle) como `privada` e, opcionalmente, restrinja ainda mais o acesso com firewalls.
- Defina a visibilidade de entrada (aplicações) como `privada` para aplicações em execução na sua organização. Se houver um caso de uso de aplicações no Azure Red Hat OpenShift que precisem ser hospedadas na internet pública, configure um cluster separado do Azure Red Hat OpenShift: ao menos um cluster para aplicações internas e outro para aplicações externas com a visibilidade de entrada definida como pública. No entanto, é possível misturar entradas `públicas` e `privadas` no mesmo cluster.

Naturalmente, a maioria das organizações falará com as equipes de segurança e rede do Azure para determinar quaisquer controles adicionais que possam ser necessários. Por exemplo, algumas organizações exigem que as aplicações web sejam protegidas por um firewall de aplicações web. Esse também é um padrão de implantação comum ao implantar aplicações no Azure Red Hat OpenShift.

Conectividade híbrida

A maioria das organizações que implanta o Azure Red Hat OpenShift executará aplicações que precisam se conectar de volta aos serviços de suporte em execução em ambientes on-premise. Realizar a conexão do Azure de volta a ambientes on-premise costuma ser chamado de conectividade híbrida, ou uma arquitetura de nuvem híbrida. Há algumas maneiras de fornecer conectividade de volta para ambientes on-premise. As duas opções mais conhecidas são:

- **VPN**, adequada para conectividade de baixa complexidade ao Azure. Normalmente, VPNs são conectadas via gateway de VPN do Azure. Para mais informações, veja O que é o gateway de VPN?
- Os **circuitos Azure ExpressRoute**, que são adequados para uma conexão dedicada, robusta e permanente ao Azure. Para mais informações, veja O que é Azure ExpressRoute?

Independentemente de qual método de conexão for usado, as duas soluções podem permitir que aplicações on-premise se conectem a aplicações em execução no Azure Red Hat OpenShift e vice-versa.

Ao se conectar de um ambiente on-premise ao Azure Red Hat OpenShift, é comum fazer essa conexão por meio de uma rede virtual de hub. A *Figura 4.3* mostra um diagrama que explica como funciona a conectividade com o Azure ExpressRoute:

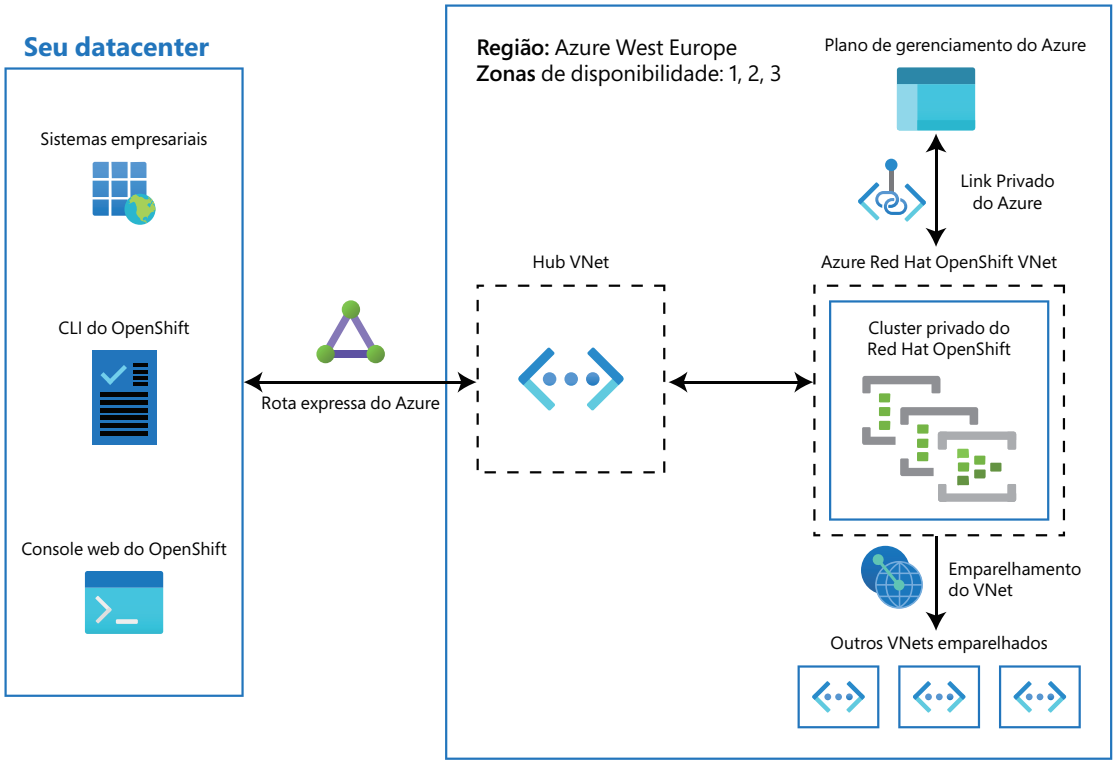


Figura 4.3: Conectividade com o Azure ExpressRoute

O diagrama anterior mostra a arquitetura geral do Azure ExpressRoute se conectando ao Azure Red Hat OpenShift. Embora esse diagrama mostre o Azure ExpressRoute, a conectividade por meio de uma VPN é conceitualmente muito semelhante.

Considerações para aplicações em execução em uma arquitetura híbrida

Ao executar aplicações no Azure Red Hat OpenShift que se conectam de volta a ambientes on-premise, há alguns pontos que os proprietários da aplicação devem considerar:

- **Latência de conexão:** enquanto o Azure ExpressRoute oferece uma conexão de latência relativamente baixa de volta para ambientes on-premise, as redes VPN que navegam pela internet pública apresentam latência consideravelmente mais alta. Para algumas aplicações, como servidores web, em que a conexão está apenas transmitindo tráfego HTTP, é pouco provável que isso cause problemas. No entanto, se uma aplicação ao lado do servidor em execução naquele servidor web precisasse se conectar a um banco de dados em execução on-premise, poderia haver um impacto considerável no desempenho.
- **Custo de tráfego de entrada/saída:** alguns tipos de conexão cobrarão pelo tráfego de entrada e saída, seja pelo Azure ou no provedor de conexão. É uma precaução para medir os custos em um ambiente de testes primeiro e depois projetar quais podem ser esses custos futuramente em uma carga de pico, apenas para garantir que não haverá surpresas.
- **Cenário de falha:** enquanto as conexões do Azure ExpressRoute são projetadas para serem permanentes e robustas, as conexões de VPN são frequentemente sujeitas à interrupção ou conexão/desconexão. Como elas também são executadas na internet pública, a latência da conexão de VPN e a qualidade do serviço podem variar com bastante frequência ao longo do dia. É importante testar o desempenho da aplicação em más condições de link híbrido e quando ela estiver com o desempenho ideal. Além disso, se a conexão for interrompida por várias horas, a aplicação em execução no Azure Red Hat OpenShift poderá continuar sendo executada de maneira degradada ou sua disponibilidade depende inteiramente da conexão híbrida?

Sua organização pode também exigir diversos outros pontos a serem verificados em seus checklists internos, mas os pontos anteriores devem ser suficientes para começar a pensar como uma arquitetura híbrida funcionaria para você.

Resumo

Neste capítulo, discutimos muitas das dúvidas normais de pré-provisionamento que você deveria perguntar e pesquisar antes de implantar o Azure Red Hat OpenShift. O próximo capítulo mostra alguns indicadores úteis para recursos na fase de implantação real do cluster.

Capítulo 5

Provisionamento de um cluster do Azure Red Hat OpenShift

Vamos recapitular o que vimos até agora. Neste livro, você viu o seguinte:

- No *capítulo 2*, *Introdução ao Red Hat OpenShift*, apresentamos brevemente o Red Hat OpenShift e explicamos as vantagens de escolher o OpenShift em vez do Kubernetes simples.
- No *capítulo 3*, *Azure Red Hat OpenShift*, descrevemos as particularidades do serviço em nuvem gerenciado do Azure Red Hat OpenShift e abordamos alguns conceitos importantes, incluindo sua arquitetura, gerenciamento, autenticação, suporte e considerações sobre preços.
- No *capítulo 4*, *Pré-provisionamento: questões sobre arquitetura empresarial*, discutimos os pontos comuns que as organizações devem considerar antes de implantar o Azure Red Hat OpenShift.

Agora que está tudo pronto para provisionar e implantar um cluster, você verá as instruções oficiais de implantação no site de documentação (provisionamento é parte da implantação; provisionar uma implantação significa garantir a presença da infraestrutura de TI necessária para uma implantação).

Tutorial: criar um cluster do Azure Red Hat OpenShift 4

Este capítulo não aborda os comandos individuais que você precisa digitar para criar um cluster, porque esses comandos podem mudar com o tempo, e a documentação já fala sobre eles em detalhes.

No entanto, ele oferecerá orientações sobre o processo geral e o que você precisa considerar ao implantar um cluster.

Implantações manuais: expectativas de tempo

Algumas organizações precisam entender quanto tempo leva para provisionar um cluster. Vamos analisar isso com uma descrição detalhada.

Um processo geral dos prerrequisitos de implantação é o seguinte:

- A linha de comando az implantada na estação de trabalho de um administrador de sistema
- Os provedores de recursos registrados para uso com sua subscrição do Azure
- Um pull secret da Red Hat (opcional)
- Um domínio para seu cluster (opcional)
- Uma rede virtual, com duas sub-redes vazias, uma para os nós de plano de controle e outra para os nós de aplicações

Em relação ao tempo necessário para configurar esses prerrequisitos, um administrador experiente do Azure e do Red Hat OpenShift poderia concluir essas tarefas em 30 minutos na primeira vez e em apenas 10 minutos se executar essas etapas de maneira repetitiva como parte de um processo manual.

Assim que os prerrequisitos são cumpridos, o processo de provisionamento automatizado geralmente leva entre 25 e 40 minutos dependendo da atividade na região do Azure.

Automação da implantação

Com a compreensão de que o processo de provisionamento do Azure Red Hat OpenShift é automatizado, é comum que uma organização tente usar ferramentas para automatizar também as etapas de prerrequisito: criar redes, sub-redes, princípios de serviço etc.

Existem muitas ferramentas que podem fazer isso. Algumas ferramentas recomendadas são as seguintes:

- A ferramenta de linha de comando az: quando automatizada, essa ferramenta é normalmente instalada em um container ou algo semelhante como parte de um processo de CI/CD. Ferramentas típicas incluem o Jenkins, Azure DevOps ou possivelmente Ansible. Observe que a ferramenta de linha de comando az precisa ser implantada somente uma vez, mas os clusters adicionais podem precisar definir o ID da subscrição do Azure para refletir diferentes partes da organização.
- Os provedores de recursos registrados para uso com sua subscrição do Azure: como anteriormente, isso é parte da configuração da ferramenta de linha de comando az.
- Um pull secret da Red Hat (opcional): a Red Hat documentou a API REST compatível para obter um pull secret, sobre o qual se pode encontrar informações neste **artigo**.
- Um domínio para seu cluster (opcional): isso depende de como você cria registros de DNS. No entanto, o Azure DNS, o Terraform, o Ansible ou outras ferramentas de automação populares do Azure poderiam fazer isso por você.
- Uma rede virtual com duas sub-redes vazias, uma para os nós (mestres) de plano de controle e outra para os nós (de trabalho) de aplicações. Isso normalmente seria automatizado pelas ferramentas de automação populares do Azure: Terraform, Ansible ou outras semelhantes, que poderiam criar essa rede e sub-redes para você.

Com as etapas de prerrequisito automatizadas, o tempo desse processo pode reduzir de 30 ou 10 minutos, como seria o caso de um processo manual, para apenas um ou dois minutos. Não é possível acelerar a implantação do cluster (que sempre leva entre 25 e 40 minutos), mas este pode ser um processo de implantação de ponta a ponta muito rápido em comparação com processos on-premise.

A automação da implantação tem muitas vantagens para além de acelerar o processo. Clientes usando automação da implantação podem criar um processo robusto e repetível que é facilmente logado e auditado. Também é extremamente comum os clientes criarem itens de catálogo de autosserviço em seus próprios portais. Isso permite que as equipes provisionem e desprovisionem com facilidade os clusters do Azure Red Hat OpenShift sem a necessidade de interação com a equipe da plataforma de nuvem.

Acesso ao cluster

Esta seção oferece uma simples referência sobre como acessar seu cluster do Azure Red Hat OpenShift após o provisionamento.

Pela IU da web

Em um shell de Bash para quem instalou a CLI, ou na sessão do Azure Cloud Shell (Bash) no seu portal do Azure, recupere a URL de login do seu cluster executando o seguinte comando:

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

Você deve ver algo parecido com `openshift.xxxxxxxxxxxxxxxxxxx.eastus.aroapp.io`. A URL de login para seu cluster será `https://` seguido pelo valor `consoleProfile.url`. Por exemplo, `https: //openshift.xxxxxxxxxxxxxxxxxxx.eastus.aroapp.io`.

Abra esta URL no seu navegador. Será solicitado que você faça login com o usuário `kubeadmin`. Use o nome de usuário e senha fornecidos durante o processo de instalação.

Depois de fazer login, você verá o console web do Azure Red Hat OpenShift.

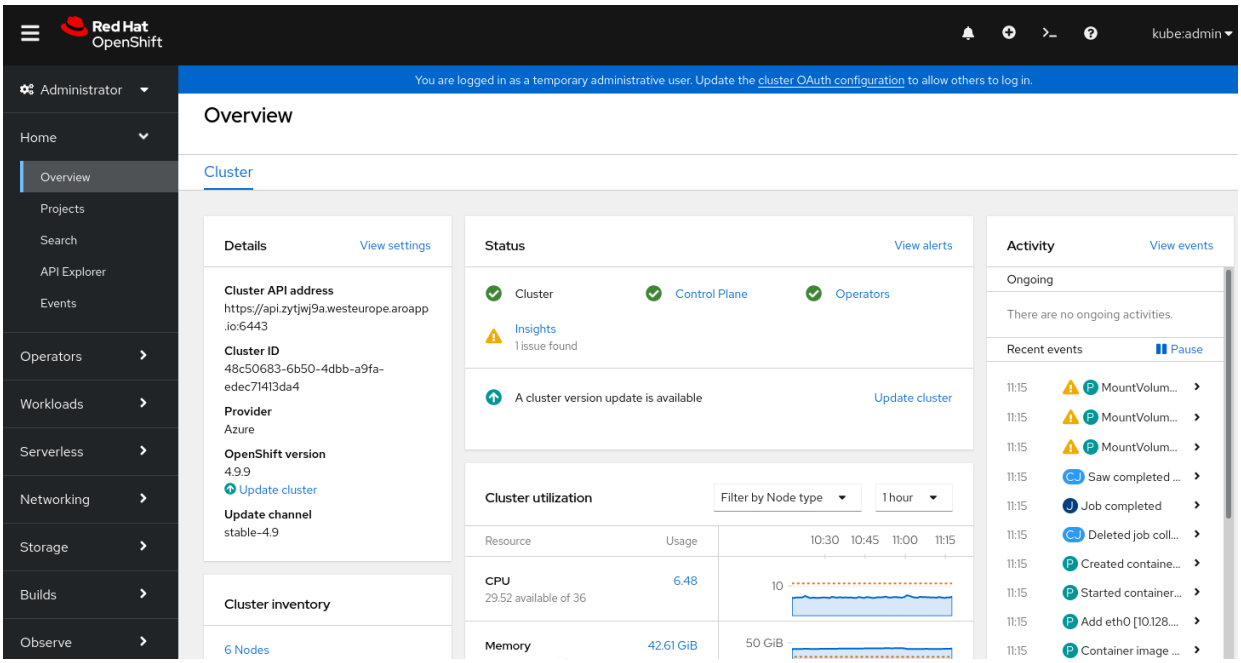


Figura 5.1: Console Web do Azure Red Hat OpenShift

Explorar o console web leva tempo. Observe que uma versão recente do OpenShift ainda deve estar em execução, e todos os componentes devem estar em um status íntegro ou a ser definido como íntegro após a instalação.

Via CLI do OpenShift (oc)

Você precisará fazer o download da CLI mais recente do OpenShift (oc). Para isso, basta fazer login em https://console.redhat.com/openshift/downloads.

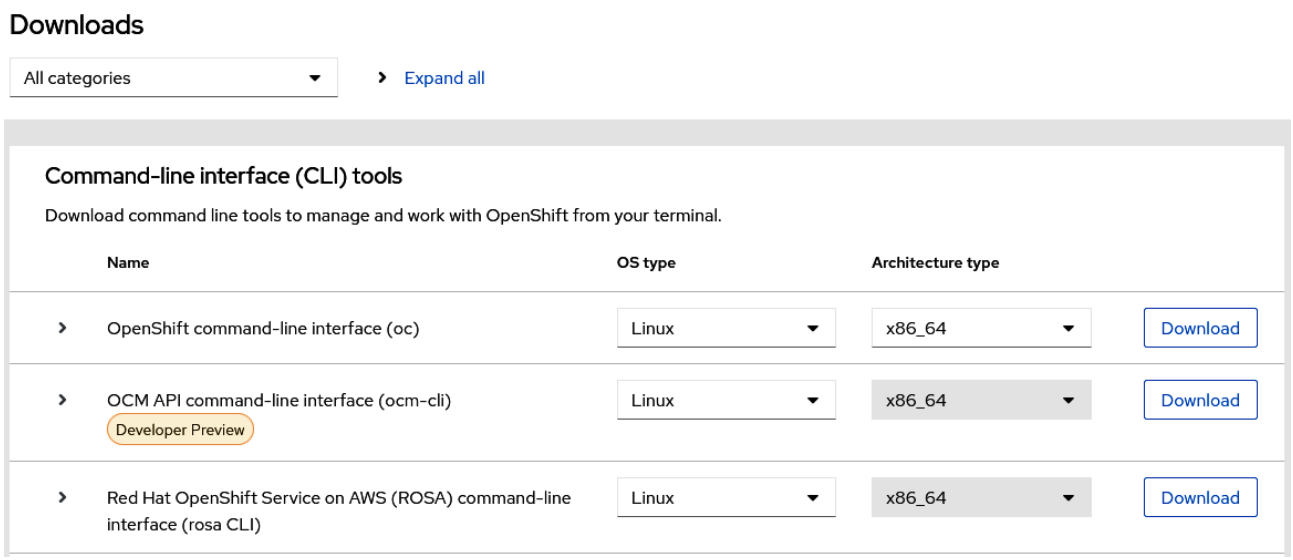


Figura 5.2: Download da interface da linha de comando do OpenShift

Extraia o arquivo (`.tar.gz` or `.zip`) para o seu sistema e, em seguida, coloque o comando oc em algum lugar em seu caminho. No Linux, é comum colocar o comando oc em `/usr/local/sbin/`.

Execução da CLI do OpenShift e login no cluster

Para a autenticação do seu cluster a partir da linha de comando, será necessário recuperar o token e comando de login no console web. Faça login no console web com seu navegador, clique no nome de usuário no canto direito superior e, em seguida, clique em **Copy login command** (Copiar comando de login).

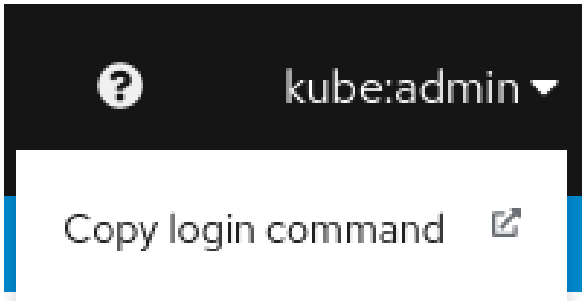


Figura 5.3: Copiar comando de login

Isso abrirá uma nova página mais ou menos assim:

Figura 5.4: Login com o token da API

Em seguida, você pode copiar esse comando e colar em um terminal para fazer login no seu cluster do Azure Red Hat OpenShift.

Por exemplo, se estiver usando a sessão do Azure Cloud Shell (Bash) no seu portal do Azure, cole o comando de login. O comando `oc status` parece com isso:

```
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'.
```

Agora você pode dedicar um tempo para explorar mais a linha de comando `oc` e se familiarizar com o novo ambiente do Azure Red Hat OpenShift. Se já for um usuário experiente do OpenShift 4, esse serviço em nuvem do Azure Red Hat OpenShift deve ser muito familiar e ter o mesmo comportamento que outros ambientes do OpenShift 4 que você pode ter usado no passado.

Resumo

Este foi um capítulo curto, já que as instruções de provisionamento oficiais ainda se mantêm, e ele deve ser considerado o guia definitivo para executar o Azure Red Hat OpenShift na sua subscrição do Azure. As instruções de provisionamento são bem mais simples do que as instruções para provisionar um ambiente de OpenShift autogerenciado. Isso ocorre porque foi dedicada uma quantidade considerável de engenharia para o serviço do Azure Red Hat OpenShift, oferecendo integração física ao Azure, além de implantar uma arquitetura prescritiva padrão que passa por bons testes e é compatível. Em geral, esta é uma vantagem para as organizações, já que menos tempo é dedicado ao provisionamento do Azure Red Hat OpenShift, permitindo que mais tempo seja dedicado à implantação de aplicações.

Capítulo 6

Pós-provisionamento (segunda etapa)

Após a implantação do Azure Red Hat OpenShift, algumas atividades de pós-provisionamento costumam ser necessárias antes que um cluster esteja pronto para produção. Este capítulo aborda muitas das atividades de pós-provisionamento comuns, por exemplo:

- Autenticação, Azure Active Directory, incluindo como sincronizar grupos de usuários a um operador de comunidade
- Operadores, incluindo o OperatorHub
- Introdução a geração de logs, incluindo encaminhamento de logs
- Introdução ao monitoramento, incluindo o monitoramento de containers do OpenShift
- Upgrades e patches, incluindo o ciclo de vida da versão compatível
- Escala de clusters, incluindo escalar o cluster manual e automaticamente
- Escala de aplicações, uma breve nota sobre escala de aplicações
- Configuração de um pull secret, registro ao OpenShift Cluster Manager
- Intervalos de limites, incluindo onde um intervalo de limite pode ser aplicado
- Armazenamento persistente, incluindo as classes de armazenamento disponíveis e compatíveis
- Segurança e conformidade, uma nota sobre controles de segurança

Essas seções, cada uma detalhando diferentes tarefas de pós-provisionamento, ajudarão a entender o que é necessário para ter um cluster recém-implantado e deixá-lo pronto para aplicações de produção.

Autenticação, Azure Active Directory

O Azure Red Hat OpenShift tem suporte para todos os provedores de autenticação listados na documentação do OpenShift. Confira a lista completa de provedores de autenticação. No entanto, a maioria dos clientes tende a usar o Azure Active Directory, que já está disponível no Azure, para fornecer autenticação e single sign-on ao Azure Red Hat OpenShift.

A configuração para integração do Azure Active Directory é intencionalmente uma operação de "segunda etapa", já que podem existir casos em que as organizações queiram usar um provedor de configuração alternativo. O processo de configuração e instalação para o Azure Active Directory leva aproximadamente 15 minutos na primeira configuração e, depois de conhecer o processo melhor, você verá que ele pode ser feito em apenas cinco minutos. Muitas organizações escolhem automatizar partes do provisionamento usando ferramentas como modelos de ARM ou Ansible Playbooks.

- Configure o Azure Active Directory para Azure Red Hat OpenShift (Portal gráfico)
- Configure o Azure Active Directory para Azure Red Hat OpenShift (Interface da linha de comando)

Uso de grupos de usuários do Active Directory

A configuração da autenticação do Azure Active Directory somente permitirá que os usuários façam login com credenciais já existentes. Ela não traz os grupos de usuários existentes para um usuário por meio do Red Hat OpenShift. É bastante comum que uma organização queira importar grupos de usuários do Active Directory para que possam ser usados na configuração de permissões de usuário no OpenShift. Um operador separado, compatível com a comunidade, está disponível para isso, o Group Sync Operator.

- Group Sync Operator no GitHub

Observe que o operador de sincronização de grupos recebe suporte da comunidade, o que significa que não recebe suporte oficial da Red Hat ou Microsoft até o momento em que este livro foi escrito. Este livro recomenda seguir as instruções detalhadas de configuração para o operador vistas no arquivo README do projeto.

Operadores

Os operadores no OpenShift 4 são um dos componentes de criação essenciais da plataforma e agregam muito valor aos clientes do Red Hat OpenShift. Os operadores são criados a partir de um código e executam um serviço em um container no cluster. Alguns operadores são responsáveis pela manutenção de coisas como redes de cluster, configuração da máquina e upgrades de cluster. Eles são normalmente chamados de operadores de clusters e estão listados na seção **Administração** do console do OpenShift.

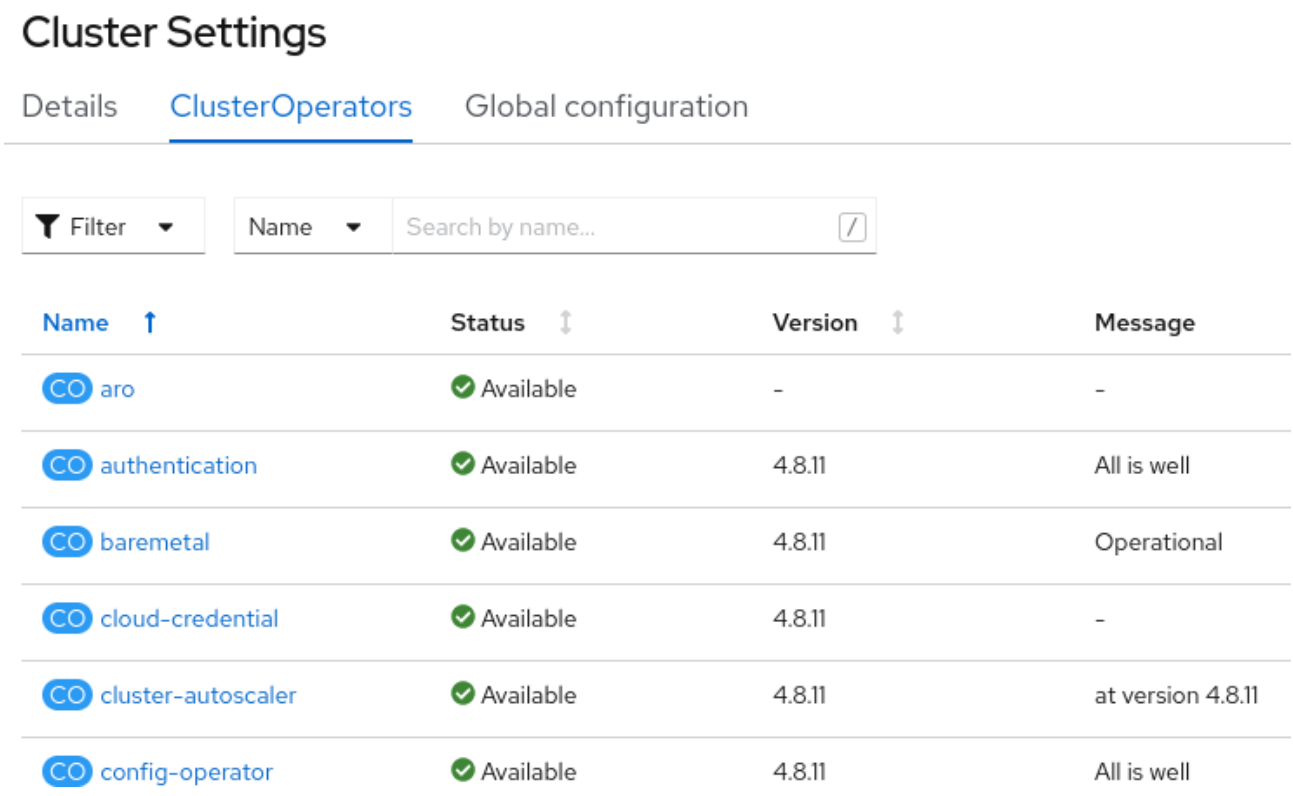


Figura 6.1: Configurações de cluster

A captura de tela anterior mostra que há um operador apenas para o Azure Red Hat OpenShift, que mantém partes do serviço e tenta garantir que o cluster se mantenha em um estado de configuração compatível.

Os operadores são capazes de manter muitas coisas, como a integridade de um serviço, atualizações, patches, escala e muitas outras funções.

OperatorHub

Além dos operadores de cluster padrão executados no plano de fundo de cada cluster, os administradores têm acesso a muitos mais operadores por meio do OperatorHub. Com o OperatorHub, é fácil encontrar operadores populares empresariais e de comunidade que um administrador pode disponibilizar na instalação do Red Hat OpenShift. O OperatorHub está na barra lateral de cada cluster do OpenShift.

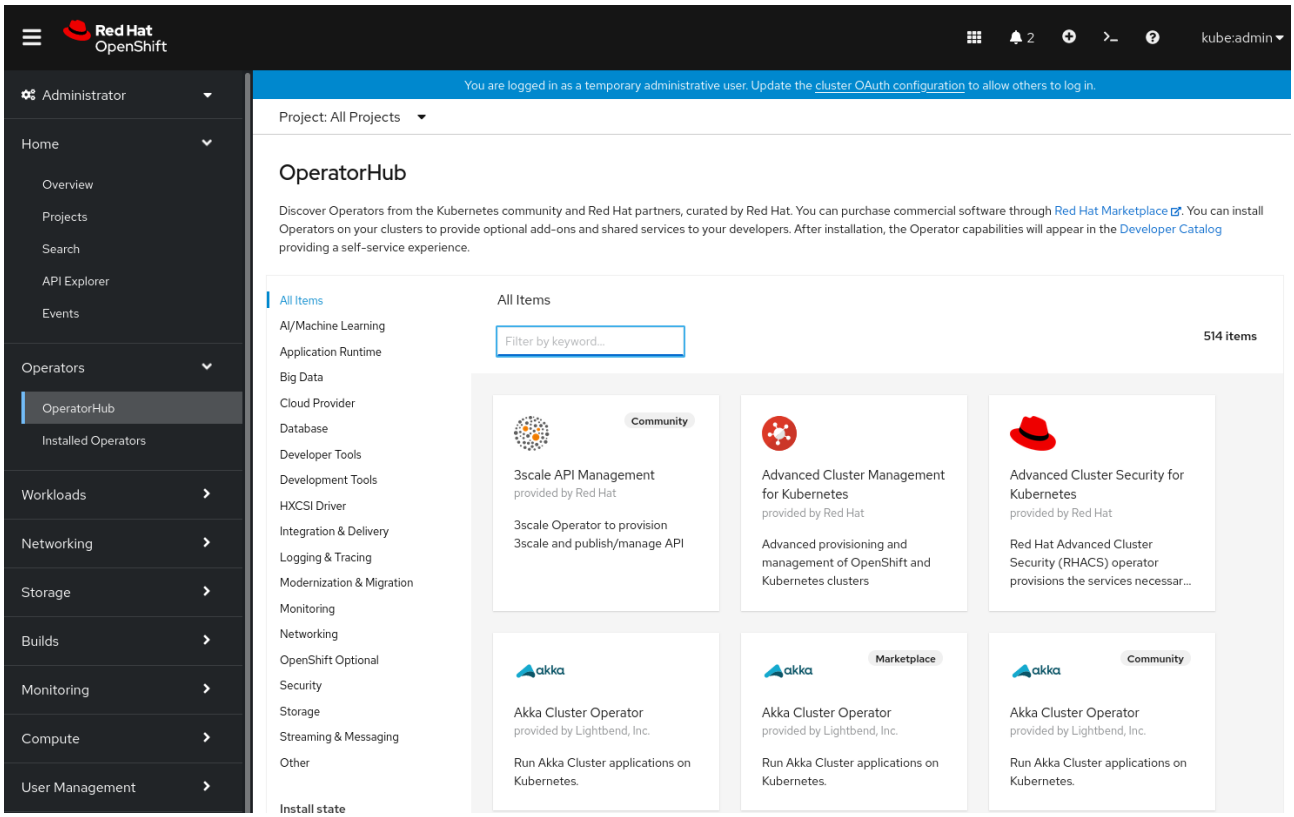


Figura 6.2: OperatorHub

Usando operadores, os administradores podem implantar e manter softwares populares de maneira rápida e fácil, sem precisar se preocupar com código e configuração do Kubernetes. Muitas soluções Red Hat agora são empacotadas como operadores, como o Advanced Cluster Management, o Red Hat OpenShiftService Mesh e o Red Hat OpenShift Pipelines. No entanto, também há uma enorme biblioteca de operadores para softwares que não são da Red Hat, como o banco de dados MariaDB, Couchbase ou unidades de armazenamento em blocos da IBM.

Acesse os links a seguir para ver mais sobre os operadores:

- Operatorhub.io
- Operadores no OpenShift

Introdução a geração de logs

O Azure Red Hat OpenShift usa a mesma arquitetura de geração de logs e monitoramento que o Red Hat OpenShift. As duas ofertas usam os mesmos operadores para geração de logs.

Os logs são divididos em três categorias:

- **Aplicações:** logs de container gerados por aplicações do usuário em execução no cluster, exceto aplicações de container de infraestrutura.
- **Infraestrutura:** logs gerados por componentes de infraestrutura em execução no cluster e nós do OpenShift Container Platform, como logs journal. Os componentes de infraestrura são pods que são executados no `openshift*`, `kube*` ou em projetos padrão.
- **Auditoria:** logs gerados por auditd, o sistema de auditoria de nós, armazenados no arquivo `/var/log/audit/audit.log` e os logs de auditoria do servidor da API do Kubernetes e do servidor da API do OpenShift.

Acesse Introdução ao Red Hat OpenShift Logging na documentação da solução para ver uma descrição detalhada sobre o assunto.

Uso do encaminhamento de logs de clusters para o Azure Monitor

Uma integração comum está enviando logs do Azure Red Hat OpenShift para o serviço de insights de containers do Azure Monitor. Isso às vezes é chamado de Azure Log Analytics. E, permite uma retenção de logs de baixo custo e persistente no Azure.

A arquitetura de geração de logs do OpenShift executa o Fluent Bit em cada nó, e a primeira ação que um operador poderia realizar seria ajustar a configuração desse serviço do Fluent Bit para enviar logs diretamente. No entanto, ajustar a configuração de logs no Azure Red Hat OpenShift está fora da política de suporte. O OpenShift contém um mecanismo integrado chamado `ClusterLogForwarder` para encaminhar logs enquanto ainda são compatíveis.

Usando o `ClusterLogForwarder`, é possível encaminhar logs para o Elasticsearch, Flutentd e syslog. No entanto, o Azure Monitor não dá suporte direto para nenhum desses protocolos. Uma alternativa seria ter uma instância intermediária adicional do Fluent Bit que receba logs do OpenShift e os encaminhe para o Azure Monitor. Essa abordagem está descrita na documentação da Microsoft e na documentação da comunidade.

Introdução ao monitoramento

Como um serviço gerenciado, não deve ser necessário implementar monitoramento personalizado complexo ao Azure Red Hat OpenShift, já que ele é fornecido como parte do serviço pago.

No entanto, é muito comum querer monitorar os containers do OpenShift usando insights do Azure Container. No momento, essa funcionalidade está em pré-visualização pública e é descrita nesta documentação.

Upgrades e patches

Ao provisionar um cluster do Azure Red Hat OpenShift, não será selecionado um canal de atualização. Isso significa que seu cluster não começará a receber atualizações por padrão.

Veja seu canal de atualização em **Administração** → **Configurações de cluster** na barra lateral de navegação. Veja as configurações do cluster logo após o provisionamento dele do Azure Red Hat OpenShift:

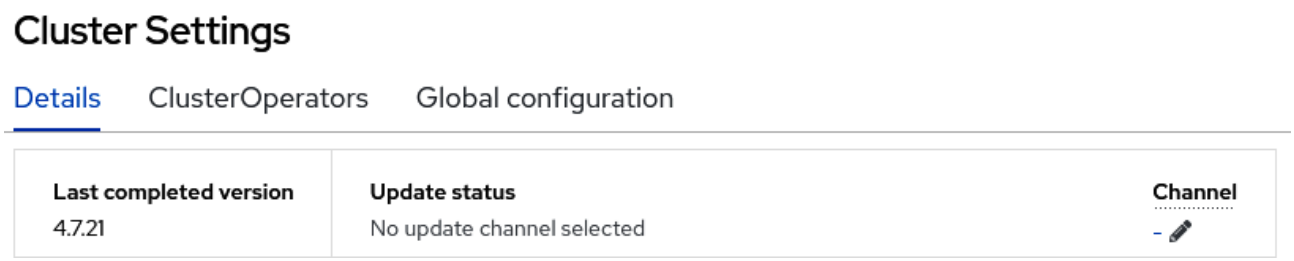


Figura 6.3: Configurações de cluster após o provisionamento dele

Para selecionar um canal de atualização, selecione o link "-" e veja as opções disponíveis:

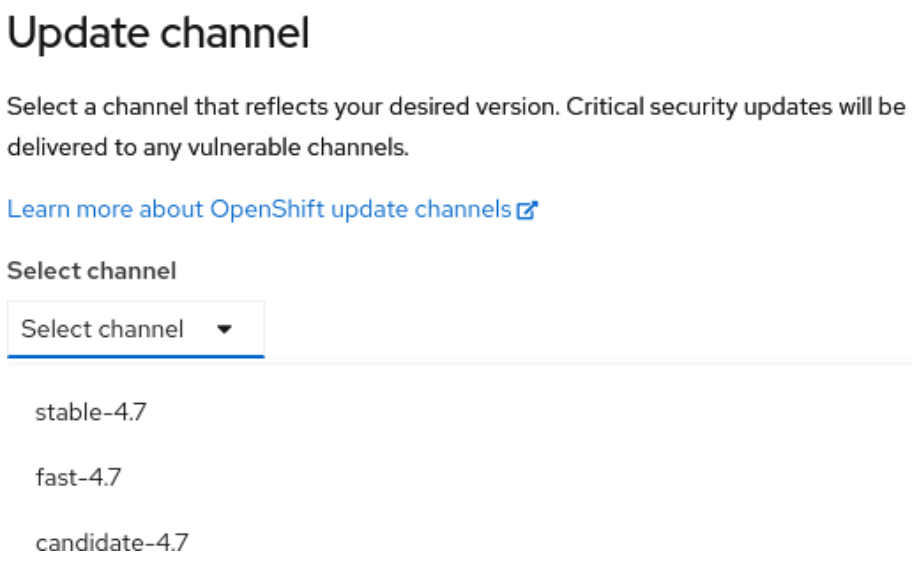


Figura 6.4: Selecionar um canal de atualização

Entenda as diferenças entre `estável`, `rápido` e `candidato` na página de documentação [Realizar upgrade de canais e lançamentos](#).

É altamente recomendável que todos os clusters em produção sejam executados em um canal **estável**.

Ciclo de vida da versão com suporte

É importante saber quais versões do Azure Red Hat OpenShift recebem suporte para planejar sua implantação de produção. A página formal de ciclo de vida conta com informações que ajudam as organizações a entender quais versões recebem suporte ou não.

[Página do ciclo de vida de suporte do Azure Red Hat OpenShift](#)

Veja as informações da página simplificadas aqui:

O Azure Red Hat OpenShift tem suporte para duas versões secundárias de **disponibilidade geral (GA)** do Red Hat OpenShift Container Platform:

- A versão secundária de GA mais recente que é lançada no Azure Red Hat OpenShift (a qual agora chamaremos de N)
- Uma versão secundária anterior (N-1)

O Red Hat OpenShift Container Platform usa o controle de versão semântico. O controle de versão semântico usa diferentes níveis de números de versões para especificar diferentes níveis de controle de versão. A tabela a seguir ilustra as diferentes partes de um número de versão semântico. Neste caso, usamos o exemplo o número de versão 4.9.3:

Versão principal (x)	Versão secundária (y)	Patch (z)
4	9	3

Cada número da versão mostra a compatibilidade geral com a versão anterior:

- **Versão principal:** nenhum lançamento de versão principal planejado no momento. As versões principais mudam quando uma API incompatível mudar ou quando a compatibilidade com versões anteriores estiver corrompida.
- **Versão secundária:** lançada aproximadamente a cada três meses. Upgrades de versões secundárias podem incluir adições de funcionalidades, melhorias, substituições, remoções, correções de bug e melhorias em segurança.
- **Patches:** normalmente lançados semanalmente ou conforme necessário. Atualizações de versões de patch podem incluir correções de bug e melhorias em segurança.

Leia a página do ciclo de vida de suporte do Azure Red Hat OpenShift para saber mais informações sobre as versões com suporte.

Escala do cluster

O Red Hat OpenShift Container Platform e, por extensão, o Azure Red Hat OpenShift são criados pensando em uma arquitetura escalável. Ao planejar a escala no contexto do OpenShift, os administradores e proprietários de aplicações geralmente precisam considerar a escala do cluster e da aplicação como dois tópicos distintos.

Esta seção aborda a escala do cluster, e temos uma nota breve falando sobre a escala de aplicações em uma seção separada.

Máximo suportado

A escala de clusters se refere a adicionar mais nós de trabalho ao cluster que, por sua vez, oferece capacidade computacional adicional para aplicações em execução no cluster. Naturalmente, a questão surge quando é necessário escalar o plano de controle também. O Azure Red Hat OpenShift já implanta três nós de plano de controle que, a princípio, vêm com capacidade suficiente para escalar o número máximo de nós de trabalho suportados em um cluster do Azure Red Hat OpenShift. No momento, este número é 60.

No momento em que o livro foi escrito, havia três tamanhos de instâncias de nós de plano de controle com suporte: `Standard_D8s_v3`, `Standard_D16s_v3` e `Standard_D16s_v3`.

Existem mais tipos de instâncias do Azure com suporte para os nós de trabalho: com otimização de computação (série F), com otimização de memória (série E) e de propósito geral (série D).

Acesse a página da política de suporte e veja uma lista de tipos de instâncias do Azure com suporte no momento para o Azure Red Hat OpenShift.

Implantação mínima e escala para zero

Ocasionalmente, as organizações talvez queiram implantar clusters menores para fins de teste e desenvolvimento ou outros casos de uso em que a disponibilidade reduzida seria aceitável. No momento, o tamanho mínimo do cluster é de três nós de plano de controle e três nós de aplicações, e não é possível escalar com tamanhos menores.

Escala manual do cluster

Os operadores que olham para o portal do Azure podem ter a tentação de experimentar e adicionar mais nós de trabalho ao cluster do Azure Red Hat OpenShift diretamente criando uma máquina virtual do Azure. No entanto, não é possível, pois o grupo de recursos que contém o Azure Red Hat OpenShift está "bloqueado" pela perspectiva do administrador do Azure. Isso acontece independentemente das permissões. Até mesmo usuários com a permissão de **proprietário** não podem modificar o conteúdo de um grupo de recursos do Azure Red Hat OpenShift. Portanto, ocorrerão erros de permissão negada se você tentar criar uma máquina virtual manualmente no grupo de recursos do Azure Red Hat OpenShift.

O mecanismo pretendido para escala do Azure Red Hat OpenShift é a funcionalidade `MachineSets` do OpenShift, através da qual o OpenShift implantará um novo nó de aplicações quando solicitado. Os usuários com privilégios de administrador de cluster podem ver o **MachineSets** em **Computação**.

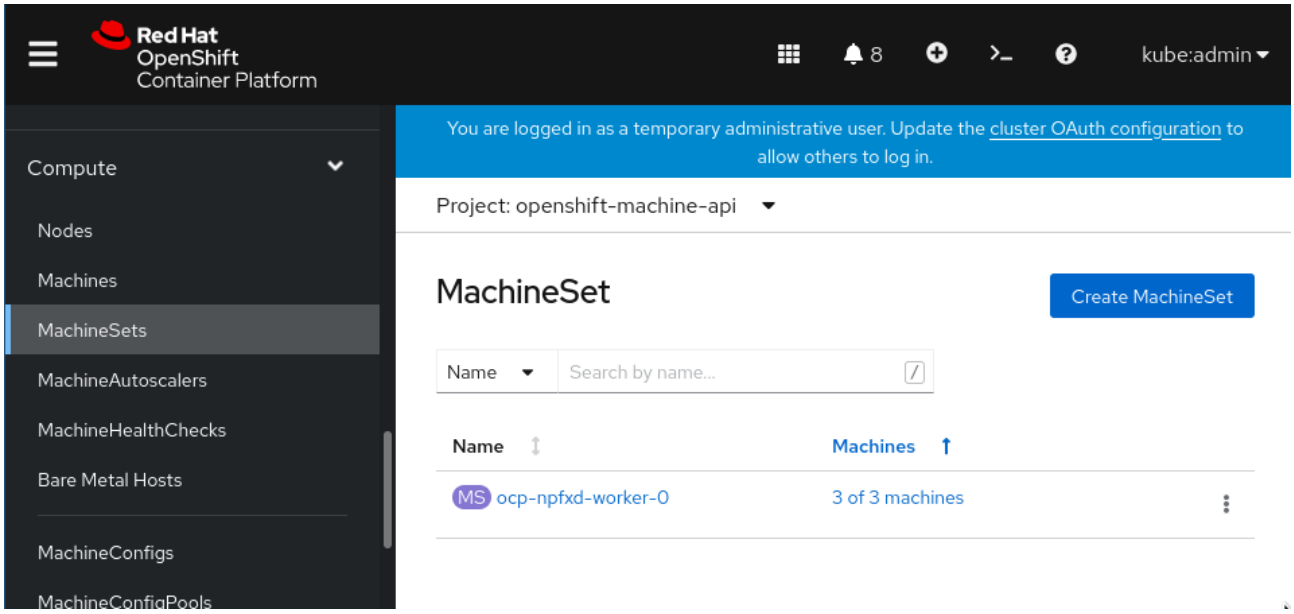


Figura 6.5: Acesso de administrador ao MachineSets

Ao clicar no menu "editar" de um nó de aplicações do `MachineSet`, você verá que é possível simplesmente provisionar uma nova máquina virtual de nó de aplicações selecionando **Editar contagem de máquina**.

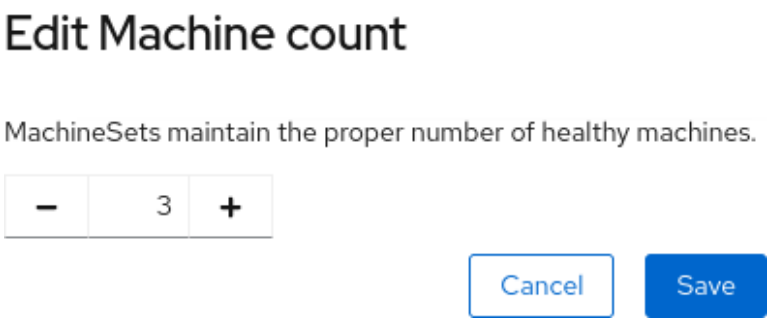


Figura 6.6: Edição da contagem de máquina

Depois que a contagem tiver sido atualizada, os administradores podem acessar o portal do Azure ou a visualização **Computação** → **Máquinas** para ver o provisionamento de uma nova máquina virtual de nó de aplicações.

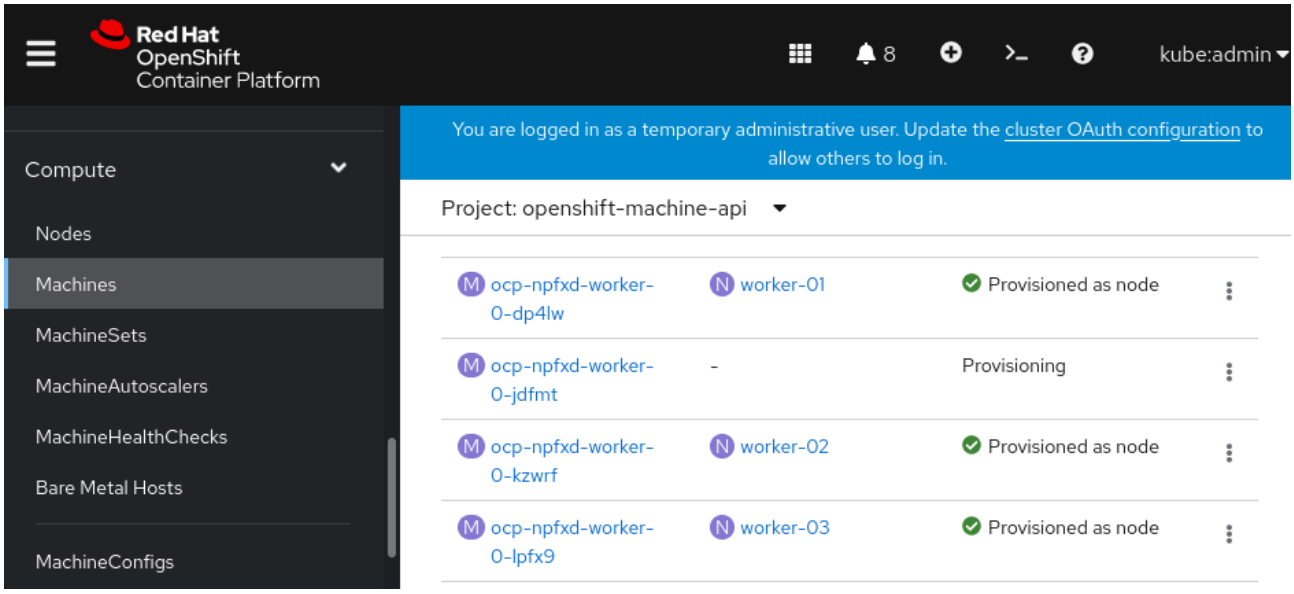


Figura 6.7: A visualização das máquinas

Normalmente leva até cinco minutos para provisionar uma máquina virtual adicional na maioria das regiões do Azure.

Os administradores não precisam fazer nenhuma atividade de pós-provisionamento para que esse novo recurso fique online. O OpenShift o disponibilizará automaticamente ao cluster e as aplicações o usarão quando necessário.

Autoescalonamento de cluster

A implantação padrão do Azure Red Hat OpenShift não está configurada com o recurso de autoescalonamento ativo, mas é fácil ativar essa funcionalidade. Basta os administradores criarem um recurso de `MachineAutoscaler`, que pode ser encontrado no menu da barra lateral **Computação** do Azure Red Hat OpenShift.

Os recursos de `MachineAutoscaler` operam em um `MachineSet` que, por sua vez, criará ou excluirá a capacidade de máquina virtual de nó de aplicações conforme necessário. O exemplo a seguir mostra que é possível manter um número mínimo ou máximo de máquinas em um `MachineSet`:

```
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
```

O OpenShift também tem o conceito de autoescaladores de cluster, o que possibilita escalar com base na manutenção de uma quantidade de RAM, CPU ou outros recursos semelhantes disponíveis. A escolha de um `MachineAutoscaler` ou um `ClusterAutoscaler` depende de como você deseja adicionar e remover capacidade no seu cluster.

Documentação do Red Hat OpenShift sobre MachineAutoscalers e ClusterAutoscalers

Escala de aplicações

A escala de aplicações de microsserviços é um assunto complicado que está fora do escopo deste livro. Entretanto, há duas indicações de páginas úteis para começar a explorar o tema:

- HorizontalPodAutoscaler: especifique o número mínimo e máximo de pods que deseja executar, bem como a utilização pretendida da CPU ou da memória pelos seus pods.
- Serverless e escala para zero com o Knative.

O cluster vai monitorar os containers em execução e sua capacidade restante. Caso o Knative ou `HorizontalPodAutoscaler` solicite mais recursos do que os disponíveis no cluster, será necessária uma escala de `ClusterAutoscaler` ou `MachineSet` para adicionar os recursos solicitados ao cluster.

Usando essa abordagem, as aplicações e o próprio cluster podem ser escalados em sincronia, tanto para vertical quanto horizontalmente, de acordo com a demanda.

Configuração de um pull secret (registro ao OpenShift Cluster Manager)

Uma implantação recente do Azure Red Hat OpenShift não tem um "pull secret" configurado para `cloud.redhat.com`. Isso significa que seus clusters não aparecerão no console de nuvem híbrida da Red Hat (http://console.redhat.com) por padrão. Chamamos isso de OpenShift Cluster Manager.

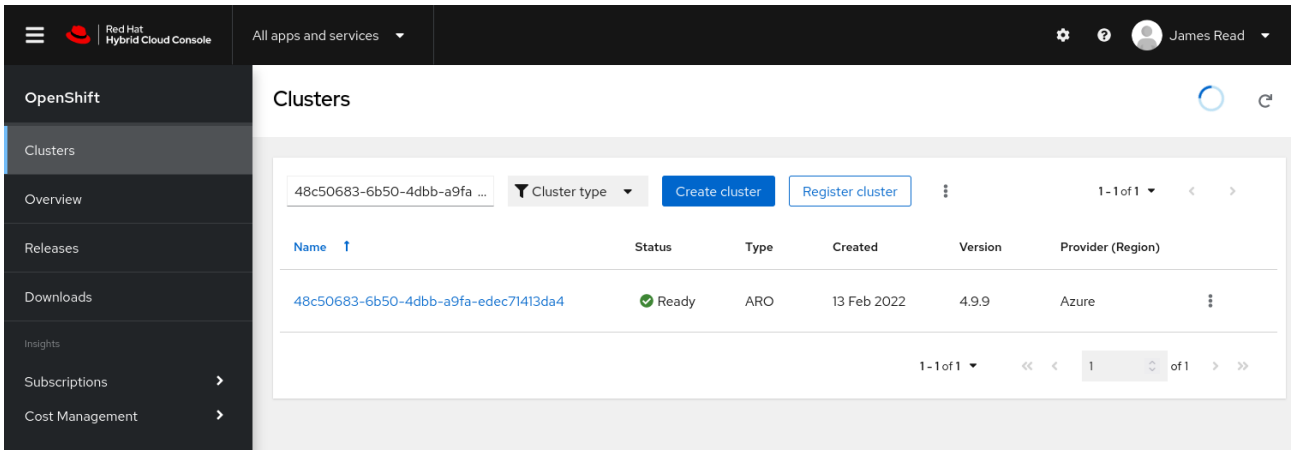


Figura 6.8: OpenShift Cluster Manager mostrando um cluster do Azure Red Hat OpenShift

É muito fácil configurar um pull secret para `cloud.redhat.com` e, além de aparecer no portal do OpenShift Cluster Manager em http://console.redhat.com, você também poderá abrir tickets de suporte diretamente com a Red Hat por meio do sistema de tickets de suporte padrão.

Veja aqui as instruções sobre como configurar um pull secret:

- Como adicionar ou atualizar um pull secret

Outro benefício de configurar um pull secret é que os clientes podem abrir tickets de suporte diretamente com a Red Hat. O pull secret dá um direito na sua conta da Red Hat de que seu cluster seja visto pela equipe de suporte. Abrir um ticket de suporte para o Azure Red Hat OpenShift funciona da mesma forma que qualquer outra solução Red Hat.

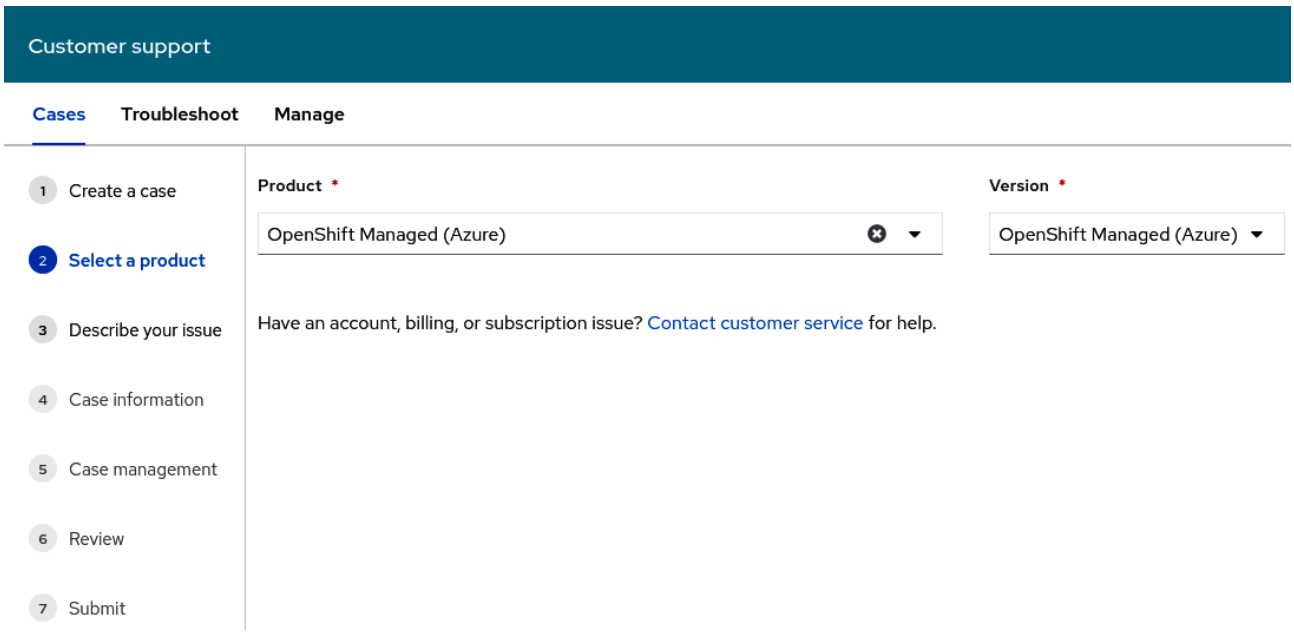


Figura 6.9: Criação de um caso de suporte

Intervalos de limites

Quando uma equipe de desenvolvimento ou de aplicações começa a implantar aplicações em containers, será questão de tempo até que uma aplicação tenha um "mau comportamento" e comece a consumir recursos desnecessários no cluster. Um exemplo pode ser uma aplicação defeituosa com vazamento de memória ou uma aplicação que foi definida erroneamente para escalar após consumir apenas 10% da CPU em vez de 100%. Para prevenir cenários em que essas aplicações escalem de maneira que fique fora de controle e consumam recursos demais, recomenda-se o uso do `LimitRange`.

O `LimitRange` permite que você restrinja o consumo de recursos para objetos específicos em um projeto. O `LimitRange` pode ser aplicado em:

- Pods e containers: é possível definir requisitos mínimos e máximos para CPU e memória para pods em seus containers.
- Fluxos de imagens: é possível definir limites no número de imagens e tags em um objeto de `ImageStream`.
- Imagens: é possível limitar o tamanho das imagens que podem ser extraídas para um registro interno.
- **Reivindicações de volumes persistentes (PVCs):** é possível restringir o tamanho das PVCs que podem ser solicitadas.

Veja um exemplo de um intervalo de limite que se aplica a containers, limitando o mínimo, o máximo e as solicitações de CPU padrão e memória permitidas:

```
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"
```

Esse snippet de código do `LimitRange` pode ser aplicado copiando e colando o YAML diretamente no editor do console do Red Hat OpenShift, ou de um arquivo com `oc apply -f limitrange.yaml`.

Documentação sobre intervalos de limites

Armazenamento persistente

As máquinas virtuais implantadas pelo Azure Red Hat OpenShift vêm com discos do Azure anexados para a instalação do Red Hat CoreOS, e com o serviço do Azure Red Hat OpenShift em execução. Esses discos não devem ser usados pelas aplicações, apenas para o próprio cluster do OpenShift.

As aplicações que exigem armazenamento persistente devem usar a funcionalidade `PersistentVolume` do Kubernetes, e a página Introdução ao armazenamento persistente na documentação do OpenShift descreve esse conceito em detalhes. Embora o Azure Red Hat OpenShift tenha suporte técnico a todos os provedores do `PersistentVolume` como uma instalação autogerenciada do OpenShift, no Azure, o armazenamento persistente mais usado é o seguinte:

Nome	Tipo	Modos de acesso	Classe de armazenamento
Arquivos do Azure	Filesystem, fora de conformidade com POSIX	ReadWriteOnce	Arquivo do Azure
Disco do Azure	Bloco	ReadWriteOnce	Disco do Azure
Red Hat OpenShift Data Foundation	Filesystem, bloco, objeto	(diversos)	Documentos OCS/ODF

Segurança e conformidade

A documentação do Red Hat OpenShift agora inclui bastante conteúdo relevante para entender como implementar muitos controles de segurança e manter a conformidade.

Documentação sobre segurança do OpenShift e conformidade

Esta seção da documentação inclui:

- Uma descrição detalhada de como a segurança de containers funciona no OpenShift. É importante entender que o OpenShift oferece muitos controles de segurança prontos para uso para containers que não são encontrados em outros serviços baseados em Kubernetes, e esses controles ajudam a manter a segurança de sua organização e suas aplicações.
- Verificação de vulnerabilidades em pods
- Acesso a logs de auditoria
- Configuração de certificados

Também inclui diversos operadores relacionados à segurança, como:

- **Operador de conformidade:** executa verificações e faz recomendações de correções de problemas de segurança comuns.
- **Verificação de integridade de arquivos:** verifica continuamente se os arquivos, principalmente arquivos sensíveis de configuração de segurança, não mudaram.

Resumo

Neste capítulo, abordamos a maioria das tarefas e tópicos normalmente considerados pelas organizações ao tornar um cluster de implantação adequado para aplicações de produção. Embora não seja necessário executar cada um desses tópicos em cada implantação subsequente, o primeiro cluster implantado deve considerar armazenamento persistente, limites, autenticação e vários outros tópicos mencionados neste capítulo.

Em geral, uma organização tentará automatizar as tarefas de pós-provisionamento o máximo possível. Por exemplo, a configuração do Azure Active Directory pode ser principalmente realizada usando um script do PowerShell ou alguns modelos de ARM. Isso ajudará a reduzir o tempo gasto em tarefas repetitivas caso você esteja implantando muitos clusters.

O próximo capítulo deste livro mostrará um exemplo de implantação de uma aplicação em cima de seu cluster pronto para produção do Azure Red Hat OpenShift.

Capítulo 7

Implantação de uma aplicação de amostra

O objetivo principal deste guia é ajudar os profissionais técnicos, nas funções operacionais e de desenvolvimento, que estão tentando entender o que é necessário para iniciar a produção com o Azure Red Hat OpenShift. Este guia supõe que o leitor tenha um conhecimento básico do Red Hat OpenShift, uma vez que grande parte da arquitetura, dos portais de gerenciamento e da experiência do usuário é igual à do Azure Red Hat OpenShift.

Este capítulo é uma breve introdução à implantação de uma aplicação de amostra simples, chamada "Fruit Smoothies". O objetivo dessa aplicação é ser um guia de início rápido para que você possa entender melhor como usar o Azure Red Hat OpenShift. Observe que ela é uma aplicação de amostra mantida para o Azure Kubernetes Service e a implantação dela no Azure Red Hat OpenShift é feita para demonstrar que o OpenShift é totalmente compatível com o Kubernetes.

A maior parte deste capítulo é baseada em conteúdos do http://aroworkshop.io, com adição de mais descrições e contextos.

Uma visão geral da aplicação de classificações

A arquitetura de aplicações é simples:

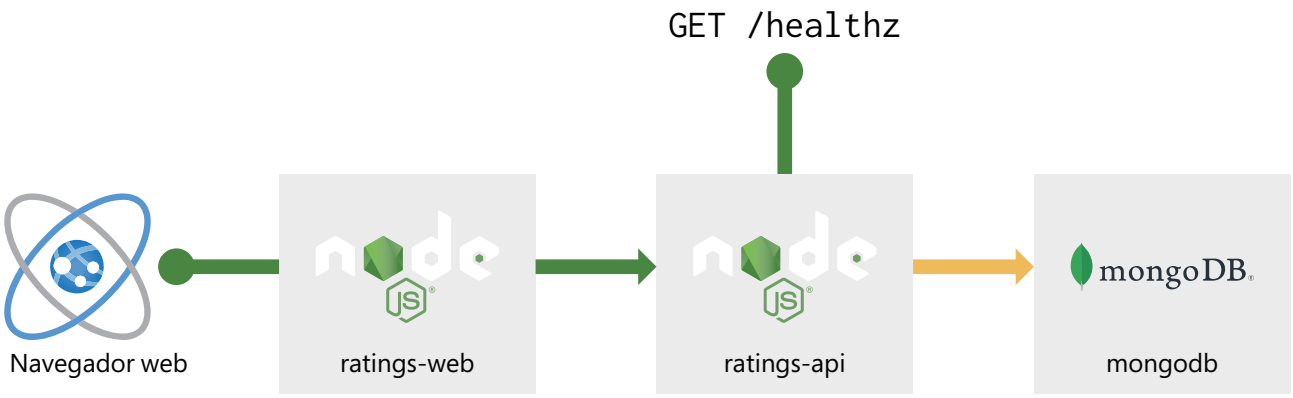


Figura 7.1: Arquitetura da aplicação Fruit Smoothies

No diagrama anterior, é possível ver que a aplicação é composta por três serviços e dois endpoints públicos, como mostrado na tabela abaixo. O primeiro endpoint à esquerda do diagrama representa o navegador do usuário, visualizando a aplicação web com HTML público. O segundo endpoint, `healthz`, é uma verificação de integridade no serviço `ratings-api`. Veja as descrições dos serviços individuais e os links para os repositórios na tabela a seguir:

Componentes	Descrição	Repositório do GitHub
`rating-web`	Um front-end web voltado para o público: o "site".	Repositório do GitHub
`rating-api`	Este serviço armazena a entrada da IU web no banco de dados. Ele também exibe resultados do banco de dados à aplicação web na porta 3000.	Repositório do GitHub
`mongodb`	Um banco de dados pré-carregados NoSQL.	Dados

Acesse os links do repositório do GitHub para conhecer e entender melhor essas aplicações. Nas instruções a seguir, há um guia passo a passo sobre como implantar cada uma dessas aplicações.

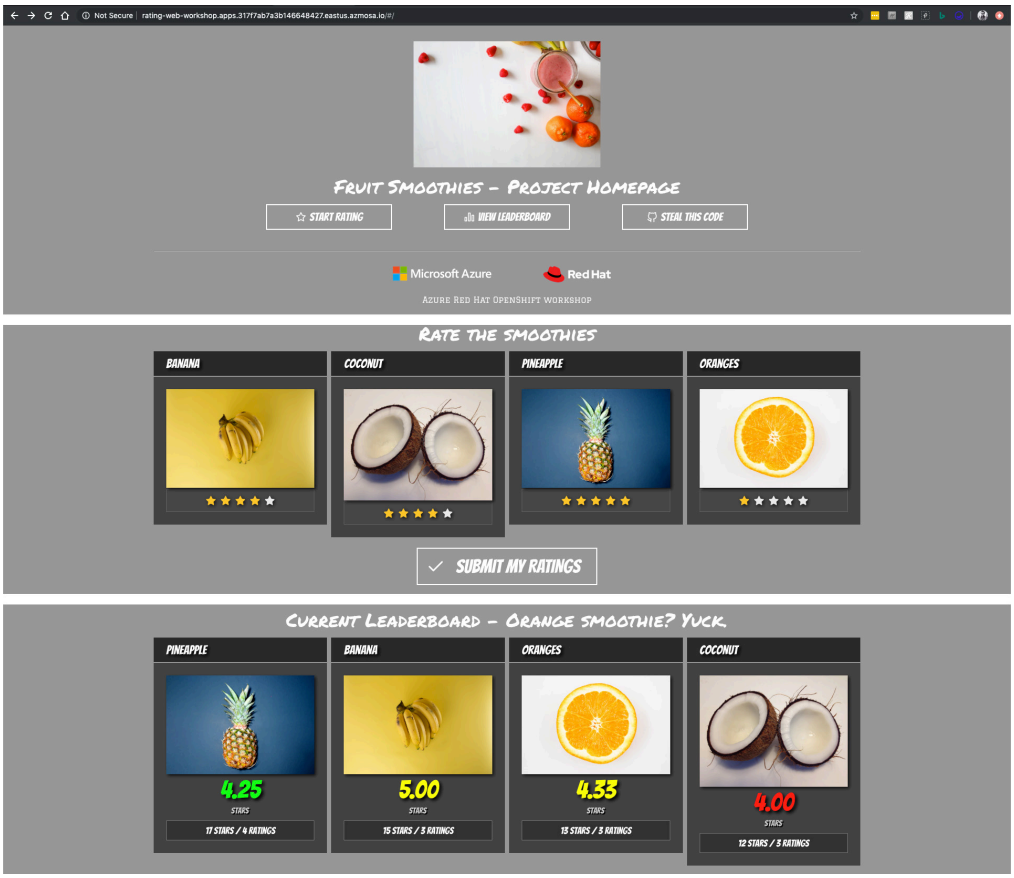


Figura 7.2: Capturas de tela mostrando uma visão geral da aplicação Fruit Smoothies.

Ao terminar, você estará executando uma aplicação web parecida com as capturas de tela anteriores. Você também entenderá melhor como as equipes operacionais e de desenvolvimento implantam aplicações no Azure Red Hat OpenShift, o que ajudará no seu próprio processo de tomada de decisões ao implantar suas aplicações no Azure Red Hat OpenShift.

Crie um cluster e se conecte a ele

No restante deste capítulo, supomos que você tem um ambiente do Azure Red Hat OpenShift em execução para trabalhar. Não existem requisitos especiais para essa aplicação de classificações, ela será implantada em um ambiente do Azure Red Hat OpenShift novo e vazio. Se você ainda não provisionou um cluster, estude estes capítulos:

- *Capítulo 4: Pré-provisionamento: questões sobre arquitetura empresarial*
- *Capítulo 5: Provisionamento de um cluster do Azure Red Hat OpenShift*

A seção *Como acessar o cluster* no *Capítulo 5: Provisionamento de um cluster do Azure Red Hat OpenShift* é uma boa revisão sobre como acessar um cluster que você já provisionou.

Faça login no console web

Todos os clusters do Azure Red Hat OpenShift têm um endereço DNS para o console web do OpenShift. Use o comando `az aro list` para listar os clusters da sua subscrição atual do Azure:

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

Será listada a URL do console web do cluster. Abra esse link em uma nova aba do navegador e faça login com o usuário `kubeadmin` ou use outra conta de usuário com permissões para criar projetos.

Depois de fazer login, você verá o console web do Azure Red Hat OpenShift.

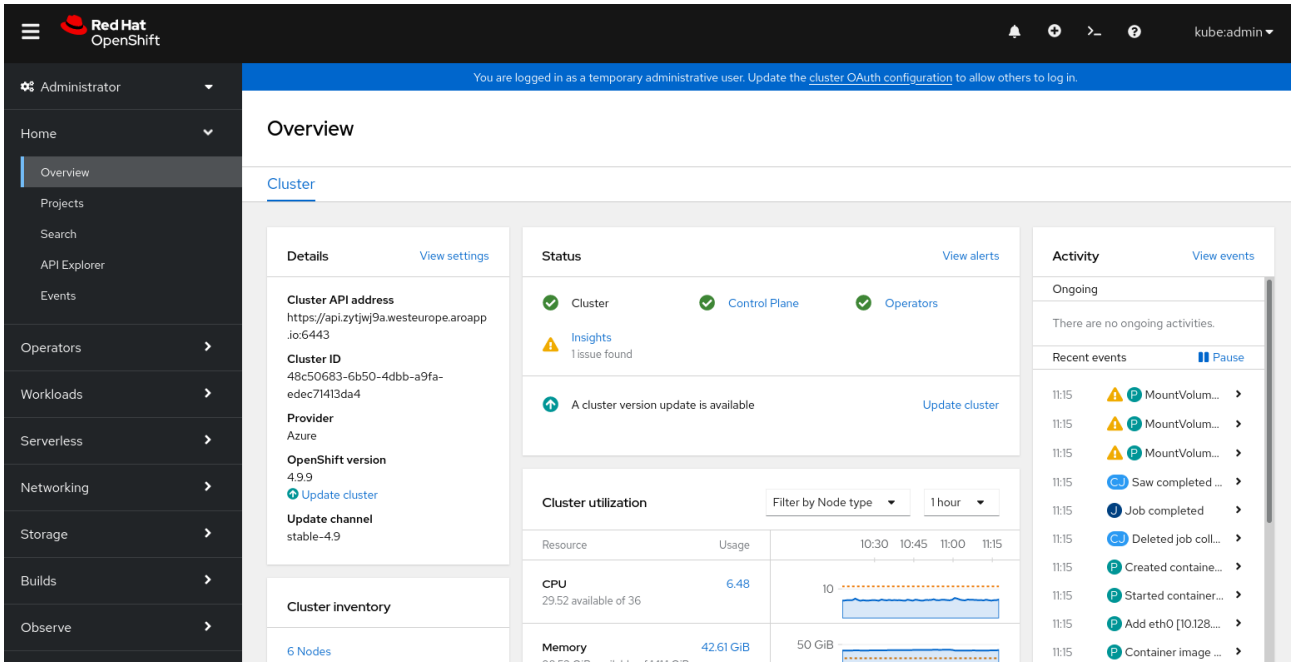


Figura 7.3: Console web do Azure Red Hat OpenShift

Instale o cliente OpenShift

Abra o Azure Cloud Shell ou use o terminal do Linux local e instale o cliente da linha de comando do OpenShift. É preciso fazer isso para acessar o cluster da linha de comando. Veja aqui as instruções para fazer isso:

```
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
```

Outra alternativa para Windows ou Mac é fazer o download nestes links:

- 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

Você será capaz de executar o comando oc de um dos pacotes contidos nos downloads.

Recupere o comando e o token do login

Depois de instalar o cliente, é preciso obter um token para fazer login no cluster. Faça login no console web do OpenShift, clique no nome de usuário no canto direito superior e, em seguida, clique em **Copiar comando de login**.

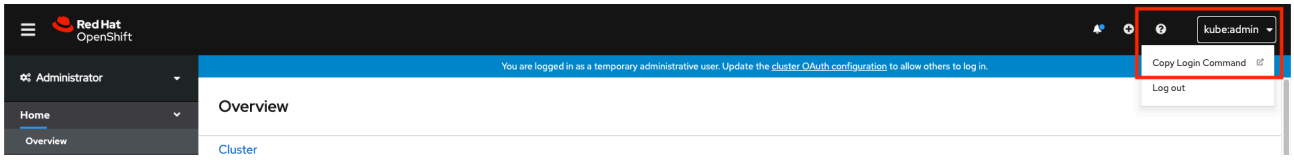


Figura 7.4: Copiar comando de login para fazer login no cluster

Cole o comando de login no seu shell (terminal local do Linux ou Azure Cloud Shell). Você será capaz de se conectar ao cluster.

Figura 7.5: Use o comando de login para se conectar ao cluster.

Depois de estabelecer a conexão, podemos criar o projeto.

Crie um projeto

Um projeto no OpenShift é como uma pasta lógica feita para armazenar a aplicação de classificações. No Red Hat OpenShift, todos os containers e aplicações precisam estar dentro de algum projeto. Você pode usar projetos para separar aplicações e até mesmo departamentos. As próximas instruções explicam como criar um projeto.

É possível criar um projeto na interface web, mas essas instruções usam a linha de comando.

```
oc new-project workshop
```

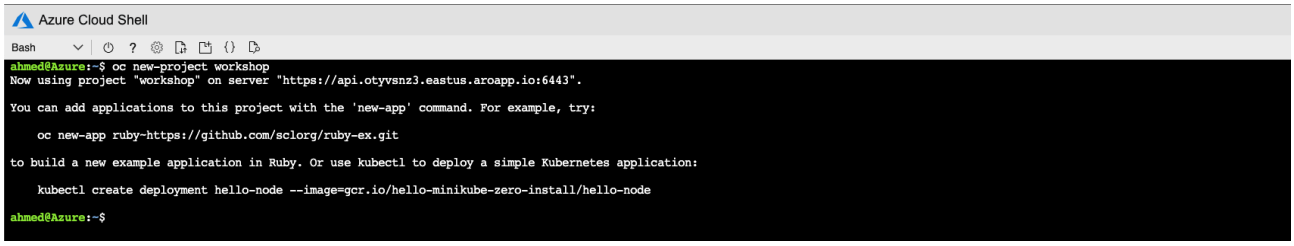


Figura 7.6: Crie um novo workshop no Azure Cloud Shell

Depois de criar o projeto, é possível mudar para ele usando o `oc project workshop`. A próxima etapa será implantar o primeiro dos três microsserviços apresentados no início deste capítulo: o MongoDB. Ele será implantado no projeto que acabamos de criar.

Recursos

- Documentação do Azure Red Hat OpenShift: introdução à CLI
- Documentação do Azure Red Hat OpenShift: projetos

Implante MongoDB

O Azure Red Hat OpenShift oferece imagem e template em containers para facilitar a criação de um novo serviço de banco de dados do MongoDB. O template oferece campos de parâmetros para definir todas as variáveis de ambiente obrigatórias (usuário, senha, nome do banco de dados etc.) com padrões pré-definidos, incluindo a autogeração dos valores de senha. Ele também define uma configuração de implantação e um serviço.

A instância do MongoDB será implantada diretamente como imagem de container do Docker Hub. Com o OpenShift, tudo isso pode ser feito pelo console web. Na parte superior do menu, mude para a perspectiva do desenvolvedor, vá até a página **Adicionar** e selecione **Imagens de container**.

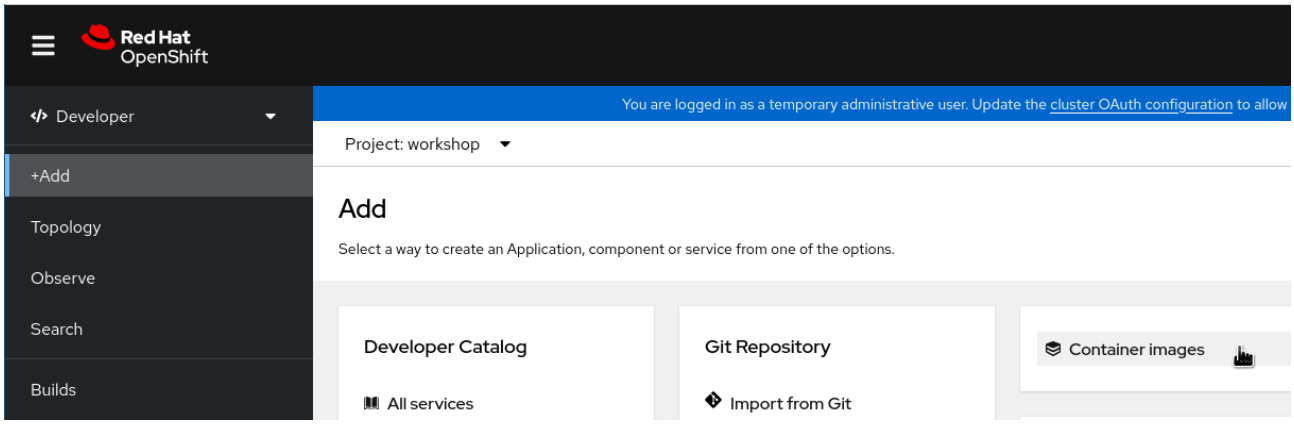


Figura 7.7: Adição de uma imagem de container

Você será direcionado à página **Implantar imagem**. Preencha o formulário desta forma:

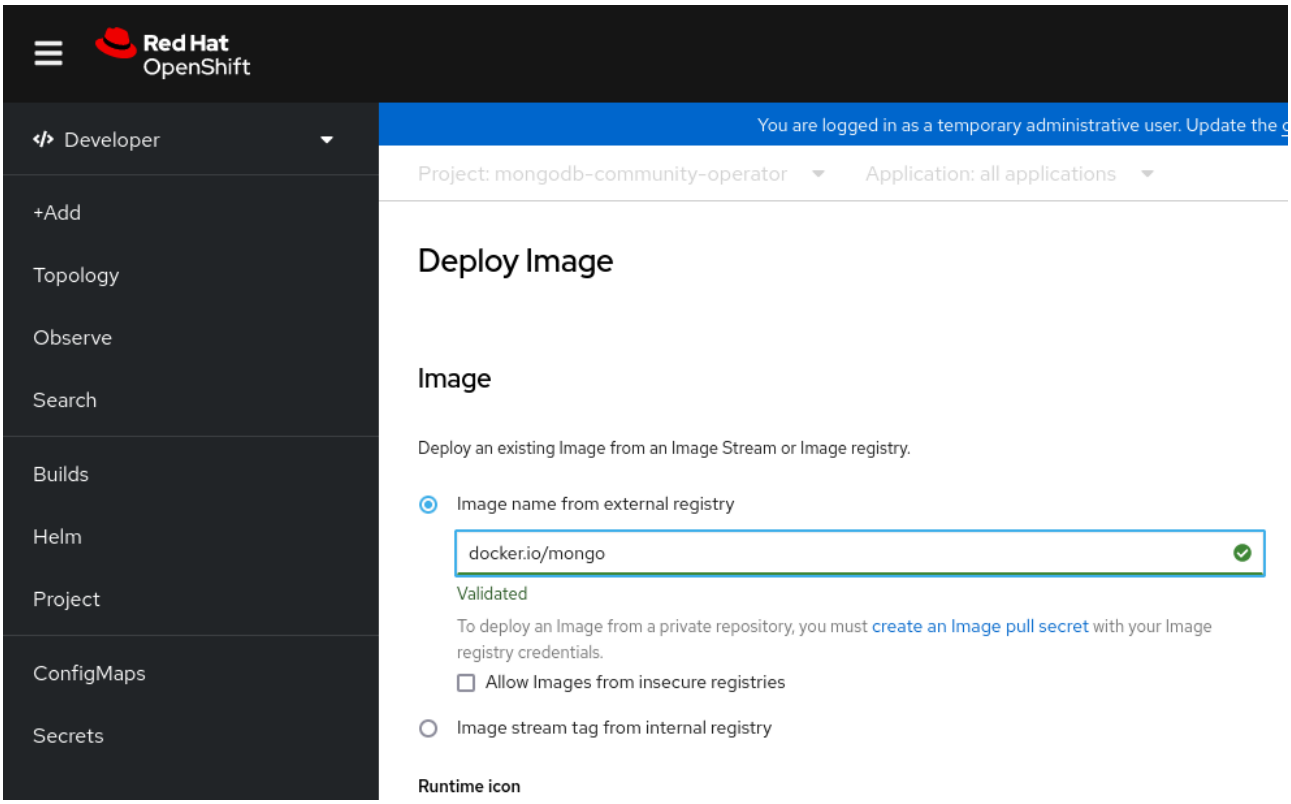


Figura 7.8: Preencha o formulário para implantar uma imagem

Defina os valores da seguinte maneira:

Campo	Valor
Nome da imagem do registro externo	`docker.io/mongo`
Ícone do ambiente de execução	`mongodb`
Nome da aplicação	`ratings`
Nome	`mongodb`
Crie uma rota para a aplicação	Desmarcado: uma rota viabilizaria o acesso externo ao banco de dados, o que não é obrigatório nesta aplicação.
Tipo de recurso	Implantação

No final do formulário, clique no link de implantação para expandi-lo e poder definir as variáveis de ambiente.

A seguinte variável de ambiente funciona como um padrão para iniciar o banco de dados na primeira inicialização:

Nome	Valor
MONGO_INITDB_DATABASE	ratingsdb

Não foi definido nenhum nome de usuário ou senha para o banco de dados do MongoDB, essa é a configuração padrão e a autenticação será desativada.

Verifique a variável de ambiente mais uma vez e clique no botão **Criar** para continuar e implantar o container do MongoDB.

Após alguns instantes, a instância do MongoDB será colocada em execução no projeto da `área de trabalho`. Para ver essa implantação, mude a visualização para **Topologia**.

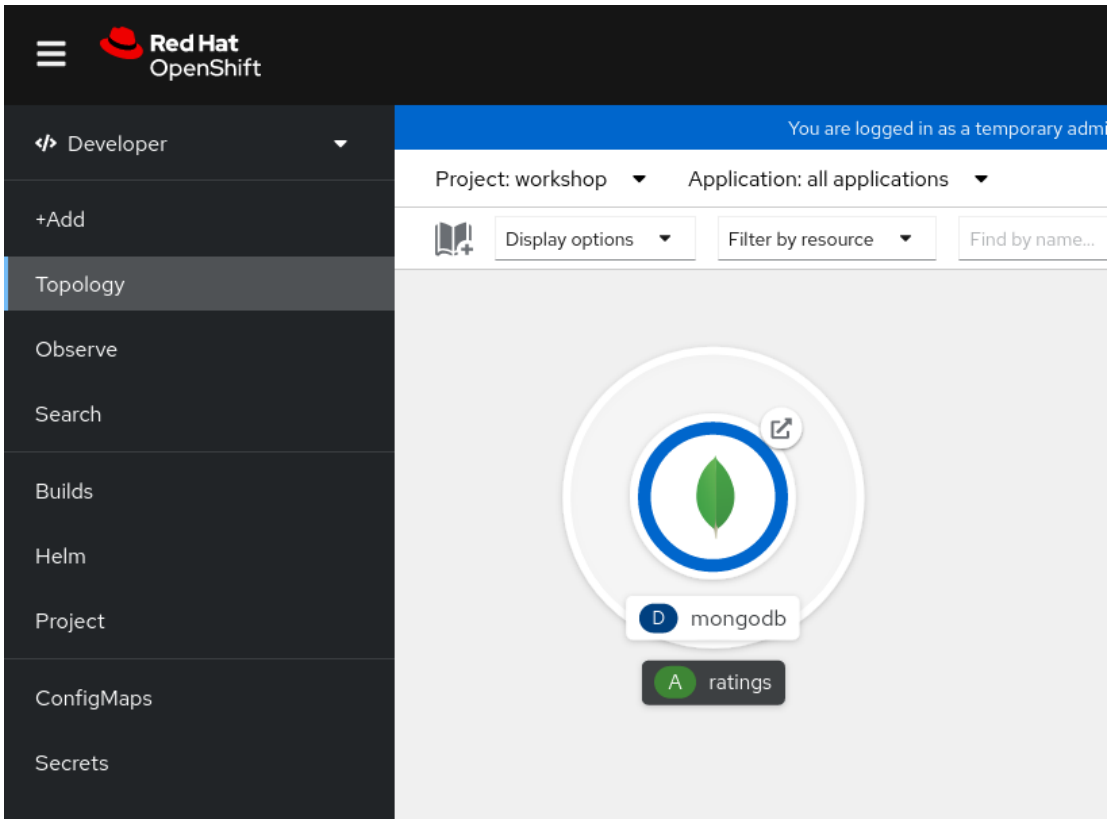


Figura 7.9: A instância do MongoDB está em execução

Execute o comando `oc get all` para ver o status da aplicação nova e verificar se a implantação do template do MongoDB foi concluída. Veja a seguir um exemplo de resultado do `oc get all`:

```
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
```

Se tudo estiver funcionando corretamente, a coluna STATUS mostrará o container como "Running".

Recupere o nome do host do serviço do MongoDB

Depois que a implantação é concluída, precisamos encontrar o serviço criado para poder acessar o banco de dados de dentro do cluster. svc é a abreviação de serviços:

```
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
```

O serviço poderá ser acessado pelo nome DNS `mongodb.workshop.svc.cluster.local`, que é formado por `[service name].[project name].svc.cluster.local`. Isso resolve apenas dentro do cluster.

Implantar a API de classificações

Agora é hora de implantar a segunda aplicação, que é `rating-api`. Ela é uma aplicação Node.js que se conecta a uma instância do MongoDB para recuperar e classificar itens. Veja alguns detalhes que você precisa saber para implantar essa aplicação:

- `rating-api` no GitHub
- O container expõe a porta `3000`
- Uma conexão MongoDB é configurada usando uma variável de ambiente chamada `MONGODB_URI`

Anote essas informações, você precisará delas nas próximas seções.

Ramifique a aplicação para seu repositório do GitHub

Para fazer alterações, como adicionar webhooks de CI/CD, você precisará da sua própria cópia do código `ratings-api`. No Git, isso se chama uma "ramificação". É possível ramificar a aplicação no seu repositório do GitHub. Acesse o repositório anterior do GitHub e clique no botão **Ramificar**.

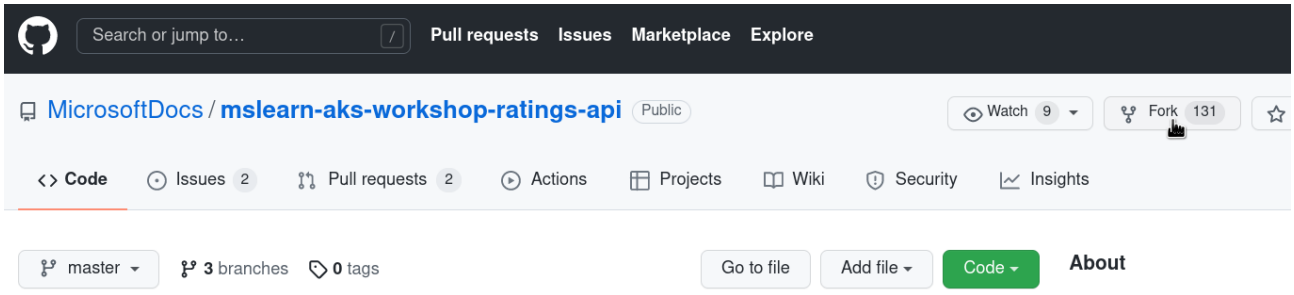


Figura 7.10: Ramificar a aplicação no repositório do GitHub

Anote o endereço desse repositório novo, você precisará usá-lo nas próximas instruções.

Use a CLI do OpenShift para implantar a rating-api

Com o OpenShift, é possível implantar o código diretamente do repositório Git. Para fazer, veja o conteúdo e selecione um "compilador de imagem" baseado nos conteúdos Java, PHP, Perl, Python ou similares. Neste caso, `rating-api` é uma aplicação JavaScript. O compilador de imagem fará o download das dependências JavaScript usando npm e o resultado será uma nova imagem de container armazenada no registro de containers interno do OpenShift. Essa estratégia de compilação se chama **Source 2 Image (S2I)** e está detalhada no glossário.

Você pode iniciar uma nova compilação S2I usando o oc new-app:

```
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
```

Mude a visualização para **Topologia** no console web para ver a compilação de aplicação inicializar e a implantação ser concluída após alguns minutos.

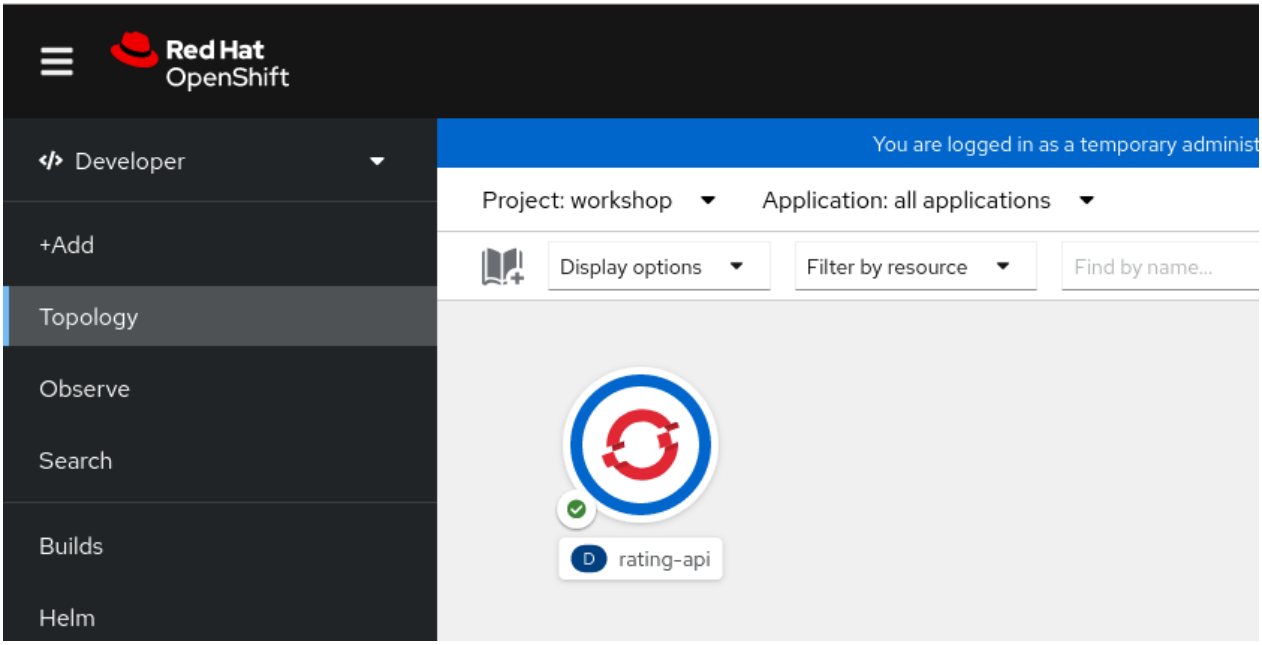


Figura 7.11: A visualização de topologia

A compilação e inicialização do pod levam um ou dois minutos.

Configure as variáveis de ambiente obrigatórias

Aqui o banco de dados é implantado, assim como a API de classificações. No entanto, é preciso ensinar a API de classificações a se conectar ao banco de dados. Normalmente, aplicações baseadas em containers são configuradas usando variáveis de ambiente.

Clique na implantação e a edite. Crie a seguinte variável de ambiente:

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

Quando essa nova variável de ambiente for salva, a reimplantação do serviço de `ratings-api` será acionada para usá-la.

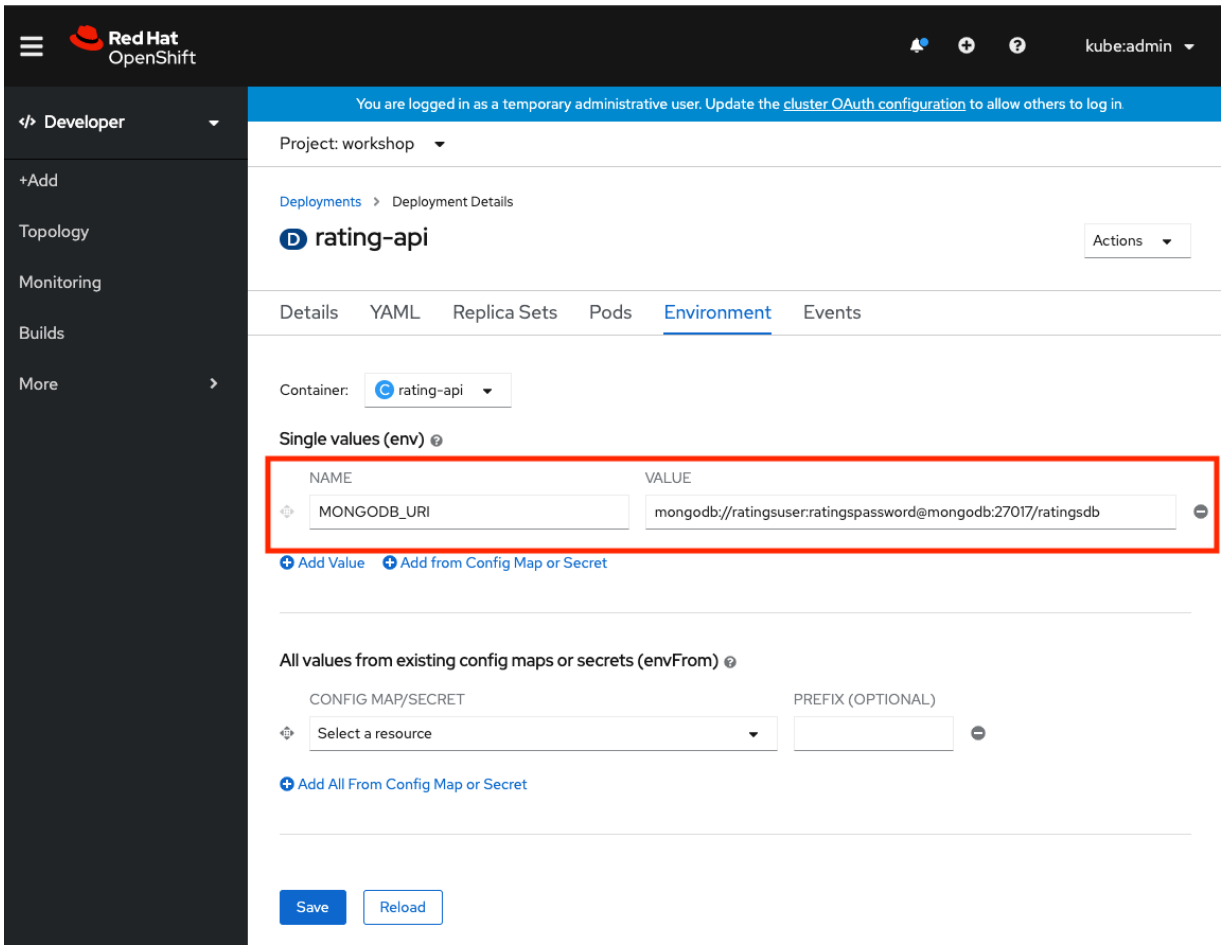


Figura 7.12: Como configurar a variável de ambiente MONGODB_URI pelo console web

Isso também pode ser feito na linha de comando:

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

Seja qual for o método, o OpenShift precisará reiniciar o container para usar as novas variáveis de ambiente.

Verifique se o serviço está em execução

Se você for até os logs da implantação `rating-api`, você verá uma mensagem de log confirmando que o código consegue se conectar ao MongoDB. Para fazer isso, na tela de detalhes da implantação, clique na aba **Pods** e, em seguida, em um dos pods.

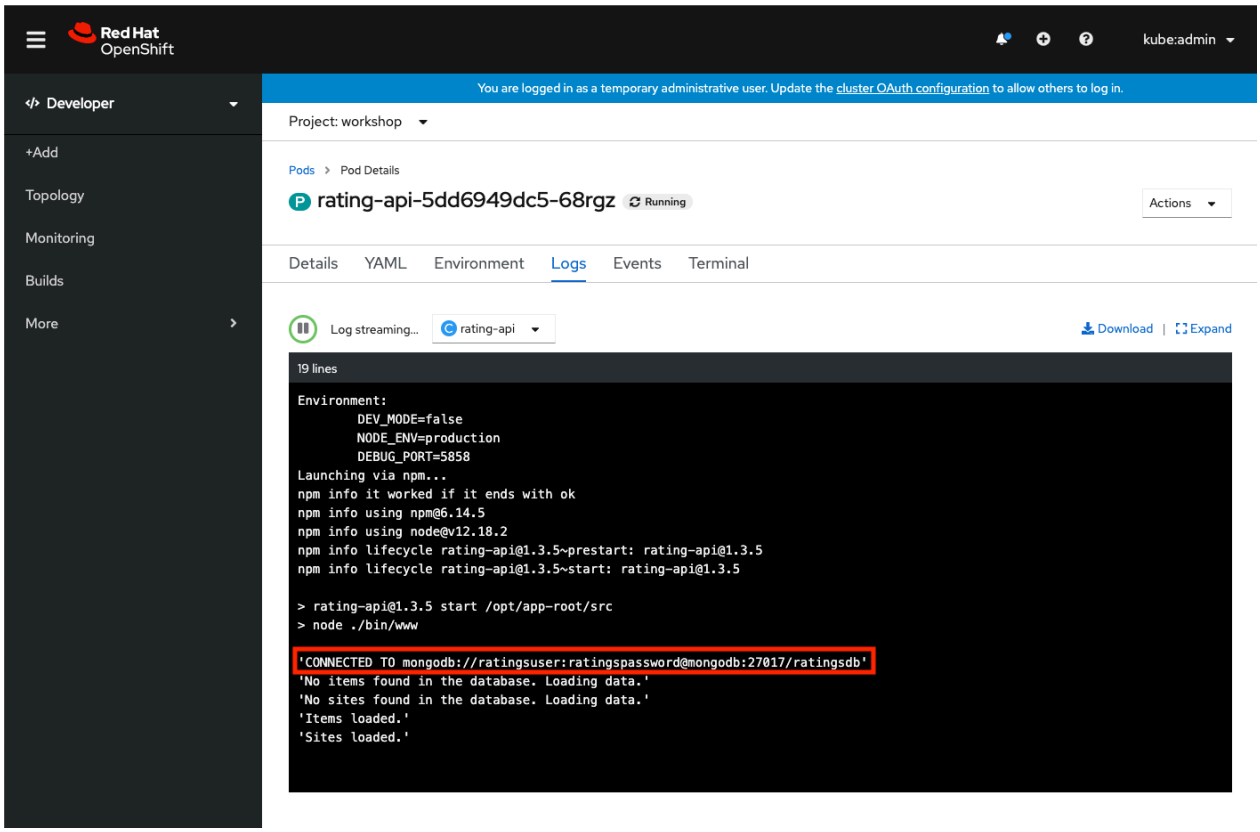


Figura 7.13: Mensagem de log confirma que o código pode se conectar ao MongoDB

Ajuste a porta do serviço rating-api

O OpenShift criará um serviço usando a porta `8080`. No entanto, devido a uma atualização da biblioteca, esse serviço é executado na porta `3000`. É preciso editar o serviço padrão.

Vá até o menu **Rede→ Serviços** e, na lista de serviços, edite o serviço `rating-api` desta forma (é só substituir `8080` por `3000`):

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

Reinicie o serviço para usar a nova porta:

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

Agora o serviço está vinculado à porta certa, `3000`, em vez de `8080`.

Recupere o nome do host do serviço rating-api

Precisamos validar a existência de um serviço `rating-api`, já que ele será usado na próxima seção, na implantação da aplicação `rating-web`:

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

O serviço poderá ser acessado pelo nome DNS `rating-api.workshop.svc.cluster.local:3000`, na porta `3000`, que é formada por `[service name].[project name].svc.cluster.local`. Isso resolve apenas dentro do cluster.

Implante o front-end das classificações

`rating-web` é uma aplicação Node.js que se conecta à `rating-api`. Veja alguns detalhes que você precisa saber para implantar essa aplicação:

- `rating-web` no GitHub
- O container expõe a porta `8080`
- A app web se conecta à API no DNS do cluster interno, usando um proxy em uma variável de ambiente chamada `API`

Use a CLI do OpenShift para implantar o rating-web

Assim como na `rating-api`, essa aplicação pode ser implantada com o S2I, usando o `oc new-app`. No entanto, agora a aplicação será criada com uma estratégia Dockerfile, usando um Dockerfile que já está no repositório Git. Então, não especifique um argumento `--strategy`:

```
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
```

A compilação das dependências em um container leva um tempo, espere entre 2 e 3 minutos para que a criação seja concluída antes de voltar para a visualização de **Topologia** e ver o serviço ficar online.

Cofigure as variáveis de ambiente obrigatórias

Crie a variável de ambiente da `API` para a configuração de implantação do `rating-web`. O valor dessa variável será o nome do host e a porta do serviço `rating-api`.

Em vez que definir a variável de ambiente no console web do Azure Red Hat OpenShift, é possível fazer isso pela CLI do OpenShift:

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

Exponha o serviço rating-web usando uma rota

Expor o serviço significa a existência de uma URL acessível publicamente que pode ser usada pelos usuários. Se o serviço não está exposto, ele só pode ser acessado de dentro do cluster:

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

Por fim, pegue a URL do serviço que acabou de ser exposto:

```
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
```

Observe que o **nome do domínio totalmente qualificado (FQDN)** contém o nome padrão da aplicação e do projeto. O restante do FQDN, o subdomínio, é seu subdomínio de apps específico para clusters do Azure Red Hat OpenShift.

Teste o serviço

Abra o nome do host no navegador. Você verá a página da app de classificações. Faça um teste, envie alguns votos e verifique o ranking.

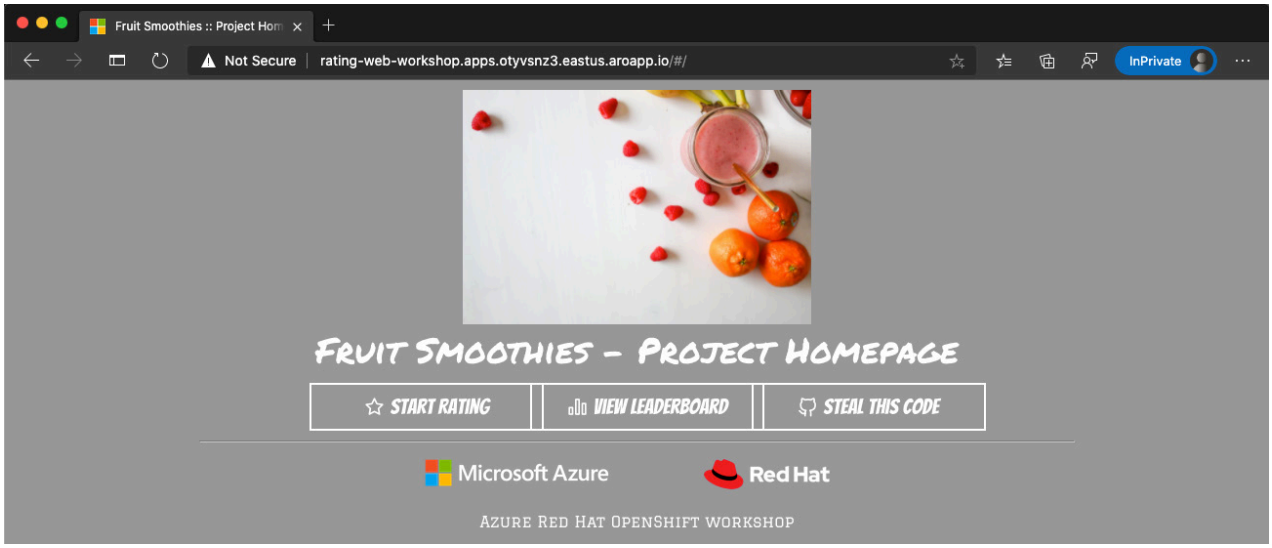


Figura 7.14: Como testar o serviço da app de classificações

Configure o webhook do GitHub

Para acionar as compilações S2I ao enviar um código para o repositório do GitHub, será preciso configurar o webhook do GitHub:

1. Recupere o segredo do gatilho do webhook do GitHub. Você precisará usar esse segredo na URL do webhook do GitHub:

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

   Anote a chave do segredo, você precisará dela em algumas etapas à frente.

2. Recupere a URL do gatilho do webhook do GitHub na configuração da compilação:

```
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. Substitua o espaço reservado <secret> pelo segredo recuperado no primeiro passo. Neste caso, o segredo é `3ffcc8d5-a243` e a URL resultante é:

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

   Você usará essa URL para configurar o webhook no repositório do GitHub.

4. Vá até seu repositório do GitHub. Selecione **Adicionar webhook** em **Configurações** → **Webhooks**.
5. No campo **URL de carga**, cole a URL do GitHub com o valor do <secret> alterado para usar seu segredo.
6. No campo **Tipo de conteúdo**, mude o padrão `application/x-www-form-urlencoded` para `application/json`.

7. Clique em **Adicionar webhook**.

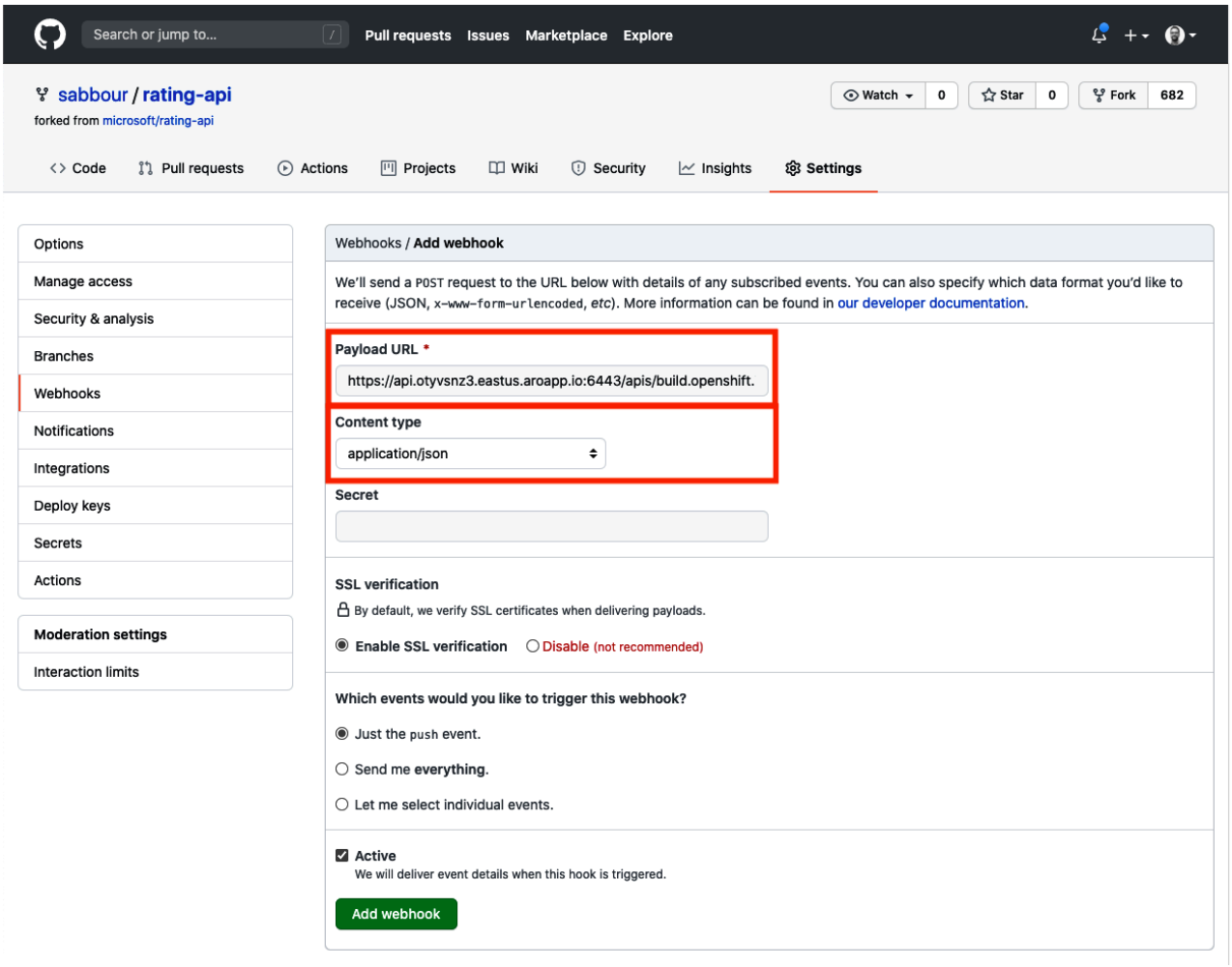


Figura 7.15: Adicionar um webhook

Você verá uma mensagem do GitHub confirmando a configuração do webhook.

Agora sempre que você enviar uma alteração para o repositório do GitHub, uma nova compilação será automaticamente inicializada. Se ela for bem-sucedida, isso iniciará um nova implantação.

Altere a app do site e veja a atualização contínua

Acesse o arquivo `https://github.com/<your GitHub username>/rating-web/blob/master/src/App.vue` no repositório do GitHub.

Edite o arquivo, altere a linha `background-color: #999;` para `background-color: #0071c5`.

Confirme as alterações no arquivo em uma ramificação `master`.

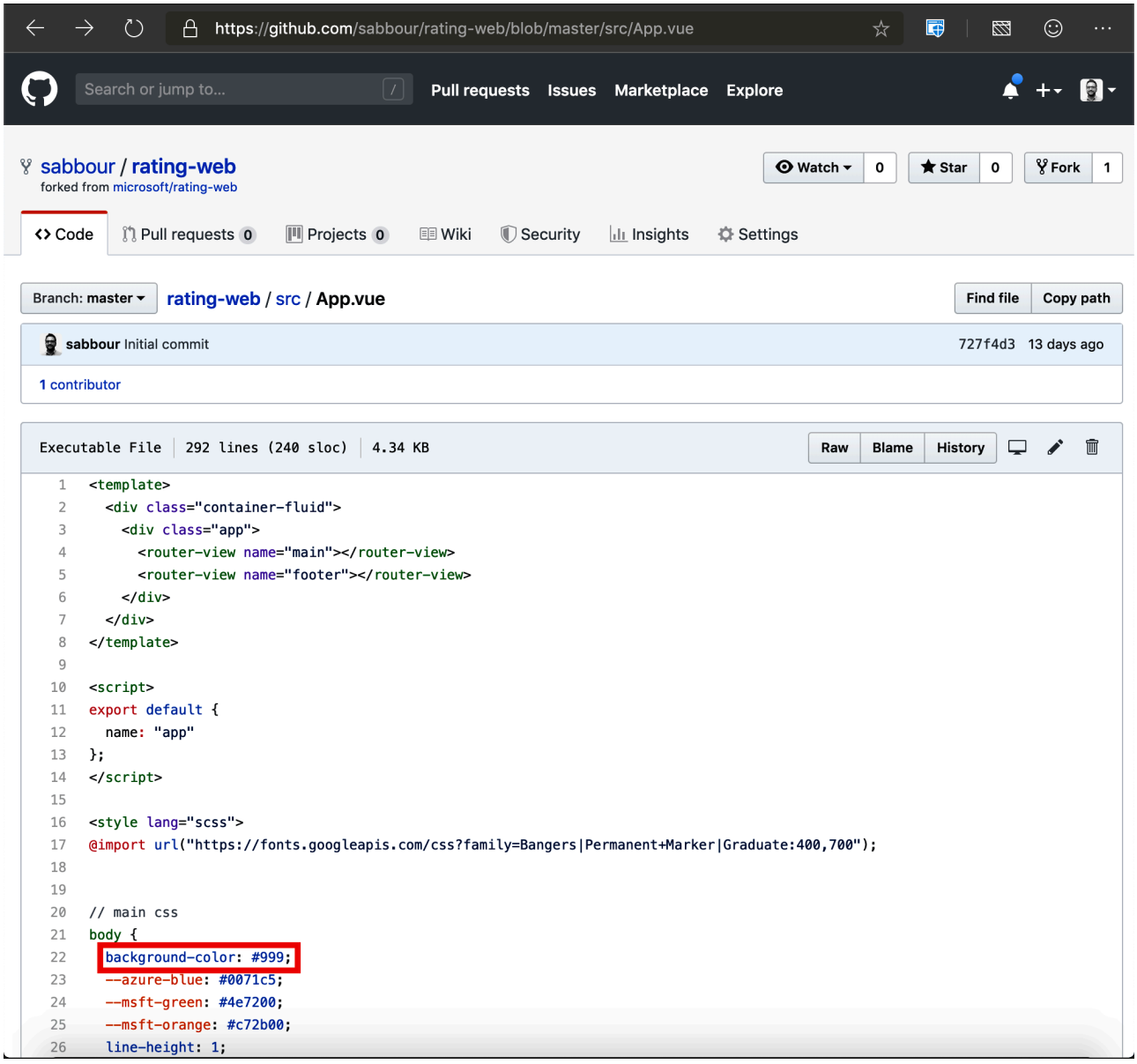


Figura 7.16: Confirme as alterações no arquivo em uma ramificação master

Logo em seguida, acesse a aba **Compilações** no console web do OpenShift. Você verá uma nova compilação enfileirada acionada pelo envio. Isso acionará uma nova implantação e você verá que a cor do site foi atualizada.

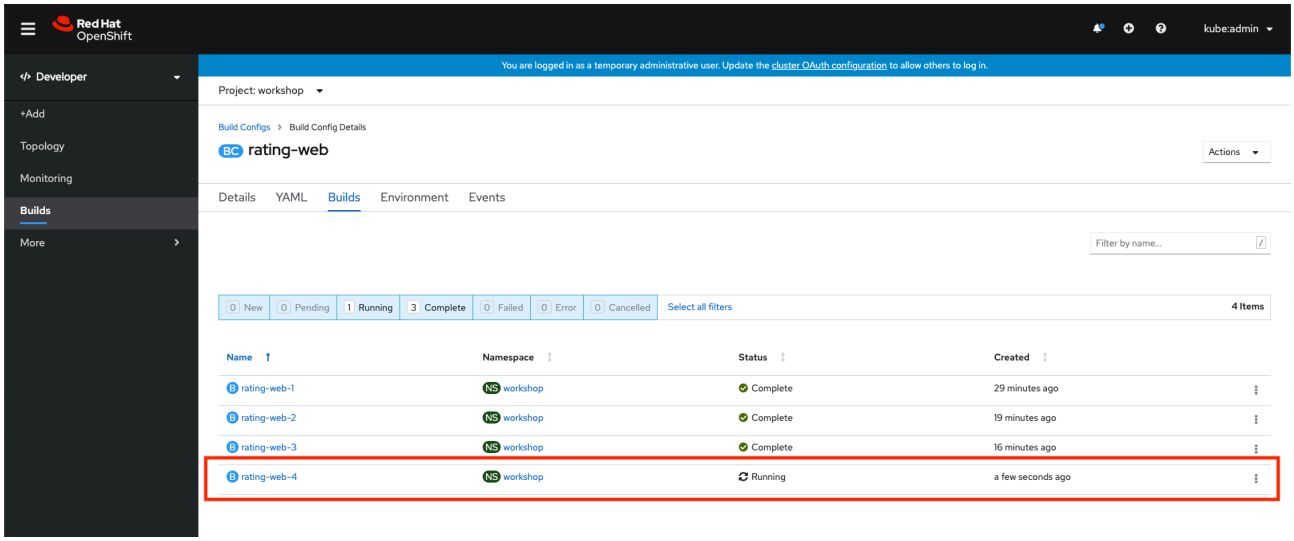


Figura 7.17: A aba "Compilações" mostra a execução de uma nova compilação

Agora volte para a página `ratings-web` e, se tudo tiver dado certo, você verá que a cor de fundo mudou.

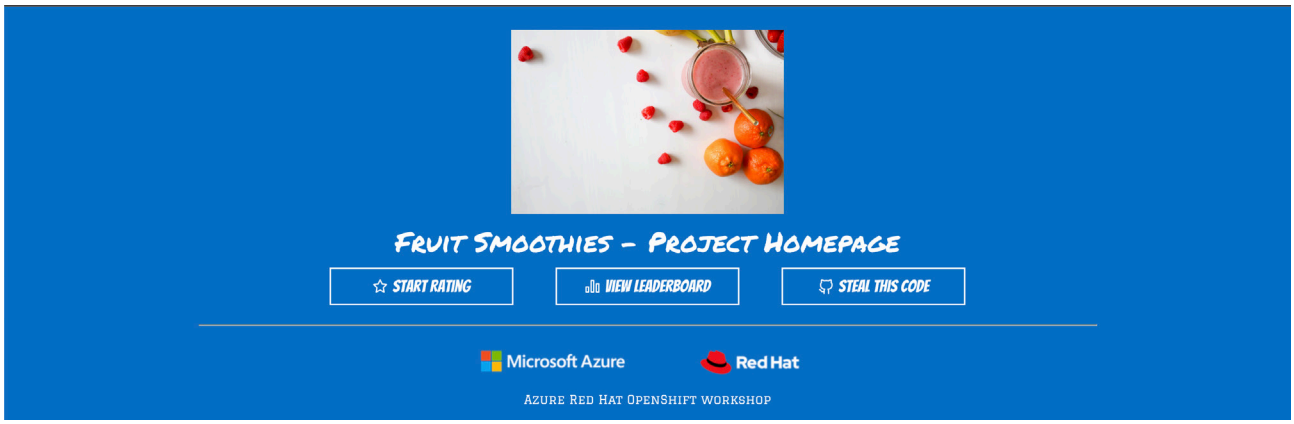


Figura 7.18: A página inicial do Fruit Smoothies com a cor nova

Resumo

Neste capítulo, nós implantamos uma aplicação básica composta por três aplicações de microsserviços menores: um banco de dados MongoDB , a `ratings-api` e o código `ratings-web`. Ela é diferente de uma aplicação de produção, mas funciona como uma rápida revisão conceitual para implantação de aplicações no Azure Red Hat OpenShift. Use essas instruções como base para implantar suas próprias aplicações.

O Red Hat OpenShift também oferece suporte para a implantação de aplicações do OperatorHub, do Helm Charts e de sistemas externos de CI/CD como o Azure DevOps. A melhor estratégia de implantação para sua empresa depende das ferramentas e tecnologias usadas internamente.

No próximo capítulo, vamos explorar outro valor do Azure Red Hat OpenShift, um valor que diferencia o OpenShift como uma plataforma de aplicações, desenvolvido com base no Kubernetes. Estudaremos as funcionalidades e os recursos feitos para dar suporte às necessidades complexas das aplicações empresariais.

Capítulo 8

Explorando a plataforma de aplicações

Nos capítulos anteriores, vimos como o Red Hat OpenShift oferece vários serviços desenvolvidos com base no Kubernetes. Esses serviços são combinados para criar uma verdadeira plataforma de aplicações e podem ser agrupados em uma destas cinco categorias: serviços de plataforma, serviços de aplicação, data services, serviços de desenvolvedor e serviços de cluster do Kubernetes.

Figura 8.1: Os serviços inclusos no Azure Red Hat OpenShift

Os componentes agrupados no **OpenShift Platform Plus (Gerenciamento multicluster**, **Segurança de cluster** e **Registro global**) são soluções adicionais que exigem subscrição. Eles são compatíveis com o Azure Red Hat OpenShift, mas não estão inclusos na oferta dessa solução.

Nas seções a seguir, vamos explorar alguns dos principais benefícios oferecidos por esses serviços e ver alguns links onde você pode encontrar maiores informações.

Serviços de cluster: registro de containers integrado

A implantação do cluster já traz um registro de containers interno e integrado no Red Hat OpenShift. Esse registro é usado tanto para serviços internos do cluster, como os operadores, quanto por padrão para containers de aplicações de clientes. Ele é padrão e não precisa de instalação. Até a manutenção é feita por um operador de infraestrutura.

Via de regra, todos os clientes que implantam um serviço de container do Kubernetes também precisam implantar o próprio registro de container e manter as imagens do container privadas. Com o OpenShift, não é preciso fazer outra instalação e configuração no segundo dia, já que o registro de container integrado já está disponível dentro do cluster. Esse exemplo mostra uma funcionalidade simples do OpenShift que economiza o seu tempo.

Visão geral do registro do OpenShift Container Platform integrado

Em geral, o administrador opta por expor o registro do container fora do cluster, para que os usuários externos ao OpenShift possam enviar imagens de container para o registro. O Azure Red Hat OpenShift oferece suporte completo para essa ação. Veja as instruções na documentação padrão do OpenShift para exposição do registro.

Serviços de plataforma: OpenShift Pipelines

Os clientes do Red Hat OpenShift podem criar aplicações de várias maneiras, e as ferramentas de CI/CD mais usadas, como Jenkins, CircleCI e GitHub Actions, têm plugins compatíveis com o OpenShift. No entanto, o OpenShift também oferece funcionalidades na plataforma para criação de aplicações usando pipelines de container nativos em nuvem, pelo operador do OpenShift Pipelines.

O OpenShift Pipelines é baseado no projeto da comunidade chamado Tekton. Todos os estágios do pipeline são executados dentro de um container, como extrair o código do repositório Git, executar um compilador Java ou montar um pacote RPM. Isso significa que desenvolvedores e operadores podem formar pipelines complexos e sofisticados, com todas as vantagens oferecidas pelos containers, para criar software.

É possível instalar o OpenShift Pipelines pelo OperatorHub. Basta ir até o **OperatorHub** e selecionar o operador para começar a instalação. Não é preciso configurar nada, e a instalação do operador será concluída em menos de um minuto.

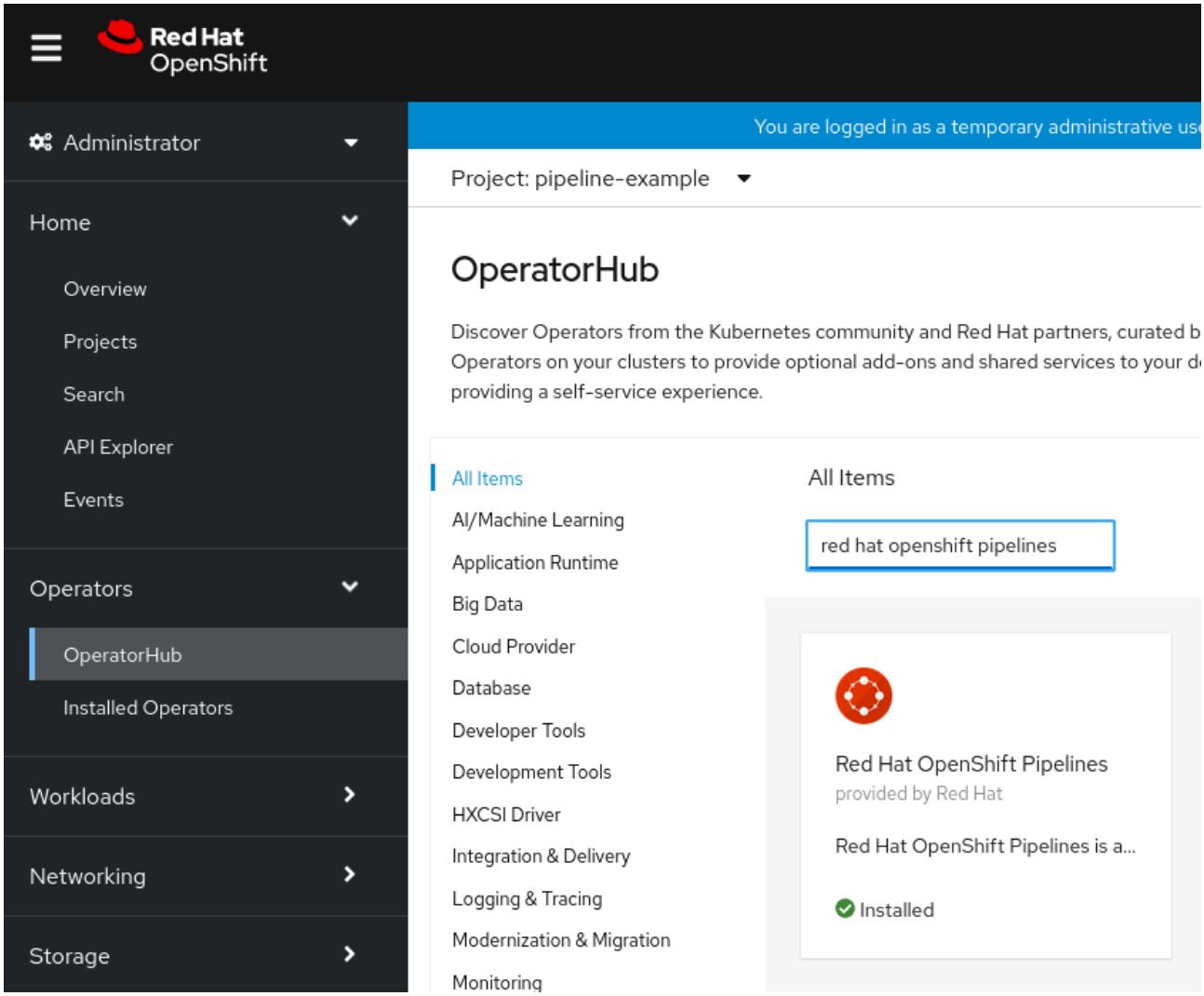


Figura 8.2: Instalação do OpenShift Pipelines pelo OperatorHub

Quando o operador do OpenShift Pipelines for instalado, você verá uma nova seção de **Pipelines** na barra lateral, assim como a opção de adicionar pipelines aos itens do catálogo, como na criação da amostra de Node.js.

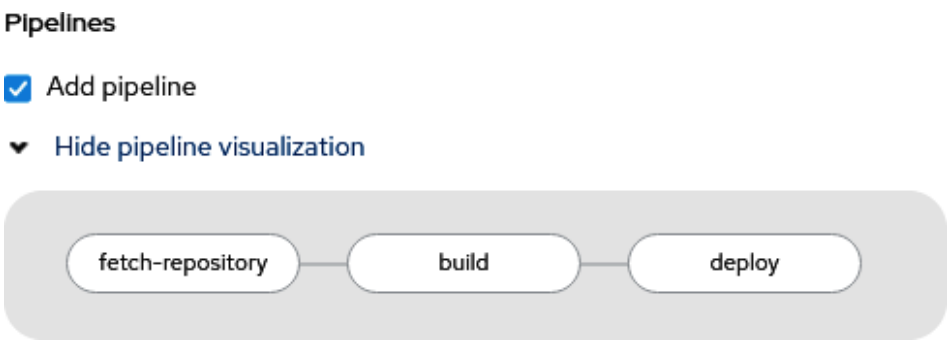


Figura 8.3: Adição de pipelines

Com o OpenShift Pipelines, é possível usar um compilador visual para configurar e criar pipelines complexos e ramificados. Veja uma captura de tela com um exemplo de pipeline mais complexo.

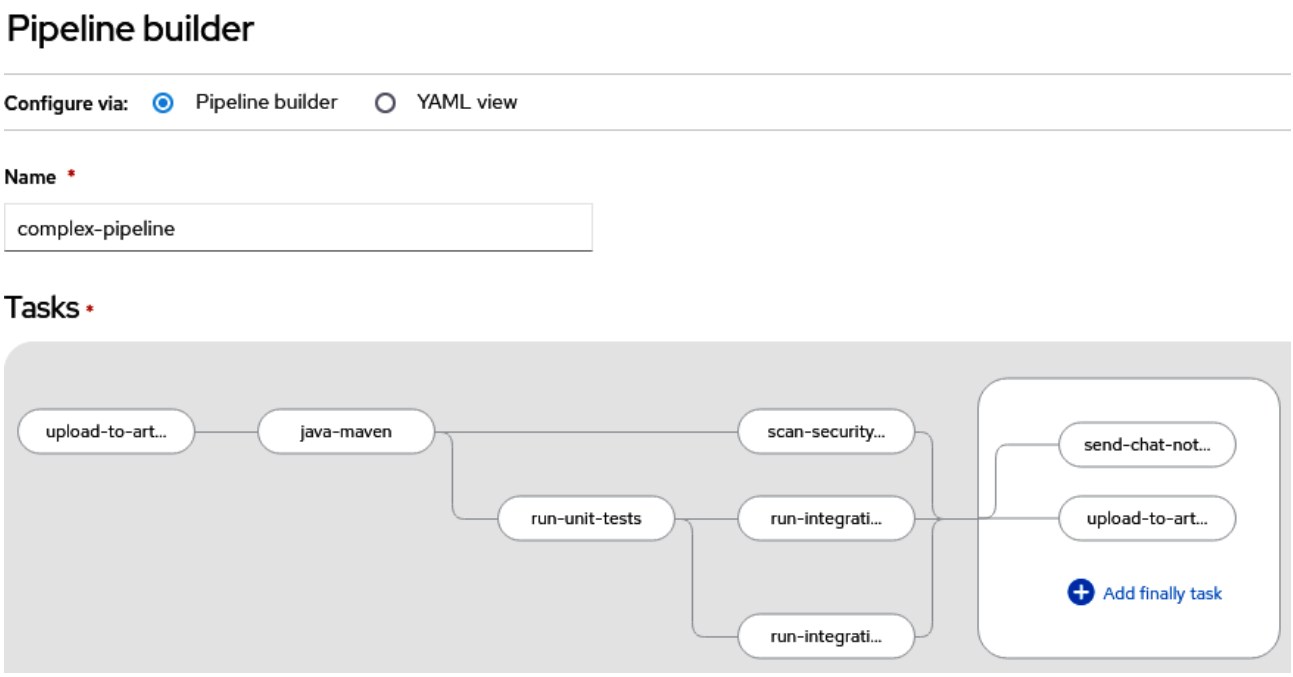


Figura 8.4: Um pipeline complexo

Muitas organizações usam várias ferramentas e tecnologias para criar softwares. No entanto, o OpenShift Pipelines facilita a integração da compilação diretamente no OpenShift, aproveitando todas as vantagens oferecidas pelos containers. Ele oferece uma solução de CI/CD consistente e fácil de usar com o mínimo de instalação necessária, independentemente da infraestrutura subjacente usada.

Leia mais

- Introdução ao OpenShift Pipelines no OpenShift
- Site da comunidade Tekton

Serviços de plataforma: OpenShift Serverless

Um equívoco comum é acreditar que os containers são uma tecnologia útil apenas para serviços de execução prolongada. Na verdade, muitas tarefas e funções serverless de curta duração são executadas com base em containers de curta duração. Devido às vantagens oferecidas pelos containers em termos de velocidade de inicialização, consistência e facilidade de encerramento, eles também funcionam bem com cargas de trabalho serverless. É claro que todos os serviços serverless precisam de servidores subjacentes para executar o código. É por isso que às vezes eles são chamados de "função como serviço".

O Red Hat OpenShift usa o operador do OpenShift Serverless para viabilizar cargas de trabalho serverless, ou função como serviço. Esse operador é baseado no famoso projeto open source chamado Knative.

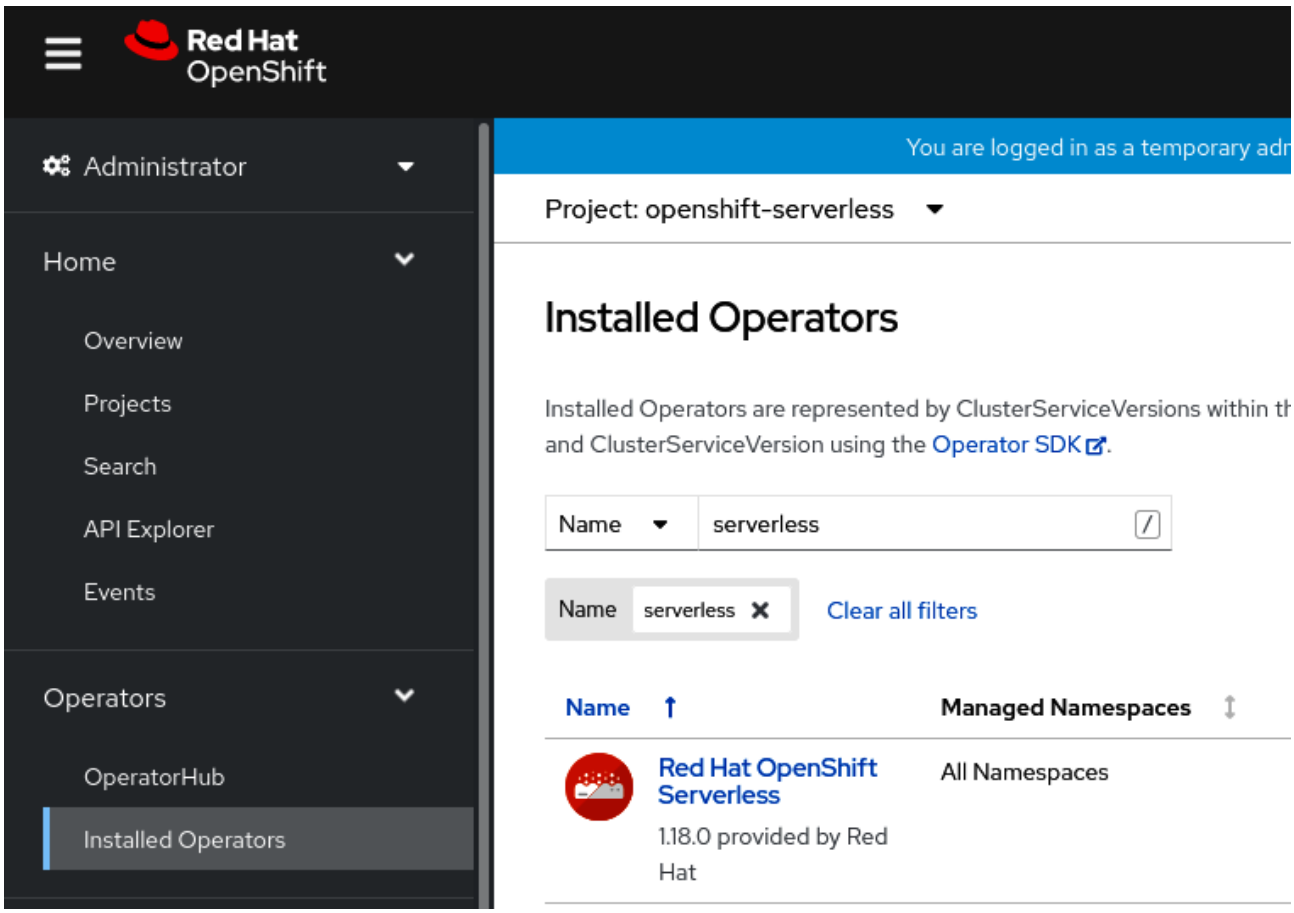


Figura 8.5: Visualização de operadores instalados

Depois de implantar o cluster (o que costuma levar entre um e dois minutos), é preciso realizar uma pequena instalação do operador. Existem duas **definições de recursos personalizados (CRDs)** que precisam de atenção especial: o **Knative Serving** e o **Knative Eventing**.

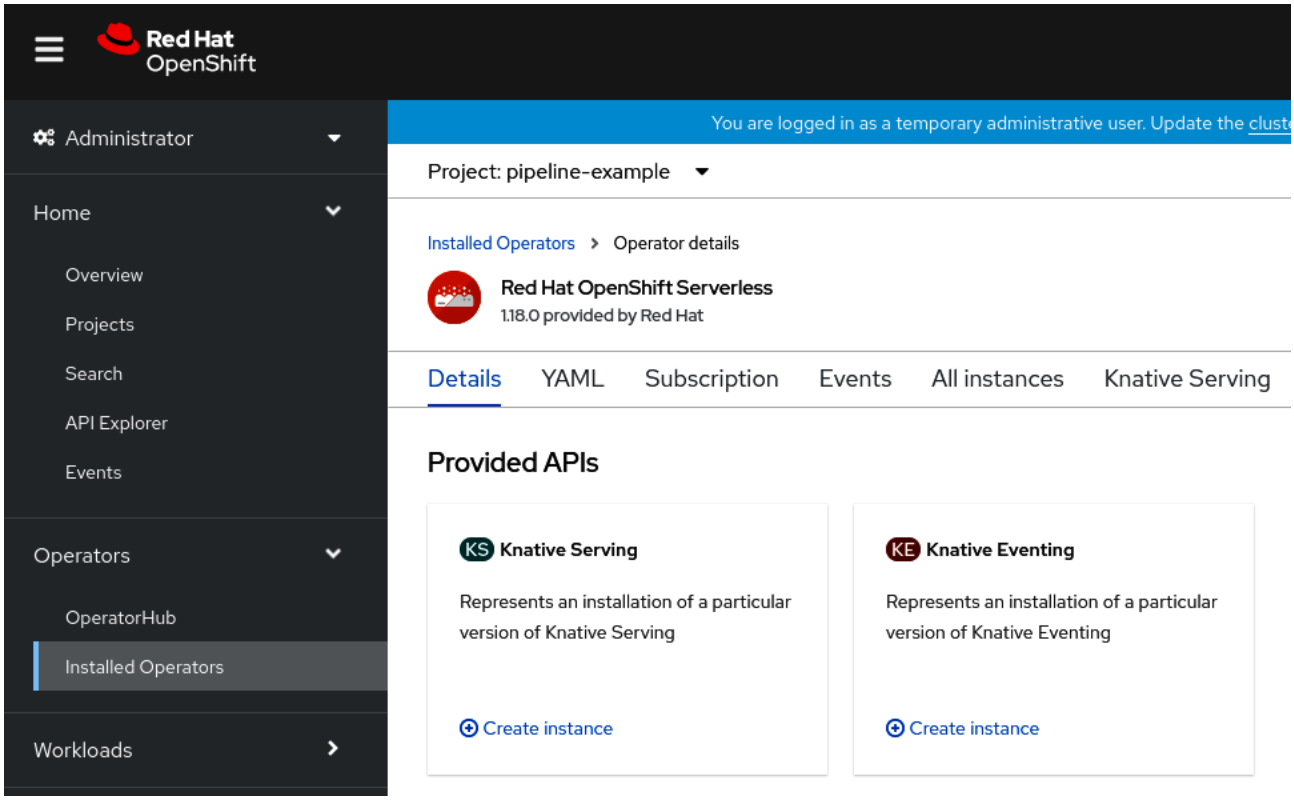


Figura 8.6: Knative Serving e Knative Eventing no menu dos operadores

- **O Knative Serving** simplifica a implantação da aplicação, escala dinamicamente baseado no tráfego de entrada e oferece suporte para estratégias de implantação personalizadas com divisão de tráfego. Por exemplo, uma função que faz upload de uma imagem para um bucket de armazenamento e registra essa ação em um banco de dados NoSQL.
- **O Knative Eventing** permite a "vinculação tardia" de origens de eventos no ambiente de execução de aplicação (em vez de fazer isso na hora da compilação). Por exemplo, uma aplicação que responde ao upload de novas imagens em um bucket de armazenamento não precisa saber da existência dele na hora da compilação ou da implantação do evento. O Knative Eventing facilita a "vinculação" da origem do evento à aplicação no ambiente de execução.

As duas próximas seções contêm exemplos para cada um desses tipos de aplicações serverless no OpenShift.

Serviços de plataforma: OpenShift Serverless – Exemplo do Knative Serving

Vamos demonstrar como o Knative Serving viabiliza a escala automática de uma aplicação, em particular a escala a zero quando não há solicitações. Uma aplicação muito simples que podemos usar é uma web page que exibe fotos ASCII de animais de estimação:

- Repositório php-ascii-pets do GitHub

Adicionar esse repositório do GitHub é muito fácil, o Red Hat OpenShift detecta automaticamente uma imagem compatível do builder de PHP.

O OpenShift identifica como criar esse projeto de maneira automática.

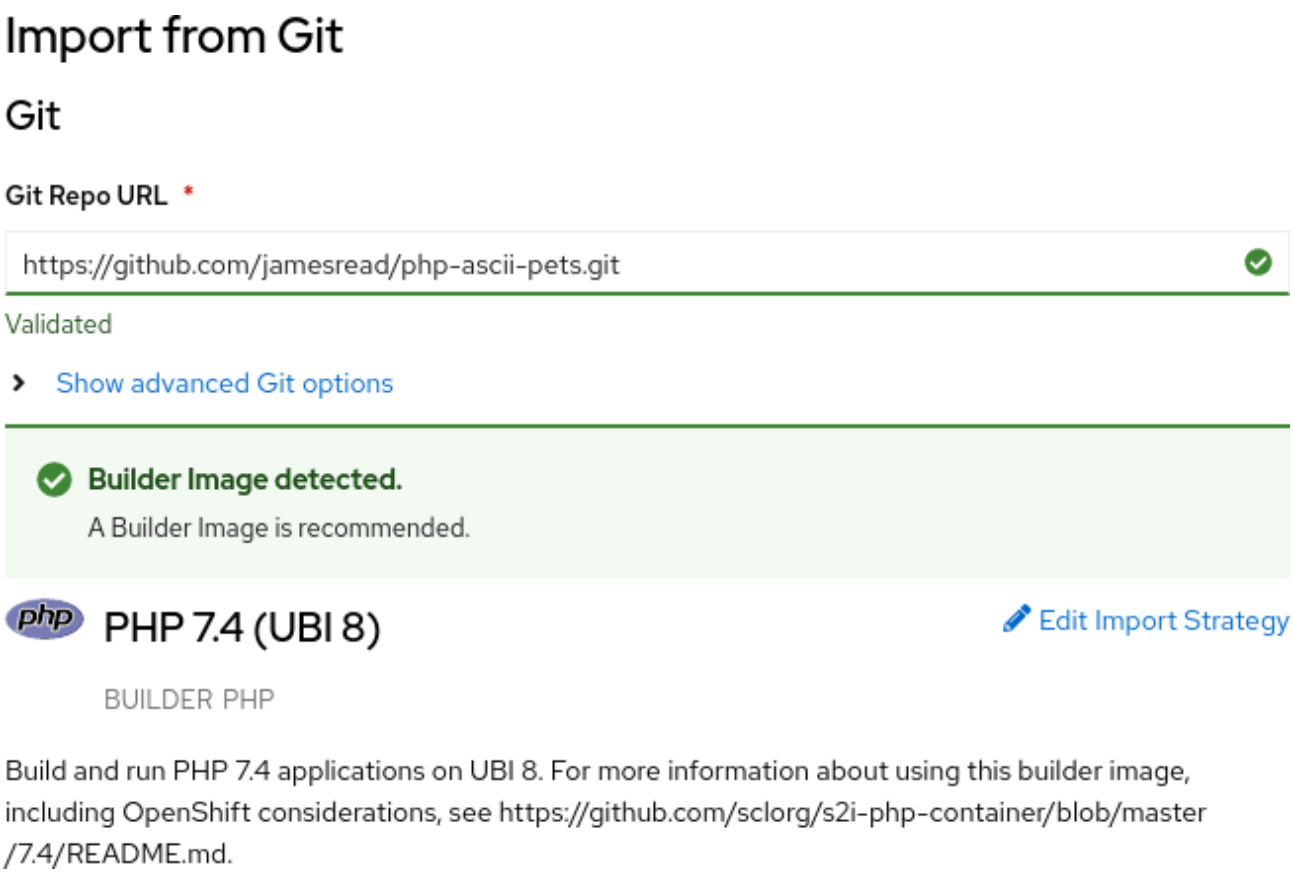


Figura 8.7: Um compilador de imagem é automaticamente detectado e recomendado

É a seleção do tipo de implantação que faz dela ser "serverless". Observe que, ao instalar o OpenShift Serverless, uma implantação "serverless" é disponibilizada.

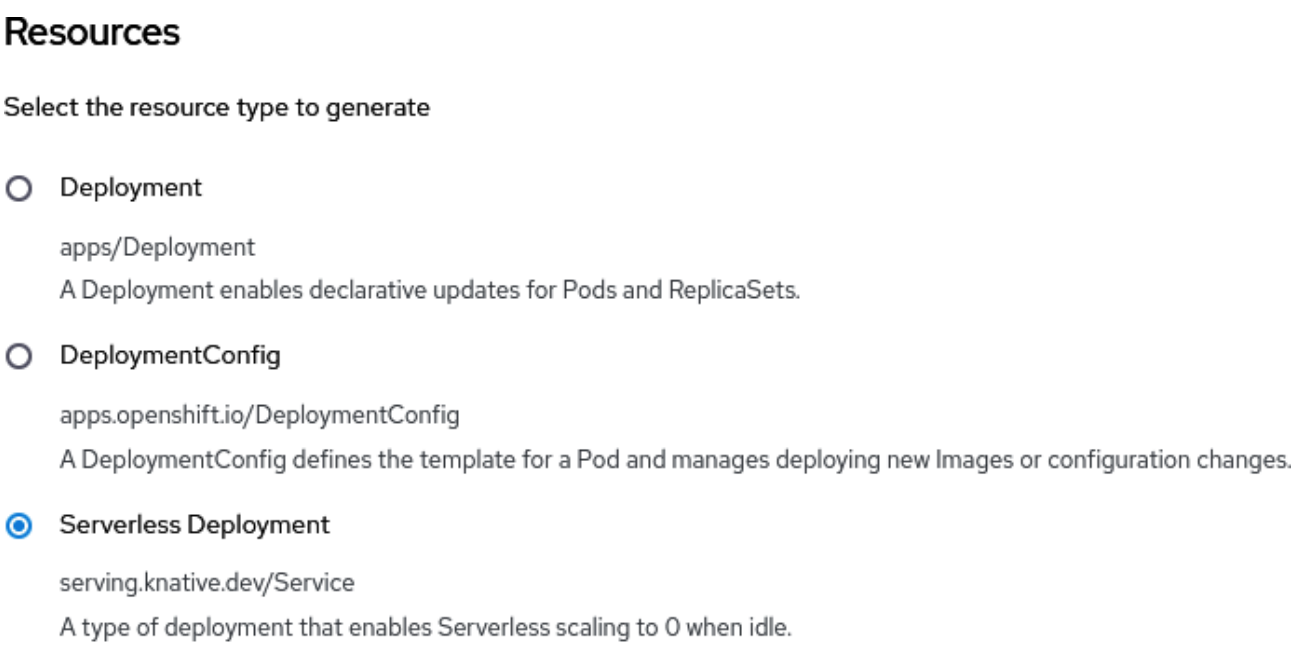


Figura 8.8: Tipo de implantação serverless

A conclusão da primeira compilação leva alguns minutos. Quando ela terminar, haverá um pod implantado. A visualização de topologia ficará assim:

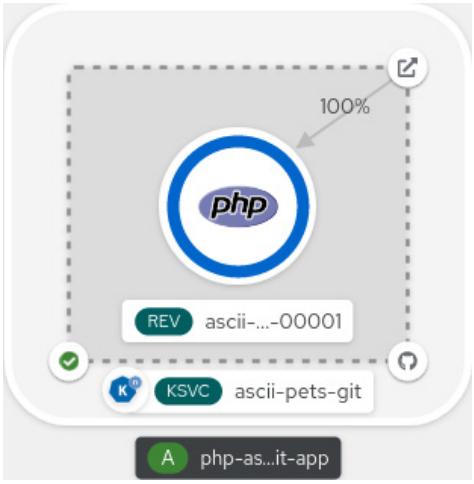


Figura 8.9: Aplicação Knative Serving

A página da aplicação real fica como na *Figura 8.9*. É uma aplicação muito simples, mas o "pet", a arte ASCII, está vinculado ao nome do host do pod. Quando colocamos muitas cargas adicionais na aplicação da nossa demonstração simples, o Knative Serving dispara pods adicionais e você terá uma percepção visual dos diferentes "pets" exibidos.

No entanto, se a aplicação não tiver nenhuma solicitação durante um minuto, o Knative Serving a escalará a zero. Se olharmos a visualização de tecnologia, veremos que a aplicação não está mais em execução.

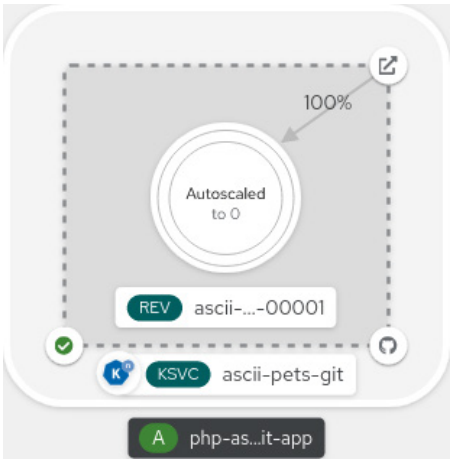


Figura 8.10: Uma implantação serverless, usando o Knative Serving, com uma aplicação escalada a zero

Por fim, se alguém visitar a página, a aplicação será escalada automaticamente de volta para uma, duas, três ou mais réplicas, dependendo de quantas instâncias o OpenShift acredita serem necessárias para atender a demanda.

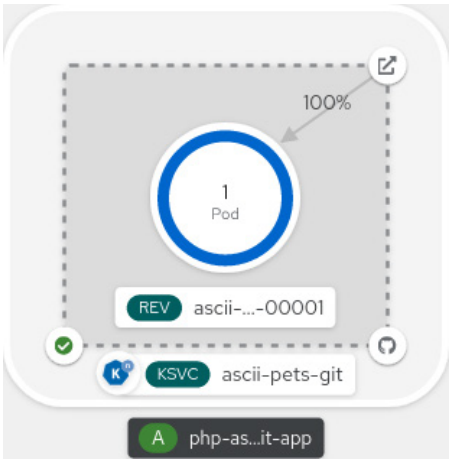


Figura 8.11: Escala automática em resposta ao tráfego

Com o Knative Serving do Red Hat OpenShift Serverless, as organizações podem escalar vertical, horizontal e dinamicamente, com poucas configurações adicionais e sem alterar as aplicações. Isso pode ser muito útil para dimensionar corretamente as implantações e evitar o pagamento e uso desnecessário dos recursos.

Leia mais

- Sobre o OpenShift Serverless
- learn.openshift.com, que inclui um curso sobre o OpenShift Serverless
- Site da comunidade Knative

Serviços de plataforma: OpenShift Serverless – Exemplo do Knative Eventing

Com base no exemplo de aplicação e nos conceitos apresentados no capítulo anterior, o OpenShift Serverless também pode escalar uma aplicação de acordo com as métricas próximas ao tráfego de entrada. Especialmente nos cenários como a arquitetura orientada por eventos, é bom poder escalar verticalmente as instâncias de uma aplicação para processar eventos de uma fila de mensagens ou inicializar com base em um timer.

Usando a mesma implantação, o console do OpenShift facilita a configuração de fontes de eventos diferentes.

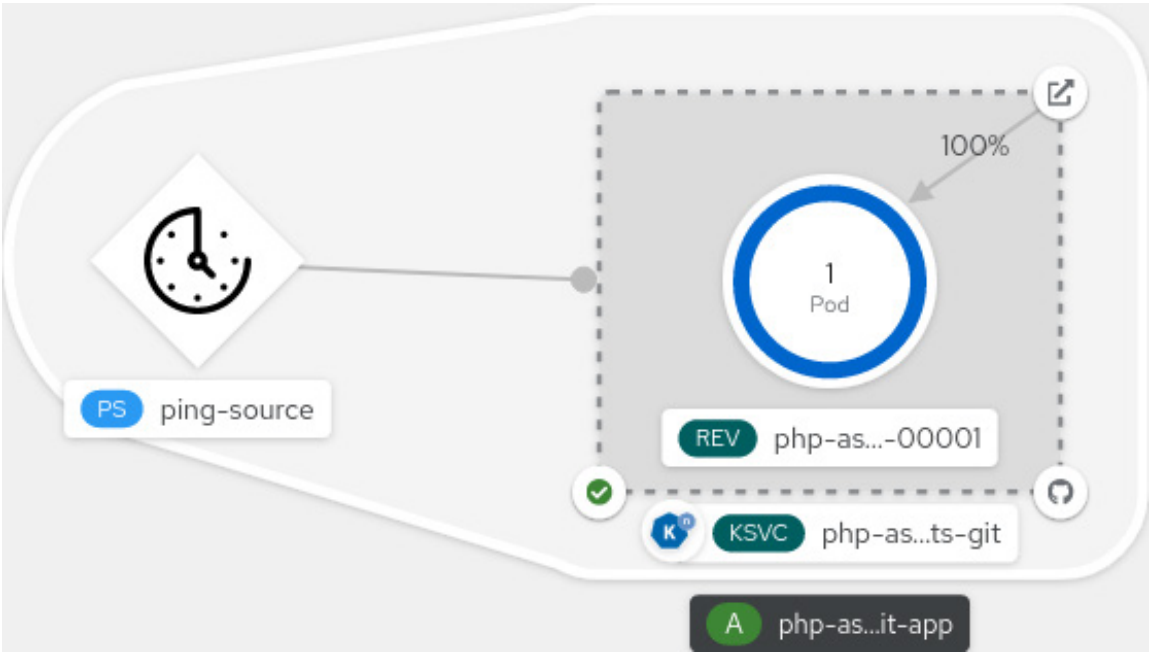


Figura 8.12: Uma fonte de eventos "ping", que envia pings para a aplicação de acordo com o timer

Um caso de uso simples sobre uma fonte de eventos ping seria "inicializar" um container de tarefas de backup todo dia à meia-noite. Outro uso para esse timer com ping pode ser um container de monitoramento com execuções periódicas.

Leia mais

- Explicação do OpenShift Serverless Eventing em cinco minutos
- Sobre o OpenShift Serverless

Serviços de plataforma: OpenShift Service Mesh

Baseado no projeto open source Istio, o Red Hat OpenShift Service Mesh, inclui uma camada transparente nas aplicações distribuídas existentes sem precisar alterar o código delas. Para adicionar o suporte do Red Hat OpenShift Service Mesh aos serviços, implante um proxy sidecar especial em todo o ambiente, interceptando todas as comunicações de rede entre microsserviços. Para configurar e gerenciar a service mesh, use as funcionalidades do plano de controle.

Estes são alguns casos de uso viabilizados pelo OpenShift Service Mesh:

- Criptografia transparente para comunicação entre serviços, com mTLS automático.
- Oferta de várias versões de um serviço e ativação de testes A/B, por exemplo.
- Observabilidade sobre a comunicação dos microsserviços, usando o Kiali.
- Rastreamento de chamadas entre serviços, usando o Jaeger.

Assim como todas as outras funcionalidades de plataforma do OpenShift, o Service Mesh é instalado com um operador da Red Hat.

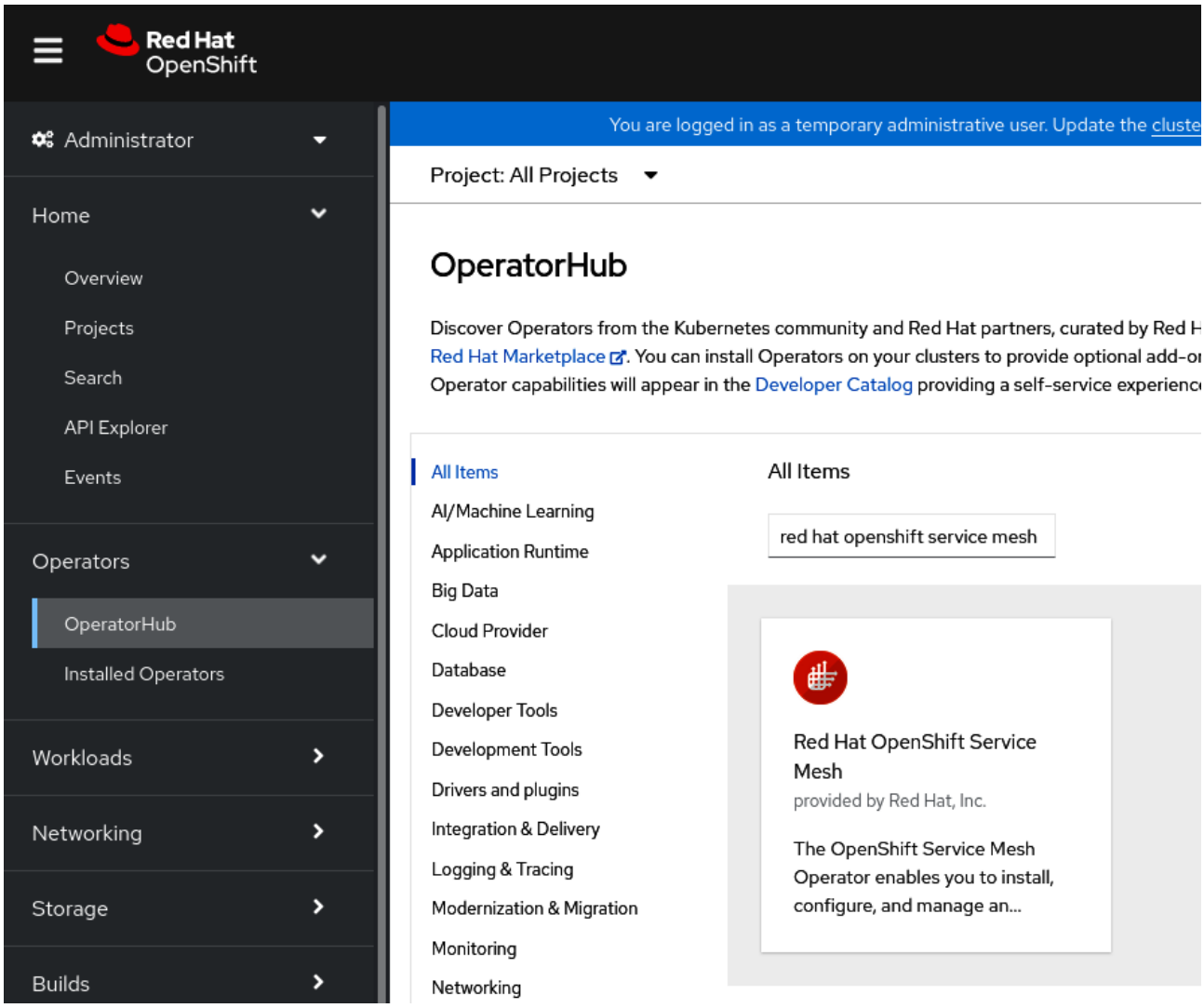


Figura 8.13: Instalação do Service Mesh pelo OperatorHub

A documentação do OpenShift Service Mesh inclui uma aplicação de exemplo excelente, chamada demonstração BookInfo. É uma aplicação simples que imita uma livraria, executada no OpenShift.

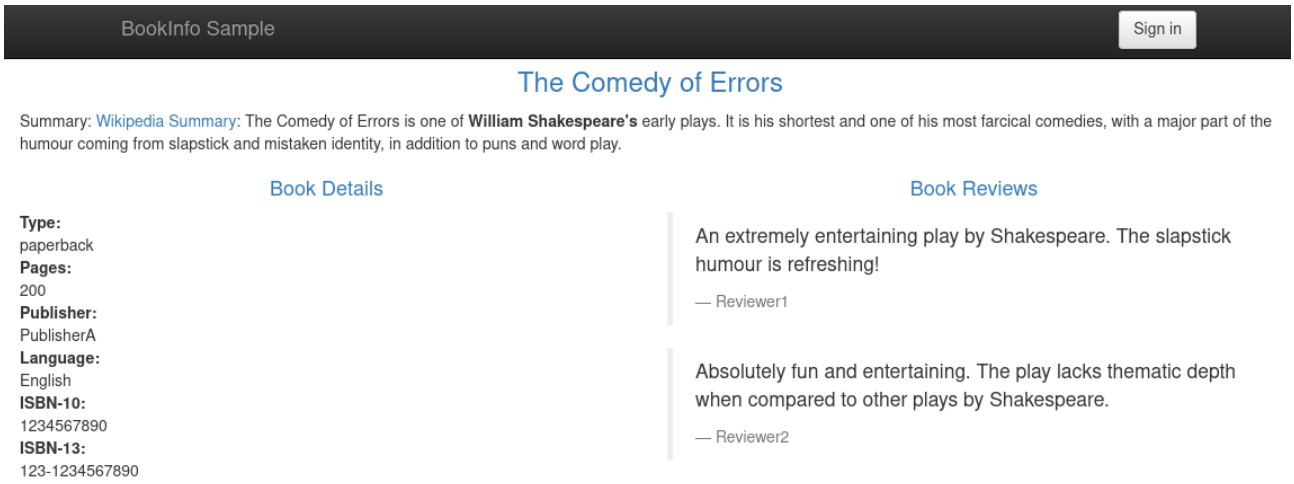


Figura 8.14: A aplicação BookInfo

Para executar a aplicação BookInfo com o Service Mesh, é preciso criar um plano de controle para ele. Isso é fácil de fazer com a interface gráfica do OpenShift, como pode ser visto na *Figura 8.15*:

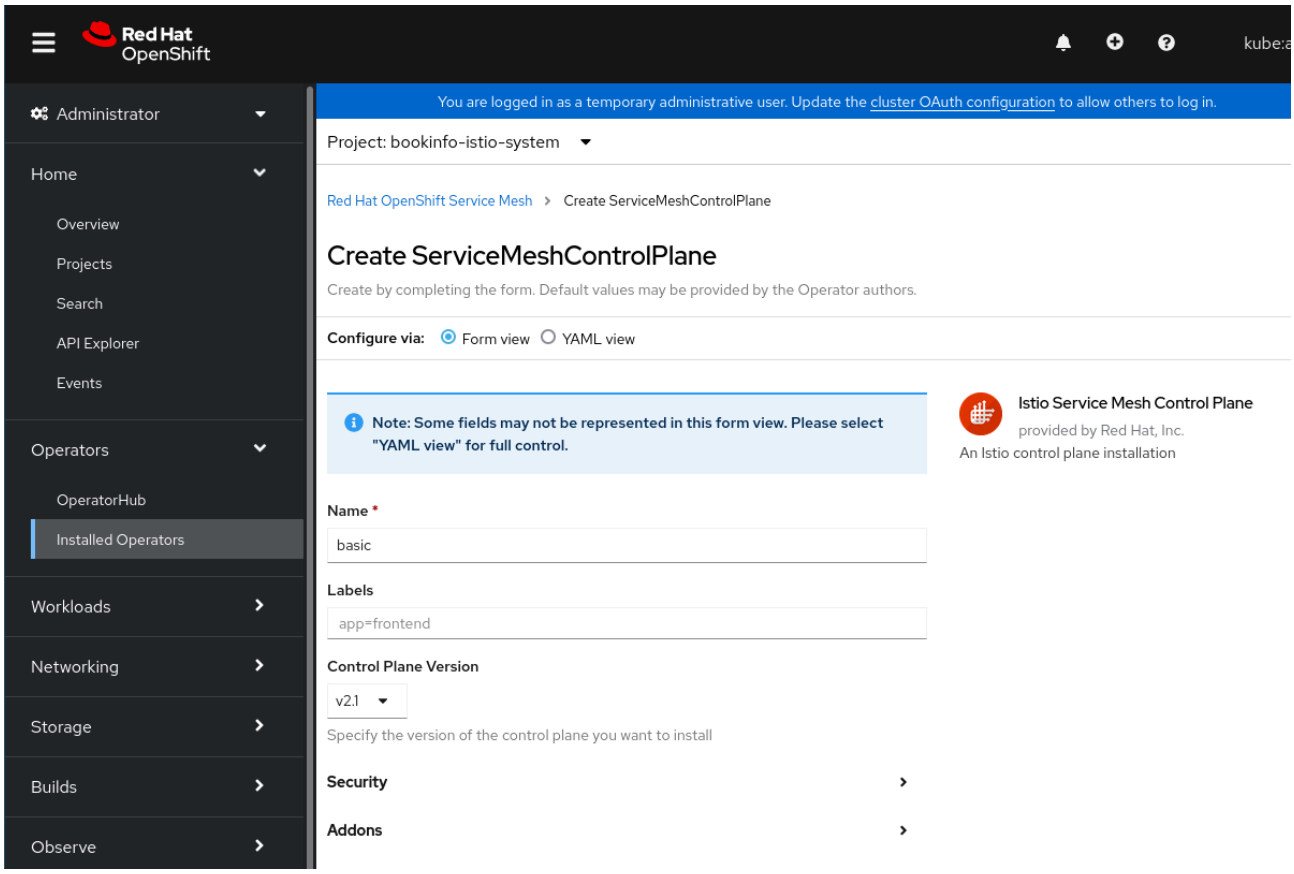


Figura 8.15: Configuração do plano de controle no projeto bookinfo-istio-system do BookInfo

O projeto BookInfo aceita tráfego de entrada pelo plano de controle e, neste caso, um outro projeto foi criado para armazená-lo, chamado `bookinfo-istio-system`.

Não é preciso separar o plano de controle do Service Mesh e o seu projeto, e talvez seja até mesmo possível compartilhá-lo entre vários projetos.

Nas próximas duas seções, vamos detalhar dois casos de uso do Service Mesh: observabilidade e rastreamento distribuído.

Serviços de plataforma: OpenShift Service Mesh – observabilidade usando o Kiali

Ao utilizar a visualização de topologia do Red Hat OpenShift para observar os pods subjacentes que formam essa aplicação, é possível ver que ela é composta por seis microsserviços.

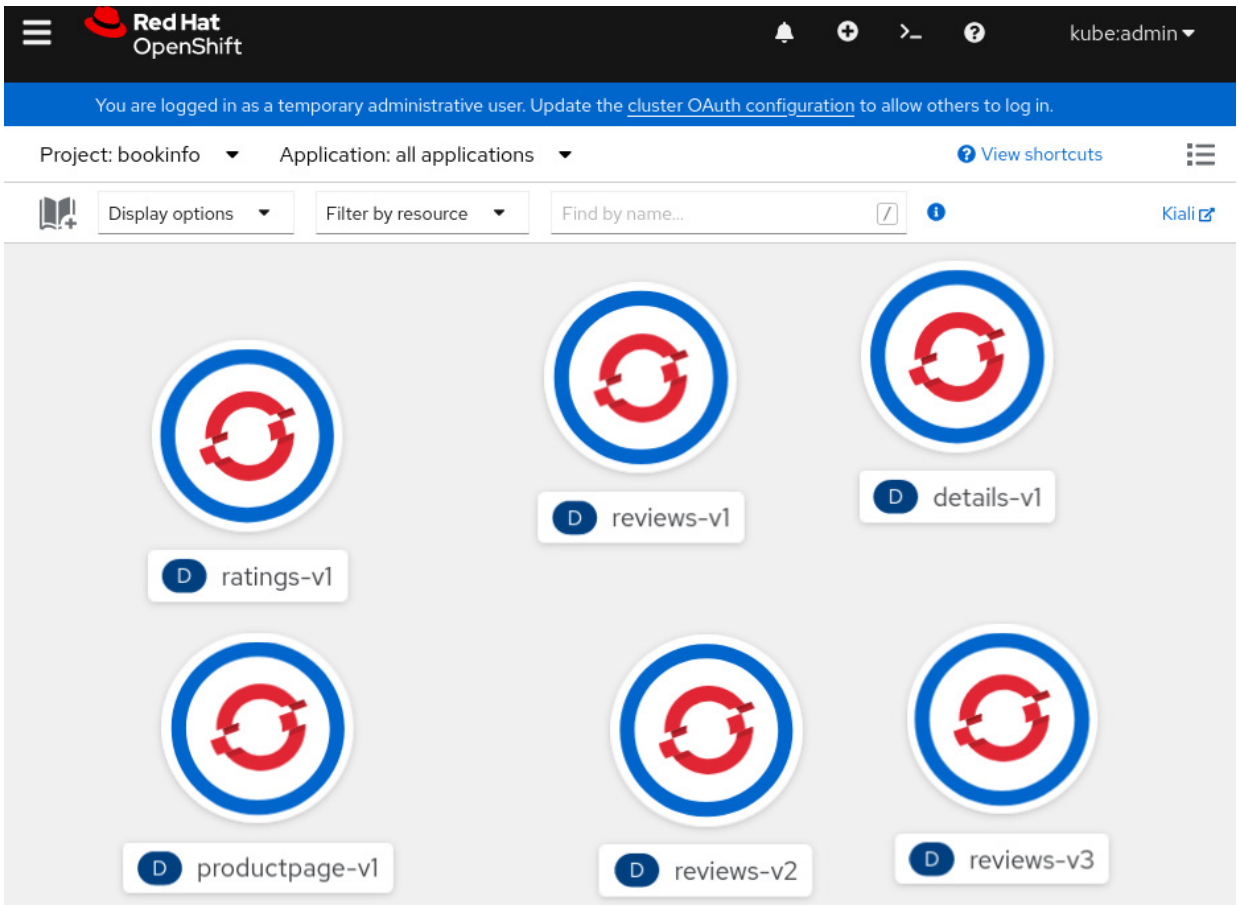


Figura 8.16: Os seis microsserviços que compõem a aplicação BookInfo

Essa visualização é útil, mas não mostra como esses serviços estão conectados. Veja o primeiro benefício do Service Mesh: observabilidade. Se abrirmos um console um pouco diferente, chamado Kiali, que vem com o Service Mesh, veremos que há uma representação arquitetônica muito mais refinada desses seis serviços.

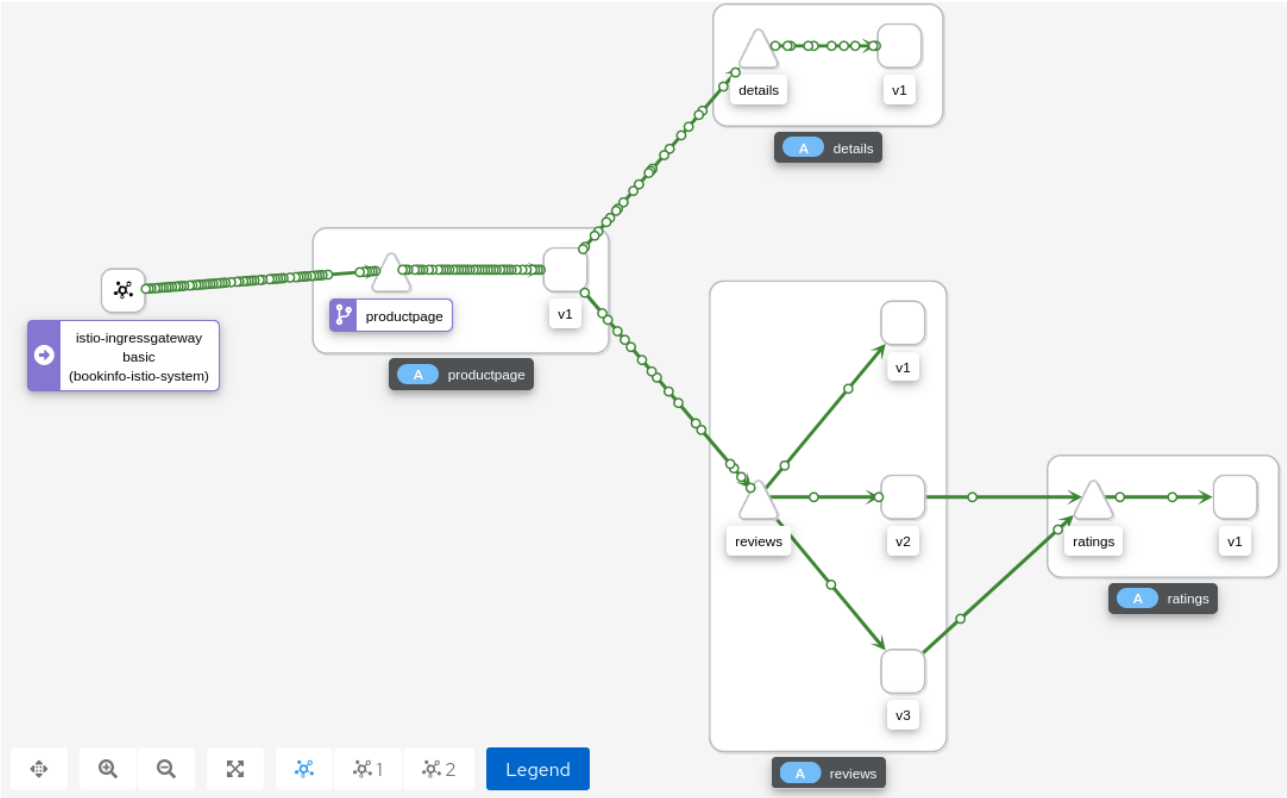


Figura 8.17: O gráfico da arquitetura da aplicação gerado automaticamente pelo Service Mesh

Além de mostrar a conectividade real entre os serviços, essa visualização também exibe um diagrama do tráfego em tempo real. Neste caso, a aplicação está recebendo um tráfego significativo, com cada solicitação representada por uma animação circular da conexão.

Para ter uma observabilidade maior, o administrador pode clicar em uma das conexões de serviços para ver o perfil de tráfego. Neste caso, vemos que a maior parte do status do tráfego é **HTTP OK**.

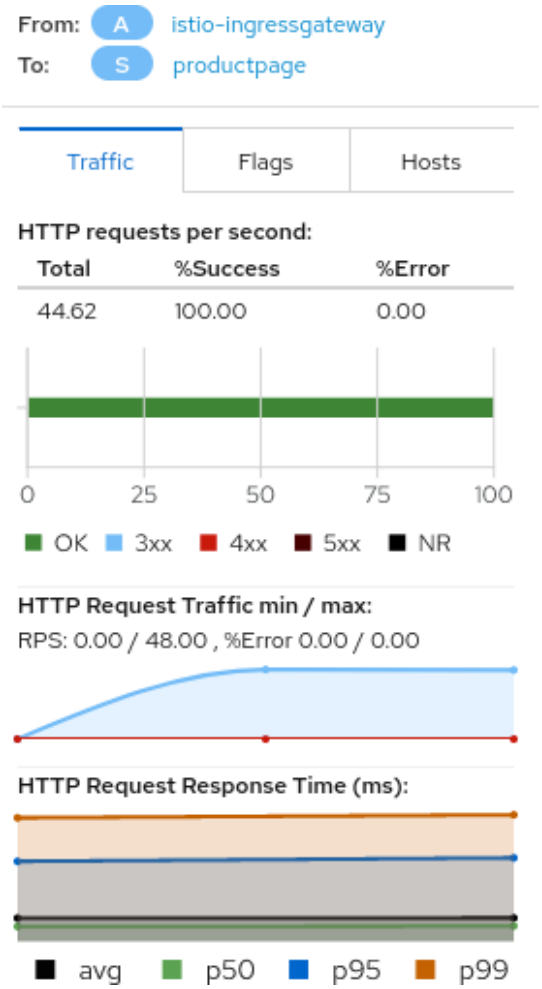


Figura 8.18: Vários status HTTP

Para inserir um erro artificial na aplicação, é só desativar o microsserviço de `detalhes`, escalando a zero réplicas:

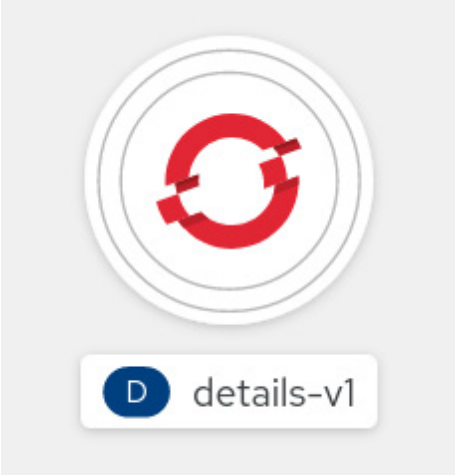


Figura 8.19: Zero réplicas do serviço de detalhes, para simular uma falha

Agora se voltarmos para a visualização do Kiali, você perceberá que alguns componentes foram adicionados ao diagrama. A conexão ao serviço de `detalhes` está destacada em vermelho para mostrar o erro.

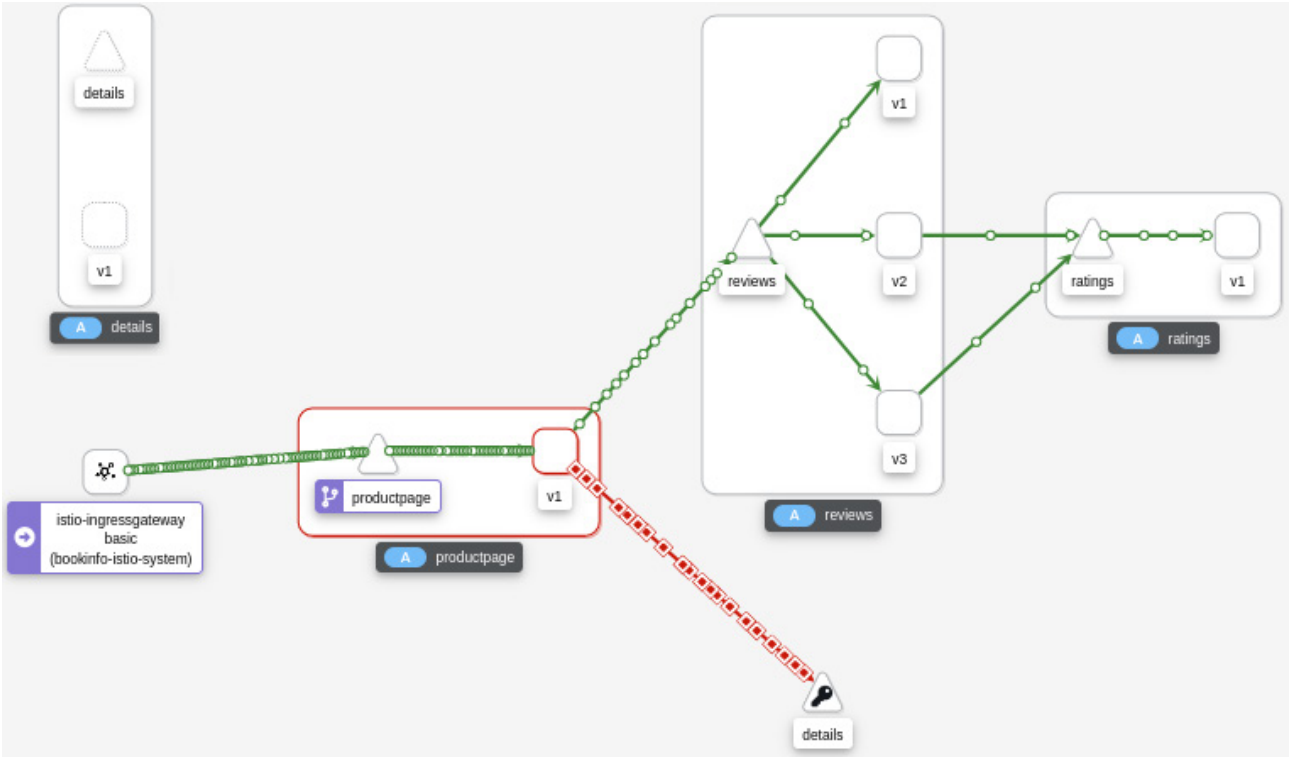


Figura 8.20: A conexão em vermelho sugere um problema com o serviço

Vimos que o Kiali do Service Mesh melhora muito a observabilidade. Em aplicações menores, como o BookInfo, o Service Mesh é útil. No entanto, quando elas vão ficando maiores e mais complexas, chegando a envolver 20 microsserviços ou mais, além de vários tipos de interação, ferramentas como o Service Mesh e o Kiali são ainda mais valiosas, quase essenciais.

Serviços de plataforma: Red Hat OpenShift Service Mesh – rastreamento distribuído com o Jaeger

Outro serviço extremamente valioso oferecido pelo Service Mesh é o Jaeger. Com ele, é possível fazer o rastreamento distribuído das aplicações de microsserviços complexas. Na seção anterior, analisamos a aplicação BookInfo e vimos que a desativação do serviço de `detalhes` resulta na falha das solicitações.

Nesse caso, é simples identificar a causa das falhas: o serviço de `detalhes` estava indisponível, e nós que causamos isso. No entanto, se a causa fosse desconhecida e exigisse uma investigação maior, o rastreamento distribuído do Jaeger seria útil.

O Jaeger rastreia as solicitações do primeiro serviço, por meio de conexões subsequentes do sistema. O Jaeger precisa de alguns códigos adicionais para ser ativado, mas os desenvolvedores conseguem fazer isso facilmente com as bibliotecas de cliente e alterações mínimas na programação.

Ao analisar o serviço `productpage` (escrito em Python), conseguimos identificar o código que possibilita o rastreamento distribuído daquele serviço pelo Jaeger. Esse código importa a biblioteca de cliente do Jaeger para o serviço `productpage`:

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

Quando a importação for concluída, a página da solução precisará configurar um novo rastreador. O código inicializa um Rastreador do Jaeger no serviço productpage service:

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

Por fim, ainda no serviço productpage, dizemos ao Jaeger que queremos criar um novo span, uma solicitação nova para vários serviços:

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

Em seguida, o Jaeger usará esse rastreador em vários serviços. Eles também precisam ser adaptados para funcionar com o Jaeger, mas as bibliotecas de cliente estão disponíveis nas linguagens de programação mais usadas.

O código-fonte completo do serviço productpage está disponível no GitHub.

Existem duas opções para instrumentalizar o código: usar as bibliotecas de cliente do Jaeger, agora consideradas obsoletas, ou usar as opções padrão open source do projeto OpenTelemetry, o que é preferível:

- Bibliotecas do OpenTelemetry

Depois que a instrumentalização for concluída, você verá rastreamentos como este:

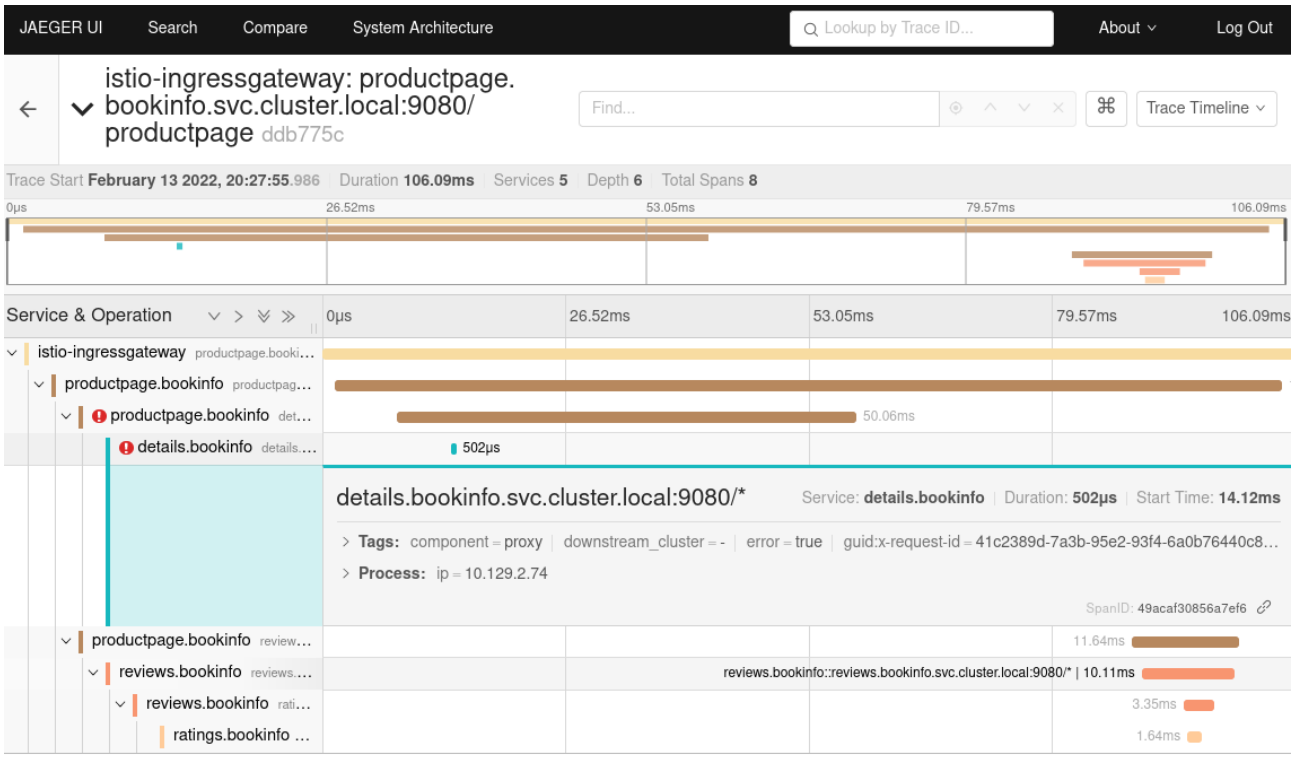


Figura 8.21: Um rastreamento de chamada

Essa imagem mostra um rastreamento de chamada parecido com o do Kiali, só que em uma visualização hierárquica. Todas as linhas podem ser expandidas para mostrar detalhes da solicitação, como duração, hora de início, endereço IP de origem e muito mais. Mais uma vez, esse nível de informações se torna quase essencial para aplicações de microsserviços complexas.

Voltando ao erro que estávamos tentando diagnosticar, a visualização do rastreamento mostra claramente que o microsserviço de `detalhes` está apresentando problemas. Neste caso, o erro é simples, mas a expansão dos detalhes mostra que o serviço está emitindo códigos de erro `HTTP 503`.

Figura 8.22: Detalhes do erro

HTTP 503 é o código de erro padrão para "serviço indisponível". Neste caso, o motivo é apenas que o serviço está escalado para zero réplicas. A escala do serviço faz com que ele seja restaurado.

O Jaeger e o Kiali juntos são ferramentas potentes para aplicações de microsserviços cada vez mais complexas. Eles são oferecidos como parte do serviço do Azure Red Hat OpenShift e não exigem subscrição adicional.

Resumo

Neste capítulo, abordamos alguns dos principais serviços de valor agregado oferecidos pela plataforma de aplicações (Red Hat OpenShift) em comparação com apenas um ambiente do Kubernetes "vazio". É possível replicar um nível parecido de funcionalidade do OpenShift instalando todos os respectivos projetos open source upstream na plataforma. No entanto, o Azure Red Hat OpenShift tem complexidades como integração, ciclo de vida e suporte.

Esses vários serviços de plataforma, aplicação, dados, desenvolvedor e cluster aumentam a produtividade dos seus desenvolvedores e operadores no Kubernetes e na implantação de várias aplicações empresariais complexas.

Capítulo 9

Integração com outros serviços

É muito comum que uma aplicação não possa existir inteiramente no Red Hat OpenShift. De um jeito ou de outro, a maioria delas depende de bancos de dados externos ou de serviços do Azure.

O Azure Red Hat OpenShift não "quebra" nenhum tipo de conectividade aqui. Por exemplo, se uma aplicação depende do Azure CosmosDB, ela ainda é capaz de se conectar do Azure Red Hat OpenShift ao Azure CosmosDB sem precisar de alterações. Dependendo da aplicação e da empresa, você pode implantar algumas dessas dependências externas ao mesmo tempo ou separadamente do Azure Red Hat OpenShift.

Se você acha que se beneficiará ao implantar essas dependências externas na sua aplicação, o Azure Service Operator pode simplificar bastante esse processo. Em vez de usar o Azure CLI, o Azure Cloud Shell ou os templates de ARM, você pode usar o Azure Service Operator para implantar esses recursos como se fossem nativos do Kubernetes.

Azure Service Operator

O Azure Service Operator é outro operador que facilita sua vida como desenvolvedor ou administrador implantando aplicações no Azure Red Hat OpenShift. Uma vantagem é que, com ele, não é preciso sair do console do OpenShift para provisionar serviços do Azure. Isso facilita e agiliza a implantação de aplicações que tenham dependências de serviços do Azure, como o Azure CosmosDB.

Outra vantagem talvez ainda maior do Azure Service Operator é armazenar as dependências dos serviços do Azure junto com a definição da aplicação do OpenShift, em uma abordagem de código como infraestrutura, usando os arquivos YAML padrão do Kubernetes.

O modo de funcionamento é simples: o Azure Service Operator procura **CRDs** novos, definidos em YAML, que sejam compatíveis com a definição de recurso do Azure. Em seguida, o Azure Service Operator traduz esse YAML nas chamadas de API do Azure necessárias para criar o que está sendo pedido. Depois da instalação do operador, os usuários podem navegar pelo catálogo de desenvolvedores do OpenShift e ver a tela de criação de vários recursos comuns do Azure, como os endereços IP públicos, os bancos de dados SQL e o Azure Firewall.

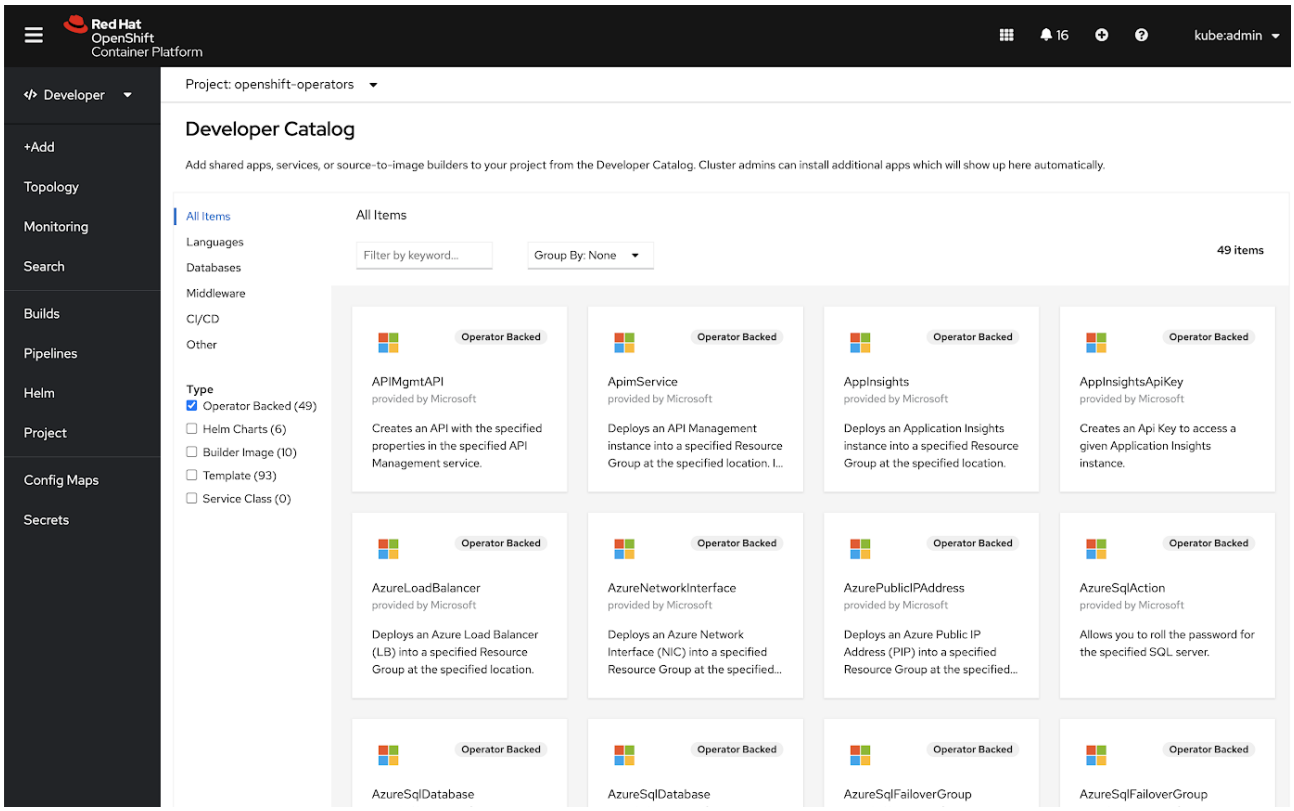


Figura 9.1: Alguns dos serviços do Azure expostos pelo Azure Service Operator

O Azure Service Operator pode ser instalado pelo OperatorHub e disponibilizado para todos os usuários do OpenShift.

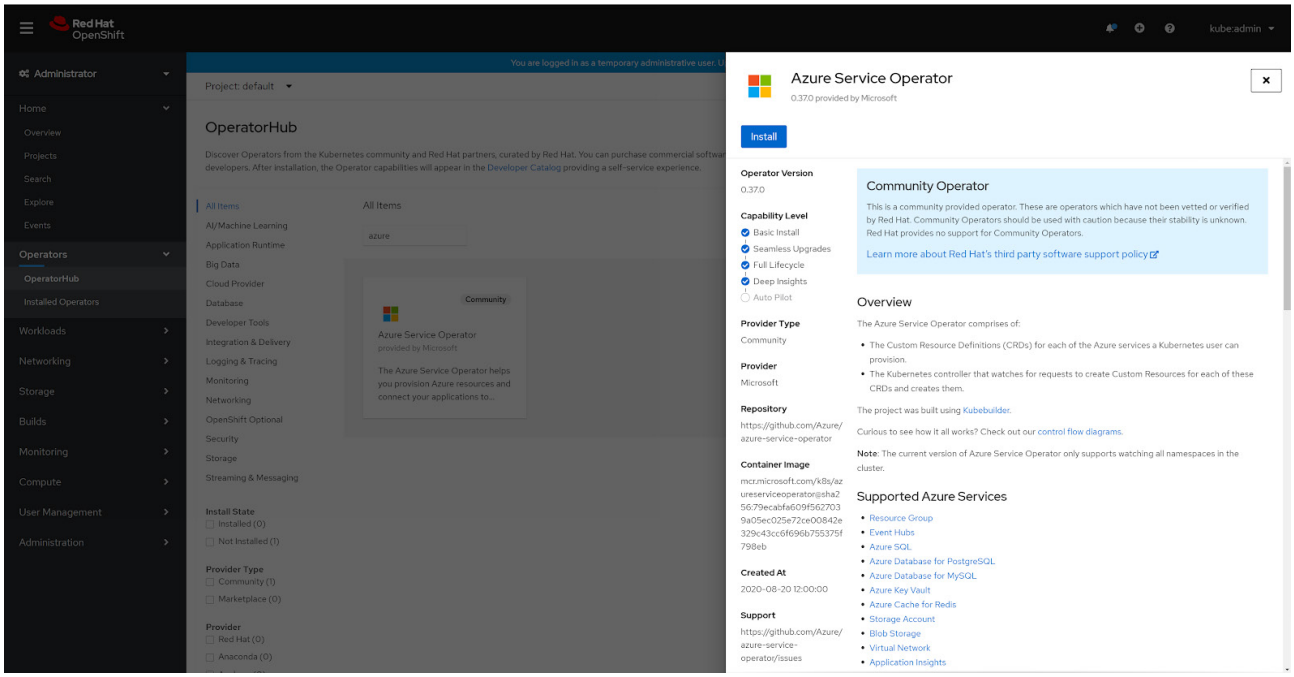


Figura 9.2: A tela de instalação do Azure Service Operator, com a galeria do OperatorHub no fundo

O Azure Service Operator não é obrigatório para os serviços em execução no Azure, e é importante entender que os serviços subjacentes do Azure são implantados de maneira nativa no Azure, e não no OpenShift. No entanto, o Azure Service Operator facilita e agiliza esse processo para as aplicações que têm dependências dos serviços do Azure.

Um caso de uso conhecido do Azure Service Operator é a implantação de bancos de dados dependentes junto com a aplicação. Por exemplo, para uma aplicação web que usa uma instância do Azure CosmosDB, implante o Azure CosmosDB com o Azure Service Operator como parte da implantação da aplicação no OpenShift. O Azure Service Operator tem suporte para o Azure SQL Database, o Azure Database for MySQL e o Azure PostgreSQL, entre alguns outros.

Supondo que o Azure Service Operator esteja instalado, este manifesto do Kubernetes poderia ser armazenado com a aplicação do OpenShift e usado para provisionar um banco de dados do MySQL:

Fonte: retirado do repositório de amostras do Azure Service Operator

```
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
```

Não seria comum usar o Azure Service Operator para serviços que envolvem vários serviços do Azure com dependências complexas. Nesses casos, ferramentas como o Bicep ou os templates de ARM do Azure podem ser mais adequados.

Para uma introdução mais detalhada do Azure Service, veja estes posts:

- Post: setembro de 2020
- Azure Service Operator no OperatorHub.io
- Azure Service Operator no GitHub

Integração com o Azure DevOps

Junto ao OpenShift Pipelines, muitos usuários querem integrar o Azure Red Hat OpenShift com o Azure DevOps. Essas soluções podem coexistir perfeitamente, com alguns projetos usando uma e outros usando a outra. Às vezes eles até podem usar as duas. Decidir quando usar uma das soluções depende de alguns fatores.

Em geral, o Azure DevOps oferece uma integração de alto nível com outras ferramentas do Azure, mas pode ser facilmente implantada no OpenShift, assim como outros recursos de computação.

Por outro lado, o OpenShift Pipelines é uma oferta bem integrada que vem com o OpenShift e oferece uma experiência de multicloud consistente.

O Azure DevOps é compatível com o OpenShift como qualquer outro cluster do Kubernetes. Assim, todas as interfaces e APIs padrão do Kubernetes funcionam como esperado.

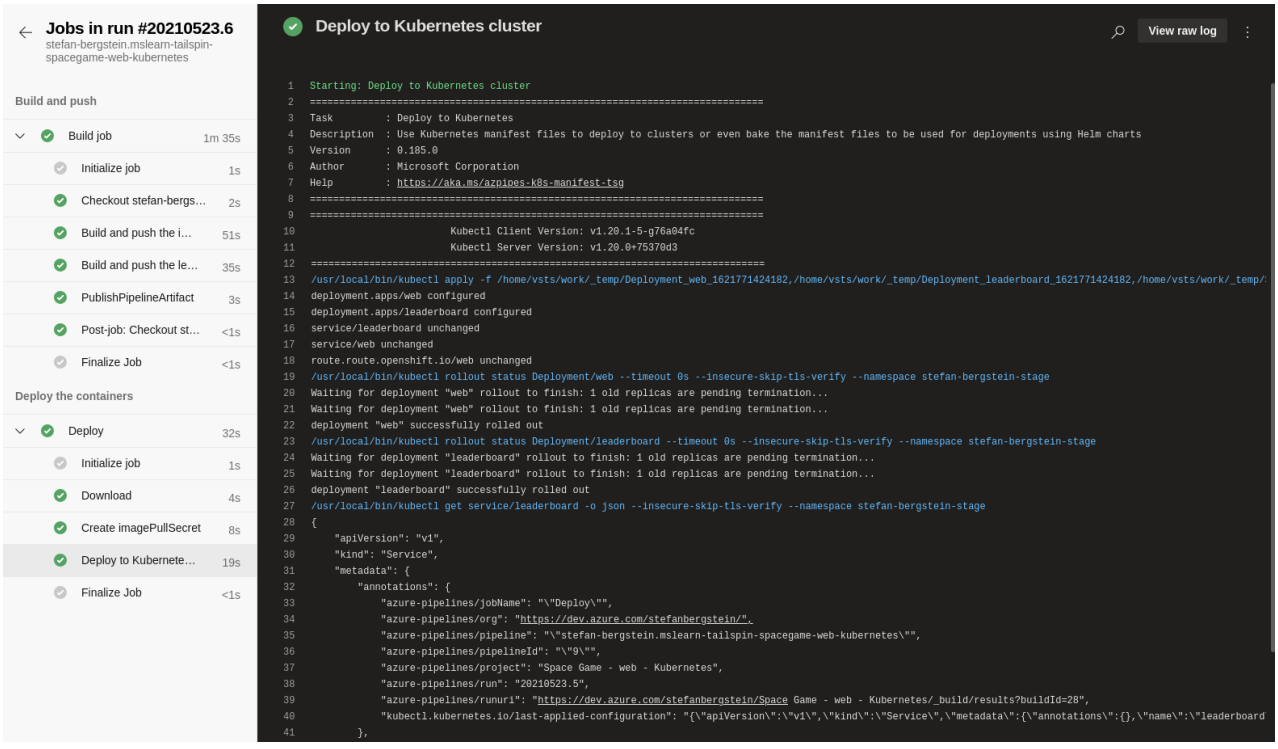


Figura 9.3: Um pipeline do Azure DevOps enviando conteúdo para o OpenShift

Leia mais

Este post do blog da comunidade oferece um excelente tutorial de introdução ao Azure Red Hat OpenShift e ao Azure DevOps:

- Post: maio de 2021

Integração com o Visual Studio Code

Como parte do valor da integração do desenvolvedor do OpenShift, existem alguns plugins para os IDEs mais conhecidos, incluindo o Visual Studio Code. Isso facilita e agiliza o acesso dos desenvolvedores e administradores aos recursos do Kubernetes e do OpenShift sem que eles precisem sair do IDE.

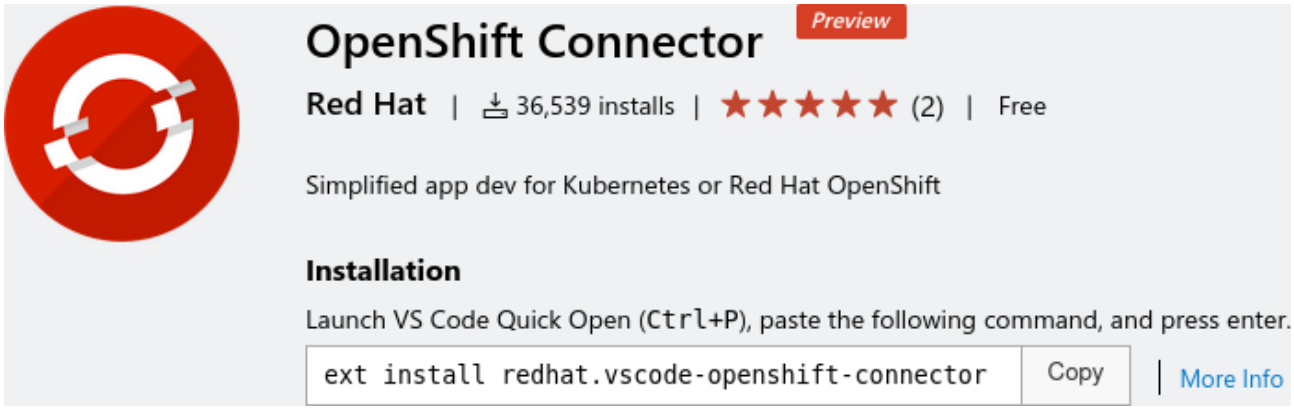


Figura 9.4: Instalação do OpenShift Connector

Após o download e a instalação do conector no Visual Studio Code, você precisará fazer login no cluster do OpenShift. Este projeto é simples e parecido com o "hello world", conectado ao Azure Red Hat OpenShift:

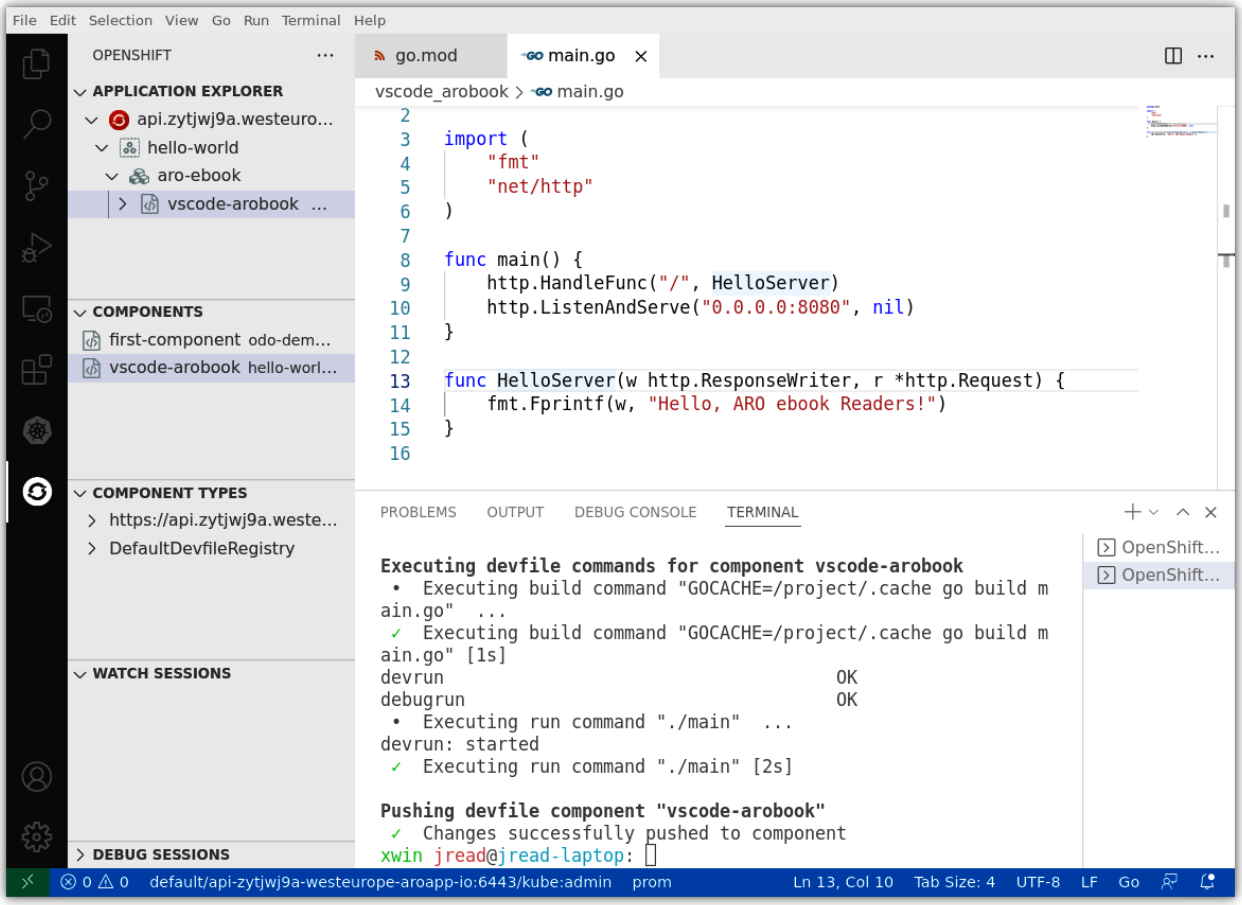


Figura 9.5: Um projeto Golang que acabou de ser implantado com o conector do Visual Studio Code OpenShift

Uma vantagem de usar o conector do OpenShift com o Visual Studio Code é a disponibilidade de um front-end gráfico para o "odo", como é conhecida a famosa ferramenta do OpenShift, a "OpenShift Do". Isso permite que os desenvolvedores pensem em conceitos mais gerais, e não apenas nos componentes do Kubernetes. Os desenvolvedores não precisam se preocupar com muitos detalhes sobre implantações, ReplicaSets etc. Na captura de tela anterior, podemos ver que esse projeto é simples, com um serviço HTTP exposto.

Além disso, o conector tem a funcionalidade de enviar alterações no código diretamente para o OpenShift, sem precisar de controle do código-fonte antes. Isso é muito útil para o desenvolvimento rápido, em vez de sempre ter que enviar o controle do código-fonte, criar um novo container e fazer a implantação.

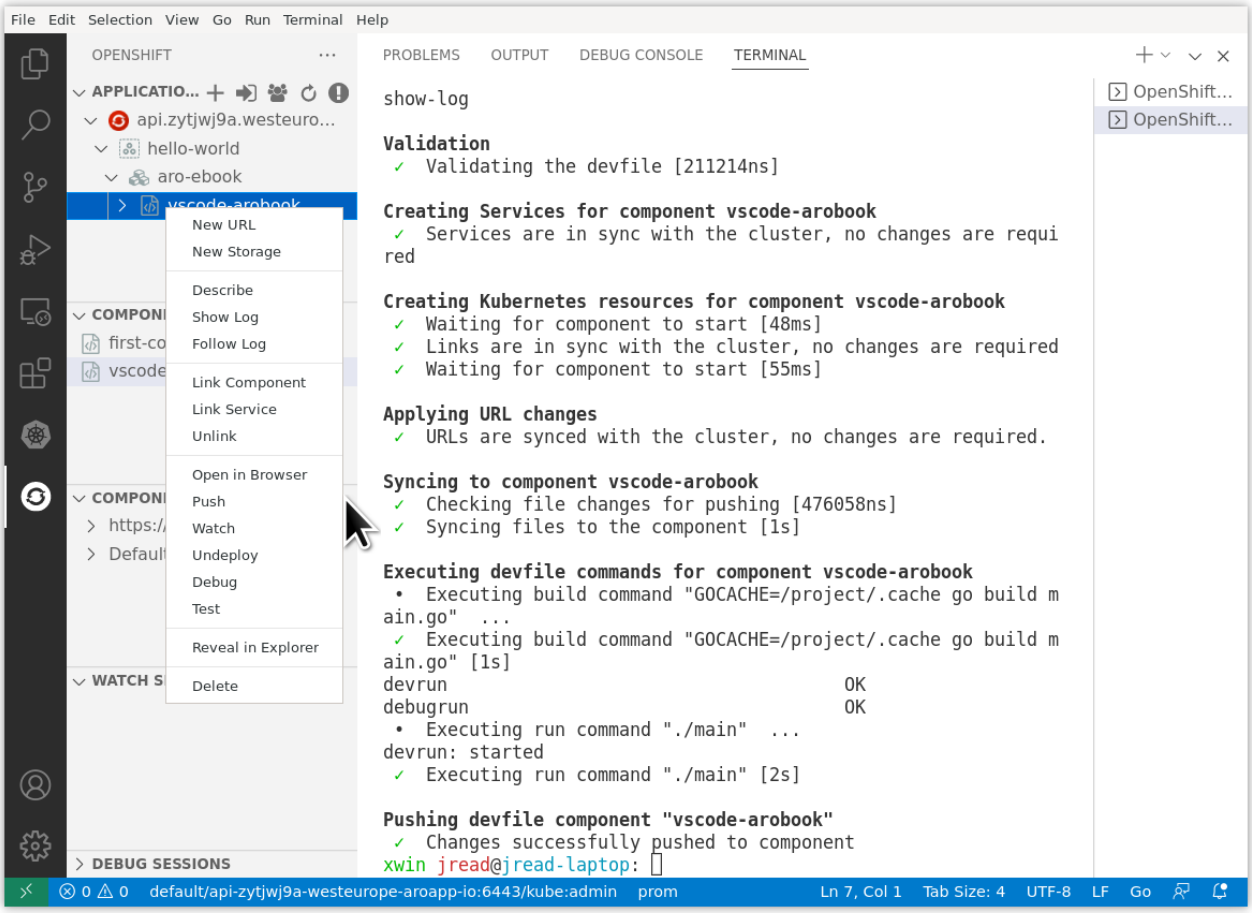


Figura 9.6: O menu "push" em um projeto e um envio recente na janela de terminal

Esse conector apenas acelerou os ciclos de desenvolvimento e teste para os desenvolvedores que preferem usar o Visual Studio Code.

Figura 9.7: A aplicação básica Golang em execução no OpenShift

É claro que os desenvolvedores não estão restritos ao Visual Studio Code, muitos trabalham com o Vim, o Eclipse e outros editores. No entanto, o Visual Studio Code é muito famoso pelos desenvolvedores que gostam de uma experiência no estilo "IDE lightweight".

Leia mais

- Vídeo de demonstração: como usar o Visual Studio Code com o OpenShift
- Conector do Visual Studio Code OpenShift

Integração com o GitHub Actions

Agora é possível usar o GitHub Actions para fazer implantações em todos os ambientes do Red Hat OpenShift, incluindo o Azure Red Hat OpenShift. De qualquer repositório do GitHub, vá até **Actions** → **New Workflow** e, na lista de ações disponíveis, selecione **OpenShift**.

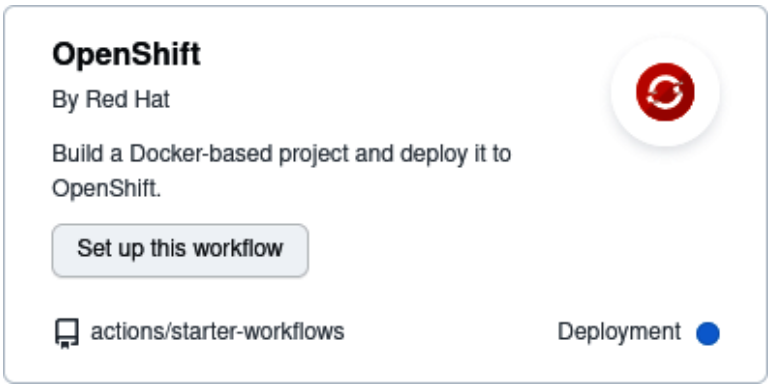


Figura 9.8: Implantação no OpenShift do GitHub

O template padrão vem com um excelente conjunto de exemplos de ações que precisam apenas de algumas variáveis de ambiente definidas para se conectar a um cluster do OpenShift.

```
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. Deixe em branco para usar o
namespace padrão de usuário.
  OPENSHIFT_NAMESPACE: ""
  ...
```

Veja aqui um ótimo post no blog descrevendo como fazer isso, junto com um vídeo de demonstração:

- Post
- Vídeo de demonstração do GitHub Actions com o OpenShift]

Resumo

Este capítulo abordou muitas das integrações que os clientes mais usam com o Azure Red Hat OpenShift. Como falamos na introdução do capítulo, as APIs padrão do OpenShift permitem a integração com muitos outros serviços não listados aqui. Vários serviços também podem ser facilmente integrados usando os operadores listados no OperatorHub.

No próximo capítulo, vamos compartilhar algumas das melhores práticas e recomendações sobre como integrar equipes de aplicações e ganhar força na adoção do serviço na sua organização.

Capítulo 10

Processo de integração de cargas de trabalho e equipes

Depois de concluir as etapas de *pré-provisionamento*, fazer provisionamento do Azure Red Hat OpenShift e seguir os passos obrigatórios de pós-provisionamento, tudo o que precisamos fazer é oferecer suporte e integração para desenvolvedores e aplicações no cluster. Como você faz isso depende, por fim, da cultura da empresa. Algumas equipes de desenvolvedores e aplicações gostam de "ir até o fundo" e colocar a mão na massa, enquanto outras preferem ter mais orientações.

Este capítulo funciona como um conjunto de checklists que oferece uma base sólida para assegurar que suas equipes de desenvolvedores e aplicações possam começar a usar o Azure Red Hat OpenShift com mais rapidez

Adapte e amplie estes checklists de acordo com sua cultura e estrutura organizacionais.

Checklist: Pré-integração

Antes de integrar uma nova carga de trabalho, é importante ser capaz de explicar para a equipe de aplicações, ou para os proprietários da carga de trabalho, que o Azure Red Hat OpenShift foi implantado e projetado para se ajustar às práticas recomendadas da sua organização:

- O Azure Red Hat OpenShift foi implantado em uma zona de destino ou em outro ambiente do Azure que já tenha sido aprovado para as aplicações da organização.
- O cluster tem todas as configurações necessárias de "segunda etapa", como autenticação e classes de armazenamento, descritas no *capítulo 6: Pós-provisionamento (segunda etapa)*.
- A configuração e o suporte do cluster já foram totalmente concluídos pela equipe da plataforma e contam com o suporte integrado oferecido pela Red Hat e pela Microsoft.
- O cluster já foi atualizado com a versão estável mais recente disponível e os patches já foram aplicados. Ele continuará recebendo aplicações de patches daqui em diante.

- O cluster do Azure Red Hat OpenShift tem conectividade com os recursos essenciais obrigatórios:
    - Conectividade com o ambiente on-premise pelo Azure ExpressRoute ou por VPN
    - Conectividade com um repositório de artefatos empresariais, como o Java Maven
    - Conectividade com a internet pública, para download de dependências durante a criação de uma aplicação

Processo de integração de cargas de trabalho piloto

Um padrão comum de implantação do Azure Red Hat OpenShift em uma empresa é a integração de, pelo menos, duas cargas de trabalho piloto:

- **A carga de trabalho significativa mínima**: idealmente como uma equipe de SRE, a primeira carga de trabalho piloto é uma aplicação pequena e conhecida, com requisitos técnicos muito simples. Isso está um pouco acima da aplicação "Hello World", uma vez que, em geral, a aplicação precisa ter um proprietário de negócios real dentro da organização. No entanto, com a primeira carga de trabalho, o foco é verificar se o cluster do Azure Red Hat OpenShift consegue atender às necessidades empresariais no nível do cluster: conectividade do Azure, login no Azure Active Directory e identificação de desafios mais simples, problemas que toda aplicação enfrenta.
- **Uma carga de trabalho de referência**: depois da implantação de uma carga de trabalho simples e mínima, uma adoção mais profunda fica mais fácil se a segunda carga de trabalho for mais complexa e significativa, importante ou até mesmo crítica para os negócios. Ela ajudará a identificar e resolver problemas como conectividade com bancos dados importantes, testes de desempenho e geração de logs/métricas em escala. E, o mais importante, é possível usar essa carga de trabalho de referência significativa como **referência interna** na sua organização. Isso ajudará as futuras equipes de desenvolvimento e aplicação a terem mais confiança na plataforma do Azure Red Hat OpenShift.

É claro que existem outros padrões de integração das primeiras cargas de trabalho que podem funcionar na sua organização. Pode ser necessário priorizar aplicações no datacenter que serão descontinuadas em breve ou que estão sendo executadas em um servidor de aplicações Java antigo.

Checklist: Reunião de integração com outras equipes

Ao integrar uma nova equipe de desenvolvimento ou aplicação ao cluster, uma prática recomendada é fazer uma reunião de integração:

- Crie um ou vários namespaces para utilização da equipe.
- Faça uma breve demonstração do Red Hat OpenShift para a equipe, ressaltando as principais funcionalidades, como a implantação do GitHub (ou similar).
- Verifique se existe espaço suficiente no cluster para implantar a carga de trabalho de destino (CPU, RAM, armazenamento etc.).
- Verifique se os membros da equipe conseguem fazer login no cluster. A conectividade com o Azure Active Directory facilita esse processo.
- Forneça as informações de contato para a equipe de SRE (ou similar) que está gerenciando o cluster.
- Forneça uma lista de links introdutórios ao OpenShift:
    - http://learn.openshift.com
    - https://github.com/openshift-labs/starter-guides
    - http://docs.openshift.com

Checklist: Chamadas regulares de verificação de integridade

Depois que as equipes de desenvolvimento e aplicação iniciarem a implantação do cluster, é aconselhável fazer verificações de integridade com regularidade. Elas podem ser feitas diariamente ou em reuniões semanais de 30 minutos. Veja esta lista de verificação sugerida:

- Como tem sido o desempenho da equipe? Alguma aplicação já foi implantada?
- Houve algum problema recente na utilização do cluster (relativo a conhecimento ou conectividade do Red Hat OpenShift)?
- Houve alguma interrupção no Azure ou no cluster que tenha afetado a experiência negativamente?
- Lembretes sobre a disponibilidade da equipe de SRE e informações de contato para perguntas.

Pense no que mais você já tenha usado em chamadas regulares para verificação de integridade em projetos similares. Inclua no checklist coisas que você acha que podem funcionar.

Antipadrão: ambientes/áreas restritas dos desenvolvedores

Um "antipadrão" comum para incentivar a adoção de desenvolvedores em uma empresa é oferecer um ambiente do tipo área restrita, normalmente chamado de "ambiente do desenvolvedor", e esperar que desenvolvedores e aplicações comecem a adotar o Azure Red Hat OpenShift. Sem nenhum suporte ou recurso de acompanhamento adicional, o Azure Red Hat OpenShift pode ser um ambiente intimidador, com complexidade demais para ser absorvida. Na maioria dos casos, adotar um padrão "área restrita" pode levar a gastos desnecessários com computação em nuvem e clusters vazios.

No entanto, se as equipes de desenvolvimento da sua organização já têm experiência com "áreas restritas", siga estas dicas para aproveitá-las ao máximo:

- Faça pelo menos uma pequena demonstração dos recursos do Azure Red Hat OpenShift.
- Ofereça documentações e links de introdução já sugeridos.
- Organize um evento do tipo "hackathon", desafiando os desenvolvedores a criar algo com o OpenShift, como uma pequena competição.
- Ofereça suporte estendido, uma introdução para a equipe de SRE e uma experiência de integração mais guiada.

A plataforma tem mais chances de ser adotada se você seguir um checklist padrão para adoção guiada, como o descrito anteriormente.

Workshop para desenvolvedores do Azure Red Hat OpenShift

O workshop para desenvolvedores do Azure Red Hat OpenShift (https://aroworkshop.io) é um curso pré-criado que pode ser usado na sua organização para oferecer uma experiência hands-on guiada do Azure Red Hat OpenShift. O workshop é dividido em dois exercícios de laboratório, ambos projetados para oferecer um tutorial detalhado para desenvolvedores ou operadores sem experiência com OpenShift. Os dois laboratórios destacam as principais funcionalidades do OpenShift e mostram como elas podem ser usadas. O primeiro é uma introdução ao design de microsserviços modernos, o segundo aprofunda e detalha o serviço.

Uma boa atividade para divulgar o Azure Red Hat OpenShift na sua empresa é utilizar os conteúdos do aroworkshop.io no seu cluster interno do Azure Red Hat OpenShift. Na discussão antes do workshop, explique como o Azure Red Hat OpenShift tem sido implantado na subscrição atual da sua organização. Fale também sobre como o serviço apresentado pelo curso pode ser uma experiência parecida com a utilização do Azure Red Hat OpenShift para hospedar aplicações de produção.

Resumo

Este capítulo ofereceu checklists e orientações úteis para a implantação bem-sucedida de equipes e cargas de trabalho no Azure Red Hat OpenShift. Descrevemos um antipadrão comum, os clusters de "área restrita" sem suporte. É difícil fornecer uma lista completa de recursos e itens de checklist aplicáveis a todas as organizações. Por isso, é importante adaptar os conteúdos deste capítulo às suas necessidades.

A nuvem oferece uma matriz atrativa de serviços, mas muitos deles são desconhecidos ou adotados de forma insuficiente, por que o foco das empresas fica exclusivamente na tecnologia e implantação. É importante entender esses assuntos, mas eles são apenas uma pequena parte da equação quando falamos de ampliar esses serviços para produção e adoção eficiente. Este capítulo tentou destacar a necessidade dos treinamentos, das verificações regulares e de atividades semelhantes para que você tenha êxito na adoção do Azure Red Hat OpenShift.

Capítulo 11

Conclusão

Às vezes suas equipes de desenvolvimento e operação gastam tempo demais lidando com provisionamento, configuração, manutenção e supervisão de clusters e pipelines de CI/CD. Quando elas fazem isso, não podem se dedicar ao que fazem de melhor: criar aplicações inovadoras.

Como vimos neste guia, com o Azure Red Hat OpenShift, você pode implantar clusters totalmente gerenciados do Red Hat OpenShift sem se preocupar em criar ou gerenciar a infraestrutura necessária para executá-los. Falamos sobre as limitações de executar o Kubernetes sozinho, especialmente em relação à atenção hands-on extra dedicada a tarefas que podem ser automatizadas com o Azure Red Hat OpenShift.

Ao escolher a estratégia de gerenciamento de clusters da sua empresa, compare os prós e contras de uma plataforma do tipo Kubernetes e do Azure Red Hat OpenShift, que é criado com base no framework do Kubernetes e oferece um pacote de benefícios prontos para uso.

Para conhecer melhor o Azure Red Hat OpenShift, visite a página da solução ou veja nossa seção de documentação. Você também pode fazer um workshop hands-on e se inscrever para assistir a um webinar sempre que quiser. E, o mais importante, esperamos que você entre em contato com a Microsoft e a Red Hat para podermos ajudar na avaliação do Azure Red Hat OpenShift.

Autores e versões

James Read <james@redhat.com>
Arquiteto-chefe de soluções da Red Hat,
responsável pela Microsoft – atualizado e revisado para AROv4

Ahmed Sabbour <asabbour@microsoft.com>
Gerente sênior de marketing de produtos na Microsoft,
responsável pelo Azure Red Hat OpenShift – versão inicial para AROv3

Autores de versões anteriores

Oren Kashi <okashi@redhat.com>
Gerente geral sênior de marketing de produto técnico na Red Hat

Agradecemos a Brooke Jackson, Nermina Miller, Jose Moreno, Ahmed Sabbour, Aditya Datar, Vince Power, Alex Patterson e outros que aceitaram se dedicar à revisão deste guia.

Capítulo 12

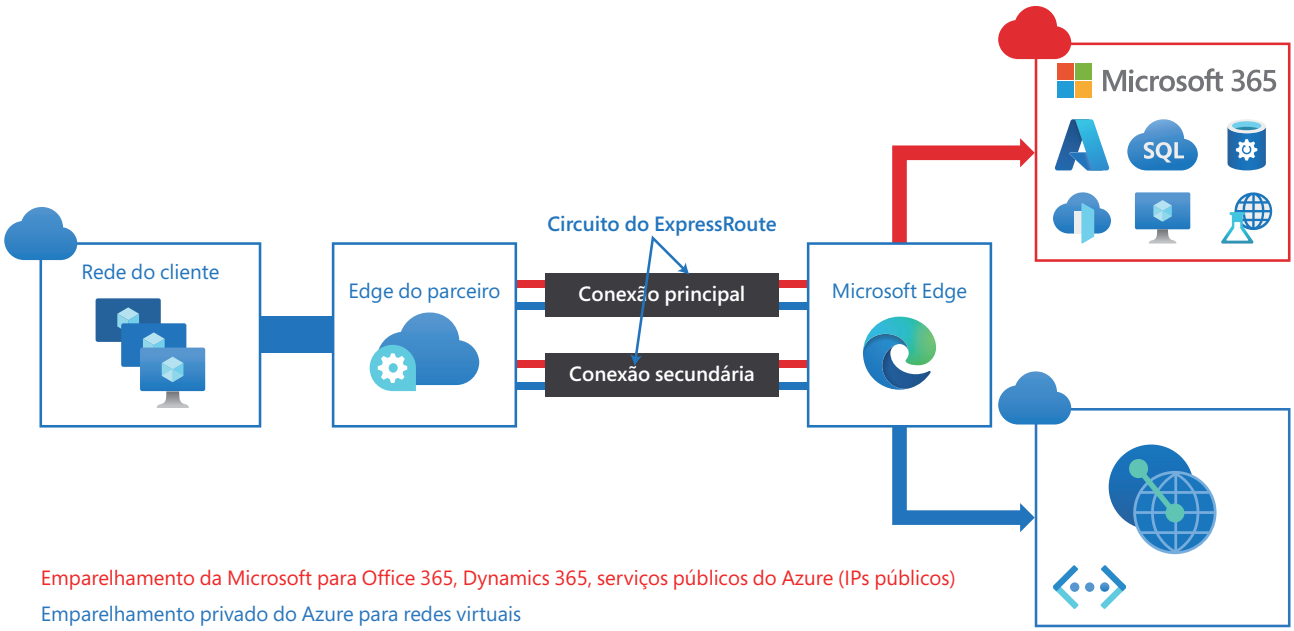
Glossário

Este glossário oferece referências rápidas e úteis para alguns dos termos usados neste guia e no ecossistema do Azure Red Hat OpenShift. Os títulos estão em ordem alfabética.

Azure ExpressRoute

O ExpressRoute permite que você amplie as redes on-premise para a nuvem da Microsoft por uma conexão privada com a assistência de um provedor de conectividade. Com o ExpressRoute, é possível se conectar aos serviços em nuvem da Microsoft, como o Microsoft Azure e o Microsoft 365.

Abaixo está um diagrama da rede do ExpressRoute, retirado da documentação do Microsoft Azure:



Para conhecer melhor o ExpressRoute, veja a página O que é o Azure ExpressRoute?, na documentação do Microsoft Azure.

Para entender como o ExpressRoute funciona com o Azure, veja o *capítulo 4, Pré-provisionamento: questões sobre arquitetura empresarial.*

Compilações e fluxos de imagens.

Uma compilação é o processo de transformar parâmetros de entrada em um objeto resultante. Esse processo costuma ser usado na transformação de parâmetros de entrada ou códigos-fonte em imagens executáveis. Um objeto BuildConfig é a definição do processo de compilação completo.

Para usar o Kubernetes, o Azure Red Hat OpenShift cria containers em formato docker de imagens de compilação e os envia para um registro de imagens em container.

Objetos de compilação compartilham algumas características: entradas para uma compilação, necessidades de concluir um processo de compilação, geração de logs do processo de compilação, publicação de recursos das compilações bem-sucedidas e publicação do status final da compilação. As compilações aproveitam as restrições de recursos, especificando as limitações de funcionalidades como utilização de CPU, uso de memória e tempo de execução do pod ou da compilação.

O sistema de compilação do Azure Red Hat OpenShift oferece suporte extensível para estratégias de compilação baseadas em tipos selecionáveis específicos na API de compilação. Existem três estratégias de compilação principais disponíveis:

- **Compilação do docker**: usando o Dockerfile, que pode ser carregado ou extraído de um repositório de origem
- **Compilação Source-to-Image (S2I)**: pega um repositório de origem, como o Git, e determina como criar a aplicação com arquivos de compilação em linguagem conhecida (por exemplo, arquivos `.pom` do Maven para projetos Java).
- **Compilação personalizada**

Por padrão, as compilações do docker ou S2I são compatíveis.

O objeto resultante de uma compilação depende do compilador usado. Em compilações do docker ou S2I, os objetos resultantes são imagens executáveis. Em compilações personalizadas, os objetos resultantes são especificados pelo autor do compilador de imagem.

Container

Os containers são as unidades básicas das aplicações do Azure Red Hat OpenShift. As tecnologias de containers Linux são mecanismos lightweight usados para isolar processos em execução, de modo que interajam apenas com os recursos designados.

Muitas instâncias de aplicações podem ser executadas em containers com apenas um host, sem visibilidade mútua de processos, arquivos, redes etc. Em geral, cada container oferece apenas um serviço (normalmente chamado de "microsserviço"), como um servidor web ou um banco de dados, embora possam ser usados com cargas de trabalho arbitrárias.

Imagens de container

Os containers do Azure Red Hat OpenShift são baseados em imagens de container no formato docker. Uma imagem é um binário que contém todos os requisitos para execução de um container, assim como os metadados que descrevem necessidades e recursos.

Pense nela com uma tecnologia de empacotamento. Os containers têm acesso somente a recursos definidos na imagem, a não ser que você conceda acessos adicionais na hora da criação. Ao implantar a mesma imagem em containers diferentes em vários hosts e fazer o balanceamento de carga entre eles, o Azure Red Hat OpenShift oferece redundância e escala horizontal para um serviço empacotado na imagem.

Implantações e configurações de implantação

Com base em controladores de replicação, o Azure Red Hat OpenShift inclui suporte estendido ao ciclo de vida de implantação e desenvolvimento do software com um novo conceito de implantação. No caso mais simples, uma implantação apenas cria um novo ReplicationController e permite a inicialização de pods. No entanto, as implantações do Azure Red Hat OpenShift também viabilizam a transição de uma imagem de implantação existente para uma nova, além de definir ganchos a serem executados antes ou depois da criação do ReplicationController.

O objeto DeploymentConfig do Azure Red Hat OpenShift define os seguintes detalhes de uma implantação:

1. Os elementos de uma definição de ReplicationController
2. Os gatilhos resultantes da criação automática de uma nova implantação
3. A estratégia de transição entre implantações
4. Gancho de ciclo de vida

Sempre que uma implantação é acionada, manual ou automaticamente, ela é gerenciada por um pod do implantador, incluindo a diminuição do ReplicationController antigo, o aumento do novo e a execução dos ganchos. Depois que a implantação é concluída, o pod continua indefinidamente com o objetivo de reter os logs desse processo. Quando uma implantação é substituída por outra, o ReplicationController anterior é mantido para facilitar a reversão, se necessário.

Para instruções detalhadas sobre como criar implantações e interagir com elas, leia esta página de Implantações e DeploymentConfigs.

Pods e serviços

O Azure Red Hat OpenShift usa o conceito de pod do Kubernetes, que é um ou mais containers implantados juntos em um host, sendo a menor unidade de computação que pode ser definida, implantada e gerenciada.

Os pods são o equivalente aproximado a uma instância de máquina, física ou virtual, para um container. Os pods são alocados no próprio endereço IP interno, controlando todo o espaço da porta. Os containers dentro dos pods podem compartilhar armazenamento e rede locais.

Eles têm um ciclo de vida: depois de definidos, são colocados em execução em um nó até que o container saia ou eles sejam removidos por algum outro motivo. Dependendo da política e do código de saída, os pods podem ser removidos depois da saída ou retidos para viabilizar o acesso aos logs dos containers.

O Azure Red Hat OpenShift trata os pods como imutáveis. Não é possível fazer alterações na definição de um pod durante a execução. Para implementar as alterações, o Azure Red Hat OpenShift encerra e recria o pod com a configuração modificada, a imagem base ou ambos. Os componentes do ambiente de execução são tratados como expansíveis e são recriados da imagem de container definida. Assim, normalmente os controladores dos pods são de nível mais alto, e não usuários diretos.

Projetos e usuários

Um projeto é um username do Kubernetes com anotações adicionais e o principal veículo usado para gerenciamento do acesso de usuários regulares aos recursos. Com ele, uma comunidade de usuários pode organizar e gerenciar conteúdos separadamente de outras comunidades. Os usuários só podem acessar os projetos com permissão dos administradores. Se tiverem autorização para criar projetos, eles têm acesso automático aos próprios projetos. Os projetos podem ter um nome, um `displayName` e uma descrição separados.

O nome obrigatório é um identificador único para o projeto e é mais visível com o uso das ferramentas de CLI ou da API. O nome pode ter até 63 caracteres. O `displayName` é o modo de exibição do projeto no console web (uso `name` como padrão). A descrição opcional pode detalhar melhor o projeto e também é visível no console web.

Desenvolvedores e administradores podem interagir com projetos usando a CLI ou o console web.

Registro de containers

O Azure Red Hat OpenShift oferece um registro integrado de imagens de container chamado **OpenShift Container Registry (OCR)**, que adiciona o recurso de provisionamento automático sob demanda de novos repositórios de imagem. Isso oferece aos usuários uma localização integrada para que as compilações das aplicações enviem as imagens resultantes.

Quando uma nova imagem é enviada para o OCR, o registro fornece todas as informações ao Azure Red Hat OpenShift, como namespace, nome e metadados da imagem. Partes diferentes do Azure Red Hat OpenShift reagem às novas imagens, criando novas compilações e implantações.

O Azure Red Hat OpenShift também pode utilizar qualquer servidor. Para isso, basta implementar a API do registro de imagens de container como uma fonte de imagens, incluindo o Docker Hub e o Azure Container Registry.

ReplicaSets

Similar ao ReplicationController, um ReplicaSet sempre garante a execução de um número especificado de réplicas de pod. A diferença entre o ReplicaSet e o ReplicationController é que o primeiro tem suporte para requisitos de seletores baseados em grupo, enquanto o segundo tem suporte apenas para requisitos de seletores baseados em igualdade.

```
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
```

No código acima, é possível ver:

- Uma consulta de rótulo sobre um conjunto de recursos. Os resultados de `matchLabels` e `matchExpressions` são logicamente conjuntos.
- Um seletor baseado em igualdade para especificar recursos com rótulos que correspondem a ele.
- Um seletor baseado em grupo para filtrar as chaves. Isso seleciona todos os recursos com key igual a `tier` e value igual a `frontend`.

ReplicationController

Um ReplicationController garante que um número especificado de réplicas sempre esteja em execução. Se o pod sair ou for excluído, o ReplicationController entra em ação para instanciar mais, até atingir o número definido. Da mesma forma, se houver mais pods em execução do que o desejado, ele exclui quantos forem preciso para atingir a quantidade definida.

Uma configuração de ReplicationController contém:

- O número de réplicas desejadas (ajustável no ambiente de execução).
- Uma definição de pod a ser usada na criação da réplica.
- Um seletor para identificação de pods gerenciados.
- Um seletor é um conjunto de rótulos atribuídos aos pods que são gerenciados pelo ReplicationController. Esses rótulos são inclusos na definição do pod que o ReplicationController instancia. O ReplicationController usa o seletor para determinar quantas instâncias do pod já estão em execução e fazer os ajustes necessários.

O ReplicationController não faz escala automática baseada na carga ou no tráfego nem rastreia os dois. Para fazer isso, a réplica precisaria ser ajustada por um escalonador automático externo.

Um ReplicationController é objeto Kubernetes central chamado ReplicationController. Veja este exemplo de uma definição do ReplicationController:

```
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
```

No código acima, é possível ver:

- O número de cópias do pod a serem executadas.
- O seletor de rótulo do pod a ser executado.
- Um template para o pod criado pelo controlador.
- Os rótulos do pod devem incluir os rótulos do seletor.
- O nome após a expansão dos parâmetros pode ter até 63 caracteres.

Rotas e entradas

O Azure Red Hat OpenShift tem suporte para rotas e entradas. As duas são usadas para expor um serviço usando um nome DNS, como www.example.com, para que clientes externos possam acessá-lo.

As rotas foram originalmente concebidas no Red Hat OpenShift versão 3. Desde o lançamento, a Red Hat trabalhou junto à comunidade do Kubernetes para padronizar esse recurso no que agora é chamado de **Ingress** (entrada).

O recurso de entrada do Kubernetes no OpenShift Container Platform implementa o controlador de entrada com um serviço de roteador compartilhado que é executado como um pod dentro do cluster. A maneira mais comum de gerenciar o tráfego de entrada é com o controlador de entrada. Você pode escalar e replicar esse pod como qualquer outro. O serviço de roteador é baseado em HAProxy, uma solução open source de balanceamento de carga.

A rota do OpenShift Container Platform oferece tráfego de entrada para serviços no cluster. As rotas oferecem funcionalidades avançadas que talvez não tenham suporte dos controladores de entrada padrão do Kubernetes, como re-criptografia de TLS, passagem de TLS e tráfego dividido para implantações azul/verde.

O tráfego de entrada usa uma rota para acessar serviços no cluster. Rotas e entradas são os principais recursos para gerenciar o tráfego de entrada. Uma entrada oferece funcionalidades parecidas com as da rota, como aceitação de solicitações externas e delegação baseada em rota. No entanto, uma entrada só permite certos tipos de conexão: HTTP/2, HTTPS e **identificação de nome de servidor (SNI)**, além de TLS com certificado. No OpenShift Container Platform, as rotas são geradas para atender às condições especificadas pelo recurso de entrada.

Source-to-Image (S2I)

S2I é um kit de ferramentas e um fluxo de trabalho para criar imagens de container reproduzíveis a partir do código-fonte. Para oferecer imagens prontas para uso, o S2I injeta o código-fonte em uma imagem de container, que prepara o código para a execução. Ao criar imagens de compilador com montagem automática, é possível controlar a versão dos ambientes de compilação exatamente como você usa as imagens de container para definir a versão dos ambientes de execução.

Para uma linguagem dinâmica como o Ruby, os ambientes de execução e de tempo de compilação são normalmente os mesmos. Começando por uma imagem de compilador que descreve esse ambiente com Ruby, Bundler, Rake, Apache, GCC e outros pacotes necessários para configurar e executar uma aplicação Ruby instalada, o S2I executa os seguintes passos:

1. Inicie o container do compilador de imagem com o código-fonte da aplicação injetado em um diretório conhecido.
2. O processo do container transforma esse código-fonte na configuração executável adequada. Neste caso, instalando as dependências com o Blunder e migrando o código-fonte para um diretório em que o Apache foi pré-configurado para procurar o arquivo `config.ru` do Ruby.
3. Confirme o novo container e defina o ponto de entrada da imagem como um script (fornecido pelo compilador de imagem) que iniciará o Apache para hospedar a aplicação Ruby.

Para linguagens compiladas, como C, Go ou Java, as dependências necessárias para compilação talvez sejam muito maiores que o tamanho dos artefatos reais do ambiente de execução. Para manter as imagens do ambiente de execução leves, o S2I viabiliza processos de compilação em várias etapas. Neles, um artefato binário, como um arquivo executável ou WAR Java, é criado no primeiro compilador de imagem, extraído e injetado em uma segunda imagem do ambiente de execução que apenas coloca o executável no local correto de execução.

Por exemplo, para criar um pipeline de compilação reproduzível para Tomcat (um conhecido servidor web Java) e Maven, siga estes passos:

1. Crie uma imagem de compilador com OpenJDK e Tomcat que espera a injeção de um arquivo WAR.
2. Crie uma segunda imagem que coloca as dependências do Maven ou de qualquer outro padrão em uma camada acima da primeira imagem e espera a injeção do projeto do Maven.
3. Invoque o S2I usando o código-fonte da aplicação Java e a imagem do Maven para criar o arquivo WAR da aplicação desejada.
4. Invoque o S2I de novo usando o arquivo WAR do passo anterior e a imagem do Tomcat inicial para criar uma imagem do ambiente de execução.

Ao inserir nossa lógica de compilação nas imagens e combiná-las em várias etapas, aproximamos o ambiente de execução do ambiente de compilação (mesmo JDK, mesmos JARs do Tomcat) sem precisar da implantação de ferramentas de build para produção.

Os objetivos e benefícios de usar o S2I como estratégia de compilação são:

- **Reprodutibilidade**: viabiliza versões leves dos ambientes de compilação, encapsulando-os dentro de uma imagem de container e definindo uma interface simples (com código-fonte injetado) para chamadores. Compilações reproduzíveis são um requisito essencial para viabilizar atualizações de segurança e integração contínua em uma infraestrutura em containers, e as imagens do compilador ajudam a garantir a repetitividade e a capacidade de trocar de ambientes de execução.
- **Flexibilidade**: todos os sistemas de compilação executáveis no Linux podem ser executados dentro de um container, e cada compilador também pode ser parte de um pipeline maior. Além disso, os scripts que processam o código-fonte da aplicação podem ser injetados no compilador de imagem, permitindo que autores adaptem as imagens existentes para viabilizar o gerenciamento do código-fonte.
- **Velocidade**: em vez de compilar várias camadas em um Dockerfile, o S2I incentiva os autores a representar uma aplicação em apenas uma camada de imagem. Isso economiza tempo durante a criação e implantação, além de possibilitar maior controle sobre a saída da imagem final.
- **Segurança**: compilações que usam Dockerfiles são executadas sem muitos dos controles operacionais de containers mais comuns, normalmente executadas como raiz e com acesso à rede do container. O S2I pode ser usado para controlar quais permissões e privilégios estão disponíveis para o compilador de imagem desde o lançamento da compilação em um só container. Junto a plataformas como o OpenShift, o S2I concede aos administradores um controle maior sobre os privilégios dos desenvolvedores durante o tempo de compilação.

Tarefas

Uma tarefa é parecida com o ReplicationController, uma vez que ela tem o objetivo de criar pods por motivos específicos. A diferença é que os controladores de replicação são feitos para pods de execução contínua, enquanto as tarefas são feitas para jobs de execução única. Uma tarefa rastreia as conclusões bem-sucedidas e finaliza quando atinge uma quantidade especificada.

Para mais informações, veja o tópico *Tarefas*.

Templates

Um template descreve um conjunto de objetos que podem ser parametrizados e processados para produzir uma lista de objetos para criação pelo Azure Red Hat OpenShift. Um template pode ser processado para criar tudo que você tem permissão para criar em um projeto. Por exemplo, serviços, configurações de compilação e configurações de implantação. Ele também pode definir um conjunto de rótulos a serem aplicados a todos os objetos definidos no template.

Com um template, é possível criar uma lista de objetos usando o CLI ou o console web, caso o template tenha sido carregado no seu projeto ou na biblioteca de templates global. Para um conjunto de templates selecionados, veja os fluxos de imagem e a biblioteca de templates do OpenShift.

Zonas de destino do Azure

As zonas de destino do Azure são um padrão de implantação conhecido para o planejamento empresarial de implantação em larga escala com o uso do Azure e que leva em consideração escala, governança de segurança, rede e identidade.

Introdução à zona de destino do Azure

Existe um projeto da comunidade do GitHub em desenvolvimento que dá algumas recomendações sobre como implantar o Azure Red Hat OpenShift em uma arquitetura de zona de destino do Azure: https://github.com/Azure/Enterprise-Scale/tree/main/workloads/ARO