

Uso e gerenciamento do vRealize Automation Code Stream

14 DE DEZEMBRO DE 2022
vRealize Automation 8.2

Você pode encontrar a documentação técnica mais atualizada no site da VMware, em:

<https://docs.vmware.com/br/>

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

VMware Brasil
Rua Surubim, 504 4º andar CEP 04571-050
Cidade Monções
São Paulo
SÃO PAULO: 04571-050
Brasil
Tel: +55 11 55097200
Fax: + 55. 11. 5509-7224
www.vmware.com/br

Copyright © 2022 VMware, Inc. Todos os direitos reservados. [Informações sobre direitos autorais e marca registrada.](#)

Conteúdo

- 1 O que é o vRealize Automation Code Stream e como ele funciona 5**
- 2 Configuração para modelar meu processo de liberação 10**
 - Como adicionar um projeto 14
 - Como gerenciar o acesso do usuário e as aprovações 15
 - O que são operações do usuário e aprovações 23
- 3 Como criar e usar pipelines 26**
 - Como executar um pipeline e ver os resultados 29
 - Que tipos de tarefas estão disponíveis 34
 - Como usar associações de variáveis em pipelines 37
 - Como usar associações de variáveis em uma tarefa de condição para executar ou parar um pipeline 48
 - Que variáveis e expressões eu posso usar ao associar tarefas de pipeline 51
 - Como enviar notificações sobre meu pipeline 69
 - Como criar um tíquete do Jira quando uma tarefa de pipeline falhar 71
 - Como reverter minha implantação 74
- 4 Como planejar compilar, integrar e entregar seu código de forma nativa 81**
 - Como planejar uma compilação nativa de CICD antes de usar o modelo de pipeline inteligente 81
 - Como planejar uma compilação nativa de CI antes de usar o modelo de pipeline inteligente 87
 - Como planejar uma compilação nativa de CD antes de usar o modelo de pipeline inteligente 88
 - Como planejar uma compilação nativa de CICD antes de adicionar tarefas manualmente 90
 - Como planejar uma reversão 96
- 5 Tutoriais 99**
 - Como integrar continuamente o código do meu repositório GitHub ou GitLab ao pipeline 100
 - Como automatizar a liberação de um aplicativo implantado a partir de um modelo de nuvem YAML 106
 - Como automatizar a liberação de um aplicativo para um cluster do Kubernetes 113
 - Como implantar meu aplicativo na implantação Azul-Verde 121
 - Como integrar minhas próprias ferramentas de compilação, teste e implantação 126
 - Como usar uma REST API para integração com outros aplicativos 136
- 6 Conectando-se a endpoints 141**
 - O que são endpoints 141
 - Como integrar ao Jenkins 143

[Como integrar ao Git 150](#)

[Como integrar ao Gerrit 153](#)

[Como integrar ao vRealize Orchestrator 155](#)

7 Como disparar pipelines 161

[Como usar o gatilho do Docker para executar um pipeline de entrega contínua 161](#)

[Como usar o gatilho Git para executar um pipeline 169](#)

[Como usar o gatilho Gerrit para executar um pipeline 177](#)

8 Como monitorar pipelines 185

[Como rastrear os principais indicadores de desempenho do pipeline 185](#)

9 Saiba mais 189

[O que é pesquisa 189](#)

[Mais recursos para administradores e desenvolvedores 195](#)

O que é o vRealize Automation Code Stream e como ele funciona

1

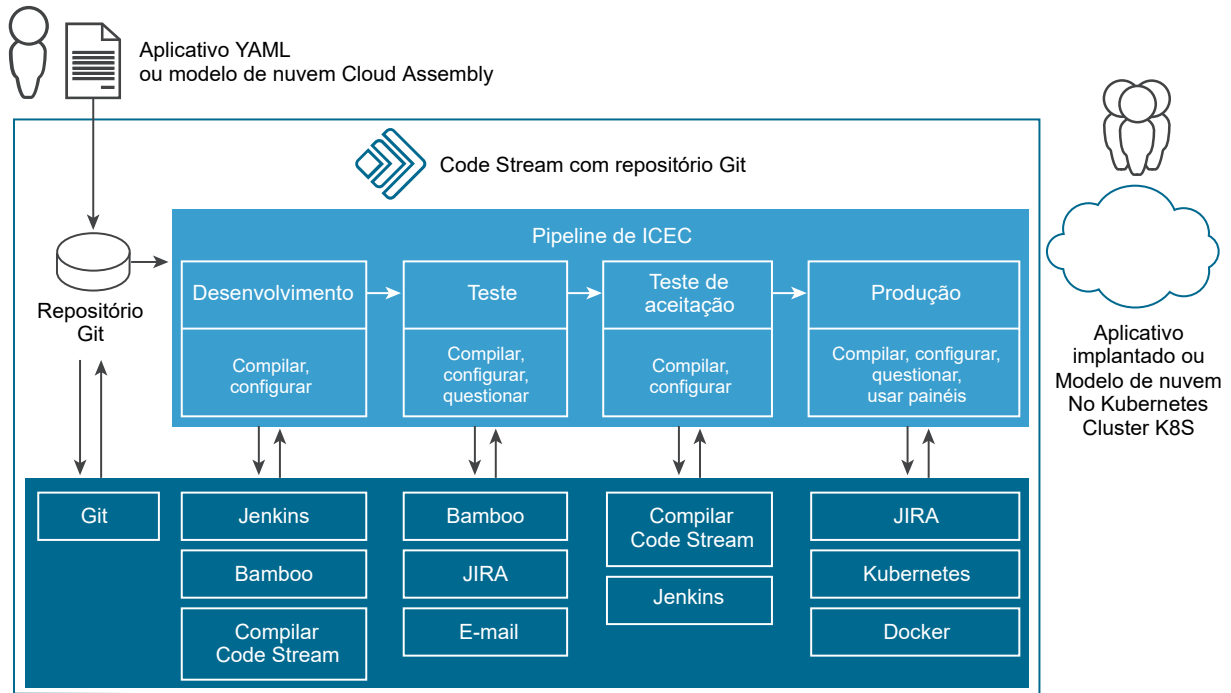
O vRealize Automation Code Stream™ é uma ferramenta de integração contínua e entrega contínua (CI/CD) usada para compilar pipelines que modelam o processo de liberação de softwares no seu ciclo de vida de DevOps. Ao criar pipelines, você compila a infraestrutura de código que entrega o software de forma rápida e contínua.



Ao usar o vRealize Automation Code Stream para entregar seu software, integre duas das partes mais importantes de seu ciclo de vida do DevOps: seu processo de liberação e suas ferramentas de desenvolvedor. Após a configuração inicial, que integra o vRealize Automation Code Stream às suas ferramentas de desenvolvimento existentes, os pipelines automatizam todo o ciclo de vida de DevOps.

A partir do vRealize Automation 8.2, blueprints se chamam VMware Cloud Templates.

Você cria um pipeline que compila, testa e libera seu software. O vRealize Automation Code Stream usa esse pipeline para conduzir seu software desde o repositório do código-fonte, passando pela fase de testes, até a produção.



Você pode saber mais sobre como planejar a integração contínua e pipelines de entrega contínua em [Capítulo 4 Planejamento para compilar, integrar e entregar seu código de forma nativa no vRealize Automation Code Stream](#).

Como os administradores do vRealize Automation Code Stream usam o vRealize Automation Code Stream

Como administrador, você cria endpoints e garante que as instâncias de trabalho estejam disponíveis para desenvolvedores. É possível criar, disparar e gerenciar pipelines e muito mais. Você tem a função `Administrator`, conforme descrito em [Como gerenciar o acesso do usuário e as aprovações no vRealize Automation Code Stream](#).

Tabela 1-1. Como os administradores do vRealize Automation Code Stream oferecem suporte aos desenvolvedores

Para oferecer suporte aos desenvolvedores...	Veja o que é possível fazer...
Fornecer e gerenciar ambientes.	<p>Criar ambientes para os desenvolvedores testarem e implantarem seu código.</p> <ul style="list-style-type: none"> ■ Rastrear o status e enviar notificações por e-mail. ■ Manter seus desenvolvedores produtivos garantindo que seus ambientes funcionem continuamente. <p>Para saber mais, consulte Mais recursos para administradores e desenvolvedores do vRealize Automation Code Stream.</p> <p>Consulte também Capítulo 5 Tutoriais para usar o vRealize Automation Code Stream.</p>
Forneça endpoints.	Certifique-se de que os desenvolvedores tenham instâncias de trabalho de endpoints que possam se conectar aos pipelines.

Tabela 1-1. Como os administradores do vRealize Automation Code Stream oferecem suporte aos desenvolvedores (continuação)

Para oferecer suporte aos desenvolvedores...	Veja o que é possível fazer...
Fornecer integrações a outros serviços.	Certifique-se de que as integrações a outros serviços estejam funcionando. Para saber mais, consulte a documentação do vRealize Automation .
Criar pipelines.	Crie pipelines que modelem os processos de liberação. Para saber mais, consulte Capítulo 3 Criando e usando pipelines no vRealize Automation Code Stream .
Disparar pipelines.	Certifique-se de que os pipelines sejam executados quando eventos ocorrerem. <ul style="list-style-type: none"> ■ Para disparar um pipeline autônomo de entrega contínua (CD), sempre que um artefato de compilação for criado ou atualizado, use o gatilho do Docker. ■ Para disparar um pipeline quando um desenvolvedor confirmar alterações no seu código, use o gatilho Git. ■ Para disparar um pipeline quando os desenvolvedores revisarem o código, fizerem uma mesclagem e mais, use o gatilho Gerrit. ■ Para executar um pipeline autônomo de entrega contínua (CD), sempre que um artefato de compilação for criado ou atualizado, use o gatilho do Docker. Para saber mais, consulte Capítulo 7 Disparando pipelines no vRealize Automation Code Stream .
Gerenciar pipelines e aprovações.	Mantenha-se atualizado sobre pipelines. <ul style="list-style-type: none"> ■ Visualize o status dos pipelines e veja quem os executou. ■ Exiba as aprovações em execuções do pipeline e gerencie as aprovações para execuções do pipeline ativas e inativas. Para saber mais, consulte O que são operações do usuário e aprovações no vRealize Automation Code Stream . Consulte também Como rastrear os principais indicadores de desempenho do pipeline no vRealize Automation Code Stream .

Tabela 1-1. Como os administradores do vRealize Automation Code Stream oferecem suporte aos desenvolvedores (continuação)

Para oferecer suporte aos desenvolvedores...	Veja o que é possível fazer...
Monitorar os ambientes de desenvolvedor.	<p>Crie painéis personalizados que monitoram o status do pipeline, suas tendências, métricas e indicadores principais. Use esses painéis personalizados para monitorar pipelines que são aprovados ou reprovados em ambientes de desenvolvimento. Você também pode identificar e relatar recursos subutilizados e liberar recursos.</p> <p>Consulte também:</p> <ul style="list-style-type: none"> ■ Há quanto tempo um pipeline foi executado antes de ser bem-sucedido. ■ Há quanto tempo um pipeline aguardou a aprovação e notificar o usuário que deve aprová-lo. ■ Estágios e tarefas que falham com mais frequência. ■ Estágios e tarefas que levam mais tempo para serem executados. ■ Liberar as equipes de desenvolvimento têm em andamento. ■ Aplicativos que foram bem-sucedidos ao serem implantados e liberados. <p>Para saber mais, consulte Capítulo 8 Monitorando pipelines no vRealize Automation Code Stream.</p>
Solucionar problemas.	<p>Solucionar problemas e resolver falhas de pipeline em ambientes de desenvolvimento.</p> <ul style="list-style-type: none"> ■ Identifique e resolva problemas em ambientes de integração contínua e entrega contínua (CI/CD). ■ Use painéis de pipeline e crie painéis personalizados para ver mais. Consulte Capítulo 8 Monitorando pipelines no vRealize Automation Code Stream. <p>Consulte também Capítulo 2 Como configurar o vRealize Automation Code Stream para modelar meu processo de liberação.</p>

O vRealize Automation Code Stream faz parte do vRealize Automation. O vRealize Automation Code Stream integra-se ao:

- Use o vRealize Automation Cloud Assembly para implantar modelos de nuvem.
- Use o vRealize Automation Service Broker para obter modelos de nuvem do catálogo.

Para saber mais sobre o que você pode fazer, consulte a [Documentação do VMware vRealize Automation](#).

Como os desenvolvedores usam o vRealize Automation Code Stream

Como desenvolvedor, você usa o vRealize Automation Code Stream para compilar e executar pipelines e monitorar a atividade do pipeline nos painéis. Você tem a função `user`, conforme descrito em [Como gerenciar o acesso do usuário e as aprovações no vRealize Automation Code Stream](#).

Depois de executar um pipeline, você desejará saber:

- Meu código foi bem-sucedido em todos os estágios do pipeline? Observe os resultados em **Execuções**.

- O que fazer se o pipeline tiver falhado e o que causou a falha? Veja os principais erros que ocorreram em **Painéis**.

Tabela 1-2. Desenvolvedores que usam o vRealize Automation Code Stream

Para integrar e liberar seu código...	Você deve...
Compilar pipelines.	Testar e implantar seu código. Atualizar seu código quando um pipeline falhar.
Conectar seu pipeline a endpoints.	Conectar as tarefas do seu pipeline a endpoints, como um repositório GitHub.
Executar pipelines.	Adicione uma tarefa de aprovação da operação do usuário para que outro usuário possa aprovar seu pipeline em pontos específicos.
Exibir painéis.	Exiba os resultados no painel do pipeline. É possível visualizar tendências, histórico, falhas e muito mais.

Para obter mais informações sobre como começar, consulte [Como começar: VMware Code Stream](#).

Encontre conteúdo de documentação adicional no painel Suporte no produto

Se você não encontrar as informações necessárias aqui, poderá obter mais ajuda no produto. 

- Clique e leia as sinalizações e as dicas de ferramentas na interface do usuário do para obter as informações específicas do contexto de que você precisa e exatamente quando precisar delas.
- Abra o painel de suporte no produto e leia os tópicos que aparecem para a página da interface do usuário ativa. Você também pode pesquisar no painel para obter respostas às perguntas.

Mais sobre webhooks

Você pode criar vários webhooks para ramificações diferentes usando o mesmo endpoint Git e fornecendo diferentes valores para o nome da ramificação na página de configuração do webhook. Para criar outro webhook para outra ramificação no mesmo repositório Git, não é necessário clonar o endpoint Git várias vezes para várias ramificações. Em vez disso, forneça o nome da ramificação no webhook, o que permite reutilizar o endpoint Git. Se a ramificação no webhook Git for a mesma que a ramificação no endpoint, você não precisará fornecer o nome da ramificação na página do webhook Git.

Como configurar o vRealize Automation Code Stream para modelar meu processo de liberação

2

Para modelar seu processo de liberação, crie um pipeline que represente os estágios, tarefas e as aprovações que você normalmente usa para liberar o software. Em seguida, o vRealize Automation Code Stream automatiza o processo que compila, testa, aprova e implanta seu código.

Agora que você tem tudo para modelar seu processo de liberação de software, veja como fazer isso no vRealize Automation Code Stream.

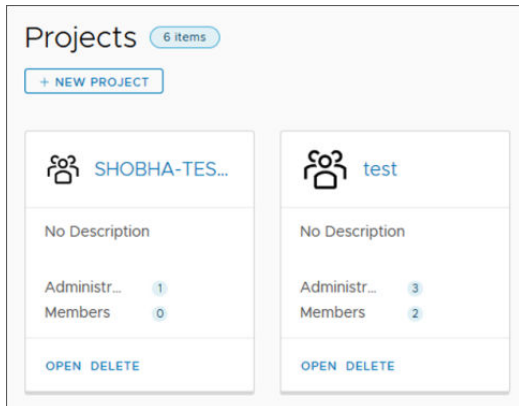
Pré-requisitos

- Verifique se já existem endpoints disponíveis. No vRealize Automation Code Stream, clique em **Endpoints**.
- Aprenda sobre maneiras nativas de criar e implantar seu código. Consulte [Capítulo 4 Planejamento para compilar, integrar e entregar seu código de forma nativa no vRealize Automation Code Stream](#).
- Determine se alguns dos recursos que serão usados no seu pipeline devem ser marcados como restritos. Consulte [Como gerenciar o acesso do usuário e as aprovações no vRealize Automation Code Stream](#).
- Se você tiver a função de usuário ou expectador em vez da função de administrador, determine quem é o administrador da sua instância do vRealize Automation Code Stream.

Procedimentos

- 1 Examine os projetos disponíveis no vRealize Automation Code Stream e selecione um que seja adequado para você.
 - Se nenhum projeto estiver listado, peça a um administrador do vRealize Automation Code Stream para criar um projeto e torná-lo um membro desse projeto. Consulte [Como adicionar um projeto no vRealize Automation Code Stream](#).

- Se você não for membro de nenhum dos projetos listados, peça a um administrador do vRealize Automation Code Stream para adicioná-lo como membro de um projeto.



- 2 Adicione novos endpoints necessários para o pipeline.

Por exemplo, você pode precisar de Git, Jenkins, Compilação de Code Stream, Kubernetes e Jira.

- 3 Crie variáveis para que você possa reutilizar valores em suas tarefas de pipeline.

Use variáveis restritas para restringir os recursos usados nos pipelines, como um computador host. Você pode impedir o pipeline de continuar a ser executado até que outro usuário o aprove explicitamente.

Os administradores podem criar variáveis secretas e variáveis restritas. Os usuários podem criar variáveis secretas.

Você pode reutilizar uma variável quantas vezes desejar em vários pipelines. Por exemplo, uma variável que define um computador host pode ser definida como `HostIPAddress`. Em seguida, para usar a variável em uma tarefa de pipeline, digite `${var.HostIPAddress}`.

Project	Name	Type	Value
Code Stream	Test	Regular	123
Code Stream	Test-Restricted	Restricted	*****
Code Stream	Test-Global-name	Secret	*****

- 4 Se você for administrador, marque quaisquer endpoints e variáveis que sejam vitais para seus negócios como recursos restritos.

Quando um usuário que não é administrador tenta executar um pipeline que inclui um recurso restrito, o pipeline é interrompido na tarefa que usa esse recurso restrito. Em seguida, um administrador deve retomar o pipeline.

5 Planeje a estratégia de compilação para seu pipeline nativo de CI/CD, CI ou CD.

Antes de criar um pipeline que integre continuamente (CI) e implante continuamente (CD) seu código, planeje sua estratégia de compilação. O plano de compilação ajuda a determinar do que o vRealize Automation Code Stream precisa para poder compilar, integrar, testar e implantar seu código de forma nativa.

Como criar uma compilação nativa do vRealize Automation Code Stream...

Use um dos modelos de pipeline inteligentes.

Resultados nesta estratégia de compilação...

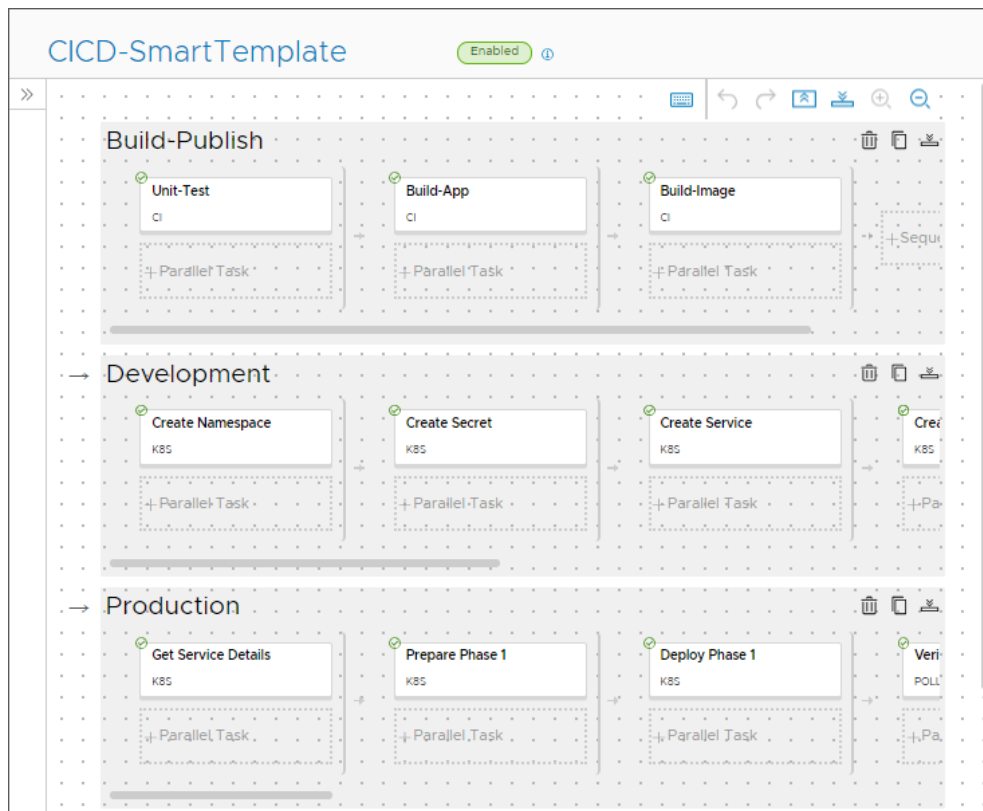
- Compila todos os estágios e tarefas para você.
- Clona o repositório de origem.
- Compila e testa seu código.
- Armazena seu código em contêineres para implantação.
- Preenche as etapas da tarefa de pipeline com base nas suas seleções.

Adicionar estágios e tarefas manualmente.

Você adiciona estágios, tarefas e digita as informações que os preenche.

6 Crie seu pipeline usando um modelo de pipeline inteligente ou adicione estágios e tarefas manualmente ao pipeline.

Em seguida, marque todos os recursos como restritos. Adicione aprovações onde necessário. Aplique qualquer variável regular, restrita ou secreta. Adicione quaisquer associações entre tarefas.



- 7 Valide, ative e execute o pipeline.
- 8 Exibir as execuções do pipeline.

The screenshot shows the 'Executions' page with a search bar and a 'NEW EXECUTION' button. It lists four pipeline executions:

- Demo-Jenkins... #95**: Status: COMPLETED. Stages: 2/2. By kr on 09/11/2018 10:32 AM. Execution Completed. Input: 8df0d9a1d365299f2... Output: NA.
- Demo-Jenkins... #94**: Status: COMPLETED. Stages: 2/2. By kr on 09/11/2018 9:17 AM. Execution Completed. Input: 6d82d079a8b8921a9... Output: NA.
- Demo-CICD-S... #51**: Status: COMPLETED. Stages: 3/3. By dk on 09/11/2018 7:13 AM. Execution Completed. Input: NA. Output: NA.
- Demo-CICD-S... #50**: Status: FAILED. Stages: 2/3. By dk on 09/11/2018 5:51 AM. Execution failed on task 'Production.Deploy Phase 1'. deployments... Input: NA. Output: NA.

- 9 Para rastrear o status e os KPIs, use os painéis de pipeline e crie painéis personalizados.

The screenshot shows the 'CICD-SmartTemplate' dashboard with the following sections:

- Execution Status Counts**: A donut chart showing 'Total: 1' completed execution.
- Latest Successful Change**: Details for 'CICD-SmartTemplate #46' (COMPLETED) a day ago. Executed by 'd'. Duration: 6m 37s (09/06/2018 10:21 AM - 09/06/2018 10:29 AM).
- Recent Executions**: A table showing the status of recent executions across three environments: Build-Publish, Development, and Production.

Execution#/Stages	Build-Publish	Development	Production
#46	Completed	Completed	Completed
#45	Completed	Completed	Completed
#44	Completed	Completed	Completed
#43	Completed	Completed	Completed
#42	Completed	Completed	Completed
#41	Completed	Completed	Completed
#40	Completed	Completed	Completed
#39	Completed	Completed	Completed
#38	Completed	Completed	Completed
#37	Completed	Completed	Completed

Resultados

Você criou um pipeline que pode ser usado no projeto selecionado.

Também é possível exportar seu YAML de pipeline para importá-lo e reutilizá-lo em outros projetos.

Próximo passo

Saiba mais sobre os casos de uso que você pode querer aplicar no seu ambiente. Consulte [Capítulo 5 Tutoriais para usar o vRealize Automation Code Stream](#).

Como adicionar um projeto no vRealize Automation Code Stream

Você cria um projeto ao qual adiciona administradores e membros, para que os membros do projeto possam usar recursos como a criação de um pipeline e a adição de um endpoint. Para criar, excluir ou atualizar um projeto para uma equipe de desenvolvimento, você deve ser administrador do vRealize Automation Code Stream.

Um projeto deve existir para que você possa criar um pipeline. Ao criar um pipeline, você seleciona o projeto ao qual associá-lo, para que todas as informações do seu pipeline sejam agrupadas. As definições de endpoint e variáveis também dependem de um projeto existente.

Pré-requisitos

- Verifique se você tem a função de administrador do vRealize Automation Code Stream. Consulte [O que são funções no vRealize Automation Code Stream](#).

Se você não tiver a função de administrador do vRealize Automation Code Stream, mas for um administrador no vRealize Automation Cloud Assembly, poderá usar a interface de usuário do vRealize Automation Cloud Assembly para criar, atualizar ou excluir projetos. Consulte [Como adicionar um projeto para minha equipe de desenvolvimento do vRealize Automation Cloud Assembly](#)

- Se você estiver adicionando grupos do Active Directory a projetos, verifique se configurou esses para a sua organização. Consulte [Como habilitar grupos do Active Directory no vRealize Automation para projetos](#). Se os grupos não forem sincronizados, eles não ficarão disponíveis quando você tentar adicioná-los a um projeto.

Procedimentos

- 1 Selecione **Projetos** e clique em **Novo projeto**.
- 2 Digite o nome do projeto.
- 3 Clique em **Criar**.
- 4 Selecione o cartão para o projeto recém-criado e clique em **Abrir**.
- 5 Clique na guia **Usuários** e adicione usuários com funções atribuídas.
 - O administrador do projeto pode adicionar membros.
 - O membro do projeto com uma função de serviço pode usar serviços.

- O expectador do projeto pode ver projetos, mas não pode criá-los, atualizá-los ou excluí-los.

Para obter mais informações sobre funções de projeto, consulte [Como gerenciar o acesso do usuário e as aprovações no vRealize Automation Code Stream](#).

6 Clique em **Salvar**.

Próximo passo

Adicione endpoints e pipelines que usam o projeto. Consulte [Capítulo 6 Como conectar o vRealize Automation Code Stream aos endpoints](#) e [Capítulo 3 Criando e usando pipelines no vRealize Automation Code Stream](#).

Como gerenciar o acesso do usuário e as aprovações no vRealize Automation Code Stream

O vRealize Automation Code Stream fornece várias maneiras de garantir que os usuários tenham a autorização e o consentimento apropriados para trabalhar com pipelines que liberam seus aplicativos de software.

Cada membro em uma equipe tem uma função atribuída, que fornece permissões específicas nos pipelines, endpoints e painéis, e a capacidade de marcar recursos como restritos.

As operações e aprovações dos usuários permitem que você controle quando um pipeline é executado e quando ele deve ser interrompido para uma aprovação. Sua função determina se é possível retomar um pipeline e executar pipelines que incluem variáveis ou endpoints restritos.

Use variáveis secretas para ocultar e criptografar informações confidenciais. Use a variável restrita para strings, senhas e URLs que devem ser ocultas e criptografadas e para restringir o uso em execuções. Por exemplo, use uma variável secreta para uma senha ou URL. Você pode usar variáveis secretas e restritas em qualquer tipo de tarefa no seu pipeline.

O que são funções no vRealize Automation Code Stream

Dependendo da sua função no vRealize Automation Code Stream, é possível realizar determinadas ações e acessar determinadas áreas. Por exemplo, sua função pode permitir a criação, atualização e execução de pipelines. Ou, pode permitir apenas a exibição de pipelines.

Todas as ações, exceto as restritas significa que essa função tem permissão para executar ações de criação, leitura, atualização e exclusão em entidades, exceto para variáveis restritas e endpoints.

Tabela 2-1. Permissões de acesso em nível de Serviço e Projeto no vRealize Automation Code Stream

Funções do vRealize Automation Code Stream					
Níveis de acesso	Administrador do Code Stream	Desenvolvedor do Code Stream	Executor do Code Stream	Espectador do Code Stream	Usuário do Code Stream
Acesso em nível de serviço do vRealize Automation Code Stream	Todas as ações	Todas as ações, exceto as restritas	Ações de execução	Somente leitura	Nenhum
Acesso em nível de projeto: Administrador do Projeto	Todas as ações	Todas as ações	Todas as ações	Todas as ações	Todas as ações
Acesso em nível de projeto: Membro do Projeto	Todas as ações	Todas as ações, exceto as restritas	Todas as ações, exceto as restritas	Todas as ações, exceto as restritas	Todas as ações, exceto as restritas
Acesso em nível de projeto: Espectador do Projeto	Todas as ações	Todas as ações, exceto as restritas	Ações de execução	Somente leitura	Somente leitura

Os usuários que têm a função de administrador de projeto podem realizar todas as ações em projetos em que eles são um administrador de projeto.

Um administrador de projeto poderá criar, ler, atualizar e excluir pipelines, variáveis, endpoints, dashboards, gatilhos e iniciar um pipeline que inclua endpoints ou variáveis restritas se esses recursos estiverem no projeto em que o usuário é um Administrador de projeto.

Os usuários com a função de Visualizador de Serviços podem ver todas as informações disponíveis ao administrador. Eles não podem realizar nenhuma ação, a menos que um administrador os torne um administrador de projeto ou membro do projeto. Se o usuário for afiliado a um projeto, ele terá as permissões relacionadas à função. A função de visualizador de projeto não abrange as permissões da mesma forma que a função de administrador ou membro. Essa função é de somente leitura em todos os projetos.

Se você tiver permissões de leitura em um projeto, ainda poderá ver recursos restritos.

- Para ver endpoints restritos, que exibem um ícone de cadeado no cartão do endpoint, clique em **Configurar > Endpoints**.
- Para ver variáveis restritas e secretas, que exibem RESTRITO ou SECRETO na coluna **Tipo**, clique em **Configurar > Variáveis**.

Tabela 2-2. Recursos de funções de serviço do vRealize Automation Code Stream

Contexto da interface do usuário	Recursos	Função Administrador do Code Stream	Função Desenvolvedor do Code Stream	Função Executor do Code Stream	Função Expectador do Code Stream	Função Usuário do Code Stream
Pipelines						
	Exibir pipelines	Sim	Sim	Sim	Sim	
	Criar pipelines	Sim	Sim			
	Executar pipelines	Sim	Sim	Sim		
	Executar pipelines que incluem variáveis ou endpoints restritos	Sim				
	Atualizar pipelines	Sim	Sim			
	Excluir pipelines	Sim	Sim			
Execução de Pipeline						
	Exibir execuções de pipeline	Sim	Sim	Sim	Sim	
	Retomar, pausar e cancelar execuções de pipeline	Sim	Sim	Sim		
	Retomar pipelines que pararam para aprovação em recursos restritos	Sim				
Integrações Personalizadas						
	Criar integrações personalizadas	Sim	Sim			
	Ler integrações personalizadas	Sim	Sim	Sim	Sim	
	Atualizar integrações personalizadas	Sim	Sim			
Endpoints						
	Exibir execuções	Sim	Sim	Sim	Sim	
	Criar execuções	Sim	Sim			
	Atualizar execuções	Sim	Sim			
	Excluir execuções	Sim	Sim			

Tabela 2-2. Recursos de funções de serviço do vRealize Automation Code Stream (continuação)

Contexto da interface do usuário	Recursos	Função Administrador do Code Stream	Função Desenvolvedor do Code Stream	Função Executor do Code Stream	Função Expectador do Code Stream	Função Usuário do Code Stream
Marcar os recursos como restritos						
	Marcar uma variável ou um endpoint como restrito	Sim				
Painéis						
	Exibir painéis	Sim	Sim	Sim	Sim	
	Criar painéis	Sim	Sim			
	Atualizar painéis	Sim	Sim			
	Excluir painéis	Sim	Sim			

Funções e permissões personalizadas no vRealize Automation Code Stream

Você pode criar funções personalizadas no vRealize Automation Cloud Assembly que estendem privilégios para usuários que trabalham com pipelines. Ao criar uma função personalizada para pipelines do vRealize Automation Code Stream, você seleciona uma ou mais permissões de **Pipeline**.

Selecione o número mínimo de permissões de **Pipeline** necessárias para usuários que receberão essa função personalizada.

Quando um usuário é atribuído a um projeto e recebe uma função nesse projeto, e também recebe uma função personalizada que inclui uma ou mais permissões de **Pipeline**, ele pode executar todas as ações permitidas por essas permissões. Por exemplo, ele pode criar variáveis restritas, gerenciar pipelines restritos, criar e gerenciar integrações personalizadas e muito mais.

Tabela 2-3. Permissões de pipeline que você pode atribuir a funções personalizadas

Permissão de Pipeline	Administrador do Code Stream	Desenvolvedor do Code Stream	Executor do Code Stream	Espectador do Code Stream	Usuário do Code Stream	Administrador do projeto	Membro do projeto	Expectador de projeto
Gerenciar Pipelines	Sim	Sim				Sim	Sim	
Gerenciar Pipelines Restritos	Sim					Sim		

Tabela 2-3. Permissões de pipeline que você pode atribuir a funções personalizadas (continuação)

Permissão de Pipeline	Administrador do Code Stream	Desenvolvedor do Code Stream	Executor do Code Stream	Espectador do Code Stream	Usuário do Code Stream	Administrador do projeto	Membro do projeto	Espectador de projeto
Gerenciar integrações personalizadas	Sim	Sim						
Executar Pipelines	Sim	Sim	Sim			Sim	Sim	
Executar Pipelines Restritos	Sim					Sim		
Gerenciar execuções	Sim					Sim		
Leitura. Essa permissão não está visível.	Sim	Sim	Sim	Sim		Sim	Sim	Sim

Tabela 2-4. Como você pode usar permissões de Pipeline com funções personalizadas

Permissão	O que é possível fazer
Gerenciar Pipelines	<ul style="list-style-type: none"> ■ Criar, atualizar, excluir e clonar pipelines. ■ Lançar e cancelar o lançamento de pipelines para o VMware Service Broker. ■ Criar, atualizar e excluir endpoints. ■ Criar, atualizar e excluir variáveis regulares e secretas. ■ Criar, clonar, atualizar e excluir um ouvinte Gerrit. ■ Conectar e desconectar um ouvinte Gerrit. ■ Criar, clonar, atualizar, excluir um gatilho Gerrit. ■ Criar, atualizar e excluir um webhook Git. ■ Criar, atualizar e excluir um webhook Docker. ■ Usar modelos de pipeline inteligentes para criar pipelines. ■ Importar pipelines do YAML e exportá-los para o YAML. ■ Criar, atualizar e excluir painéis personalizados. ■ Pode ler todas as integrações personalizadas. ■ Pode ler todos os endpoints e variáveis restritos, mas não pode visualizar seus valores.
Gerenciar Pipelines Restritos	<ul style="list-style-type: none"> ■ Criar, atualizar e excluir endpoints. ■ Marcar endpoints como restritos, atualizar endpoints restritos e excluí-los. ■ Criar, atualizar e excluir variáveis regulares e secretas. ■ Criar, atualizar e excluir variáveis restritas. ■ Todas as permissões que você pode fazer com Gerenciar Pipelines.

Tabela 2-4. Como você pode usar permissões de Pipeline com funções personalizadas (continuação)

Permissão	O que é possível fazer
Gerenciar integrações personalizadas	<ul style="list-style-type: none"> ■ Criar e atualizar integrações personalizadas. ■ Controlar versões e lançamentos de integrações personalizadas. ■ Excluir e substituir versões de integração personalizadas. ■ Excluir Integrações personalizadas.
Executar Pipelines	<ul style="list-style-type: none"> ■ Executar pipelines. ■ Pausar, retomar e cancelar execuções de pipeline. ■ Repetir a execução de pipelines. ■ Reiniciar, repetir a execução e acionar manualmente um evento de gatilho Gerrit. ■ Pode aprovar uma operação de usuário e pode fazer aprovações em lote de operações de usuários.
Executar Pipelines Restritos	<ul style="list-style-type: none"> ■ Executar pipelines. ■ Pausar, retomar, cancelar e excluir execuções de pipeline. ■ Repetir a execução de pipelines. ■ Sincronizar uma execução de pipeline em execução. ■ Forçar a exclusão de uma execução de pipeline em execução. ■ Reiniciar, repetir a execução, excluir e acionar manualmente um evento de gatilho Gerrit. ■ Resolver itens restritos e continuar a execução do pipeline. ■ Alternar o contexto de usuário e continuar a execução do pipeline após a aprovação de uma tarefa de Operação do Usuário. ■ Todas as permissões que você pode fazer com Executar Pipelines.
Gerenciar execuções	<ul style="list-style-type: none"> ■ Executar pipelines. ■ Pausar, retomar, cancelar e excluir execuções de pipeline. ■ Repetir a execução de pipelines. ■ Reiniciar, repetir a execução, excluir e acionar manualmente um evento de gatilho Gerrit. ■ Todas as permissões que você pode fazer com Executar Pipelines.

Funções personalizadas podem incluir combinações de permissões. Essas permissões são organizadas em grupos de recursos que permitem aos usuários gerenciar ou executar pipelines, com e sem recursos restritos. Essas permissões representam todos os recursos que cada função pode desempenhar no vRealize Automation Code Stream.

Por exemplo, se você criar uma função personalizada e incluir a permissão chamada **Gerenciar Pipelines Restritos**, os usuários que tiverem a função de Desenvolvedor do vRealize Automation Code Stream poderão:

- Criar, atualizar e excluir endpoints.
- Marcar endpoints como restritos, atualizar endpoints restritos e excluí-los.
- Criar, atualizar e excluir variáveis regulares e secretas.
- Criar, atualizar e excluir variáveis restritas.

Tabela 2-5. Exemplos de combinações de permissões de Pipeline em funções personalizadas

Número de permissões atribuídas à função personalizada	Exemplos de permissões combinadas	Como usar essa combinação
Permissão única	Executar Pipelines	
Duas permissões	Gerenciar Pipelines e Executar Pipelines	
Três permissões	Gerenciar Pipelines e Execute Pipelines e Executar Pipelines Restritos	
	Gerenciar Pipelines e Gerenciar Integrações Personalizadas e Executar Pipelines Restritos	Essa combinação pode se aplicar a uma função de Desenvolvedor do vRealize Automation Code Stream, mas está limitada aos projetos dos quais o usuário é membro.
	Gerenciar Pipelines e Gerenciar Integrações Personalizadas e Gerenciar Execuções	Essa combinação pode se aplicar a um Administrador do vRealize Automation Code Stream, mas está limitado aos projetos dos quais o usuário é membro.
	Gerenciar Pipelines, Gerenciar Pipelines Restritos e Gerenciar Integrações Personalizadas	Com essa combinação, um usuário tem permissões totais e pode criar e excluir qualquer coisa no vRealize Automation Code Stream.

Com a função de administrador

Como administrador, você pode criar integrações, endpoints, variáveis, gatilhos, pipelines e dashboards personalizados.

Projetos permitem que pipelines acessem recursos de infraestrutura. Os administradores criam projetos para que os usuários possam agrupar pipelines, endpoints e painéis juntos. Em seguida, os usuários selecionam o projeto em seus pipelines. Cada projeto inclui um administrador e usuários com funções atribuídas.

Com a função de Administrador, é possível marcar endpoints e variáveis como recursos restritos e executar pipelines que usam recursos restritos. Se um usuário não administrativo executar o pipeline que inclui um endpoint ou variável restrita, o pipeline será interrompido na tarefa em que a variável restrita é usada, e um administrador deve retomar esse pipeline.

Como administrador, você também pode solicitar que os pipelines sejam publicados no vRealize Automation Service Broker.

Com a função de desenvolvedor

É possível trabalhar com pipelines, como administrador, exceto trabalhar com variáveis ou endpoints restritos.

Se você executar um pipeline que usa endpoints ou variáveis restritas, o pipeline apenas será executado até a tarefa que usa o recurso restrito. Em seguida, ele será interrompido, e um administrador do vRealize Automation Code Stream ou administrador de projeto deverá retomar o pipeline.

Se você tiver a função Usuário

Você pode acessar o vRealize Automation Code Stream, mas não tem nenhum dos privilégios que as outras funções fornecem.

Se você tiver a função Expectador

Você pode ver os mesmos recursos visíveis para um administrador, como pipelines, endpoints, execuções de pipeline, dashboards, integrações personalizadas e gatilhos, mas não pode criá-los, atualizá-los ou excluí-los. Para realizar ações, a função de Espectador também deve receber a função de administrador de projeto ou membro do projeto.

Os usuários que têm a função de Espectador podem ver projetos. Eles também podem ver endpoints restritos e variáveis restritas, mas não podem ver as informações detalhadas sobre eles.

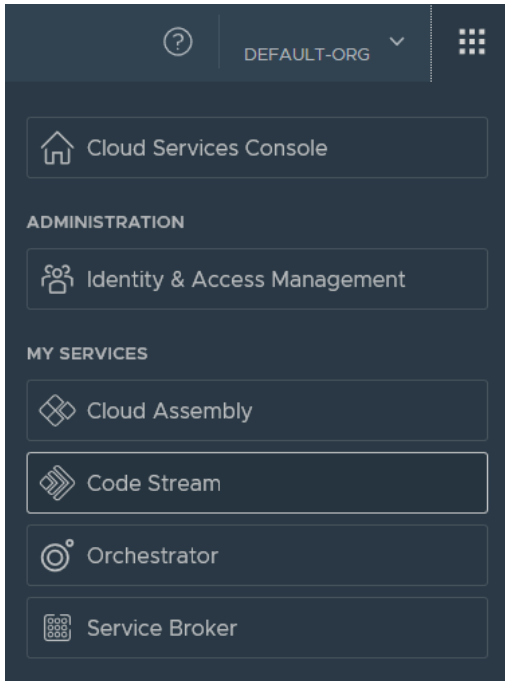
Se você tiver a função Executor

Você poderá executar pipelines e realizar ações em tarefas de operações do usuário. Também poderá retomar, pausar e cancelar execuções de pipeline. No entanto, não poderá modificar pipelines.

Como atribuir e atualizar funções

Para atribuir e atualizar funções de outros usuários, é necessário ser um administrador.

- 1 Para ver os usuários ativos e suas funções, no vRealize Automation, clique nos nove pontos no canto superior direito.
- 2 Clique em **Gerenciamento de Identidades e Acessos**.



- 3 Para exibir nomes de usuário e funções, clique em **Usuários Ativos**.



- 4 Para adicionar funções para um usuário ou alterar suas funções, clique na caixa de seleção ao lado do nome do usuário e depois clique em **Editar Funções**.
- 5 Ao adicionar ou alterar funções de usuário, você também pode adicionar acesso a serviços.
- 6 Para salvar as alterações, clique em **Salvar**.

O que são operações do usuário e aprovações no vRealize Automation Code Stream

A área Operações do Usuário exibe as execuções de pipeline que precisam de aprovação. O usuário designado como aprovador pode aprovar ou rejeitar a execução do pipeline.

Ao criar um pipeline, talvez seja necessário adicionar uma aprovação a um pipeline se:

- Um membro da equipe precisar revisar seu código.
- Outro usuário precisar confirmar um artefato de compilação.
- Você deve garantir que todos os testes estejam concluídos.

- Uma tarefa usa um recurso que um administrador marcou como restrito, e essa tarefa requer aprovação.
- O pipeline liberará o software para produção.

O aprovador necessário para uma tarefa de pipeline deve ter permissão e experiência para determinar se ela deve ou não ser aprovada.

Ao adicionar uma tarefa de Operação do Usuário, você pode definir o tempo limite de expiração em dias, horas ou minutos. Por exemplo, talvez seja necessário que o usuário responsável aprove o pipeline em 30 minutos. Se ele não o aprovar o pipeline em 30 minutos, este falhará conforme esperado.

Depois que o usuário necessário aprovar a tarefa de operação do usuário:

- A execução do pipeline pendente poderá continuar.
- Quando o pipeline continuar, todas as solicitações pendentes anteriores para aprovação dessa mesma tarefa de operação do usuário serão canceladas.

User Operations

GUIDED SETUP

Active Items
Inactive Items

✓ APPROVE
✗ REJECT

<input type="checkbox"/>	Index#	Execution	Summary	Requested By	Request Date	Approvers
<input type="checkbox"/>	> c07b12	Demo2-Jenkins-K8s#7	Testing	fritz	Nov 13, 2019, 11:32:31 AM	f...om
<input type="checkbox"/>	> a0a990	Demo2-Jenkins-K8s#6	Testing	fritz	Nov 11, 2019, 1:34:11 PM	k...om, f...m
<input checked="" type="checkbox"/>	▼	User Operation #8f1728 <div> <div>Request Details</div> <div> <div>Execution</div> <div>Demo-Jenkins-K8s #5</div> </div> <div> <div>Summary</div> <div>Testing</div> </div> <div> <div>Approvers</div> <div>k...om, f...com</div> </div> <div> <div>Requested By</div> <div>fritz</div> </div> <div> <div>Requested On</div> <div>Nov 11, 2019, 1:22:21 PM</div> </div> <div> <div>Expires On</div> <div>Nov 14, 2019, 1:22:21 PM</div> </div> </div>				

✓ 1
Items per page 20
1 - 7 of 7 items

Na área Operações do Usuário, os itens a serem aprovados ou rejeitados aparecem como itens ativos ou inativos. Cada item é mapeado para uma tarefa de operação do usuário em um pipeline.

- **Itens Ativos** aguardam o aprovador revisar a tarefa e aprová-la ou rejeitá-la. Se você for um usuário que está na lista de aprovadores, poderá expandir a linha de operação do usuário e clicar em **Aceitar** ou **Rejeitar**.
- **Itens Inativos** foram aprovados ou rejeitados. Se um usuário tiver rejeitado a operação do usuário ou se a aprovação na tarefa tiver atingido o tempo limite, ela não poderá mais ser aprovada.

O Índice# é uma cadeia de caracteres alfanumérica exclusiva de seis caracteres que você pode usar como filtro para procurar uma determinada aprovação.

Aprovações de pipeline também aparecem na área **Execuções**.

- Os pipelines que estão aguardando aprovação indicam seu status como aguardando.
- Outros status são: em fila, concluído e com falha.
- Se o pipeline estiver em estado de espera, o aprovador necessário deverá aprovar a tarefa de pipeline.

Criando e usando pipelines no vRealize Automation Code Stream

3

Você pode usar o vRealize Automation Code Stream para modelar o processo de compilação, teste e implantação. Com o vRealize Automation Code Stream, configure a infraestrutura que oferece suporte ao seu ciclo de liberação e crie pipelines que modelam suas atividades de liberação de software. O vRealize Automation Code Stream entrega seu software desde o código de desenvolvimento, passando pela fase de testes, e o implanta nas suas instâncias de produção.

Cada pipeline inclui estágios e tarefas. Os estágios representam suas fases de desenvolvimento, e as tarefas executam as ações necessárias para entregar o aplicativo de software pelos estágios.

O que são pipelines no vRealize Automation Code Stream

Um pipeline é um modelo de integração e entrega contínua do seu processo de liberação do software. Ele libera seu software desde o código-fonte, passando pela fase de testes até a produção. Inclui uma sequência de estágios com tarefas que representam as atividades no ciclo de liberação do software. Seu aplicativo de software flui de um estágio para o outro por meio do pipeline.

Adicione endpoints para que as tarefas no pipeline possam se conectar a fontes de dados, repositórios ou sistemas de notificação.

Como criar pipelines

Você pode criar um pipeline começando com uma tela em branco, usando um modelo de pipeline inteligente ou importando o código YAML.

- Use a tela em branco. Para obter um exemplo, consulte [Planejando uma compilação nativa de CI/CD no vRealize Automation Code Stream antes de adicionar tarefas manualmente](#).
- Use um modelo de pipeline inteligente. Para obter um exemplo, consulte [Capítulo 4 Planejamento para compilar, integrar e entregar seu código de forma nativa no vRealize Automation Code Stream](#).
- Importe o código YAML. Clique em **Pipelines > Importar**. Na caixa de diálogo **Importar**, selecione o arquivo YAML ou insira o código YAML e clique em **Importar**.

Ao usar a tela em branco para criar um pipeline, você adiciona estágios, tarefas e aprovações. O pipeline automatiza o processo de compilação, teste, implantação e liberação do seu aplicativo. As tarefas em cada estágio executam ações que compilam, testam e liberam seu código em cada estágio.

Tabela 3-1. Exemplo de estágios e usos de pipelines

Exemplo de estágio...	Exemplos do que é possível fazer...
Desenvolvimento	<p>Em um estágio de desenvolvimento, é possível provisionar uma máquina, recuperar um artefato, adicionar uma tarefa de compilação para criar um host do Docker a ser usado para a integração contínua do seu código e muito mais.</p> <p>Por exemplo:</p> <ul style="list-style-type: none"> ■ Para planejar e criar uma compilação de integração contínua (CI), que entrega seu código usando o recurso de compilação nativo no vRealize Automation Code Stream, consulte Como planejar uma compilação nativa de CI no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente.
Teste	<p>Em um estágio de teste, é possível adicionar uma tarefa de Jenkins para testar o aplicativo de software e incluir ferramentas de teste pós-processamento, como JUnit, JaCoCo e muito mais.</p> <p>Por exemplo:</p> <ul style="list-style-type: none"> ■ Integre o vRealize Automation Code Stream ao Jenkins e execute um trabalho do Jenkins no seu pipeline, que compila e testa seu código-fonte. Consulte Como integrar o vRealize Automation Code Stream ao Jenkins. ■ Crie scripts personalizados que estendem a capacidade do vRealize Automation Code Stream de se integrar às suas próprias ferramentas de compilação, teste e implantação. Consulte Como integrar minhas próprias ferramentas de compilação, teste e implantação com o vRealize Automation Code Stream. ■ Rastreie tendências no pós-processamento para um pipeline de integração contínua (CI). Consulte Como rastrear os principais indicadores de desempenho do pipeline no vRealize Automation Code Stream.
Produção	<p>Em um estágio de produção, você pode fazer a integração com modelo de nuvem no vRealize Automation Cloud Assembly para provisionar sua infraestrutura, implantar seu software em um cluster do Kubernetes e muito mais.</p> <p>Por exemplo:</p> <ul style="list-style-type: none"> ■ Para ver os estágios de exemplo para desenvolvimento e produção, que podem implantar seu aplicativo de software no seu próprio modelo de implantação Azul-Verde, consulte Como implantar meu aplicativo no vRealize Automation Code Stream na implantação Azul-Verde. ■ Para integrar um modelo de nuvem ao seu pipeline, consulte Como automatizar a liberação de um aplicativo implantado a partir de um modelo de nuvem YAML no vRealize Automation Code Stream. Você também pode adicionar uma tarefa de implantação que executa um script para implantar o aplicativo. ■ Para automatizar a implantação dos seus aplicativos de software em um cluster do Kubernetes, consulte Como automatizar a liberação de um aplicativo no vRealize Automation Code Stream para um cluster do Kubernetes. ■ Para integrar o código ao seu pipeline e implantar a imagem de compilação, consulte Como integrar continuamente o código do meu repositório do GitHub ou GitLab ao pipeline no vRealize Automation Code Stream.

Você pode exportar seu pipeline como um arquivo YAML. Clique em **Pipelines**, clique em um cartão de pipeline e, em seguida, clique em **Ações > Exportar**.

Aprovando pipelines

Você pode obter uma aprovação de outro membro da equipe em pontos específicos no pipeline.

- Para exigir a aprovação em um pipeline, incluindo uma tarefa de operação do usuário em um pipeline, consulte [Como executar um pipeline e ver os resultados](#). Essa tarefa envia uma notificação por e-mail ao usuário que deve revisá-la. O revisor deve aprovar ou rejeitar a aprovação antes que o pipeline possa continuar sendo executado. Se a tarefa Operação do Usuário tiver um tempo limite de expiração definido em dias, horas ou minutos, o usuário necessário deverá aprovar o pipeline antes da expiração da tarefa. Caso contrário, o pipeline falhará conforme esperado.
- Em qualquer estágio de um pipeline, se uma tarefa ou estágio falhar, será possível especificar que o vRealize Automation Code Stream crie um tíquete Jira. Consulte [Como criar um tíquete do Jira no vRealize Automation Code Stream quando uma tarefa de pipeline falhar](#).

Como disparar pipelines

Os pipelines podem ser disparados quando os desenvolvedores fazem o check-in ou a revisão do código ou quando um artefato de compilação é criado ou atualizado.

- Para integrar o vRealize Automation Code Stream ao ciclo de vida do Git e disparar um pipeline quando os desenvolvedores atualizarem seu código, use o gatilho do Git. Consulte [Como usar o gatilho Git no vRealize Automation Code Stream para executar um pipeline](#).
- Para integrar o vRealize Automation Code Stream ao ciclo de vida de revisão de código do Gerrit e disparar um pipeline em revisões de código, use o gatilho do Gerrit. Consulte [Como usar o gatilho Gerrit no vRealize Automation Code Stream para executar um pipeline](#).
- Para disparar um pipeline quando um artefato de compilação do Docker for criado ou atualizado, use o gatilho do Docker. Consulte [Como usar o gatilho do Docker no vRealize Automation Code Stream para executar um pipeline de entrega contínua](#).

Para obter mais informações sobre os gatilhos para os quais o vRealize Automation Code Stream oferece suporte, consulte [Capítulo 7 Disparando pipelines no vRealize Automation Code Stream](#).

Este capítulo inclui os seguintes tópicos:

- [Como executar um pipeline e ver os resultados](#)
- [Quais tipos de tarefa estão disponíveis no vRealize Automation Code Stream](#)
- [Como usar associações de variáveis em pipelines do vRealize Automation Code Stream](#)
- [Como usar associações de variáveis em uma tarefa de condição para executar ou parar um pipeline no vRealize Automation Code Stream](#)
- [Que variáveis e expressões eu posso usar ao associar tarefas de pipeline no vRealize Automation Code Stream](#)
- [Como enviar notificações sobre meu pipeline no vRealize Automation Code Stream](#)

- [Como criar um tíquete do Jira no vRealize Automation Code Stream quando uma tarefa de pipeline falhar](#)
- [Como faço para reverter minha implantação no vRealize Automation Code Stream](#)

Como executar um pipeline e ver os resultados

Você pode executar um pipeline a partir do cartão de pipeline, no modo de edição de pipeline e a partir da execução do pipeline. Você também pode usar os gatilhos disponíveis para que o vRealize Automation Code Stream execute um pipeline quando determinados eventos ocorrerem.

Quando todos os estágios e tarefas do pipeline forem válidos, ele estará pronto para ser liberado, executado ou disparado.

Para executar ou disparar seu pipeline usando o vRealize Automation Code Stream, você pode ativar e executar o pipeline do cartão de pipeline ou enquanto estiver no pipeline. Em seguida, você pode visualizar a execução do pipeline para confirmar se o pipeline criou, testou e implantou seu código.

Quando uma execução de pipeline estiver em andamento, você poderá excluir a execução se for um administrador ou um usuário não administrador.

- Administrador: para excluir a execução de um pipeline quando estiver em execução, clique em **Execuções**. Na execução a ser excluída, clique em **Ações > Excluir**.
- Usuário não administrador: para excluir uma execução de pipeline em execução, clique em **Execuções** e em **Alt Shift d**.

Quando uma execução de pipeline estiver em andamento e parecer travada, um administrador poderá atualizá-la na página de Execuções ou na página de Detalhes da execução.

- Página Execuções: clique em **Execuções**. Na execução a ser atualizada, clique em **Ações > Sincronizar**.
- Página Detalhes da execução: clique em **Execuções**, no link para ver os detalhes da execução e em **Ações > Sincronizar**.

Para executar um pipeline quando ocorrerem eventos específicos, use os gatilhos.

- O gatilho Git poderá executar um pipeline quando os desenvolvedores atualizarem o código.
- O gatilho Gerrit poderá executar um pipeline quando ocorrerem revisões de código.
- O gatilho Docker poderá executar um pipeline quando um artefato for criado em um registro de Docker.
- O comando `curl` poderá fazer com que o Jenkins execute um pipeline após a conclusão de uma compilação do Jenkins.

Para obter mais informações sobre como usar os gatilhos, consulte [Capítulo 7 Disparando pipelines no vRealize Automation Code Stream](#).

O procedimento a seguir mostra como executar um pipeline do cartão de pipeline, visualizar execuções, ver os detalhes da execução e usar as ações. Ele também mostra como liberar um pipeline para que você possa adicioná-lo ao vRealize Automation Service Broker.

Pré-requisitos

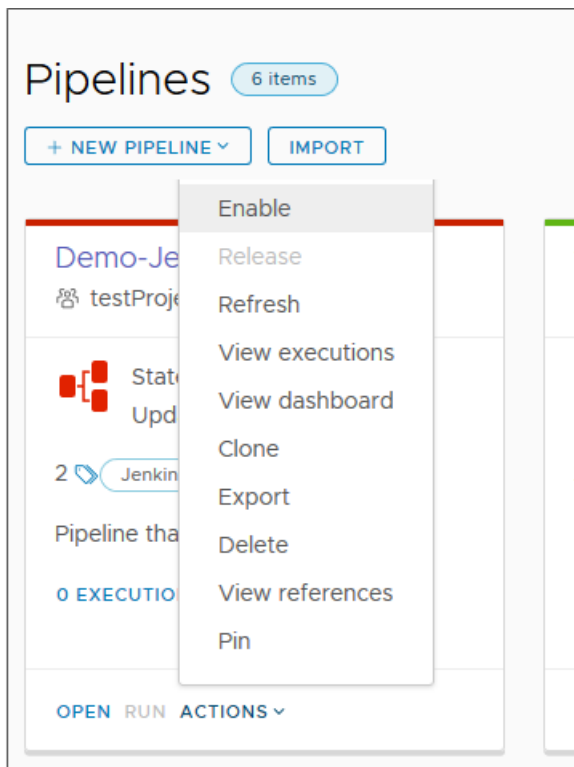
- Verifique se um ou mais pipelines foram criados. Consulte os exemplos em [Capítulo 5 Tutoriais para usar o vRealize Automation Code Stream](#).

Procedimentos

- 1 Habilite seu pipeline.

Para executar ou liberar um pipeline, você deve habilitá-lo primeiro.

- a Clique em **Pipelines**.
- b No seu cartão de pipeline, clique em **Ações > Ativar**.



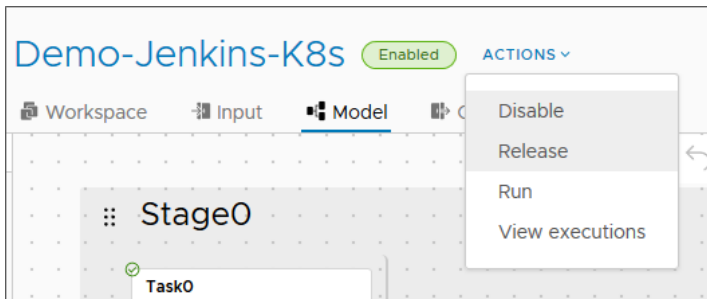
Você também pode habilitar seu pipeline enquanto estiver nele. Se o seu pipeline já estiver habilitado, o comando **Executar** estará ativo, e o menu **Ações** exibirá **Desativar**.

2 (Opcional) Liberte seu pipeline.

Se quiser disponibilizar o pipeline como um item de catálogo no vRealize Automation Service Broker, deverá liberá-lo no vRealize Automation Code Stream.

- a Clique em **Pipelines**.
- b No seu cartão de pipeline, clique em **Ações > Liberar**.

Você também pode liberar seu pipeline enquanto estiver nele.



Depois de liberar o pipeline, abra o vRealize Automation Service Broker para adicionar o pipeline como um item de catálogo e executá-lo. Consulte [Adicionar pipelines do vRealize Automation Code Stream ao catálogo do vRealize Automation Service Broker](#).

Observação Se o pipeline exigir mais de 120 minutos para ser executado, forneça um tempo de execução aproximado como um valor de tempo limite da solicitação. Para definir ou revisar o tempo limite da solicitação para um projeto, abra o vRealize Automation Service Broker como administrador e selecione **Infraestrutura > Projetos**. Clique no nome do projeto e depois em **Provisionando**.

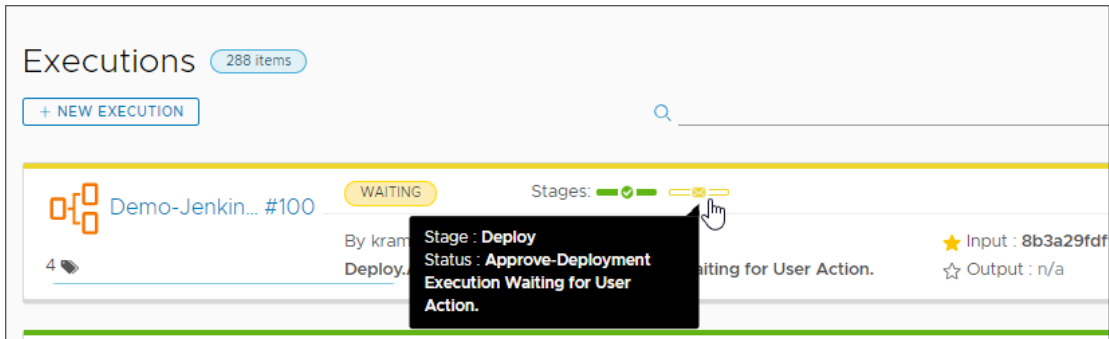
Se o valor de tempo limite da solicitação não estiver definido, uma execução que exige mais de 120 minutos para ser executada aparece como falha com um erro de solicitação de tempo limite de retorno de chamada. No entanto, a execução do pipeline não é afetada.

- 3 No cartão de pipeline, clique em **Executar**.
- 4 Para exibir o pipeline enquanto ele é executado, clique em **Execuções**.

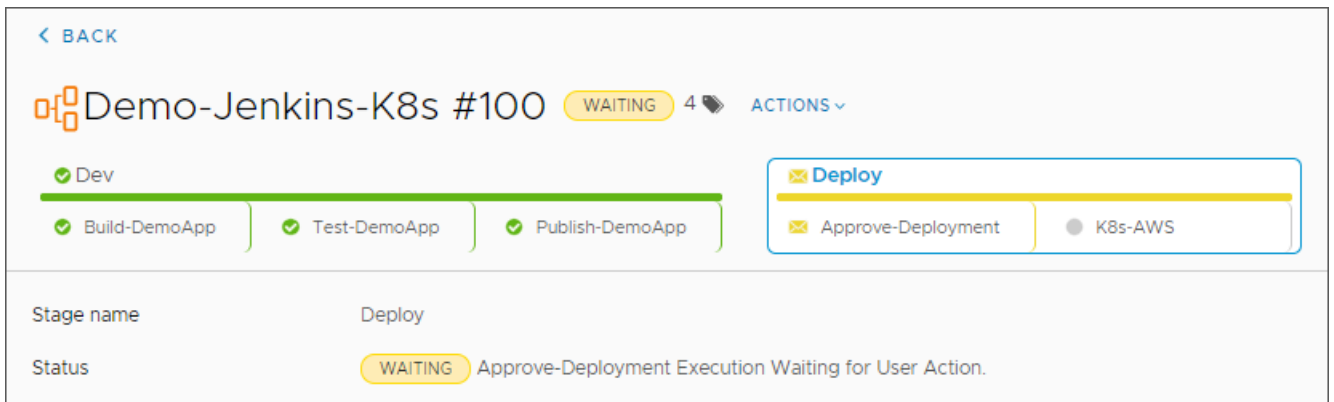
O pipeline executa cada estágio em sequência, e a execução do pipeline exibe um ícone de status para cada estágio. Se o pipeline incluir uma tarefa de operação do usuário, um usuário deverá aprovar a tarefa para que o pipeline continue a ser executado. Quando uma tarefa de operação do usuário é usada, o pipeline para de ser executado e aguarda o usuário necessário aprovar a tarefa.

Por exemplo, você pode usar a tarefa de operação do usuário para aprovar a implantação do código em um ambiente de produção.

Se a tarefa Operação do Usuário tiver um tempo limite de expiração definido em dias, horas ou minutos, o usuário necessário deverá aprovar o pipeline antes da expiração da tarefa. Caso contrário, o pipeline falhará conforme esperado.



- 5 Para ver o estágio do pipeline que está aguardando a aprovação do usuário, clique no ícone de status do estágio.



- 6 Para ver os detalhes da tarefa, clique na tarefa.

Após o usuário necessário aprovar a tarefa, um usuário que tiver a função apropriada deverá retomar o pipeline. Para conhecer as funções necessárias, consulte [Como gerenciar o acesso do usuário e as aprovações no vRealize Automation Code Stream](#).

Se uma execução falhar, você deverá fazer a triagem e corrigir a causa da falha. Em seguida, vá até a execução e clique em **Ações > Executar novamente**.

Você poderá retomar as execuções de pipelines primários e execuções aninhadas.

< BACK

Demo-Jenkins-K8s #100 WAITING 4 ACTIONS ▾

✓ Dev

✓ Build-DemoApp } ✓ Test-DemoApp } ✓ Publish-DemoApp }

✖ Deploy

✖ [Approve-Deployment](#)

Task name Approve-Deployment

Type UserOperation

Status WAITING Execution Waiting for User Action.

Execute Task ☒ Always ☐ On Condition

Input ▾

Summary Demo-Jenkins-K8s is pending deployment for your approval

Description Demo-Jenkins-K8s is pending deployment for your approval

Users

- 7 Na execução do pipeline, você pode clicar em **Ações** para visualizar o pipeline e selecionar uma ação, como **Pausar**, **Cancelar** e muito mais. Quando uma execução de pipeline estiver em andamento, se você for um administrador, poderá excluir ou sincronizar a execução do pipeline. Se você for um usuário não administrador, poderá excluir um pipeline em execução.
- 8 Para navegar facilmente entre as execuções e ver os detalhes de uma tarefa, clique em **Execuções** e clique em uma execução de pipeline. Em seguida, clique na guia no topo da execução e selecione uma execução.

Executions for Demo2-Jenkins-K8s

✖ #6 ✖ #5 ✖ #4 ✖ #3 ✖ #2 ✖ #1

✖ Stage0

✖ Task0

Resultados

Parabéns! Você executou um pipeline, examinou a execução do pipeline e visualizou uma tarefa de operação do usuário que exigiu aprovação para que o pipeline continuasse a ser executado. Você também usou o menu **Ações** na execução do pipeline para retornar ao modelo de pipeline, para poder fazer quaisquer alterações necessárias.

Próximo passo

Para saber mais sobre como usar o vRealize Automation Code Stream para automatizar o ciclo de liberação do software, consulte [Capítulo 5 Tutoriais para usar o vRealize Automation Code Stream](#).

Quais tipos de tarefa estão disponíveis no vRealize Automation Code Stream

É possível configurar seu pipeline para executar determinadas ações ao adicionar tipos de tarefa específicos a ela. Cada tipo de tarefa integra-se a outro aplicativo para permitir que o seu pipeline realize o que você o preparou para fornecer.

Caso precise receber artefatos de um repositório para implantação, executar um script remoto ou exigir a aprovação de um membro da equipe para que seu pipeline seja executado, o vRealize Automation Code Stream tem o tipo de tarefa para você!

Antes de usar o tipo de tarefa no seu pipeline, verifique se o endpoint correspondente está disponível.

Tabela 3-2. Obtenha uma aprovação ou defina um ponto de decisão

Tipo de tarefa...	O que ele faz...	Exemplos e detalhes...
Operação do Usuário	Permite uma aprovação obrigatória que controla quando um pipeline é executado e quando deve parar para uma aprovação.	Consulte Como executar um pipeline e ver os resultados , e Como gerenciar o acesso do usuário e as aprovações no vRealize Automation Code Stream .
Condição	Adiciona um ponto de decisão, que determina se o pipeline deve continuar a ser executado ou para, com base nas expressões de condição. Quando a condição for verdadeira, o pipeline executará tarefas sucessivas. Quando for falso, o pipeline para.	Consulte Como usar associações de variáveis em uma tarefa de condição para executar ou parar um pipeline no vRealize Automation Code Stream .

Tabela 3-3. Automatize a integração contínua e a implantação

Tipo de tarefa...	O que ele faz...	Exemplos e detalhes...
Modelo de nuvem	Implanta um modelo de nuvem de automação a partir do GitHub e provisiona um aplicativo, automatiza a integração contínua e entrega contínua (CICD) desse modelo de nuvem para sua implantação.	<p>Consulte Como automatizar a liberação de um aplicativo implantado a partir de um modelo de nuvem YAML no vRealize Automation Code Stream.</p> <p>Quando você selecionar Criar ou Atualizar na tarefa de modelo de nuvem e depois selecionar Modelo de Nuvem e Versão, os parâmetros do modelo de nuvem serão exibidos. Você pode adicionar estes elementos, que acomodam associações de variáveis, às áreas de texto de entrada na tarefa de modelo de nuvem:</p> <ul style="list-style-type: none"> ■ Inteiro ■ Cadeia de caracteres de enumeração ■ Boolean ■ Variável de matriz <p>Ao usar a associação de variáveis nos campos de entrada, esteja ciente dessas exceções. Para enumerações, você deve selecionar um valor de enumeração de um conjunto fixo. Para Valores boolean, você deve inserir o valor na área de texto de entrada.</p> <p>O parâmetro de modelo de nuvem aparece na tarefa de modelo de nuvem quando um modelo de nuvem no vRealize Automation Cloud Assembly inclui variáveis de entrada. Por exemplo, se um modelo de nuvem tiver um tipo de entrada de <code>Integer</code>, você poderá inserir o número inteiro diretamente ou como uma variável usando a associação de variáveis.</p>
IC	Permite a integração contínua do código ao pipeline, recebendo uma imagem de compilação do Docker de um endpoint de registro e implantando-a em um cluster do Kubernetes.	Consulte Como planejar uma compilação nativa de CICD no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente .
Personalizado	Integra o vRealize Automation Code Stream com suas próprias ferramentas de compilação, teste e implantação.	Consulte Como integrar minhas próprias ferramentas de compilação, teste e implantação com o vRealize Automation Code Stream .

Tabela 3-3. Automatize a integração contínua e a implantação (continuação)

Tipo de tarefa...	O que ele faz...	Exemplos e detalhes...
Kubernetes	Automatize a implantação de seus aplicativos de software para clusters do Kubernetes no AWS.	Consulte Como automatizar a liberação de um aplicativo no vRealize Automation Code Stream para um cluster do Kubernetes .
Pipeline	Aninha um pipeline em um pipeline principal. Quando um pipeline é aninhado, ele se comporta como uma tarefa no pipeline principal. Na guia Tarefa do pipeline principal, é possível navegar facilmente até o pipeline aninhado clicando no link dele. O pipeline aninhado é aberto em uma nova guia do navegador.	Para encontrar pipelines aninhados em Execuções , digite nested na área de pesquisa.

Tabela 3-4. Integre aplicativos de desenvolvimento, teste e implantação

Tipo de tarefa...	O que ele faz...	Exemplos e detalhes...
Bamboo	Interage com um servidor de integração contínua (CI) Bamboo, que continuamente compila, testa e integra o software na preparação para implantação, além de gerar códigos de disparo quando os desenvolvedores confirmam as alterações. Ele expõe os locais de artefato que a compilação de Bamboo produz para que a tarefa possa produzir os parâmetros de outras tarefas a serem usados para compilação e implantação.	Conecte-se a um endpoint do servidor Bamboo e inicie um plano de compilação de Bamboo a partir de seu pipeline.
Jenkins	Dispara os trabalhos do Jenkins que compilam e testam o código-fonte, executa casos de teste e pode usar scripts personalizados.	Consulte Como integrar o vRealize Automation Code Stream ao Jenkins .
TFS	Permite que você conecte seu pipeline ao Team Foundation Server para gerenciar e invocar projetos de compilação, incluindo trabalhos configurados que compilam e testam seu código.	O vRealize Automation Code Stream oferece suporte ao Team Foundation Server 2013 e 2015.
vRO	Estende o recurso do vRealize Automation Code Stream executando fluxos de trabalho predefinidos ou personalizados no vRealize Orchestrator.	Consulte Como integrar o vRealize Automation Code Stream ao vRealize Orchestrator .

Tabela 3-5. Integre outros aplicativos por meio de uma API

Tipo de tarefa...	O que ele faz...	Exemplos e detalhes...
REST	Integra o vRealize Automation Code Stream a outros aplicativos que usam uma REST API para que você possa desenvolver e entregar continuamente os aplicativos de software que interagem entre si.	Consulte Como usar uma REST API para integrar o vRealize Automation Code Stream a outros aplicativos .
Sondagem	Invoca uma REST API e a sonda até que a tarefa do pipeline atenda aos critérios de saída e seja concluída.	Consulte Como usar uma REST API para integrar o vRealize Automation Code Stream a outros aplicativos .

Tabela 3-6. Execute scripts remotos e definidos pelo usuário

Tipo de tarefa...	O que ele faz...	Exemplos e detalhes...
PowerShell	<p>Permite que o tipo de tarefa de script do PowerShell execute comandos de script em um host remoto. Por exemplo, um script pode automatizar tarefas de teste e executar tipos administrativos de comandos.</p> <p>O script pode ser remoto ou definido pelo usuário. Ele pode se conectar por HTTP ou HTTPS e pode usar o TLS.</p> <p>O serviço chamado winrm deve ser configurado no host do Windows, e o winrm deve estar configurado para MaxShellsPerUser e MaxMemoryPerShellMB.</p>	<p>Quando você configura MaxShellsPerUser e MaxMemoryPerShellMB:</p> <ul style="list-style-type: none"> ■ O valor aceitável para MaxShellsPerUser é 500 para 50 pipelines simultâneos, com 5 tarefas de PowerShell para cada pipeline. Para definir o valor, execute: <code>winrm set winrm/config/winrs '@{MaxShellsPerUser="500"}'</code> ■ O valor de memória aceitável para MaxMemoryPerShellMB é 2048. Para definir o valor, execute: <code>winrm set winrm/config/winrs '@{MaxMemoryPerShellMB="2048"}'</code> <p>O script escreve a saída em um arquivo de resposta que outro pipeline pode consumir.</p>
SSH	<p>Permite que o tipo de tarefa de script do shell de bash execute comandos de script em um host remoto. Por exemplo, um script pode automatizar tarefas de teste e executar tipos administrativos de comandos.</p> <p>O script pode ser remoto ou definido pelo usuário. Ele pode se conectar por HTTP ou HTTPS e requer uma chave privada ou senha.</p> <p>O serviço SSH deve ser configurado no host do Linux, e a configuração de SSHD de MaxSessions deve ser definida para 50.</p>	<p>O script pode ser remoto ou definido pelo usuário. Por exemplo, um script pode ser semelhante a:</p> <pre>message="Hello World" echo \$message</pre> <p>O script escreve a saída em um arquivo de resposta que outro pipeline pode consumir.</p>

Como usar associações de variáveis em pipelines do vRealize Automation Code Stream

A associação de uma tarefa de pipeline significa que você cria uma dependência para a tarefa quando esse pipeline é executado. É possível criar uma associação para uma tarefa de pipeline de várias maneiras. Você pode associar uma tarefa a outra, associá-la a uma variável e uma expressão ou associá-la a uma condição.

Como aplicar associações de cifrão a variáveis de modelo de nuvem em uma tarefa de modelo de nuvem

É possível aplicar associações de cifrão a variáveis de modelo de nuvem em uma tarefa de modelo de nuvem de pipeline do vRealize Automation Code Stream. A maneira como você modifica as variáveis no vRealize Automation Code Stream depende de como as propriedades dessas variáveis estão codificadas no modelo de nuvem.

Se você precisar usar associações de cifrão em uma tarefa de modelo de nuvem, mas a versão atual do modelo de nuvem que você está usando na tarefa de modelo de nuvem não permitir isso, modifique o modelo de nuvem no vRealize Automation Cloud Assembly e implante uma nova versão. Em seguida, use a nova versão do modelo de nuvem na sua tarefa de modelo de nuvem e adicione as associações de cifrão onde necessário.

Para aplicar associações de cifrão aos tipos de propriedade fornecidos pelo modelo de nuvem do vRealize Automation Cloud Assembly, você deve ter as permissões corretas.

- Você deve ter a mesma função que a pessoa que criou a implantação do modelo de nuvem no vRealize Automation Cloud Assembly.
- A pessoa que modela o pipeline e a pessoa que executa o pipeline podem ser dois usuários diferentes e pode ter funções diferentes.
- Se um desenvolvedor tiver a função Executor do vRealize Automation Code Stream e modelar o pipeline, o desenvolvedor também deverá ter a mesma função do vRealize Automation Cloud Assembly que a pessoa que implantou o modelo de nuvem. Por exemplo, a função necessária pode ser Administrador do vRealize Automation Cloud Assembly.
- Somente a pessoa que modelar o pipeline terá permissão para criar o pipeline e criar a implantação.

Para usar um token de API na tarefa de modelo de nuvem:

- A pessoa que modelar o pipeline poderá fornecer um token de API para outro usuário que tiver a função Executor do vRealize Automation Code Stream. Dessa maneira, quando o Executor executar o pipeline, ele usará o token de API e as credenciais criadas por esse token de API.
- Quando um usuário insere um token de API na tarefa de modelo de nuvem, ele cria as credenciais necessárias para que o pipeline seja executado.
- Para criptografar o valor do token de API, clique em **Criar Variável**.
- Se você não criar uma variável para o token de API e usá-lo na tarefa de modelo de nuvem, o valor desse token de API aparecerá em texto sem formatação.

Para aplicar associações de cifrão a variáveis de modelo de nuvem em uma tarefa de modelo de nuvem, siga estas etapas.

Comece com um modelo de nuvem que tenha propriedades de variáveis de entrada definidas, como `integerVar`, `stringVar`, `flavorVar`, `BooleanVar`, `objectVar` e `arrayVar`. As propriedades da imagem são definidas na seção `resources`. As propriedades no código do modelo de nuvem podem ser semelhantes ao seguinte:

```
formatVersion: 1
inputs:
  integerVar:
    type: integer
    encrypted: false
    default: 1
  stringVar:
    type: string
    encrypted: false
    default: bkix
  flavorVar:
    type: string
    encrypted: false
    default: medium
  BooleanVar:
    type: boolean
    encrypted: false
    default: true
  objectVar:
    type: object
    encrypted: false
    default:
      bkix2: bkix2
  arrayVar:
    type: array
    encrypted: false
    default:
      - '1'
      - '2'
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    properties:
      image: ubuntu
      flavor: micro
      count: '${input.integerVar}'
```

Você pode usar variáveis de cifrão (\$) para `image` e `flavor`. Por exemplo:

```
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    properties:
      input: '${input.image}'
      flavor: '${input.flavor}'
```

Para usar um modelo de nuvem em um pipeline do vRealize Automation Code Stream e adicionar associações de cifrão a ele, faça o seguinte:

- 1 No vRealize Automation Code Stream, clique em **Pipelines > Tela em Branco**.
- 2 Adicione uma tarefa de **Modelo de nuvem** ao pipeline.
- 3 Na tarefa de modelo de nuvem, para **Origem do modelo de nuvem**, selecione **Modelos de nuvem do Cloud Assembly**, insira o nome do modelo de nuvem e selecione a versão do modelo de nuvem.
- 4 Observe que é possível inserir um token de API, que pode fornecer credenciais para o pipeline a ser executado, e usar **Criar Variável** para criar uma variável que criptografa o token de API na tarefa do modelo de nuvem.
- 5 Na tabela de **Parâmetros e Valores** exibida, observe os valores do parâmetro. O valor padrão para **Tipo** é **Pequeno** e o valor padrão para **Imagem** é **ubuntu**.
- 6 Digamos que você precise alterar o modelo de nuvem no vRealize Automation Cloud Assembly. Por exemplo:
 - a Defina o **Tipo** para usar uma propriedade do tipo **Matriz**. O vRealize Automation Cloud Assembly permite valores separados por vírgula para **Tipo** quando o tipo é **Matriz**.
 - b Clique em **Implantar**.
 - c Na página **Tipo de Implantação**, insira um nome de implantação e selecione a versão do modelo de nuvem.
 - d Na página **Entradas de Implantação**, você pode definir um ou mais valores para **Tipo**.
 - e Observe que as entradas de Implantação incluem todas as variáveis definidas no seu código de modelo de nuvem e aparecerão conforme definido no código do modelo de nuvem. Por exemplo: `Integer Var`, `String Var`, `Flavor Var`, `Boolean Var`, `Object Var` e `Array Var`. `String Var` e `Flavor Var` são valores de cadeia de caracteres, enquanto `Boolean Var` é uma caixa de seleção.
 - f Clique em **Implantar**.
- 7 No vRealize Automation Code Stream, selecione a nova versão do modelo de nuvem e insira os valores na tabela **Parâmetros e Valores**. Modelos de nuvem oferecem suporte aos seguintes tipos de parâmetro e à capacidade do vRealize Automation Code Stream de permitir associações de variáveis de cifrão a eles, o que resulta em pequenas diferenças na interface do usuário de tarefas de modelo de nuvem do vRealize Automation Code Stream e na interface de modelo de nuvem do vRealize Automation Cloud Assembly. Dependendo de como um modelo de nuvem é codificado, você pode ou não ter permissão para inserir valores na tarefa.
 - a Para **flavorVar**, se o modelo de nuvem tiver definido o tipo como cadeia de caracteres ou matriz, insira uma cadeia de caracteres ou uma matriz de valores separados por vírgula. Uma matriz de exemplo é semelhante a **test, test**.

- b Para **BooleanVar**, no menu suspenso, selecione **true** ou **false**. Ou, para usar uma associação de variável, insira \$ e selecione uma associação de variável na

The screenshot shows a table with two columns: 'Parameter' and 'Value'. The rows are:

Parameter	Value
stringVar	raj
integerVar	1
flavorVar	medium
BooleanVar	\$
objectVar	
arrayVar	

Below the table, there is a section labeled 'Output Parameter' with a dropdown menu. The dropdown menu is open, showing a list of variables: var, input, comments, requestBy, executionIndex, executionId, executionUrl, name, description, and Stage0. The 'var' option is selected.

At the bottom of the interface, there are several buttons: 'status', 'deploymentCriteria', 'deploymentId', and 'deploymentName'.

lista.

- c Para **objectVar**, insira o valor entre chaves e aspas neste formato: `{"bkix": "bkix": }`.
 - d O **objectVar** será transmitido ao modelo de nuvem e poderá ser usado de várias maneiras, dependendo do modelo de nuvem. Ele permite um formato de cadeia de caracteres para um objeto JSON e pode adicionar pares de chave/valor como valores separados por vírgula na tabela de chaves/valores. Você pode inserir texto sem formatação para um objeto JSON ou um par de chave/valor como um formato de cadeia de caracteres normal para o JSON.
 - e Para **arrayVar**, insira o valor de entrada separado por vírgula como uma matriz no seguinte formato: `["1", "2"]`.
- 8 No pipeline, é possível associar um parâmetro de entrada a uma matriz.
- a Clique na guia **Entrada**.
 - b Insira um nome para a entrada. Por exemplo, **arrayInput**.
 - c Na tabela **Parâmetros e Valores**, clique em **arrayVar** e insira `${input.arrayInput}`.
 - d Depois de salvar o pipeline e habilitá-lo, quando o pipeline for executado, você deverá fornecer um valor de entrada de matriz. Por exemplo, insira `["1", "2"]` e clique em **Executar**.

Agora, você aprendeu a usar associações de variáveis de cifrão (\$) em um modelo de nuvem em uma tarefa de modelo de nuvem de pipeline do vRealize Automation Code Stream.

Como transmitir um parâmetro a um pipeline quando ele é executado

Você pode adicionar parâmetros de entrada ao pipeline para que o vRealize Automation Code Stream os transmita ao pipeline. Em seguida, quando o pipeline for executado, um usuário deverá inserir o valor para o parâmetro de entrada. É possível adicionar parâmetros de saída ao pipeline para que as tarefas desse pipeline possam usar o valor de saída de uma tarefa. O vRealize Automation Code Stream oferece suporte ao uso de parâmetros de várias maneiras que dão suporte às suas próprias necessidades de pipeline.

Por exemplo, para solicitar que um usuário insira a URL do servidor Git quando um pipeline com uma tarefa REST for executada, é possível associar a tarefa REST a uma URL do servidor Git.

Para criar a associação de variáveis, adicione uma variável de associação de URL à tarefa REST. Quando o pipeline for executado e atingir a tarefa REST, um usuário deverá inserir a URL do servidor Git. Veja como criar a associação:

- 1 No pipeline, clique em a guia **Entrada**.
- 2 Para definir o parâmetro, para **Parâmetros de injeção automática** clique em **Git**.
A lista de parâmetros Git é exibida e inclui **GIT_SERVER_URL**. Se você precisar usar um valor padrão para a URL do servidor Git, edite esse parâmetro.
- 3 Clique em **Modelo** e clique na tarefa REST.
- 4 Na guia **Tarefa**, na área **URL**, insira \$ e, em seguida, selecione **entrada** e **GIT_SERVER_URL**.

The screenshot shows the configuration window for a task named 'Task3'. The interface includes tabs for 'Task :Task3', 'Notifications', and 'Rollback', with a 'VALIDATE TASK' button in the top right. The configuration fields are as follows:

- Task name:** Task3
- Type:** REST
- Continue on failure:** ☐
- Execute task:** ☒ Always ☐ On condition
- REST Request:**
 - Action:** GET
 - URL:** \${input.GIT_SERVER_URL} (A dropdown menu is open showing a list of Git-related variables: GIT_BRANCH_NAME, GIT_CHANGE_SUBJECT, GIT_COMMIT_ID, GIT_EVENT_DESCRIPTION, GIT_EVENT_OWNER_NAME, GIT_EVENT_TIMESTAMP, GIT_REPO_NAME, and GIT_SERVER_URL, which is highlighted.)
 - Agent endpoint:**
 - Headers:**
- Output Parameters:** status, responseHeaders, responseBody, responseJson, responseCode

A entrada é semelhante a: `${input.GIT_SERVER_URL}`

- Para verificar a integridade da associação de variáveis da tarefa, clique em **Validar Tarefa**.

O vRealize Automation Code Stream indica que a tarefa foi validada com êxito.

- Quando o pipeline executar a tarefa REST, um usuário deverá inserir a URL do servidor Git. Caso contrário, a tarefa não terminará de ser executada.

Como associar duas tarefas de pipeline criando parâmetros de entrada e saída

Ao associar tarefas ao mesmo tempo, adicione uma variável de associação à configuração de entrada da tarefa de recebimento. Em seguida, quando o pipeline for executado, um usuário deverá substituir a variável de associação pela entrada obrigatória.

Para vincular tarefas de pipeline ao mesmo tempo, use a variável de cifrão (\$) nos parâmetros de entrada e saída. Veja como.

Digamos que o pipeline precise chamar uma URL em uma tarefa REST e a saída de uma resposta. Para fazer isso, você inclui os parâmetros de entrada e saída na sua tarefa REST. Você também precisa que um usuário aprove a tarefa e, por isso, também inclua uma tarefa Operações do Usuário para que outro usuário a aprove quando o pipeline for executado. Esse exemplo mostra como usar expressões nos parâmetros de entrada e saída e fazer com que o pipeline aguarde a aprovação na tarefa.

- 1 No pipeline, clique em a guia **Entrada**.

The screenshot shows the 'rest-ix-1' task configuration in the 'Input' tab. The 'Auto inject parameters' section has four radio buttons: 'Gerrit', 'Git', 'Docker', and 'None' (which is selected). Below this is an 'ADD' button and a text box 'ADD/REMOVE INJECTED PARAMETERS'. A table lists the injected parameters:

Starred	Name	Value	Description
<input type="checkbox"/>	URL	{Stage0.Task3.input.http://www.docs.vmware.com}	Docs URL

- 2 Deixe **Parâmetros de injeção automática** como **Nenhum**.
- 3 Clique em **Adicionar** e insira o nome do parâmetro, o valor e a descrição e clique em **OK**. Por exemplo:
 - a Insira um nome de URL.
 - b Insira o valor: {Stage0.Task3.input.http://www.docs.vmware.com}
 - c Insira uma descrição.
- 4 Clique na guia **Saída**, clique em **Adicionar** e insira o nome e o mapeamento de parâmetros de saída.

Add Pipeline Output Parameter

Name *

Reference \$ *

responseHeaders

responseBody

responseJson

responseCode

- a Insira um nome de parâmetro de saída exclusivo.
- b Clique na área **Referência** e insira \$.
- c Insira o mapeamento de saída da tarefa selecionando as opções à medida que elas são exibidas. Selecione **Stage0**, **Task3**, **output** e **responseCode**. Em seguida, clique em **OK**.

rest-ix-1
Enabled
ACTIONS


Workspace
Input
Model
Output

Output Parameters ⓘ

ADD

Starred ⓘ	Name ▼	Reference
⋮ ☆	RESTResponse	\${Stage0.Task3.output.responseCode}

- 5 Salve seu pipeline.
- 6 No menu **Ações**, clique em **Executar**.
- 7 Clique em **Ações > Exibir execuções**.
- 8 Clique na execução e consulte os parâmetros de entrada e saída que você definiu.

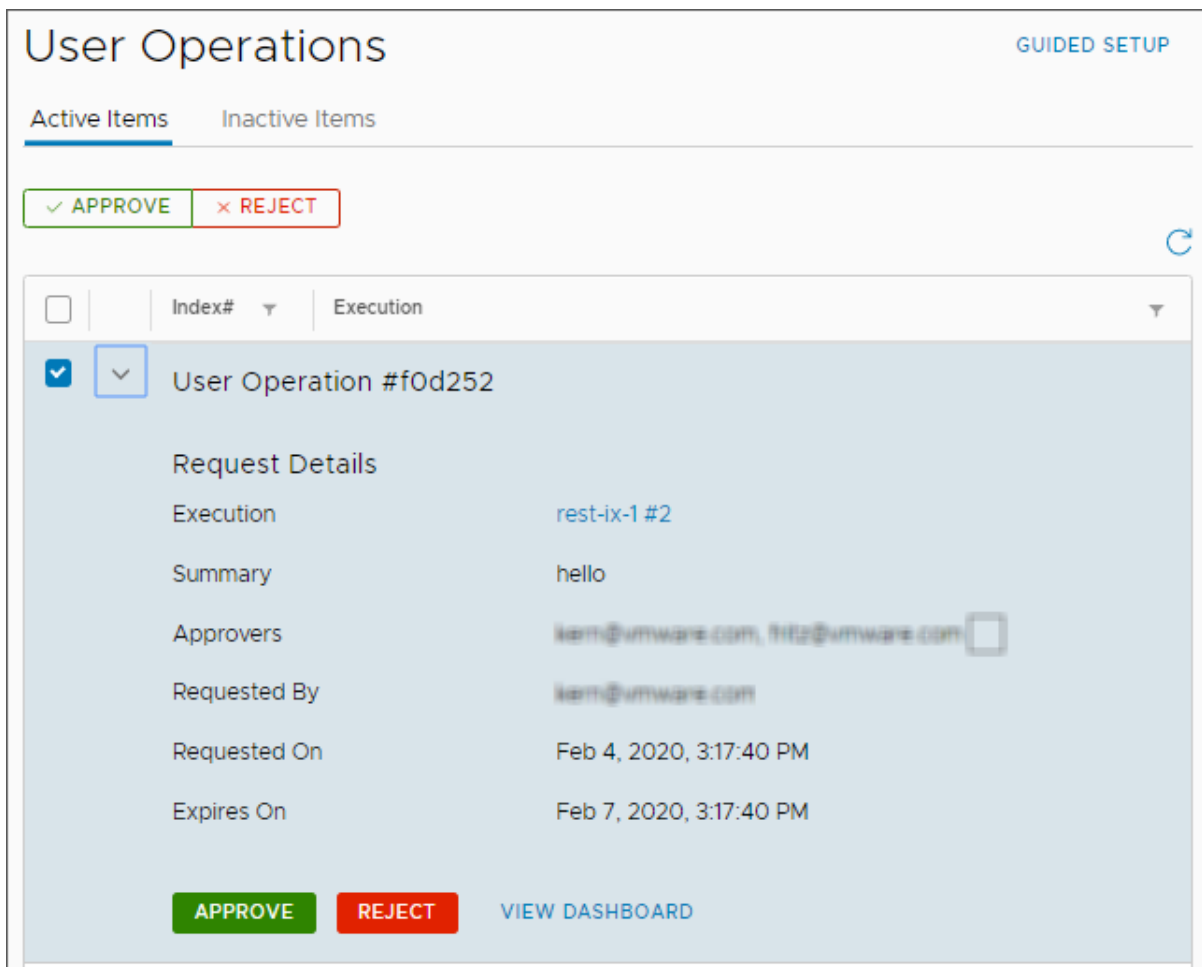

rest-ix-1 #2
WAITING
0
ACTIONS ▾

✉ Stage0

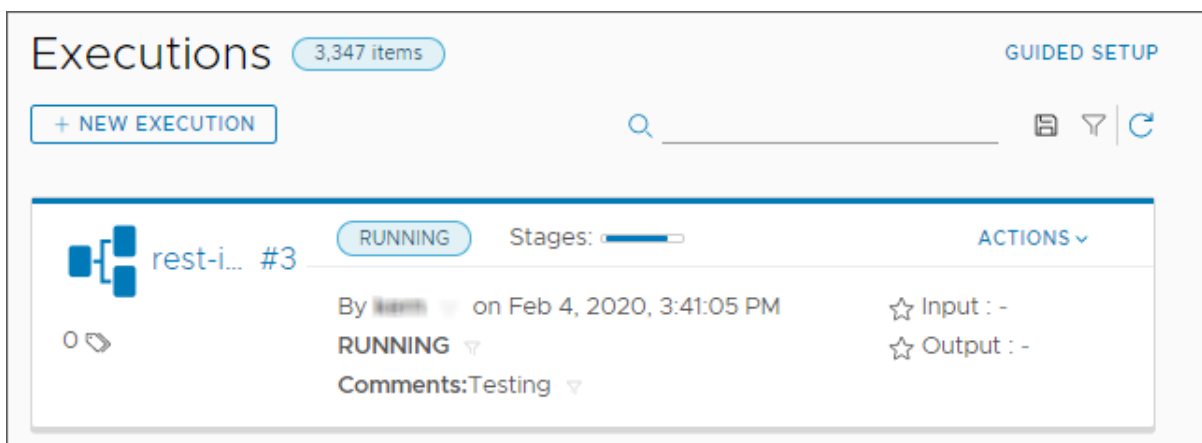
✉ Task2
● Task3

Project	chim
Execution	rest-ix-1 #2
Status	WAITING Stage0.Task2: Execution Waiting for User Action.
Updated By	
Executed By	user@vmware.com
Comments	Test Vars Expressions
Duration	37 seconds (Feb 4, 2020, 3:17:31 PM - Feb 4, 2020, 3:17:42 PM)
Input Parameters ▾	
URL	{Stage0.Task3.input.http://www.docs.vmware.com}
Workspace	No details available
Output Parameters ▾	
Response	tasks['Stage0.Task3']['output.responseCode']

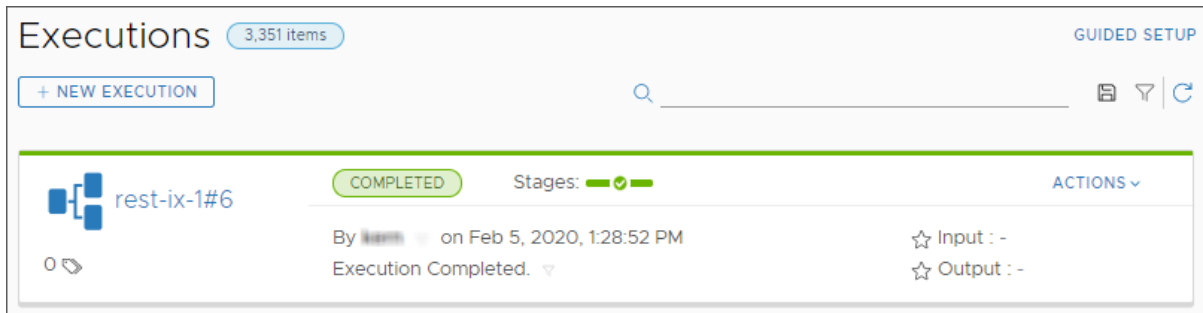
- 9 Para aprovar o pipeline, clique em **Operações do Usuário** e exiba a lista de aprovações na guia **Itens Ativos**. Ou permaneça em Execuções, clique na tarefa e clique em **Aprovar**.
- 10 Para habilitar os botões **Aprovar** e **Rejeitar**, clique na caixa de seleção ao lado da execução.
- 11 Para ver os detalhes, expanda a seta suspensa.
- 12 Para aprovar a tarefa, clique em **APROVAR**, insira um motivo e clique em **OK**.



- 13 Clique em **Execuções** e observe que o pipeline continuará a ser executado.



- 14 Se o pipeline falhar, corrija os erros, salve-o e execute-o novamente.



Como saber mais sobre variáveis e expressões

Para ver detalhes sobre como usar variáveis e expressões ao associar tarefas de pipeline, consulte [Que variáveis e expressões eu posso usar ao associar tarefas de pipeline no vRealize Automation Code Stream](#).

Para saber como usar a saída da tarefa de pipeline com uma associação de variável de condição, consulte [Como usar associações de variáveis em uma tarefa de condição para executar ou parar um pipeline no vRealize Automation Code Stream](#).

Como usar associações de variáveis em uma tarefa de condição para executar ou parar um pipeline no vRealize Automation Code Stream

É possível fazer com que a saída de uma tarefa no pipeline determine se o pipeline é executado ou interrompido com base em uma condição fornecida. Para aprovar ou reprovar o pipeline com base na saída da tarefa, use o tipo de tarefa Condição.

O tipo de tarefa **Condição** pode ser usado como um ponto de decisão no pipeline. Usando a tarefa Condição com uma expressão de condição fornecida, é possível avaliar quaisquer propriedades em seu pipeline, estágios e tarefas.

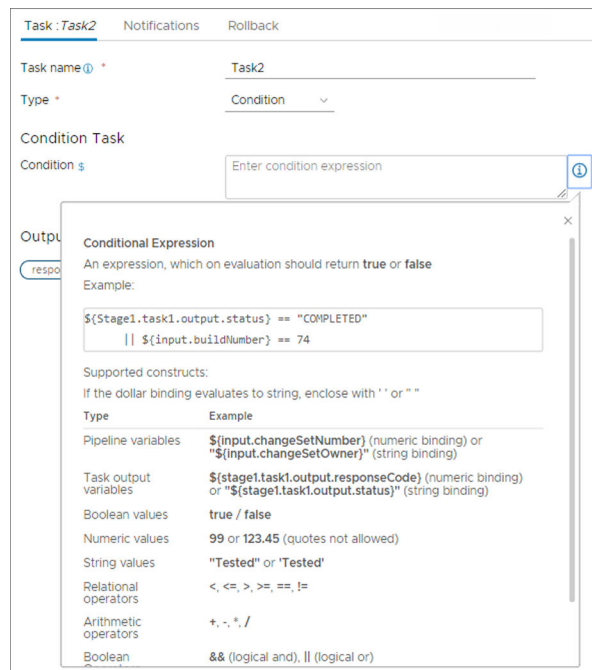
O resultado da tarefa Condição determina se a próxima tarefa no pipeline será executada.

- Uma condição verdadeira permite que o pipeline continue a ser executado.
- Uma condição falsa interrompe o pipeline.

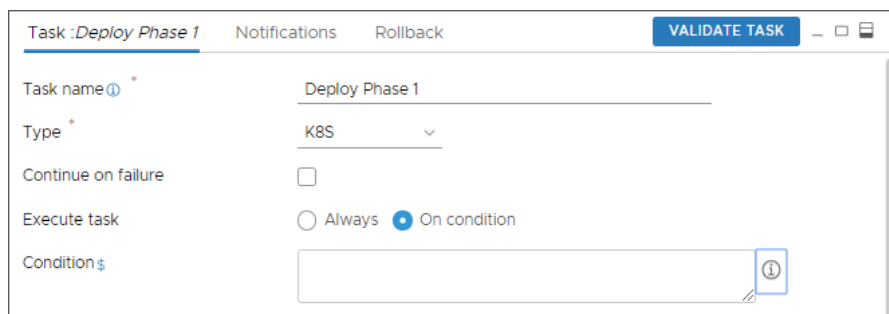
Para obter exemplos de como usar o valor de saída de uma tarefa como a entrada para a próxima tarefa vinculando as tarefas a uma tarefa Condição, consulte [Como usar associações de variáveis em pipelines do vRealize Automation Code Stream](#).

Tabela 3-7. Como a tarefa de Condição e sua expressão de condição se relacionam com o pipeline

Tipo de tarefa	O que afeta...	O que ele faz...
Condição...		
Tarefa de Condição	Pipeline	O tipo de tarefa de Condição determina se o pipeline é executado ou interrompido nesse ponto, dependendo se a saída da tarefa for verdadeira ou falsa.
Expressão de Condição	Saída da tarefa de Condição	Quando o pipeline é executado, a expressão de condição incluída na tarefa de Condição produz um status de saída verdadeiro ou falso. Por exemplo, uma expressão de condição pode exigir que o status de saída da tarefa de Condição seja <code>Concluído</code> ou use um número de compilação de 74. A expressão de condição é exibida na guia Tarefa no tipo de tarefa de Condição.



O tipo de tarefa de **Condição** é diferente na função e no comportamento da configuração **Na Condição** em outros tipos de tarefa.



Em outros tipos de tarefa, a configuração **Na Condição**:

- Determina se a tarefa atual é executada, em vez de tarefas sucessivas, com base na avaliação de sua expressão de condição prévia de verdadeiro ou falso. A expressão de condição da configuração Na Condição produz um status de saída verdadeiro ou falso para a tarefa atual quando o pipeline é executado.
- É exibido na guia Tarefa com sua própria expressão de condição.

Este exemplo usa a tarefa de Condição.

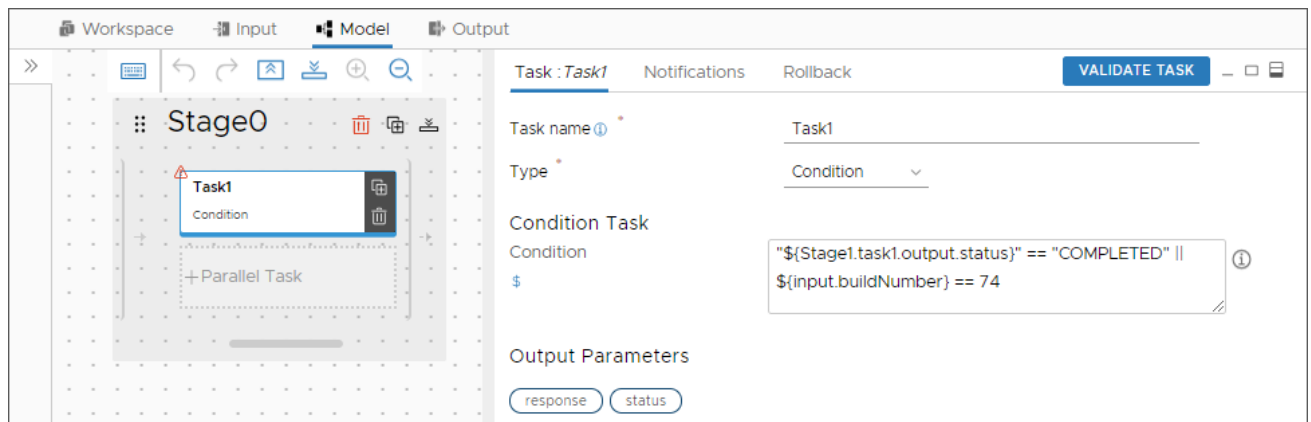
Pré-requisitos

- Verifique a existência de um pipeline e se ele inclui estágios e tarefas.

Procedimentos

- 1 No pipeline, determine o ponto de decisão no qual a tarefa de Condição deve aparecer.
- 2 Adicione a tarefa de Condição antes da tarefa que depende do seu status de falha ou de aprovação.
- 3 Adicione uma expressão de condição à tarefa de Condição.

Por exemplo: `"${Stage1.task1.output.status}" == "COMPLETED" || ${input.buildNumber} == 74`



- 4 Valide a tarefa.
- 5 Salve o pipeline, em seguida, ative e execute-o.

Resultados

Observe as execuções do pipeline e observe se o pipeline continua a ser executado ou é interrompido na tarefa de Condição.

Próximo passo

Também é possível usar o tipo de tarefa de Condição se decidir reverter uma implantação de pipeline. Por exemplo, em um pipeline de reversão, a tarefa de Condição ajuda o vRealize Automation Code Stream a marcar uma falha de pipeline com base na expressão de condição e pode disparar um único fluxo de reversão para vários tipos de falha.

Para reverter uma implantação, consulte [Como faço para reverter minha implantação no vRealize Automation Code Stream](#).

Que variáveis e expressões eu posso usar ao associar tarefas de pipeline no vRealize Automation Code Stream

Com variáveis e expressões, você pode usar parâmetros de entrada e saída com suas tarefas de pipeline. Os parâmetros inseridos vinculam a tarefa de pipeline a uma ou mais variáveis, expressões ou condições e determinam o comportamento do pipeline quando ele é executado.

Ao associar tarefas de pipeline, você pode incluir expressões padrão e complexas para que o pipeline possa executar soluções de entrega de software simples ou complexas. Para criar os parâmetros no seu pipeline, clique na guia **Entrada** ou **Saída** e adicione uma variável inserindo o símbolo de cifrão **\$** e uma expressão. Por exemplo, esse parâmetro é usado como uma entrada de tarefa que chama uma URL: `${Stage0.Task3.input.URL}`.

O formato de associações de variáveis usa componentes de sintaxe chamados de escopos e chaves. O **SCOPE** define o contexto como entrada ou saída, e o **KEY** define os detalhes. No exemplo de parâmetro `${Stage0.Task3.input.URL}`, o `input` é o **SCOPE** e a `URL` é a **KEY**.

As propriedades de saída de qualquer tarefa podem ser resolvidas para qualquer número de níveis aninhados de vinculação de variável.

Para saber mais sobre como usar associações de variáveis em pipelines, consulte [Como usar associações de variáveis em pipelines do vRealize Automation Code Stream](#).

Usando expressões de cifrão com escopos e chaves para associar tarefas de pipeline

Você pode associar tarefas de pipeline usando expressões em variáveis de símbolo de cifrão. Insira expressões como `${SCOPE.KEY.<PATH>}`.

Em cada expressão, **SCOPE** é o contexto que o vRealize Automation Code Stream usa para determinar o comportamento de uma tarefa de pipeline. O escopo procura um **KEY**, que define os detalhes da ação que a tarefa realiza. Quando o valor para **KEY** é um objeto aninhado, você pode fornecer um **PATH** opcional.

Esses exemplos descrevem **SCOPE** e **KEY** e mostram como usá-los no pipeline.

Tabela 3-8. Usando SCOPE e KEY

SCOPE	Finalidade da expressão e do exemplo	KEY	Como usar SCOPE e KEY no pipeline
input	Propriedades de entrada de um pipeline: <code>\${input.input1}</code>	Nome da propriedade de entrada	<p>Para fazer referência à propriedade de entrada de um pipeline em uma tarefa, use este formato:</p> <pre>tasks: mytask: type: REST input: url: \$ {input.url} action: get</pre> <pre>input: url: https:// www.vmware.com</pre>
output	Propriedades de saída de um pipeline: <code>\${output.output1}</code>	Nome da propriedade de saída	<p>Para fazer referência a uma propriedade de saída para que o pipeline possa enviar uma notificação, use o seguinte formato:</p> <pre>notifications: email: - endpoint: MyEmailEndpoint subject: "Deployment Successful" event: COMPLETED to: - user@example.org body: Pipeline deployed the service successfully. Refer \$ {output.serviceURL}</pre>

Tabela 3-8. Usando SCOPE e KEY (continuação)

SCOPE	Finalidade da expressão e do exemplo	KEY	Como usar SCOPE e KEY no pipeline
entrada da tarefa	Entrada para uma tarefa: \$ {MY_STAGE.MY_TASK.input. SOMETHING}	Indica a entrada de uma tarefa em uma notificação	<p>Quando um trabalho do Jenkins é iniciado, ele pode fazer referência ao nome do trabalho disparado na entrada da tarefa. Nesse caso, envie uma notificação usando este formato:</p> <pre> notifications: email: - endpoint: MyEmailEndpoint stage: MY_STAGE task: MY_TASK subject: "Build Started" event: STARTED to: - user@example.org body: Jenkins job \$ {MY_STAGE.MY_TASK.i nput.job} started for commit id \$ {input.COMMITID}). </pre>
saída da tarefa	Saída de uma tarefa: \$ {MY_STAGE.MY_TASK.output .SOMETHING}	Indica a saída de uma tarefa em uma tarefa subsequente	<p>Para fazer referência à saída da tarefa 1 do pipeline na tarefa 2, use o seguinte formato:</p> <pre> taskOrder: - task1 - task2 tasks: task1: type: REST input: action: get url: https:// www.example.org/api/ /status task2: type: REST input: action: post url: https:// status.internal.exa mple.org/api/ activity payload: \$ {MY_STAGE.task1.out put.responseBody} </pre>

Tabela 3-8. Usando SCOPE e KEY (continuação)

SCOPE	Finalidade da expressão e do exemplo	KEY	Como usar SCOPE e KEY no pipeline
var	Variável: <code>\${var.myVariable}</code>	Fazer referência à variável em um endpoint	<p>Para fazer referência a uma variável secreta em um endpoint para uma senha, use este formato:</p> <pre> --- project: MyProject kind: ENDPOINT name: MyJenkinsServer type: jenkins properties: url: https:// jenkins.example.com username: jenkinsUser password: \$ {var.jenkinsPassword} </pre>
var	Variável: <code>\${var.myVariable}</code>	Fazer referência à variável em um pipeline	<p>Para fazer referência à variável em um URL de pipeline, use este formato:</p> <pre> tasks: task1: type: REST input: action: get url: \$ {var.MY_SERVER_URL} </pre>
status da tarefa	<p>Status de uma tarefa:</p> <pre> \$ {MY_STAGE.MY_TASK.status} </pre> <p>\$</p> <pre> {MY_STAGE.MY_TASK.status Message} </pre>		
status do estágio	<p>Status de um estágio:</p> <pre> \${MY_STAGE.status} </pre> <p>\$</p> <pre> {MY_STAGE.statusMessage} </pre>		

Expressões padrão

Você pode usar variáveis com expressões no seu pipeline. Esse resumo inclui as expressões padrão que você pode usar.

Expressão	Descrição
<code>\${comments}</code>	Comentários fornecidos quando a execução foi solicitada.
<code>\${duration}</code>	Duração da execução do pipeline.
<code>\${endTime}</code>	Hora de término da execução do pipeline em UTC, se concluída.
<code>\${executedOn}</code>	O mesmo que a hora de início, a hora de início da execução do pipeline em UTC.
<code>\${executionId}</code>	ID da execução do pipeline.
<code>\${executionUrl}</code>	URL que navega até a execução do pipeline na interface do usuário.
<code>\${name}</code>	Nome do pipeline.
<code>\${requestBy}</code>	Nome do usuário que solicitou a execução.
<code>\${stageName}</code>	Nome do estágio atual, quando usado no escopo de um estágio.
<code>\${startTime}</code>	Hora de início da execução do pipeline em UTC.
<code>\${status}</code>	Status da execução.
<code>\${statusMessage}</code>	Mensagem de status da execução do pipeline.
<code>\${taskName}</code>	Nome da tarefa atual, quando usada em uma entrada de tarefa ou notificação.

Usando SCOPE and KEY em tipos de tarefa de pipeline

Você pode usar expressões com qualquer um dos tipos de tarefa de pipeline compatíveis. Use estes exemplos como referência para como definir o `SCOPE` e `KEY` e confirme a sintaxe. Esses exemplos de códigos usam `MY_STAGE` e `MY_TASK` como nomes de estágios e tarefas de pipeline.

Para saber mais sobre os tipos de tarefa disponíveis, consulte [Quais tipos de tarefa estão disponíveis no vRealize Automation Code Stream](#).

Tabela 3-9. Tarefas de isolamento

Tipo de tarefa	Escopo	Key	Como usar SCOPE e KEY na tarefa
Operação do Usuário			
	Input	<p>summary: resumo da solicitação para Operação do Usuário</p> <p>description: descrição da solicitação de Operação do Usuário</p> <p>approvers: lista de endereços de e-mail de aprovadores, em que cada entrada pode ser uma variável com uma vírgula, ou use ponto-e-vírgula para separar e-mails</p> <p>approverGroups: lista de endereços de grupos de aprovadores para a plataforma e a identidade</p> <p>sendemail: envia opcionalmente uma notificação por e-mail após a solicitação ou resposta quando definido como "true"</p> <p>expirationInDays: número de dias que representa o tempo de expiração da solicitação</p>	<pre> \${MY_STAGE.MY_TASK.input.summary} \${MY_STAGE.MY_TASK.input.description} \${MY_STAGE.MY_TASK.input.approvers} \$ {MY_STAGE.MY_TASK.input.approverGroups} \${MY_STAGE.MY_TASK.input.sendemail} \$ {MY_STAGE.MY_TASK.input.expirationInDays} </pre>
	Output	<p>index: string hexadecimal de seis dígitos que representa a solicitação</p> <p>respondedBy: nome da conta da pessoa que aprovou/rejeitou a Operação do Usuário</p> <p>respondedByEmail: endereço de e-mail da pessoa que respondeu</p> <p>comments: comentários fornecidos durante a resposta</p>	<pre> \${MY_STAGE.MY_TASK.output.index} \${MY_STAGE.MY_TASK.output.respondedBy} \$ {MY_STAGE.MY_TASK.output.respondedByEmail} \${MY_STAGE.MY_TASK.output.comments} </pre>
Condição			
	Input	<p>condition: condição a ser avaliada. Quando a condição é avaliada como "true", a tarefa é marcada como concluída, enquanto outras respostas reprovam a tarefa</p>	<pre> \${MY_STAGE.MY_TASK.input.condition} </pre>
	Output	<p>result: resultado após a avaliação</p>	<pre> \${MY_STAGE.MY_TASK.output.response} </pre>

Tabela 3-10. Tarefas de pipeline

Tipo de tarefa	Escopo	Key	Como usar SCOPE e KEY na tarefa
Pipeline			
	Input	name: nome do pipeline a ser executado inputProperties: propriedades de entrada a serem transmitidas à execução de pipeline aninhada	<pre> \${MY_STAGE.MY_TASK.input.name} \${MY_STAGE.MY_TASK.input.inputProperties} # Referência a todas as propriedades \$ {MY_STAGE.MY_TASK.input.inputProperties.input1} # Referência ao valor de input1 </pre>
	Output	executionStatus: status da execução do pipeline executionIndex: índice da execução do pipeline outputProperties: propriedades de saída de uma execução de pipeline	<pre> \${MY_STAGE.MY_TASK.output.executionStatus} \${MY_STAGE.MY_TASK.output.executionIndex} \${MY_STAGE.MY_TASK.output.outputProperties} # Referência a todas as propriedades \$ {MY_STAGE.MY_TASK.output.outputProperties.output1} # Referência ao valor de output1 </pre>

Tabela 3-11. Automatizar tarefas de integração contínua

Tipo de tarefa	Escopo	Key	Como usar SCOPE e KEY na tarefa
IC			
	Input	steps: um conjunto de cadeias de caracteres que representam comandos para execução export: variáveis de ambiente a serem preservadas após a execução das etapas artifacts: caminhos de artefatos a serem preservados no caminho compartilhado process: conjunto de elementos de configuração para processamento de JUnit, JaCoCo, Checkstyle, FindBugs	<pre> \${MY_STAGE.MY_TASK.input.steps} \${MY_STAGE.MY_TASK.input.export} \${MY_STAGE.MY_TASK.input.artifacts} \${MY_STAGE.MY_TASK.input.process} \$ {MY_STAGE.MY_TASK.input.process[0].path} # Referência ao caminho da primeira configuração </pre>

Tabela 3-11. Automatizar tarefas de integração contínua (continuação)

Tipo de tarefa	Escopo	Key	Como usar SCOPE e KEY na tarefa
	Output	<p>exports: par de chave/valor, que representa as variáveis de ambiente exportadas da entrada <code>export</code></p> <p>artifacts: caminho dos artefatos preservados com êxito</p> <p>processResponse: conjunto de resultados processados para a entrada <code>process</code></p>	<pre> \${MY_STAGE.MY_TASK.output.exports} # Referência a todas as exportações \$ {MY_STAGE.MY_TASK.output.exports.myvar} # Referência ao valor de myvar \${MY_STAGE.MY_TASK.output.artifacts} \$ {MY_STAGE.MY_TASK.output.processResponse} \$ {MY_STAGE.MY_TASK.output.processResponse[0].result} # Resultado da primeira configuração do processo </pre>
Personalizado			
	Input	<p>name: nome da integração personalizada</p> <p>version: uma versão da integração personalizada, liberada ou preterida</p> <p>properties: propriedades a serem enviadas à integração personalizada</p>	<pre> \${MY_STAGE.MY_TASK.input.name} \${MY_STAGE.MY_TASK.input.version} \${MY_STAGE.MY_TASK.input.properties} # Referência a todas as propriedades \$ {MY_STAGE.MY_TASK.input.properties.property1} # Referência ao valor de property1 </pre>
	Output	<p>properties: propriedades de saída da resposta de integração personalizada</p>	<pre> \${MY_STAGE.MY_TASK.output.properties} # Referência a todas as propriedades \$ {MY_STAGE.MY_TASK.output.properties.property1} # Referência ao valor de property1 </pre>

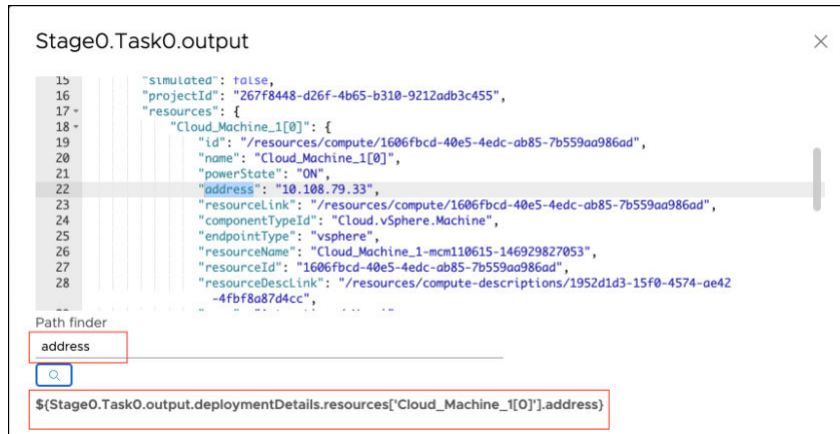
Tabela 3-12. Automatizar tarefas de implantação contínua: Modelo de nuvem

Tipo de tarefa	Escopo	Key	Como usar SCOPE e KEY na tarefa
Modelo de nuvem			
	Input	<p>action: Uma das opções createDeployment, updateDeployment, deleteDeployment, rollbackDeployment</p> <p>blueprintInputParams: usado para criar e atualizar ações de implantação</p> <p>allowDestroy: as máquinas podem ser destruídas no processo de implantação da atualização.</p> <p>CREATE_DEPLOYMENT</p> <ul style="list-style-type: none"> ■ blueprintName: Nome do modelo de nuvem ■ blueprintVersion: versão do modelo de nuvem <p>OU</p> <ul style="list-style-type: none"> ■ fileUrl: URL do modelo de nuvem remota YAML, após selecionar um servidor GIT. <p>UPDATE_DEPLOYMENT</p> <p>Qualquer uma destas combinações:</p> <ul style="list-style-type: none"> ■ blueprintName: Nome do modelo de nuvem ■ blueprintVersion: versão do modelo de nuvem <p>OU</p> <ul style="list-style-type: none"> ■ fileUrl: URL do modelo de nuvem remota YAML, após selecionar um servidor GIT. <p>-----</p> <ul style="list-style-type: none"> ■ deploymentId: ID da implantação <p>OU</p> <ul style="list-style-type: none"> ■ deploymentName: nome da implantação <p>-----</p> <p>DELETE_DEPLOYMENT</p> <ul style="list-style-type: none"> ■ deploymentId: ID da implantação 	

Tabela 3-12. Automatizar tarefas de implantação contínua: Modelo de nuvem (continuação)

Tipo de tarefa	Escopo	Key	Como usar SCOPE e KEY na tarefa
		<p>OU</p> <ul style="list-style-type: none"> ■ <code>deploymentName</code>: nome da implantação <p>ROLLBACK_DEPLOYMENT</p> <p>Qualquer uma destas combinações:</p> <ul style="list-style-type: none"> ■ <code>deploymentId</code>: ID da implantação <p>OU</p> <ul style="list-style-type: none"> ■ <code>deploymentName</code>: nome da implantação <p>-----</p> <ul style="list-style-type: none"> ■ <code>blueprintName</code>: Nome do modelo de nuvem ■ <code>rollbackVersion</code>: versão para a qual reverter 	
	Output		<p>Parâmetros que podem se associar a outras tarefas ou à saída de um pipeline:</p> <ul style="list-style-type: none"> ■ O nome da implantação pode ser acessado como <code>\${Stage0.Task0.output.deploymentName}</code> ■ O Id de implantação pode ser acessado como <code>\${Stage0.Task0.output.deploymentId}</code> ■ Detalhes de Implantação são um objeto complexo, e detalhes internos podem ser acessados usando os resultados do JSON. <p>Para acessar qualquer propriedade, use o operador de ponto para seguir a hierarquia JSON. Por exemplo, para acessar o endereço do recurso <code>Cloud_Machine_1[0]</code>, a associação <code>\$</code> é:</p> <pre><code>\${Stage0.Task0.output.deploymentDetails.resources['Cloud_Machine_1[0]'].address}</code></pre> <p>Da mesma forma, para o tipo, a associação <code>\$</code> é:</p> <pre><code>\${Stage0.Task0.output.deploymentDetails.resources['Cloud_Machine_1[0]'].flavor}</code></pre> <p>Na interface do usuário do vRealize Automation Code Stream, você pode obter as associações <code>\$</code> para qualquer propriedade.</p> <ol style="list-style-type: none"> 1 Na área de propriedade de saída da tarefa, clique em EXIBIR JSON DE SAÍDA. 2 Para encontrar a associação <code>\$</code>, insira qualquer propriedade. 3 Clique no ícone de pesquisa, que exibe a associação <code>\$</code> correspondente.

Exemplo de saída JSON:



Amostra de objeto de detalhes de implantação:

```

{
  "id": "6a031f92-d0fa-42c8-bc9e-3b260ee2f65b",
  "name": "deployment_6a031f92-d0fa-42c8-bc9e-3b260ee2f65b",
  "description": "Pipeline Service triggered operation",
  "orgId": "434f6917-4e34-4537-b6c0-3bf3638a71bc",
  "blueprintId": "8d1dd801-3a32-4f3b-adde-27f8163dfe6f",
  "blueprintVersion": "1",
  "createdAt": "2020-08-27T13:50:24.546215Z",
  "createdBy": "user@vmware.com",
  "lastUpdatedAt": "2020-08-27T13:52:50.674957Z",
  "lastUpdatedBy": "user@vmware.com",
  "inputs": {},
  "simulated": false,
  "projectId": "267f8448-d26f-4b65-b310-9212adb3c455",
  "resources": {
    "Cloud_Machine_1[0]": {
      "id": "/resources/compute/1606fbcd-40e5-4edc-ab85-7b559aa986ad",
      "name": "Cloud_Machine_1[0]",
      "powerState": "ON",
      "address": "10.108.79.33",
      "resourceLink": "/resources/compute/1606fbcd-40e5-4edc-ab85-7b559aa986ad",
      "componentTypeId": "Cloud.vSphere.Machine",
      "endpointType": "vsphere",
      "resourceName": "Cloud_Machine_1-mcm110615-146929827053",
      "resourceId": "1606fbcd-40e5-4edc-ab85-7b559aa986ad",
      "resourceDescLink": "/resources/compute-descriptions/1952d1d3-15f0-4574-ae42-4fbf8a87d4cc",
      "zone": "Automation / Vms",
      "countIndex": "0",
      "image": "ubuntu",
      "count": "1",
      "flavor": "small",
      "region": "MYBU",
      "_clusterAllocationSize": "1",
      "osType": "LINUX",
      "componentType": "Cloud.vSphere.Machine",

```

```
        "account": "bha"
      },
      "status": "CREATE_SUCCESSFUL",
      "deploymentURI": "https://api.yourenv.com/automation-ui/#/deployment-ui;ash=/deployment/6a031f92-d0fa-42c8-bc9e-3b260ee2f65b"
    }
  }
```

Tabela 3-13. Automatizar tarefas de implantação contínua: Kubernetes

Tipo de tarefa	Escopo	Key	Como usar SCOPE e KEY na tarefa
Kubernetes			
	Input	<p>action: uma das opções OBTER, CRIAR, APLICAR, EXCLUIR, REVERTER</p> <ul style="list-style-type: none"> ■ timeout: tempo limite geral para qualquer ação ■ filterByLabel: rótulo adicional para filtragem da ação OBTER usando K8S labelSelector <p>OBTER, CRIAR, EXCLUIR, APLICAR</p> <ul style="list-style-type: none"> ■ yaml: YAML inline para processar e enviar ao Kubernetes ■ parameters: par de CHAVE, VALOR - Substitua \$\$KEY por VALOR na área de entrada do YAML inline ■ filePath: caminho relativo do endpoint SCM Git, se fornecido, a partir do qual obter o YAML ■ scmConstants: par de CHAVE, VALOR - Substitua \$\$KEY por VALOR no YAML obtido no SCM. ■ continueOnConflict: quando definido como verdadeiro, se um recurso já estiver presente, a tarefa continuará. <p>REVERTER</p> <ul style="list-style-type: none"> ■ resourceType: tipo de recurso para reverter ■ resourceName: nome do recurso para reverter ■ namespace: namespace no qual a reversão deve ser realizada ■ revision: revisão para a qual reverter 	<p><code>\${MY_STAGE.MY_TASK.input.action}</code> # Determina a ação a ser executada.</p> <p><code>\${MY_STAGE.MY_TASK.input.timeout}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.filterByLabel}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.yaml}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.parameters}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.filePath}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.scmConstants}</code></p> <p><code>\$</code></p> <p><code>{MY_STAGE.MY_TASK.input.continueOnConflict}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.resourceType}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.resourceName}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.namespace}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.revision}</code></p>
	Output	<p>response: captura toda a resposta</p> <p>response.<RESOURCE>: o recurso corresponde a configMaps, implantações, endpoints, entradas, trabalhos, namespaces, pods, replicaSets, replicationControllers, segredos, serviços, statefulSets, nós, loadBalancers.</p> <p>response.<RESOURCE>.<KEY>: a chave corresponde a um de apiVersion, tipo, metadados, especificação</p>	<p><code>\${MY_STAGE.MY_TASK.output.response}</code></p> <p><code>\${MY_STAGE.MY_TASK.output.response. }</code></p>

Tabela 3-14. Integre aplicativos de desenvolvimento, teste e implantação

Tipo de tarefa	Escopo	Key	Como usar SCOPE e KEY na tarefa
Bamboo			
	Input	plan: nome do plano planKey: chave do plano variables: variáveis a serem transmitidas ao plano parameters: parâmetros a serem transmitidos ao plano	<pre> \${MY_STAGE.MY_TASK.input.plan} \${MY_STAGE.MY_TASK.input.planKey} \${MY_STAGE.MY_TASK.input.variables} \${MY_STAGE.MY_TASK.input.parameters} # Referência a todos os parâmetros \${MY_STAGE.MY_TASK.input.parameters.param1} # Referência ao valor de param1 </pre>
	Output	resultUrl: a URL da compilação resultante buildResultKey: a chave da compilação resultante buildNumber: número da compilação buildTestSummary: resumo da execução dos testes successfulTestCount: resultado do teste aprovado failedTestCount: resultado do teste reprovado skippedTestCount: resultado de teste ignorado artifacts: artefatos da compilação	<pre> \${MY_STAGE.MY_TASK.output.resultUrl} \${MY_STAGE.MY_TASK.output.buildResultKey} \${MY_STAGE.MY_TASK.output.buildNumber} \${MY_STAGE.MY_TASK.output.buildTestSummary} # Referência a todos os resultados \${MY_STAGE.MY_TASK.output.successfulTestCount} # Referência à contagem de testes específica \${MY_STAGE.MY_TASK.output.buildNumber} </pre>
Jenkins			
	Input	job: nome do trabalho do Jenkins parameters: parâmetros a serem transmitidos ao trabalho	<pre> \${MY_STAGE.MY_TASK.input.job} \${MY_STAGE.MY_TASK.input.parameters} # Referência a todos os parâmetros \${MY_STAGE.MY_TASK.input.parameters.param1} # Referência ao valor de um parâmetro </pre>
	Output	job: nome do trabalho do Jenkins jobId: ID do trabalho resultante, como 1234 jobStatus: status no Jenkins jobResults: coleção de resultados de cobertura de teste/código jobUrl: URL da execução do trabalho resultante	<pre> \${MY_STAGE.MY_TASK.output.job} \${MY_STAGE.MY_TASK.output.jobId} \${MY_STAGE.MY_TASK.output.jobStatus} \${MY_STAGE.MY_TASK.output.jobResults} # Referência a todos os resultados \${MY_STAGE.MY_TASK.output.jobResults.junitResponse} # Referência aos resultados do JUnit \${MY_STAGE.MY_TASK.output.jobResults.jacocoResponse} # Referência aos resultados do JaCoCo \${MY_STAGE.MY_TASK.output.jobUrl} </pre>
TFS			

Tabela 3-14. Integre aplicativos de desenvolvimento, teste e implantação (continuação)

Tipo de tarefa	Escopo	Key	Como usar SCOPE e KEY na tarefa
	Input	projectCollection: coleção de projetos do TFS teamProject: projeto selecionado da coleção disponível buildDefinitionId: ID de Definição da Compilação a ser executado	<code>\${MY_STAGE.MY_TASK.input.projectCollection}</code> <code>\${MY_STAGE.MY_TASK.input.teamProject}</code> <code>\${MY_STAGE.MY_TASK.input.buildDefinitionId}</code>
	Output	buildId: ID da compilação resultante buildUrl: URL para visitar o resumo da compilação logUrl: URL para visitar logs dropLocation: local de destino dos artefatos, se houver	<code>\${MY_STAGE.MY_TASK.output.buildId}</code> <code>\${MY_STAGE.MY_TASK.output.buildUrl}</code> <code>\${MY_STAGE.MY_TASK.output.logUrl}</code> <code>\${MY_STAGE.MY_TASK.output.dropLocation}</code>
vRO			
	Input	workflowId: ID do fluxo de trabalho a ser executado parameters: parâmetros a serem transmitidos ao fluxo de trabalho	<code>\${MY_STAGE.MY_TASK.input.workflowId}</code> <code>\${MY_STAGE.MY_TASK.input.parameters}</code>
	Output	workflowExecutionId: ID da execução do fluxo de trabalho properties: propriedades de saída da execução do fluxo de trabalho	<code>\${MY_STAGE.MY_TASK.output.workflowExecutionId}</code> <code>\${MY_STAGE.MY_TASK.output.properties}</code>

Tabela 3-15. Integre outros aplicativos por meio de uma API

Tipo de tarefa	Escopo	Key	Como usar SCOPE e KEY na tarefa
REST			
	Input	url: URL para chamar action: método HTTP a ser usado headers: cabeçalhos HTTP a serem aprovados payload: carga útil da solicitação fingerprint: impressão digital a ser correspondida para a uma URL que é https allowAllCerts: quando definido como "true", pode ser qualquer certificado que tenha uma URL https	<code>\${MY_STAGE.MY_TASK.input.url}</code> <code>\${MY_STAGE.MY_TASK.input.action}</code> <code>\${MY_STAGE.MY_TASK.input.headers}</code> <code>\${MY_STAGE.MY_TASK.input.payload}</code> <code>\${MY_STAGE.MY_TASK.input.fingerprint}</code> <code>\${MY_STAGE.MY_TASK.input.allowAllCerts}</code>

Tabela 3-15. Integre outros aplicativos por meio de uma API (continuação)

Tipo de tarefa	Escopo	Key	Como usar SCOPE e KEY na tarefa
	Output	<p>responseCode: código da resposta HTTP</p> <p>responseHeaders: cabeçalhos da resposta HTTP</p> <p>responseBody: formato de cadeia de caracteres da resposta recebida</p> <p>responseJson: resposta de passagem se o tipo de conteúdo for application/json</p>	<p><code>\${MY_STAGE.MY_TASK.output.responseCode}</code></p> <p><code>\${MY_STAGE.MY_TASK.output.responseHeaders}</code></p> <p><code>\$</code></p> <p><code>{MY_STAGE.MY_TASK.output.responseHeaders.header1}</code> # Referência ao cabeçalho de resposta "header1"</p> <p><code>\${MY_STAGE.MY_TASK.output.responseBody}</code></p> <p><code>\${MY_STAGE.MY_TASK.output.responseJson}</code> # Referência à resposta como JSON</p> <p><code>\${MY_STAGE.MY_TASK.output.responseJson.a.b.c}</code> # Referência a um objeto aninhado após o caminho JSON a.b.c na resposta</p>
Sondagem			
	Input	<p>url: URL para chamar</p> <p>headers: cabeçalhos HTTP a serem aprovados</p> <p>exitCriteria: critérios a serem atendidos para que a tarefa seja bem-sucedida ou falhe. Um par de chave/valor de "success" → Expressão "failure" → Expressão</p> <p>pollCount: número de iterações a serem executadas</p> <p>pollIntervalSeconds: número de segundos para aguardar entre cada iteração</p> <p>ignoreFailure: quando definido como "true", ignora as falhas de resposta intermediárias</p> <p>fingerprint: impressão digital a ser correspondida para a uma URL que é https</p> <p>allowAllCerts: quando definido como "true", pode ser qualquer certificado que tenha uma URL https</p>	<p><code>\${MY_STAGE.MY_TASK.input.url}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.headers}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.exitCriteria}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.pollCount}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.pollIntervalSeconds}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.ignoreFailure}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.fingerprint}</code></p> <p><code>\${MY_STAGE.MY_TASK.input.allowAllCerts}</code></p>
	Output	<p>responseCode: código da resposta HTTP</p> <p>responseBody: formato de cadeia de caracteres da resposta recebida</p> <p>responseJson: resposta de passagem se o tipo de conteúdo for application/json</p>	<p><code>\${MY_STAGE.MY_TASK.output.responseCode}</code></p> <p><code>\${MY_STAGE.MY_TASK.output.responseBody}</code></p> <p><code>\${MY_STAGE.MY_TASK.output.responseJson}</code> # Refer to response as JSON</p>

Tabela 3-16. Execute scripts remotos e definidos pelo usuário

Tipo de tarefa	Escopo	Key	Como usar SCOPE e KEY na tarefa
PowerShell			
	Input	<p>host: endereço IP ou nome do host da máquina</p> <p>username: nome de usuário a ser usado para conexão</p> <p>password: senha a ser usada para conexão</p> <p>useTLS: tentar conexão https</p> <p>trustCert: quando definido como "true", confia em certificados autoassinados</p> <p>script: script a ser executado</p> <p>workingDirectory: caminho do diretório para o qual alternar antes de executar o script</p> <p>environmentVariables: um par de chave/valor de variável de ambiente a ser definido</p> <p>arguments: argumentos a serem transmitidos ao script</p>	<pre> \${MY_STAGE.MY_TASK.input.host} \${MY_STAGE.MY_TASK.input.username} \${MY_STAGE.MY_TASK.input.password} \${MY_STAGE.MY_TASK.input.useTLS} \${MY_STAGE.MY_TASK.input.trustCert} \${MY_STAGE.MY_TASK.input.script} \$ {MY_STAGE.MY_TASK.input.workingDirectory} } \$ {MY_STAGE.MY_TASK.input.environmentVariables} \${MY_STAGE.MY_TASK.input.arguments} </pre>
	Output	<p>response: conteúdo do arquivo \$SCRIPT_RESPONSE_FILE</p> <p>responseFilePath: valor de \$SCRIPT_RESPONSE_FILE</p> <p>exitCode: código de saída do processo</p> <p>logFilePath: caminho para o arquivo contendo stdout</p> <p>errorFilePath: caminho para o arquivo contendo stderr</p>	<pre> \${MY_STAGE.MY_TASK.output.response} \$ {MY_STAGE.MY_TASK.output.responseFilePath} \${MY_STAGE.MY_TASK.output.exitCode} \${MY_STAGE.MY_TASK.output.logFilePath} \${MY_STAGE.MY_TASK.output.errorFilePath} </pre>
SSH			

Tabela 3-16. Execute scripts remotos e definidos pelo usuário (continuação)

Tipo de tarefa	Escopo	Key	Como usar SCOPE e KEY na tarefa
	Input	host: endereço IP ou nome do host da máquina username: nome de usuário a ser usado para conexão password: senha a ser usada para conexão (opcionalmente, é possível usar privateKey) privateKey: chave privada a ser usada para conexão passphrase: senha opcional para desbloquear privateKey script: script a ser executado workingDirectory: caminho do diretório para o qual alternar antes de executar o script environmentVariables: par de chave/valor da variável de ambiente a ser definida	<pre> \${MY_STAGE.MY_TASK.input.host} \${MY_STAGE.MY_TASK.input.username} \${MY_STAGE.MY_TASK.input.password} \${MY_STAGE.MY_TASK.input.privateKey} \${MY_STAGE.MY_TASK.input.passphrase} \${MY_STAGE.MY_TASK.input.script} \$ {MY_STAGE.MY_TASK.input.workingDirectory} } \$ {MY_STAGE.MY_TASK.input.environmentVariables} </pre>
	Output	response: conteúdo do arquivo \$SCRIPT_RESPONSE_FILE responseFilePath: valor de \$SCRIPT_RESPONSE_FILE exitCode: código de saída do processo logFilePath: caminho para o arquivo contendo stdout errorFilePath: caminho para o arquivo contendo stderr	<pre> \${MY_STAGE.MY_TASK.output.response} \$ {MY_STAGE.MY_TASK.output.responseFilePath} \${MY_STAGE.MY_TASK.output.exitCode} \${MY_STAGE.MY_TASK.output.logFilePath} \${MY_STAGE.MY_TASK.output.errorFilePath} </pre>

Como usar uma associação de variáveis entre tarefas

Este exemplo mostra como usar associações de variáveis nas suas tarefas de pipeline.

Tabela 3-17. Exemplos de formatos de sintaxe

Exemplo	Sintaxe
Para usar um valor de saída de tarefa para notificações de pipeline e propriedades de saída de pipeline	<code>\${<Stage Key>.<Task Key>.output.<Task output key>}</code>
Para fazer referência ao valor de saída da tarefa anterior como entrada para a tarefa atual	<code>\${<Previous/Current Stage key>.<Previous task key not in current Task group>.output.<task output key>}</code>

Para saber mais

Para saber mais sobre como associar variáveis, consulte:

- [Como usar associações de variáveis em pipelines do vRealize Automation Code Stream](#)
- [Como usar associações de variáveis em uma tarefa de condição para executar ou parar um pipeline no vRealize Automation Code Stream](#)
- [Quais tipos de tarefa estão disponíveis no vRealize Automation Code Stream](#)

Como enviar notificações sobre meu pipeline no vRealize Automation Code Stream

Notificações são maneiras de se comunicar com suas equipes e informá-las sobre o status dos pipelines no vRealize Automation Code Stream.

É possível configurar o vRealize Automation Code Stream para enviar notificações quando um preparar for executado, com base no status de todo o pipeline, estágio ou tarefa.

- Uma notificação por e-mail envia um e-mail em:
 - Conclusão, espera, falha, cancelamento ou início do pipeline.
 - Conclusão, falha ou início do estágio.
 - Conclusão, espera, falha ou início da tarefa.
- Uma notificação de tíquete cria um tíquete e o atribui a um membro da equipe em:
 - Falha ou conclusão do pipeline.
 - Falha do estágio.
 - Falha da tarefa.
- Uma notificação de webhook envia uma solicitação a outro aplicativo em:
 - Falha, conclusão, espera, cancelamento ou início do pipeline.
 - Falha, conclusão ou início do estágio.
 - Falha, conclusão, espera ou início da tarefa.

Por exemplo, você pode configurar uma notificação por e-mail sobre uma tarefa de operação do usuário para obter aprovação em um ponto específico do seu pipeline. Quando o pipeline for executado, essa tarefa enviará um e-mail para a pessoa que deve aprová-la. Se a tarefa Operação do Usuário tiver um tempo limite de expiração definido em dias, horas ou minutos, o usuário necessário deverá aprovar o pipeline antes da expiração da tarefa. Caso contrário, o pipeline falhará conforme esperado.

Também é possível configurar uma notificação para criar um tíquete do Jira quando uma tarefa de pipeline falhar. Ou você pode configurar uma notificação de webhook para enviar uma solicitação para um canal do Slack sobre o status de um pipeline com base no evento de pipeline.

Você pode usar variáveis em todos os tipos de notificações. Por exemplo, é possível usar `${var}` na URL de uma notificação de Webhook.

Pré-requisitos

- Verifique se um ou mais pipelines foram criados. Veja os casos de uso em [Capítulo 5 Tutoriais para usar o vRealize Automation Code Stream](#).
- Para enviar notificações por e-mail, confirme se é possível acessar um servidor de e-mail ativo. Para obter ajuda, consulte o administrador.
- Para criar tíquetes, como um tíquete do Jira, confirme se o endpoint existe. Consulte [O que são endpoints no vRealize Automation Code Stream](#).
- Para enviar uma notificação com base em uma integração, crie uma notificação de webhook. Em seguida, confirme se o webhook foi adicionado e está funcionando. Você pode usar notificações com aplicativos como o Slack, o GitHub ou o GitLab.

Procedimentos

- 1 Abra um pipeline.
- 2 Para criar uma notificação para o status geral do pipeline, ou o status de um estágio ou tarefa:

Para criar uma notificação com base em...	O que fazer...
Status do pipeline	Clique em uma área em branco na tela do pipeline.
Status de um estágio	Clique em uma área em branco em um estágio do pipeline.
Status de uma tarefa	Clique em uma tarefa em um estágio do pipeline.

- 3 Clique na guia **Notificações**.
- 4 Clique em **Adicionar**, selecione o tipo de notificação e configure os detalhes da notificação.
- 5 Para criar uma notificação de Slack quando um pipeline for bem-sucedido, crie uma notificação de webhook.
 - a Selecione **Webhook**.
 - b Insira as informações para configurar a notificação de Slack.
 - c Clique em **Salvar**.
 - d Quando o pipeline é executado, o canal de Slack recebe a notificação sobre o status do pipeline. Por exemplo, os usuários podem ver o seguinte no canal de Slack:

```
Codestream APP [12:01 AM]
Tested by User1 - Staging Pipeline 'User1-Pipeline', Pipeline ID
'e9b5884d809ce2755728177f70f8a' succeeded
```

- 6 Para criar um tíquete do Jira, configure as informações do tíquete.
 - a Selecione **Tíquete**.
 - b Insira as informações para configurar a notificação do Jira.
 - c Clique em **Salvar**.

Notification

Send notification type ☐ Email ☒ Ticket ☐ Webhook

When pipeline ☒ Fails ☐ Completes

Jira endpoint

Create Ticket

Jira project

Issue type

Assignee

Summary

Description

Resultados

Parabéns! Você aprendeu que pode criar vários tipos de notificações em diversas áreas do pipeline no vRealize Automation Code Stream.

Próximo passo

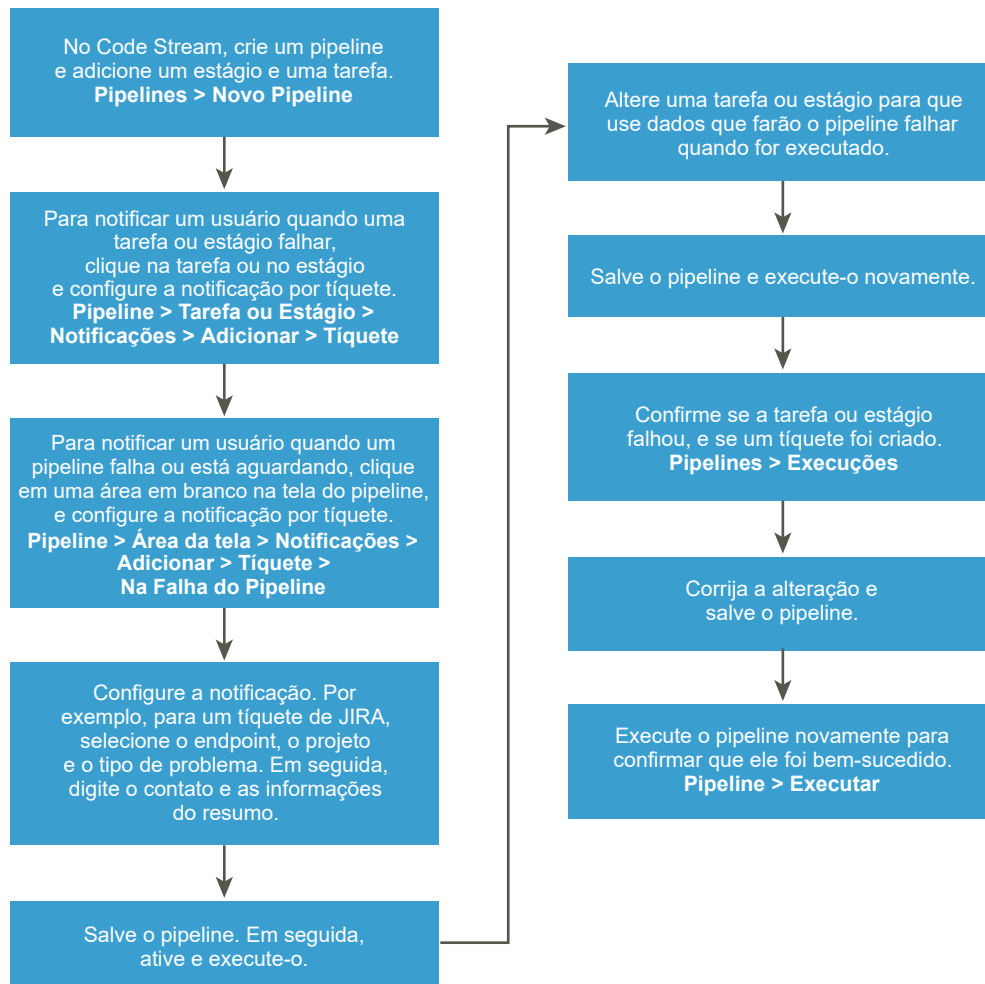
Para obter um exemplo detalhado sobre como criar uma notificação, consulte [Como criar um tíquete do Jira no vRealize Automation Code Stream quando uma tarefa de pipeline falhar](#).

Como criar um tíquete do Jira no vRealize Automation Code Stream quando uma tarefa de pipeline falhar

Se um estágio ou uma tarefa de pipeline falhar, será possível fazer com que o vRealize Automation Code Stream crie um tíquete do Jira. Você pode atribuir esse tíquete à pessoa que deve resolver o problema. Também é possível criar um tíquete quando o pipeline estiver aguardando ou quando for bem-sucedido.

Você pode adicionar e configurar notificações em uma tarefa, um estágio ou um pipeline. O vRealize Automation Code Stream cria o tíquete com base no status da tarefa, do estágio ou do pipeline no qual você adicionou a notificação. Por exemplo, se um endpoint não estiver disponível, será possível fazer com que o vRealize Automation Code Stream crie um tíquete do Jira para a tarefa que falha porque ela não pode se conectar ao endpoint.

Também é possível criar notificações quando o pipeline for bem-sucedido. Por exemplo, você pode informar sua equipe de controle de qualidade sobre pipelines bem-sucedidos, para que ela possa confirmar a compilação e executar um pipeline de teste diferente. Ou você pode informar sua equipe de desempenho para que ela seja capaz de medir o desempenho do pipeline e se preparar para uma atualização para validação ou produção.



Este exemplo cria um tíquete do Jira quando uma tarefa de pipeline falha.

Pré-requisitos

- Verifique se você tem uma conta válida no Jira e poderá fazer login na instância do Jira.
- Verifique a existência de um endpoint do Jira e se ele está ativo.

Procedimentos

- 1 No pipeline, clique em uma tarefa.
- 2 Na área de configuração de tarefas, clique em **Notificações**.
- 3 Clique em **Adicionar** e configure as informações do tíquete.
 - a Clique em **Tíquete**.
 - b Selecione o endpoint do Jira.
 - c Digite o projeto Jira e o tipo de problema.
 - d Insira o endereço de e-mail da pessoa que recebe o tíquete.
 - e Digite um resumo e uma descrição do tíquete e clique em **Salvar**.

Notification

Send notification type

☐ Email
 ☒ Ticket
 ☐ Webhook

When task *

☒ Fails

Jira endpoint *

TestJira

▼

Create Ticket

Jira project *

YourProject

Issue type *

Bug

Assignee *

username@yourcompany.com

Summary \$ *

CI task failed

Description \$

Research and correct

CANCEL

SAVE

- 4 Salve o pipeline, em seguida, ative e execute-o.
- 5 Teste o tíquete.
 - a Altere as informações da tarefa para incluir dados que façam a tarefa falhar.
 - b Salve o pipeline e execute-o novamente.

- c Clique em **Execuções** e confirme que o pipeline falhou.
- d Na execução, confirme que o vRealize Automation Code Stream criou o tíquete e o enviou.
- e Altere as informações da tarefa novamente para corrigi-la e execute o pipeline novamente, para verificar se ela é bem-sucedida.

Resultados

Parabéns! O vRealize Automation Code Stream criou um tíquete do Jira quando a tarefa de pipeline falhou e o atribuiu à pessoa responsável pela sua solução.

Próximo passo

Continue a adicionar notificações para alertar sua equipe sobre os pipelines.

Como faço para reverter minha implantação no vRealize Automation Code Stream

Configure a reversão como um pipeline com tarefas que retornam sua implantação a um estado estável anterior após uma falha em um pipeline de implantação. Anexe o pipeline de reversão a tarefas ou estágios que deseja reverter em caso de falha.

Dependendo da sua função, os motivos para a reversão podem variar.

- Como engenheiro de liberação, quero que o vRealize Automation Code Stream verifique o sucesso durante uma liberação para eu saber se devo continuar com a versão ou revertê-la. Possíveis falhas incluem falha na tarefa, uma rejeição em UserOps, excedendo o limite de métricas.
- Como proprietário do ambiente, desejo reimplantar uma versão anterior para poder retornar rapidamente um ambiente para um estado em boas condições.
- Como proprietário do ambiente, quero oferecer suporte à reversão de uma implantação de Azul-Verde para que eu possa minimizar o tempo de inatividade das versões com falha.

Ao usar um modelo de pipeline inteligente para criar um pipeline de CD com a opção reversão selecionada, a reversão é automaticamente adicionada às tarefas no pipeline. Nesse caso de uso, o modelo de pipeline inteligente será usado para definir a reversão de uma implantação de aplicativo em um cluster Kubernetes usando o modelo de implantação de upgrade contínuo. O modelo de pipeline inteligente cria um pipeline de implantação e um ou mais pipelines de reversão.

- No pipeline de implantação, a reversão será necessária se as tarefas de implantação de atualização ou de implantação de verificação falharem.
- No pipeline de reversão, a implantação é atualizada com uma imagem antiga.

Também é possível criar manualmente um pipeline de reversão usando um modelo em branco. Antes de criar um pipeline de reversão, é necessário planejar o fluxo de reversão. Para obter mais informações básicas sobre a reversão, consulte [Planejando uma reversão no vRealize Automation Code Stream](#).

Pré-requisitos


- Verifique se você é membro de um projeto no vRealize Automation Code Stream. Se não for, peça a um administrador do vRealize Automation Code Stream para adicioná-lo como membro de um projeto. Consulte [Como adicionar um projeto no vRealize Automation Code Stream](#).
- Configure os clusters do Kubernetes em que o pipeline implantará seu aplicativo. Configure um cluster de desenvolvimento e um cluster de produção.
- Crie os endpoints de produção e desenvolvimento do Kubernetes que implantam a imagem do aplicativo nos clusters do Kubernetes.
- Verifique se há uma configuração de registro do Docker.
- Verifique se há um arquivo YAML do Kubernetes para aplicar à implantação.
- Familiarize-se com o modelo de pipeline inteligente de CD. Consulte [Como planejar uma compilação nativa de CD no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente](#).



Procedimentos


- 1 Clique em **Pipelines > Novo Pipeline > Modelo Inteligente > Entrega Contínua**.
- 2 Digite as informações no modelo de pipeline inteligente.
 - a Selecione um projeto.
 - b Digite o nome do pipeline, como **RollingUpgrade-Example**.
 - c Selecione os ambientes para seu aplicativo. Para adicionar a reversão à implantação, é necessário selecionar **Prod..**
 - d Clique em **Selecionar**, escolha um arquivo YAML do Kubernetes e clique em **Processar**.
O modelo de pipeline inteligente exibe os serviços e os ambientes de implantação disponíveis.
 - e Selecione o serviço que o pipeline usará para a implantação.
 - f Selecione os endpoints do cluster para ambientes de desenvolvimento e produção.
 - g Para a origem da imagem, selecione **Entrada do tempo de execução do pipeline**.
 - h Para o modelo de implantação, selecione **Atualização Contínua**.




- i Clique em **Reverter**.
- j Forneça a **URL de verificação de integridade**.


Smart Template: Continuous Delivery

Endpoint prerequisites  Kubernetes Docker Registry

Project  test1 


Pipeline name  RollbackUpgrade-Example

Environment   Development  Production

Kubernetes YAML files  SELECT PROCESS

Processed files: cdTemplate.yaml

Select service

Deployment name	Service	Namespace	Image
 codestream-demo	codestream-demo	bgreen	symphony-tango-beta2.jfrog.io/codestream-demo

1 selected

Deployment







Environment	Cluster Endpoint	Namespace
Development	Dev-VKE-Cluster 	bgreen-596788
Production	Prod-VKE-Cluster 	bgreen

Image source  ☐ Docker trigger ☒ Pipeline runtime input

Deployment model  ☐ Canary ☒ Rolling upgrade ☐ Blue-Green

Rollback 

Health check URL  /health-check.json

CREATE CANCEL

- 3 Para criar o pipeline chamado RollbackUpgrade-Example, clique em **Criar**.

O pipeline RollbackUpgrade-Example aparece com o ícone de reversão nas tarefas dos estágios de desenvolvimento e produção que podem ser revertidos.

RollbackUpgrade-Example
Disabled

Workspace
Input
Model
Output

>>

Development

```

graph LR
    A[Create Namespace  
Kubernetes] --> B[Create Secret  
Kubernetes]
    B --> C[Create Service  
Kubernetes]
            
```

Production

```

graph LR
    D[Update Deployment  
Kubernetes] --> E[Verify Deployment  
ROLLBACK]
            
```

Task : Create Secret

Notifications

Rollback ⚡

Task name ⓘ *

Type * Kubernetes ▾

Continue on failure ☐

Execute task ☒ Always ☐ On condition

Kubernetes Task Properties

Kubernetes cluster * Dev-VKE-Cluster ▾

Timeout (in Mins) * 5

Action * ☐ Get ☒ Create ☐ Apply ☐ Delete ☐ Rollout

Continue on conflict ☒

Payload source * ☐ Source control ☒ Local definition

Local YAML definition ⓘ *

FILE

```

1 apiVersion: v1
2 data:
3   .dockercfg: eyJZeWlwaG9ueS10YwNby1lZXR...
4 kind: Secret
5 metadata:
6   name: jfrog-beta2
7   namespace: bgreen-549938
8   type: kubernetes.io/dockercfg
            
```

Parameters

Key	Value

Output Parameters

status
response

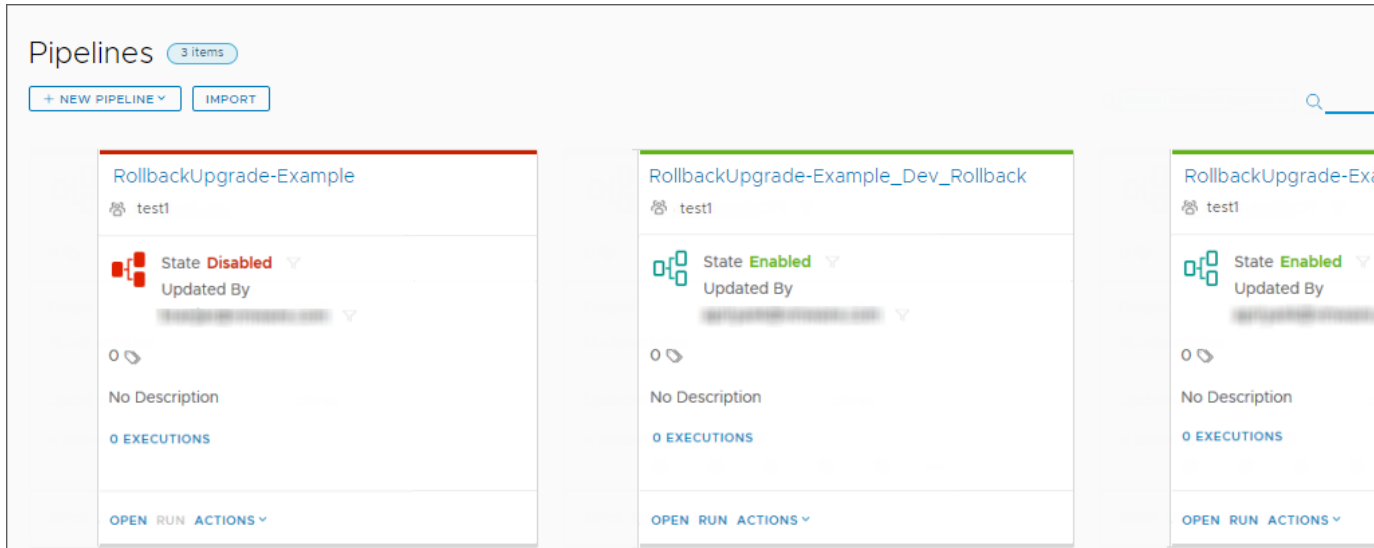
EDIT
RUN
CLOSE
Last saved 9 minutes ago

4 Feche o pipeline.

Na página Pipelines, serão exibidos o pipeline criado e um novo pipeline para cada estágio do seu pipeline.

- RollingUpgrade-Example. O vRealize Automation Code Stream desativa o pipeline que você criou por padrão, o que garante que você possa revisá-lo antes de o executar.
- RollingUpgrade-Example_Dev_Rollback. Esse pipeline de desenvolvimento de reversão é invocado na falha de tarefas no estágio de desenvolvimento, como criar serviço, criar segredo, criar implantação, verificar implantação. O pipeline de desenvolvimento de reversão é ativado por padrão para garantir a reversão das tarefas de desenvolvimento.

- RollingUpgrade-Example_Prod_Rollback. Esse pipeline de produção de reversão é invocado na falha de tarefas no estágio de produção, como Implantar fase 1, Verificar fase 1, Implantar fase de reversão, Concluir fase de reversão, Verificar fase de reversão. O pipeline de produção de reversão é ativado por padrão para garantir a reversão das tarefas de produção.




5 Ative e execute o pipeline criado.

Ao iniciar a execução, serão solicitados os parâmetros de entrada. Forneça a imagem e a tag para o endpoint no repositório Docker que está sendo usado.

6 Na página Execuções, selecione **Ações > Exibir Execução** para observar a execução do pipeline.

O pipeline começa a **EXECUÇÃO** e passa pelas tarefas de estágio de desenvolvimento. Se o pipeline não conseguir executar uma tarefa durante o estágio de desenvolvimento, o pipeline chamado RollingUpgrade-Example_Dev_Rollback será disparado para reverter a implantação e o status do pipeline será alterado para **ROLLING_BACK**.

[< BACK](#)


RollbackUpgrade-Example #1
ROLLING_BACK
0
ACTIONS ▾

● Development

✓ Create Namespace
✓ Create Secret
✓ Create Service
● Create Deployment
● Verify Deployment

Project	test1
Execution	RollbackUpgrade-Example #1
Status	ROLLING_BACK RUNNING
Updated by	
Executed by	
Duration	12m 9s 186ms (01/11/2019 1:24 PM -)
Input Parameters ▾	
image	demo-image-cs
tag	latest
Workspace	
Details not available	
Output Parameters ▾	
The Execution did not output any properties	

Após a reversão, a página Execuções listará duas execuções do pipeline RollingUpgrade-Example.

- O pipeline que você criou e que foi revertido exibe **ROLLBACK_COMPLETED**.
- O pipeline de desenvolvimento de reversão que foi disparado para realizar a reversão exibe **CONCLUÍDO**.

Executions 604 items

+ NEW EXECUTION

RollbackUpgrade-Example_Dev...#1

COMPLETED

Stages:

1 Rollback for RollbackUpgrade-Example#1

By on 01/11/2019 1:36 PM
Execution Completed.
Comments:Triggered to rollback Development.Create Deployment of RollbackUpgrade-E

RollbackUpgrade-Example#1

ROLLBACK_COMPLETED

Stages:

0

By on 01/11/2019 1:24 PM
Create Deployment ROLLBACK_COMPLETED

Resultados

Parabéns! Você definiu com sucesso um pipeline com reversão e observou o vRealize Automation Code Stream reverter o pipeline no ponto de falha.

Planejamento para compilar, integrar e entregar seu código de forma nativa no vRealize Automation Code Stream

Antes que o vRealize Automation Code Stream compile, integre e entregue seu código usando o recurso nativo que cria um pipeline de CICD, CI ou CD para você, planeje sua compilação nativa. Em seguida, você poderá criar seu pipeline usando um dos modelos de pipeline inteligentes ou adicionando manualmente os estágios e as tarefas.

Fornecemos vários exemplos que mostram como planejar a sua compilação de integração e entrega contínuas. Esses planos descrevem os pré-requisitos que você precisará cumprir e as visões gerais para ajudá-lo a se preparar para usar o recurso de compilação nativo de forma eficiente para compilar seus pipelines.

Este capítulo inclui os seguintes tópicos:

- Como planejar uma compilação nativa de CICD no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente
- Como planejar uma compilação nativa de CI no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente
- Como planejar uma compilação nativa de CD no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente
- Planejando uma compilação nativa de CICD no vRealize Automation Code Stream antes de adicionar tarefas manualmente
- Planejando uma reversão no vRealize Automation Code Stream

Como planejar uma compilação nativa de CICD no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente

Para criar um pipeline de integração contínua e entrega contínua (CICD) no vRealize Automation Code Stream, é possível usar o modelo de pipeline inteligente de CICD. Para planejar a compilação nativa de CICD, colete as informações necessárias para preencher o modelo de pipeline inteligente antes de usá-lo para criar o pipeline neste plano de exemplo.

Depois que você inserir as informações no modelo de pipeline inteligente e salvá-las, o modelo criará um pipeline que inclui estágios e tarefas. Ele também indicará onde implantar a imagem com base nos tipos de ambiente selecionados, como Des e Prod. O pipeline publicará a imagem do Docker e realizará as ações obrigatórias para executá-la. Após a execução do pipeline, você poderá monitorar as tendências nas execuções de pipeline.

Para criar um pipeline de CICD, é necessário planejar os estágios de integração contínua (CI) e entrega contínua (CD) do pipeline.

Quando um pipeline inclui uma imagem do Docker Hub, você deve certificar-se de que essa imagem tenha uma cURL incorporada antes de executar o pipeline. Quando o pipeline é executado, o vRealize Automation Code Stream baixa um arquivo binário que usa essa cURL para executar comandos.

Como planejar o estágio de integração contínua (CI)

Para planejar o estágio de CI do seu pipeline, defina os requisitos externos e internos, e determine as informações a serem inseridas na parte CI do modelo de pipeline inteligente. Aqui está um resumo.

Endpoints e repositórios que serão necessários:

- Um repositório de código-fonte de Git no qual seus desenvolvedores verificam o código. O vRealize Automation Code Stream recebe o código mais recente para o pipeline quando os desenvolvedores confirmam as alterações.
- Um endpoint do Git para o repositório no qual reside o código-fonte do desenvolvedor.
- Um endpoint do Docker para o host de compilação do Docker que executará os comandos de compilação dentro de um contêiner.
- Um endpoint do Kubernetes para que o vRealize Automation Code Stream possa implantar sua imagem em um cluster do Kubernetes.
- Uma imagem do construtor que cria o contêiner no qual os testes de integração contínua são executados.
- Um endpoint de registro de imagem para que o host de compilação do Docker possa receber a imagem do construtor a partir dele.

Você precisará de acesso a um projeto. O projeto agrupa todo o seu trabalho, incluindo pipeline, endpoints e painéis. Verifique se você é membro de um projeto no vRealize Automation Code Stream. Se não for, peça a um administrador do vRealize Automation Code Stream para adicioná-lo como membro de um projeto. Consulte [Como adicionar um projeto no vRealize Automation Code Stream](#).

Você precisará de um webhook Git que permita que o vRealize Automation Code Stream use o gatilho Git para disparar o pipeline quando os desenvolvedores confirmam alterações de código. Consulte [Como usar o gatilho Git no vRealize Automation Code Stream para executar um pipeline](#).

Seus conjuntos de ferramentas de compilação:

- Seu tipo de compilação, como Maven.
- Todas as ferramentas de compilação pós-processo usadas, como JUnit, JaCoCo, Checkstyle e FindBugs.

Sua ferramenta de publicação:

- Uma ferramenta como o Docker que implantará seu contêiner de compilação.
- Uma tag de imagem, que é a ID de confirmação ou o número da compilação.

Seu espaço de trabalho de compilação:

- Um host de compilação do Docker, que é o endpoint do Docker.
- Um registro de imagem. A parte de CI do pipeline recebe a imagem do endpoint do registro selecionado. O contêiner executa as tarefas de CI e implanta sua imagem. Se o registro precisar de credenciais, será necessário primeiro criar um endpoint de registro de imagem e, em seguida, selecioná-lo aqui para que o host possa receber a imagem do registro.
- O URL da imagem do construtor que cria o contêiner no qual as tarefas de integração contínua são executadas.

Como planejar o estágio de entrega contínua (CD)

Para planejar o estágio de CD do seu pipeline, defina os requisitos externos e internos, e determine as informações a serem inseridas na parte CD do modelo de pipeline inteligente.

Endpoints que serão necessários:

- Um endpoint do Kubernetes para que o vRealize Automation Code Stream possa implantar sua imagem em um cluster do Kubernetes.

Arquivos e tipos de ambiente:

- Todos os tipos de ambiente em que o vRealize Automation Code Stream implantará seu aplicativo, como Des e Prod. O modelo de pipeline inteligente cria os estágios e as tarefas em seu pipeline com base nos tipos de ambiente selecionados.

Tabela 4-1. Estágios de pipeline criados pelo modelo de pipeline inteligente de CICD


Conteúdo do pipeline	O que ele faz
Estágio de compilação-publicação	Compila e testa seu código, cria a imagem do construtor e publica a imagem no host do Docker.
Estágio de desenvolvimento	Usa um cluster de desenvolvimento do Amazon Web Services (AWS) para criar e implantar a imagem. Neste estágio, é possível criar um namespace no cluster e criar uma chave secreta.
Estágio de produção	Usa uma versão de produção do VMware Tanzu Kubernetes Grid Integrated Edition (anteriormente conhecido como VMware Enterprise PKS) para implantar sua imagem em um cluster Kubernetes de produção.


- Um arquivo YAML do Kubernetes selecionado por você na seção de CD do modelo de pipeline inteligente de CICD.

Para aplicar o arquivo, clique em **Selecionar**, selecione o arquivo YAML do Kubernetes e clique em **Processar**. O modelo de pipeline inteligente exibe os serviços e os ambientes de implantação disponíveis. Selecione um serviço, o endpoint do cluster e a estratégia de implantação. Por exemplo, para usar o modelo de implantação Canário, selecione **Canário** e digite o percentual da fase de implantação.

Smart Template: CI/CD


Step 2 of 2

Environment  ☒ Development ☒ Production

Kubernetes YAML files  SELECT PROCESS

Processed files: codestream.yaml

Select service

Deployment name	Service	Namespace	Image
 codestream-demo	codestream-demo	codestream	https://codestream/Myapp

1 services

Deployment







Environment	Cluster Endpoint	Namespace
Development	Dev-AWS-Cluster 	codestream-454709
Production	Prod-AWS-Cluster 	codestream

Image source  ☐ Docker trigger ☒ Pipeline runtime input

Deployment model  ☒ Canary ☐ Rolling upgrade ☐ Blue-Green

Phase 1  %

Rollback ☐

Health check URL 

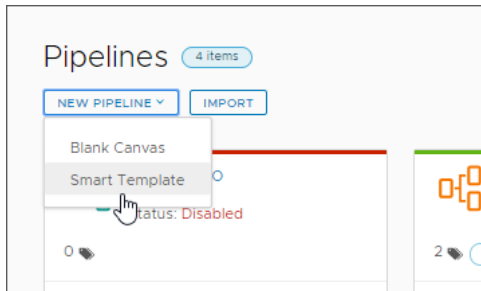
CREATE BACK CANCEL

Para ver um exemplo sobre como usar o modelo de pipeline inteligente para criar um pipeline para uma implantação Azul-Verde, consulte [Como implantar meu aplicativo no vRealize Automation Code Stream na implantação Azul-Verde](#).

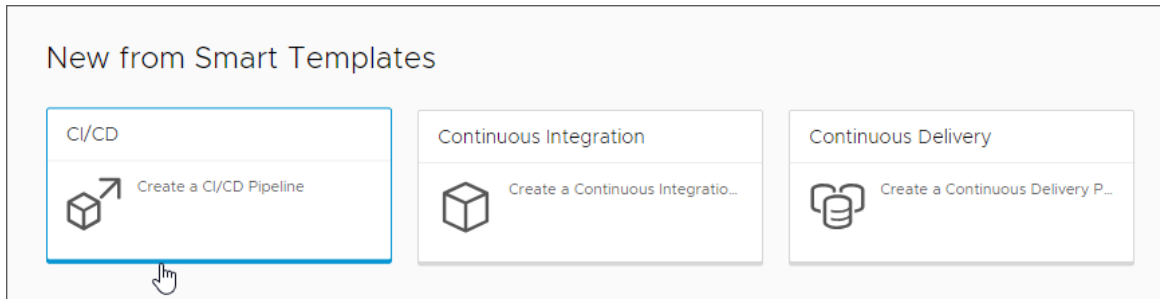
Como criar o pipeline de CI/CD usando o modelo de pipeline inteligente

Depois de reunir todas as informações e definir o que você precisa, veja como criar um pipeline a partir do modelo de pipeline inteligente de CI/CD.

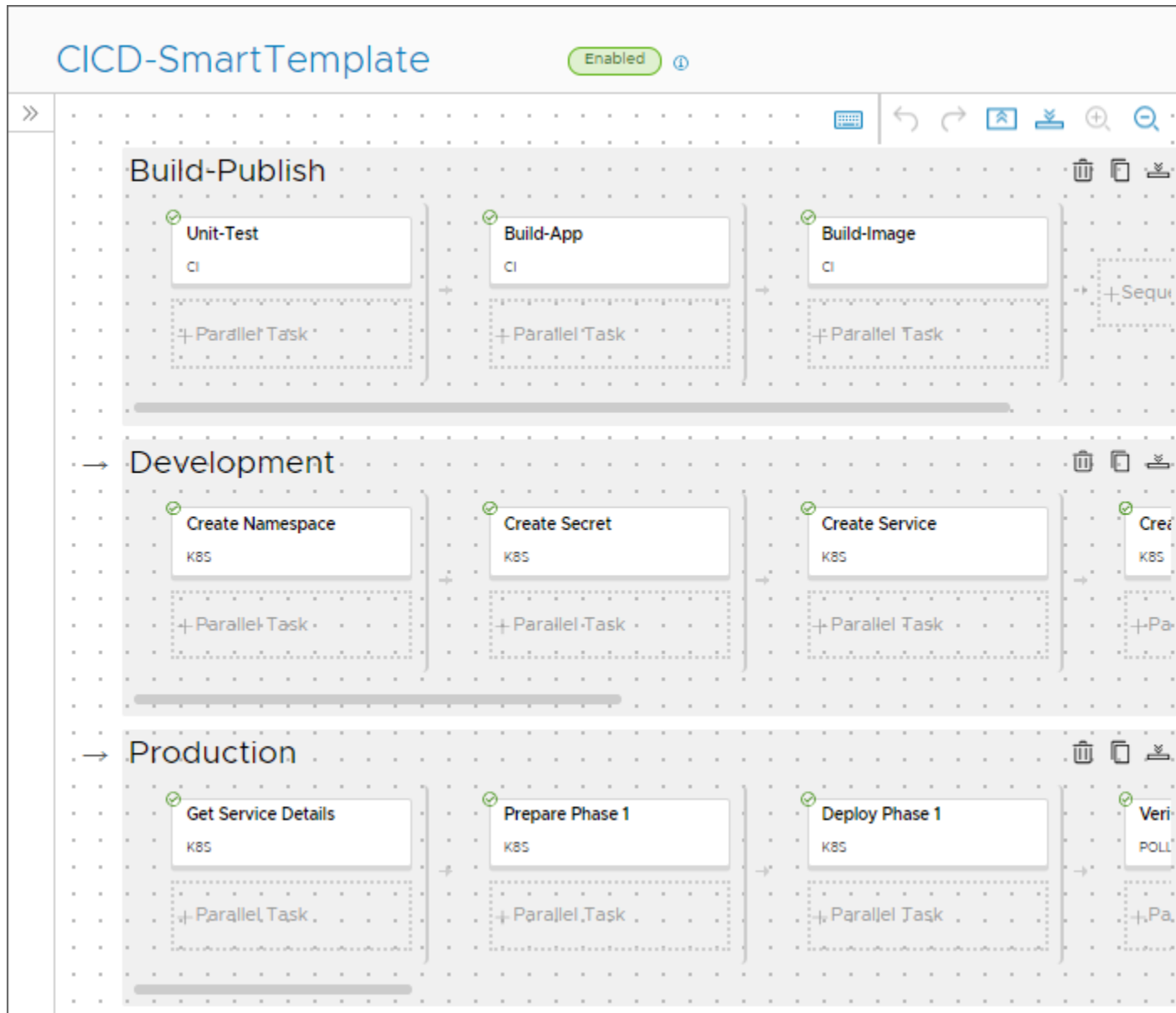
Em Pipelines, selecione **Novo Pipeline > Modelos Inteligentes**.



Você selecionará o modelo de pipeline inteligente de CI/CD.



Preencha o modelo e salve o pipeline com os estágios criados por ele. Se você precisar fazer quaisquer alterações finais, poderá editar o pipeline e salvá-lo.



Em seguida, ative e execute o pipeline. Após a execução, veja algumas coisas que você pode procurar:

- Verifique se o seu pipeline foi bem-sucedido. Clique em **Execuções** e pesquise por seu pipeline. Se falhou, corrija todos os erros e execute-o novamente.
- Verifique se o Git webhook está funcionando corretamente. A guia Git **Atividade** exibe os eventos. Clique em **Gatilhos > Git > Atividade**.
- Observe o painel de pipeline e examine as tendências. Clique em **Painéis** e pesquise pelo painel do seu pipeline. Também é possível criar um painel personalizado para relatar KPIs adicionais.

Para obter um exemplo detalhado, consulte [Como integrar continuamente o código do meu repositório do GitHub ou GitLab ao pipeline no vRealize Automation Code Stream](#).

Como planejar uma compilação nativa de CI no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente

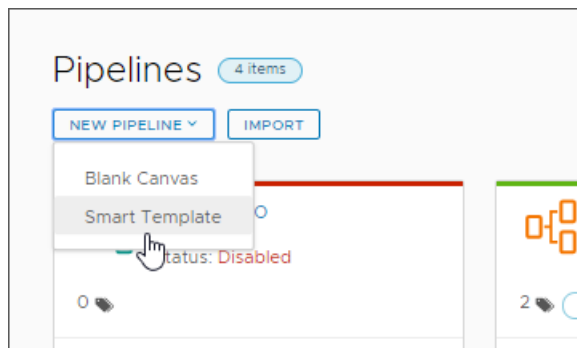
Para criar um pipeline de integração contínua (CI) no VMware Code Stream, você pode usar o modelo de pipeline inteligente de CI. Para planejar a compilação nativa de CI, colete as informações necessárias para preencher o modelo de pipeline inteligente antes de usá-lo para criar o pipeline neste plano de exemplo.

Ao preencher o modelo de pipeline inteligente, ele cria um pipeline de CI no repositório e executa as ações obrigatórias para executá-lo. Após a execução do pipeline, você poderá monitorar as tendências nas execuções de pipeline.

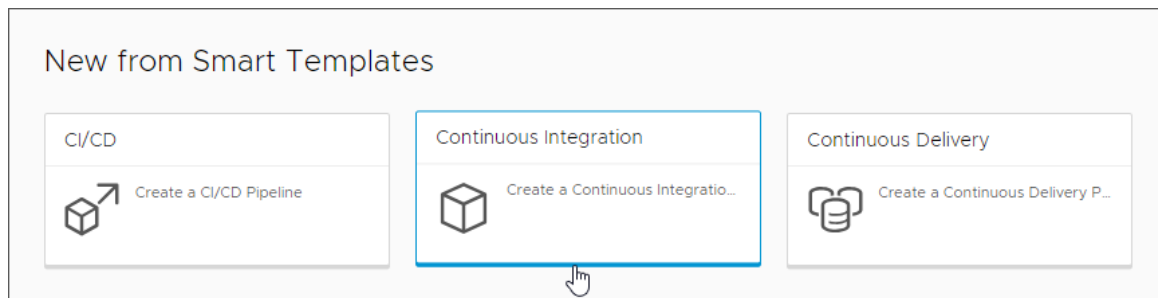
Para planejar sua compilação antes de usar o modelo de pipeline inteligente de CI, colete as informações para a compilação e, depois, siga a parte de CI de [Como planejar uma compilação nativa de CI/CD no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente](#).

Depois de reunir todas as informações e definir o que você precisa, veja como criar um pipeline a partir do modelo de pipeline inteligente de CI.

Em Pipelines, selecione **Modelos Inteligentes**.



Você selecionará o modelo de pipeline inteligente de CI.



Preencha o modelo e clique em **Criar** para salvar o pipeline com os estágios criados por ele.

É possível editar o pipeline para fazer quaisquer alterações finais que possam ser necessárias. Em seguida, você pode ativar o pipeline e executá-lo. Após a execução do pipeline, estes itens devem ser seguidos:

- Verifique se o seu pipeline foi bem-sucedido. Clique em **Execuções** e pesquise por seu pipeline. Se falhou, corrija todos os erros e execute-o novamente.
- Verifique se o Git webhook está funcionando corretamente. A guia Git **Atividade** exibe os eventos. Clique em **Gatilhos > Git > Atividade**.
- Observe o painel de pipeline e examine as tendências. Clique em **Painéis** e pesquise pelo painel do seu pipeline. Também é possível criar um painel personalizado para relatar KPIs adicionais.

Para obter um exemplo detalhado, consulte [Como integrar continuamente o código do meu repositório do GitHub ou GitLab ao pipeline no vRealize Automation Code Stream](#).

Como planejar uma compilação nativa de CD no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente

Para criar um pipeline de entrega contínua (CD) no vRealize Automation Code Stream, você pode usar o modelo de pipeline inteligente de CD. Para planejar a compilação nativa de CD, colete as informações necessárias para preencher o modelo de pipeline inteligente antes de usá-lo para criar o pipeline neste plano de exemplo.

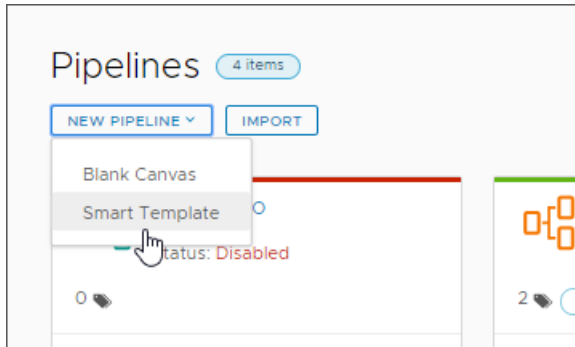
Ao preencher o modelo de pipeline inteligente, ele cria um pipeline de CD no repositório e executa as ações obrigatórias para executá-lo. Após a execução do pipeline, você poderá monitorar as tendências nas execuções de pipeline.

Para planejar sua compilação antes de usar o modelo de pipeline inteligente de CD, você fará o seguinte:

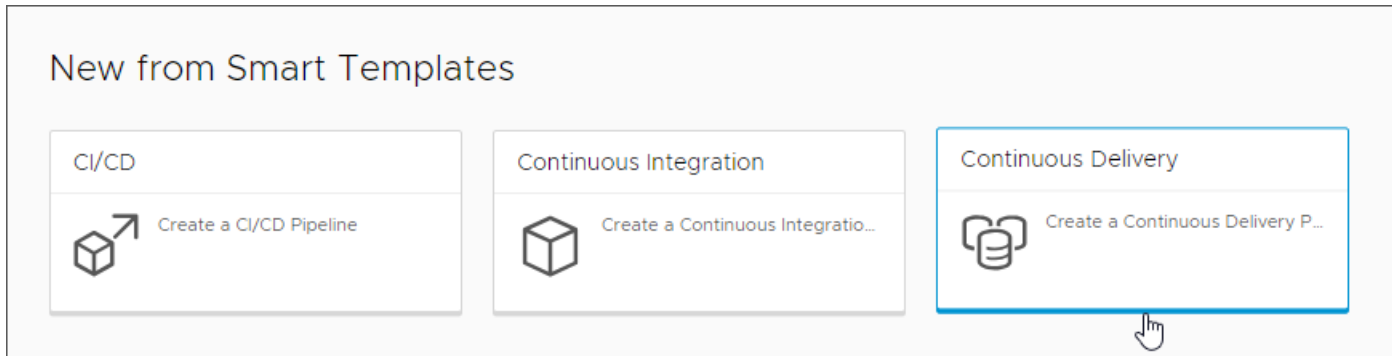
- Colete as informações para a compilação e siga a parte de CD de [Como planejar uma compilação nativa de CI/CD no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente](#).
- Adicione um endpoint do Kubernetes em que o vRealize Automation Code Stream implantará o contêiner.
- Identifique um projeto que agrupará todo o seu trabalho, incluindo o pipeline, os endpoints e os painéis.

Depois de reunir todas as informações e definir o que você precisa, veja como criar um pipeline a partir do modelo de pipeline inteligente de CD.

Em Pipelines, selecione **Modelos Inteligentes**.



Você selecionará o modelo de pipeline inteligente de CD.



Preencha o modelo, digite um nome para o pipeline e clique em **Criar** para salvar o pipeline com os estágios criados por ele.

É possível editar o pipeline para fazer quaisquer alterações finais que possam ser necessárias. Em seguida, você pode ativar o pipeline e executá-lo. Após a execução do pipeline, estes itens devem ser seguidos:

- Verifique se o seu pipeline foi bem-sucedido. Clique em **Execuções** e pesquise por seu pipeline. Se falhou, corrija todos os erros e execute-o novamente.
- Verifique se o Git webhook está funcionando corretamente. A guia Git **Atividade** exibe os eventos. Clique em **Gatilhos > Git > Atividade**.
- Observe o painel de pipeline e examine as tendências. Clique em **Painéis** e pesquise pelo painel do seu pipeline. Também é possível criar um painel personalizado para relatar KPIs adicionais.

Para obter um exemplo detalhado, consulte [Como integrar continuamente o código do meu repositório do GitHub ou GitLab ao pipeline no vRealize Automation Code Stream](#).

Planejando uma compilação nativa de CI/CD no vRealize Automation Code Stream antes de adicionar tarefas manualmente

Para criar um pipeline de integração contínua e entrega contínua (CI/CD) no vRealize Automation Code Stream, é possível adicionar manualmente estágios e tarefas. Para planejar sua compilação nativa de CI/CD, colete as informações necessárias e, em seguida, crie um pipeline e adicione manualmente os estágios e as tarefas a ele.

Será necessário planejar os estágios de integração contínua (CI) e de entrega contínua (CD) do seu pipeline. Após criar o pipeline e executá-lo, você pode monitorar as tendências nas execuções do pipeline.

Para planejar os estágios de CI e CD do seu pipeline, verifique se todos os requisitos foram atendidos antes de criar o pipeline.

Como planejar requisitos externos e internos

Para criar um pipeline a partir do plano de exemplo, você usará um host do Docker, um repositório Git, um Maven e várias ferramentas de compilação pós-processo.

Endpoints e repositórios que serão necessários:

- Um repositório de código-fonte de Git no qual seus desenvolvedores verificam o código. O vRealize Automation Code Stream recebe o código mais recente para o pipeline quando os desenvolvedores confirmam as alterações.
- Um endpoint do Docker para o host de compilação do Docker que executará os comandos de compilação dentro de um contêiner.
- Um endpoint do Kubernetes para que o vRealize Automation Code Stream possa implantar sua imagem em um cluster do Kubernetes.
- Uma imagem do construtor que cria o contêiner no qual os testes de integração contínua são executados.
- Um endpoint de registro de imagem para que o host de compilação do Docker possa receber a imagem do construtor a partir dele.

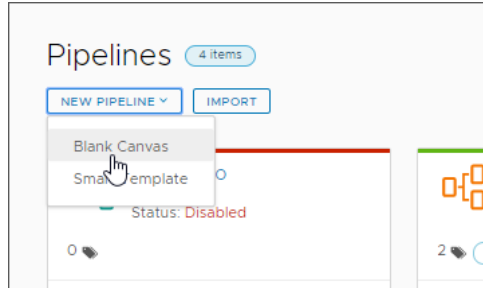
Você precisará de acesso a um projeto. O projeto agrupa todo o seu trabalho, incluindo pipeline, endpoints e painéis. Verifique se você é membro de um projeto no vRealize Automation Code Stream. Se não for, peça a um administrador do vRealize Automation Code Stream para adicioná-lo como membro de um projeto. Consulte [Como adicionar um projeto no vRealize Automation Code Stream](#).

Você precisará de um webhook Git que permita que o vRealize Automation Code Stream use o gatilho Git para disparar o pipeline quando os desenvolvedores confirmam alterações de código. Consulte [Como usar o gatilho Git no vRealize Automation Code Stream para executar um pipeline](#).

Como criar o pipeline de CI/CD e configurar o espaço de trabalho

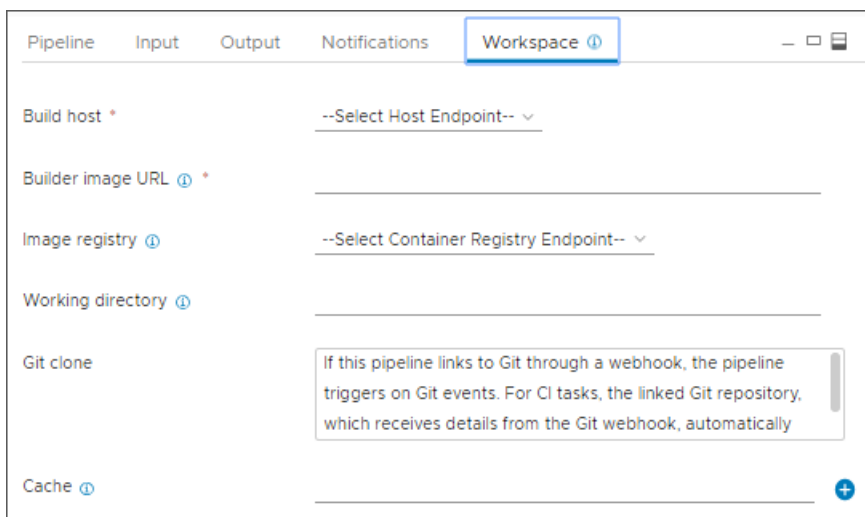
Será necessário criar o pipeline e depois configurar o espaço de trabalho, os parâmetros de entrada de pipeline e as tarefas.

Para criar o pipeline, clique em **Pipelines > Novo Pipeline > Tela em Branco**.



Na guia Espaço de Trabalho, insira as informações de integração contínua:

- Inclua o host de compilação do Docker.
- Digite o URL da sua imagem do construtor.
- Selecione o endpoint de registro de imagem para que o pipeline possa receber a imagem dele. O contêiner executa as tarefas de CI e implanta sua imagem. Se o registro precisar de credenciais, será necessário primeiro criar o endpoint de registro de imagem e, em seguida, selecioná-lo aqui para que o host possa receber a imagem do registro.
- Adicione os artefatos que devem ser armazenados em cache. Para que uma compilação seja bem-sucedida, os artefatos, como diretórios, são baixados como dependências. O cache é o local onde esses artefatos residem. Por exemplo, os artefatos dependentes podem incluir o diretório `.m2` para Maven e o diretório `node_modules` para Node.js. Esses diretórios são armazenados em cache nas execuções do pipeline para economizar tempo durante as compilações.



Na guia de entrada, configure os parâmetros de entrada do pipeline.

- Se o pipeline for usar parâmetros de entrada de um evento de gatilho Git, Gerrit ou Docker, selecione o tipo de gatilho para Parâmetros de injeção automática. Os eventos podem incluir Alterar Assunto para o Gerrit ou Git ou Nome do Proprietário do Evento para o Docker. Se o seu pipeline não usar parâmetros de entrada transmitidos do evento, deixe Parâmetros de injeção automática definido como **Nenhum**.
- Para aplicar um valor e uma descrição a um parâmetro de entrada de pipeline, clique nos três pontos verticais, e clique em **Editar**. O valor digitado é usado como entrada para tarefas, estágios ou notificações.
- Para adicionar um parâmetro de entrada de pipeline, clique em **Adicionar**. Por exemplo, é possível adicionar `approvers` para exibir um valor padrão para cada execução, mas que pode ser substituído por um aprovador diferente em tempo de execução.
- Para adicionar ou remover um parâmetro injetado, clique em **Adicionar/Remover Parâmetro Injetado**. Por exemplo, remova um parâmetro não utilizado para reduzir a desordem na página de resultados e exibir apenas os parâmetros em uso.

Pipeline Input Parameters ⓘ

Auto inject properties ☐ Gerrit ☒ Git ☐ None

ADD

Starred ⓘ	Name ▾	Value ▾	Description ▾
⋮ ☆	GIT_BRANCH_NAME		
⋮ ☆	GIT_CHANGE_SUBJECT		
⋮ ☆	GIT_COMMIT_ID		
⋮ ☆	GIT_EVENT_DESCRIPTOR		
⋮ ☆	GIT_EVENT_OWNER		
⋮ ☆	GIT_EVENT_TIMESTAMP		
⋮ ☆	GIT_REPO_NAME		
⋮ ☆	GIT_SERVER_URL		

Configure o pipeline para testar seu código:

- Adicione e configure uma tarefa de CI.
- Inclua etapas para executar o `mvn test` no código.
- Execute as ferramentas de compilação pós-processo, como JUnit, JaCoCo, FindBugs e Checkstyle para identificar quaisquer problemas após a execução da tarefa.

Task : *Unit-Test*
Notifications
VALIDATE TASK

Task name ⓘ *
Unit-Test

Type *
CI

Continue On Failure
☐

Execute Task
☒ Always
☐ On Condition

Continuous Integration

Steps *

```

1 cd demo-project
2 mvn test

```

Preserve Artifacts ⓘ
+

Export

Process
ADD

JUnit
JUnit /demo-project +

JaCoCo
Jacoco /demo-project +

Checkstyle
Checkstyle /demo-project +

FindBugs
Findbugs /demo-project +

Output Parameters
status exports

Configure o pipeline para compilar seu código:

- Adicione e configure uma tarefa de CI.
- Inclua etapas para executar o `mvn clean install` no código.
- Inclua o local e o nome de arquivo JAR para que ele preserve o artefato.

Task : *Build-App*
Notifications
VALIDATE TASK

Task name ⓘ *
Build-App

Type *
CI

Continue On Failure
☐

Execute Task
☒ Always ☐ On Condition

Continuous Integration

Steps *

```

1 cd demo-project
2 mvn clean install -DskipTests

```

Preserve Artifacts ⓘ
+

Export

Process
ADD

JUnit
JUnit /demo-project +

JaCoCo
Jacoco /demo-project +

Checkstyle
Checkstyle /demo-project +

FindBugs
Findbugs /demo-project +

Output Parameters
status exports

Configure o pipeline para publicar a imagem no host do Docker:

- Adicione e configure uma tarefa de CI.
- Adicione etapas que vão confirmar, exportar, compilar e enviar por push a sua imagem.
- Adicione a chave de exportação de `IMAGE` para a tarefa seguinte consumir.

The screenshot shows the configuration for a task named 'Build-Image'. The interface includes a 'Task name' field with the value 'Build-Image', a 'Type' dropdown set to 'CI', and a 'Continue On Failure' checkbox that is unchecked. Under 'Execute Task', the 'Always' radio button is selected. The 'Continuous Integration' section contains a 'Steps' field with a code editor showing a Docker build script. Below this, there is a 'Preserve Artifacts' section with a plus icon, an 'Export' field set to 'IMAGE', and a 'Process' button with a dropdown arrow. At the bottom, there are 'Output Parameters' tabs for 'status' and 'exports'.

Task : Build-Image Notifications VALIDATE TASK

Task name * Build-Image

Type * CI

Continue On Failure ☐

Execute Task ☒ Always ☐ On Condition

Continuous Integration

Steps *

```

1 cd demo-pro -
2 export IMAGE=automation/demo-cicd-smart-template:${executionIndex}-
3 export DOCKER_HOST=tcp://10.10.10.10:2376
4 docker login --username=auto --password=VM
5 docker build -t $IMAGE --file ./docker/Dockerfile .
6 docker push $IMAGE

```

Preserve Artifacts

Export IMAGE

Process ADD

Output Parameters status exports

Depois de configurar o espaço de trabalho, os parâmetros de entrada, as tarefas de teste e as tarefas de compilação, salve seu pipeline.

Como ativar e executar o pipeline

Depois de configurar seu pipeline com estágios e tarefas, você pode salvar e habilitar o pipeline.

Em seguida, aguarde até que o pipeline seja executado e concluído e verifique se foi bem-sucedido. Se falhou, corrija todos os erros e execute-o novamente.

Após o pipeline ter sido executado com êxito, estes são alguns aspectos que você pode querer confirmar:

- Examine a execução do pipeline e visualize os resultados das etapas da tarefa.
- No espaço de trabalho da execução do pipeline, localize os detalhes sobre seu contêiner e o repositório Git clonado.
- No espaço de trabalho, confira os resultados das suas ferramentas pós-processo e verifique se há erros, problemas de cobertura de código, bugs e problemas de estilo.
- Confirme se o seu artefato está preservado. Confirme também se a imagem foi exportada com o nome e o valor IMAGE.
- Vá para o repositório do Docker e verifique se o pipeline publicou o seu contêiner.

Para obter um exemplo detalhado que mostra como o vRealize Automation Code Stream integra continuamente seu código, consulte [Como integrar continuamente o código do meu repositório do GitHub ou GitLab ao pipeline no vRealize Automation Code Stream](#).

Planejando uma reversão no vRealize Automation Code Stream

No caso de uma falha na execução do pipeline, você pode usar o recurso de reversão para retornar seu ambiente a um estado anteriormente estável. Para usar a reversão, você deve planejar um fluxo de reversão e compreender como implementá-lo.

Um fluxo de reversão determina as etapas obrigatórias para reverter uma falha na implantação. O fluxo tem o formato de um pipeline de reversão que inclui uma ou mais tarefas sequenciais que variam dependendo do tipo de implantação que foi executada e falhou. Por exemplo, a implantação e a reversão de um aplicativo tradicional são diferentes da implantação e da reversão de um aplicativo de contêiner.

Para retornar a um bom estado de implantação, um pipeline de reversão normalmente inclui tarefas para:

- Limpar estados ou ambientes.
- Executar um script especificado pelo usuário para reverter as alterações.
- Implantar uma revisão anterior de uma implantação.

Para adicionar a reversão a um pipeline de implantação existente, anexe o pipeline de reversão às tarefas ou estágios no pipeline de implantação que deseja reverter antes de executá-lo.

Como configurar a reversão

Para configurar a reversão na implantação, é necessário:

- Criar um pipeline de implantação.
- Identificar os possíveis pontos de falha no pipeline de implantação que acionarão a reversão para que você possa anexar o pipeline de reversão. Por exemplo, é possível anexar seu pipeline de reversão a uma condição ou tipo de tarefa de sondagem no pipeline de implantação que verifica se uma tarefa anterior foi concluída com êxito. Para obter informações sobre tarefas de condição, consulte [Como usar associações de variáveis em uma tarefa de condição para executar ou parar um pipeline no vRealize Automation Code Stream](#).
- Determine o escopo da falha que disparará o pipeline de reversão, como uma falha de tarefa ou estágio. Também é possível anexar a reversão a um estágio.
- Decida quais tarefas ou tarefas de reversão serão executadas em caso de falha. Você criará seu pipeline de reversão com essas tarefas.

É possível criar manualmente um pipeline de reversão ou o vRealize Automation Code Stream pode criar um para você.

- Usando uma tela em branco, é possível criar manualmente um pipeline de reversão que segue um fluxo em paralelo a um pipeline de implantação existente. Em seguida, anexe o pipeline de reversão a uma ou mais tarefas no pipeline de implantação que dispara a reversão em caso de falha.

- Usando um modelo de pipeline inteligente, é possível configurar um pipeline de implantação com a ação de reversão. Em seguida, o vRealize Automation Code Stream cria automaticamente um ou mais pipelines de reversão padrão com tarefas predefinidas que reverterem a implantação em caso de falha.

Para obter um exemplo detalhado sobre como configurar um pipeline de CD com reversão usando um modelo de pipeline inteligente, consulte o [Como faço para reverter minha implantação no vRealize Automation Code Stream](#).

O que acontece se meu pipeline de implantação tiver várias tarefas ou estágios com reversão

Se você tiver várias tarefas ou tarefas e estágios com a reversão adicionada, lembre-se de que a sequência de reversão varia.

Tabela 4-2. Como determinar a sequência de reversão

Se você adicionar a reversão para...	Quando a reversão ocorre...
Tarefas paralelas	Se uma das tarefas paralelas falhar, a reversão para essa tarefa ocorrerá depois que todas as tarefas paralelas forem concluídas ou falharem. A reversão não ocorrerá imediatamente após a falha na tarefa.
A tarefa em um estágio e o estágio	Se uma tarefa falhar, a reversão da tarefa será executada. Se a tarefa estiver em um grupo de tarefas paralelas, a reversão da tarefa será executada depois que todas as tarefas paralelas forem concluídas ou falharem. Depois que a reversão da tarefa for concluída ou falhar ao ser concluída, a reversão do estágio será executada.

Considere um pipeline com:

- Um estágio de produção com reversão.
- Um grupo de tarefas paralelas, cada tarefa com sua própria reversão.

A tarefa chamada **UPD Deploy US** tem o pipeline de reversão **RB_Deploy_US**. Se o **UPD Deploy US** falhar, a reversão seguirá o fluxo definido no pipeline de **RB_Deploy_US**.

The screenshot displays the vRealize Automation Code Stream interface for a workflow titled "RollbackUpgrade-Example", which is currently "Enabled". The interface is divided into four main sections: Workspace, Input, Model, and Output. The "Model" section is active, showing a workflow diagram for a "Production" stage. This stage contains three sequential tasks: "UPD Deploy US", "UPD Deploy UK", and "UPD Deploy AU", all utilizing the "Kubernetes" provider. A "Parallel Task" block is also present below these tasks. The right sidebar shows the "Rollback" tab, indicating a pipeline named "RB_Deploy_US". The bottom bar features "SAVE", "RUN", and "CLOSE" buttons, a timestamp "Last saved 12 days ago", and a chat icon.

Se o **UPD Deploy US** falhar, o pipeline de **RB_Deploy_US** será executado após o **UPD Deploy UK** e o **UPD Deploy AU** terem sido concluídos ou falharem. A reversão não ocorrerá imediatamente após o **UPD Deploy US** falhar. E como o estágio de produção também tem reversão, após a execução do pipeline de **RB_Deploy_US**, o pipeline de reversão de estágio será executado.

Tutoriais para usar o vRealize Automation Code Stream

5

Use o vRealize Automation Code Stream para modelar e dar suporte ao seu ciclo de vida de liberação do DevOps, e também para testar e liberar continuamente seus aplicativos.

Você já configurou tudo o que precisa para que possa usar o vRealize Automation Code Stream. Consulte [Capítulo 2 Como configurar o vRealize Automation Code Stream para modelar meu processo de liberação](#).

Agora, é possível criar pipelines que automatizam a compilação e o teste do código do desenvolvedor antes de liberá-lo para produção. O vRealize Automation Code Stream pode implantar aplicativos tradicionais ou baseados em contêiner.

Tabela 5-1. Como usar o vRealize Automation Code Stream no seu ciclo de vida do DevOps

Usando recursos...	Exemplos do que é possível fazer...
Use o recurso de compilação nativo no vRealize Automation Code Stream.	<p>Crie pipelines de CI/CD, CI e CD que integrem, gerenciem em contêiner e forneçam seu código de forma contínua.</p> <ul style="list-style-type: none">■ Use um modelo de pipeline inteligente para criar um pipeline para você.■ Adicione manualmente os estágios e tarefas a um pipeline.
Libere seus aplicativos e automatize as liberações.	<p>Integre e libere seus aplicativos de várias maneiras.</p> <ul style="list-style-type: none">■ Integre continuamente seu código no GitHub ou GitLab no seu pipeline.■ Automatize a implantação do seu aplicativo usando um modelo de nuvem YAML.■ Automatize a implantação do seu aplicativo em um cluster do Kubernetes.■ Libere seu aplicativo para uma implantação Azul-Verde.■ Integre o vRealize Automation Code Stream às suas ferramentas de compilação, teste e implantação.■ Use uma REST API que integre o vRealize Automation Code Stream a outros aplicativos.
Rastreie tendências, métricas e KPIs.	<p>Crie painéis personalizados e obtenha informações sobre o desempenho de seus pipelines.</p>
Resolva os problemas.	<p>Quando uma execução do pipeline falhar, o vRealize Automation Code Stream criará um tíquete de JIRA.</p>

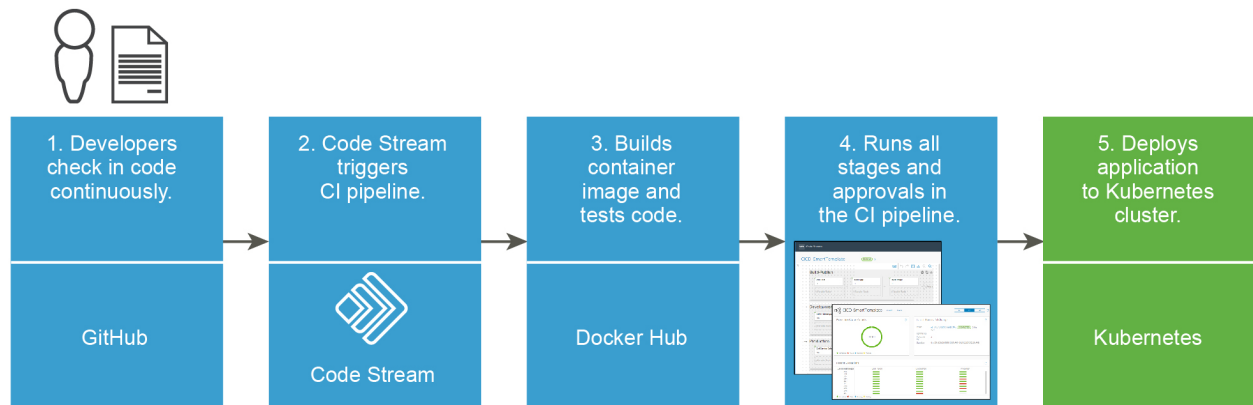
Este capítulo inclui os seguintes tópicos:

- [Como integrar continuamente o código do meu repositório do GitHub ou GitLab ao pipeline no vRealize Automation Code Stream](#)

- Como automatizar a liberação de um aplicativo implantado a partir de um modelo de nuvem YAML no vRealize Automation Code Stream
- Como automatizar a liberação de um aplicativo no vRealize Automation Code Stream para um cluster do Kubernetes
- Como implantar meu aplicativo no vRealize Automation Code Stream na implantação Azul-Verde
- Como integrar minhas próprias ferramentas de compilação, teste e implantação com o vRealize Automation Code Stream
- Como usar uma REST API para integrar o vRealize Automation Code Stream a outros aplicativos

Como integrar continuamente o código do meu repositório do GitHub ou GitLab ao pipeline no vRealize Automation Code Stream

Como desenvolvedor, você deseja integrar continuamente o código de um repositório corporativo do GitHub ou do GitLab. Sempre que os desenvolvedores atualizam o código e confirmam alterações no repositório, o vRealize Automation Code Stream pode detectar essas alterações e disparar o pipeline.



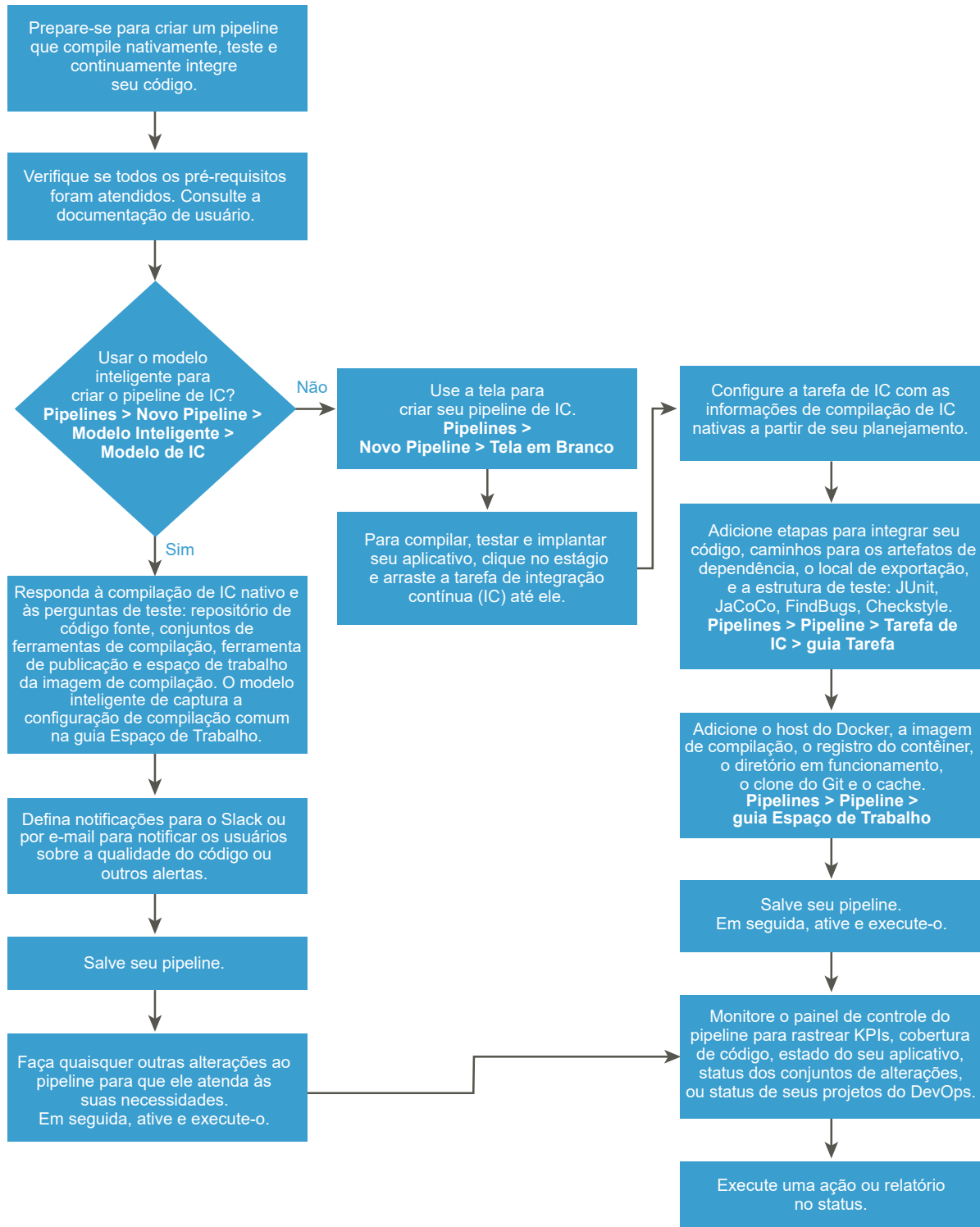
Para que o vRealize Automation Code Stream dispare seu pipeline nas alterações de código, use o gatilho Git. O vRealize Automation Code Stream disparará seu pipeline toda vez que confirmar alterações no código.

Para compilar seu código, você usará um host do Docker. É possível usar o JUnit e o JaCoCo como ferramentas de estrutura de teste, que executam testes de unidade e cobertura de código, a serem incluídas no pipeline.

Em seguida, você usará o modelo de pipeline inteligente de integração contínua (CI) para criar um pipeline de CI que compila, testa e implanta o código no cluster Kubernetes de sua equipe de projeto na AWS. Você usará um cache para armazenar os artefatos de dependência de código para a tarefa de IC, o que economizará tempo na compilação do código.

Na tarefa de pipeline que compila e testa o código, é possível incluir várias etapas de integração contínua. Essas etapas residirão no mesmo diretório de trabalho que o código-fonte é clonado quando o pipeline é disparado.

Para implantar o código no cluster do Kubernetes, use uma tarefa do Kubernetes no pipeline. Você ativará e executará o pipeline. Em seguida, fará uma alteração em seu código no repositório e observará o pipeline disparar. Depois, monitorará e relatará as tendências de pipeline após a execução do pipeline, e usará os painéis.



Neste exemplo, você usará o modelo de pipeline inteligente de integração contínua (CI) para criar um pipeline de CI para poder integrar continuamente o código no pipeline.

Outra alternativa é a criação manual do pipeline, adicionando estágios e tarefas a ele. Para obter mais informações sobre como planejar uma compilação de integração contínua e criar manualmente o pipeline, consulte [Planejando uma compilação nativa de CI/CD no vRealize Automation Code Stream antes de adicionar tarefas manualmente](#).

Pré-requisitos

- Planeje sua compilação de integração contínua. Consulte [Como planejar uma compilação nativa de CI no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente](#) e a seção sobre planejamento do estágio de integração contínua (CI).
- Verifique a existência de um repositório de códigos-fonte GitLab. Para obter ajuda, consulte o administrador do vRealize Automation Code Stream.
- Adicione um endpoint do Git. Para obter um exemplo, consulte [Como usar o gatilho Git no vRealize Automation Code Stream para executar um pipeline](#).
- Para que o vRealize Automation Code Stream detecte as alterações no repositório GitHub ou GitLab e dispare um pipeline quando houver alterações, adicione um webhook. Para obter um exemplo, consulte [Como usar o gatilho Git no vRealize Automation Code Stream para executar um pipeline](#).
- Adicione um endpoint do host do Docker, que cria um contêiner para a tarefa de CI, que pode ser usado por várias tarefas de CI. Para obter mais informações sobre endpoint, consulte [O que são endpoints no vRealize Automation Code Stream](#).
- Obtenha o URL da imagem, o host de compilação e o URL da imagem da compilação. Para obter ajuda, consulte o administrador do vRealize Automation Code Stream.
- Verifique se você usa JUnit e JaCoCo para as ferramentas de estrutura de teste.
- Configure uma instância externa para sua compilação de CI: Jenkins, TFS ou Bamboo. O plug-in Kubernetes implantará seu código. Para obter ajuda, consulte o administrador do vRealize Automation Code Stream.

Procedimentos

- 1 Siga os pré-requisitos.
- 2 Para criar o pipeline usando o modelo de pipeline inteligente, abra o modelo de pipeline inteligente de CI e preencha o formulário.
 - a Clique em **Pipelines > Novo Pipeline > Modelo Inteligente > Integração Contínua**.
 - b Responda às perguntas no modelo sobre o repositório de códigos-fonte, os conjuntos de ferramentas de compilação, a ferramenta de publicação e o espaço de trabalho de imagem de compilação.
 - c Adicione notificações por e-mail ou Slack para sua equipe.
 - d Para que o modelo de pipeline inteligente crie o pipeline, clique em **Criar**.

- e Para fazer outras alterações no pipeline, clique em **Editar**, faça as alterações e clique em **Salvar**.
 - f Ative o pipeline e execute-o.
- 3 Para criar o pipeline manualmente, adicione estágios e tarefas à tela e tenha as informações de compilação de CI nativas prontas para configurar a tarefa de integração contínua (CI).
- a Clique em **Pipelines > Novo Pipeline > Tela em Branco**.
 - b Clique no estágio e, em seguida, arraste as várias tarefas de CI do painel de navegação para o estágio.
 - c Para configurar a tarefa de IC, clique nela e clique na guia **Tarefa**.
 - d Adicione as etapas que integram continuamente o código.
 - e Inclua os caminhos para os artefatos de dependência.
 - f Adicione o local de exportação.
 - g Adicione as ferramentas de estrutura de teste que serão usadas.
 - h Adicione o host do Docker e a imagem de compilação.
 - i Adicione o registro do contêiner, o diretório de trabalho e o cache.
 - j Salve o pipeline e, em seguida, ative-o.
- 4 Faça uma alteração no código em seu repositório GitHub ou GitLab.
- O gatilho Git ativa o pipeline, que começa a ser executado.
- 5 Para verificar se a alteração do código disparou o pipeline, clique em **Gatilhos > Git > Atividade**.

- Para exibir a execução do pipeline, clique em **Execuções** e verifique se as etapas criaram e exportaram a imagem de compilação.

The screenshot displays the vRealize Automation Code Stream interface. On the left, a sidebar contains navigation options: Dashboards, Executions, User Operations, Pipelines, Manage (with sub-options Endpoints, Variables, and Triggers), and Triggers (with sub-options Gerrit and Git). The main panel shows the execution details for 'CICD-SmartTemplate #51', which is marked as 'COMPLETED'. A progress bar at the top indicates the status of various stages: Build-Publish (Unit-Test, Build-App, Build-Image) and Development (Create Namespace, Create Secret, Create Service, Create Dep). The 'Build-Image' task is selected, showing its details: Task name (Build-Image), Type (CI), Status (COMPLETED), Duration (5s), and Execute Task (Always). The Result section shows the command output, including Docker login and build commands. The Preserved Artifacts section shows the path to the build image artifacts. The Exports section shows the exported value for the IMAGE variable: automation/cicd-smart-template:51.

- Para monitorar o painel do pipeline e rastrear KPIs e tendências, clique em **Painéis > Painéis de Pipeline**.

Resultados

Parabéns! Você criou um pipeline que integra continuamente o código a partir de um repositório GitHub ou GitLab ao pipeline e implanta a imagem de compilação.

Próximo passo

Para saber mais, consulte [Mais recursos para administradores e desenvolvedores do vRealize Automation Code Stream](#).

Como automatizar a liberação de um aplicativo implantado a partir de um modelo de nuvem YAML no vRealize Automation Code Stream

Como desenvolvedor, você precisa de um pipeline que obtenha um modelo de nuvem de automação de uma instância do GitHub local todas as vezes que uma alteração é confirmada. Você precisa do pipeline para implantar um aplicativo WordPress no Amazon Web Services (AWS) EC2 ou em um centro de dados. O vRealize Automation Code Stream chama o modelo de nuvem do pipeline e automatiza a integração contínua e a entrega contínua (CI/CD) desse modelo de nuvem para implantar seu aplicativo.

Para criar e acionar seu pipeline, você precisará de um Modelo de Nuvem VMware.

Para **Origem do modelo de nuvem** na tarefa de modelo de nuvem do vRealize Automation Code Stream, é possível selecionar:

- **Modelo do Cloud Assembly** como controle de origem. Nesse caso, não é necessário um repositório GitLab ou GitHub.
- **Controle de Origem** se você usar o GitLab ou o GitHub para controle de origem. Nesse caso, é necessário ter um webhook do Git e disparar o pipeline por meio desse webhook.

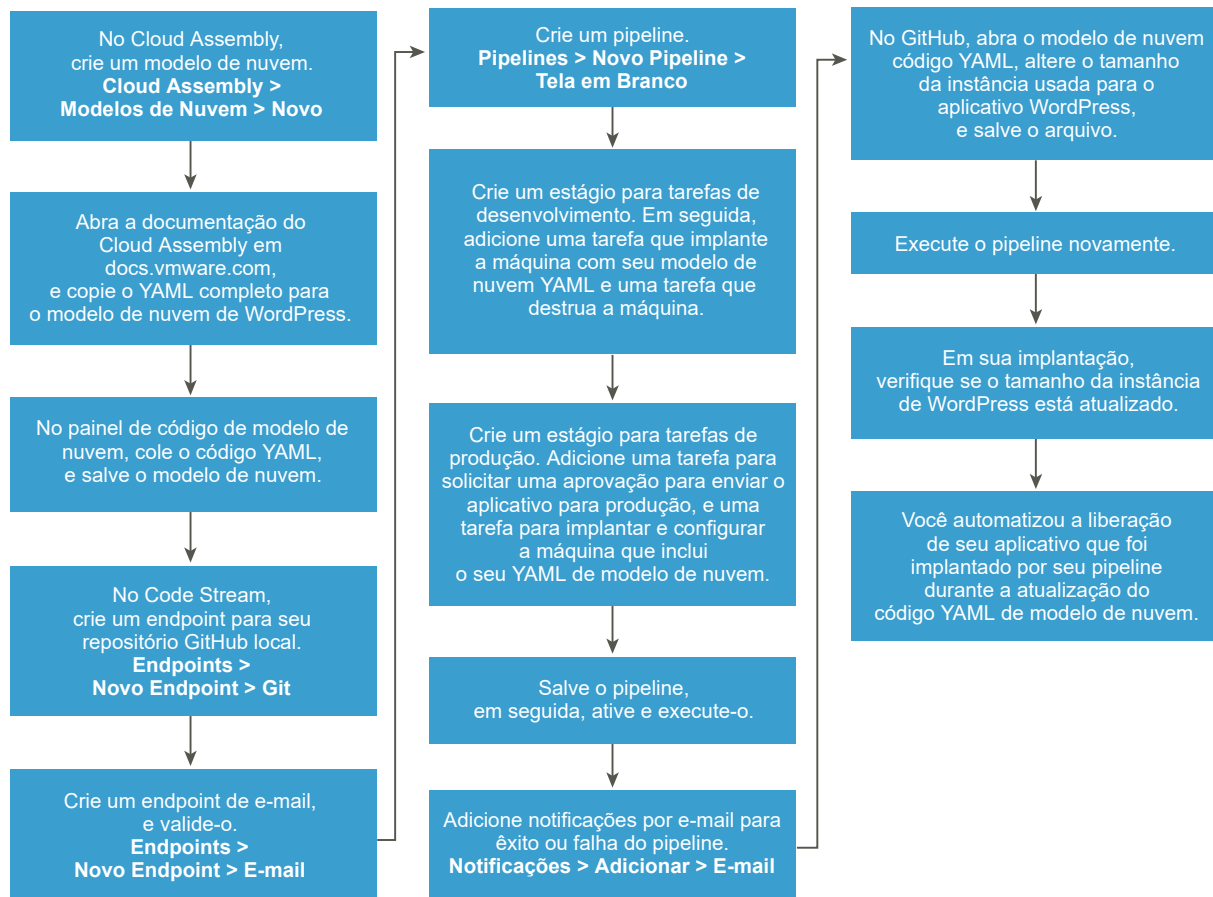
Se você tiver um modelo de nuvem YAML no repositório do GitHub e quiser usar esse modelo de nuvem no pipeline, será necessário fazer o seguinte:

- 1 No vRealize Automation Cloud Assembly, envie o modelo de nuvem ao seu repositório do GitHub.
- 2 No vRealize Automation Code Stream, crie um endpoint do Git. Em seguida, crie um webhook do Git que use seu endpoint do Git e seu pipeline.
- 3 Para disparar o pipeline, atualize qualquer arquivo no repositório do GitHub e confirme a alteração.

Se você não tiver um modelo de nuvem do YAML no repositório do GitHub e quiser usar um modelo de nuvem do controle de origem, siga este procedimento para saber como. Ele mostra como criar um modelo de nuvem para um aplicativo WordPress e dispará-lo a partir de um repositório do GitHub local. Sempre que você fizer uma alteração no modelo de nuvem do YAML, o pipeline será disparado e automatizará a liberação do seu aplicativo.

- No vRealize Automation Cloud Assembly, você adicionará uma conta de nuvem, adicionará uma zona de nuvem e criará o modelo de nuvem.
- No vRealize Automation Code Stream, você adicionará um endpoint para o repositório do GitHub local que hospeda seu modelo de nuvem. Em seguida, adicionará esse modelo de nuvem ao pipeline.

Este exemplo de caso de uso mostra como usar um modelo de nuvem a partir de um repositório do GitHub local.



Pré-requisitos

- Adicione uma conta de nuvem e uma zona de nuvem na infraestrutura do vRealize Automation Cloud Assembly. Consulte a documentação do vRealize Automation Cloud Assembly.
- Para criar o modelo de nuvem no procedimento a seguir, copie o código YAML do WordPress para a área de transferência. Consulte o código YAML do modelo de nuvem no caso de uso do WordPress na documentação do vRealize Automation Cloud Assembly.
- Adicione o código YAML para o aplicativo WordPress à instância do GitHub.
- Adicione um webhook para o gatilho do Git para que o pipeline possa extrair o código YAML sempre que você confirmar alterações nele. No vRealize Automation Code Stream, clique em **Gatilhos > Git > Webhooks para Git**.
- Para trabalhar com uma tarefa de modelo de nuvem, você deve ter qualquer uma das funções do vRealize Automation Cloud Assembly.

Procedimentos

- 1 No vRealize Automation Cloud Assembly, siga estas etapas.
 - a Clique em **VMware Cloud Templates** e crie um modelo de nuvem e uma implantação para o aplicativo WordPress.
 - b Cole o código YAML do WordPress copiado para a área de transferência no modelo de nuvem e implante-o.

The screenshot displays the vRealize Automation Cloud Assembly interface for a template named "Wordpress-CT-Test". The interface is divided into several sections:

- Left Panel (Component Catalog):** A search bar labeled "Search component" is at the top. Below it, a list of components is organized into categories: Cloud Agnostic (Machine, Network, Load Balancer, Volume), vSphere (Machine, Network, Disk), and AWS (Instance, Volume, Kinesis, Kinesis Fire). The "Machine" component under "Cloud Agnostic" is currently selected.
- Central Canvas:** A grid-based workspace where components are placed. It contains three main components: "DBTier", "WebTier", and "WP-Network-P...". Each component has a small icon and a label. The "DBTier" and "WebTier" components have a "tag:zon..." label below them. A line connects the "WP-Network-P..." component to the "WebTier" component.
- Right Panel (YAML Editor):** A code editor showing the YAML configuration for the template. The code is as follows:


```

1 resources:
2   DBTier:
3     type: Cloud.Machine
4     properties:
5       name: mysql
6       image: 'ubuntu-16'
7       flavor: 'small'
8       constraints:
9         - tag: zone:dev
10  WebTier:
11    type: Cloud.Machine
12    properties:
13      name: wordpress
14      image: 'ubuntu-16'
15      flavor: 'small'
16      constraints:
17        - tag: zone:dev
18  WP-Network-Private:
19    type: Cloud.Network
20    properties:
21      name: WP-Network-Private
22      networkType: existing
23      constraints:
24        - tag: 'type:isolation'
25        - tag: 'zone:dev'
            
```
- Bottom Bar:** Contains three buttons: "DEPLOY", "VERSION", and "CLOSE". To the right of these buttons, it says "Last saved 5 minutes ago". A mouse cursor is pointing at the "DEPLOY" button.

2 No vRealize Automation Code Stream, crie endpoints.

- Crie um endpoint do Git para o repositório GitHub local no qual o arquivo YAML reside.
- Adicione um endpoint de e-mail para notificar os usuários sobre o status do pipeline quando ele for executado.

Add Endpoint

Project * Codestream

Type * Email

Name * Enter value here

Description

Mark as restricted ☐ non-restricted

Sender's Address * eg: abc@xyz.com

Encryption Method * SSL

Outbound Host * myimap.org

Outbound Port * Port number

Outbound Protocol * smtp

Outbound Username username

Outbound Password password

CREATE VALIDATE CANCEL

3 Crie um pipeline e adicione notificações para quando o pipeline for bem-sucedido ou falhar.

Notification

Send notification type ☒ Email ☐ Ticket ☐ Webhook

When pipeline ☒ Completes ☐ Is Waiting ☐ Fails ☐ Is cancelled ☐ Starts to run

Email server ⓘ * --Select Email server--

Send Email

To ⓘ \$ * Email IDs of recipients

Subject \$ * Email Subject

Body ⓘ \$ * 1

CANCEL SAVE

4 Adicione um estágio para o desenvolvimento e adicione uma tarefa de modelo em nuvem.

- a Adicione uma tarefa de modelo de nuvem que implante a máquina e configure-a para usar o YAML do modelo de nuvem para o aplicativo WordPress.

```
resources:
  DBTier:
    type: Cloud.Machine
    properties:
      name: mysql
      image: 'ubuntu-16'
      flavor: 'small'
      constraints:
        - tag: zone:dev
  WebTier:
    type: Cloud.Machine
    properties:
      name: wordpress
      image: 'ubuntu-16'
      flavor: 'small'
      constraints:
        - tag: zone:dev
  WP-Network-Private:
    type: Cloud.Network
    properties:
      name: WP-Network-Private
      networkType: existing
      constraints:
        - tag: 'type:isolated-net'
        - tag: 'zone:dev'
```

- b Adicione uma tarefa de modelo de nuvem que destrua a máquina para liberar recursos.

- 5 Adicione um estágio para produção e inclua tarefas de aprovação e implantação.
 - a Adicione uma tarefa de operação do usuário para exigir aprovação para enviar o aplicativo WordPress para produção.
 - b Adicione uma tarefa de modelo de nuvem para implantar a máquina e configure-a com o YAML de modelo de nuvem para o aplicativo WordPress.

Quando você seleciona **Criar**, o nome da implantação deve ser exclusiva. Se você deixar o nome em branco, o vRealize Automation Code Stream atribuirá a ele um nome aleatório exclusivo.

Veja o que você precisa saber se selecionar **Reverter** no seu próprio caso de uso: se você selecionar a ação **Reverter** e inserir uma **Versão de Reversão**, essa versão deverá estar no formato de **n-X**. Por exemplo, **n-1**, **n-2**, **n-3** e assim por diante. Se você criar e atualizar a implantação em qualquer local diferente do vRealize Automation Code Stream, a reversão não será permitida.

Quando você faz login no vRealize Automation Code Stream, ele obtém um token de usuário, que é válido por 30 minutos. Para durações de pipeline de longa execução, quando a tarefa anterior à tarefa de modelo de nuvem demora 30 minutos ou mais para ser executada, o token de usuário expira. Como resultado, a tarefa de modelo de nuvem falha.

Para garantir que o pipeline possa ser executado por mais de 30 minutos, você pode inserir um token de API opcional. Quando o vRealize Automation Code Stream invocar o modelo de nuvem, o token de API persistirá, e a tarefa de modelo de nuvem continuará a usar o token de API.

Quando você usa o token de API como uma variável, ele é criptografado. Caso contrário, ele será usado como texto simples.

6 Execute o pipeline.

Para verificar se cada tarefa foi concluída com êxito, clique na tarefa em execução e verifique o status nos detalhes da implantação para ver informações detalhadas sobre os recursos.

7 No GitHub, modifique o tipo de instância de servidor do WordPress de `small` para `medium`.

Quando você confirmar as alterações, o pipeline será disparado. Ele extrairá o código atualizado do repositório do GitHub e compilará o aplicativo.

```
WebTier:
  type: Cloud.Machine
  properties:
    name: wordpress
    image: 'ubuntu-16'
    flavor: 'medium'
    constraints:
      - tag: zone:dev
```

8 Execute o pipeline novamente, verifique se foi bem-sucedido e se ele alterou o tipo de instância do WordPress de `pequena` para `média`.

Resultados

Parabéns! Você automatizou o lançamento do aplicativo implantado a partir de um modelo de nuvem YAML.

Próximo passo

Para saber mais sobre como usar o vRealize Automation Code Stream, consulte [Capítulo 5 Tutoriais para usar o vRealize Automation Code Stream](#).

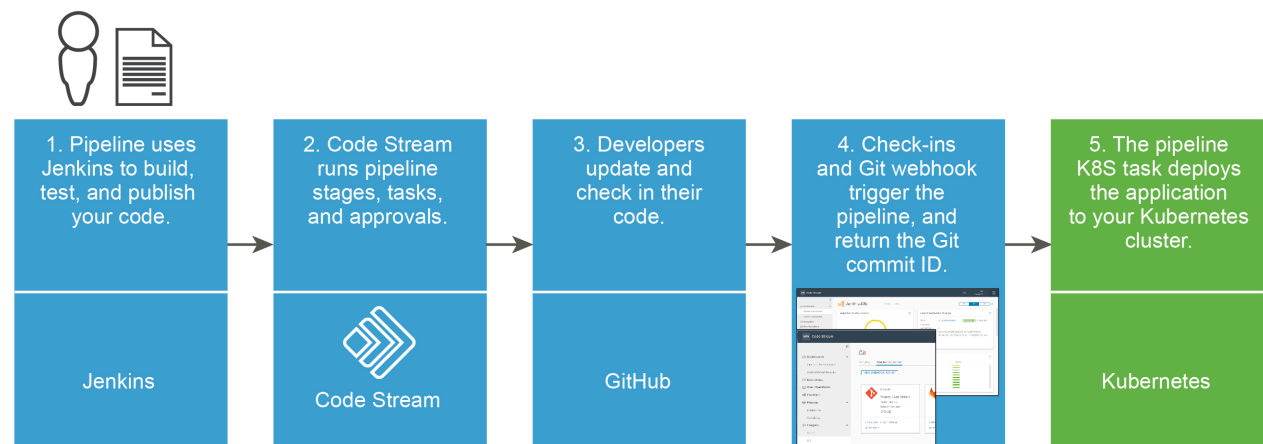
Para obter referências adicionais, consulte [Mais recursos para administradores e desenvolvedores do vRealize Automation Code Stream](#).

Como automatizar a liberação de um aplicativo no vRealize Automation Code Stream para um cluster do Kubernetes

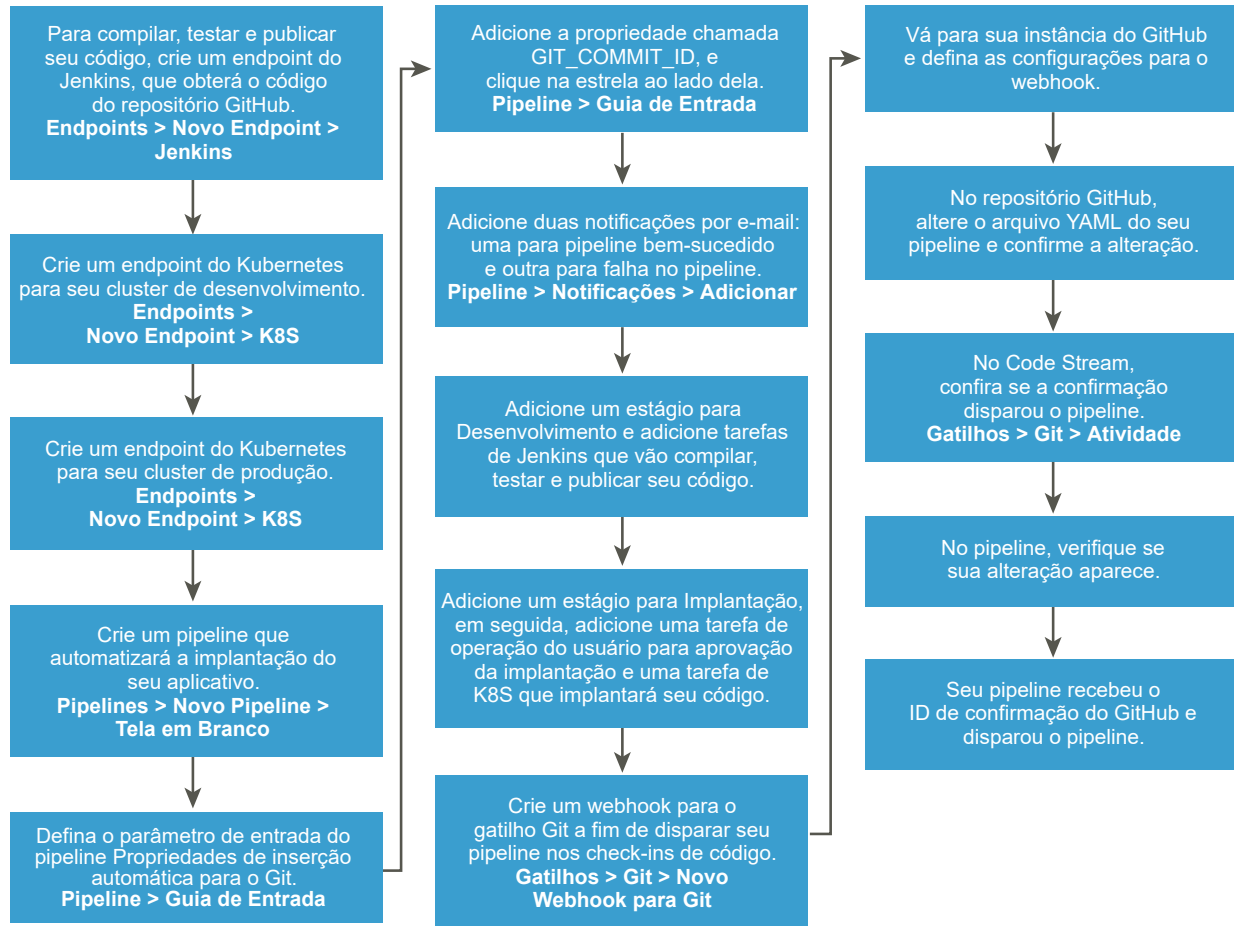
Como administrador ou desenvolvedor do vRealize Automation Code Stream, você pode usar o vRealize Automation Code Stream e o VMware Tanzu Kubernetes Grid Integrated Edition (anteriormente conhecido como VMware Enterprise PKS) para automatizar a implantação dos seus aplicativos de software em um cluster Kubernetes. Esse caso de uso menciona outros métodos possíveis para automatizar a liberação do aplicativo.

Neste caso de uso, você criará um pipeline que inclui dois estágios e usará o Jenkins para criar e implantar o aplicativo.

- O primeiro estágio é para desenvolvimento. Ele usa o Jenkins para extrair o código de uma ramificação no repositório GitHub, depois o compila, testa e publica.
- O segundo estágio é para implantação. Ele executa uma tarefa de operação do usuário que requer a aprovação de usuários principais antes que o pipeline possa implantar o aplicativo no cluster do Kubernetes.



As ferramentas de desenvolvimento, instâncias de implantação e o arquivo YAML de pipeline devem estar disponíveis para que o pipeline possa compilar, testar, publicar e implantar o aplicativo. O pipeline implantará o aplicativo em instâncias de desenvolvimento e de produção dos clusters do Kubernetes no AWS.



Outros métodos que automatizam a liberação do aplicativo:

- Em vez de usar o Jenkins para compilar o aplicativo, é possível usar o recurso de compilação nativo do vRealize Automation Code Stream e um host de compilação do Docker.
- Em vez de implantar o aplicativo em um cluster do Kubernetes, é possível implantá-lo em um cluster do Amazon Web Services (AWS).

Para obter mais informações sobre como usar o recurso de compilação nativo do vRealize Automation Code Stream e um host do Docker, consulte:

- [Como planejar uma compilação nativa de CI/CD no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente](#)
- [Planejando uma compilação nativa de CI/CD no vRealize Automation Code Stream antes de adicionar tarefas manualmente](#)

Pré-requisitos

- Verifique se o código do aplicativo a ser implantado reside em um repositório GitHub ativo.
- Verifique se há uma instância de trabalho Jenkins ativa.
- Verifique se há um servidor de e-mail ativo.

- No vRealize Automation Code Stream, crie um endpoint de e-mail que se conecte ao servidor de e-mail.
- Configure dois clusters do Kubernetes no Amazon Web Services (AWS), para desenvolvimento e produção, onde o pipeline implantará o aplicativo.
- Verifique se o repositório GitHub contém o código YAML para o pipeline e, como alternativa, um arquivo YAML que defina os metadados e as especificações do ambiente.

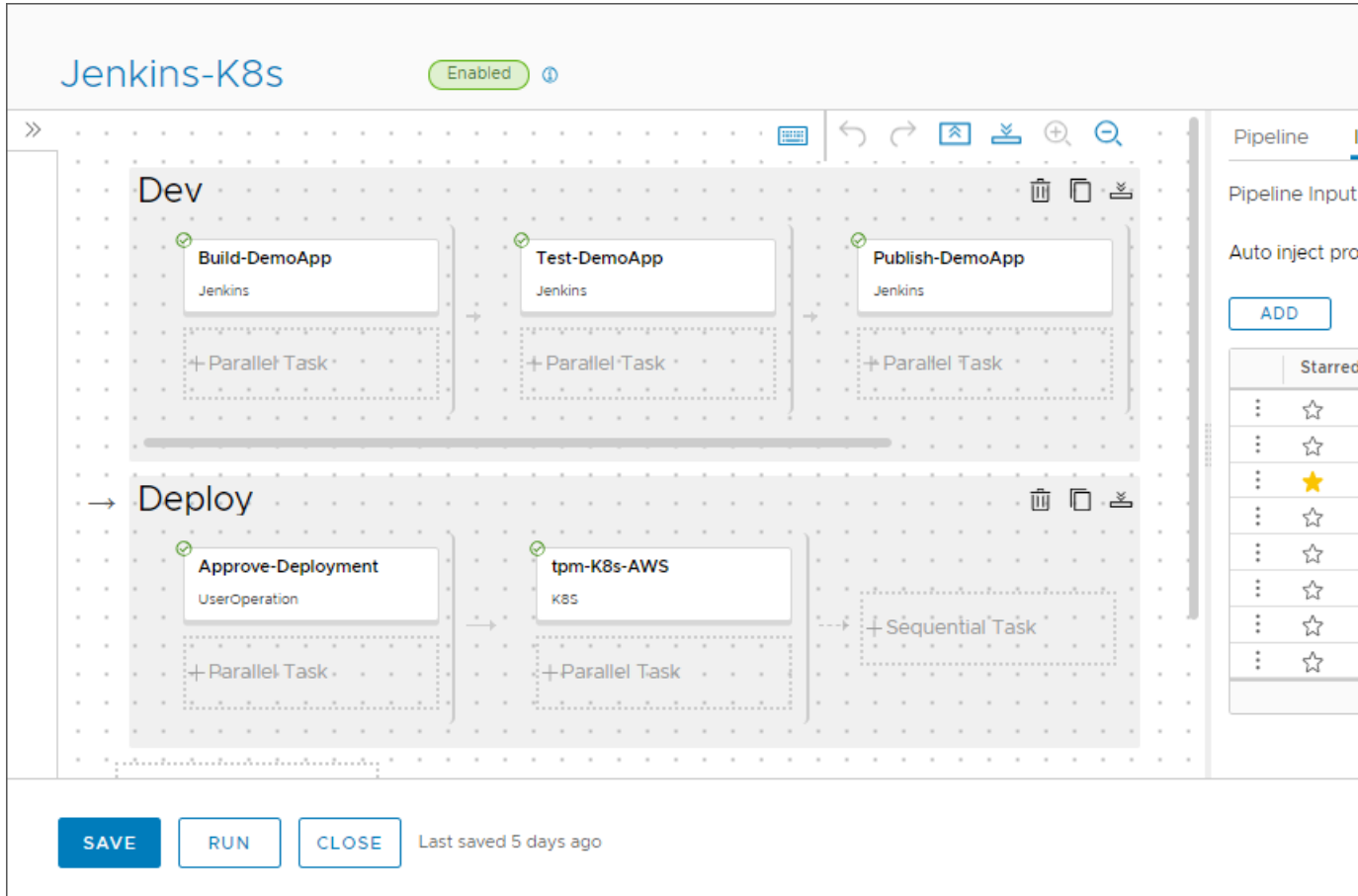
Procedimentos

- 1 No vRealize Automation Code Stream, clique em **Endpoints > Novo Endpoint** e crie um endpoint do Jenkins que será usado no pipeline para extrair o código do seu repositório GitHub.
- 2 Para criar endpoints do Kubernetes, clique em **Novo Endpoint**.
 - a Crie um endpoint para seu cluster de desenvolvimento do Kubernetes.
 - b Crie um endpoint para seu cluster de produção do Kubernetes.

3 Crie um pipeline que implante um contêiner do aplicativo, como o WordPress, no cluster de desenvolvimento do Kubernetes e defina as propriedades de entrada para o pipeline.

- Para permitir que o seu pipeline reconheça uma confirmação de código no GitHub que disparará o pipeline, clique na guia **Entrada**, no próprio pipeline, e selecione **Propriedades de Inserção Automática**.
- Adicione a propriedade denominada **GIT_COMMIT_ID** e clique na estrela ao lado dela.

Quando o pipeline for executado, sua execução exibirá a ID de confirmação retornada pelo gatilho Git.



- 4 Adicione notificações para enviar um e-mail quando o pipeline for bem-sucedido ou falhar.
 - a No pipeline, clique na guia **Notificações** e clique em **Adicionar**.
 - b Para adicionar uma notificação por e-mail quando o pipeline terminar de ser executado, selecione **E-mail** e depois **Completa**. Em seguida, selecione o servidor de e-mail, digite os endereços de e-mail e clique em **Salvar**.
 - c Para adicionar outra notificação por e-mail para uma falha de pipeline, selecione **Falhar** e clique em **Salvar**.

Notification

Send notification type

☒ Email
 ☐ Ticket
 ☐ Webhook

When pipeline

☒ Completes
 ☐ Is Waiting
 ☐ Fails
 ☐ Is cancelled
 ☐ Starts to run

Email server ⓘ *

--Select Email server-- ▾

Send Email

To ⓘ \$ *

Subject \$ *

Body ⓘ \$ *

Email IDs of recipients

Email Subject

1

CANCEL

SAVE

- 5 Adicione um estágio de desenvolvimento ao pipeline e tarefas que compilem, testem e publiquem seu aplicativo. Em seguida, valide cada tarefa.
 - a Para compilar seu aplicativo, adicione uma tarefa de Jenkins que use o endpoint do Jenkins e execute um trabalho de compilação a partir do servidor Jenkins. Depois, para que o pipeline receba seu código, insira a ramificação Git desta forma: `${input.GIT_BRANCH_NAME}`
 - b Para testar seu aplicativo, adicione uma tarefa de Jenkins que use o mesmo endpoint do Jenkins e execute um trabalho de teste no servidor Jenkins. Em seguida, insira a mesma ramificação Git.
 - c Para publicar seu aplicativo, adicione uma tarefa de Jenkins que use o mesmo endpoint do Jenkins e execute um trabalho de publicação no servidor Jenkins. Em seguida, insira a mesma ramificação Git.

The screenshot displays the Jenkins-K8s pipeline editor. At the top, the pipeline is named 'Jenkins-K8s' and is marked as 'Enabled'. The main workspace shows a 'Dev' stage containing three sequential tasks: 'Build-DemoApp', 'Test-DemoApp', and 'Publish-DemoApp'. Each task is configured to use the 'Jenkins' endpoint. Below the 'Dev' stage, there is a 'Deploy' task and a '+ Stage' button to add more stages. The right sidebar provides configuration options for the selected task, including 'Task name', 'Type', 'Continue On Failure', 'Execute Task', 'Jenkins Job', 'Endpoint', 'Job', 'branchName', and 'Output Parameters'. At the bottom, there are 'SAVE', 'RUN', and 'CLOSE' buttons, along with a timestamp 'Last saved 5 days ago'.

- 6 Adicione um estágio de implantação ao seu pipeline e, depois, adicione uma tarefa que exija uma aprovação para a implantação do aplicativo, e outra tarefa que implante o aplicativo no cluster do Kubernetes. Em seguida, valide cada tarefa.
 - a Para exigir uma aprovação na implantação do seu aplicativo, adicione uma tarefa de Operação do Usuário, adicione os endereços de e-mail para os usuários que devem aprová-la e digite uma mensagem. Em seguida, ative **Enviar E-mail**.
 - b Para implantar o aplicativo, adicione uma tarefa do Kubernetes. Depois, nas propriedades da tarefa do Kubernetes, selecione seu cluster de desenvolvimento do Kubernetes, selecione a ação **Criar** e selecione a fonte de payload **Definição de Local**. Em seguida, selecione seu arquivo YAML local.
- 7 Adicione um webhook do Git para permitir que o vRealize Automation Code Stream use o gatilho Git, que dispara seu pipeline quando os desenvolvedores confirmam o código.

The screenshot displays the 'Git' configuration page with the 'Webhooks for Git' tab selected. The form includes the following fields and options:

- Webhook URL:** `https://...vmware.com/pipeline/api/git-webhook-listeners/d4c4b02804780`
- Project:** Code Stream
- Name:** muser-Demo-WH
- Description:** (empty text area)
- Endpoint:** tpm-GitHub
- Branch:** master
- Secret token:** (masked with dots) and a **GENERATE** button.
- File:** (empty text area)
- Inclusions:** --Select-- dropdown and Value field with a plus icon.
- Exclusions:** --Select-- dropdown and Value field with a plus icon.
- Prioritize Exclusion:** Toggle switch (currently off).
- Trigger:** PUSH (selected) and PULL REQUEST (radio buttons).
- API token:** (masked with dots) and buttons for **CREATE VARIABLE** and **GENERATE TOKEN**.
- Pipeline:** Jenkins-K8s (selected from a dropdown).
- Comments:** (empty text area).

- 8 Para testar o pipeline, acesse o repositório GitHub, atualize o arquivo YAML do aplicativo e confirme a alteração.
 - a No vRealize Automation Code Stream, verifique se a confirmação é exibida.
 - a Clique em **Gatilhos > Git > Atividade**.
 - b Procure o gatilho do pipeline.

- c Clique em **Painéis > Painéis de Pipeline**.
- d No painel de pipeline, encontre o GIT_COMMIT_ID na última área de alteração bem-sucedida.

9 Verifique o código de pipeline e verifique se a alteração é exibida.

Resultados

Parabéns! Você automatizou a implantação do seu aplicativo de software no cluster do Kubernetes.

Exemplo: Exemplo de YAML de pipeline que implanta um aplicativo em um cluster do Kubernetes

Para o tipo de pipeline usado nesse exemplo, o YAML deve ser semelhante ao seguinte código:

```
apiVersion: v1
kind: Namespace
metadata:
  name: ${input.GIT_BRANCH_NAME}
  namespace: ${input.GIT_BRANCH_NAME}
---
apiVersion: v1
data:
  .dockercfg:
eyJzeWlwG9ueS10YW5nbyliZXRhMi5qZnJvZy5pbyI6eyJlc2VybmFtZSI6InRhbmdvLWJldGEyIiwicGFzc3dvcmQI Oi
JhRGstcmVOLWlUQi1IejciLCJlbWVpbCI6InRhbmdvLWJldGEyQHZtd2FyZS5jb20iLCJhdXRoIjoizEdGdVoyOHRZbVYw
WVRJNllVUnJMWEPsVGkxdFZFSXRTSG8zIn19
kind: Secret
metadata:
  name: jfrog
  namespace: ${input.GIT_BRANCH_NAME}
type: kubernetes.io/dockercfg
---
apiVersion: v1
kind: Service
metadata:
  name: codestream
  namespace: ${input.GIT_BRANCH_NAME}
  labels:
    app: codestream
spec:
  ports:
    - port: 80
  selector:
    app: codestream
    tier: frontend
  type: LoadBalancer
---
apiVersion: extensions/v1
kind: Deployment
metadata:
  name: codestream
  namespace: ${input.GIT_BRANCH_NAME}
```



```

labels:
  app: codestream
spec:
  selector:
    matchLabels:
      app: codestream
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: codestream
        tier: frontend
    spec:
      containers:
        - name: codestream
          image: cas.jfrog.io/codestream:${input.GIT_BRANCH_NAME}-${Dev.PublishApp.output.jobId}
          ports:
            - containerPort: 80
              name: codestream
          imagePullSecrets:
            - name: jfrog

```

Próximo passo

Para implantar o aplicativo de software no cluster de produção do Kubernetes, execute as etapas novamente e selecione o cluster de produção.

Para saber mais sobre a integração do vRealize Automation Code Stream com o Jenkins, consulte [Como integrar o vRealize Automation Code Stream ao Jenkins](#).

Como implantar meu aplicativo no vRealize Automation Code Stream na implantação Azul-Verde

Azul-Verde é um modelo de implantação que usa dois hosts do Docker, implantados e configurados de forma idêntica em um cluster do Kubernetes. Com o modelo de implantação Blue-Green, reduz-se o tempo de inatividade que pode ocorrer no ambiente quando os pipelines no vRealize Automation Code Stream implantam os aplicativos.

As instâncias azul e verde no modelo de implantação têm finalidades diferentes. Somente uma instância por vez aceita o tráfego dinâmico que implanta seu aplicativo, e cada instância aceita esse tráfego em momentos específicos. A instância azul recebe a primeira versão do aplicativo, e a instância verde recebe a segunda.

O balanceador de carga no ambiente Azul-Verde determina qual rota o tráfego dinâmico assumirá ao implantar o aplicativo. Ao usar o modelo Azul-Verde, seu ambiente permanece em operação, os usuários não percebem nenhum tempo de inatividade, o pipeline faz a integração de forma contínua e implanta o aplicativo no ambiente de produção.

O pipeline criado no vRealize Automation Code Stream representa o modelo de implantação Azul-Verde em dois estágios. Um estágio é para desenvolvimento, e o outro estágio é para produção.

Tabela 5-2. Tarefas do estágio de desenvolvimento para implantação Azul-Verde

Tipo de tarefa	Tarefa
Kubernetes	Cria um namespace para sua implantação Azul-Verde.
Kubernetes	Cria uma chave secreta para o Hub do Docker.
Kubernetes	Cria o serviço usado para implantar o aplicativo.
Kubernetes	Cria a implantação azul.
Pesquisa	Verifica a implantação azul.
Kubernetes	Remove o namespace.

Tabela 5-3. Tarefas do estágio de produção para implantação Azul-Verde

Tipo de tarefa	Tarefa
Kubernetes	O verde obtém os detalhes de serviço do azul.
Kubernetes	Obtém os detalhes para o conjunto de réplicas verde.
Kubernetes	Cria a implantação verde e use a chave secreta para receber a imagem do contêiner.
Kubernetes	Atualiza o serviço.
Pesquisa	Verifica se a implantação foi bem-sucedida no URL de produção.
Kubernetes	Conclui a implantação azul.
Kubernetes	Remove a implantação azul.

Para implantar o aplicativo no seu próprio modelo de implantação Azul-Verde, crie um pipeline no vRealize Automation Code Stream com dois estágios. O primeiro estágio inclui as tarefas azuis que implantam o aplicativo na instância azul, e o segundo estágio inclui tarefas verdes que implantam o aplicativo na instância verde.

É possível criar seu pipeline usando o modelo de pipeline inteligente de CI/CD. O modelo cria os estágios e as tarefas do pipeline, além de incluir as seleções de implantação.

Se você criar o pipeline manualmente, deverá planejar os estágios do pipeline. Para obter um exemplo, consulte [Planejando uma compilação nativa de CI/CD no vRealize Automation Code Stream antes de adicionar tarefas manualmente](#).

Neste exemplo, você usa o modelo de pipeline inteligente de CI/CD para criar o pipeline Azul-Verde.

Pré-requisitos

- Verifique se é possível acessar um cluster do Kubernetes ativo no AWS.

- Verifique se há um ambiente de implantação Azul-Verde configurado e se há instâncias azul e verde configuradas de forma idêntica.
- Crie um endpoint do Kubernetes no vRealize Automation Code Stream que implanta a imagem do aplicativo no cluster do Kubernetes no AWS.
- Familiarize-se usando o modelo de pipeline inteligente de CI/CD. Consulte [Como planejar uma compilação nativa de CI/CD no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente](#).

Procedimentos

- 1 Clique em **Pipelines > Novo Pipeline > Modelos Inteligentes > Modelo de CI/CD**.
- 2 Digite as informações para a parte de CI do modelo de pipeline inteligente de CI/CD e clique em **Seguinte**.

Para obter ajuda, consulte [Como planejar uma compilação nativa de CI/CD no vRealize Automation Code Stream antes de usar o modelo de pipeline inteligente](#).
- 3 Conclua a parte de CD do modelo de pipeline inteligente
 - a Selecione os ambientes para a implantação do aplicativo. Por exemplo, **Des.** e **Prod.**.
 - b Selecione o serviço que o pipeline usará para a implantação.
 - c Na área de implantação, selecione o endpoint do cluster para os ambientes Des. e Prod.
 - d Para o modelo de implantação de produção, selecione **Azul-Verde** e clique em **Criar**.

Smart Template: CI/CD

Step 2 of 2

Environment ⓘ * ☒ Dev ☒ Prod

K8s YAML files *

Processed files: codestream.yaml

Select service

Deployment name	Service	Namespace	Image
codestream-demo	codestream-demo	codestream	https://codestream/Myapp

1 services

Deployment

Environment	Cluster Endpoint	Namespace
Dev	Dev-AWS-Cluster	codestream-139606
Prod	Prod-AWS-Cluster	codestream

Prod deployment model * ☐ Canary ☐ Rolling Upgrade ☒ Blue-Green

Rollback strategy ☐

Health check URL *

Resultados

Parabéns! O modelo de pipeline inteligente foi usado para criar um pipeline que implanta o aplicativo em suas instâncias Azul-Verde no cluster de produção do Kubernetes na AWS.

Exemplo: Exemplo de código de YAML para algumas tarefas de implantação Azul-Verde

O código YAML que aparece nas tarefas de pipeline do Kubernetes para sua implantação Azul-Verde pode ser semelhante aos exemplos a seguir. Depois que o modelo de pipeline inteligente cria o pipeline, é possível modificar as tarefas conforme necessário para sua própria implantação.

Código do YAML para criar um exemplo de namespace:

```
apiVersion: v1
kind: Namespace
metadata:
  name: codestream-82855
  namespace: codestream-82855
```

Código YAML para criar um exemplo de serviço:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: codestream-demo
  name: codestream-demo
  namespace: bluegreen-799584
spec:
  minReadySeconds: 0
  ports:
    - port: 80
  selector:
    app: codestream-demo
    tier: frontend
  type: LoadBalancer
```

Código YAML para criar um exemplo de implantação:

```
apiVersion: extensions/v1
kind: Deployment
metadata:
  labels:
    app: codestream-demo
  name: codestream-demo
  namespace: bluegreen-799584
spec:
  minReadySeconds: 0
  replicas: 1
  selector:
    matchLabels:
      app: codestream-demo
      tier: frontend
  template:
    metadata:
      labels:
        app: codestream-demo
        tier: frontend
    spec:
      containers:
        - image: ${input.image}:${input.tag}
          name: codestream-demo
          ports:
            - containerPort: 80
              name: codestream-demo
          imagePullSecrets:
            - name: jfrog-2
          minReadySeconds: 0
```

Próximo passo

Para saber mais sobre como usar o vRealize Automation Code Stream, consulte [Capítulo 5 Tutoriais para usar o vRealize Automation Code Stream](#).

Para reverter uma implantação, consulte [Como faço para reverter minha implantação no vRealize Automation Code Stream](#).

Para obter referências adicionais, consulte [Mais recursos para administradores e desenvolvedores do vRealize Automation Code Stream](#).

Como integrar minhas próprias ferramentas de compilação, teste e implantação com o vRealize Automation Code Stream

Como administrador ou desenvolvedor de DevOps, é possível criar scripts personalizados que estendem o recurso do vRealize Automation Code Stream. Com o script, é possível integrar o vRealize Automation Code Stream às suas próprias ferramentas de integração contínua (IC) e entrega contínua (EC) e APIs que compilam, testam e implantam os aplicativos. Os scripts personalizados são especialmente úteis se as APIs do aplicativo não forem expostas publicamente.

O script personalizado pode fazer quase tudo o que é necessário para se integrar às suas ferramentas de compilação, teste e implantação. Por exemplo, ele pode trabalhar com o espaço de trabalho no pipeline para oferecer suporte a tarefas de CI que compilam e testam o aplicativo, além de tarefas de CD que implantam o aplicativo. Ele pode enviar uma mensagem para o Slack quando um pipeline é concluído e muito mais.

Escreva o script personalizado em um dos idiomas compatíveis. No script, inclua a lógica de negócios e defina entradas e saídas. Os tipos de saída podem incluir número, cadeia de caracteres, texto e senha. Você pode criar várias versões de um script personalizado com diferentes lógicas de negócios, entradas e saídas.

Você instrui o pipeline a executar uma versão do seu script em uma tarefa personalizada. Os scripts criados residem na instância do vRealize Automation Code Stream.

Quando um pipeline usar uma integração personalizada, se você tentar excluir a integração personalizada, uma mensagem de erro será exibida e indicará que não é possível excluí-la.

A exclusão de uma integração personalizada remove todas as versões do seu script personalizado. Se você tiver um pipeline existente com uma tarefa personalizada que usa qualquer versão do script, esse pipeline falhará. Para garantir que os pipelines existentes não falhem, você poderá reprovar e retirar a versão do script que não deseja mais usar. Se nenhum pipeline estiver usando essa versão, você poderá excluí-la.

Tabela 5-4. O que fazer após escrever o script personalizado

O que fazer...	Mais informações sobre esta ação...
Adicione uma tarefa personalizada ao pipeline.	<p>A tarefa personalizada:</p> <ul style="list-style-type: none"> ■ É executada no mesmo contêiner de outras tarefas de IC no pipeline. ■ Inclui as variáveis de entrada e saída que o script preenche antes que o pipeline execute a tarefa personalizada. ■ Oferece suporte a vários tipos de dados e vários tipos de metadados definidos como entradas e saídas no script.
Selecione o script na tarefa personalizada.	Declare as propriedades de entrada e saída no script.
Salve seu pipeline e, em seguida, ative-o e execute-o.	Quando o pipeline for executado, a tarefa personalizada chamará a versão do script especificada e executará a lógica de negócios nele, que integra sua ferramenta de compilação, teste e implantação ao vRealize Automation Code Stream.
Após a execução do pipeline, observe as execuções.	Verifique se o pipeline gerou os resultados esperados.

Este exemplo cria uma integração personalizada que conecta o vRealize Automation Code Stream à instância do Slack e publica uma mensagem em um canal do Slack.

Pré-requisitos

- Para escrever seu script personalizado, verifique se há uma destas linguagens: Python 2, Python 3, Node.js ou qualquer uma dessas linguagens Shell: Bash, sh ou zsh.
- Gere uma imagem de contêiner usando o Node.js ou o tempo de execução do Python instalado.

Procedimentos

1 Crie a integração personalizada.

- Clique em **Integrações Personalizadas > Novo** e insira um nome relevante.
- Selecione o ambiente de tempo de execução preferencial.
- Clique em **Criar**.

Seu script é aberto e exibe o código, que inclui o ambiente de tempo de execução necessário. Por exemplo, `runtime: "nodejs"`. O script deve incluir o tempo de execução, utilizado pela imagem do compilador, para que a tarefa personalizada adicionada ao pipeline seja bem-sucedida quando o pipeline for executado. Caso contrário, a tarefa personalizada falhará.

As principais áreas do YAML de integração personalizada incluem o tempo de execução, código, propriedades de entrada e propriedades de saída. Esse procedimento explica vários tipos e sintaxes.

Chaves de YAML de integração personalizada	Descrição
tempo de execução	<p>Ambiente do tempo de execução da tarefa em que o vRealize Automation Code Stream executa o código, que pode ser uma dessas cadeias de caracteres que não diferenciam maiúsculas de minúsculas:</p> <ul style="list-style-type: none"> ■ nodejs ■ python2 ■ python3 ■ shell <p>Se nada for fornecido, o shell será o padrão presumido.</p>
código	Lógica de negócios personalizada a ser executada como parte da tarefa personalizada.
inputProperties	Matriz de propriedades de entrada a serem capturadas como parte da configuração da tarefa personalizada. Essas propriedades normalmente são usadas no código.
outputProperties	Matriz de propriedades de saída que você pode exportar da tarefa personalizada para propagar ao pipeline.

- 2 Declare as propriedades de entrada no script usando os tipos de dados e metadados disponíveis.

As propriedades de entrada são transmitidas como contexto ao script na seção `code:` do YAML.

Chaves de entrada YAML da tarefa personalizada	Descrição	Obrigatório
type	<p>Tipos de entrada para renderização:</p> <ul style="list-style-type: none"> ■ text ■ textarea ■ number ■ checkbox ■ password ■ select 	Sim
name	Nome ou cadeia de caracteres da entrada à tarefa personalizada, que é injetada no código YAML de integração personalizada. Deve ser exclusivo para cada propriedade de entrada definida para uma integração personalizada.	Sim
title	Rótulo da cadeia de caracteres de texto da propriedade de entrada à tarefa personalizada na tela do modelo de pipeline. Se deixado em branco, o name será usado por padrão.	Não
required	Determina se um usuário deve inserir a propriedade de entrada ao configurar a tarefa personalizada. Defina como verdadeiro ou falso. Quando verdadeiro, se um usuário não fornecer um valor ao configurar a tarefa personalizada na tela do pipeline, o estado da tarefa permanecerá como não configurado.	Não

Chaves de entrada YAML da tarefa personalizada		
	Descrição	Obrigatório
placeholder	Texto padrão na área de entrada da propriedade de entrada quando nenhum valor está presente. Mapeia para o atributo do marcador de posição HTML. Compatível apenas com certos tipos de propriedades de entrada.	Não
defaultValue	Valor padrão que preenche a área de entrada da propriedade de entrada quando a tarefa personalizada é renderizada na página do modelo de pipeline.	Não
bindable	Determina se a propriedade de entrada aceita variáveis de cifrão ao modelar a tarefa personalizada na tela do pipeline. Adiciona o indicador \$ ao lado do título. Compatível apenas com certos tipos de propriedades de entrada.	Não
labelMessage	Cadeia de caracteres que funciona como uma dica de ajuda aos usuários. Adiciona um ícone de dica de ferramenta i próximo ao título de entrada.	Não
enum	<p>Recebe uma matriz de valores que exibe as opções para selecionar propriedades de entrada. Compatível apenas com certos tipos de propriedades de entrada.</p> <p>Quando um usuário seleciona uma opção e a salva para a tarefa personalizada, o valor de inputProperty corresponde a esse valor e aparece na modelagem da tarefa personalizada.</p> <p>Por exemplo, o valor 2015.</p> <ul style="list-style-type: none"> ■ 2015 ■ 2016 ■ 2017 ■ 2018 ■ 2019 ■ 2020 	Não
options	<p>Recebe uma matriz de objetos usando optionKey e optionValue.</p> <ul style="list-style-type: none"> ■ optionKey. Valor propagado para a seção de código da tarefa. ■ optionValue. Cadeia de caracteres que exibe a opção na interface do usuário. <p>Compatível apenas com certos tipos de propriedades de entrada.</p> <p>Options:</p> <p>optionKey: key1. Quando selecionado e salvo para a tarefa personalizada, o valor deste inputProperty corresponderia a key1 na seção de código.</p> <p>optionValue: "Rótulo para 1". Exibe o valor de key1 na interface do usuário e não aparece em nenhum outro lugar da tarefa personalizada.</p> <p>optionKey: key2</p> <p>optionValue: "Rótulo para 2"</p> <p>optionKey: key3</p> <p>optionValue: "Rótulo para 3"</p>	Não

Chaves de entrada YAML da tarefa personalizada		
	Descrição	Obrigatório
<code>minimum</code>	Recebe um número que atua como o valor mínimo válido para essa propriedade de entrada. Compatível apenas com a propriedade de entrada do tipo de número.	Não
<code>maximum</code>	Recebe um número que atua como o valor máximo válido para essa propriedade de entrada. Compatível apenas com a propriedade de entrada do tipo de número.	Não

Tabela 5-5. Tipos de dados e metadados de dados compatíveis para scripts personalizados

Tipos de dados compatíveis	Metadados compatíveis para entrada
<ul style="list-style-type: none"> ■ Cadeia de caracteres ■ Texto ■ Lista: como uma lista de qualquer tipo ■ Mapa: como um mapa [string] qualquer ■ Seguro: apresentado como uma caixa de texto de senha, criptografado quando você salva a tarefa personalizada ■ Número ■ Boolean: aparece como caixas de texto ■ URL: igual à cadeia de caracteres, com validação adicional ■ Botão de seleção, opção 	<ul style="list-style-type: none"> ■ tipo: uma da cadeia de caracteres Texto... ■ padrão: valor padrão ■ opções: lista ou um mapa de opções, a ser usado com botão de seleção ou opção ■ mínimo: valor ou tamanho mínimo ■ máximo: valor ou tamanho máximo ■ título: nome detalhado da caixa de texto ■ marcador de posição: marcador de posição da IU ■ descrição: se torna uma dica de ferramenta

Por exemplo:

```
inputProperties:
  - name: message
    type: text
    title: Message
    placeholder: Message for Slack Channel
    defaultValue: Hello Slack
    bindable: true
    labelInfo: true
    labelMessage: This message is posted to the Slack channel link provided in the
code
```

3 Declare as propriedades de saída no script.

O script captura as propriedades de saída da seção de lógica de negócios `code:` do script, na qual você declara o contexto para a saída.

Quando o pipeline é executado, é possível inserir o código de resposta para a saída da tarefa. Por exemplo, 200.

Chaves que o vRealize Automation Code Stream suporta para cada **outputProperty**.

chave	Descrição
type	Atualmente inclui um valor único de label .
name	Chave que o bloco de código do YAML de integração personalizada emite.
title	Rótulo na interface do usuário que exibe outputProperty .

Por exemplo:

```
outputProperties:
  - name: statusCode
    type: label
    title: Status Code
```

- 4 Para interagir com a entrada e saída do script personalizado, obtenha uma propriedade de entrada ou defina uma propriedade de saída usando **context**.

Para uma propriedade de entrada: `(context.getInput("key"))`

Para uma propriedade de saída: `(context.setOutput("key", "value"))`

Para Node.js:

```
var context = require("./context.js")
var message = context.getInput("message");
//Your Business logic
context.setOutput("statusCode", 200);
```

Para Python:

```
from context import getInput, setOutput
message = getInput('message')
//Your Business logic
setOutput('statusCode', '200')
```

Para Shell:

```
# Input, Output properties are environment variables
echo ${message} # Prints the input message
//Your Business logic
export statusCode=200 # Sets output property statusCode
```

- 5 Na seção `code:`, declare toda a lógica de negócios para sua integração personalizada.

Por exemplo, com o ambiente de tempo de execução Node.js:

```
code: |
  var https = require('https');
  var context = require("./context.js")

  //Get the entered message from task config page and assign it to message var
  var message = context.getInput("message");
  var slackPayload = JSON.stringify(
    {
```

```

        text: message
    });

    const options = {
        hostname: 'hooks.slack.com',
        port: 443,
        path: '/YOUR_SLACK_WEBHOOK_PATH',
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'Content-Length': Buffer.byteLength(slackPayload)
        }
    };

    // Makes a https request and sets the output with statusCode which
    // will be displayed in task result page after execution
    const req = https.request(options, (res) => {
        context.setOutput("statusCode", res.statusCode);
    });

    req.on('error', (e) => {
        console.error(e);
    });
    req.write(slackPayload);
    req.end();

```

- 6 Antes que você faça a versão e liberação de script de integração personalizado, faça o download do arquivo de contexto para Python ou Node.js e teste a lógica de negócios incluída no script.
 - a Coloque o cursor no script e, em seguida, clique no botão do arquivo de contexto na parte superior da tela. Por exemplo, se o script estiver em Python, clique em **CONTEXT.PY**.
 - b Modifique o arquivo e salve-o.
 - c No sistema de desenvolvimento, execute e teste o script personalizado com a ajuda do arquivo de contexto.
- 7 Aplique uma versão ao script de integração personalizado.
 - a Clique em **Versão**.
 - b Insira as informações de versão.

- c Clique em **Versão de Liberação** para poder selecionar o script na tarefa personalizada.
- d Para criar a versão, clique em **Criar**.

Creating Version

Version * 1.0

Description New

Change Log New for 1.0

Release Version ☒

CANCEL CREATE

- 8 Para salvar o script, clique em **Salvar**.
- 9 No pipeline, configure o espaço de trabalho.
 - a Clique na guia **Espaço de Trabalho**.
 - b Selecione o host do Docker e o URL da imagem do compilador.

Demo-customTask-nodejs Enabled

Workspace Input Model Output

Workspace

Host * Docker-saas

Builder image URL * node:latest

Image registry --Select Container Registry Endpoint--

Working directory

Cache

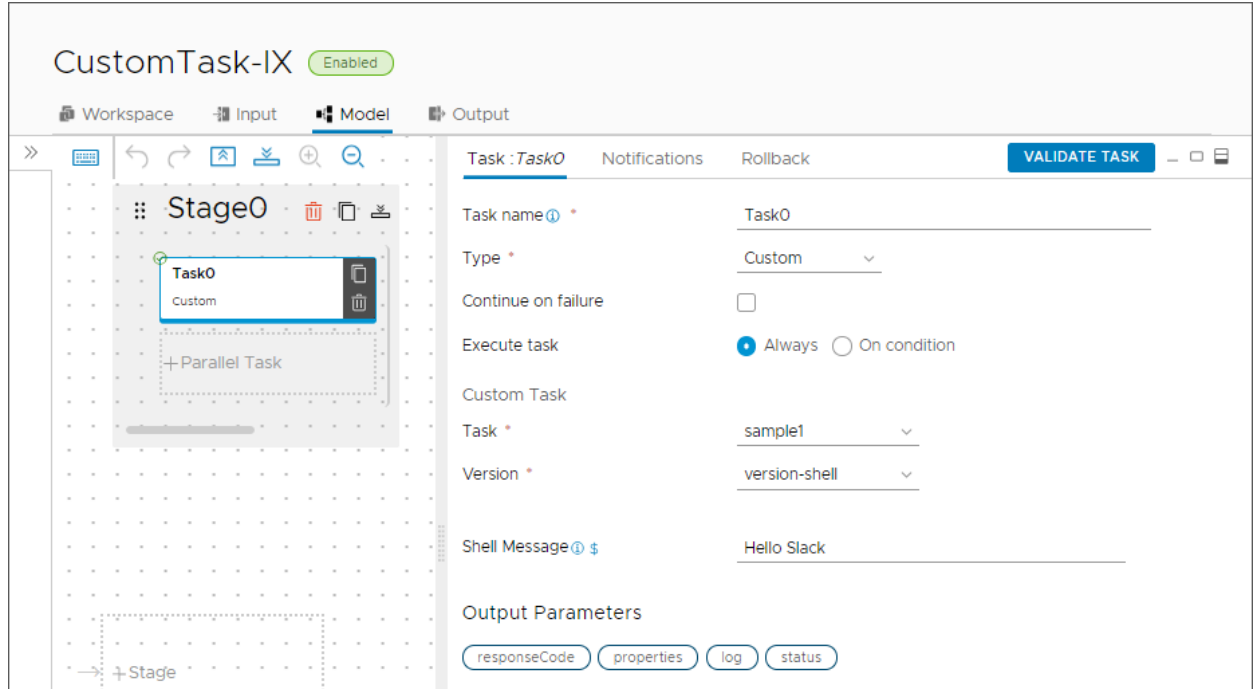
Git clone

If this pipeline links to Git through a webhook, the pipeline triggers on Git events. For CI tasks, the linked Git repository, which receives details from the Git webhook, automatically clones the workspace.

- 10 Adicione uma tarefa personalizada ao pipeline e configure-a.
 - a Clique na guia **Modelo**.
 - b Adicione uma tarefa, selecione o tipo como **Personalizado** e digite um nome relevante.

- c Selecione a versão e o script da integração personalizada.
- d Para exibir uma mensagem personalizada no Slack, digite o texto da mensagem.

Qualquer texto digitado substituirá o `defaultValue` no script de integração personalizado.
Por exemplo:

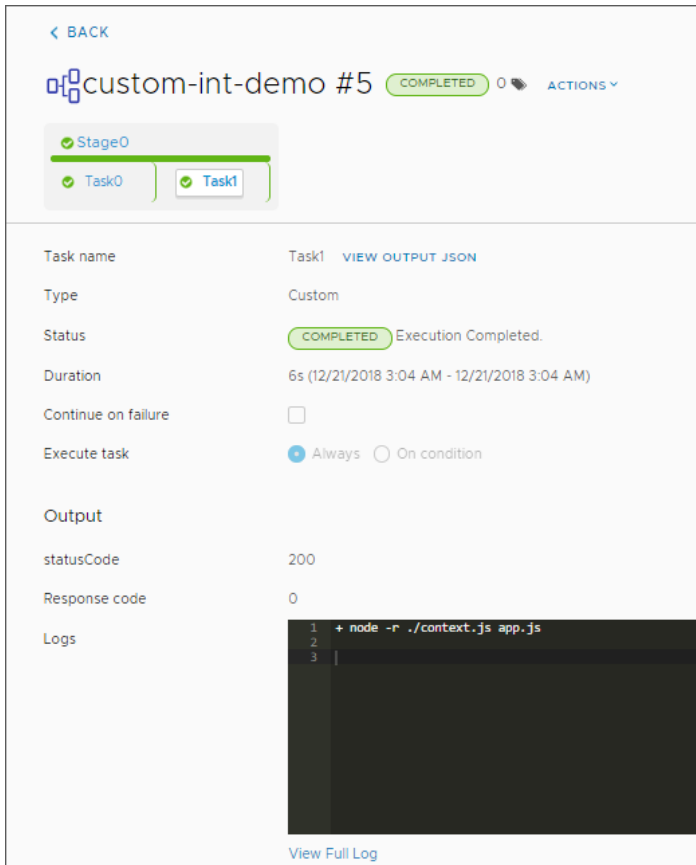


- 11 Salve e ative o pipeline.
 - a Clique em **Salvar**.
 - b Na guia Pipeline, clique em **Ativar Pipeline** para que o círculo se mova para a direita.
- 12 Execute o pipeline.
 - a Clique em **Executar**.
 - b Observe a execução do pipeline.

- c Confirme se a saída inclui o código de status esperado, o código de resposta, o status e a saída declarada.

Você definiu **statusCode** como uma propriedade de saída. Por exemplo, um **statusCode** de 200 pode indicar uma publicação de Slack bem-sucedida e um **responseCode** de 0 pode indicar que o script foi bem-sucedido, sem erros.

- d Para confirmar a saída nos logs de execução, clique em **Execuções**, clique no link do pipeline, clique na tarefa e examine os dados registrados. Por exemplo:



- 13 Se ocorrer um erro, solucione o problema e execute o pipeline novamente.

Por exemplo, se um arquivo ou módulo na imagem de base estiver ausente, será necessário criar outra imagem de base que inclua o arquivo ausente. Em seguida, forneça o arquivo Docker e envie a imagem por meio do pipeline.

Resultados

Parabéns! Você criou um script de integração personalizado que conecta o vRealize Automation Code Stream à instância de Slack e publica uma mensagem em um canal de Slack.

Próximo passo

Continue a criar integrações personalizadas para oferecer suporte ao uso de tarefas personalizadas nos pipelines, para poder ampliar o recurso do vRealize Automation Code Stream na automação do ciclo de liberação do software.

Como usar uma REST API para integrar o vRealize Automation Code Stream a outros aplicativos

O vRealize Automation Code Stream fornece um plug-in REST, que permite integrar o vRealize Automation Code Stream com outros aplicativos que usam uma REST API para que você possa desenvolver e entregar continuamente aplicativos de software que devem interagir entre si. O plug-in REST invoca uma API, que envia e recebe informações entre o vRealize Automation Code Stream e outro aplicativo.

Com o plug-in REST, é possível:

- Integrar sistemas externos com base em REST API em um pipeline do vRealize Automation Code Stream.
- Integrar um pipeline do vRealize Automation Code Stream como parte do fluxo de sistemas externos.

O plug-in REST funciona com qualquer REST API e oferece suporte aos métodos GET, POST, PUT, PATCH e DELETE para enviar ou receber informações entre o vRealize Automation Code Stream e outros aplicativos.

Tabela 5-6. Como preparar um pipeline para se comunicar pela REST API

O que fazer...	O que acontece...
Adicione uma tarefa REST ao pipeline.	A tarefa REST comunica informações entre aplicativos e pode fornecer informações de status para uma tarefa sucessiva no estágio do pipeline.
Na tarefa REST, selecione a ação REST e inclua o URL.	A tarefa de pipeline chama o URL quando o pipeline é executado. Para ações de POST, PUT e PATCH, você deve incluir um payload. No payload, é possível associar as propriedades de pipeline e de tarefa quando o pipeline é executado.
Considere este exemplo.	Exemplo de uso do plug-in REST: É possível adicionar uma tarefa REST para criar uma tag em uma confirmação de Git para uma compilação e fazer com que a tarefa publique uma solicitação para obter o ID de verificação do repositório. A tarefa pode enviar um payload para o repositório e criar uma tag para a compilação, e o repositório pode retornar a resposta com a tag.

Semelhante a usar o plug-in REST para invocar uma API, é possível incluir uma tarefa de Sondagem no pipeline para invocar uma REST API e sondá-la até que seja concluída e a tarefa do pipeline atenda aos critérios de saída.

Também é possível usar as REST APIs para importar e exportar um pipeline, e usar os scripts de exemplo para executar um pipeline.

Este procedimento obtém um URL simples.

Procedimentos

- 1 Para criar um pipeline, clique em **Pipelines > Novo Pipeline > Tela em Branco**.
- 2 No estágio do pipeline, clique em **+ Tarefa Sequencial**.

3 No painel de tarefas, adicione a tarefa REST:

- a Digite um nome para a tarefa.
- b No menu suspenso Tipo, selecione **REST**.
- c Na área de Solicitação REST, selecione **GET**.

Para que a tarefa REST solicite dados de outro aplicativo, selecione o método GET. Para enviar dados para outro aplicativo, selecione o método POST.

- d Digite o URL que identifique o endpoint da REST API. Por exemplo, `https://www.google.com`.

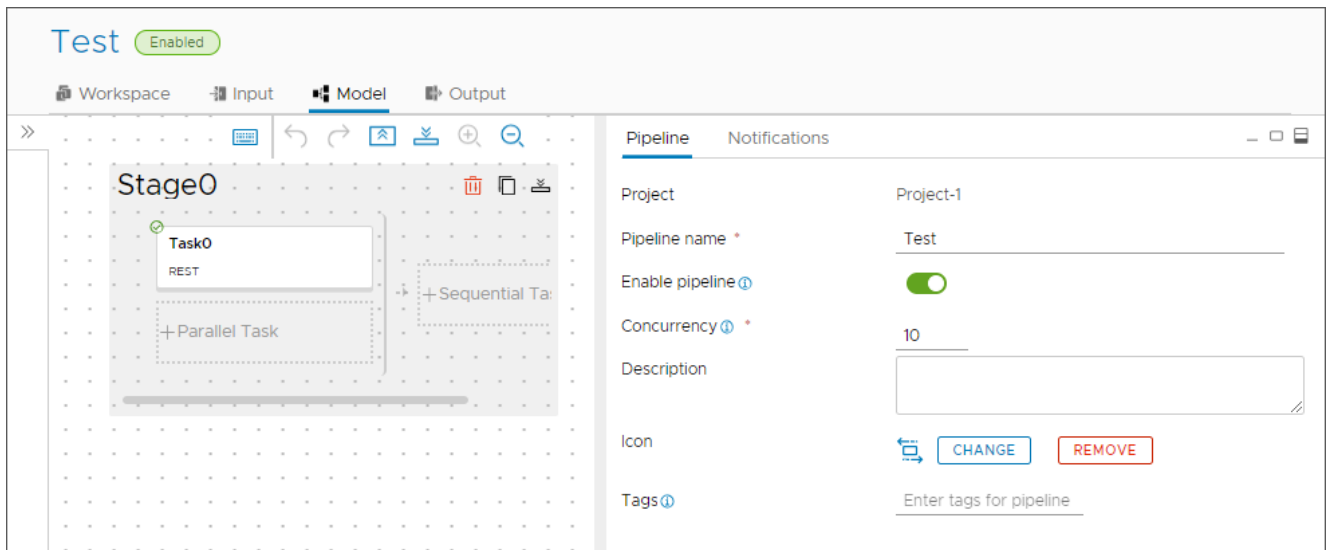
Para que uma tarefa REST importe dados de outro aplicativo, é possível incluir a variável de payload. Por exemplo, para uma ação de importação, é possível digitar `{Stage0.export.responseBody}`. Se o tamanho dos dados da resposta exceder 5 MB, a tarefa REST poderá falhar.

- e Para fornecer autorização para a tarefa, clique em **Adicionar Cabeçalhos** e digite uma chave de cabeçalho e um valor.

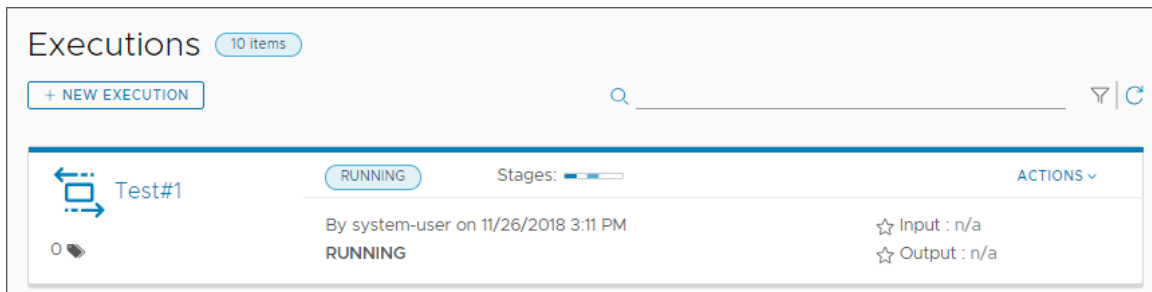
The screenshot displays the vRealize Automation Code Stream interface. The top bar shows 'Test' with a green 'Enabled' button. Below the bar are tabs for 'Workspace', 'Input', 'Model', and 'Output'. The 'Model' tab is active, showing a visual workflow editor. A 'Stage0' container holds a 'Task0' task of type 'REST'. The task is connected to a '+ Sequential Task' block. Below the task, there are options for '+ Parallel Task' and '+ Stage'. The right-hand panel provides configuration details for 'Task : Task0'. It includes fields for 'Task name' (Task0), 'Type' (REST), 'Continue on failure' (checkbox), and 'Execute task' (radio buttons for 'Always' and 'On condition'). The 'REST Request' section shows 'Action' set to 'GET', 'URL' with a placeholder 'Enter URL', and 'Agent endpoint' set to '--Select Agent endpoint--'. The 'Headers' section has 'Accept' and 'Content-Type' set to 'application'. The 'Output Parameters' section shows a 'status' parameter. At the bottom, there are 'SAVE', 'RUN', and 'CLOSE' buttons, along with a timestamp 'Last saved an hour ago'.

4 Para salvar o pipeline, clique em **Salvar**.

- 5 Na guia pipeline, clique em **Ativar pipeline**.



- 6 Clique em **Salvar**, depois clique em **Fechar**.
- 7 Clique em **Executar**.
- 8 Para observar a execução do pipeline, clique em **Execuções**.



- 9 Para verificar se o plug-in REST retorna as informações esperadas, examine a execução do pipeline e os resultados da tarefa.
- Após a conclusão do pipeline, para confirmar que o outro aplicativo retornou os dados solicitados, clique no link para a execução do pipeline.
 - Clique na tarefa REST no pipeline.
 - Na execução do pipeline, clique na tarefa, veja os detalhes da tarefa e verifique se o plug-in REST retornou os resultados esperados.

Os detalhes da tarefa exibem o código de resposta, o corpo, as chaves de cabeçalho e os valores.

[< BACK](#)

Test #2 COMPLETED [ACTIONS](#)

Stage0

Task0

Task name	Task0 VIEW OUTPUT JSON												
Type	REST												
Status	COMPLETED Execution Completed.												
Duration	1s (11/26/2018 3:45 PM - 11/26/2018 3:45 PM)												
Continue on failure	<input type="checkbox"/>												
Execute task	<input checked="" type="radio"/> Always <input type="radio"/> On condition												
Response													
Code	200												
Body	<pre><!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en-IN"><head><meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta content="/images/branding/googleg/1x/googleg_standard_color_128dp.png" itemprop="image"><title>Google</title><script nonce="aMw/ydugkGr9CHU6QQGzg==">(function(){window.google={kEI:'cnf8W6KpJIeVkwXx-aLoDA',kEXPI:'0,1353747,57,50,1150,454,303,1017,1120,286,698,527,730,142,184,293,132,278,420,350,30,524,27,275,401,457,110,114,56,164,2336158,235,32,45,23,6,1,329219,1294,12383,4855,19577,13114,8163,7085,867,605,6,636,2239,3232,5281,1100,3335,2,2,4605,2196,369,1212,2102,4133,1372,224,887,1331,260,1028,2714,1367,573,835,284,2,57,9,727,612,1820,58,2,2,2,189,1108,1712,28,2584,402,1693,664,630,8,300,1270,773,276,1230,609,134,978,430,2487,850,525,2,2,599,5,2,2,1963,528,3,1959,105,465,556,905,1378,966,942,108,334,130,1190,154,386,8,1003,81,7,3,25,463,620,29,989,406,458,1847,93,676,536,427,269,1456,1,2833,313,876,412,2,557,73,1483,698,59,318,273,108,167,323,744,101,1119,38,363,557,438,135,145,155,497,2,718,383,978,487,47,1080,901,387,422,659,359,8,59,32,416,283,9,1,211,2,460,25,60,386,282,528,307,2,67,30,13,1,255,122,143,217,37,628,255,1,1125,264,28,7,2,479,241,129,43,200,188,481,709,29,57,201,337,65,97,167,82,24,7,109,1049,14,758,7,127,179,9,21,261,1413,5977597,12,1861,681,134,43,5997424,90,2800095,4,1572,549,332,445,1,2,80,1,90,0,583,6,307,1,8,1,2,2132,1,1,1,1,1,414,1,748,141,297,169,301,24,2,8,96,50,2,47,22307501',authuser:0,kscs:'c9c918f0_cnf8W6KpJIeVkwXx-aLoDA',kGL:'IN'};google.kHL='en-IN'}});google.time=function(){return(new Date).getTime()};(function(){google.lc=[];google.li=0;google.getEI=function(a){for(var b;a&&(!a.getAttribute) !(b=a.getAttribute("eid")));}a=a.p</pre>												
Headers	<table> <thead> <tr> <th>Header Key</th><th>Header Value</th></tr> </thead> <tbody> <tr> <td>X-Frame-Options</td><td>SAMEORIGIN</td></tr> <tr> <td>Transfer-Encoding</td><td>chunked</td></tr> <tr> <td>Cache-Control</td><td>private, max-age=0</td></tr> <tr> <td>Server</td><td>gws</td></tr> <tr> <td>Alt-Svc</td><td>quic="443";ma=350000;u="44.43.20.25"</td></tr> </tbody> </table>	Header Key	Header Value	X-Frame-Options	SAMEORIGIN	Transfer-Encoding	chunked	Cache-Control	private, max-age=0	Server	gws	Alt-Svc	quic="443";ma=350000;u="44.43.20.25"
Header Key	Header Value												
X-Frame-Options	SAMEORIGIN												
Transfer-Encoding	chunked												
Cache-Control	private, max-age=0												
Server	gws												
Alt-Svc	quic="443";ma=350000;u="44.43.20.25"												

10 Para exibir a saída JSON, clique em **EXIBIR JSON DE SAÍDA**.

```

Stage0.Task0.output
1  {
2    "responseHeaders": {
3      "X-Frame-Options": "SAMEORIGIN",
4      "Transfer-Encoding": "chunked",
5      "Cache-Control": "private, max-age=0",
6      "Server": "gws",
7      "Alt-Svc": "quic=\":443\"; ma=2592000; v=\"44,43,39,35\"",
8      "Set-Cookie": "NID=148
          =RTUKVjVhyg9KvAZR1S8yCCSEw8WosYfn9MwDfQ1N5fNd5DavrXUm5B3J8PyKMX1Z_zRnp3usxtbtpd7YiqRUOSfMkTC7cTERbd
          UmOnj3cTpphe3PHIXJPGHnTSZEWeb3CxtjvIhVols85ezVxatSRyFcg0B_XIHZBkq88uwl1aE; expires=Tue, 28-May-2019
          22:45:06 GMT; path=/; domain=.google.com; HttpOnly",
9      "Expires": "-1",
10     "P3P": "CP=\"This is not a P3P policy! See g.co/p3phelp for more info.\"\"",
11     "X-XSS-Protection": "1; mode=block",
12     "Date": "Mon, 26 Nov 2018 22:45:06 GMT",
13     "Content-Type": "text/html; charset=ISO-8859-1"
14   },
15   "responseBody": "<!doctype html><html itemscope=\"\" itemtype=\"http://schema.org/WebPage\" lang=\"en-IN\"
          ><head><meta content=\"text/html; charset=UTF-8\" http-equiv=\"Content-Type\"><meta content=\"images
          /branding/google/1x/google_standard_color_128dp.png\" itemprop=\"image\"><title>Google</title><script
          nonce=\"aWwW/ydugKgr9CHU6QGg==\"><function() {window.google={kEI: 'cnf8W6Kp3IEvKw0X-aLODA', kEXPI: '0
          ,1353747,57,50,1150,454,303,1017,1120,206,690,527,730,142,184,293,132,278,420,350,30,524,27,275,401,457
          ,110,114,56,164,2336158,235,32,45,23,6,1,329219,1294,12383,4855,19577,13114,8163,7085,867,6056,636,2239
          ,3232,5281,1100,3335,2,2,4605,2196,369,1212,2102,4133,1372,224,887,1331,260,1028,2714,1367,573,835,284
          ,2,579,727,612,1820,58,2,2,2,189,1108,1712,28,2584,402,1693,664,630,8,300,1270,773,276,1230,609,134,978
          ,430,2487,850,525,22,599,5,2,2,1963,528,3,1959,105,465,556,905,1378,966,942,108,334,130,1190,154,386,8
          ,1003,81,7,3,25,463,620,29,989,406,458,1847,93,676,536,427,269,1456,1,2833,313,876,412,2,557,73,1483
          ,698,59,318,273,108,167,323,744,101,1119,30,363,557,430,135,145,155,497,2,718,383,970,487,47,1080,901
          ,387,422,659,359,8,59,32,416,283,9,1,211,2,460,25,60,386,282,528,307,2,67,30,13,1,255,122,143,217,37
          ,628,255,1,1125,264,28,7,2,479,241,129,43,200,188,481,709,29,57,201,337,65,97,167,82,247,109,1049,14

```

Resultados

Parabéns! Você configurou uma tarefa REST que invocou uma REST API e enviou informações entre o vRealize Automation Code Stream e outro aplicativo usando o plug-in REST.

Próximo passo

Continue a usar tarefas REST nos pipelines para executar comandos e integrar o vRealize Automation Code Stream a outros aplicativos para poder desenvolver e fornecer seus aplicativos de software. Considere o uso de tarefas de sondagem para sondar a API até que ela seja concluída e a tarefa do pipeline atenda aos critérios de saída.

Como conectar o vRealize Automation Code Stream aos endpoints

6

O vRealize Automation Code Stream integra-se às ferramentas de desenvolvimento por meio de plug-ins. Os plug-ins com suporte incluem Jenkins, Bamboo, vRealize Operations, Bugzilla, Team Foundation Server, Git e muito mais!

Também é possível desenvolver seus próprios plug-ins para integrar o vRealize Automation Code Stream a outros aplicativos de desenvolvimento.

Para integrar o vRealize Automation Code Stream ao JIRA, não é necessário ter um plug-in externo porque o recurso de criação de tíquetes do JIRA é criado no vRealize Automation Code Stream como um tipo de notificação. Para criar tíquetes de JIRA no status do pipeline, é necessário adicionar um endpoint do JIRA.

Este capítulo inclui os seguintes tópicos:

- [O que são endpoints no vRealize Automation Code Stream](#)
- [Como integrar o vRealize Automation Code Stream ao Jenkins](#)
- [Como integrar o vRealize Automation Code Stream ao Git](#)
- [Como integrar o vRealize Automation Code Stream ao Gerrit](#)
- [Como integrar o vRealize Automation Code Stream ao vRealize Orchestrator](#)

O que são endpoints no vRealize Automation Code Stream

Um endpoint é uma instância de um aplicativo DevOps que se conecta ao vRealize Automation Code Stream para fornecer dados para a execução de seus pipelines, como uma fonte de dados, um repositório ou um sistema de notificação.

Sua função no vRealize Automation Code Stream determina como você usa os endpoints.

- Os administradores e desenvolvedores podem criar, atualizar, excluir e visualizar endpoints.
- Os administradores podem marcar um endpoint como restrito e executar pipelines que usem endpoints restritos.
- Os usuários que têm a função de visualizador podem ver os endpoints, mas não podem criá-los, atualizá-los ou excluí-los.

Para obter mais informações, consulte [Como gerenciar o acesso do usuário e as aprovações no vRealize Automation Code Stream](#).

Para conectar o vRealize Automation Code Stream a um endpoint, adicione uma tarefa ao pipeline e a configure para que ela se comunique com o endpoint. Para verificar se o vRealize Automation Code Stream pode se conectar ao endpoint, clique em **Validar**. Em seguida, ao executar o pipeline, a tarefa de pipeline se conecta ao endpoint para executar a tarefa.

Tabela 6-1. Endpoints que o vRealize Automation Code Stream suporta

Endpoint	O que ele fornece	Versões com suporte	Requisitos
Bamboo	Cria planos de compilação.	6.9.*	
Docker	Compilações nativas podem usar hosts do Docker para implantação.		Quando um pipeline inclui uma imagem do Docker Hub, você deve certificar-se de que essa imagem tenha uma cURL incorporada antes de executar o pipeline. Quando o pipeline for executado, o vRealize Automation Code Stream baixará um arquivo binário que usa cURL para executar comandos.
Registro do Docker	Registra as imagens do contêiner para que um host de compilação do Docker possa receber imagens.	2.7.1	
Gerrit	Conecta-se a um servidor Gerrit para revisões e gatilho	2.14.*	
Git	Dispara pipelines quando os desenvolvedores atualizam o código e o verificam no repositório.	Git Hub Enterprise 2.1.8 Git Lab Enterprise 11.9.12-ee	
Jenkins	Compila artefatos de código.	1.6.* e 2.*	
Jira	Cria um tíquete do Jira quando uma tarefa de pipeline falha.	8.3.*	
Kubernetes	Automatiza as etapas que implantam, dimensionam e gerenciam aplicativos em contêiner.	1.9.*	
PowerShell	Crie tarefas que executam scripts do PowerShell em máquinas Windows ou Linux.	4 e 5	
SSH	Crie tarefas que executam scripts SSH em máquinas Windows ou Linux.	7.0	

Tabela 6-1. Endpoints que o vRealize Automation Code Stream suporta (continuação)

Endpoint	O que ele fornece	Versões com suporte	Requisitos
TFS, Team Foundation Server	Gerencia código-fonte, compilações automatizadas, testes e atividades relacionadas.	2015 e 2017	
vRealize Orchestrator	Organiza e automatiza os fluxos de trabalho no seu processo de compilação.	7.* e 8.*	

Exemplo de código YAML para um endpoint do GitHub

Este exemplo de código YAML define um endpoint do GitHub que pode ser consultado em uma tarefa Git.

```
---
name: github-k8s
tags: [
]
kind: ENDPOINT
properties:
  serverType: GitHub
  repoURL: https://github.com/autouser/testrepok8s
  branch: master
  userName: autouser
  password: encryptedpassword
  privateToken: ''
description: ''
type: scm:git
isLocked: false
---
```

Como integrar o vRealize Automation Code Stream ao Jenkins

O vRealize Automation Code Stream fornece um plug-in Jenkins, que dispara trabalhos do Jenkins que compilam e testam seu código-fonte. O plug-in Jenkins executa casos de teste e pode usar scripts personalizados.

Para executar um trabalho do Jenkins no pipeline, use um servidor Jenkins e adicione o endpoint do Jenkins no vRealize Automation Code Stream. Em seguida, crie um pipeline e adicione uma tarefa Jenkins a ele.

Pré-requisitos

- Configure um servidor Jenkins que execute a versão 1.561 ou mais recente.

- Verifique se você é membro de um projeto no vRealize Automation Code Stream. Se não for, peça a um administrador do vRealize Automation Code Stream para adicioná-lo como membro de um projeto. Consulte [Como adicionar um projeto no vRealize Automation Code Stream](#).
- Verifique se existe um trabalho no servidor Jenkins para que a tarefa do pipeline possa executá-lo.

Procedimentos

- 1 Adicione e valide um endpoint do Jenkins.
 - a Clique em **Endpoints > Novo Endpoint**.
 - b Selecione um projeto e para o tipo de endpoint selecione **Jenkins**. Em seguida, digite um nome e uma descrição.
 - c Se esse endpoint for um componente crítico para os negócios em sua infraestrutura, ative **Marcar como restrito**.
 - d Insira o URL do servidor Jenkins.

- e Digite o nome do usuário e a senha para fazer login no servidor Jenkins. Em seguida, insira as informações restantes.

Tabela 6-2. Informações restantes para o endpoint do Jenkins

Entrada do endpoint	Descrição
Caminho da pasta	<p>Caminho para a pasta que agrupa seus trabalhos. O Jenkins pode executar todos os trabalhos na pasta. É possível criar subpastas. Por exemplo:</p> <ul style="list-style-type: none"> ■ <code>folder_1</code> pode incluir <code>job_1</code> ■ <code>folder_1</code> pode incluir <code>folder_2</code>, que pode incluir <code>job_2</code> <p>Quando você cria um endpoint para <code>folder_1</code>, o caminho da pasta é <code>job/folder_1</code> e o endpoint lista somente <code>job_1</code>.</p> <p>Para obter a lista de trabalhos na pasta herdeira chamada <code>folder_2</code>, é necessário criar outro endpoint que use o caminho da pasta como <code>/job/folder_1/job/folder_2/</code>.</p>
URL	URL do host do servidor Jenkins. Digite o URL na forma de <code>protocol://host:port</code> . Por exemplo: <code>http://192.10.121.13:8080</code>
Intervalo de Sondagem	Duração do intervalo para o vRealize Automation Code Stream pesquisar atualizações para o servidor Jenkins.

Tabela 6-2. Informações restantes para o endpoint do Jenkins (continuação)

Entrada do endpoint	Descrição
Contagem de repetição de solicitação	Número de vezes para repetir a solicitação de compilação agendada para o servidor Jenkins.
Tempo de espera de repetição	Número de segundos a aguardar antes de tentar novamente a solicitação de compilação para o servidor Jenkins.

- f Clique em **Validar** e verifique se o endpoint se conecta ao vRealize Automation Code Stream. Se ele não se conectar, corrija os erros e clique em **Salvar**.

Edit Endpoint

Project: test1

Type: Jenkins

Name *: aa

Description:

Mark restricted: ☐ non-restricted

URL *: http(s)://<server_url>:<port>

Username: username

Password: Enter password [CREATE VARIABLE](#)

Folder Path: /job/DevFolder/

Poll Interval (sec) *: 15

Request Retries *: 5

Retry Wait Time (sec) *: 60

[SAVE](#) [VALIDATE](#) [CANCEL](#)

- 2 Para compilar seu código, crie um pipeline e adicione uma tarefa que use o endpoint do Jenkins.
 - a Clique em **Pipelines > Novo Pipeline > Tela em Branco**.
 - b Clique no estágio padrão.
 - c Na área de tarefas, digite um nome para a tarefa.
 - d Selecione o tipo de tarefa como **Jenkins**.
 - e Selecione o endpoint do Jenkins criado.
 - f No menu suspenso, selecione um trabalho no servidor Jenkins que será executado pelo pipeline.

- g Digite os parâmetros do trabalho.
- h Insira o token de autenticação para o trabalho do Jenkins.

Build and Deploy Enabled ⓘ

Stage0

Build
Jenkins

Test
Jenkins

Parallel Task*

+ Stage

Task : Build

Notifications

VALIDATE TASK

Task name ⓘ *

Type *

Continue On Failure

Execute Task

Jenkins

Endpoint

Job *

Num1 \$

Num2 \$

Token

Output Parameters

Build

Jenkins

☐

☒ Always ☐ On Condition

aa

add_numbers

22

22

status job jobId jobResults jobUri

SAVE


RUN

CLOSE

Last saved a month ago

3 Ative e execute o pipeline e exiba a execução dele.

[< BACK](#)


Build and Deploy #28
COMPLETED
ACTIONS ▾

Stage0

Build
Approval for Deployment
Deployment
Wait for application to start

Task name

Build [VIEW OUTPUT JSON](#)

Type

Jenkins

Status

COMPLETED Execution Completed.

Duration

11s (08/06/2018 12:27 AM - 08/06/2018 12:27 AM)

Continue On Failure

☐

Execute Task

☒ Always ☐ On Condition

Jenkins Job

Endpoint

aa

Job Name

add_numbers

Job ID

1428

Job URL

http://.../job/add_numbers/1428/

Job Result

Key	Value
junitResponse.failCount	0
junitResponse.skipCount	0
junitResponse.totalCount	0
junitResponse.successCount	0
jacocoResponse.lineCoverage	0
jacocoResponse.classCoverage	0

4 Observe os detalhes da execução e o status no painel de pipeline.

É possível identificar quaisquer falhas e por que elas ocorreram. Também é possível ver tendências sobre as durações, conclusões e falhas de execução do pipeline.

Build and Deploy [CLONE](#) [BACK](#)

Recent Executions

Execution#/Stages	Stage0
#29	Failed
#28	Completed
#27	Completed
#26	Failed
#25	Completed
#24	Completed
#23	Failed
#22	Completed
#21	Completed
#20	Failed

● Completed ● Failed ● Running ● Waiting

Execution Details

Execution#	Status	Status Message	Duration	Updated On
#29	FAILED	Execution failed on task 'Stage0.Deployment'. namespaces "prod1" already exists	1m 32s	08/19 10:49PM
#28	COMPLETED	Execution Completed.	3m 42s	08/06 12:30AM
#27	COMPLETED	Execution Completed.	1m 45s	08/06 12:24AM
#26	FAILED	Execution failed on task 'Stage0.Deployment'. Conflict	1m 8s	08/06 12:19AM
#25	COMPLETED	Execution Completed.	2m 11s	08/06 12:07AM
#24	COMPLETED	Execution Completed.	58s	08/05 11:59PM
#23	FAILED	Execution failed on task 'Stage0.Approval for Deployment'. User Operation request has been	4m 55s	08/06 12:03AM

⏏ ▶ ○ ● ○

Resultados

Parabéns! Você integrou o vRealize Automation Code Stream ao Jenkins adicionando um endpoint, criando um pipeline e configurando uma tarefa Jenkins que compila seu código.

Exemplo: Exemplo de YAML para uma tarefa de compilação Jenkins

Para o tipo de tarefa de compilação Jenkins usado neste exemplo, o YAML deve ser semelhante ao código a seguir, com notificações ativadas:

```
test:
  type: Jenkins
```

```

endpoints:
  jenkinsServer: jenkins
input:
  job: Add two numbers
parameters:
  Num1: '23'
  Num2: '23'

```

Próximo passo

Revise as outras seções para saber mais. Consulte [Capítulo 6 Como conectar o vRealize Automation Code Stream aos endpoints](#).

Como integrar o vRealize Automation Code Stream ao Git

O vRealize Automation Code Stream fornece uma maneira de disparar um pipeline se uma alteração de código ocorrer no repositório GitHub, GitLab ou Bitbucket. O gatilho Git usa um endpoint do Git na ramificação do repositório que você deseja monitorar. O vRealize Automation Code Stream conecta-se ao endpoint do Git por meio de um webhook.

Para definir um endpoint do Git no vRealize Automation Code Stream, selecione um projeto e insira a ramificação do repositório Git em que o endpoint está localizado. O projeto agrupa o pipeline com o endpoint e outros objetos relacionados. Ao escolher o projeto na definição de webhook, selecione o endpoint e o pipeline para disparar.

Observação Se você definir um webhook com seu endpoint e, em seguida, editar o endpoint, não poderá alterar os detalhes do endpoint no webhook. Para alterar os detalhes do endpoint, você deve excluir e redefinir o webhook com o endpoint. Consulte [Como usar o gatilho Git no vRealize Automation Code Stream para executar um pipeline](#).

Você pode criar vários webhooks para ramificações diferentes usando o mesmo endpoint Git e fornecendo diferentes valores para o nome da ramificação na página de configuração do webhook. Para criar outro webhook para outra ramificação no mesmo repositório Git, não é necessário clonar o endpoint Git várias vezes para várias ramificações. Em vez disso, forneça o nome da ramificação no webhook, o que permite reutilizar o endpoint Git. Se a ramificação no webhook Git for a mesma que a ramificação no endpoint, você não precisará fornecer o nome da ramificação na página do webhook Git.

Pré-requisitos

- Verifique se é possível acessar o repositório GitHub, GitLab ou Bitbucket ao qual pretende se conectar.
- Verifique se você é membro de um projeto no vRealize Automation Code Stream. Se não for, peça a um administrador do vRealize Automation Code Stream para adicioná-lo como membro de um projeto. Consulte [Como adicionar um projeto no vRealize Automation Code Stream](#).

Procedimentos

1 Defina um endpoint do Git.

- a Clique em **Endpoints > Novo Endpoint**.
- b Selecione um projeto e para o tipo de endpoint e selecione **Git**. Em seguida, digite um nome e uma descrição.
- c Se esse endpoint for um componente crítico para os negócios em sua infraestrutura, ative **Marcar como restrito**.

Quando você usa um endpoint restrito em um pipeline, um administrador pode executar o pipeline e deve aprovar a execução desse pipeline. Se um endpoint ou uma variável estiver marcado como restrito, e um usuário não administrativo disparar o pipeline, este fará uma pausa nessa tarefa e esperará que um administrador a retome.

Um administrador de projeto poderá iniciar um pipeline que inclua endpoints ou variáveis restritos se esses recursos estiverem no projeto em que o usuário é um Administrador de projeto.

Quando um usuário que não é administrador tenta executar um pipeline que inclui um recurso restrito, o pipeline é interrompido na tarefa que usa esse recurso restrito. Em seguida, um administrador deve retomar o pipeline.

Para obter mais informações sobre recursos restritos e funções personalizadas que incluem a permissão chamada **Gerenciar Pipelines Restritos**, consulte:

- [Como gerenciar o acesso do usuário e as aprovações no vRealize Automation Code Stream](#)
- [Capítulo 2 Como configurar o vRealize Automation Code Stream para modelar meu processo de liberação](#)

- d Selecione um dos tipos de servidor Git compatíveis.
- e Insira a URL do repositório com o gateway de API para o servidor no caminho. Por exemplo, insira **`https://api.github.com/vmware-example/repo-example`**.

- f Insira a ramificação no repositório no qual o endpoint está localizado.
- g Selecione o tipo de autenticação e digite o nome de usuário para GitHub, GitLab ou BitBucket. Em seguida, digite a senha, o token privado ou a chave privada que acompanha o nome de usuário.

- Senha. Sua senha fornece acesso completo ao repositório. Você também pode criar uma variável para a senha.

Use variáveis secretas para ocultar e criptografar informações confidenciais. Use a variável restrita para strings, senhas e URLs que devem ser ocultadas e criptografadas e para restringir o uso em execuções. Por exemplo, use uma variável secreta para uma senha ou URL. Você pode usar variáveis secretas e restritas em qualquer tipo de tarefa no seu pipeline.

- Token privado. Esse token é específico para o Git e fornece acesso a uma ação específica. Consulte https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html. Você também pode criar uma variável para o token privado.
- Chave privada. Essa chave SSH é uma chave privada que fornece acesso a um repositório específico. Quando ocorre um evento Git, o vRealize Automation Code Stream usa essa chave para clonar um repositório. Consulte <https://help.github.com/articles/reviewing-your-ssh-keys/>.

- 2 Clique em **Validar** e verifique se o endpoint se conecta ao vRealize Automation Code Stream. Se ele não se conectar, corrija os erros e clique em **Criar**.

New endpoint

Project * test

Type * GIT

Name * DemoApp-Git

Description Git example branch

Mark restricted ☐ non-restricted

Git Server Type * GitHub

Repo URL ⓘ * https://api.github.com/vmware-example/repo-example

ACCEPT CERTIFICATE

Branch * master

Authentication Type * Password

Username * ExampleUser

Password * CREATE VARIABLE

CREATE VALIDATE CANCEL

Próximo passo

Para saber mais, leia as outras seções. Consulte [Como usar o gatilho Git no vRealize Automation Code Stream para executar um pipeline](#).

Como integrar o vRealize Automation Code Stream ao Gerrit

O vRealize Automation Code Stream fornece uma maneira de disparar um pipeline se uma revisão de código ocorrer no seu projeto Gerrit. A definição de gatilho Gerrit inclui o projeto Gerrit e os pipelines a serem executados para diferentes tipos de eventos.

O gatilho do Gerrit usa um ouvinte Gerrit no servidor Gerrit que você deseja monitorar. Para definir um endpoint do Gerrit no vRealize Automation Code Stream, selecione um projeto e insira a URL do servidor Gerrit. Em seguida, especifique o endpoint ao criar um ouvinte Gerrit nesse servidor.

Pré-requisitos

- Verifique se é possível acessar o servidor Gerrit ao qual pretende se conectar.

- Verifique se você é membro de um projeto no vRealize Automation Code Stream. Se não for, peça a um administrador do vRealize Automation Code Stream para adicioná-lo como membro de um projeto. Consulte [Como adicionar um projeto no vRealize Automation Code Stream](#).

Procedimentos

1 Defina um endpoint do Gerrit.

- a Clique em **Configurar > Endpoints** e clique em **Novo Endpoint**.
- b Selecione um projeto e para o tipo de endpoint selecione Gerrit. Em seguida, digite um nome e uma descrição.
- c Se esse endpoint for um componente crítico para os negócios em sua infraestrutura, ative **Marcar como restrito**.
- d Digite o URL do servidor Gerrit.

É possível fornecer um número de porta com o URL ou deixar o valor em branco para usar a porta padrão.

- e Digite o nome de usuário e a senha para o servidor Gerrit.

Se quiser que a senha seja criptografada, clique em **Criar Variável** e selecione o tipo:

- **Segredo**. A senha é resolvida no momento da execução por um usuário com qualquer função.
- **Restrito**. A senha é resolvida no momento da execução por um usuário com a função de Administrador.

Para valor, insira a senha que você deseja proteger, como a senha de um servidor Jenkins.

- f Para a chave privada, digite a chave SSH usada para acessar o servidor Gerrit de forma segura.

Essa chave é a chave privada RSA, localizada no diretório .ssh.

- g (Opcional) Se uma frase-chave estiver associada à chave privada, digite a frase-chave.

Se quiser que a senha seja criptografada, clique em **Criar Variável** e selecione o tipo:

- **Segredo**. A senha é resolvida no momento da execução por um usuário com qualquer função.
- **Restrito**. A senha é resolvida no momento da execução por um usuário com a função Administrador.

Para valor, insira a senha que você deseja proteger, como a senha de um servidor SSH.

- 2 Clique em **Validar** e verifique se o endpoint do Gerrit no vRealize Automation Code Stream se conecta ao servidor Gerrit.

Se ele não se conectar, corrija os erros e, em seguida, tente validá-lo novamente.

New endpoint

Project: test

Type: Gerrit

Name *: Gerrit-Demo-Endpoint

Description:

Mark restricted: ☐ non-restricted

URL *: http://example-gerrit.mycompany.com:8080

Username *: CS_user

Password *:

Private Key *:

Pass Phrase ⓘ:

- 3 Clique em **Criar**.

Próximo passo

Para saber mais, leia as outras seções. Consulte [Como usar o gatilho Gerrit no vRealize Automation Code Stream para executar um pipeline](#).

Como integrar o vRealize Automation Code Stream ao vRealize Orchestrator

O vRealize Automation Code Stream pode se integrar ao vRealize Orchestrator (vRO) para estender seu recurso executando fluxos de trabalho do vRO. O vRealize Orchestrator inclui muitos fluxos de trabalho predefinidos que podem ser integrados a ferramentas de terceiros. Esses fluxos de trabalho ajudam a automatizar e gerenciar seus processos de DevOps, automatizar as operações em massa e muito mais.

Por exemplo, é possível usar um fluxo de trabalho em uma tarefa do vRO no pipeline para ativar um usuário, remover um usuário, mover VMs, integrar a estruturas de teste para testar o código enquanto o pipeline é executado, e muito mais. É possível procurar exemplos de código para os fluxos de trabalho do vRealize Orchestrator em code.vmware.com.

Com um fluxo de trabalho do vRealize Orchestrator, seu pipeline pode executar uma ação enquanto compila, testa e implanta o aplicativo. É possível incluir fluxos de trabalho predefinidos no pipeline ou criar e usar fluxos de trabalho personalizados. Cada fluxo de trabalho inclui entradas, tarefas e saídas.

Para executar um fluxo de trabalho do vRO no pipeline, o fluxo de trabalho deve aparecer na lista de fluxos de trabalho disponíveis na tarefa do vRO incluída no pipeline.

Antes que o fluxo de trabalho possa aparecer na tarefa do vRO no pipeline, um administrador deve realizar as seguintes etapas no vRealize Orchestrator:

- 1 Aplique a tag `CODESTREAM` ao fluxo de trabalho do vRO.
- 2 Marque o fluxo de trabalho do vRO como global.

Pré-requisitos

- Verifique se, como administrador, você pode acessar uma instância local do vRealize Orchestrator. Para obter ajuda, consulte seu próprio administrador e a [documentação do vRealize Orchestrator](#).
- Verifique se você é membro de um projeto no vRealize Automation Code Stream. Se não for, peça a um administrador do vRealize Automation Code Stream para adicioná-lo como membro de um projeto. Consulte [Como adicionar um projeto no vRealize Automation Code Stream](#).
- No vRealize Automation Code Stream, crie um pipeline e adicione um estágio.

Procedimentos

- 1 Como administrador, prepare um fluxo de trabalho do vRealize Orchestrator para que seu pipeline seja executado.
 - a No vRealize Orchestrator, localize o fluxo de trabalho que você precisa usar no pipeline, como um fluxo de trabalho para habilitar um usuário.
Se um fluxo de trabalho que não existe for necessário, é possível criá-lo.
 - b Na barra de pesquisa, insira **Fluxo de trabalho de tag** para encontrar o fluxo de trabalho denominado `Fluxo de trabalho de tag`.
 - c No cartão denominado `Fluxo de trabalho de tag`, clique em **Executar**, o que exibe a área de configuração.
 - d Na área de texto `Fluxo de trabalho marcado`, insira o nome do fluxo de trabalho a ser usado no pipeline do vRealize Automation Code Stream e, em seguida, selecione-o na lista.
 - e Nas áreas de texto `Tag` e `Valor` áreas de texto, insira `CODESTREAM` em letras maiúsculas.
 - f Clique na caixa de seleção denominada **Tag global**.

- g Clique em **Executar**, o que anexa a tag chamada CODESTREAM ao fluxo de trabalho que você precisa selecionar no pipeline do vRealize Automation Code Stream.
- h No painel de navegação, clique em **Fluxos de trabalho** e confirme se a tag chamada CODESTREAM aparece no cartão de fluxo de trabalho em que o pipeline será executado.

Depois de fazer login no vRealize Automation Code Stream e adicionar uma tarefa do vRO ao pipeline, o fluxo de trabalho marcado aparecerá na lista de fluxos de trabalho.

2 No vRealize Automation Code Stream, crie um endpoint para a instância do vRealize Orchestrator.

- a Clique em **Endpoints > Novo Endpoint**.
- b Selecione um projeto.
- c Digite um nome relevante.
- d Insira a URL do endpoint do vRealize Orchestrator.
Use este formato: **https://cava-n-01-234.eng.vmware.com:8281**
Não use este formato: https://cava-n-01-234.eng.vmware.com:8281/vco/api
- e Clique em **Aceitar Certificado** caso a URL inserida precise de um certificado.
- f Digite o nome do usuário e a senha para o servidor vRealize Orchestrator.

3 Prepare seu pipeline para executar a tarefa do vRO.

- a Adicione uma tarefa do vRO ao estágio do pipeline.
- b Digite um nome relevante.
- c Na área Propriedades do Fluxo de Trabalho, selecione o endpoint do vRealize Orchestrator.
- d Selecione o fluxo de trabalho que você marcou como CODESTREAM no vRealize Orchestrator.

Se selecionar um fluxo de trabalho personalizado seu, talvez seja necessário digitar os valores do parâmetro de entrada.

- e Para **Executar tarefa**, clique em **Na condição**.

Task : *vRO workflow*
Notifications
Rollback
VALIDATE TASK

Task name ⓘ *
vRO workflow

Type *
vRO

Duration
NaNms (-)

Continue on failure
☐

Execute task
☐ Always
☒ On condition

Condition ⓘ

Workflow Properties

Endpoint
vROEP

Workflow
Test

Greeting
Hello!

Output Parameters

status
properties

- f Insira as condições aplicáveis à execução do pipeline.

Quando executar o pipeline...	Selecionar condições...
Na Condição	<p>Executa a tarefa de pipeline somente se a condição definida for avaliada como verdadeira. Se a condição for falsa, a tarefa será ignorada.</p> <p>A tarefa do vRO permite que você inclua uma expressão booleana que use os seguintes operandos e operadores.</p> <ul style="list-style-type: none"> ■ Variáveis de pipeline, como <code>\${pipeline.variableName}</code>. Use chaves somente ao inserir variáveis. ■ Variáveis de saída da tarefa, como <code>{Stage1.task1.machines[0].value.hostIp[0]}</code>. ■ Variáveis de associação de pipeline padrão, como <code>{releasePipelineName}</code>. ■ Valores booleanos que não fazem distinção de maiúsculas e minúsculas, como <code>true</code>, <code>false</code>, <code>'true'</code>, <code>'false'</code>. ■ Valores inteiros ou decimais sem aspas. ■ Valores de cadeia de caracteres usados com aspas simples ou duplas, como <code>"test"</code>, <code>'test'</code>. ■ Tipos numéricos e de cadeia de caracteres de valores, como <code>==</code>, <code>Equals</code> e <code>!= Not Equals</code>. ■ Operadores relacionais, como <code>></code>, <code>>=</code>, <code><</code> e <code><=</code>. ■ Lógica booleana, como <code>&&</code> e <code> </code>. ■ Operadores aritméticos, como <code>+</code>, <code>-</code>, <code>*</code> e <code>/</code>. ■ Expressões aninhadas usando parênteses. ■ As cadeias de caracteres que incluem o valor literal ABCD são avaliadas como falso e a tarefa é ignorada. ■ Não há suporte para operadores unários. <p>Um exemplo de condição pode ser <code>\${Stage1.task1.output} == "Passed" \${pipeline.variableName} == 39</code></p>
Sempre	Se você selecionar Sempre , o pipeline executará a tarefa sem condições.

- g Digite uma mensagem de saudação.
- h Clique em **Validar Tarefa** e corrija os erros que ocorrerem.
- 4 Salve, ative e execute o pipeline.
- 5 Após a execução do pipeline, examine os resultados.
- Clique em **Execuções**.
 - Clique no pipeline.
 - Clique na tarefa.
 - Examine os resultados, o valor de entrada e as propriedades.

É possível identificar o ID de execução do fluxo de trabalho, quem respondeu à tarefa e quando, bem como quaisquer comentários incluídos.

Resultados

Parabéns! Você marcou um fluxo de trabalho do vRealize Orchestrator para uso no vRealize Automation Code Stream e adicionou uma tarefa do vRO no pipeline do vRealize Automation Code Stream para que ele execute um fluxo de trabalho que automatiza uma ação no seu ambiente do DevOps.

Exemplo: Formato de saída da tarefa do vRO

O formato de saída para uma tarefa do vRO é semelhante a este exemplo.

```
[{
    "name": "result",
    "type": "STRING",
    "description": "Result of workflow run.",
    "value": ""
},
{
    "name": "message",
    "type": "STRING",
    "description": "Message",
    "value": ""
}]
```

Próximo passo

Continue incluindo tarefas de fluxo de trabalho do vRO nos pipelines para poder automatizar tarefas em seus ambientes de desenvolvimento, teste e produção.

Disparando pipelines no vRealize Automation Code Stream

7

O vRealize Automation Code Stream pode disparar um pipeline para você quando determinados eventos ocorrerem.

Por exemplo, você pode:

- Usar o gatilho do Docker para executar um pipeline quando um novo artefato for criado ou atualizado.
- Usar o gatilho Git para disparar um pipeline quando os desenvolvedores atualizarem o código.
- Usar o gatilho Gerrit para disparar um pipeline quando os desenvolvedores revisarem o código.
- Usar o comando `curl` para que o Jenkins dispare o pipeline após a conclusão de uma compilação.

Este capítulo inclui os seguintes tópicos:

- [Como usar o gatilho do Docker no vRealize Automation Code Stream para executar um pipeline de entrega contínua](#)
- [Como usar o gatilho Git no vRealize Automation Code Stream para executar um pipeline](#)
- [Como usar o gatilho Gerrit no vRealize Automation Code Stream para executar um pipeline](#)

Como usar o gatilho do Docker no vRealize Automation Code Stream para executar um pipeline de entrega contínua

Como administrador ou desenvolvedor do vRealize Automation Code Stream, você pode usar o gatilho Docker no vRealize Automation Code Stream. O gatilho do Docker executa um pipeline de entrega contínua (CD) independente sempre que um artefato de construção é criado ou atualizado. O gatilho do Docker executa o pipeline de CD, que envia o artefato novo ou atualizado como uma imagem de contêiner para um repositório do Docker Hub. O pipeline de CD pode ser executado como parte de suas compilações automatizadas.

Por exemplo, para implantar continuamente a imagem do contêiner atualizado por meio do seu pipeline de CD, use o gatilho do Docker. Quando a imagem do contêiner é verificada no registro do Docker, o webhook no Docker Hub notifica o vRealize Automation Code Stream que a imagem foi alterada. Essa notificação aciona o pipeline de CD para ser executado com a imagem de contêiner atualizada e carrega a imagem no repositório do Docker Hub.

Para usar o gatilho do Docker, você executa várias etapas no vRealize Automation Code Stream.

Tabela 7-1. Como usar o gatilho do Docker

O que fazer...	Mais informações sobre esta ação...
Criar um endpoint de registro do Docker.	<p>Para que o vRealize Automation Code Stream dispare o pipeline, é necessário ter um endpoint de registro do Docker. Se o endpoint não existir, é possível selecionar uma opção que o crie ao adicionar o webhook para o gatilho do Docker.</p> <p>O endpoint de registro do Docker inclui o URL para o repositório do Docker Hub.</p>
Adicionar parâmetros de entrada ao pipeline que injeta parâmetros do Docker automaticamente quando o pipeline é executado.	<p>Você pode injetar parâmetros do Docker no pipeline. Os parâmetros podem incluir o nome do proprietário do evento do Docker, a imagem, o nome do repositório, o namespace do repositório e a tag.</p> <p>No seu pipeline de CD, inclua parâmetros de entrada que o webhook do Docker passa para o pipeline antes do pipeline disparar.</p>
Criar um webhook do Docker.	<p>Ao criar o webhook do Docker no vRealize Automation Code Stream, ele também cria um webhook correspondente no Docker Hub. O webhook do Docker no vRealize Automation Code Stream se conecta ao Docker Hub por meio do URL que você inclui no webhook.</p> <p>Os webhooks se comunicam uns com os outros e disparam o pipeline quando um artefato é criado ou atualizado no Docker Hub.</p> <p>Se você atualizar ou excluir o webhook do Docker no vRealize Automation Code Stream, o webhook no Docker Hub também será atualizado ou excluído.</p>
Adicionar e configurar uma tarefa do Kubernetes no pipeline.	<p>Quando um artefato é criado ou atualizado no repositório do Docker Hub, o pipeline é acionado. Em seguida, ele implanta o artefato pelo pipeline no host do Docker no seu cluster do Kubernetes.</p>
Incluir uma definição de YAML local na tarefa.	<p>A definição de YAML que você aplica à tarefa de implantação inclui a imagem do contêiner do Docker e todas as chaves secretas obrigatórias para a transferência da imagem do repositório para implantação.</p>

Quando um artefato é criado ou atualizado no repositório do Docker Hub, o webhook no Docker Hub notifica o webhook no vRealize Automation Code Stream, que dispara o pipeline. As seguintes ações ocorrem:

- 1 O Docker Hub envia uma solicitação POST para o URL no webhook.
- 2 O vRealize Automation Code Stream executa o gatilho do Docker.
- 3 O gatilho do Docker inicia seu pipeline de CD.
- 4 O pipeline de CD envia o artefato para o repositório do Docker Hub.

- 5 O vRealize Automation Code Stream dispara o webhook do Docker, que executa um pipeline de CD que implanta o artefato no host do Docker.

Neste exemplo, é criado um endpoint do Docker e um webhook do Docker no vRealize Automation Code Stream que implanta seu aplicativo no cluster do Kubernetes de desenvolvimento. As etapas incluem o código de exemplo para o payload que o Docker publica para o URL no webhook, o código de API que ele usa e o código de autenticação com o token seguro.

Pré-requisitos

- Verifique se existe um pipeline de entrega contínua (CD) na instância do vRealize Automation Code Stream. Verifique também se ele inclui uma ou mais tarefas do Kubernetes que implantam seu aplicativo. Consulte [Capítulo 4 Planejamento para compilar, integrar e entregar seu código de forma nativa no vRealize Automation Code Stream](#).
- Verifique se é possível acessar um cluster do Kubernetes existente no qual o seu pipeline de CD pode implantar seu aplicativo para desenvolvimento.
- Verifique se você é membro de um projeto no vRealize Automation Code Stream. Se não for, peça a um administrador do vRealize Automation Code Stream para adicioná-lo como membro de um projeto. Consulte [Como adicionar um projeto no vRealize Automation Code Stream](#).

Procedimentos

- 1 Criar um endpoint de registro do Docker.
 - a Clique em **Endpoints**.
 - b Clique em **Novo Endpoint**.
 - c Digite um nome relevante.
 - d Selecione o tipo de servidor como **Docker Hub**.

- e Digite o URL do repositório do Docker Hub.
- f Digite o nome e a senha usados para acessar o repositório.

Edit Endpoint

Project: AWS_PGProj

Type: Docker Registry

Name:

Description:

Mark as restricted: ☐ non-restricted

Server Type:

Repo URL:

Authentication Type:

Username:

Password: [CREATE VARIABLE](#)

[SAVE](#) [VALIDATE](#) [CANCEL](#)

- 2 No seu pipeline de CD, defina as propriedades de entrada para injetar automaticamente os parâmetros do Docker quando o pipeline for executado.

sm-1 Enabled

Workspace **Input** Model Output

Input Parameters ⓘ

Auto inject parameters: ☐ Gerrit ☐ Git ☒ Docker ☐ None

[ADD](#) [ADD/REMOVE INJECTED PARAMETERS](#)

Starred ⓘ	Name
<input type="checkbox"/>	DOCKER_EVENT_OWNER_NAME
<input type="checkbox"/>	DOCKER_IMAGE
<input type="checkbox"/>	DOCKER_REPO_NAME
<input type="checkbox"/>	DOCKER_REPO_NAMESPACE
<input type="checkbox"/>	DOCKER_TAG

- 3 Criar um webhook do Docker.
 - a Clique em **Gatilhos > Docker**.
 - b Clique em **Novo Webhook para Docker**.
 - c Selecione um projeto.
 - d Digite um nome relevante.

- e Selecione o endpoint de registro do Docker.

Se o endpoint ainda não existir, clique em **Criar Endpoint** e crie-o.

- f Selecione o pipeline com os parâmetros injetados do Docker para o webhook disparar. Consulte [Etapa 2](#).

Se o pipeline tiver sido configurado com os parâmetros de entrada adicionados personalizados, a lista de parâmetros de entrada exibirá parâmetros e valores. É possível digitar valores para os parâmetros de entrada que serão passados para o pipeline com o evento de gatilho. Também é possível deixar os valores em branco ou usar os valores padrão, se definidos.

Para obter mais informações sobre parâmetros na guia de entrada, consulte [Planejando uma compilação nativa de CI/CD no vRealize Automation Code Stream antes de adicionar tarefas manualmente](#).

- g Digite o Token de API.

O token de API do CSP autentica você para conexões de API externas com o vRealize Automation Code Stream. Para obter o token de API:

- 1 Clique em **Gerar Token**.
- 2 Insira o endereço de e-mail associado ao seu nome de usuário e senha e clique em **Gerar**.

O token gerado será válido por seis meses. Ele também é conhecido como token de atualização.

- Para manter o token como uma variável para uso futuro, clique em **Criar Variável**, insira um nome para a variável e clique em **Salvar**.
- Para manter o token como um valor de texto para uso futuro, clique em **Copiar** e cole o token em um arquivo de texto para salvar localmente.

Você pode optar por criar uma variável e armazenar o token em um arquivo de texto para uso futuro.

- 3 Clique em **Fechar**.

- h Insira a imagem de compilação.

- i Insira uma tag.

Docker

Activity Webhooks for Docker

Webhook URL [ⓘ] https://[redacted]m/codestream/api/registry-webhook-listeners/54bd030d...

Project test

Name * sm-1-Docker-WH

Description Docker webhook trigger for sm-1

Docker Registry Docker-Register-Endpoint

Pipeline * sm-1 [ⓧ]

API token * [redacted] [✖] [CREATE VARIABLE](#) [GENERATE TOKEN](#)

Image [ⓘ] Image

Tag [ⓘ] Tags

[SAVE](#) [CANCEL](#)

- j Clique em **Salvar**.

O cartão de webhook aparece com o webhook do Docker habilitado. Se você quiser fazer um envio fictício ao repositório do Docker Hub sem disparar o webhook do Docker e executar um pipeline, clique em **Desativar**.

- 4 No pipeline de CD, configure a tarefa de implantação do Kubernetes.
 - a Nas propriedades de tarefa do Kubernetes, selecione o cluster do Kubernetes de desenvolvimento.
 - b Selecione a ação **Criar**.

- c Selecione a **Definição Local** para a fonte de payload.
- d Em seguida, selecione seu arquivo YAML local.

Por exemplo, o Docker Hub pode publicar essa definição de YAML local como o payload para o URL no webhook:

```
{
  "callback_url": "https://registry.hub.docker.com/u/svendowideit/testhook/hook/2141b5bi5i5b02bec211i4eeih0242eg11000a/",
  "push_data": {
    "images": [
      "27d47432a69bca5f2700e4dff7de0388ed65f9d3fb1ec645e2bc24c223dc1cc3",
      "51a9c7c1f8bb2fa19bcd09789a34e63f35abb80044bc10196e304f6634cc582c",
      "...",
    ],
    "pushed_at": 1.417566161e+09,
    "pusher": "trustedbuilder",
    "tag": "latest"
  },
  "repository": {
    "comment_count": 0,
    "date_created": 1.417494799e+09,
    "description": "",
    "dockerfile": "#\n# BUILD\u0009\u0009docker build -t svendowideit/apt-cacher .\n# RUN\u0009\u0009docker run -d -p 3142:3142 -name apt-cacher-run apt-cacher\n#\n# and then you can run containers with:\n#\n\u0009\u0009docker run -t -i -rm -e http_proxy http://192.168.1.2:3142/debian bash\n#\nFROM\u0009\u0009ubuntu\n\n\nVOLUME\u0009\u0009[/var/cache/apt-cacher-ng]\nRUN\u0009\u0009apt-get update ; apt-get install -yq apt-cacher-ng\n\nEXPOSE\n\u0009\u000993142\nCMD\u0009\u0009chmod 777 /var/cache/apt-cacher-ng ; /etc/init.d/apt-cacher-ng start ; tail -f /var/log/apt-cacher-ng/*\n",
    "full_description": "Docker Hub based automated build from a GitHub repo",
    "is_official": false,
    "is_private": true,
    "is_trusted": true,
    "name": "testhook",
    "namespace": "svendowideit",
    "owner": "svendowideit",
    "repo_name": "svendowideit/testhook",
    "repo_url": "https://registry.hub.docker.com/u/svendowideit/testhook/",
    "star_count": 0,
    "status": "Active"
  }
}
```

A API que cria o webhook no Docker Hub

usa o formulário: https://cloud.docker.com/v2/repositories/%3CUSERNAME%3E/%3CREPOSITORY%3E/webhook_pipeline/

O corpo do código JSON é semelhante a:

```
{
  "name": "demo_webhook",
```

```
"webhooks": [
{
"name": "demo_webhook",
"hook_url": "http://www.google.com"
}
]
}
```

Para receber eventos do servidor Docker Hub, o esquema de autenticação do webhooker do Docker que você criar no vRealize Automation Code Stream usará um mecanismo de autenticação de lista de permissões com um token de cadeia de caracteres aleatório no webhook. Ele filtra eventos com base no token seguro, que pode ser anexado a `hook_url`.

O vRealize Automation Code Stream pode verificar qualquer solicitação do servidor Docker Hub usando o token seguro configurado. Por exemplo: `hook_url = IP:Port/pipelines/api/docker-hub-webhooks?secureToken = ""`

- 5 Crie um artefato do Docker no repositório do Docker Hub. Ou atualize um artefato existente.
- 6 Para confirmar que o disparo ocorreu e ver a atividade no webhook do Docker, clique em **Gatilhos > Docker > Atividade**.

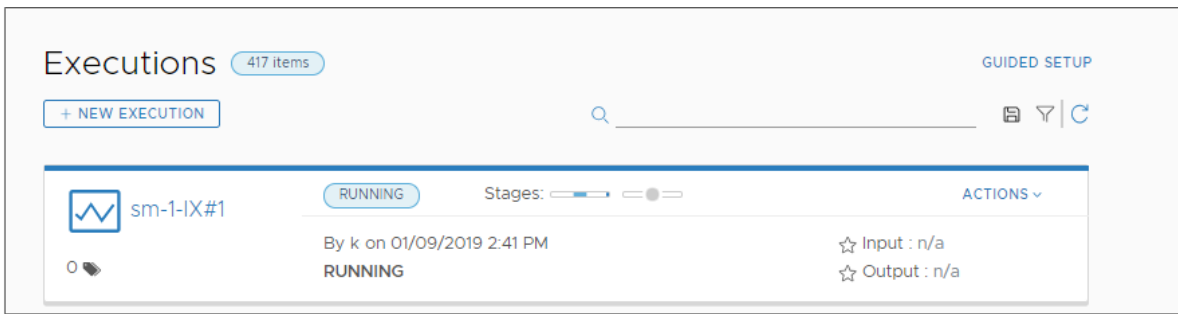
Docker

GUIDED SETUP

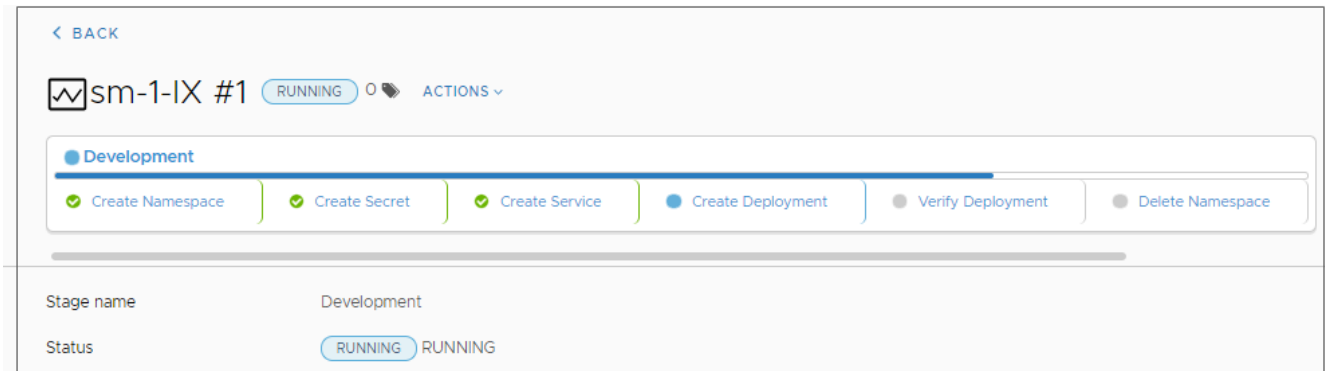
Activity
Webhooks for Docker

Commit Time	Webhook	Image	Tag	Owner	Repository	Pipeline	Execution Status
01/09/2019 10:59 AM	dt11-Docker-WH	admin/repo:s1	s1	admin	repo		SKIPPED
01/09/2019 10:59 AM	fvxd-Docker-WH	admin/repo:s1	s1	admin	repo		SKIPPED
01/09/2019 10:59 AM	test-do-Docker-WH	admin/repo:s1	s1	admin	repo		SKIPPED
01/09/2019 10:59 AM	sm-Docker-WH	admin/repo:s1	s1	admin	repo		SKIPPED
01/09/2019 10:59 AM	t-token-Docker-WH	admin/repo:s1	s1	admin	repo		FAILED
01/09/2019 10:57 AM	dt11-Docker-WH	admin/repo:s01	s01	admin	repo		SKIPPED
01/09/2019 10:57 AM	sm-Docker-WH	admin/repo:s01	s01	admin	repo		SKIPPED
01/09/2019 10:57 AM	test-do-Docker-WH	admin/repo:s01	s01	admin	repo		SKIPPED
01/09/2019 10:57 AM	fvxd-Docker-WH	admin/repo:s01	s01	admin	repo		SKIPPED

- 7 Clique em **Execuções** e rastreie o pipeline à medida que ele é executado.



- 8 Clique no estágio de execução e exiba as tarefas à medida que o pipeline é executado.



Resultados

Parabéns! Configure o gatilho do Docker para executar o pipeline de CD continuamente. Seu pipeline agora pode carregar artefatos novos e atualizados do Docker para o repositório Docker Hub.

Próximo passo

Verifique se o artefato novo ou atualizado está implantado no host do Docker no cluster do Kubernetes de desenvolvimento.

Como usar o gatilho Git no vRealize Automation Code Stream para executar um pipeline

Como administrador ou desenvolvedor do vRealize Automation Code Stream, você pode usar o gatilho Git para integrar o vRealize Automation Code Stream ao ciclo de vida do Git. Quando você faz uma alteração de código no GitHub, GitLab ou Bitbucket Enterprise, o evento se comunica com o vRealize Automation Code Stream por meio de um webhook e dispara um pipeline para execução. O webhook funciona com versões empresariais locais do GitLab, do GitHub e Bitbucket quando o vRealize Automation Cloud Assembly e a versão empresarial estão ambos acessíveis na mesma rede.

Ao adicionar o webhook para Git no vRealize Automation Code Stream, ele também cria um webhook no repositório do GitHub, GitLab ou Bitbucket. Se você atualizar ou excluir o webhook posteriormente, o webhook no GitHub, GitLab ou Bitbucket também será atualizado ou excluído.

Sua definição de webhook deve incluir um endpoint do Git na ramificação do repositório que deseja monitorar. O vRealize Automation Code Stream usa o endpoint do Git para criar o webhook. Se o endpoint não existir, você poderá criá-lo ao adicionar o webhook. Este exemplo pressupõe que você tenha um endpoint predefinido do Git no GitHub.

Você pode criar vários webhooks para ramificações diferentes usando o mesmo endpoint Git e fornecendo diferentes valores para o nome da ramificação na página de configuração do webhook. Para criar outro webhook para outra ramificação no mesmo repositório Git, não é necessário clonar o endpoint Git várias vezes para várias ramificações. Em vez disso, forneça o nome da ramificação no webhook, o que permite reutilizar o endpoint Git. Se a ramificação no webhook Git for a mesma que a ramificação no endpoint, você não precisará fornecer o nome da ramificação na página do webhook Git.

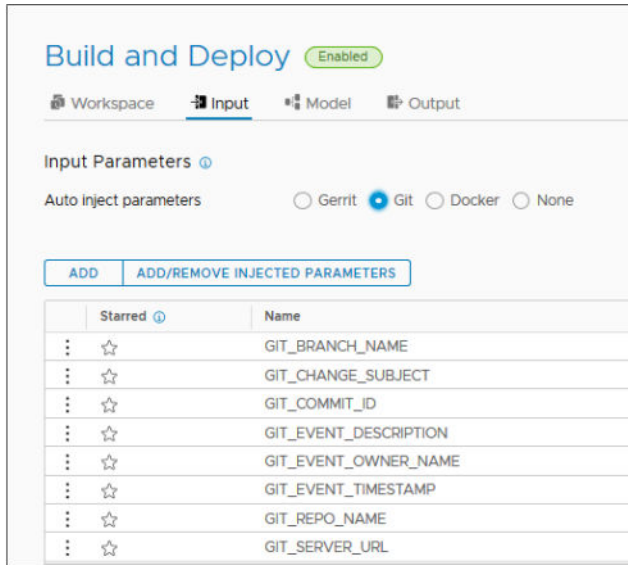
Este exemplo mostra como usar o gatilho Git com um repositório GitHub, mas os pré-requisitos incluem as preparações necessárias se outro tipo de servidor Git for usado.

Pré-requisitos

- Verifique se você é membro de um projeto no vRealize Automation Code Stream. Se não for, peça a um administrador do vRealize Automation Code Stream para adicioná-lo como membro de um projeto. Consulte [Como adicionar um projeto no vRealize Automation Code Stream](#).
- Verifique a existência de um endpoint do Git na ramificação do GitHub que deseja monitorar. Consulte [Como integrar o vRealize Automation Code Stream ao Git](#).
- Verifique se você tem direitos para criar um webhook no repositório Git.
- Se estiver configurando um webhook no GitLab, altere as configurações de rede padrão no GitLab Enterprise para habilitar solicitações de saída e permitir a criação de webhooks locais.

Observação Essa alteração só é necessária para o GitLab Enterprise. Essas configurações não se aplicam ao GitHub ou Bitbucket.

- a Faça login na instância do GitLab Enterprise como administrador.
- b Acesse as configurações de rede usando uma URL, como `http://{gitlab-server}/admin/application_settings/network`.
- c Expanda **Solicitações de saída** e clique em:
 - Permita solicitações para a rede local a partir de webhooks e serviços Web.
 - Permita solicitações para a rede local a partir de um gancho de sistema.
- Para os pipelines que você deseja disparar, verifique se você definiu as propriedades de entrada para injetar parâmetros Git quando o pipeline for executado.



Para obter informações sobre os parâmetros de entrada, consulte [Planejando uma compilação nativa de CI/CD no vRealize Automation Code Stream](#) antes de adicionar tarefas manualmente.

Procedimentos

- 1 No vRealize Automation Code Stream, clique em **Gatilhos > Git**.
- 2 Clique na guia **Webhooks para Git**, depois clique em **Novo Webhook para Git**.
 - a Selecione um projeto.
 - b Digite um nome significativo e uma descrição para o webhook.
 - c Selecione um endpoint Git configurado para a ramificação que você deseja monitorar.

Quando você cria seu webhook, a definição do webhook inclui os detalhes atuais do endpoint.

- Mais tarde, se você alterar o tipo Git, o tipo de servidor Git ou a URL do repositório Git no endpoint, o webhook não poderá mais disparar um pipeline, pois ele tentará acessar o repositório Git usando os detalhes do endpoint original. Você deverá excluir o webhook e criá-lo novamente com o endpoint.
- Mais tarde, se você alterar o tipo de autenticação, o nome do usuário ou a senha no endpoint, o webhook continuará a funcionar.

Consulte [Como integrar o vRealize Automation Code Stream ao Git](#).

- d (Opcional) Insira a ramificação que você deseja que o webhook monitore.

Se ela não for especificada, o webhook monitorará a ramificação configurada para o endpoint Git.

- e (Opcional) Gere um token secreto para o webhook.

Se usado, o vRealize Automation Code Stream gera um token de cadeia de caracteres aleatório para o webhook. Depois, quando o webhook recebe os dados de evento do Git, ele os envia com o token secreto. O vRealize Automation Code Stream usa as informações para determinar se as chamadas são provenientes da origem esperada, como a instância do GitHub, o repositório e a ramificação configurados. O token secreto fornece uma camada extra de segurança que é usada para verificar se os dados de evento do Git são provenientes da origem correta.

- f (Opcional) Forneça inclusões de arquivo ou exclusões como condições para o gatilho.
- Forneça inclusões de arquivo para que, quando qualquer um dos arquivos em uma confirmação corresponder aos arquivos especificados nos caminhos de inclusão ou no Regex, os pipelines sejam disparados. Quando um regex especificado, o vRealize Automation Code Stream apenas dispara os pipelines quando os nomes de arquivo no conjunto de alterações correspondem à expressão fornecida. O filtro Regex é útil ao configurar um gatilho para vários pipelines em um único repositório.
 - Forneça exclusões de arquivos para que, quando todos os arquivos em uma confirmação corresponderem aos arquivos especificados nos caminhos de exclusão ou no Regex, os pipelines não sejam disparados.
 - Quando ativada, priorizar exclusão garante que os pipelines não sejam disparados, mesmo que qualquer um dos arquivos em uma confirmação corresponda aos arquivos especificados nos caminhos de exclusão ou no Regex. A configuração padrão é desativada.

Se as condições para inclusão e exclusão forem atendidas, os pipelines não serão disparados.

No exemplo a seguir, as inclusões e as exclusões de arquivo são condições para o gatilho.

The screenshot shows a configuration window titled "File" with an information icon. It contains two sections: "Inclusions" and "Exclusions".

Section	Filter Type	Path/Expression	Action
Inclusions	PLAIN	runtime/src/main/a.java	-
	REGEX	([a-z A-Z]+[/])[a-z A-Z])+	- +
Exclusions	PLAIN	runtime/pom.xml	-
	PLAIN	runtime/demo.yaml	- +

At the bottom, there is a toggle switch for "Prioritize Exclusion", which is currently turned off.

- Para inclusões de arquivo, uma confirmação com qualquer alteração em `runtime/src/main/a.java` ou qualquer arquivo Java disparará pipelines configurados na configuração de eventos.
 - Para exclusões de arquivos, uma confirmação com alterações somente em ambos os arquivos não disparará os pipelines configurados nas configurações de evento.
- g Para o evento do Git, selecione uma solicitação **Push** ou **Pull**.

h Digite o Token de API.

O token de API do CSP autentica você para conexões de API externas com o vRealize Automation Code Stream. Para obter o token de API:

1 Clique em **Gerar Token**.2 Insira o endereço de e-mail associado ao seu nome de usuário e senha e clique em **Gerar**.

O token gerado será válido por seis meses. Ele também é conhecido como token de atualização.

- Para manter o token como uma variável para uso futuro, clique em **Criar Variável**, insira um nome para a variável e clique em **Salvar**.
- Para manter o token como um valor de texto para uso futuro, clique em **Copiar** e cole o token em um arquivo de texto para salvar localmente.

Você pode optar por criar uma variável e armazenar o token em um arquivo de texto para uso futuro.

3 Clique em **Fechar**.

i Selecione o pipeline para o webhook disparar.

Se o pipeline tiver sido configurado com os parâmetros de entrada adicionados personalizados, a lista de parâmetros de entrada exibirá parâmetros e valores. É possível digitar valores para os parâmetros de entrada que serão passados para o pipeline com o evento de gatilho. Também é possível deixar os valores em branco ou usar os valores padrão, se definidos.

Para obter informações sobre a inserção automática de parâmetros de entrada para gatilhos do Git, consulte [Pré-requisitos](#).

j Clique em **Criar**.

O webhook aparece como um novo cartão.

3 Clique no cartão de webhook.

Quando o formulário de dados do webhook reaparecer, você verá um URL do webhook adicionado à parte superior do formulário. O Git webhook conecta-se ao repositório do GitHub por meio do URL do webhook.

Git

Activity
Webhooks for Git

Webhook URL ⓘ

https://ca[REDACTED]om/codestream/api/git-webhook-listeners/963b2287-527f-4e9b

Project

test

Name *

test-webhook

Description

Description

Endpoint

DemoApp-Git

Branch ⓘ

master

Secret token ⓘ *

GYH0cBWZx4dUn47Y/KA8H/BOKts=

GENERATE

File ⓘ

--Select--
Value
+

--Select--
Value
+

Prioritize Exclusion

☐

Trigger

☒ PUSH
☐ PULL REQUEST

API token *

.....

✖

CREATE VARIABLE

GENERATE TOKEN

Pipeline *

CICD-2 ⓘ

Comments

Execution trigger delay ⓘ

1

⌵

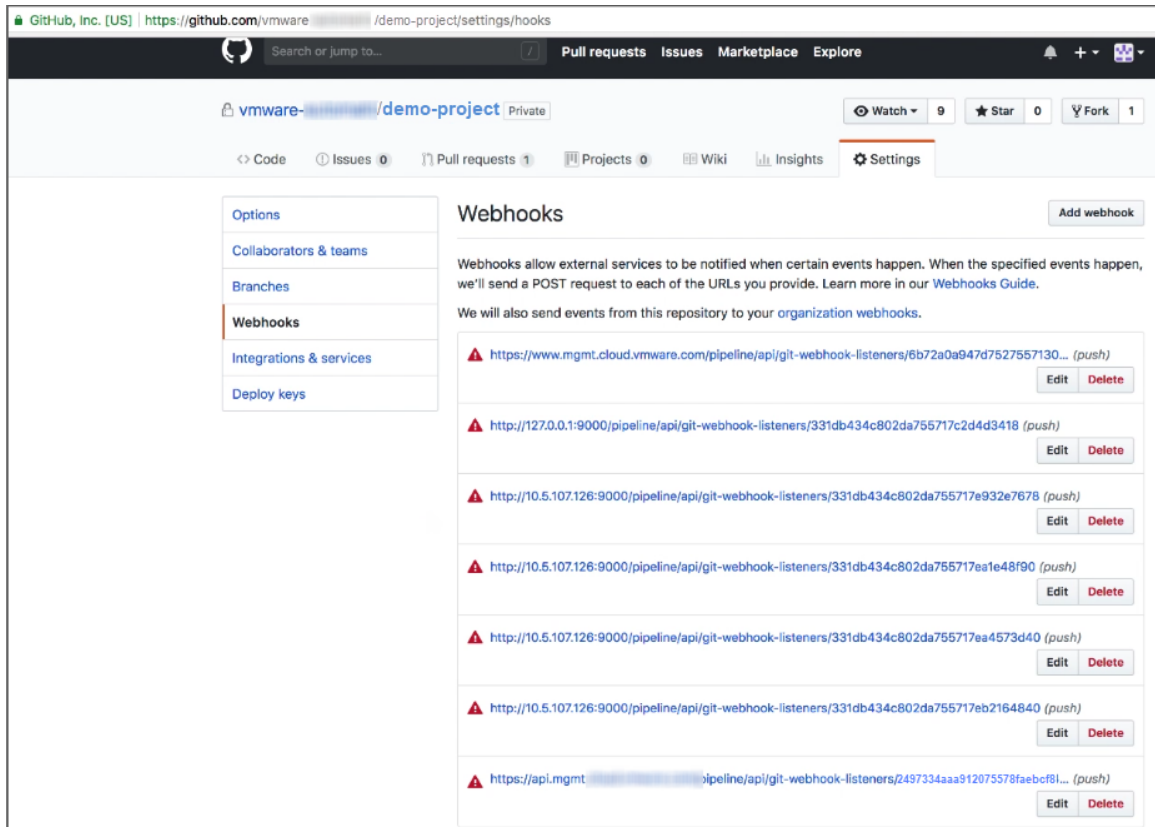
SAVE

CANCEL

- 4 Em uma nova janela do navegador, abra o repositório GitHub que está conectado por meio do webhook.

- a Para ver o webhook adicionado por você no vRealize Automation Code Stream, clique na guia **Configurações** e selecione **Webhooks**.

No final da lista de webhooks, você verá o mesmo URL do webhook.



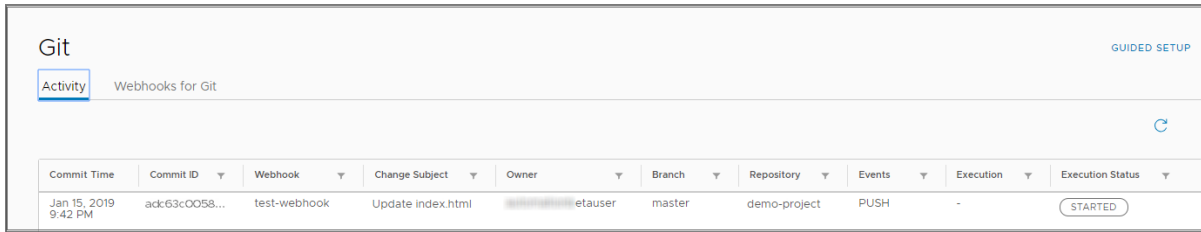
- b Para fazer uma alteração no código, clique na guia **Código** e selecione um arquivo a ser editado na ramificação a ser editada. Confirme a alteração.
 - c Para verificar se o URL do webhook está funcionando, clique na guia **Configurações** e selecione **Webhooks** novamente.

Na parte inferior da lista de webhooks, uma marca de seleção verde aparece ao lado do URL do webhook.



- 5 Volte para o vRealize Automation Code Stream para visualizar a atividade no Git webhook. Clique em **Gatilhos > Git > Atividade**.

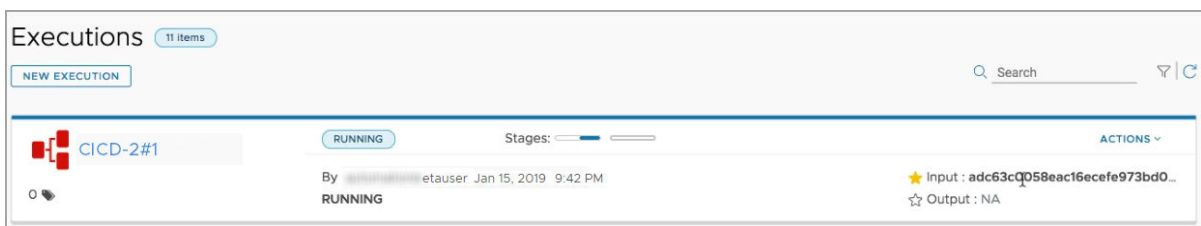
Em Status de Execução, verifique se a execução do pipeline foi iniciada.



Commit Time	Commit ID	Webhook	Change Subject	Owner	Branch	Repository	Events	Execution	Execution Status
Jan 15, 2019 9:42 PM	adc63c0058...	test-webhook	Update index.html	etauser	master	demo-project	PUSH	-	STARTED

- 6 Clique em **Execuções** para rastrear seu pipeline à medida que ele é executado.

É possível pressionar o botão Atualizar para observar a execução do pipeline.



Execution Name	Status	Stages	By	Input	Output
CICD-2#1	RUNNING	Stages: 1/1	By etauser Jan 15, 2019 9:42 PM	Input : adc63c0058ac16ecef973bd0...	Output : NA

Resultados

Parabéns! Você usou com êxito um gatilho Git para executar um pipeline.

Como usar o gatilho Gerrit no vRealize Automation Code Stream para executar um pipeline

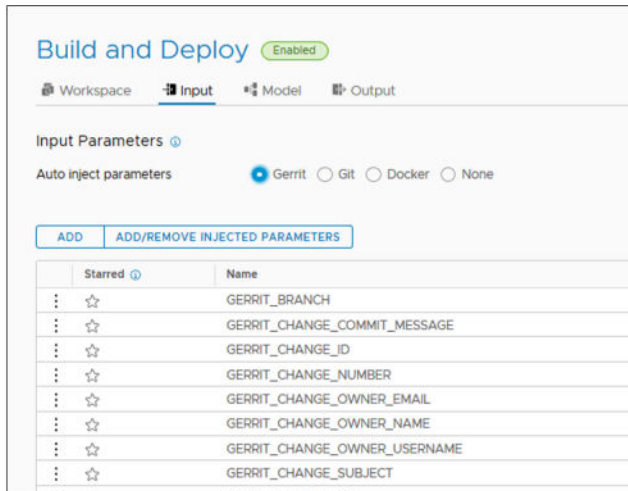
Como administrador ou desenvolvedor do vRealize Automation Code Stream, você pode usar o gatilho Gerrit para integrar o vRealize Automation Code Stream ao ciclo de vida de revisão do código Gerrit. O evento aciona um pipeline a ser executado quando você cria um conjunto de patches, publica rascunhos, mescla alterações de código no projeto Gerrit ou envia alterações diretamente na ramificação do Git.

Ao adicionar o gatilho para Gerrit, selecione um Ouvinte Gerrit, um projeto Gerrit no servidor e configure os eventos Gerrit. Neste exemplo, você configura primeiro um ouvinte Gerrit e, em seguida, usa esse ouvinte em um gatilho do Gerrit com dois eventos em três pipelines diferentes.

Pré-requisitos

- Verifique se você é membro de um projeto no vRealize Automation Code Stream. Se não for, peça a um administrador do vRealize Automation Code Stream para adicioná-lo como membro de um projeto. Consulte [Como adicionar um projeto no vRealize Automation Code Stream](#).
- Verifique se você tem um endpoint do Gerrit configurado no vRealize Automation Code Stream. Consulte [Como integrar o vRealize Automation Code Stream ao Gerrit](#).

- Para que os pipelines sejam acionados, verifique se você definiu as propriedades de entrada para injetar parâmetros Gerrit quando o pipeline for executado.



Para obter informações sobre os parâmetros de entrada, consulte [Planejando uma compilação nativa de CI/CD no vRealize Automation Code Stream](#) antes de adicionar tarefas manualmente.

Procedimentos

- 1 No vRealize Automation Code Stream, clique em **Gatilhos > Gerrit**.
- 2 (Opcional) Clique na guia **Ouvintes** e depois em **Novo Ouvinte**.

Observação Pule essa etapa se o ouvinte Gerrit que você pretende usar para o gatilho do Gerrit já estiver definido.

- a Selecione um projeto.
- b Insira um nome para o ouvinte Gerrit.
- c Selecione um endpoint do Gerrit.

- d Digite o Token de API.

O token de API do CSP autentica você para conexões de API externas com o vRealize Automation Code Stream. Para obter o token de API:

- 1 Clique em **Gerar Token**.
- 2 Insira o endereço de e-mail associado ao seu nome de usuário e senha e clique em **Gerar**.

O token gerado será válido por seis meses. Ele também é conhecido como token de atualização.

- Para manter o token como uma variável para uso futuro, clique em **Criar Variável**, insira um nome para a variável e clique em **Salvar**.
- Para manter o token como um valor de texto para uso futuro, clique em **Copiar** e cole o token em um arquivo de texto para salvar localmente.

Você pode optar por criar uma variável e armazenar o token em um arquivo de texto para uso futuro.

- 3 Clique em **Fechar**.

Se você tiver criado uma variável, o token de API exibirá o nome da variável que você inseriu. Se você tiver copiado o token, o token de API exibirá o token mascarado.

The screenshot shows the 'Gerrit' configuration window with the 'Listeners' tab selected. The form contains the following fields and controls:

- Project:** A text field containing 'test1' with a clear icon (X) to its right.
- Name:** A text field containing 'Gerrit-Demo-Listener'.
- Endpoint:** A dropdown menu showing 'corporate-gerrit'.
- API token:** A text field containing the variable reference '\${var.CSuser API Token}'. To the right of this field is a green checkmark icon.
- Buttons:** At the bottom left are 'CREATE', 'VALIDATE', and 'CANCEL' buttons. To the right of the API token field are 'CREATE VARIABLE' and 'GENERATE TOKEN' buttons.

- e Para validar os detalhes do token e do endpoint, clique em **Validar**.
Seu token expira após 90 dias.

- f Clique em **Criar**.
- g No cartão do ouvinte, clique em **Conectar**.

O ouvinte começa a monitorar todas as atividades no servidor Gerrit e escuta todos os gatilhos habilitados nesse servidor. Para parar de escutar um gatilho nesse servidor. Desative o gatilho.

Observação Para atualizar um endpoint do Gerrit que está conectado a um ouvinte, você deve desconectar o ouvinte antes de atualizar o endpoint.

- Clique em **Configurar > Gatilhos > Gerrit**.
 - Clique na guia **Ouvintes**.
 - Clique em **Desconectar** no ouvinte que está conectado ao endpoint que você deseja atualizar.
-

- 3 Clique na guia **Gatilhos**, depois clique em **Novo Gatilho Gerrit**.

- 4 Selecione um projeto no servidor Gerrit.

- 5 Digite um nome.

O nome do gatilho Gerrit deve ser exclusivo.

- 6 Selecione um ouvinte Gerrit configurado.

O vRealize Automation Code Stream usa a seleção do ouvinte Gerrit para fornecer uma lista de projetos do Gerrit que estão disponíveis no servidor.

- 7 Selecione um projeto no servidor Gerrit.

- 8 Insira a ramificação no repositório que deve ser monitorado.

- 9 (Opcional) Forneça inclusões de arquivo ou exclusões como condições para o gatilho.

- Você fornece inclusões de arquivo para ter pipelines de gatilho. Quando qualquer um dos arquivos em uma confirmação corresponder aos arquivos especificados nos caminhos de inclusão ou no Regex, os pipelines serão disparados. Com um Regex especificado, o vRealize Automation Code Stream apenas disparará os pipelines com nomes de arquivo no conjunto de alterações que corresponderem à expressão fornecida. O filtro Regex é útil ao configurar um gatilho para vários pipelines em um único repositório.
- Você fornece exclusões de arquivos para evitar que os pipelines sejam disparados. Quando todos os arquivos em uma confirmação corresponderem aos arquivos especificados nos caminhos de exclusão ou no Regex, os pipelines não serão disparados.
- Quando alternado, a opção Priorizar Exclusão garante que os pipelines não sejam disparados. Os pipelines não serão acionados, mesmo que qualquer um dos arquivos em uma confirmação corresponder aos arquivos especificados nos caminhos de exclusão ou no Regex. A configuração padrão é desativada.

Se as condições para inclusão e exclusão forem atendidas, os pipelines não serão disparados.

No exemplo a seguir, as inclusões e as exclusões de arquivo são condições para o gatilho.

The screenshot shows a configuration window titled "File" with an information icon. It contains two sections: "Inclusions" and "Exclusions".

Section	Pattern Type	Pattern	Actions
Inclusions	PLAIN	runtime/src/main/a.java	-
	REGEX	([a-z A-Z]+[/][a-z A-Z])+	- +
Exclusions	PLAIN	runtime/pom.xml	-
	PLAIN	runtime/demo.yaml	- +

At the bottom, there is a "Prioritize Exclusion" toggle switch, which is currently turned off.

- Para inclusões de arquivo, uma confirmação com qualquer alteração em `runtime/src/main/a.java` ou qualquer arquivo Java disparará pipelines configurados na configuração de eventos.
- Para exclusões de arquivos, uma confirmação com alterações somente em ambos os arquivos não disparará os pipelines configurados nas configurações de evento.

10 Clique em **Nova Configuração**.

- Para um evento Gerrit, selecione **Conjunto de Patches Criado**, **Rascunho Publicado** ou **Alteração Mesclada**. Ou para um push direto para o Git que ignora o Gerrit, selecione **Push Git direito**.
- Selecione o pipeline a ser disparado.

Se o pipeline tiver sido configurado com os parâmetros de entrada adicionados personalizados, a lista de parâmetros de entrada exibirá parâmetros e valores. É possível inserir valores para os parâmetros de entrada que serão transmitidos ao pipeline com o evento de gatilho. Também é possível deixar os valores em branco ou usar os valores padrão.

Observação Se valores padrão estiverem definidos:

- Qualquer valor inserido para os parâmetros de entrada substituirá os valores padrão definidos no modelo de pipeline.
- Os valores padrão usados para configurar o gatilho não serão atualizados se os valores de parâmetros no modelo de pipeline forem alterados.

Para obter informações sobre a inserção automática de parâmetros de entrada para gatilhos do Gerrit, consulte [Pré-requisitos](#).

- c Para **Conjunto de Patches Criado**, **Rascunho Publicado** e **Alterar Mesclados**, algumas ações aparecem com rótulos por padrão. Você pode alterar o rótulo ou adicionar comentários. Em seguida, quando o pipeline for executado, o rótulo ou comentário aparecerá na guia Atividade como a **Ação realizada** para o pipeline.

- d Clique em **Salvar**.

Para adicionar vários eventos de gatilho em vários pipelines, clique em **Nova Configuração** novamente.

No exemplo a seguir, é possível ver os eventos de três pipelines:

- Se um evento de **Alteração mesclada** ocorrer no projeto Gerrit, então o **Pipeline de Gerrit** será disparado.
- Se um evento de **Conjunto de Patches Criado** ocorrer no projeto Gerrit, então o **Gerrit-Gatilho-Pipeline** e **Gerrit-Demo-Pipeline** serão disparados.

Gerrit GUIDED SETUP

Activity **Triggers** Listeners

Project * test1

Name * Gerrit-Demo-Trigger

Gerrit Listener * Gerrit-Demo-Listener

Gerrit project * Gerrit-Demo-Project

Branch * master

File ⓘ

Inclusions -- Select Type -- value +

Exclusions -- Select Type -- value +

Prioritize Exclusion ☐

+ NEW CONFIGURATION

Event Type	Pipeline	Label
Change Merged	Gerrit-Pipeline	Verified
Patchset Created	Gerrit-Trigger-Pipeline	Verified
Patchset Created	Gerrit-Demo-Pipeline	Verified

3 configurations

- 11 Clique em **Criar**.

O gatilho Gerrit aparece como um novo cartão na guia **Gatilhos** e é definido como **Desativado** por padrão.

12 No cartão de gatilho, clique em **Ativar**.

Quando o gatilho está habilitado, ele usa o ouvinte do Gerrit para iniciar eventos de monitoramento que ocorrem na ramificação do projeto Gerrit.

Ao criar o gatilho, você inclui o repositório em que ocorre a confirmação do código. Se quiser criar um gatilho com as mesmas condições de inclusão ou exclusão de arquivos, mas com um repositório diferente, por exemplo, você poderá clicar em **Ações > Clonar** no cartão de gatilho. Em seguida, clique em **Abrir** no novo gatilho e altere os parâmetros.

Resultados

Parabéns! Você configurou com êxito um gatilho Gerrit com dois eventos em três pipelines diferentes.

Próximo passo

Depois de confirmar uma alteração de código no projeto Gerrit, verifique a guia Atividade do evento Gerrit no vRealize Automation Code Stream. Verifique se a lista de atividades inclui entradas correspondentes a cada execução do pipeline configurada no gatilho. Quando ocorre um evento, são executados somente os pipelines no gatilho Gerrit que estão relacionados ao tipo de evento específico. Neste exemplo, se um conjunto de patches for criado, somente o **Gerrit-Gatilho-Pipeline** e o **Gerrit-Demo-Pipeline** serão executados.

As informações nas colunas da guia Atividade descrevem cada evento do gatilho Gerrit. Você pode escolher as colunas a serem exibidas.

- As colunas Alterar Assunto e Execução estarão vazias se o disparador tiver sido um push Git direto.
- A coluna Gatilho Gerrit mostra o gatilho que criou o evento.
- O Ouvinte está desativado por padrão. Quando selecionado, ele mostra o ouvinte Gerrit que recebeu o evento. Um ouvinte pode ser associado a vários gatilhos.
- O Tipo de Gatilho fica desativado por padrão. Quando selecionado, ele mostra se o gatilho foi acionado de forma manual ou automática.

Gerrit

Activity Triggers Listeners

TRIGGER MANUALLY ⓘ

	Commit Time	Change#	Change Subject	Execution	Status	Message	Action taken	User	Gerrit project
⋮	Nov 12, 2019, 12:47:53 PM	19570 /4	111Dummy	Gerrit-Pipeline #1	COMPLETED	Execution Completed.	Verified +1	Praveen Kumar spnyank@vm	test1
⋮	Nov 12, 2019, 12:50:04 PM	19570 /6	11111Dummy	Gerrit-Pipeline #2	WAITING	Stage0.Task0: Execution Waiting for User Action.		Praveen Kumar spnyank@vm	test1
⋮			11111Dummy	Gerrit-Demo-Pipeline #1	FAILED	Stage0.Task0: User Operation request has been rejected by fritz.	Verified -1	Praveen Kumar spnyank@vm	test1
⋮			11111Dummy	Gerrit-Trigger-Pipeline #1	WAITING	Stage0.Task0: Execution Waiting for User Action.		Praveen Kumar spnyank@vm	test1

Show columns

- ☒ Change#
- ☒ Change Subject
- ☒ Execution
- ☒ Status
- ☒ Message
- ☒ Action taken
- ☒ User
- ☒ Gerrit project
- ☒ Gerrit Trigger
- ☐ Listener
- ☒ Branch
- ☒ Event
- ☐ Trigger Type

SELECT ALL

Para controlar a atividade para uma execução concluída ou que falhou, clique nos três pontos à esquerda de qualquer entrada na tela Atividade.

- Se o pipeline falhar ao ser executado devido a um erro no modelo de pipeline ou outro problema, corrija o erro e selecione **Executar novamente** para executá-lo novamente.
- Se o pipeline falhar ao ser executado devido a um problema de conectividade de rede ou outro problema, selecione **Retomar** para reiniciar a mesma execução do pipeline. Fazer isso economiza tempo de execução.
- Use **Exibir Execução** para transferir para a tela Execução. Consulte [Como executar um pipeline e ver os resultados](#).
- Use **Excluir** para excluir a entrada da tela Atividade.

Se um evento Gerrit falhar em disparar um pipeline, é possível clicar no botão **Disparar Manualmente** e digitar o nome do gatilho Gerrit e o ID de alteração.

Monitorando pipelines no vRealize Automation Code Stream



Como administrador ou desenvolvedor do vRealize Automation Code Stream, você usa os painéis do vRealize Automation Code Stream para monitorar as tendências e os resultados de uma execução de pipeline. É possível usar os painéis de pipeline padrão para monitorar um único pipeline ou criar painéis personalizados para monitorar vários pipelines.

O que são painéis de pipeline

Um painel de pipeline é uma visualização dos resultados de um pipeline específico que foi executado, como tendências, principais falhas e alterações bem-sucedidas. O vRealize Automation Code Stream cria o painel de pipeline quando você cria um pipeline.

O que são painéis personalizados

Um painel personalizado é uma visualização dos resultados de um ou mais pipelines que foram executados. Você cria o painel personalizado e adiciona widgets para exibir os resultados que deseja ver. Por exemplo, é possível criar um painel de todo o projeto com KPIs e métricas coletadas de vários pipelines. Se um aviso ou falha de execução for relatado, você poderá usar o painel para solucionar a falha.

Este capítulo inclui os seguintes tópicos:

- [Como rastrear os principais indicadores de desempenho do pipeline no vRealize Automation Code Stream](#)

Como rastrear os principais indicadores de desempenho do pipeline no vRealize Automation Code Stream

Como administrador ou desenvolvedor do vRealize Automation Code Stream, você precisa de informações sobre o desempenho dos pipelines no vRealize Automation Code Stream. É necessário saber com que eficácia os pipelines liberam o código desde o desenvolvimento, passando por testes, até a produção.

Para obter informações, é possível usar o painel padrão para seu pipeline ou usar um painel personalizado.

- As métricas do pipeline incluem estatísticas, como tempos médios, que estão disponíveis no painel de pipeline.
- Para ver as métricas em vários pipelines, use os painéis personalizados.

Você pode fazer com que o vRealize Automation Code Stream meça os tempos médios para recuperação, entrega ou falha de um pipeline ao longo do tempo e exiba as tendências para esses tempos médios.

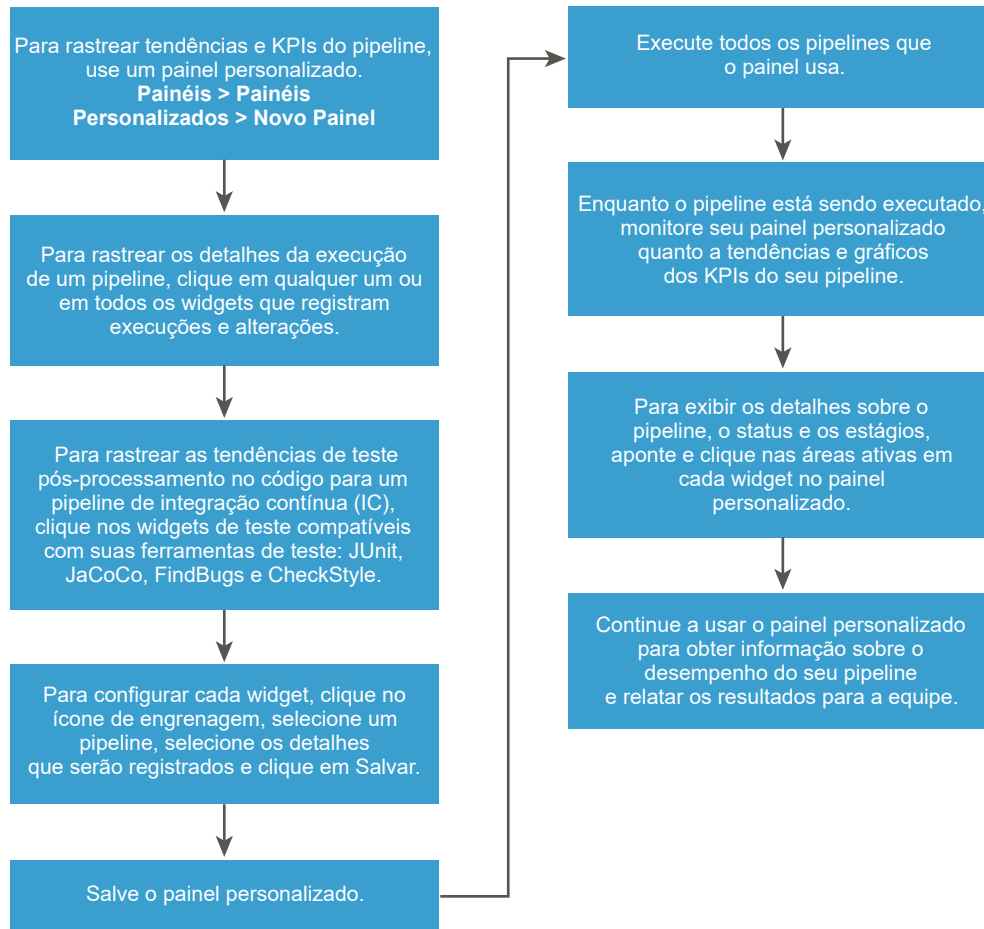
Tabela 8-1. Medição de tempos médios

O que é avaliado...	O que significa...
CI média	Tempo médio gasto na fase de integração contínua, avaliado por tempo no tipo de tarefa de CI.
Tempo médio para entregar um pipeline	Tempo médio exigido pelo pipeline para entregar uma atualização depois do disparo.
Tempo médio entre implantações bem-sucedidas	Tempo entre implantações bem-sucedidas, que indicam a frequência com que um ambiente de produção é atualizado.
Tempo médio de falha de um pipeline	O tempo que um pipeline leva para falhar após ser disparado.
Tempo médio de recuperação de uma falha de pipeline	Tempo médio para a entrega de um pipeline bem-sucedido após uma falha de pipeline. Mede o tempo entre a falha de critérios de compilação ou teste e a compilação seguinte que produz uma execução do pipeline bem-sucedida, calculado por média sobre uma semana ou mês.

Também é possível fazer com que o vRealize Automation Code Stream exiba os principais estágios e tarefas que falharam em um pipeline. Essa medição registra o número e o percentual de falhas para ambientes de desenvolvimento e pós-desenvolvimento para cada pipeline e projeto, calculado por média sobre uma semana ou mês. Visualize as principais falhas para solucionar problemas no processo de automação da liberação.

Por exemplo, é possível configurar a exibição para uma duração específica, como os últimos sete dias, e anotar as principais tarefas que falharam durante esse período. Se você fizer uma alteração em seu ambiente ou pipeline e executar o pipeline novamente, verifique as principais tarefas que falharam em uma duração maior, como nos últimos 14 dias, as principais tarefas com falha podem ter mudado. Com esse resultado, é possível saber que a alteração no processo de automação da liberação melhorou a taxa de êxito da execução do pipeline.

Para rastrear as tendências e os principais indicadores de desempenho dos pipelines usando um painel personalizado, adicione widgets ao painel e configure-os para registrar os pipelines.



Pré-requisitos

- Verifique a existência de um ou mais pipelines. Na interface do usuário, clique em **Pipelines**.
- Verifique se os pipelines que você pretende monitorar foram executados com êxito. Clique em **Execuções**.

Procedimentos

- 1 Para criar um painel personalizado, clique em **Painéis > Painéis Personalizados > Novo Painel**.

- 2 Para personalizar o painel de forma que ele registre tendências específicas e os principais indicadores de desempenho do pipeline, clique em um widget.

Por exemplo, para exibir detalhes sobre o status, estágios e tarefas do pipeline, por quanto tempo ele foi executado e quem o executou, clique no widget **Detalhes da Execução**.

Ou, para um pipeline de integração contínua (CI), é possível rastrear as tendências no pós-processo usando os widgets para JUnit, JaCoCo, FindBugs e CheckStyle.

IX KPIs								
EDIT DELETE CLONE BACK								
Execution Details								
	Execution ID	Execution#	Status	Status Message	Stages	Tasks	Task0 (Stage0)	Duration
⋮	178f62eef...	#2	WAITING	Stage0.Task0 Execution Waiting for User Action.	●	⚠	⚠	15s
⋮	5503c1e51...	#1	COMPLETED	Execution Completed.	●	✓	✓	1h 28m 7s

- 3 Configure cada widget adicionado.
 - a No widget, clique no ícone de engrenagem.
 - b Selecione um pipeline, defina as opções disponíveis e selecione as colunas a serem exibidas.
 - c Para salvar a configuração do widget, clique em **Salvar**.
 - d Para salvar o painel personalizado, clique em **Salvar** e, depois, em **Fechar**.
- 4 Para exibir mais informações sobre o pipeline, clique nas áreas ativas nos widgets.

Por exemplo, no widget **Detalhes da Execução**, clique em uma entrada na coluna de Status para exibir mais informações sobre a execução do pipeline. Ou, no widget **Última Alteração Bem-Sucedida** para exibir um resumo do estágio e da tarefa do pipeline, clique no link ativo.

Resultados

Parabéns! Você criou um painel personalizado que monitora tendências e KPIs dos pipelines.

Próximo passo

Continue a monitorar o desempenho dos pipelines no vRealize Automation Code Stream e compartilhe os resultados com o gerente e as equipes para melhorar o processo de liberação dos aplicativos.

Saiba mais sobre o Code Stream

9

Há muitas maneiras para os administradores e desenvolvedores do vRealize Automation Code Stream aprenderem mais sobre o vRealize Automation Code Stream e o que ele pode fazer por você.

Você pode usar esta documentação para saber mais sobre pipelines e suas execuções, como adicionar endpoints, como adicionar projetos e muito mais.

Entenda as permissões que as funções fornecem. Saiba como usar recursos restritos e exigir aprovações nos pipelines. Consulte [Como gerenciar o acesso do usuário e as aprovações no vRealize Automation Code Stream](#).

Consulte o valor da pesquisa ao descobrir onde os trabalhos ou componentes específicos estão localizados nos pipelines, execuções ou endpoints.

Este capítulo inclui os seguintes tópicos:

- [O que é a pesquisa no vRealize Automation Code Stream](#)
- [Mais recursos para administradores e desenvolvedores do vRealize Automation Code Stream](#)

O que é a pesquisa no vRealize Automation Code Stream

É possível usar a pesquisa para descobrir onde itens específicos ou outros componentes estão localizados. Por exemplo, talvez você queira procurar por pipelines ativados ou desativados. Porque, se um pipeline estiver desativado, ele não poderá ser executado.

O que é possível pesquisar

Você pode pesquisar em:

- Projetos
- Endpoints
- Pipelines
- Execuções
- Painéis de pipeline, painéis personalizados
- Servidores e gatilhos Gerrit

- Git Webhooks
- Webhooks do Docker

Você pode realizar a pesquisa de filtro com base em coluna em:

- Operações do Usuário
- Variáveis
- Atividade de gatilho para Gerrit, Git e Docker

É possível executar a pesquisa de filtro com base em grade na página **Atividade** para cada gatilho.

Como funciona a pesquisa

Os critérios para a pesquisa variam de acordo com a página em que você está. Cada página tem diferentes critérios de pesquisa.

Onde você pesquisa	Crítérios a serem usados para pesquisa
Painéis de Pipeline	Projeto, Nome, Descrição, Tags, Link
Painéis Personalizados	Projeto, Nome, Descrição, Link (UUID de um item no painel)
Execuções	Nome, Comentários, Motivo, Tags, Índice, Status, Projeto, Mostrar, Executado por, Executado por mim, Link (UUID da execução) e Parâmetros de entrada, Parâmetros de saída ou Mensagem de status usando este formato: <key>:<value>
Pipelines	Nome, Descrição, Estado, Tags, Criado por, Criado por mim, Atualizado por, Atualizado por mim, Projeto
Projetos	Nome, Descrição
Endpoints	Nome, Descrição, Tipo, Atualizado por, Projeto
Gatilhos Gerrit	Nome, Status, Projeto
Servidores Gerrit	Nome, URL do Servidor, Projeto
Git Webhooks	Nome, Tipo de Servidor, Repositório, Ramificação, Projeto

Onde:

- O link é o UUID de um pipeline, execução ou widget em um painel.
- A notação e exemplos de parâmetros de entrada, parâmetros de saída e mensagens de status incluem:
 - Notação: `input.<inputKey>:<inputValue>`
Exemplo: `input.GERRIT_CHANGE_OWNER_EMAIL:joe_user`
 - Notação: `output.<outputKey>:<outputValue>`

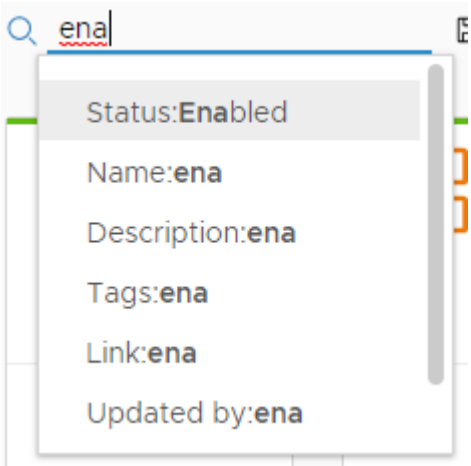
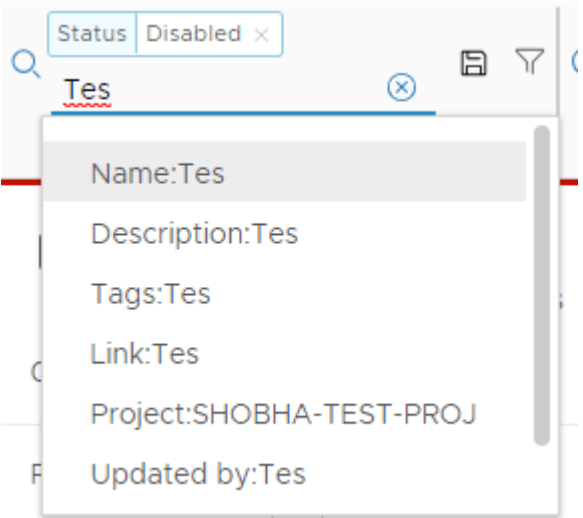
Exemplo: `output.BuildNo:29`

- Notação: `statusMessage:<value>`

Exemplo: `statusMessage:Execution failed`

- O status ou estado depende da página de pesquisa.
 - Para execuções, os valores possíveis incluem: concluído, com falha, reversão_falhou ou cancelado.
 - Para pipelines, os possíveis valores de estado incluem: habilitado, desabilitado ou liberado.
 - Para gatilhos, os possíveis valores de status incluem: habilitado ou desabilitado.
- Executado, Criado ou Atualizado por Mim se referem a mim, o usuário conectado.

A pesquisa aparece no canto superior direito de cada página válida. Quando você começa a digitar na pesquisa em branco, o vRealize Automation Code Stream reconhece o contexto da página e sugere opções para a pesquisa.

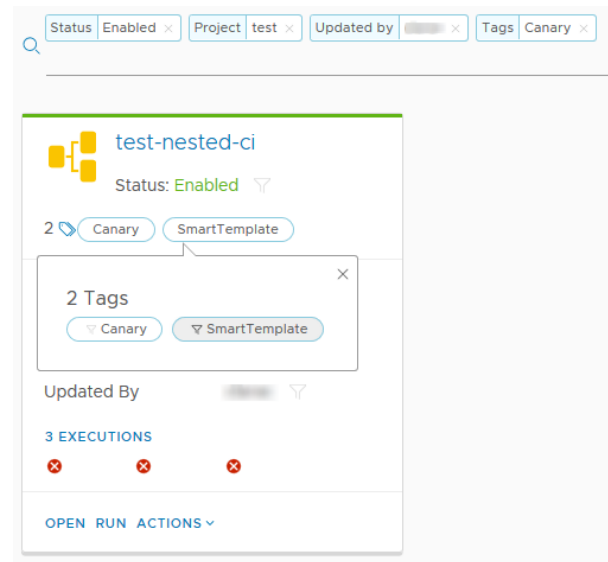
Métodos que podem ser usados para pesquisar	Como digitar
<p>Digite uma parte do parâmetro de pesquisa.</p> <p>Por exemplo, digite ena para adicionar um filtro de status que lista todos os pipelines ativados.</p>	 <p>The screenshot shows a search bar with the text 'ena' entered. A dropdown menu is displayed below the search bar, listing several suggestions: 'Status:Enabled', 'Name:ena', 'Description:ena', 'Tags:ena', 'Link:ena', and 'Updated by:ena'. The 'Status:Enabled' option is highlighted.</p>
<p>Adicione um filtro para reduzir o número de itens encontrados.</p> <p>Por exemplo, digite Tes para adicionar um filtro de nome. O filtro funciona como E com o filtro existente de Status: Desativado para mostrar apenas os pipelines desativados com Tes no nome.</p>	 <p>The screenshot shows a search bar with the text 'Tes' entered. Above the search bar, there is a filter button labeled 'Status: Disabled'. A dropdown menu is displayed below the search bar, listing several suggestions: 'Name:Tes', 'Description:Tes', 'Tags:Tes', 'Link:Tes', 'Project:SHOBHA-TEST-PROJ', and 'Updated by:Tes'. The 'Name:Tes' option is highlighted.</p>

Métodos que podem ser usados para pesquisar

Clique no ícone de filtro nas propriedades de um pipeline ou execução para reduzir o número de itens exibidos.

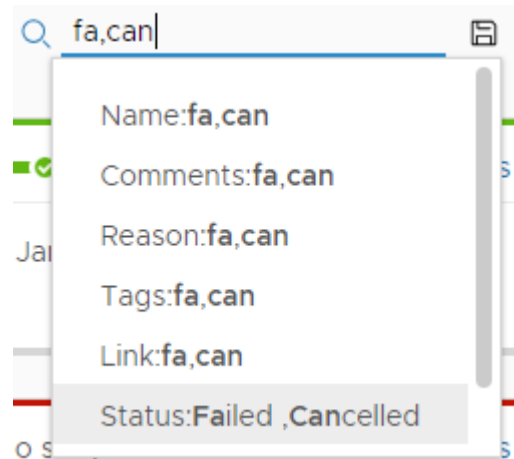
- Para pipelines, **Status**, **Tags**, **Projeto** e **Atualizado por** têm, cada um, um ícone de filtro.
- Para execuções, **Tags**, **Executado por** e **Mensagem de status** têm, cada um, um ícone de filtro.

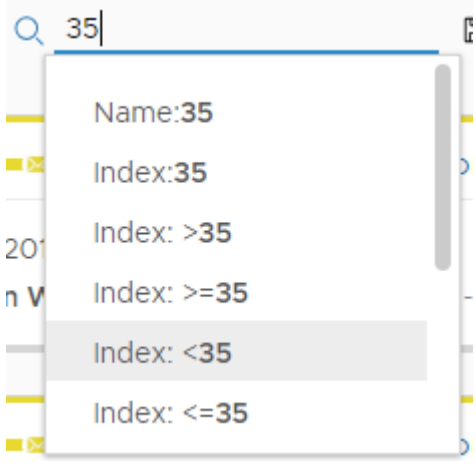
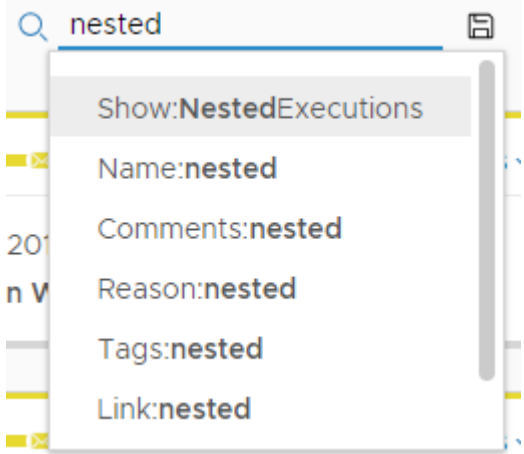
Por exemplo, no cartão de pipeline, clique no ícone para adicionar o filtro para a tag **SmartTemplate** aos filtros existentes para: **Status:Enabled**, **Project:test**, **Updated by:user** e **Tags:Canary**.

Como digitar

Use um separador de vírgula para incluir todos os itens em dois estados de execução.

Por exemplo, digite **fa,can** para criar um filtro de status que funcione como um **OR** para listar todas as execuções que falharam ou cancelamento.

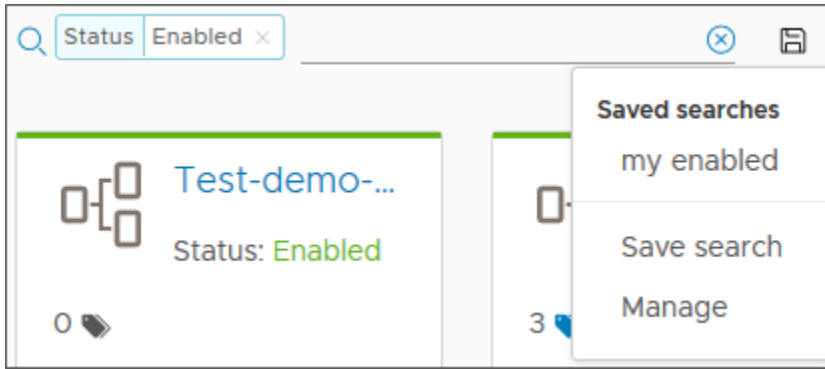


Métodos que podem ser usados para pesquisar	Como digitar
<p>Digite um número para incluir todos os itens em um intervalo de índice.</p> <p>Por exemplo, digite 35 e selecione < para listar todas as execuções com um número de índice inferior a 35.</p>	
<p>Os pipelines são modelados conforme as tarefas se tornam execuções aninhadas e não são listadas com todas as execuções por padrão.</p> <p>Para mostrar execuções aninhadas, digite nested e selecione o filtro Exibir.</p>	

Como salvar uma pesquisa favorita

É possível salvar as pesquisas favoritas para serem usadas em todas as páginas clicando no ícone de disco ao lado da área de pesquisa.

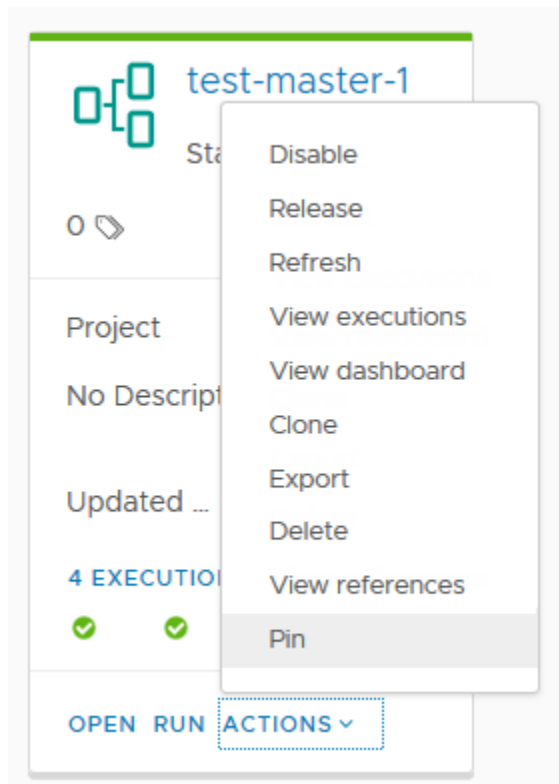
- Salve uma pesquisa digitando os parâmetros para pesquisa e clicando no ícone para dar um nome de pesquisa como **ativado por mim**.
- Depois de salvar uma pesquisa, clique no ícone para acessar a pesquisa. Também é possível selecionar **Gerenciar** para renomear, excluir ou mover a pesquisa na lista de pesquisas salvas.



As pesquisas são ligadas ao seu nome do usuário e aparecem apenas nas páginas às quais a pesquisa se aplica. Por exemplo, se você tiver salvo uma pesquisa chamada **ativado por mim** para **Status: ativado** na página Pipelines, a pesquisa **ativado por mim** não está disponível na página de gatilhos Gerrit, mesmo que o **Status: ativado** seja uma pesquisa válida para um gatilho.

Posso salvar um pipeline favorito?

Se você tiver um pipeline ou um painel favorito, poderá fixá-los para que sempre apareçam na parte superior da sua página de pipelines ou painéis. No cartão de pipeline, clique em **Ações > Fixar**.



Mais recursos para administradores e desenvolvedores do vRealize Automation Code Stream

Como administrador ou desenvolvedor do vRealize Automation Code Stream, você pode saber mais sobre o vRealize Automation Code Stream.

Tabela 9-1. Mais recursos para administradores

Para saber mais sobre...	Consulte estes recursos...
<p>Outras maneiras em que os administradores podem usar o vRealize Automation Code Stream:</p> <ul style="list-style-type: none"> ■ Configurar os pipelines para automatizar o teste e a liberação dos aplicativos nativos da nuvem. ■ Automatizar e testar o código-fonte do desenvolvedor, por meio de testes até a produção. ■ Configurar os pipelines para os desenvolvedores testarem as alterações antes que eles as confirmem para a ramificação primária. ■ Rastrear as principais métricas de pipeline. 	<p>vRealize Automation Code Stream</p> <ul style="list-style-type: none"> ■ Documentação do vRealize Automation ■ Site do produto vRealize Automation <p>VMware Hands On</p> <ul style="list-style-type: none"> ■ Use a Comunidade do vRealize Automation. ■ Use a Zona de Aprendizagem do VMware. ■ Pesquise nos Blogs de VMware. ■ Experimente os Laboratórios do VMware Hands On.

Tabela 9-2. Mais recursos para desenvolvedores

Para saber mais sobre...	Consulte estes recursos...
<p>Outras maneiras em que os desenvolvedores podem usar o vRealize Automation Code Stream:</p> <ul style="list-style-type: none"> ■ Usar imagens de registro públicas e privadas para compilar ambientes para novos aplicativos ou serviços. ■ Definir ambientes de desenvolvimento para que você possa criar ramificações a partir da compilação estável mais recente. ■ Atualizar os ambientes de desenvolvimento com as alterações mais recentes de código e artefatos. ■ Testar as alterações de código não confirmadas nas compilações estáveis mais recentes de outros serviços dependentes. ■ Receber uma notificação quando uma alteração confirmada em um pipeline de ICEC primário interromper outros serviços. 	<p>vRealize Automation Code Stream</p> <ul style="list-style-type: none"> ■ Documentação do vRealize Automation ■ Site do produto vRealize Automation <p>VMware Hands On</p> <ul style="list-style-type: none"> ■ Use a Comunidade do vRealize Automation. ■ Use a Zona de Aprendizagem do VMware. ■ Pesquise nos Blogs de VMware. ■ Experimente os Laboratórios do VMware Hands On.