

Usar y administrar vRealize Automation Code Stream

14 de diciembre de 2022
vRealize Automation 8.1

Puede encontrar la documentación técnica más actualizada en el sitio web de VMware:

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

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

VMware Spain, S.L.
Calle Rafael Boti 26
2.ª planta
Madrid 28023
Tel.: +34 914125000
www.vmware.com/es

Copyright © 2022 VMware, Inc. Todos los derechos reservados. [Información sobre el copyright y la marca comercial.](#)

Contenido

- 1 Qué es vRealize Automation Code Stream y cómo funciona 5**
- 2 Configuración para modelar el proceso de lanzamiento 10**
 - Cómo agregar un proyecto 14
 - Cómo se administran las autorizaciones y el acceso de los usuarios 15
 - ¿Qué son las aprobaciones y las operaciones de usuario? 19
- 3 Crear y usar canalizaciones 22**
 - Cómo ejecutar una canalización y ver los resultados 25
 - Qué tipos de tareas se encuentran disponibles 29
 - Cómo utilizar las variables de enlace en canalizaciones 34
 - Cómo utilizar enlaces de variables en una tarea de condición para ejecutar o detener una canalización 44
 - Qué variables y expresiones se pueden utilizar al enlazar tareas de canalización 46
 - Cómo enviar notificaciones acerca de la canalización 60
 - Cómo crear un ticket de JIRA cuando se produce un error en una tarea de canalización 62
 - Cómo revertir una implementación 65
- 4 Planificar la compilación, la integración y la distribución nativa del código 71**
 - Planificar una compilación nativa de CICD antes de usar la plantilla inteligente 71
 - Planificar una compilación nativa de CI antes de usar la plantilla inteligente 76
 - Planificar una compilación nativa de CD antes de usar la plantilla inteligente 77
 - Planificación de una compilación nativa de CICD antes de agregar tareas manualmente 78
 - Planificación de una reversión 82
- 5 Casos prácticos comunes 86**
 - Cómo integrar de forma continua código de un repositorio de GitLab o GitHub en la canalización 87
 - Cómo automatizar la publicación de una aplicación implementada a partir de un blueprint de YAML 93
 - Cómo automatizar el lanzamiento de una aplicación en un clúster de Kubernetes 99
 - Cómo implementar la aplicación en una implementación azul-verde 107
 - Cómo se integran las herramientas personalizadas de compilación, pruebas e implementación 112
 - Cómo usar una API de REST para integrar con otras aplicaciones 122
- 6 Conectarse a endpoints 127**
 - Qué son los endpoints 127
 - Cómo se integra con Jenkins 129

[Cómo se puede integrar con Git](#) 134

[Cómo se integra con Gerrit](#) 136

[Cómo se integra con vRealize Orchestrator](#) 138

7 Activar canalizaciones 143

[Cómo usar el activador de Docker para ejecutar una canalización de entrega continua](#) 143

[Cómo usar el activador de Git para ejecutar una canalización](#) 150

[Cómo usar el activador de Gerrit para ejecutar una canalización](#) 157

8 Supervisar canalizaciones 165

[Cómo realizar un seguimiento de los indicadores clave de rendimiento de la canalización](#) 165

9 Más información 169

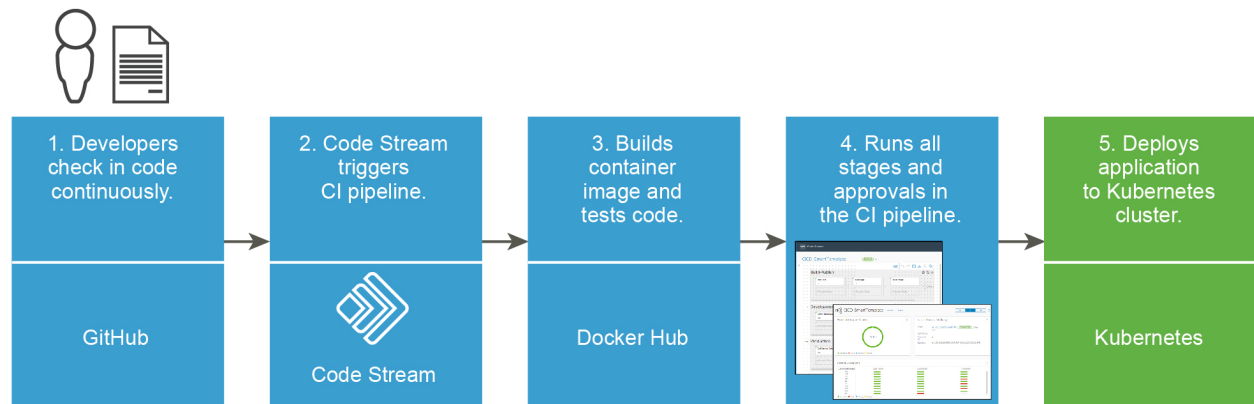
[¿Qué es la función de búsqueda?](#) 169

[Más recursos para desarrolladores y administradores](#) 174

Qué es vRealize Automation Code Stream y cómo funciona

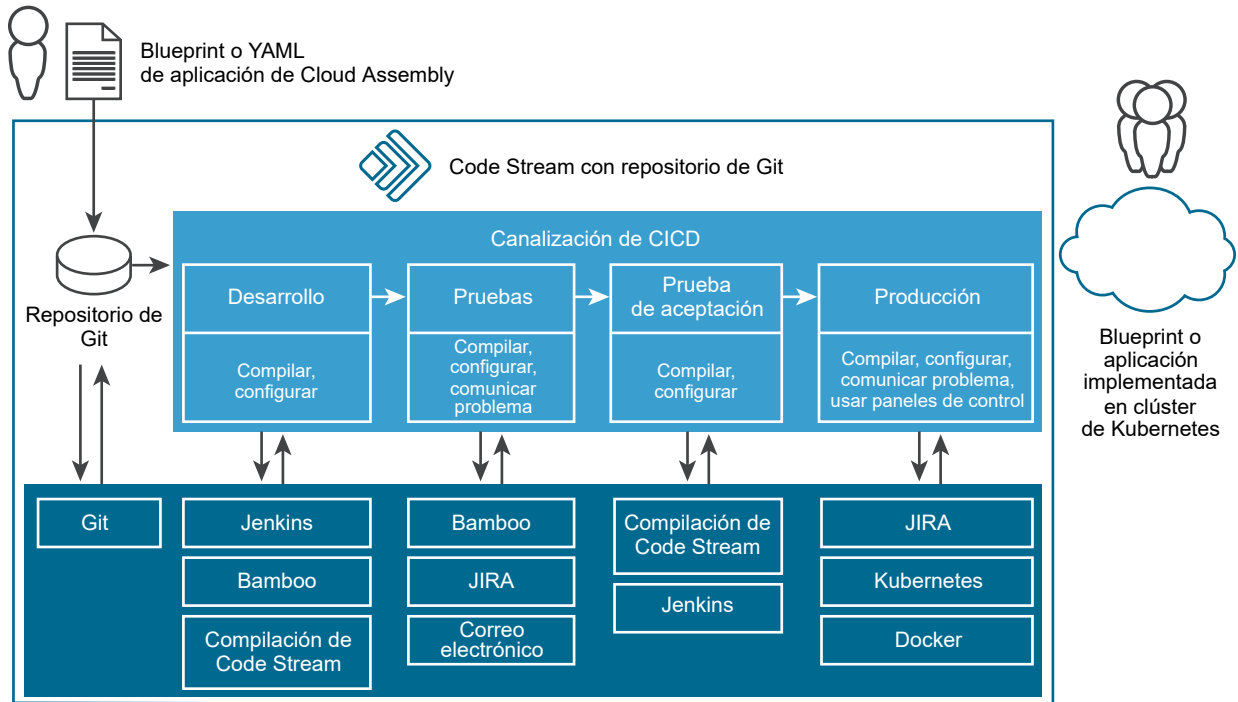
1

vRealize Automation Code Stream™ es una herramienta de integración continua y distribución continua (Continuous Integration and Continuous Delivery, CICD) que se utiliza para crear canalizaciones que modelan el proceso de publicación de software en el ciclo de vida de DevOps. Al crear canalizaciones, se genera la infraestructura del código que distribuye el software de forma rápida y continua.



Cuando utiliza vRealize Automation Code Stream para distribuir el software, integra dos de las partes más importantes del ciclo de vida de DevOps: el proceso de publicación y las herramientas de desarrollador. Después de la configuración inicial, que integra vRealize Automation Code Stream con las herramientas de desarrollo existentes, las canalizaciones automatizan el ciclo de vida completo de DevOps.

Cree una canalización que compile, pruebe y publique el software. vRealize Automation Code Stream utiliza esa canalización para que el software pase del repositorio de código fuente hasta producción, pasando por las pruebas.



Puede obtener más información sobre cómo planificar la integración continua y las canalizaciones de entrega continua en [Capítulo 4 Planificar la compilación, la integración y la distribución nativas del código en vRealize Automation Code Stream](#).

Cómo usan vRealize Automation Code Stream los administradores de DevOps

Como administrador de DevOps, debe crear endpoints y asegurarse de que hay instancias operativas disponibles para los desarrolladores. Puede crear, activar y administrar canalizaciones, entre otras cosas. Tiene la función `Administrator`, como se describe en [Cómo se administran las autorizaciones y el acceso de los usuarios en vRealize Automation Code Stream](#).

Tabla 1-1. Administradores de DevOps que usan vRealize Automation Code Stream

Para colaborar con los desarrolladores...	Puede hacer lo siguiente...
Proporcione y administre entornos.	<p>Cree entornos para que los desarrolladores prueben e implementen el código.</p> <ul style="list-style-type: none"> ■ Realice un seguimiento del estado y envíe notificaciones por correo electrónico. ■ Garantice que los entornos funcionen continuamente para mantener la productividad de los desarrolladores. <p>Para obtener más información, consulte Más recursos para desarrolladores y administradores de vRealize Automation Code Stream.</p> <p>Consulte también Capítulo 5 Ejemplos de casos prácticos comunes para vRealize Automation Code Stream.</p>
Proporcione endpoints.	Asegúrese de que los desarrolladores tengan instancias operativas de endpoints que puedan conectarse con las canalizaciones.

Tabla 1-1. Administradores de DevOps que usan vRealize Automation Code Stream (continuación)

Para colaborar con los desarrolladores...	Puede hacer lo siguiente...
Proporcione integraciones con otros servicios.	Asegúrese de que las integraciones con otros servicios funcionen. Para obtener más información, consulte la documentación de vRealize Automation .
Cree canalizaciones.	Cree canalizaciones que modelen los procesos de publicación. Para obtener más información, consulte Capítulo 3 Crear y usar canalizaciones en vRealize Automation Code Stream .
Active canalizaciones.	Asegúrese de que las canalizaciones se ejecuten cuando se produzcan los eventos. <ul style="list-style-type: none"> ■ Para activar una canalización de entrega continua (Continuous Delivery, CD) independiente cada vez que se cree o se actualice un artefacto de compilación, use el activador de Docker. ■ Para activar una canalización cuando un desarrollador confirma cambios en el código, utilice el activador de Git. ■ Para activar una canalización cuando los desarrolladores revisen código, combinaciones, etc., utilice el activador de Gerrit. ■ Para activar una canalización de entrega continua (Continuous Delivery, CD) independiente cada vez que se cree o se actualice un artefacto de compilación, use el activador de Docker. Para obtener más información, consulte Capítulo 7 Activar canalizaciones en vRealize Automation Code Stream .
Administre canalizaciones y aprobaciones.	Manténgase actualizado sobre las canalizaciones. <ul style="list-style-type: none"> ■ Vea el estado de las canalizaciones y quién las ejecutó. ■ Vea las aprobaciones de ejecuciones de canalizaciones y administre las aprobaciones para ejecuciones de canalizaciones activas e inactivas. Para obtener más información, consulte ¿Qué son las aprobaciones y las operaciones de usuario en vRealize Automation Code Stream? . Consulte también Cómo realizar un seguimiento de los indicadores clave de rendimiento de la canalización en vRealize Automation Code Stream .

Tabla 1-1. Administradores de DevOps que usan vRealize Automation Code Stream (continuación)

Para colaborar con los desarrolladores...	Puede hacer lo siguiente...
Supervise los entornos de desarrollo.	<p>Cree paneles de control personalizados que supervisen el estado de la canalización, las tendencias, las métricas y los indicadores clave. Utilice los paneles de control personalizados para supervisar las canalizaciones que superan o no los requisitos de los entornos de desarrollo. También puede identificar recursos infrutilizados e informar sobre ellos, así como liberar recursos.</p> <p>También puede ver lo siguiente:</p> <ul style="list-style-type: none"> ■ Tiempo que una canalización se ejecutó antes de completarse correctamente. ■ Tiempo que una canalización esperó una autorización y tardó en notificarla al usuario que debe autorizarla. ■ Etapas y tareas en las que se producen errores con mayor frecuencia. ■ Etapas y tareas que tardan más tiempo en ejecutarse. ■ Versiones en las que están trabajando los equipos de desarrollo. ■ Aplicaciones que se implementaron y publicaron correctamente. <p>Para obtener más información, consulte Capítulo 8 Supervisar canalizaciones en vRealize Automation Code Stream.</p>
Solucione problemas.	<p>Solucione problemas y resuelva errores de canalización en entornos de desarrollo.</p> <ul style="list-style-type: none"> ■ Identifique y solucione problemas en los entornos de integración continua y entrega continua (Continuous Integration and Continuous Delivery, CICD). ■ Utilice los paneles de control de canalizaciones y cree paneles de control personalizados para ver más. Consulte Capítulo 8 Supervisar canalizaciones en vRealize Automation Code Stream. <p>Consulte también Capítulo 2 Configuración de vRealize Automation Code Stream para modelar el proceso de lanzamiento.</p>

vRealize Automation Code Stream forma parte de vRealize Automation. vRealize Automation Code Stream se integra con:

- Use vRealize Automation Cloud Assembly para implementar blueprints.
- Use vRealize Automation Service Broker para obtener blueprints y plantillas del catálogo.

Para realizar otras acciones, consulte la [documentación de VMware vRealize Automation](#).

Cómo utilizan vRealize Automation Code Stream los desarrolladores

Como desarrollador, puede usar vRealize Automation Code Stream para crear y ejecutar canalizaciones, así como para supervisar la actividad de las canalizaciones en los paneles de control. Tiene la función `user`, como se describe en [Cómo se administran las autorizaciones y el acceso de los usuarios en vRealize Automation Code Stream](#).

Después de ejecutar una canalización, se recomienda que obtenga la siguiente información:

- ¿El código se completó correctamente en todas las etapas de la canalización? Observe los resultados en **Ejecuciones**.

- ¿Qué hago si se produjo un error en la canalización? ¿Cuál es la causa? En **Paneles de control**, puede ver los principales errores ocurridos.

Tabla 1-2. Desarrolladores que usan vRealize Automation Code Stream

Para integrar y publicar el código...	Debe hacer lo siguiente...
Genere canalizaciones.	Pruebe e implemente el código. Actualice el código cuando se produzca un error en una canalización.
Conecte la canalización a endpoints.	Conecte las tareas de la canalización a endpoints, como un repositorio de GitHub.
Ejecute las canalizaciones.	Agregue una tarea de autorización de operaciones de usuario para que otro usuario pueda autorizar la canalización en puntos específicos.
Consultar paneles de control.	Vea los resultados en el panel de control de la canalización. Puede ver las tendencias, el historial, los errores y mucho más.

Para obtener más información de introducción, consulte [Introducción a VMware Code Stream](#).

Buscar más documentación en el panel de soporte técnico en el producto

Si no encuentra la información que necesita aquí, puede obtener más ayuda en el producto. 

- Haga clic y lea los mensajes informativos y la información sobre herramientas en la interfaz de usuario para obtener la información específica del contexto que necesite en el momento y el lugar indicados.
- Abra el panel de soporte del producto y lea los temas que aparecen en la página de la interfaz de usuario activa. También puede buscar en el panel para obtener respuestas a las preguntas.

Configuración de vRealize Automation Code Stream para modelar el proceso de lanzamiento

2

Para modelar el proceso de lanzamiento, hay que crear una canalización que represente las etapas, las tareas y las aprobaciones que se suelen utilizar para lanzar el software. Tras ello, vRealize Automation Code Stream automatiza el proceso que compila, comprueba, aprueba e implementa el código.

Ahora que tiene todo lo que necesita para modelar el proceso de lanzamiento de software, así es como se lleva a cabo en vRealize Automation Code Stream.

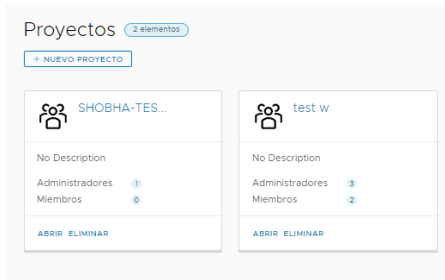
Requisitos previos

- Compruebe si ya hay disponible algún endpoint. En vRealize Automation Code Stream, haga clic en **Endpoints**.
- Obtenga información sobre las diversas formas nativas de compilar e implementar código. Consulte [Capítulo 4 Planificar la compilación, la integración y la distribución nativas del código en vRealize Automation Code Stream](#).
- Decida si algunos de los recursos que va a utilizar en la canalización deben estar marcados como restringidos. Consulte [Cómo se administran las autorizaciones y el acceso de los usuarios en vRealize Automation Code Stream](#).
- Si posee la función de usuario o de visor en lugar de la de administrador, decida quién va a ser el administrador de la instancia de vRealize Automation Code Stream.

Procedimiento

- 1 Examine los proyectos disponibles en vRealize Automation Code Stream y seleccione uno que se ajuste a sus necesidades.
 - Si no aparece ningún proyecto, solicite a un administrador de vRealize Automation Code Stream que cree un proyecto y lo convierta en miembro del proyecto. Consulte [Cómo agregar un proyecto en vRealize Automation Code Stream](#).

- Si no es miembro de ningún proyecto de la lista, solicite a un administrador de vRealize Automation Code Stream que lo agregue como miembro de un proyecto.



- 2 Agregue los nuevos endpoints que necesite para la canalización.

Por ejemplo, puede que necesite Git, Jenkins, Code Stream Build, Kubernetes y Jira.

- 3 Cree variables para poder reutilizar los valores en las tareas de la canalización.

Utilice variables restringidas para restringir los recursos utilizados en las canalizaciones, como una máquina host. Puede restringir que la canalización continúe ejecutándose hasta que otro usuario la apruebe explícitamente.

Los administradores pueden crear variables secretas y restringidas. Los usuarios pueden crear variables secretas.

Puede reutilizar una variable tantas veces como desee en varias canalizaciones. Por ejemplo, una variable que define una máquina host podría definirse como `HostIPAddress`.

A continuación, para utilizar la variable en una tarea de canalización, introduzca `${var.HostIPAddress}`.

Variables ⓘ 18 elementos

+ NUEVA VARIABLE

	Proyecto	Nombre	▼	Tipo	▼	Valor
⋮	0709-AWS-w2 観家表があA 中庄@観停@選UBau^n	2000000000000		SECRET		*****
⋮	0709-AWS-w2 観家表があA 中庄@観停@選UBau^n	20		RESTRICTED		*****
⋮	0709-AWS-w2 観家表があA 中庄@観停@選UBau^n	2		SECRET		*****

- 4 Si es administrador, marque como recursos restringidos los endpoints y las variables que sean fundamentales para su empresa.

Cuando un usuario que no es administrador intenta ejecutar una canalización que incluye un recurso restringido, la canalización se detiene en la tarea que utiliza el recurso restringido. A continuación, un administrador debe reanudar la canalización.

- 5 Trace la estrategia de compilación de su canalización nativa de CI/CD, de integración continua o de entrega continua.

Antes de crear una canalización que integre e implemente código de forma continua, planifique su estrategia de compilación. El plan de compilación le ayuda a determinar qué necesita vRealize Automation Code Stream para que pueda compilar, integrar, probar e implementar el código de forma nativa.

El modo en que se cree una compilación nativa de vRealize Automation Code Stream...

Dará como resultado esta estrategia de compilación...

Usando una plantilla inteligente

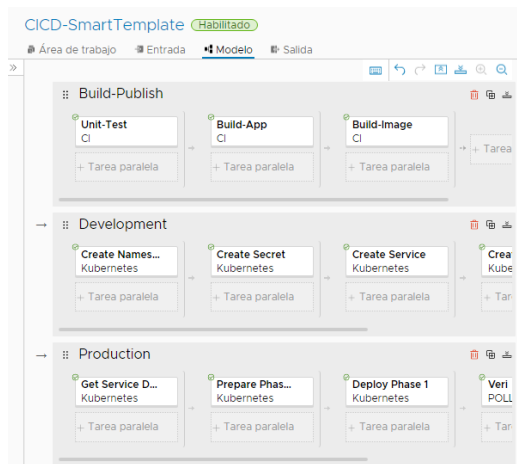
- Se crean automáticamente todas las etapas y las tareas.
- Se clona el repositorio de origen.
- Se compila y se comprueba el código.
- El código se inserta en un contenedor para implementarse.
- Se rellenan los pasos de tareas de canalización en función de las selecciones realizadas.

Agregando etapas y tareas manualmente

Se agregan etapas y tareas, y se introduce la información para rellenarlas.

- 6 Cree la canalización mediante una plantilla inteligente o agregando manualmente fases y tareas a la canalización.

Luego, marque los recursos como restringidos. Agregue las aprobaciones que considere necesarias. Aplique las variables normales, restringidas o secretas que quiera. Establezca enlaces entre las tareas.



- 7 Valide, habilite y ejecute la canalización.

8 Ve a las ejecuciones de la canalización.

Ejecuciones 468 elementos CONFIGURACIÓN GUIADA

[+ NUEVA EJECUCIÓN](#)

Icono	Nombre	ID	Estado	Etapas	Acciones
	Demo-Jenkins-K...	#93	COMPLETED		ACCIONES
Por sestervil el 19 ene. 2020 16:17:14 Execution Completed.					
Entrada: 6d82d079a8b8921a... Salida: -					
	Demo-Jenkins-K...	#13	COMPLETED		ACCIONES
Por sestervil el 19 ene. 2020 16:14:35 Execution Completed.					
Entrada: 6d82d079a8b8921a9 Salida: -					
	Demo-Jenkins-K...	#92	COMPLETED		ACCIONES
Por sestervil el 19 ene. 2020 16:11:06 Execution Completed.					
Entrada: 8b3a29df... Salida: -					
	Demo-CICD-Simp#48	#48	FAILED		ACCIONES
Por sestervil el 19 ene. 2020 16:09:20 Production.Deploy Phase 1: Failed script execution: Failed to execu...					
Entrada: - Salida: -					

9 Para realizar un seguimiento del estado y los KPI, use los paneles de control de canalizaciones y cree los paneles de control personalizados que necesite.

CICD-SmartTemplate [ACCIONES](#)

Última ejecución correcta [↻](#)

Demo-Jenkins-K8s #110 **COMPLETED**
hace 8 minutos

Ejecutado por pmartini	Duración 25 segundos
Entrada: 8b3a2... 8b3a29df...	Salida: -

Recuento de est... Últimos 7 días [↻](#)

Total: 107

COMPLETED FAILED WAITING

Estadísticas de ... Últimos 14 días [↻](#)

MTTD ⓘ 1 minuto	MTTF ⓘ 3 días
MTBD ⓘ 47 minutos	MTTR ⓘ 15 minutos

Ejecuciones recientes [↻](#)

Ejecución #/Etapas	Dev
#110	
#109	
#107	
#104	
#103	
#102	
#101	
#100	
#99	
#98	

COMPLETED FAILED RUNNING WAITING CANCELED ROLLBACK_COMPLETED ROLLBACK_FAILED

Resultados

Creó una canalización que puede utilizar en el proyecto seleccionado.

También puede exportar el YAML de la canalización para importarlo y reutilizarlo en otros proyectos.

Pasos siguientes

Obtenga información sobre algunos casos prácticos que puede que le convenga aplicar a su entorno. Consulte [Capítulo 5 Ejemplos de casos prácticos comunes para vRealize Automation Code Stream](#).

Cómo agregar un proyecto en vRealize Automation Code Stream

Es necesario crear un proyecto al que se puedan agregar administradores y miembros para que los miembros del proyecto puedan utilizar funciones, como crear una canalización y agregar un endpoint. Para crear, eliminar o actualizar un proyecto para un equipo de desarrollo, debe ser administrador de vRealize Automation Code Stream.

Debe existir un proyecto para poder crear una canalización. Al crear una canalización, seleccione el proyecto al que desea asociarla para que toda la información de la canalización esté agrupada. Las definiciones de variables y endpoints también dependen de un proyecto existente.

Requisitos previos

- Compruebe que tiene la función de administrador de vRealize Automation Code Stream. Consulte [Qué son las funciones en vRealize Automation Code Stream](#).

Si no tiene la función de administrador de vRealize Automation Code Stream, pero es administrador de vRealize Automation Cloud Assembly, puede utilizar la interfaz de usuario de vRealize Automation Cloud Assembly para crear, actualizar o eliminar proyectos. Consulte [Cómo agrego un proyecto para el equipo de desarrollo de vRealize Automation Cloud Assembly](#)

- Si planea agregar grupos de Active Directory a los proyectos, compruebe que configuró grupos de Active Directory para su organización. Consulte [Cómo habilitar grupos de Active Directory en vRealize Automation](#). Si los grupos no están sincronizados, no estarán disponibles al intentar agregarlos a un proyecto.

Procedimiento

- 1 Seleccione **Proyectos** y haga clic en **Nuevo proyecto**.
- 2 Introduzca el nombre del proyecto.
- 3 Haga clic en **Crear**.
- 4 Seleccione la tarjeta del proyecto creado recientemente y haga clic en **Abrir**.
- 5 Haga clic en la pestaña **Usuarios** y agregue usuarios con funciones asignadas.
 - El administrador del proyecto puede agregar miembros.
 - El miembro del proyecto que tiene una función de servicio puede utilizar los servicios.
 - El visor de proyectos puede ver los proyectos, pero no puede crearlos, actualizarlos ni eliminarlos.

Para obtener más información acerca de las funciones del proyecto, consulte [Cómo se administran las autorizaciones y el acceso de los usuarios en vRealize Automation Code Stream](#).

- 6 Haga clic en **Guardar**.

Pasos siguientes

Agregue endpoints y canalizaciones que utilicen el proyecto. Consulte [Capítulo 6 Conectar vRealize Automation Code Stream a endpoints](#) y [Capítulo 3 Crear y usar canalizaciones en vRealize Automation Code Stream](#).

Cómo se administran las autorizaciones y el acceso de los usuarios en vRealize Automation Code Stream

vRealize Automation Code Stream ofrece varias maneras de garantizar que los usuarios tengan la autorización y el consentimiento adecuados para trabajar con las canalizaciones que publican las aplicaciones de software.

Cada miembro de un equipo tiene una función asignada, la cual otorga permisos específicos en las canalizaciones, los endpoints y los paneles de control, así como la posibilidad de marcar recursos como restringidos.

Las operaciones de usuario y las autorizaciones permiten controlar cuándo se ejecuta una canalización y cuándo debe detenerse para su autorización. La función determina si puede reanudar una canalización y ejecutar canalizaciones que incluyan variables o endpoints restringidos.

Utilice variables secretas para ocultar y cifrar información confidencial. Utilice variables restringidas para las cadenas, las contraseñas y las URL que deben estar ocultas y cifradas, así como para restringir el uso en ejecuciones. Por ejemplo, utilice una variable secreta para una contraseña o una URL. Puede usar variables de tipo secreto y restringido en cualquier tipo de tarea de la canalización.

Qué son las funciones en vRealize Automation Code Stream

Según su función en vRealize Automation Code Stream, puede realizar ciertas acciones y acceder a determinadas áreas. Por ejemplo, su función puede permitirle crear, actualizar y ejecutar canalizaciones. También es posible que solo tenga permiso para ver canalizaciones.

Todo: `restringido` significa que esta función tiene permiso para realizar acciones de creación, lectura, actualización y eliminación sobre las entidades, excepto para los endpoints y las variables restringidas.

Tabla 2-1. Permisos de acceso a nivel de proyecto y servicio en vRealize Automation Code Stream

Funciones de vRealize Automation Code Stream					
Niveles de acceso	Administrador	Desarrollador	Ejecutor	Visor	Usuario
Acceso a nivel de servicio de vRealize Automation Code Stream	Todas las acciones	Todo: restringido	Acciones de ejecución	Solo lectura	Ninguna
Acceso a nivel de proyecto: administrador del proyecto	Todas las acciones	Todo: restringido	Todo: restringido	Todo: restringido	Todo: restringido
Acceso a nivel de proyecto: miembro del proyecto	Todas las acciones	Todo: restringido	Todo: restringido	Todo: restringido	Todo: restringido
Acceso a nivel de proyecto: visor de proyectos	Todas las acciones	Todo: restringido	Acciones de ejecución	Solo lectura	Solo lectura

Los usuarios que tengan la función de visor de servicios pueden ver toda la información que está disponible para el administrador. No pueden realizar ninguna acción, a menos que un administrador la realice como un administrador del proyecto o un miembro del proyecto. Si el usuario está afiliado a un proyecto, tiene los permisos relacionados con la función. El visor de proyectos no extenderá sus permisos de la forma en que lo hace el administrador o el miembro. Esta función es de solo lectura en todos los proyectos.

Tabla 2-2. Permisos y funciones en vRealize Automation Code Stream

Permiso	Función de administrador	Función de desarrollador	Función de ejecutor	Función de visor	Función de usuario
Canalizaciones: ver	Sí	Sí	Sí	Sí	
Canalizaciones: crear	Sí	Sí			
Canalizaciones: ejecutar	Sí	Sí	Sí		
Canalizaciones: ejecutar canalizaciones que incluyan endpoints o variables restringidos.	Sí				
Canalizaciones: actualizar	Sí	Sí			
Canalizaciones: eliminar	Sí	Sí			
Ejecuciones de la canalización: ver	Sí	Sí	Sí	Sí	

Tabla 2-2. Permisos y funciones en vRealize Automation Code Stream (continuación)

Permiso	Función de administrador	Función de desarrollador	Función de ejecutor	Función de visor	Función de usuario
Ejecuciones de la canalización: reanudar, pausar, cancelar	Sí	Sí	Sí		
Ejecuciones de la canalización: reanudar canalizaciones que se detienen para su autorización en recursos restringidos.	Sí				
Integraciones personalizadas: Crear	Sí	Sí			
Integraciones personalizadas: Leer	Sí	Sí			
Integraciones personalizadas: Actualizar	Sí	Sí			
Endpoints: ver	Sí	Sí	Sí	Sí	
Endpoints: crear	Sí	Sí			
Endpoints: actualizar	Sí	Sí			
Endpoints: eliminar	Sí	Sí			
Endpoint o variable: marcar como restringido	Sí				
Paneles de control: ver	Sí	Sí	Sí	Sí	
Paneles de control: crear	Sí	Sí			
Paneles de control: actualizar	Sí	Sí			
Paneles de control: eliminar	Sí	Sí			

Si tiene la función de administrador

Como administrador, puede crear endpoints de integración, activadores, nuevas canalizaciones y paneles de control.

Los proyectos permiten que las canalizaciones accedan a recursos de infraestructura. Los administradores crean proyectos para que los usuarios puedan agrupar canalizaciones, endpoints y paneles de control. A continuación, los usuarios seleccionan el proyecto en sus canalizaciones. Cada proyecto incluye un administrador y usuarios con funciones asignadas.

Con la función de administrador, puede marcar endpoints y variables como recursos restringidos en una canalización, y ejecutar canalizaciones que utilicen recursos restringidos. El endpoint o la variable restringidos que utiliza una canalización requieren una autorización para mantener la canalización en ejecución. De lo contrario, la canalización se detiene en la tarea donde se utiliza

la variable restringida hasta que se concede la aprobación, momento en el que un administrador debe reanudar la canalización para que se ejecute. Cuando una tarea de canalización incluye un recurso restringido, la tarea en la canalización muestra un icono que indica que el recurso es restringido.

Como administrador, también puede solicitar que las canalizaciones se publiquen en vRealize Automation Service Broker.

Si tiene la función de desarrollador

Puede trabajar con canalizaciones igual que un administrador, excepto que no puede trabajar con endpoints ni variables restringidos.

Si ejecuta una canalización que utiliza variables o extremos restringidos, la canalización solo se ejecutará hasta la tarea que utiliza el recurso restringido. A continuación, se detiene. Después, debe obtener la autorización para la tarea de canalización y solicitar a un administrador que reanude la canalización.

Si tiene la función de usuario

Puede acceder a vRealize Automation Code Stream, pero no tiene ningún privilegio, como brindan las otras funciones.

Si tiene la función de visor

Puede ver canalizaciones, endpoints, ejecuciones de canalizaciones y paneles de control, pero no puede crearlos, actualizarlos ni eliminarlos.

Un usuario que también tiene la función de visor de servicios puede ver toda la información que está disponible para el administrador. No puede realizar ninguna acción a menos que se lo convierta en un administrador del proyecto o en un miembro del proyecto. Si el usuario está afiliado a un proyecto, tiene los permisos relacionados con la función. El visor de proyectos no extenderá sus permisos de la forma en que lo hace el administrador o el miembro.

Si tiene la función de ejecutor

Puede ejecutar canalizaciones y realizar acciones en las tareas de operación de usuario. También puede reanudar, pausar y cancelar ejecuciones de canalizaciones. Sin embargo, no puede modificar las canalizaciones.

Cómo se asignan y se actualizan las funciones

Para asignar y actualizar las funciones de otros usuarios, debe ser un administrador.

- 1 Para ver los usuarios activos y sus funciones, en vRealize Automation, haga clic en los nueve puntos de la parte superior derecha.
- 2 Haga clic en **Administración de identidades y acceso**.



- Para mostrar los nombres de usuario y las funciones, haga clic en **Usuarios activos**.



- Para agregar funciones a un usuario o cambiar sus funciones, haga clic en la casilla de verificación situada junto al nombre de usuario y, a continuación, haga clic en **Editar funciones**.
- Al agregar o cambiar las funciones de usuario, también puede agregar acceso a los servicios.
- Para guardar los cambios, haga clic en **Guardar**.

¿Qué son las aprobaciones y las operaciones de usuario en vRealize Automation Code Stream?

El área Operaciones de usuario muestra las ejecuciones de canalizaciones que necesitan aprobación. El usuario que se requiere como aprobador puede aprobar o rechazar la ejecución de la canalización.

Al crear una canalización, puede que sea necesario agregar una aprobación a una canalización en los siguientes casos:

- Un miembro del equipo debe revisar el código.
- Otro usuario debe confirmar un artefacto de compilación.
- Debe asegurarse de que todas las pruebas se completaron.
- Una tarea utiliza un recurso que un administrador marcó como restringido, y la tarea necesita aprobación.
- La canalización publicará el software en la etapa de producción.

El aprobador requerido para una tarea de canalización debe tener los permisos y la experiencia necesarios para determinar si desea aprobarla.

Después de que el usuario requerido apruebe la tarea de operación del usuario:

- La ejecución de la canalización pendiente puede continuar.
- Cuando la canalización continúa, se cancela cualquier solicitud de aprobación pendiente anterior de la misma tarea de operación de usuario.

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	fritz@vmware.com												
<input type="checkbox"/>	> a0a990	Demo2-Jenkins-K8s#6	Testing	fritz	Nov 11, 2019, 1:34:11 PM	k...@vmware.com, fritz@vmware.com												
<input checked="" type="checkbox"/>	▼	User Operation #8f1728 <hr/> Request Details <table> <tr> <td>Execution</td> <td>Demo-Jenkins-K8s #5</td> </tr> <tr> <td>Summary</td> <td>Testing</td> </tr> <tr> <td>Approvers</td> <td>k...@vmware.com, fritz@vmware.com</td> </tr> <tr> <td>Requested By</td> <td>fritz</td> </tr> <tr> <td>Requested On</td> <td>Nov 11, 2019, 1:22:21 PM</td> </tr> <tr> <td>Expires On</td> <td>Nov 14, 2019, 1:22:21 PM</td> </tr> </table>					Execution	Demo-Jenkins-K8s #5	Summary	Testing	Approvers	k...@vmware.com, fritz@vmware.com	Requested By	fritz	Requested On	Nov 11, 2019, 1:22:21 PM	Expires On	Nov 14, 2019, 1:22:21 PM
Execution	Demo-Jenkins-K8s #5																	
Summary	Testing																	
Approvers	k...@vmware.com, fritz@vmware.com																	
Requested By	fritz																	
Requested On	Nov 11, 2019, 1:22:21 PM																	
Expires On	Nov 14, 2019, 1:22:21 PM																	

1 Items per page 20 1 - 7 of 7 items

En el área Operaciones de usuario, los elementos que se deben aprobar o rechazar aparecen como elementos activos o inactivos. Cada elemento se asigna a una tarea de operación de usuario en una canalización.

- **Los elementos activos** esperan a que el aprobador revise la tarea, y la apruebe o la rechace. Si es un usuario de la lista de aprobadores, puede expandir la fila de operaciones de usuario y hacer clic en **Aceptar** o **Rechazar**.
- **Los elementos inactivos** se aprobaron o rechazaron. Si un usuario rechazó la operación de usuario o se agotó el tiempo de espera para la aprobación de la tarea, ya no se puede aprobar.

Index# es una cadena única de seis caracteres alfanuméricos que puede usar como filtro para buscar una aprobación determinada.

Las aprobaciones de canalización también aparecen en el área **Ejecuciones**.

- Las canalizaciones que están en espera de aprobación indican su estado como en espera.
- Otros estados son en cola, completado y error.
- Si la canalización está en estado de espera, otro usuario debe aprobar la tarea de canalización.

Crear y usar canalizaciones en vRealize Automation Code Stream

3

Puede utilizar vRealize Automation Code Stream para modelar el proceso de compilación, prueba e implementación. Con vRealize Automation Code Stream, puede configurar la infraestructura que admite el ciclo de publicación y crear canalizaciones que modelen las actividades de la publicación de software. vRealize Automation Code Stream proporciona software desde el código de desarrollo, pasando por las pruebas, hasta implementarlo en las instancias de producción.

Cada canalización incluye etapas y tareas. Las etapas representan las fases de desarrollo y las tareas realizan las acciones necesarias para distribuir la aplicación de software a través de las etapas.

Descripción de las canalizaciones en vRealize Automation Code Stream

Una canalización es un modelo de integración y entrega continuas del proceso de publicación de software. Publica el software desde el código fuente hasta la producción, pasando por las pruebas. Incluye una secuencia de etapas con tareas que representan las actividades del ciclo de publicación de software. La aplicación de software pasa de una etapa a otra a través de la canalización.

Agregue endpoints para que las tareas de la canalización se puedan conectar a orígenes de datos, repositorios o sistemas de notificaciones.

Crear canalizaciones

Para crear una canalización, puede comenzar con un lienzo en blanco, utilizar una plantilla inteligente o importar un código YAML.

- Use el lienzo en blanco. Para ver un ejemplo, consulte [Planificar una compilación nativa de CI/CD en vRealize Automation Code Stream antes de agregar tareas manualmente](#).
- Use una plantilla inteligente. Para ver un ejemplo, consulte [Capítulo 4 Planificar la compilación, la integración y la distribución nativas del código en vRealize Automation Code Stream](#).
- Importe el código YAML. Haga clic en **Canalizaciones > Importar**. En el cuadro de diálogo **Importar**, seleccione el archivo YAML o introduzca el código YAML y haga clic en **Importar**.

Cuando se utiliza el lienzo en blanco para crear una canalización, se agregan fases, tareas y aprobaciones. La canalización automatiza el proceso para compilar, probar, implementar y publicar la aplicación. Las tareas de cada etapa ejecutan acciones que compilan, prueban y publican el código a través de cada etapa.

Tabla 3-1. Etapas de canalización de ejemplo y usos

Ejemplo de etapa...	Ejemplos de lo que puede hacer...
Desarrollo	<p>En la etapa de desarrollo, puede aprovisionar una máquina, recuperar un artefacto y agregar una tarea de compilación a fin de crear un host de Docker que se usará para la integración continua del código, entre otras tareas.</p> <p>Por ejemplo:</p> <ul style="list-style-type: none"> ■ Para planificar y crear una compilación de integración continua (Continuous Integration, CI), la cual proporciona el código mediante la capacidad de compilación nativa en vRealize Automation Code Stream, consulte Planificar una compilación nativa de CI en vRealize Automation Code Streamantes de usar la plantilla inteligente.
Pruebas	<p>En la etapa de pruebas, puede agregar una tarea de Jenkins para probar la aplicación de software e incluir herramientas de pruebas de procesamiento posterior, como JUnit, JaCoCo, entre otras.</p> <p>Por ejemplo:</p> <ul style="list-style-type: none"> ■ Integre vRealize Automation Code Stream con Jenkins y ejecute un trabajo de Jenkins en la canalización que compile y pruebe el código fuente. Consulte Cómo se integra vRealize Automation Code Stream con Jenkins. ■ Puede crear scripts personalizados que amplíen la capacidad de vRealize Automation Code Stream de integración con sus propias herramientas de compilación, prueba e implementación. Consulte Cómo se integran las herramientas personalizadas de compilación, pruebas e implementación con vRealize Automation Code Stream. ■ Realice un seguimiento de las tendencias del procesamiento posterior para una canalización de integración continua (continuous integration, CI). Consulte Cómo realizar un seguimiento de los indicadores clave de rendimiento de la canalización en vRealize Automation Code Stream.
Producción	<p>En una etapa de producción, es posible integrar con blueprints en vRealize Automation Cloud Assembly para aprovisionar la infraestructura, implementar el software en un clúster de Kubernetes y más.</p> <p>Por ejemplo:</p> <ul style="list-style-type: none"> ■ Para ver ejemplos de etapas de desarrollo y producción, que puedan implementar la aplicación de software en su propio modelo de implementación azul-verde, consulte Cómo implementar la aplicación en vRealize Automation Code Stream en una implementación azul-verde. ■ Para integrar un blueprint en la canalización, consulte Cómo automatizar la publicación de una aplicación implementada a partir de un blueprint de YAML en vRealize Automation Code Stream. También puede agregar una tarea de implementación que ejecute un script para implementar la aplicación. ■ Para automatizar la implementación de la aplicación de software en el clúster de Kubernetes, consulte Cómo automatizar el lanzamiento de una aplicación en vRealize Automation Code Stream en un clúster de Kubernetes. ■ Para integrar el código en la canalización e implementar la imagen de compilación, consulte Cómo integrar de forma continua código de un repositorio de GitLab o GitHub en la canalización en vRealize Automation Code Stream.

Puede exportar la canalización como un archivo YAML. Haga clic en **Canalizaciones**, haga clic en una tarjeta de canalización y, a continuación, haga clic en **Acciones > Exportar**.

Aprobar canalizaciones

Puede obtener una aprobación de otro miembro del equipo en puntos específicos de la canalización.

- Para solicitar la aprobación en una canalización mediante la inclusión de una tarea de operación del usuario en una canalización, consulte [Cómo ejecutar una canalización y ver los resultados](#). Esta tarea envía una notificación por correo electrónico al usuario que debe revisarla. El revisor debe aprobar o rechazar la aprobación para que la canalización pueda continuar ejecutándose.
- En cualquier etapa de una canalización, si se produce un error en una tarea o una etapa, puede hacer que vRealize Automation Code Stream cree un ticket de Jira. Consulte [Cómo crear un ticket de JIRA en vRealize Automation Code Stream cuando se produce un error en una tarea de canalización](#).

Activar canalizaciones

Se pueden activar canalizaciones cuando los desarrolladores comprueban código, realizan revisiones de código, o cuando se crea o se actualiza un artefacto de compilación.

- Para integrar vRealize Automation Code Stream con el ciclo de vida de Git y activar una canalización cuando los desarrolladores actualicen el código, utilice el activador de Git. Consulte [Cómo usar el activador de Git en vRealize Automation Code Stream para ejecutar una canalización](#).
- Para integrar vRealize Automation Code Stream con el ciclo de vida de revisión de código de Gerrit y activar una canalización en las revisiones de código, use el activador de Gerrit. Consulte [Cómo usar el activador de Gerrit en vRealize Automation Code Stream para ejecutar una canalización](#).
- Para activar una canalización cuando se crea o se actualiza un artefacto de compilación de Docker, utilice el activador de Docker. Consulte [Cómo usar el activador de Docker en vRealize Automation Code Stream para ejecutar una canalización de entrega continua](#).

Para obtener más información sobre los activadores admitidos por vRealize Automation Code Stream, consulte [Capítulo 7 Activar canalizaciones en vRealize Automation Code Stream](#).

Este capítulo incluye los siguientes temas:

- [Cómo ejecutar una canalización y ver los resultados](#)
- [Qué tipos de tareas se encuentran disponibles en vRealize Automation Code Stream](#)
- [Cómo utilizar las variables de enlace en canalizaciones de vRealize Automation Code Stream](#)
- [Cómo utilizar enlaces de variables en una tarea de condición para ejecutar o detener una canalización en vRealize Automation Code Stream](#)

- Qué variables y expresiones se pueden utilizar al enlazar tareas de canalización en vRealize Automation Code Stream
- Cómo enviar notificaciones acerca de la canalización en vRealize Automation Code Stream
- Cómo crear un ticket de JIRA en vRealize Automation Code Stream cuando se produce un error en una tarea de canalización
- Cómo revertir una implementación en vRealize Automation Code Stream

Cómo ejecutar una canalización y ver los resultados

Puede ejecutar una canalización desde la tarjeta de canalización, en el modo de edición de canalización, y desde la ejecución de la canalización. También puede utilizar los activadores disponibles para que vRealize Automation Code Stream ejecute una canalización cuando se produzcan determinados eventos.

Cuando todas las etapas y las tareas de la canalización son válidas, la canalización está lista para publicarse, ejecutarse o activarse.

Para ejecutar o activar la canalización con vRealize Automation Code Stream, puede habilitarla y ejecutarla desde la tarjeta de canalización o mientras se encuentra en la canalización. A continuación, puede ver la ejecución de la canalización para confirmar que la canalización haya compilado, probado e implementado el código.

Cuando la ejecución de una canalización está en curso, puede eliminarla si es un administrador o un usuario que no es administrador.

- Administrador: para eliminar la ejecución de una canalización cuando se está ejecutando, haga clic en **Ejecuciones**. En la ejecución que desea eliminar, haga clic en **Acciones > Eliminar**.
- Usuario no administrador: para eliminar la ejecución de una canalización en ejecución, haga clic en **Ejecuciones** y, a continuación, en **Alt Shift d**.

Cuando la ejecución de una canalización está en curso y parece estar bloqueada, un administrador puede actualizarla desde la página Ejecuciones o la página de detalles de la ejecución.

- Página Ejecuciones: haga clic en **Ejecuciones**. En la ejecución que desea actualizar, haga clic en **Acciones > Sincronizar**.
- Página de detalles de la ejecución: haga clic en **Ejecuciones**, haga clic en el vínculo a los detalles de la ejecución y, a continuación, en **Acciones > Sincronizar**.

Para ejecutar una canalización cuando se produzcan eventos específicos, utilice los activadores.

- El activador de Git puede ejecutar una canalización cuando los desarrolladores actualicen el código.
- El activador de Gerrit puede ejecutar una canalización cuando se produzcan las revisiones de código.
- El activador de Docker puede ejecutar una canalización cuando se crea un artefacto en un registro de Docker.

- El comando `curl` puede hacer que Jenkins ejecute una canalización después de que finalice una compilación de Jenkins.

Para obtener más información sobre cómo usar activadores, consulte [Capítulo 7 Activar canalizaciones en vRealize Automation Code Stream](#).

El siguiente procedimiento muestra cómo ejecutar una canalización desde la tarjeta de canalización, ver las ejecuciones, consultar los detalles de la ejecución y usar las acciones. También muestra cómo publicar una canalización para poder agregarla a vRealize Automation Service Broker.

Requisitos previos

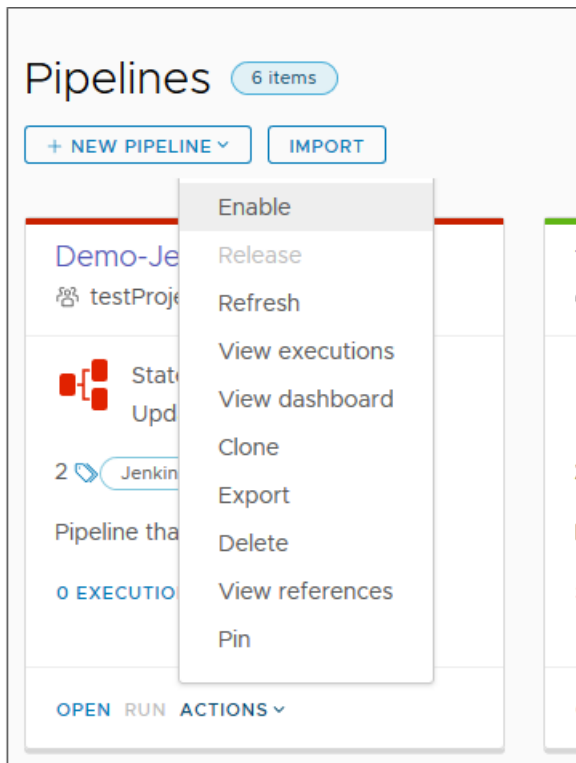
- Confirme que se crearon una o varias canalizaciones. Vea los ejemplos de [Capítulo 5 Ejemplos de casos prácticos comunes para vRealize Automation Code Stream](#).

Procedimiento

- 1 Habilite su canalización.

Para ejecutar o publicar una canalización, primero debe habilitarla.

- a Haga clic en **Canalizaciones**.
- b En la tarjeta de canalización, haga clic en **Acciones > Habilitar**.



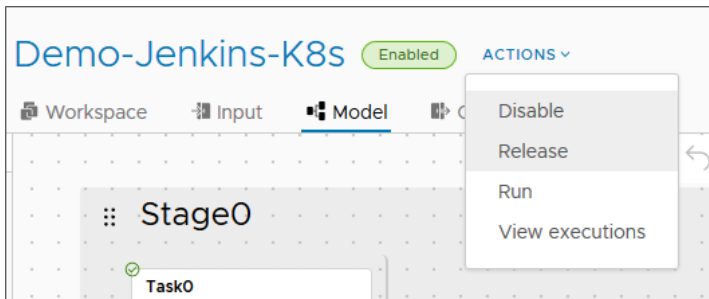
También puede habilitar la canalización mientras se encuentra en la canalización. Si ya se habilitó la canalización, la opción **Ejecutar** se muestra activa y el menú **Acciones** indica **Deshabilitar**.

2 (opcional) Publique la canalización.

Si desea que la canalización esté disponible como un elemento del catálogo en vRealize Automation Service Broker, debe publicarla en vRealize Automation Code Stream.

- a Haga clic en **Canalizaciones**.
- b En la tarjeta de canalización, haga clic en **Acciones > Publicar**.

También puede publicar la canalización mientras se encuentra en ella.



Después de publicar la canalización, abra vRealize Automation Service Broker para agregarla como un elemento del catálogo y ejecutarla. Consulte [Agregar canalizaciones de vRealize Automation Code Stream al catálogo devRealize Automation Service Broker](#).

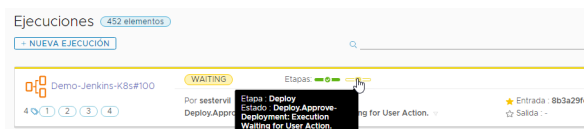
Nota Si la canalización requiere más de 120 minutos para ejecutarla, proporcione una hora de ejecución aproximada como un valor de tiempo de espera de solicitud. Para establecer o revisar el tiempo de espera de solicitud de un proyecto, abra vRealize Automation Service Broker como administrador y seleccione **Infraestructura > Proyectos**. Haga clic en el nombre del proyecto y, a continuación, en **Aprovisionar**.

Si no se establece el valor de tiempo de espera de la solicitud, una ejecución que requiera más de 120 minutos para ejecutarse mostrará errores del tipo de solicitud de tiempo de espera para la devolución de llamada. Sin embargo, esto no afectará a la ejecución de la canalización.

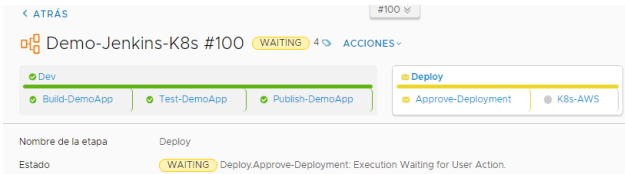
- 3 En la tarjeta de canalización, haga clic en **Ejecutar**.
- 4 Para ver la canalización mientras se ejecuta, haga clic en **Ejecuciones**.

La canalización ejecuta cada etapa en secuencia y la ejecución de la canalización muestra un icono de estado para cada etapa. Si la canalización incluye una tarea de operación de usuario, el usuario debe aprobar la tarea para que la canalización continúe ejecutándose. Cuando se utiliza una tarea de operación de usuario, la canalización deja de ejecutarse y espera a que el usuario requerido apruebe la tarea.

Por ejemplo, puede utilizar la tarea operación de usuario para aprobar la implementación de código en un entorno de producción.



- 5 Para ver la etapa de la canalización a la espera de la aprobación del usuario, haga clic en el icono de estado de la etapa.



- 6 Para ver los detalles de la tarea, haga clic en la tarea.

Una vez que el usuario requerido apruebe la tarea, un usuario que tenga la función adecuada debe reanudar la canalización. Para conocer las funciones requeridas, consulte [Cómo se administran las autorizaciones y el acceso de los usuarios en vRealize Automation Code Stream](#).

Si se produce un error en la ejecución, debe clasificar y solucionar la causa del error. A continuación, vaya a la ejecución y haga clic en **Acciones > Volver a ejecutar**.

Puede reanudar las ejecuciones de la canalización principal y las ejecuciones anidadas.



- 7 Desde la ejecución de la canalización, puede hacer clic en **Acciones** para ver la canalización y seleccionar una acción, como **Pausar** o **Cancelar**, entre otras. Cuando la ejecución de una canalización está en curso, si es un administrador, puede eliminar o sincronizar esta ejecución. Si es un usuario no administrador, puede eliminar una canalización que está en ejecución.

- 8 Para desplazarse fácilmente entre ejecuciones y ver los detalles de una tarea, haga clic en **Ejecuciones** y, a continuación, en una ejecución de canalización. Luego, haga clic en la pestaña en la parte superior de la ejecución y seleccione una ejecución.



Resultados

Enhorabuena. Ejecutó una canalización, examinó la ejecución de la canalización y vio una tarea de operación de usuario que requirió aprobación para que la canalización continúe ejecutándose. También utilizó el menú **Acciones** en la ejecución de la canalización para regresar al modelo de canalización y realizar los cambios necesarios.

Pasos siguientes

Para obtener más información sobre cómo usar vRealize Automation Code Stream para automatizar el ciclo de lanzamiento de software, consulte [Capítulo 5 Ejemplos de casos prácticos comunes para vRealize Automation Code Stream](#).

Qué tipos de tareas se encuentran disponibles en vRealize Automation Code Stream

Es posible agregar tipos de tareas específicos a la canalización para que esta realice ciertas acciones. Cada tipo de tarea se integra con otra aplicación para permitir que la canalización logre el propósito para el cual se la diseñó.

Si necesita extraer artefactos de un repositorio para la implementación, ejecutar un script remoto o solicitar la aprobación de un miembro del equipo para que se ejecute la canalización, vRealize Automation Code Stream tiene el tipo de tarea para usted.

Antes de utilizar el tipo de tarea en la canalización, compruebe que el endpoint correspondiente esté disponible.

Tabla 3-2. Obtención de una aprobación o establecimiento de un punto de decisión

Tipo de tarea...	Lo que hace...	Ejemplos y detalles...
Operación de usuario	Habilita una aprobación requerida que controla cuándo se ejecuta una canalización y cuándo debe detenerse para su aprobación.	Consulte Cómo ejecutar una canalización y ver los resultados. y Cómo se administran las autorizaciones y el acceso de los usuarios en vRealize Automation Code Stream.
Condición	Agrega un punto de decisión, que determina si la canalización debe continuar ejecutándose o detenerse en función de las expresiones de condición. Cuando la condición es verdadera, la canalización ejecuta las tareas sucesivas. Cuando el valor es falso, la canalización se detiene.	Consulte Cómo utilizar enlaces de variables en una tarea de condición para ejecutar o detener una canalización en vRealize Automation Code Stream.

Tabla 3-3. Automatización de la implementación y la integración continua

Tipo de tarea...	Lo que hace...	Ejemplos y detalles...
Blueprint	Implementa un blueprint de automatización de GitHub y aprovisiona una aplicación, y automatiza la integración continua y entrega continua (Continuous Integration and Continuous Delivery, CI/CD) de ese blueprint para la implementación.	<p>Consulte Cómo automatizar la publicación de una aplicación implementada a partir de un blueprint de YAML en vRealize Automation Code Stream.</p> <p>Si selecciona Crear o Actualizar en la tarea de blueprint, entonces, seleccione Blueprint y versión y aparecerán los parámetros de blueprint. Puede agregar estos elementos, que admiten enlaces de variables, a las áreas de texto de entrada de la tarea de blueprint:</p> <ul style="list-style-type: none"> ■ Entero ■ Cadena de enumeración ■ Booleano ■ Variable de matriz <p>Cuando utilice el enlace de variables en los campos de entrada, tenga en cuenta estas excepciones. Para enumeraciones, debe seleccionar un valor de enumeración de un conjunto fijo. Para valores booleanos, debe introducir el valor en el área de texto de entrada.</p> <p>El parámetro blueprint aparece en la tarea de blueprint cuando un blueprint de vRealize Automation Cloud Assembly incluye variables de entrada. Por ejemplo, si un blueprint tiene un tipo de entrada <code>Integer</code>, puede introducir el entero directamente o como una variable mediante el enlace de variables.</p>
CI	Permite la integración continua del código en la canalización mediante la extracción de una imagen de compilación de Docker de un endpoint de registro y su implementación en un clúster de Kubernetes.	Consulte Planificar una compilación nativa de CI/CD en vRealize Automation Code Streamantes de usar la plantilla inteligente .
Personalizada	Integra vRealize Automation Code Stream con sus propias herramientas de compilación, prueba e implementación.	Consulte Cómo se integran las herramientas personalizadas de compilación, pruebas e implementación con vRealize Automation Code Stream .

Tabla 3-3. Automatización de la implementación y la integración continua (continuación)

Tipo de tarea...	Lo que hace...	Ejemplos y detalles...
Kubernetes	Automatiza la implementación de las aplicaciones de software en clústeres de Kubernetes en AWS.	Consulte Cómo automatizar el lanzamiento de una aplicación en vRealize Automation Code Stream en un clúster de Kubernetes .
Canalización	<p>Anida una canalización en una canalización principal. Cuando se anida una canalización, esta funciona como una tarea en la canalización principal.</p> <p>En la pestaña Tarea de la canalización principal, puede desplazarse fácilmente hasta la canalización anidada si hace clic en el vínculo que dirige a ella. La canalización anidada se abrirá en una nueva pestaña del explorador.</p>	Para buscar canalizaciones anidadas en Ejecuciones , introduzca nested en el área de búsqueda.

Tabla 3-4. Integración de aplicaciones de desarrollo, prueba e implementación

Tipo de tarea...	Lo que hace...	Ejemplos y detalles...
Bamboo	Interactúa con un servidor de integración continua (continuous integration, CI) de Bamboo que crea, prueba e integra continuamente el software en preparación para la implementación, y activa las compilaciones de código cuando los desarrolladores confirman cambios. Expone las ubicaciones de los artefactos que genera la compilación de Bamboo para que la tarea pueda generar los parámetros de otras tareas que se utilizarán para la compilación y la implementación.	Conéctese a un endpoint de servidor de Bamboo e inicie un plan de compilación de Bamboo desde la canalización.
Jenkins	Activa los trabajos de Jenkins que compilan y prueban el código fuente, ejecuta casos de prueba y puede utilizar scripts personalizados.	Consulte Cómo se integra vRealize Automation Code Stream con Jenkins .
TFS	Permite conectar la canalización a Team Foundation Server para administrar e invocar proyectos de compilación, incluidos los trabajos configurados que compilan y prueban código.	vRealize Automation Code Stream es compatible con Team Foundation Server 2013 y 2015.
vRO	Amplía la capacidad de vRealize Automation Code Stream mediante la ejecución de flujos de trabajo predefinidos o personalizados en vRealize Orchestrator.	Consulte Cómo se integra vRealize Automation Code Stream con vRealize Orchestrator .

Tabla 3-5. Integración de otras aplicaciones a través de una API

Tipo de tarea...	Lo que hace...	Ejemplos y detalles...
REST	Integra vRealize Automation Code Stream con otras aplicaciones que utilizan REST API para que usted pueda desarrollar y distribuir continuamente aplicaciones de software que interactúen entre sí.	Consulte Cómo usar una API de REST para integrar vRealize Automation Code Stream con otras aplicaciones .
Sondeo	Invoca una instancia de REST API y la sondea hasta que la tarea de canalización cumple con los criterios de salida y se completa.	Consulte Cómo usar una API de REST para integrar vRealize Automation Code Stream con otras aplicaciones .

Tabla 3-6. Ejecución de scripts remotos y definidos por el usuario

Tipo de tarea...	Lo que hace...	Ejemplos y detalles...
PowerShell	<p>Permite que el tipo de tarea Script de PowerShell ejecute comandos de script en un host remoto. Por ejemplo, un script puede automatizar las tareas de prueba y ejecutar tipos de comandos administrativos.</p> <p>El script puede ser remoto o definido por el usuario. Puede conectarse a través de HTTP o HTTPS, y puede utilizar TLS.</p> <p>Se debe configurar el servicio con el nombre <code>winrm</code> en el host de Windows y se debe configurar <code>winrm</code> para <code>MaxShellsPerUser</code> y <code>MaxMemoryPerShellMB</code>.</p>	<p>Al configurar <code>MaxShellsPerUser</code> y <code>MaxMemoryPerShellMB</code>:</p> <ul style="list-style-type: none"> El valor aceptable para <code>MaxShellsPerUser</code> es 500 para 50 canalizaciones simultáneas, con 5 tareas de PowerShell para cada canalización. Para establecer el valor, ejecute: <code>winrm set winrm/config/winrs '@{MaxShellsPerUser="500"}'</code> El valor de memoria aceptable para <code>MaxMemoryPerShellMB</code> es 2048. Para establecer el valor, ejecute: <code>winrm set winrm/config/winrs '@{MaxMemoryPerShellMB="2048"}'</code> <p>El script escribe la salida en un archivo de respuesta que puede consumir otra canalización.</p>
SSH	<p>Permite que el tipo de tarea Script de shell de Bash ejecute comandos de script en un host remoto. Por ejemplo, un script puede automatizar las tareas de prueba y ejecutar tipos de comandos administrativos.</p> <p>El script puede ser remoto o definido por el usuario. Puede conectarse a través de HTTP o HTTPS, y requiere una clave privada o una contraseña.</p> <p>Se debe configurar el servicio SSH en el host de Linux y la configuración <code>SSHD</code> de <code>MaxSessions</code> se debe establecer en 50.</p>	<p>El script puede ser remoto o definido por el usuario. Por ejemplo, un script puede ser similar al siguiente:</p> <pre>message="Hello World" echo \$message</pre> <p>El script escribe la salida en un archivo de respuesta que puede consumir otra canalización.</p>

Cómo utilizar las variables de enlace en canalizaciones de vRealize Automation Code Stream

Enlazar una tarea de canalización significa que se crea una dependencia para la tarea cuando se ejecuta la canalización. Puede crear un enlace para una tarea de canalización de varias maneras. Puede enlazar una tarea a otra tarea, enlazarla a una variable y expresión, o enlazarla a una condición.

Cómo aplicar enlaces de símbolo de dólar a variables de blueprint en una tarea de blueprint

Se pueden aplicar enlaces de símbolo de dólar a variables de blueprint en una tarea de blueprint de canalización de vRealize Automation Code Stream. La manera en la que se modifican las variables en vRealize Automation Code Stream depende de cómo se codifiquen las propiedades de variable en el blueprint.

Si necesita utilizar enlaces de símbolo de dólar en una tarea de blueprint, pero la versión actual del blueprint que utiliza en la tarea de blueprint no lo permite, modifique el blueprint en vRealize Automation Cloud Assembly e implemente una versión nueva. Después, utilice la versión nueva de blueprint en la tarea de blueprint y agregue los enlaces de símbolo de dólar donde sea necesario.

Para aplicar enlaces de símbolo de dólar en los tipos de propiedad que proporciona el blueprint de vRealize Automation Cloud Assembly, debe tener los permisos correctos.

- Debe tener la misma función que la persona que creó la implementación del blueprint en vRealize Automation Cloud Assembly.
- La persona que modela la canalización y la persona que ejecuta la canalización pueden ser dos usuarios distintos y pueden tener funciones diferentes.
- Si un desarrollador tiene la función de ejecutor de vRealize Automation Code Stream y modela la canalización, el desarrollador deberá tener también la misma función de vRealize Automation Cloud Assembly que la persona que implementó el blueprint. Por ejemplo, la función que se requiere podría ser la de administrador de vRealize Automation Cloud Assembly.
- Solo la persona que modela la canalización tiene permiso para crear la canalización y crear la implementación.

Para utilizar un token de API en la tarea de blueprint:

- La persona que modela la canalización puede proporcionar un token de API a otro usuario que tenga la función de ejecutor de vRealize Automation Code Stream. Después, cuando el ejecutor ejecute la canalización, usará el token de API y las credenciales que crea el token de API.
- Cuando un usuario introduce un token de API en la tarea de blueprint, crea las credenciales que se necesitan para ejecutar la canalización.
- Para cifrar el valor del token de API, haga clic en **Crear variable**.

- Si no crea una variable para el token de API y lo utiliza en la tarea de blueprint, el valor del token de API aparecerá como texto sin formato.

Para aplicar enlaces de símbolo de dólar a variables de blueprint en una tarea de blueprint, realice los siguientes pasos.

Comience con un blueprint que tenga definidas propiedades de variable de entrada; por ejemplo, `integerVar`, `stringVar`, `flavorVar`, `BooleanVar`, `objectVar` y `arrayVar`. Las propiedades de imagen se definen en la sección `resources`. Las propiedades del código de blueprint pueden ser similares a las siguientes:

```
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}'
```

Puede utilizar las variables de signo de dólar (\$) para `image` y `flavor`. Por ejemplo:

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

Para utilizar un blueprint en una canalización de vRealize Automation Code Stream y agregar enlaces de símbolo de dólar a él, siga estos pasos.

- 1 En vRealize Automation Code Stream, haga clic en **Canalizaciones > Lienzo en blanco**.
- 2 Agregue una tarea de **Blueprint** a la canalización.
- 3 En la tarea de blueprint, en **Origen de blueprint**, seleccione **Blueprints de Cloud Assembly**, escriba el nombre del blueprint y seleccione la versión del blueprint.
- 4 Tenga en cuenta que puede introducir un token de API, el cual puede proporcionar credenciales para que la canalización se ejecute, y que puede utilizar **Crear variable** para crear una variable que cifre el token de API en la tarea de blueprint.
- 5 En la tabla **Parámetro y valor** que se muestra, observe los valores de los parámetros. El valor predeterminado de `flavor` es `small` y el valor predeterminado de `image` es `ubuntu`.
- 6 Supongamos que necesita cambiar el blueprint en vRealize Automation Cloud Assembly. Por ejemplo, puede hacer lo siguiente:
 - a Establezca `flavor` para utilizar una propiedad de tipo `array`. vRealize Automation Cloud Assembly permite valores separados por coma para `flavor` cuando el tipo es **array**.
 - b Haga clic en **Implementar**.
 - c En la página Tipo de implementación, introduzca un nombre para la implementación y seleccione la versión del blueprint.
 - d En la página Entradas de implementación, puede definir uno o varios valores para `flavor`.
 - e Tenga en cuenta que las entradas de la implementación incluyen todas las variables definidas en el código de blueprint y aparecerán como se definen en el código del blueprint. Por ejemplo: `Integer Var`, `String Var`, `Flavor Var`, `Boolean Var`, `Object Var` y `Array Var`. `String Var` y `Flavor Var` son valores de cadena y `Boolean Var` es una casilla de verificación.
 - f Haga clic en **Implementar**.
- 7 En vRealize Automation Code Stream, seleccione la nueva versión del blueprint e introduzca los valores en la tabla **Parámetro y valor**. Los blueprints admiten los siguientes tipos de parámetro y la capacidad de que vRealize Automation Code Stream permita enlaces de

variables de símbolo de dólar en ellos, lo que provoca pequeñas diferencias en la interfaz de usuario de la tarea de blueprint de vRealize Automation Code Stream y la interfaz de blueprint de vRealize Automation Cloud Assembly. En función de cómo se codifique un blueprint, podría permitir o no permitir introducir valores en la tarea.

- a Para **flavorVar**, si el blueprint definió el tipo como cadena o matriz, introduzca una cadena o una matriz de valores separados por comas. Una matriz de ejemplo se asemeja a **prueba, prueba**.
- b Para **BooleanVar**, seleccione **true** o **false** en el menú desplegable. Como alternativa, para usar un enlace de variables, introduzca **\$** y seleccione un enlace de variable de la

The screenshot shows a table with two columns: 'Parameter' and 'Value'. The parameters listed are stringVar (value: raj), integerVar (value: 1), flavorVar (value: medium), BooleanVar (value: \$), objectVar, and arrayVar. Below the table, there is a section for 'Output Parameter' with buttons for status, deployment, and deployment. A dropdown menu is open for the BooleanVar parameter, showing a list of variables: var, input, comments, requestBy, executionIndex, executionId, executionUrl, name, description, and Stage0.

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

Output Parameter

status deployment deployment deployment

var
input
comments
requestBy
executionIndex
executionId
executionUrl
name
description
Stage0

- c Para **objectVar**, introduzca el valor con llaves y comillas en este formato: **{"bkix": "bkix":}**.
 - d **objectVar** se transferirá al blueprint y se podrá utilizar de distintas formas en función del blueprint. Permite un formato de cadena para un objeto JSON y se pueden agregar pares clave-valor como valores separados por coma en la tabla de clave-valor. Puede introducir texto sin formato para un objeto JSON o un par clave-valor como un formato de cadena normal para JSON.
 - e Para **arrayVar**, introduzca el valor de entrada separado por comas como una matriz con este formato: **["1", "2"]**.
- 8 En la canalización, puede enlazar un parámetro de entrada a una matriz.
- a Haga clic en la pestaña **Entrada**.
 - b Introduzca un nombre para la entrada. Por ejemplo, **arrayInput**.
 - c En la tabla **Parámetro y valor**, haga clic en **arrayVar** e introduzca **\${input.arrayInput}**.
 - d Después de guardar la canalización y habilitarla, cuando se ejecute la canalización, tendrá que proporcionar un valor de entrada de matriz. Por ejemplo, introduzca **["1", "2"]** y haga clic en **Ejecutar**.

Ahora ya ha aprendido a utilizar los enlaces de variables de signo de dólar (\$) en un blueprint en una tarea de blueprint de canalización de vRealize Automation Code Stream.

Cómo pasar un parámetro a una canalización cuando se ejecuta

Puede agregar parámetros de entrada a la canalización para que vRealize Automation Code Stream las transfiera a la canalización. A continuación, cuando se ejecuta la canalización, un usuario debe introducir el valor del parámetro de entrada. Puede agregar parámetros de salida a la canalización para que las tareas de la canalización puedan utilizar el valor de salida de una tarea. vRealize Automation Code Stream admite el uso de parámetros de muchas maneras que son compatibles con las necesidades de su propia canalización.

Por ejemplo, para solicitar a un usuario la dirección URL de su servidor de Git cuando se ejecuta una canalización con una tarea de REST, puede enlazar la tarea de REST a la URL de un servidor de Git.

Para crear el enlace de variables, agregue una variable de enlace de URL a la tarea de REST. Cuando la canalización ejecute y llegue a la tarea de REST, el usuario deberá introducir la dirección URL en el servidor de Git. Aquí se muestra cómo crearía el enlace:

- 1 En la canalización, haga clic en la pestaña **Entrada**.
- 2 Para establecer el parámetro, para **Parámetros de inserción automática**, haga clic en **Git**.
Aparece la lista de parámetros de Git y se incluye **GIT_SERVER_URL**. Si necesita utilizar un valor predeterminado para la URL del servidor de Git, edite este parámetro.
- 3 Haga clic en **Modelo** y, a continuación, haga clic en la tarea de REST.
- 4 En la pestaña **Tarea**, en el área **URL**, introduzca \$ y, a continuación, seleccione **entrada** y **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 (dropdown menu)
- Continue on failure:** ☐
- Execute task:** ☒ Always ☐ On condition
- REST Request:**
 - Action:** GET (dropdown menu)
 - URL:** \${input.| (with a dropdown menu open showing a list of Git 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. The 'GIT_SERVER_URL' option is highlighted by the mouse cursor).
 - Agent endpoint:**
 - Headers:**
- Output Parameters:** A row of buttons labeled 'status', 'responseHeaders', 'responseBody', 'responseJson', and 'responseCode'. The 'responseBody' button is currently selected.

La entrada es similar a la siguiente: `${input.GIT_SERVER_URL}`

- 5 Para comprobar la integridad del enlace de variables de la tarea, haga clic en **Validar tarea**.
vRealize Automation Code Stream indica que la tarea se validó correctamente.
- 6 Cuando la canalización ejecute la tarea de REST, el usuario deberá introducir la dirección URL en el servidor de Git. De lo contrario, la tarea no finaliza su ejecución.

Cómo enlazar dos tareas de canalización mediante la creación de parámetros de entrada y de salida

Cuando se enlazan dos tareas, se agrega una variable de enlace a la configuración de entrada de la tarea receptora. Posteriormente, cuando se ejecute la canalización, el usuario deberá reemplazar la variable de enlace por la entrada requerida.

Para enlazar las tareas de canalización, utilice la variable de signo de dólar (\$) en los parámetros de entrada y de salida. Aquí le explicamos cómo hacerlo.

Supongamos que necesita la canalización para llamar a una URL en una tarea de REST y generar una respuesta. Para ello, debe incluir parámetros de entrada y de salida en la tarea de REST. También necesita un usuario para aprobar la tarea, por lo que también puede incluir una tarea de operaciones de usuario para que otro usuario la apruebe cuando se ejecute la canalización. Este ejemplo muestra cómo utilizar expresiones en los parámetros de entrada y de salida, y hacer que la canalización espere a la aprobación de la tarea.

- 1 En la canalización, haga clic en la pestaña **Entrada**.

The screenshot shows the 'rest-ix-1' workflow configuration in the 'Input' tab. The 'Auto inject parameters' section has four radio buttons: 'Gerrit', 'Git', 'Docker', and 'None'. The 'None' button is selected. Below this is a table with one parameter:

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

- 2 Deje los **parámetros de inserción automática** como **Ninguno**.
- 3 Haga clic en **Agregar** e introduzca el nombre, el valor y la descripción del parámetro, y haga clic en **Aceptar**. Por ejemplo:
 - a Introduzca un nombre de URL.
 - b Introduzca el valor: `{Stage0.Task3.input.http://www.docs.vmware.com}`
 - c Introduzca una descripción.
- 4 Haga clic en la pestaña **Salida**, haga clic en **Agregar** e introduzca el nombre y la asignación del parámetro de salida.

Add Pipeline Output Parameter

Name *

Reference \$ *

responseHeaders
responseBody
responseJson
responseCode

- Introduzca un nombre de parámetro de salida único.
- Haga clic en el área **Referencia** e introduzca \$.
- Para introducir la asignación de salida de la tarea, seleccione las opciones que aparecen en la ventana emergente. Seleccione **Stage0**, seleccione **Task3**, seleccione **salida** y seleccione **responseCode**. Haga clic en **Aceptar**.

rest-ix-1 Enabled ACTIONS ▾

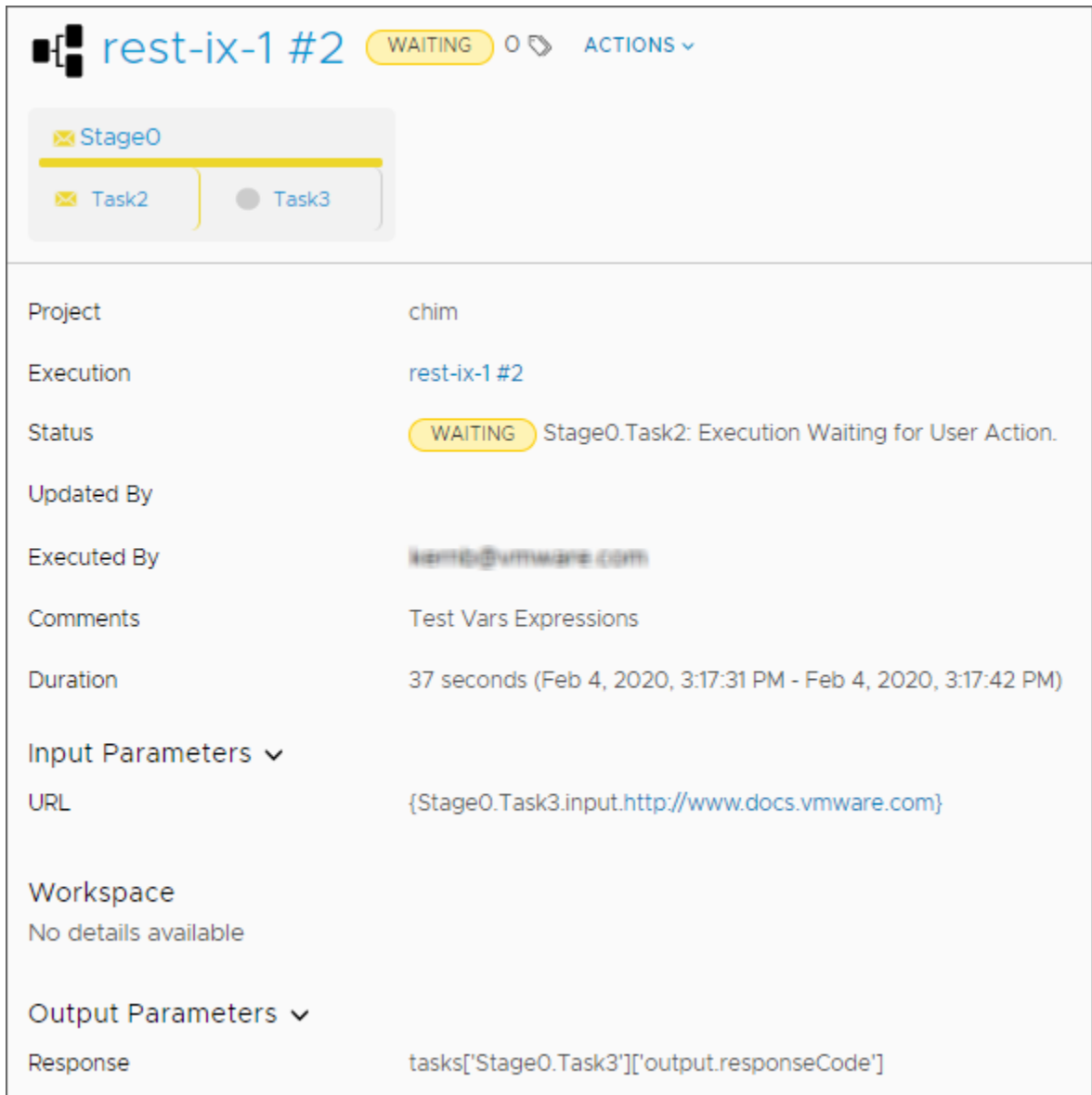
Workspace
Input
Model
Output

Output Parameters ⓘ

ADD

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

- Guarde la canalización.
- En el menú **Acciones**, haga clic en **Ejecutar**.
- Haga clic en **Acciones > Ver ejecuciones**.
- Haga clic en la ejecución y consulte los parámetros de entrada y de salida que definió.



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	kent@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 aprobar la canalización, haga clic en **Operaciones de usuario** y vea la lista de aprobaciones en la pestaña **Elementos activos**. O bien, continúe en las ejecuciones, haga clic en la tarea y, a continuación, haga clic en **Aprobar**.
- 10 Para habilitar los botones **Aprobar** y **Rechazar**, haga clic en la casilla de verificación situada junto a la ejecución.
- 11 Para ver los detalles, expanda la flecha desplegable.
- 12 Para aprobar la tarea, haga clic en **APROBAR**, introduzca un motivo y haga clic en **Aceptar**.

User Operations GUIDED SETUP

Active Items Inactive Items

✓ APPROVE × REJECT

Index# Execution

☒ ☐ User Operation #f0d252

Request Details

Execution rest-ix-1 #2

Summary hello

Approvers kern@vmware.com, fritz@vmware.com

Requested By kern@vmware.com

Requested On Feb 4, 2020, 3:17:40 PM

Expires On Feb 7, 2020, 3:17:40 PM

APPROVE REJECT VIEW DASHBOARD

- 13 Haga clic en **Ejecuciones** y compruebe que la canalización continúe ejecutándose.

Executions 3,347 items GUIDED SETUP

+ NEW EXECUTION

rest-i... #3 RUNNING Stages: ACTIONS

By kern on Feb 4, 2020, 3:41:05 PM

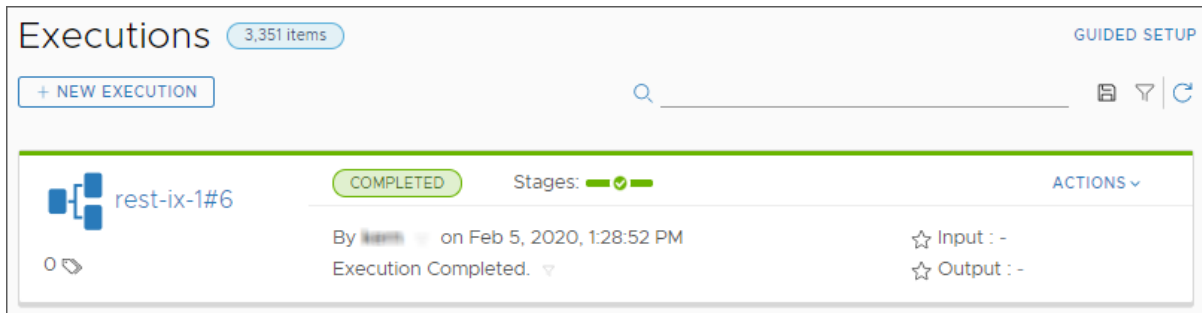
RUNNING

Comments: Testing

☆ Input : -

☆ Output : -

- 14 Si se produce un error en la canalización, corrija los errores y, a continuación, guarde la canalización y ejecútela de nuevo.



Cómo obtener más información sobre variables y expresiones

Para ver los detalles sobre el uso de variables y expresiones al enlazar tareas de canalización, consulte [Qué variables y expresiones se pueden utilizar al enlazar tareas de canalización en vRealize Automation Code Stream](#).

Para obtener información sobre cómo utilizar la salida de la tarea de canalización con un enlace de variable de condición, consulte [Cómo utilizar enlaces de variables en una tarea de condición para ejecutar o detener una canalización en vRealize Automation Code Stream](#).

Cómo utilizar enlaces de variables en una tarea de condición para ejecutar o detener una canalización en vRealize Automation Code Stream

Es posible hacer que la salida de una tarea en la canalización determine si la canalización se ejecuta o se detiene en función de una condición que se proporciona. Para que la canalización se complete correctamente o con errores según la salida de la tarea, utilice el tipo de tarea Condición.

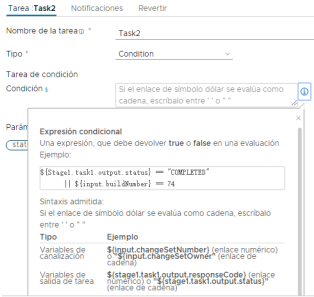
El tipo de tarea **Condición** se utiliza como punto de decisión en la canalización. Al usar la tarea Condición con una expresión de condición proporcionada, es posible evaluar cualquier propiedad de la canalización, las etapas y las tareas.

El resultado de la tarea Condición determina si se ejecuta la siguiente tarea en la canalización.

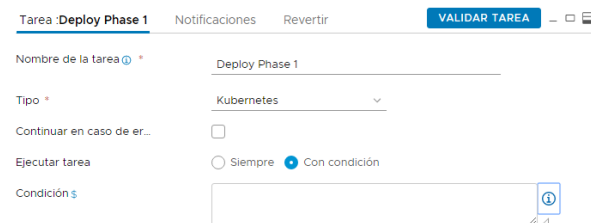
- Una condición verdadera permite que se siga ejecutando la canalización.
- Una condición falsa detiene la canalización.

Si desea obtener ejemplos sobre la forma de enlazar tareas con una tarea Condición para usar el valor de salida de una tarea como la entrada de la siguiente tarea, consulte [Cómo utilizar las variables de enlace en canalizaciones de vRealize Automation Code Stream](#).

Tabla 3-7. Formas en que la tarea Condición y su expresión de condición se relacionan con la canalización

Tipo de tarea		
Condición...	Lo que afecta...	Lo que hace...
Tarea Condición	Canalización	El tipo de tarea Condición determina si la canalización se ejecuta o se detiene en ese punto, en función de si la salida de la tarea es verdadera o falsa.
Expresión de condición	Salida de la tarea de condición	<p>Cuando se ejecuta la canalización, la expresión de condición que se incluye en la tarea Condición genera un estado de salida verdadera o falsa. Por ejemplo, una expresión de condición puede requerir que el estado de salida de la tarea Condición sea <code>Completado</code> o que use el número de compilación 74.</p> <p>La expresión de condición aparece en la pestaña Tarea del tipo de tarea Condición.</p> 

El tipo de tarea **Condición** difiere en función y comportamiento de la opción **Con condición** de otros tipos de tareas.



En otros tipos de tareas, la opción **Con condición**:

- Determina si se ejecuta la tarea actual, en lugar de las tareas sucesivas, en función de la evaluación de su expresión de condición previa verdadera o falsa. La expresión de condición de la opción **Con condición** produce un estado de salida verdadera o falsa para la tarea actual cuando se ejecuta la canalización.
- Aparece en la pestaña Tarea con su propia expresión de condición.

En este ejemplo, se utiliza la tarea Condición.

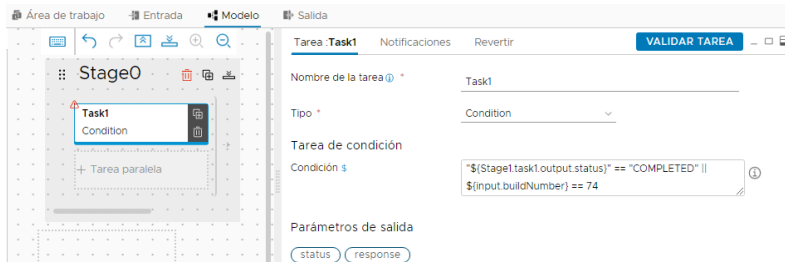
Requisitos previos

- Compruebe que exista una canalización y que esta incluya copias intermedias y tareas.

Procedimiento

- 1 En la canalización, determine el punto de decisión en el que debe aparecer la tarea Condición.
- 2 Agregue la tarea Condición antes de la tarea que depende de su estado correcto o con errores.
- 3 Agregue una expresión de condición a la tarea Condición.

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



- 4 Valide la tarea.
- 5 Guarde la canalización y, a continuación, habilítela y ejecútela.

Resultados

Observe las ejecuciones de la canalización y vea si la canalización se sigue ejecutando o se detiene en la tarea Condición.

Pasos siguientes

También puede utilizar el tipo de tarea Condición si decide revertir una implementación de canalización. Por ejemplo, en una canalización de reversión, la tarea Condición ayuda a vRealize Automation Code Stream a marcar un error de canalización en función de la expresión de condición y puede activar un único flujo de reversión para varios tipos de errores.

Para revertir una implementación, consulte [Cómo revertir una implementación en vRealize Automation Code Stream](#).

Qué variables y expresiones se pueden utilizar al enlazar tareas de canalización en vRealize Automation Code Stream

Las variables y las expresiones permiten usar parámetros de entrada y de salida con las tareas de la canalización. Los parámetros que introduce enlazan la tarea de canalización a una o más variables, expresiones o condiciones, y determinan el comportamiento de la canalización cuando se ejecuta.

Cuando enlaza tareas de canalización, puede incluir expresiones predeterminadas y complejas para que la canalización pueda ejecutar soluciones de distribución de software sencillas o complejas. Para crear los parámetros en la canalización, haga clic en la pestaña **Entrada** o **Salida** y agregue una variable mediante la introducción del signo **\$** y una expresión.

Por ejemplo, este parámetro se utiliza como una entrada de tarea que llama a una URL: `$`

```
{Stage0.Task3.input.URL}.
```

El formato de enlaces de variables utiliza componentes de sintaxis denominados ámbitos y claves. El `SCOPE` define el contexto como entrada o salida, y la `KEY` define los detalles. En la `${Stage0.Task3.input.URL}` de ejemplo del parámetro, `input` es el componente `SCOPE` y la `URL` es el componente `KEY`.

Para obtener más información sobre el uso de enlaces de variables en canalizaciones, consulte [Cómo utilizar las variables de enlace en canalizaciones de vRealize Automation Code Stream](#).

Usar expresiones de dólar con ámbitos y claves para enlazar tareas de canalización

Puede enlazar las tareas de canalización mediante expresiones en variables de signo de dólar. Las expresiones se introducen como `${SCOPE.KEY.<PATH>}`.

En cada expresión, `SCOPE` es el contexto que vRealize Automation Code Stream utiliza para determinar el comportamiento de una tarea de canalización. El ámbito busca una `KEY`, que define el detalle de la acción que lleva a cabo la tarea. Cuando el valor de `KEY` es un objeto anidado, se puede proporcionar una `PATH` opcional.

En estos ejemplos, se describen `SCOPE` y `KEY`, y se muestra cómo utilizarlos en la canalización.

Tabla 3-8. Utilizar SCOPE y KEY

SCOPE	Propósito de expresión y ejemplo	KEY	Cómo utilizar SCOPE y KEY en la canalización
input	Propiedades de entrada de una canalización: <code>\${input.input1}</code>	Nombre de la propiedad de entrada	<p>Para hacer referencia a la propiedad de entrada de una canalización en una tarea, utilice este formato:</p> <pre>tasks: mytask: type: REST input: url: \${input.url} action: get</pre> <pre>input: url: https:// www.vmware.com</pre>
output	Propiedades de salida de una canalización: <code>\${output.output1}</code>	Nombre de la propiedad de salida	<p>Para hacer referencia a una propiedad de salida a fin de que la canalización pueda enviar una notificación, utilice el siguiente 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>

Tabla 3-8. Utilizar SCOPE y KEY (continuación)

SCOPE	Propósito de expresión y ejemplo	KEY	Cómo utilizar SCOPE y KEY en la canalización
task input	<p>Entrada de una tarea:</p> <pre>\$ {MY_STAGE.MY_TASK.input. SOMETHING}</pre>	Indica la entrada de una tarea en una notificación.	<p>Cuando se inicia un trabajo de Jenkins, puede hacer referencia al nombre del trabajo que se activa desde la entrada de la tarea. En este caso, envíe una notificación con 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>
task output	<p>Salida de una tarea:</p> <pre>\$ {MY_STAGE.MY_TASK.output .SOMETHING}</pre>	Indica la salida de una tarea en una tarea posterior.	<p>Para hacer referencia a la salida de la tarea de canalización 1 en la tarea 2, utilice este 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>

Tabla 3-8. Utilizar SCOPE y KEY (continuación)

SCOPE	Propósito de expresión y ejemplo	KEY	Cómo utilizar SCOPE y KEY en la canalización
var	Variable: <code>\${var.myVariable}</code>	Referencia a la variable de un endpoint	<p>Para hacer referencia a una variable secreta en un endpoint de un campo de contraseña, utilice 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	Variable: <code>\${var.myVariable}</code>	Referencia a la variable en una canalización	<p>Para hacer referencia a la variable en una URL de canalización, utilice este formato:</p> <pre> tasks: task1: type: REST input: action: get url: \$ {var.MY_SERVER_URL} </pre>
task status	<p>Estado de una tarea:</p> <pre> \$ {MY_STAGE.MY_TASK.status} </pre> <p>\$</p> <pre> {MY_STAGE.MY_TASK.status Message} </pre>		
stage status	<p>Estado de una etapa:</p> <pre> \${MY_STAGE.status} </pre> <p>\$</p> <pre> {MY_STAGE.statusMessage} </pre>		

Expresiones predeterminadas

Puede utilizar variables con expresiones en la canalización. Este resumen incluye las expresiones predeterminadas que puede utilizar.

Expresión	Descripción
<code>\${comments}</code>	Comentarios proporcionados cuando se solicitó la ejecución
<code>\${duration}</code>	Duración de la ejecución de la canalización
<code>\${endTime}</code>	Hora de finalización de la ejecución de la canalización en UTC (si finalizó)
<code>\${executedOn}</code>	Igual que la hora de inicio, hora de inicio de la ejecución de la canalización en UTC
<code>\${executionId}</code>	Identificador de la ejecución de la canalización
<code>\${executionUrl}</code>	URL que permite acceder a la ejecución de la canalización en la interfaz de usuario
<code>\${name}</code>	Nombre de la canalización
<code>\${requestBy}</code>	Nombre del usuario que solicitó la ejecución
<code>\${stageName}</code>	Nombre de la etapa actual (cuando se utiliza en el ámbito de una etapa)
<code>\${startTime}</code>	Hora de inicio de la ejecución de la canalización en UTC
<code>\${status}</code>	Estado de la ejecución
<code>\${statusMessage}</code>	Mensaje de estado de la ejecución de la canalización
<code>\${taskName}</code>	Nombre de la tarea actual (cuando se utiliza en una notificación o una entrada de tarea)

Cómo usar SCOPE y KEY en tipos de tareas de canalización

Puede utilizar expresiones con cualquiera de los tipos de tareas de canalización compatibles. Utilice estos ejemplos como referencia para ver cómo definir el `SCOPE` y la `KEY`, y compruebe la sintaxis. Estos ejemplos de código utilizan `MY_STAGE` y `MY_TASK` como los nombres de la etapa y la tarea de canalización.

Para obtener más información sobre los tipos de tarea disponibles, consulte [Qué tipos de tareas se encuentran disponibles en vRealize Automation Code Stream](#).

Tabla 3-9. Tareas de activación periódica

Tipo de tarea	Scope	Key	Cómo utilizar SCOPE y KEY en la tarea
Operación de usuario			
	Input	<p>summary: Resumen de la solicitud de operación de usuario</p> <p>description: Descripción de la solicitud de operación de usuario</p> <p>approvers: Lista de direcciones de correo electrónico del aprobador, en las que cada entrada puede ser una variable con una coma, o un punto y coma para correos electrónicos distintos</p> <p>approverGroups: Lista de direcciones de grupos de aprobadores para la plataforma y la identidad</p> <p>sendemail: De manera opcional, se envía una notificación por correo electrónico según solicitud o respuesta cuando se establece en true</p> <p>expirationInDays: Número de días que representa el tiempo de caducidad de la solicitud</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: Cadena hexadecimal de 6 dígitos que representa la solicitud</p> <p>respondedBy: Nombre de la cuenta de la persona que aprobó o rechazó la operación de usuario</p> <p>respondedByEmail: Dirección de correo electrónico de la persona que respondió</p> <p>comments: Comentarios proporcionados durante la respuesta</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>
Condición			
	Input	<p>condition: Condición que se va a evaluar. Cuando la condición se evalúa como true, la tarea se marca como completada, mientras que otras respuestas tienen errores en la tarea</p>	<pre> \${MY_STAGE.MY_TASK.input.condition} </pre>
	Output	<p>result: Resultado tras la evaluación</p>	<pre> \${MY_STAGE.MY_TASK.output.response} </pre>

Tabla 3-10. Tareas de canalización

Tipo de tarea	Scope	Key	Cómo utilizar SCOPE y KEY en la tarea
Canalización			
	Input	<p>name: Nombre de la canalización que se va a ejecutar</p> <p>inputProperties: Propiedades de entrada para transferir a la ejecución anidada</p>	<pre> \${MY_STAGE.MY_TASK.input.name} \${MY_STAGE.MY_TASK.input.inputProperties} # Haga referencia a todas las propiedades \$ {MY_STAGE.MY_TASK.input.inputProperties.input1} # Haga referencia al valor de input1 </pre>
	Output	<p>executionStatus: Estado de la ejecución de la canalización</p> <p>executionIndex: Índice de ejecución de la canalización</p> <p>outputProperties: Propiedades de salida de la ejecución de la canalización</p>	<pre> \${MY_STAGE.MY_TASK.output.executionStatus} \${MY_STAGE.MY_TASK.output.executionIndex} \${MY_STAGE.MY_TASK.output.outputProperties} # Haga referencia a todas las propiedades \$ {MY_STAGE.MY_TASK.output.outputProperties.output1} # Haga referencia al valor de output1 </pre>

Tabla 3-11. Automatizar tareas de integración continua

Tipo de tarea	Scope	Key	Cómo utilizar SCOPE y KEY en la tarea
CI			
	Input	<p>steps: Un conjunto de cadenas, que representan comandos que se ejecutarán</p> <p>export: Variables de entorno que se conservarán después de ejecutar los pasos</p> <p>artifacts: Rutas de acceso de artefactos que se conservarán en la ruta de acceso compartida</p> <p>process: Conjunto de elementos de configuración para procesamiento de JUnit, JaCoCo, Checkstyle, FindBugs</p>	<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} # Haga referencia a la ruta de acceso de la primera configuración </pre>

Tabla 3-11. Automatizar tareas de integración continua (continuación)

Tipo de tarea	Scope	Key	Cómo utilizar SCOPE y KEY en la tarea
	Output	<p>exports: Par clave-valor, que representa las variables de entorno exportadas de la entrada <code>export</code></p> <p>artifacts: Ruta de acceso de los artefactos conservados correctamente</p> <p>processResponse: Conjunto de resultados procesados para la entrada <code>process</code></p>	<pre> \${MY_STAGE.MY_TASK.output.exports} # Haga referencia a todas las exportaciones \$ {MY_STAGE.MY_TASK.output.exports.myvar} # Haga referencia al valor de myvar \${MY_STAGE.MY_TASK.output.artifacts} \$ {MY_STAGE.MY_TASK.output.processResponse} \$ {MY_STAGE.MY_TASK.output.processResponse[0].result} # Resultado de la primera configuración del proceso </pre>
Personalizada			
	Input	<p>name: Nombre de la integración personalizada</p> <p>version: Una versión de la integración personalizada, publicada u obsoleta</p> <p>properties: Propiedades que se enviarán a la integración personalizada</p>	<pre> \${MY_STAGE.MY_TASK.input.name} \${MY_STAGE.MY_TASK.input.version} \${MY_STAGE.MY_TASK.input.properties} # Haga referencia a todas las propiedades \$ {MY_STAGE.MY_TASK.input.properties.property1} # Haga referencia al valor de property1 </pre>
	Output	<p>properties: Propiedades de salida de la respuesta de integración personalizada</p>	<pre> \${MY_STAGE.MY_TASK.output.properties} # Haga referencia a todas las propiedades \$ {MY_STAGE.MY_TASK.output.properties.property1} # Haga referencia al valor de property1 </pre>

Tabla 3-12. Integración de aplicaciones de desarrollo, prueba e implementación

Tipo de tarea	Scope	Key	Cómo utilizar SCOPE y KEY en la tarea
Bamboo			
	Input	<p>plan: Nombre del plan</p> <p>planKey: Clave del plan</p> <p>variables: Variables que se transferirán al plan</p> <p>parameters: Parámetros que se transferirán al plan</p>	<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} # Haga referencia a todos los parámetros \${MY_STAGE.MY_TASK.input.parameters.param1} # Haga referencia al valor de param1 </pre>

Tabla 3-12. Integración de aplicaciones de desarrollo, prueba e implementación (continuación)

Tipo de tarea	Scope	Key	Cómo utilizar SCOPE y KEY en la tarea
	Output	resultUrl: URL de la compilación resultante buildResultKey: Clave de la compilación resultante buildNumber: Número de compilación buildTestSummary: Resumen de las pruebas ejecutadas successfulTestCount: Resultado de la prueba aprobado failedTestCount: Resultado de la prueba erróneo skippedTestCount: Resultado de la prueba omitido artifacts: artefactos desde la compilación	<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} # Haga referencia a todos los resultados \${MY_STAGE.MY_TASK.output.successfulTestCount} # Haga referencia a un recuento de pruebas específico \${MY_STAGE.MY_TASK.output.buildNumber} </pre>
Jenkins			
	Input	job: Nombre del trabajo de Jenkins parameters: Parámetros que se transferirán al trabajo	<pre> \${MY_STAGE.MY_TASK.input.job} \${MY_STAGE.MY_TASK.input.parameters} # Haga referencia a todos los parámetros \${MY_STAGE.MY_TASK.input.parameters.param1} # Haga referencia al valor de un parámetro </pre>
	Output	job: Nombre del trabajo de Jenkins jobId: Identificador del trabajo resultante, por ejemplo, 1234 jobStatus: Estado en Jenkins jobResults: Recopilación de resultados de cobertura de prueba/código jobUrl: URL de la ejecución del trabajo 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} # Haga referencia a todos los resultados \${MY_STAGE.MY_TASK.output.jobResults.junitResponse} # Haga referencia a los resultados de JUnit \${MY_STAGE.MY_TASK.output.jobResults.jacocoResponse} # Haga referencia a los resultados de JaCoCo \${MY_STAGE.MY_TASK.output.jobUrl} </pre>
TFS			
	Input	projectCollection: Recopilación de proyectos desde TFS teamProject: Proyecto seleccionado de la recopilación disponible buildDefinitionId: Identificador de definición de compilación que se va a ejecutar	<pre> \${MY_STAGE.MY_TASK.input.projectCollection} \${MY_STAGE.MY_TASK.input.teamProject} \${MY_STAGE.MY_TASK.input.buildDefinitionId} </pre>

Tabla 3-12. Integración de aplicaciones de desarrollo, prueba e implementación (continuación)

Tipo de tarea	Scope	Key	Cómo utilizar SCOPE y KEY en la tarea
	Output	buildId: Identificador de compilación resultante buildUrl: URL para visitar el resumen de la compilación logUrl: URL para visitar los logs dropLocation: Ubicación de destino de los artefactos si los hubiera	<pre> \${MY_STAGE.MY_TASK.output.buildId} \${MY_STAGE.MY_TASK.output.buildUrl} \${MY_STAGE.MY_TASK.output.logUrl} \${MY_STAGE.MY_TASK.output.dropLocation} </pre>
vRO			
	Input	workflowId: Identificador del flujo de trabajo que se va a ejecutar parameters: Parámetros que se transferirán al flujo de trabajo	<pre> \${MY_STAGE.MY_TASK.input.workflowId} \${MY_STAGE.MY_TASK.input.parameters} </pre>
	Output	workflowExecutionId: Identificador de la ejecución del flujo de trabajo properties: Propiedades de salida de la ejecución del flujo de trabajo	<pre> \${MY_STAGE.MY_TASK.output.workflowExecutionId} \${MY_STAGE.MY_TASK.output.properties} </pre>

Tabla 3-13. Integración de otras aplicaciones a través de una API

Tipo de tarea	Scope	Key	Cómo utilizar SCOPE y KEY en la tarea
REST			
	Input	url: URL que se invocará action: Método de HTTP por usar headers: Encabezados de HTTP por transferir payload: Carga útil de la solicitud fingerprint: Huella digital que debe coincidir si la URL es https allowAllCerts: Establecida en true, permite cualquier certificado si la URL es https	<pre> \${MY_STAGE.MY_TASK.input.url} \${MY_STAGE.MY_TASK.input.action} \${MY_STAGE.MY_TASK.input.headers} \${MY_STAGE.MY_TASK.input.payload} \${MY_STAGE.MY_TASK.input.fingerprint} \${MY_STAGE.MY_TASK.input.allowAllCerts} </pre>

Tabla 3-13. Integración de otras aplicaciones a través de una API (continuación)

Tipo de tarea	Scope	Key	Cómo utilizar SCOPE y KEY en la tarea
	Output	<p>responseCode: Códigos de respuesta HTTP</p> <p>responseHeaders: Encabezado de respuesta HTTP</p> <p>responseBody: Formato de cadena de respuesta recibida</p> <p>responseJson: Respuesta interactiva si el tipo de contenido es aplicación/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> # Haga referencia al encabezado de respuesta "header1"</p> <p><code>\${MY_STAGE.MY_TASK.output.responseBody}</code></p> <p><code>\${MY_STAGE.MY_TASK.output.responseJson}</code> # Haga referencia a la respuesta como JSON</p> <p><code>\${MY_STAGE.MY_TASK.output.responseJson.a.b.c}</code> # Haga referencia a un objeto anidado siguiendo la ruta de acceso de a.b.c JSON en la respuesta</p>
Sondeo			
	Input	<p>url: URL que se invocará</p> <p>headers: Encabezados de HTTP por transferir</p> <p>exitCriteria: Criterios que se deben cumplir para que la tarea se realice correctamente o con errores. Un par clave-valor "correcto" → Expresión, "error" → Expresión</p> <p>pollCount: Número de iteraciones que se realizarán</p> <p>pollIntervalSeconds: Número de segundos de espera entre cada iteración</p> <p>ignoreFailure: Cuando se establece en true, ignora los errores de respuesta intermedios</p> <p>fingerprint: Huella digital que debe coincidir si la URL es https</p> <p>allowAllCerts: Cuando se establece como true, permite cualquier certificado si la URL es 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: Códigos de respuesta HTTP</p> <p>responseBody: Formato de cadena de respuesta recibida</p> <p>responseJson: Respuesta interactiva si el tipo de contenido es aplicación/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>

Tabla 3-14. Ejecución de scripts remotos y definidos por el usuario

Tipo de tarea	Scope	Key	Cómo utilizar SCOPE y KEY en la tarea
PowerShell			
	Input	<p>host: Dirección IP o nombre de host de la máquina</p> <p>username: Nombre de usuario que se utilizará para conectarse</p> <p>password: Contraseña que se utilizará para conectarse</p> <p>useTLS: Intentos de conexión HTTPS</p> <p>trustCert: Cuando se establece en true, confía en certificados autofirmados</p> <p>script: Script que se ejecutará</p> <p>workingDirectory: Ruta de acceso del directorio al que se debe cambiar antes de ejecutar el script</p> <p>environmentVariables: Un par clave-valor de la variable de entorno que se va a establecer</p> <p>arguments: Argumentos para transferir al 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: Contenido del archivo \$SCRIPT _RESPONSE_FILE</p> <p>responseFilePath: Valor de \$SCRIPT _RESPONSE_FILE</p> <p>exitCode: Procesar código de salida</p> <p>logFilePath: Ruta de acceso al archivo que contiene stdout</p> <p>errorFilePath: Ruta de acceso al archivo que contiene 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			

Tabla 3-14. Ejecución de scripts remotos y definidos por el usuario (continuación)

Tipo de tarea	Scope	Key	Cómo utilizar SCOPE y KEY en la tarea
	Input	host: Dirección IP o nombre de host de la máquina username: Nombre de usuario que se utilizará para conectarse password: Contraseña que se utilizará para conectarse (de manera opcional, puede utilizar privateKey) privateKey: instancia de privateKey que se utilizará para conectarse passphrase: Frase de contraseña opcional para desbloquear la instancia de privateKey script: Script que se ejecutará workingDirectory: Ruta de acceso del directorio al que se debe cambiar antes de ejecutar el script environmentVariables: Par clave-valor de la variable de entorno que se va a establecer	<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: Contenido del archivo \$SCRIPT _RESPONSE_FILE responseFilePath: Valor de \$SCRIPT _RESPONSE_FILE exitCode: Procesar código de salida logFilePath: Ruta de acceso al archivo que contiene stdout errorFilePath: Ruta de acceso al archivo que contiene 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>

Cómo utilizar un enlace de variables entre tareas

Este ejemplo muestra cómo utilizar enlaces de variables en las tareas de la canalización.

Tabla 3-15. Formatos de sintaxis de ejemplo

Ejemplo	Sintaxis
Para utilizar un valor de salida de la tarea para las notificaciones de canalización y las propiedades de salida de la canalización	<code>\${<Stage Key>.<Task Key>.output.<Task output key>}</code>
Para hacer referencia al valor de salida de la tarea anterior como entrada para la tarea actual	<code>\${<Previous/Current Stage key>.<Previous task key not in current Task group>.output.<task output key>}</code>

Más información

Para obtener más información sobre cómo enlazar variables, consulte:

- [Cómo utilizar las variables de enlace en canalizaciones de vRealize Automation Code Stream](#)
- [Cómo utilizar enlaces de variables en una tarea de condición para ejecutar o detener una canalización en vRealize Automation Code Stream](#)
- [Qué tipos de tareas se encuentran disponibles en vRealize Automation Code Stream](#)

Cómo enviar notificaciones acerca de la canalización en vRealize Automation Code Stream

Las notificaciones son maneras para comunicarse con los equipos y hacerles saber el estado de las canalizaciones en vRealize Automation Code Stream.

Puede configurar vRealize Automation Code Stream para enviar notificaciones cuando se ejecuta la canalización con base en el estado de la etapa, la tarea o la canalización completas.

- Una notificación se envía por correo electrónico cuando se produce lo siguiente:
 - Finalización, espera, error, cancelación o inicio de la canalización.
 - Finalización, error o inicio de la etapa.
 - Finalización, espera, error o inicio de la tarea.
- Una notificación de ticket crea un ticket y lo asigna a un miembro del equipo cuando se produce lo siguiente:
 - Error o finalización de la canalización.
 - Error de la etapa.
 - Error de la tarea.
- Una notificación de webhook envía una solicitud a otra aplicación cuando se produce lo siguiente:
 - Error, finalización, espera, cancelación o inicio de la canalización.
 - Error, finalización o inicio de la etapa.
 - Error, finalización, espera o inicio de la tarea.

Por ejemplo, puede configurar una notificación de correo electrónico en una tarea de operación de usuario para obtener la aprobación en un punto específico de la canalización. Cuando se ejecuta la canalización, esta tarea envía un correo electrónico a la persona que debe aprobar la tarea. También puede configurar una notificación para crear un ticket de Jira cuando se produce un error en una tarea de canalización. Opcionalmente, puede configurar una notificación de webhook para enviar una solicitud a un canal de Slack sobre el estado de una canalización en función del evento de canalización.

Puede utilizar variables en todos los tipos de notificaciones. Por ejemplo, puede utilizar `${var}` en la URL de una notificación de webhook.

Requisitos previos

- Compruebe que se hayan creado una o varias canalizaciones. Consulte los casos prácticos en [Capítulo 5 Ejemplos de casos prácticos comunes para vRealize Automation Code Stream](#).
- Para enviar notificaciones por correo electrónico, compruebe que pueda acceder a un servidor de correo electrónico que funcione. Para obtener ayuda, consulte al administrador.
- Para crear tickets (como un ticket de Jira), compruebe que exista el endpoint. Consulte [Qué son los endpoints en vRealize Automation Code Stream](#).
- Para enviar una notificación basada en una integración, cree una notificación de webhook. A continuación, compruebe que el webhook se ha agregado y funciona. Puede utilizar notificaciones con aplicaciones como Slack, GitHub o GitLab.

Procedimiento

- 1 Abra una canalización.
- 2 Para crear una notificación sobre el estado general de la canalización o el estado de una tarea o una etapa:

Para crear una notificación basada en...	Lo que debe hacer...
Estado de la canalización	Haga clic en un área en blanco en el lienzo de la canalización.
Estado de una etapa	Haga clic en un área en blanco en una etapa de la canalización.
Estado de una tarea	Haga clic en una tarea en una etapa de la canalización.

- 3 Haga clic en la pestaña **Notificaciones**.
- 4 Haga clic en **Agregar**, seleccione el tipo de notificación y configure los detalles de notificación.
- 5 Para crear una notificación de Slack cuando se realiza correctamente una canalización, cree una notificación de webhook.
 - a Seleccione **Webhook**.
 - b Introduzca la información para configurar la notificación de Slack.
 - c Haga clic en **Guardar**.
 - d Cuando se ejecuta la canalización, el canal de Slack recibe la notificación sobre el estado de la canalización. Por ejemplo, es posible que los usuarios vean lo siguiente en el canal de Slack:

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

- 6 Para crear un ticket de Jira, configure la información del ticket.
 - a Seleccione **Ticket**.
 - b Introduzca la información para configurar la notificación de Jira.
 - c Haga clic en **Guardar**.

Notification

Type ☐ Email ☒ Ticket ☐ Webhook

Event * ☒ On Pipeline Failure ☐ On Pipeline Completion

Jira endpoint * Jira-Notification

Create Ticket

Jira project * YourProject

Issue type * Bug

Assignee * username@yourcompany.com

Summary \$ * Pipeline failed

Description \$ Research and correct

CANCEL SAVE

Resultados

Felicidades. Aprendió que puede crear varios tipos de notificaciones en varias áreas de la canalización en vRealize Automation Code Stream.

Pasos siguientes

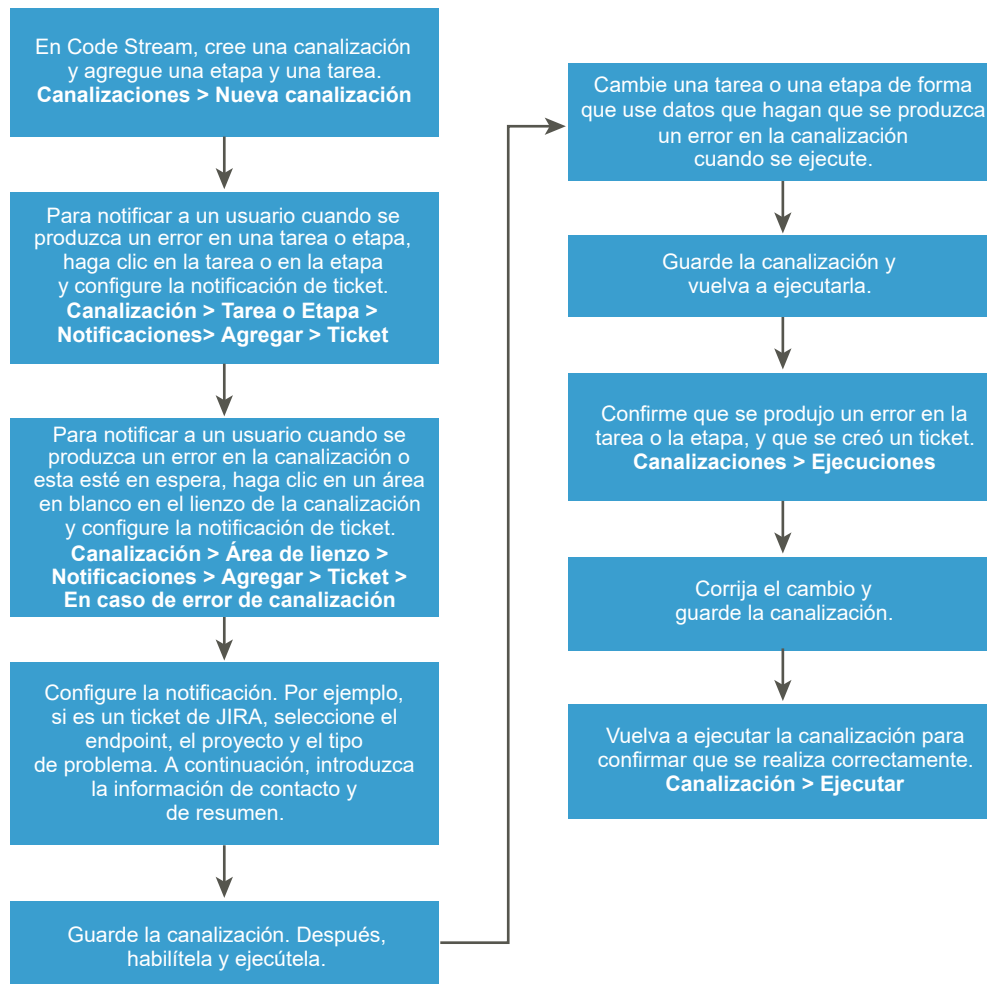
Para acceder a un ejemplo detallado de cómo crear una notificación, consulte [Cómo crear un ticket de JIRA en vRealize Automation Code Stream cuando se produce un error en una tarea de canalización](#).

Cómo crear un ticket de JIRA en vRealize Automation Code Stream cuando se produce un error en una tarea de canalización

Si se produce un error en una etapa o una tarea de la canalización, puede hacer que vRealize Automation Code Stream cree un ticket de Jira. Puede asignar el ticket a la persona que necesita resolver el problema. También se puede crear un ticket cuando la canalización esté en espera o cuando se lleve a cabo correctamente.

Puede agregar y configurar notificaciones en una tarea, una etapa o una canalización. vRealize Automation Code Stream crea el ticket en función del estado de la tarea, la etapa o la canalización donde esté agregada la notificación. Por ejemplo, si un endpoint no está disponible, puede hacer que vRealize Automation Code Stream cree un ticket de JIRA de la tarea que genera un error porque no se puede conectar al endpoint.

También puede crear notificaciones cuando la canalización se realice correctamente. Por ejemplo, puede informar al equipo de control de calidad de si una canalización se realizó correctamente para que, de este modo, pueda comprobar la compilación y ejecutar otra canalización de prueba. O puede informar al equipo de rendimiento para que pueda calibrar el rendimiento de la canalización y prepararse para una actualización de la realización de copias intermedias o la producción.



En este ejemplo, se crea un ticket de JIRA cuando se produce un error en una tarea de canalización.

Requisitos previos

- Confirme que tiene una cuenta de JIRA válida y que puede iniciar sesión en la instancia de JIRA.

- Confirme que existe un endpoint de JIRA en funcionamiento.

Procedimiento

- 1 En la canalización, haga clic en una tarea.
- 2 En el panel de configuración de tareas, haga clic en **Notificaciones**.
- 3 Haga clic en **Agregar** y configure la información del ticket.
 - a Haga clic en **Ticket**.
 - b Seleccione el endpoint de JIRA.
 - c Introduzca el proyecto de JIRA y el tipo de problema.
 - d Introduzca la dirección de correo electrónico de la persona que recibirá el ticket.
 - e Introduzca un resumen y una descripción del ticket y, a continuación, haga clic en **Guardar**.

Notification

Type ☐ Email ☒ Ticket ☐ Webhook

Event ☒ On Task Failure

Jira endpoint

Create Ticket

Jira project

Issue type

Assignee

Summary

Description

- 4 Guarde la canalización y, a continuación, habilítela y ejecútela.
- 5 Compruebe el ticket.
 - a Cambie la información de la tarea de forma que incluya datos que hagan que se produzca un error en la tarea.
 - b Guarde la canalización y vuelva a ejecutarla.
 - c Haga clic en **Ejecuciones** y compruebe que se produjo un error en la canalización.

- d En la ejecución, compruebe que vRealize Automation Code Stream creó el ticket y lo envió.
- e Cambie la información de la tarea para corregirla a como era antes, ejecute la canalización de nuevo y asegúrese de que se realiza correctamente.

Resultados

Enhorabuena. Hizo que vRealize Automation Code Stream creara un ticket de JIRA al producirse un error en la tarea de canalización y lo asignara a la persona que debía resolverlo.

Pasos siguientes

Siga agregando notificaciones para avisar al equipo de las canalizaciones.

Cómo revertir una implementación en vRealize Automation Code Stream

La reversión se configura como una canalización con tareas que devuelven la implementación a un estado estable anterior después de un error en una canalización de implementación. Es necesario asociar la canalización de reversión a las tareas o etapas que desee revertir en caso de que se produzca un error.

Los motivos de la reversión pueden variar según su función.

- Como ingeniero de publicaciones, desea que vRealize Automation Code Stream compruebe que el proceso de publicación se complete correctamente para saber si debe continuar con la publicación o revertirla. Los posibles errores incluyen un error en la tarea, un rechazo en las operaciones de usuario o un umbral de métricas excedido.
- Como propietario de un entorno, desea volver a implementar una versión anterior para poder revertir rápidamente el entorno a un estado correcto conocido.
- Como propietario de un entorno, desea admitir la reversión de una implementación azul-verde para poder minimizar el tiempo de inactividad de las versiones con errores.

Cuando se utiliza una plantilla inteligente para crear una canalización de entrega continua con la opción de reversión activada, la reversión se agrega automáticamente a las tareas en la canalización. En este caso práctico, utilizará la plantilla inteligente para definir la reversión de una implementación de aplicación en un clúster de Kubernetes mediante el modelo de implementación de actualización gradual. La plantilla inteligente crea una canalización de implementación y una o varias canalizaciones de reversión.

- En la canalización de implementación, se requiere la reversión si se produce un error en las tareas Actualizar implementación o Verificar implementación.
- En la canalización de reversión, se actualiza la implementación con una imagen antigua.

También puede crear manualmente una canalización de reversión con una plantilla en blanco. Antes de crear una canalización de reversión, se recomienda planificar el flujo de reversión. Para obtener más información de fondo sobre la reversión, consulte [Planificar una reversión en vRealize Automation Code Stream](#).

Requisitos previos

- Compruebe si es un miembro de un proyecto en vRealize Automation Code Stream. Si no lo es, solicite al administrador de vRealize Automation Code Stream que lo agregue como miembro de un proyecto. Consulte [Cómo agregar un proyecto en vRealize Automation Code Stream](#).
- Configure los clústeres de Kubernetes donde la canalización implementará la aplicación. Configure un clúster de desarrollo y un clúster de producción.
- Cree los endpoints de producción y desarrollo de Kubernetes que implementan la imagen de la aplicación en los clústeres de Kubernetes.
- Compruebe que tiene una configuración de registro de Docker.
- Compruebe que tiene un archivo YAML de Kubernetes para aplicarlo a la implementación.
- Familiarícese con la plantilla inteligente de distribución continua (Continuous Delivery, CD). Consulte [Planificar una compilación nativa de CD en vRealize Automation Code Stream antes de usar la plantilla inteligente](#).

Procedimiento

- 1 Haga clic en **Canalizaciones > Nueva canalización > Plantilla inteligente > Entrega continua**.
- 2 Introduzca la información en la plantilla inteligente.
 - a Seleccione un proyecto.
 - b Introduzca un nombre para la canalización, como **RollingUpgrade-Example**.
 - c Seleccione los entornos de la aplicación. Para agregar la reversión a la implementación, debe seleccionar **Prod**.
 - d Haga clic en **Seleccionar**, elija un archivo YAML de Kubernetes y haga clic en **Procesar**.
La plantilla inteligente muestra los servicios y los entornos de implementación disponibles.
 - e Seleccione el servicio que utilizará la canalización para la implementación.
 - f Seleccione los endpoints del clúster para los entornos Dev y Prod.
 - g Como origen de imagen, seleccione **Entrada de tiempo de ejecución de canalización**.
 - h Como modelo de implementación, seleccione **Actualización gradual**.

- i Haga clic en **Revertir**.
- j Proporcione la URL de comprobación de estado.

Plantilla inteligente: Entrega continua

Requisitos previos de e... Kubernetes Registro de Docker

Proyecto *

Nombre de canalización *

Entorno * ☒ Desarrollo ☒ Producción

Archivos YAML de Kube... SELECCIONAR PROCESAR
Archivos procesados: Kubernetesbgreen.yaml

Seleccionar servicio

Nombre de la implementación	Servicio	Espacio de nombres	Imagen
codestream-demo	codestream-demo	bgreen1	symphony-lango-beta2/frog.io/codestream-demo

1 servicios

Implementación

Entorno	Endpoint del clúster	Espacio de nombres
Desarrollo	Kubernetes-Endpoint-Staging	bgreen-912272
Producción	Kubernetes-Endpoint-Staging	bgreen

Origen de imagen * ☐ Activador de Docker ☒ Entrada de tiempo de ejecución de canalización

Modelo de implementac... * ☐ Canary ☒ Actualización gradual ☐ azul-verde

Revertir ☒

URL de comprobación d... *

CREAR CANCELAR

- 3 Para crear la canalización llamada RollbackUpgrade-Example, haga clic en **Crear**.

La canalización RollbackUpgrade-Example aparece con el icono de reversión en tareas de las etapas de desarrollo y producción que se pueden revertir.

RollbackUpgrade-Example Deshabilitado

Área de trabajo | Entrada | **Modelo** | Salida

Development

Create Namesp...
Kubernetes

Create Secret
Kubernetes

Create Sever
Kubernetes

+ Tarea paralela

Production

Create Se_trans...
Kubernetes

Update Deploy...
Kubernetes

Verify Deploy...
Kubernetes

+ Tarea paralela

+ Etapa

Tarea: **Create Secret** | Notificaciones | Revertir

Nombre de la tarea *

Tipo *

Continuar en caso ... ☐

Ejecutar tarea ☒ Siempre ☐ Con condición

Propiedades de la tarea de Kubernetes

Clúster de Kuberne...

Tiempo de espera ...

Acción * ☐ Obtener ☒ Crear ☐ Aplicar ☐ Eliminar ☐ Revertir

Continuar en confil... ☒

Tipo de origen * ☐ Control de origen ☒ Definición local

Definición de YAM... LEER DEL ARCHIVO

```

1 apiVersion: v1
2 - data:
3   .dockercfg: eyJ0cmluaw69ueS10WVh5bnY1ZXRhWiSod1afjke12odeFrng2hch2hsh
4   2FdsH5xwde2dfh5ss13h8df5S453hFdsfhF3as15ghh1fs31Sh3F1ds5h5s3df15
5   h315sdF15h53108fds45h04fsd54h56h4In19
6 Kind: Secret
7 - metadata:
8   name: jfrog-beta2
9   namespace: bgreen-549930
10 type: kubernetes.io/dockercfg

```

Parámetros de salida

status k8SRollbackTaskFields endpoint response yamls operation config

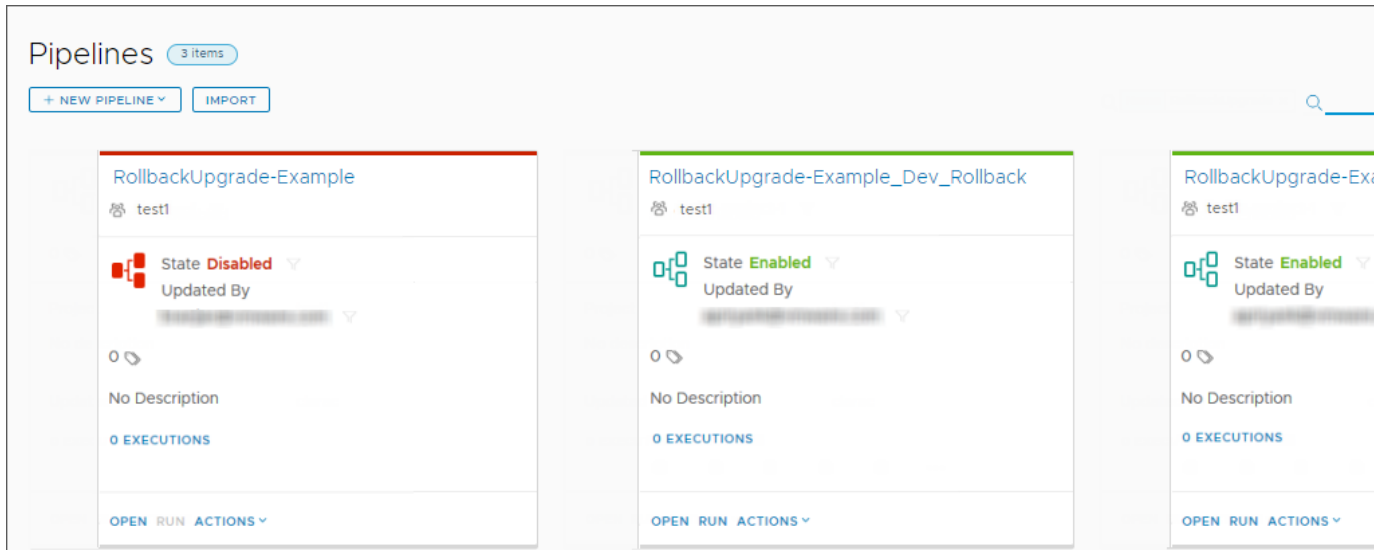
GUARDAR EJECUTAR CERRAR Guardado por última vez: hace 4 minutos

- 4 Cierre la canalización.

En la página Canalizaciones, verá la canalización que ha creado y una nueva canalización para cada etapa de esta.

- RollingUpgrade-Example. vRealize Automation Code Stream desactiva la canalización que creó de forma predeterminada, lo que garantiza que la revise antes de ejecutarla.

- **RollingUpgrade-Example_Dev_Rollback.** Esta canalización de desarrollo de reversión se invoca en caso de errores en las tareas de la etapa de desarrollo, como las tareas de crear servicio, crear secreto, crear implementación y verificar implementación. La canalización de desarrollo de reversión está habilitada de forma predeterminada para garantizar la reversión de las tareas de desarrollo.
- **RollingUpgrade-Example_Prod_Rollback.** Esta canalización de producción de reversión se invoca en caso de errores en las tareas de la etapa de producción, como la fase 1 de implementación, la fase 1 de verificación, la fase de implementación de lanzamiento, la fase de finalización de lanzamiento y la fase de verificación de lanzamiento. La canalización de producción de reversión está habilitada de forma predeterminada para garantizar la reversión de las tareas de producción.



- 5 Habilite y ejecute la canalización que ha creado.

Cuando inicie la ejecución, se le pedirán los parámetros de entrada. Proporcione la imagen y la etiqueta del endpoint en el repositorio de Docker que está utilizando.

- 6 En la página Ejecuciones, seleccione **Acciones > Ver ejecución** para ver la ejecución de la canalización.

La canalización se muestra como **RUNNING** y pasa por las tareas de la etapa de desarrollo. Si la canalización no logra ejecutar una tarea durante la etapa de desarrollo, la canalización denominada **RollingUpgrade-Example_Dev_Rollback** se activa para revertir la implementación, y el estado de la canalización cambia a **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: [redacted]

Executed by: [redacted]

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


Después de la reversión, la página Ejecuciones muestra dos ejecuciones de la canalización RollingUpgrade-Example.

- La canalización que creó y que se revirtió muestra **ROLLBACK_COMPLETED**.
- La canalización de desarrollo de reversión que se activó para realizar la reversión muestra **COMPLETED**.


Executions

604 items


[+ NEW EXECUTION](#)🔍




RollbackUpgrade-Example_Dev...#1

1  Rollback for RollbackUpgrade-Example#1


COMPLETED

Stages: 


By  on 01/11/2019 1:36 PM

Execution Completed.


Comments:Triggered to rollback Development.Create Deployment of RollbackUpgrade-E




RollbackUpgrade-Example#1

0 

ROLLBACK_COMPLETED

Stages: 

By  on 01/11/2019 1:24 PM

Create Deployment ROLLBACK_COMPLETED

Resultados

Enhorabuena. Definió correctamente una canalización con reversión y observó a vRealize Automation Code Stream revertir la canalización al punto de error.

Planificar la compilación, la integración y la distribución nativas del código en vRealize Automation Code Stream

Antes de hacer que vRealize Automation Code Stream compile, integre y distribuya el código mediante la capacidad nativa que crea una canalización de CICD, CI o CD, planifique la compilación nativa. A continuación, puede crear la canalización mediante una de las plantillas inteligentes o agregando etapas y tareas manualmente.

Hemos incluido varios ejemplos que muestran cómo planificar la integración continua y la compilación de distribuciones. Estos planes describen los requisitos previos que debe cumplir, y brinda información general que lo ayudará a prepararse para utilizar la capacidad nativa de compilación de forma efectiva para compilar las canalizaciones.

Este capítulo incluye los siguientes temas:

- Planificar una compilación nativa de CICD en vRealize Automation Code Streamantes de usar la plantilla inteligente
- Planificar una compilación nativa de CI en vRealize Automation Code Streamantes de usar la plantilla inteligente
- Planificar una compilación nativa de CD en vRealize Automation Code Streamantes de usar la plantilla inteligente
- Planificar una compilación nativa de CICD en vRealize Automation Code Stream antes de agregar tareas manualmente
- Planificar una reversión en vRealize Automation Code Stream

Planificar una compilación nativa de CICD en vRealize Automation Code Streamantes de usar la plantilla inteligente

Para crear una canalización de integración continua y entrega continua (Continuous Integration And Continuous Delivery, CICD) en vRealize Automation Code Stream, puede utilizar la plantilla inteligente de CICD. Para planificar una compilación nativa de CICD, recopile la información que necesita para rellenar la plantilla inteligente antes de utilizarla para crear la canalización en este plan de ejemplo.

Después de introducir la información en la plantilla inteligente y guardarla, la plantilla crea una canalización que incluye etapas y tareas. También indica dónde implementar la imagen en función de los tipos de entorno que seleccione, como desarrollo y producción. La canalización publicará la imagen de Docker y realizará las acciones necesarias para ejecutarla. Después de ejecutar la canalización, puede supervisar las tendencias en las ejecuciones de la canalización.

Para crear una canalización de CICD, es necesario planificar las etapas de integración continua (Continuous Integration, CI) y de entrega continua (Continuous Delivery, CD) de esa canalización.

Cuando una canalización incluye una imagen de Docker Hub, debe asegurarse de que la imagen tenga una cURL integrada antes de ejecutar la canalización. Cuando se ejecuta la canalización, vRealize Automation Code Stream descarga un archivo binario que utiliza cURL para ejecutar los comandos.

Planificación de la etapa de integración continua (Continuous Integration, CI)

Para planificar la etapa de integración continua de la canalización, configurará los requisitos internos y externos y determinará la información que se debe introducir en la parte de integración continua de la plantilla inteligente. A continuación se resume la planificación.

Endpoints y repositorios que necesitará:

- Un repositorio de código fuente de Git donde los desarrolladores comprueban el código. vRealize Automation Code Stream extrae el código más reciente a la canalización cuando los desarrolladores confirman cambios.
- Un endpoint de Git para el repositorio donde reside el código fuente del desarrollador.
- Un endpoint de Docker para el host de compilación de Docker que ejecutará los comandos de compilación dentro de un contenedor.
- Un endpoint de Kubernetes para que vRealize Automation Code Stream pueda implementar la imagen en un clúster de Kubernetes.
- Una imagen de Builder que crea el contenedor donde se ejecutan las pruebas de integración continua.
- Un endpoint del registro de imágenes del que el host de compilación de Docker extrae la imagen de Builder.

Necesitará acceso a un proyecto. El proyecto agrupa todo el trabajo, incluidos la canalización, los endpoints y los paneles de control. Compruebe si es un miembro de un proyecto en vRealize Automation Code Stream. Si no lo es, solicite al administrador de vRealize Automation Code Stream que lo agregue como miembro de un proyecto. Consulte [Cómo agregar un proyecto en vRealize Automation Code Stream](#).

Necesitará un webhook de Git que permita a vRealize Automation Code Stream utilizar el activador de Git para activar la canalización cuando los desarrolladores confirmen cambios de código. Consulte [Cómo usar el activador de Git en vRealize Automation Code Stream para ejecutar una canalización](#).

Sus conjuntos de herramientas de compilación:

- Su tipo de compilación, como Maven.
- Todas las herramientas de compilación posteriores al proceso que utilice, como JUnit, JaCoCo, Checkstyle y FindBugs.

Su herramienta de publicación:

- Una herramienta como Docker que implementará el contenedor de compilación.
- Una etiqueta de imagen, que puede ser el identificador de confirmación o el número de compilación.

Su área de trabajo de compilación:

- Un host de compilación de Docker, que es el endpoint de Docker.
- Un registro de imágenes. La parte de integración continua de la canalización extrae la imagen del endpoint del registro seleccionado. El contenedor ejecuta las tareas de integración continua e implementa la imagen. Si el registro necesita credenciales, primero debe crear un endpoint del registro de imágenes y, seguidamente, seleccionarlo para que el host pueda extraer la imagen del registro.
- URL de la imagen de Builder que crea el contenedor donde se ejecutan las tareas de integración continua.

Planificación de la etapa de entrega continua

Para planificar la etapa de entrega continua de la canalización, configurará los requisitos internos y externos y determinará la información que se debe introducir en la parte de entrega continua de la plantilla inteligente.

Endpoints que necesitará:

- Un endpoint de Kubernetes para que vRealize Automation Code Stream pueda implementar la imagen en un clúster de Kubernetes.

Tipos de entorno y archivos:

- Todos los tipos de entorno donde vRealize Automation Code Stream implementará la aplicación, como desarrollo y producción. La plantilla inteligente crea las etapas y las tareas en la canalización en función de los tipos de entorno que seleccione.

Tabla 4-1. Etapas de canalización que la plantilla inteligente de CICD crea

Contenido de la canalización	Qué función tiene
Etapas de compilación y publicación	El código se compila y comprueba, la imagen de Builder se crea y se publica en el host de Docker.
Etapas de desarrollo	Se usa un clúster de Amazon Web Services (AWS) de desarrollo para crear e implementar la imagen. En esta etapa se puede crear un espacio de nombres en el clúster y crear una clave secreta.
Etapas de producción	Se usa una versión de producción de VMware Cloud PKS para implementar la imagen en un clúster de Kubernetes de producción.

- Un archivo YAML de Kubernetes que se selecciona en la parte de entrega continua de la plantilla inteligente de CICD.

Para aplicar el archivo, haga clic en **Seleccionar**, elija el archivo YAML de Kubernetes y haga clic en **Procesar**. La plantilla inteligente muestra los servicios y los entornos de implementación disponibles. Debe seleccionar un servicio, el endpoint del clúster y la estrategia de implementación. Por ejemplo, para usar el modelo de implementación Canary, seleccione **Canary** e introduzca el porcentaje de la fase de la implementación.

Plantilla inteligente: CI/CD

Paso 2 de 2

Entorno ☒ Desarrollo ☒ Producción

Archivos YAML de Kube... **SELECCIONAR** **PROCESAR**

Archivos procesados: codestream.yaml

Seleccionar servicio

Nombre de la implementación	Servicio	Espacio de nombres	Imagen
codestream-demo	codestream-demo	bgreen1	3

1 servicios

Implementación

Entorno	Endpoint del clúster	Espacio de nombres
Desarrollo	1030Endpoint-Kubernetes:開発環境にあるKubernetesのEndpoint	bgreen1-570394
Producción	1030Endpoint-Kubernetes:開発環境にあるKubernetesのEndpoint	bgreen1

Modelo de implementac... ☒ Canary ☐ Actualización gradual ☐ azul-verde

Fase 1 20 %

Revertir ☐

URL de comprobación d... /health-check.json

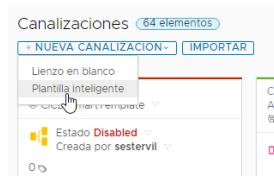
CREAR **ATRÁS** **CANCELAR**

Para ver un ejemplo de uso de la plantilla inteligente para crear una canalización de una implementación azul-verde, consulte [Cómo implementar la aplicación en vRealize Automation Code Stream en una implementación azul-verde](#).

Cómo crear la canalización de CICD mediante la plantilla inteligente

Después de recopilar toda la información y configurar lo que necesita, le indicamos cómo crear una canalización a partir de la plantilla inteligente de CICD.

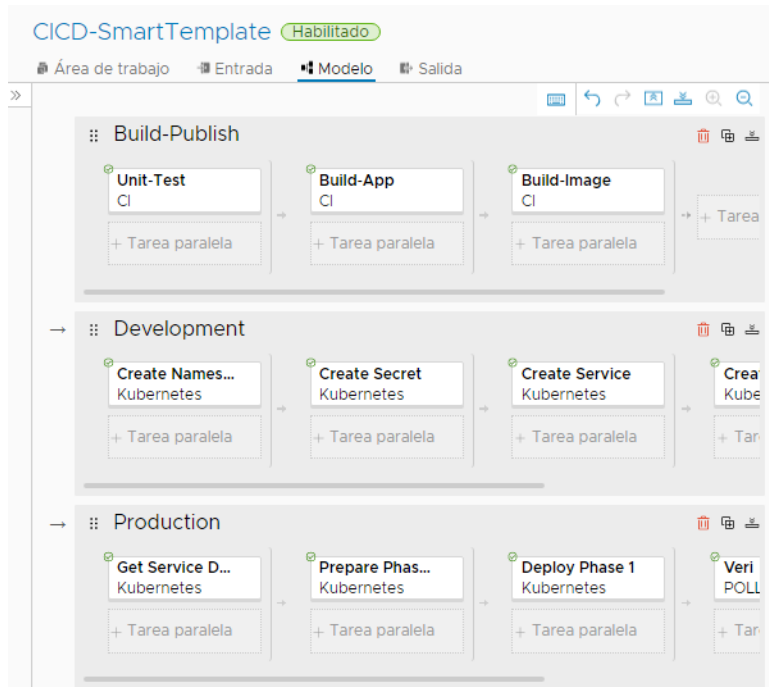
En Canalizaciones, debe seleccionar **Nueva canalización > Plantillas inteligentes**.



Luego, debe seleccionar la plantilla inteligente de CI/CD.



Debe rellenar la plantilla y guardar la canalización con las etapas que se hayan creado. Si necesita rematar con algún cambio, puede editar la canalización y guardarla.



A continuación, deberá habilitar la canalización y ejecutarla. Tras ejecutarla, estas son algunas de las cosas que puede buscar:

- Compruebe que la canalización se haya completado correctamente. Haga clic en **Ejecuciones** y busque la canalización. Si se produjeron errores, corríjalos y vuelva a ejecutarla.
- Compruebe que el webhook de Git esté funcionando correctamente. La pestaña **Actividad** de Git muestra los eventos. Haga clic en **Activadores > Git > Actividad**.
- Observe el panel de control de la canalización y examine las tendencias. Haga clic en **Paneles de control** y busque el panel de control de la canalización. También puede crear un panel de control personalizado para informar sobre KPI adicionales.

Para obtener un ejemplo detallado, consulte [Cómo integrar de forma continua código de un repositorio de GitLab o GitHub en la canalización en vRealize Automation Code Stream](#).

Planificar una compilación nativa de CI en vRealize Automation Code Streamantes de usar la plantilla inteligente

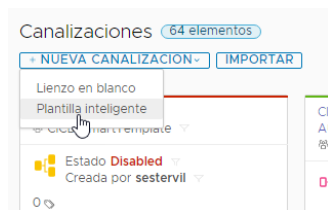
Para crear una canalización de integración continua (Continuous Integration, CI) en VMware Code Stream, puede utilizar la plantilla inteligente de CI. Para planificar una compilación nativa de CI, recopile la información que necesita para rellenar la plantilla inteligente antes de utilizarla para crear la canalización en este plan de ejemplo.

Al rellenar la plantilla inteligente, esta crea una canalización de CI en el repositorio y realiza las acciones necesarias para ejecutarla. Después de ejecutar la canalización, puede supervisar las tendencias en las ejecuciones de la canalización.

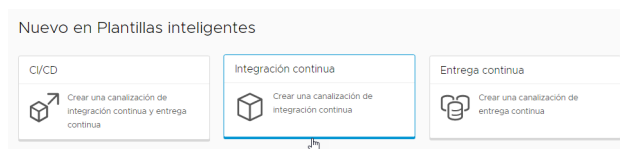
Para planificar la compilación antes de usar la plantilla de inteligente de CI, recopile la información para la compilación y siga la sección CI de [Planificar una compilación nativa de CI en vRealize Automation Code Streamantes de usar la plantilla inteligente](#).

Después de recopilar toda la información y configurar lo que necesita, siga los siguientes pasos para crear una canalización desde la plantilla inteligente de CI.

En Canalizaciones, seleccione **Plantillas inteligentes**.



Seleccione la plantilla inteligente de CI.



Rellene la plantilla y haga clic en **Crear** para guardar la canalización con las etapas que crea.

Puede editar la canalización para realizar cualquier cambio final que necesite. A continuación, puede habilitar la canalización y ejecutarla. Después de que se ejecute la canalización, estos son algunos de los aspectos que debe observar:

- Compruebe que la canalización se completó correctamente. Haga clic en **Ejecuciones** y busque la canalización. Si se produjo algún error, corríjalo y vuelva a ejecutar la canalización.
- Compruebe que el webhook de Git funciona correctamente. La pestaña **Actividad** de Git muestra los eventos. Haga clic en **Activadores > Git > Actividad**.

- Observe el panel de control de la canalización y examine las tendencias. Haga clic en **Paneles de control** y busque el panel de control de la canalización. También puede crear un panel de control personalizado para informar sobre indicadores clave de rendimiento adicionales.

Para obtener un ejemplo detallado, consulte [Cómo integrar de forma continua código de un repositorio de GitLab o GitHub en la canalización en vRealize Automation Code Stream](#).

Planificar una compilación nativa de CD en vRealize Automation Code Streamantes de usar la plantilla inteligente

Para crear una canalización de distribución continua (Continuous Delivery, CD) en vRealize Automation Code Stream, puede usar la plantilla inteligente de CD. Para planificar una compilación nativa de CD, recopile la información que necesita para rellenar la plantilla inteligente antes de utilizarla para crear la canalización en este plan de ejemplo.

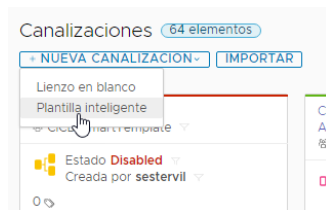
Al rellenar la plantilla inteligente, esta crea una canalización de CD en el repositorio y realiza las acciones necesarias para ejecutarla. Después de ejecutar la canalización, puede supervisar las tendencias en las ejecuciones de la canalización.

Para planificar la compilación antes de utilizar la plantilla inteligente de CD, deberá hacer lo siguiente:

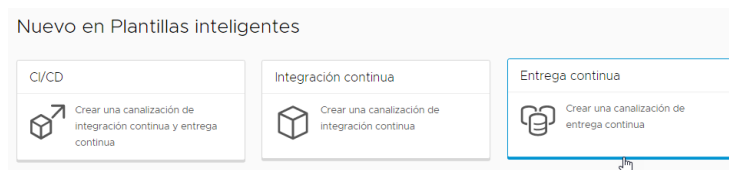
- Recopile la información para la compilación y, a continuación, siga la sección de CD de [Planificar una compilación nativa de CI/CD en vRealize Automation Code Streamantes de usar la plantilla inteligente](#).
- Agregue un endpoint de Kubernetes en el que vRealize Automation Code Stream implementará el contenedor.
- Identifique un proyecto que agrupe todo su trabajo, incluidos la canalización, los endpoints y los paneles de control.

Después de recopilar toda la información y configurar lo que necesite, siga este procedimiento para crear una canalización a partir de la plantilla inteligente de CD.

En Canalizaciones, seleccione **Plantillas inteligentes**.



Seleccione la plantilla inteligente de CD.



Re llene la plantilla, introduzca un nombre para la canalización y, a continuación, haga clic en **Crear** para guardar la canalización con las etapas que cree.

Puede editar la canalización para realizar cualquier cambio final que necesite. A continuación, puede habilitar la canalización y ejecutarla. Después de que se ejecute la canalización, estos son algunos de los aspectos que debe observar:

- Compruebe que la canalización se haya completado correctamente. Haga clic en **Ejecuciones** y busque la canalización. Si se produjeron errores, corríjalos y vuelva a ejecutarla.
- Compruebe que el webhook de Git esté funcionando correctamente. La pestaña **Actividad** de Git muestra los eventos. Haga clic en **Activadores > Git > Actividad**.
- Observe el panel de control de la canalización y examine las tendencias. Haga clic en **Paneles de control** y busque el panel de control de la canalización. También puede crear un panel de control personalizado para informar sobre KPI adicionales.

Para obtener un ejemplo detallado, consulte [Cómo integrar de forma continua código de un repositorio de GitLab o GitHub en la canalización en vRealize Automation Code Stream](#).

Planificar una compilación nativa de CI/CD en vRealize Automation Code Stream antes de agregar tareas manualmente

Para crear una canalización de integración continua y distribución continua (Continuous Integration and Continuous Delivery, CI/CD) en vRealize Automation Code Stream, puede agregar manualmente etapas y tareas. Para planificar la compilación nativa de CI/CD, recopile la información que necesite y, a continuación, cree una canalización y agréguele manualmente etapas y tareas.

Necesitará planificar las etapas de integración continua (Continuous Integration, CI) y distribución continua (Continuous Delivery, CD) de la canalización. Después de crear la canalización y ejecutarla, puede supervisar las tendencias en las ejecuciones de la canalización.

Para planificar las etapas de CI y CD de la canalización, compruebe que se cumplan todos los requisitos antes de crear la canalización.

Planificación de los requisitos internos y externos

Para crear una canalización a partir de este plan de ejemplo, debe utilizar un host de Docker, un repositorio de Git, Maven y varias herramientas de compilación de posprocesamiento.

Endpoints y repositorios que necesitará:

- Un repositorio de código fuente de Git donde los desarrolladores comprueban el código. vRealize Automation Code Stream extrae el código más reciente a la canalización cuando los desarrolladores confirman cambios.
- Un endpoint de Docker para el host de compilación de Docker que ejecutará los comandos de compilación dentro de un contenedor.

- Un endpoint de Kubernetes para que vRealize Automation Code Stream pueda implementar la imagen en un clúster de Kubernetes.
- Una imagen de Builder que crea el contenedor donde se ejecutan las pruebas de integración continua.
- Un endpoint del registro de imágenes del que el host de compilación de Docker extrae la imagen de Builder.

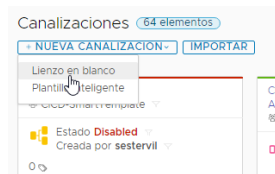
Necesitará acceso a un proyecto. El proyecto agrupa todo el trabajo, incluidos la canalización, los endpoints y los paneles de control. Compruebe si es un miembro de un proyecto en vRealize Automation Code Stream. Si no lo es, solicite al administrador de vRealize Automation Code Stream que lo agregue como miembro de un proyecto. Consulte [Cómo agregar un proyecto en vRealize Automation Code Stream](#).

Necesitará un webhook de Git que permita a vRealize Automation Code Stream utilizar el activador de Git para activar la canalización cuando los desarrolladores confirmen cambios de código. Consulte [Cómo usar el activador de Git en vRealize Automation Code Stream para ejecutar una canalización](#).

Cómo se crea la canalización de CI/CD y se configura el área de trabajo

Deberá crear la canalización y, a continuación, configurar el área de trabajo, los parámetros de entrada de la canalización y las tareas.

Para crear la canalización, haga clic en **Canalizaciones > Nueva canalización > Lienzo en blanco**.



En la pestaña Área de trabajo, introduzca la información de integración continua:

- Incluya el host de compilación de Docker.
- Introduzca la dirección URL de la imagen de Builder.
- Seleccione el endpoint del registro de imágenes para que la canalización pueda extraer la imagen de este. El contenedor ejecuta las tareas de CI e implementa la imagen. Si el registro necesita credenciales, primero debe crear el endpoint del registro de imágenes y, a continuación, seleccionarlo para que el host pueda extraer la imagen del registro.
- Agregue los artefactos que se deben almacenar en caché. Para que una compilación se realice correctamente, los artefactos, como los directorios, se descargan como dependencias. La memoria caché es la ubicación en la que residen estos artefactos. Por ejemplo, los artefactos dependientes pueden incluir el directorio `.m2` para Maven y el directorio `node_modules` para Node.js. Estos directorios se almacenan en caché a través de ejecuciones de canalización para ahorrar tiempo durante las compilaciones.

Proporcione detalles sobre el contenedor y el host para ejecutar tareas de integración continua.

Host * --Seleccionar endpoint de host--

Dirección URL de la i... * Introducir nombre/URL de imagen

Registro de imágenes --Seleccionar endpoint de registro de imágenes--

Directorio de trabajo

Memoria caché

Límite de CPU *

Límite de memoria *

Clon de Git ☐

Si esta canalización se vincula a Git a través de un webhook, la canalización se activará en los eventos de Git. Para las tareas de CI, el repositorio de Git vinculado (detalles de los parámetros de webhook de Git) se clona automáticamente en el área de trabajo.

En la pestaña Entrada, configure los parámetros de entrada de la canalización.

- Si la canalización utilizará parámetros de entrada de un evento activador de Git, Gerrit o Docker, seleccione el tipo de activador para los parámetros de inserción automática. Los eventos pueden incluir Cambiar asunto para Gerrit o Git, o el nombre de propietario del evento para Docker. Si la canalización no utilizará ningún parámetro de entrada transmitido desde el evento, deje Parámetros de inserción automática establecidos en **Ninguno**.
- Para aplicar un valor y una descripción a un parámetro de entrada de la canalización, haga clic en los tres puntos verticales y, a continuación, en **Editar**. El valor que introduzca se utilizará como entrada en tareas, etapas o notificaciones.
- Para agregar un parámetro de entrada de la canalización, haga clic en **Agregar**. Por ejemplo, puede agregar `approvers` para mostrar un valor predeterminado para cada ejecución, pero puede anularlo para mostrar un aprobador diferente en tiempo de ejecución.
- Para agregar o quitar un parámetro insertado, haga clic en **Agregar o quitar parámetro insertado**. Por ejemplo, elimine cualquier parámetro que no se utilice para reducir el desorden en la página de resultados y muestre solamente los parámetros de entrada que se están utilizando.

Parámetros de entrada

Parámetros de inser... ☐ Gerrit ☒ Git ☐ Docker ☐ Ninguno

AGREGAR AGREGAR O ELIMINAR PARÁMETROS INSERTADOS

Destacado	Nombre	Valor	Descripción
☆	GIT_BRANCH_NAME		
☆	GIT_CHANGE_SUBJECT		
☆	GIT_COMMIT_ID		
☆	GIT_EVENT_DESCRIPTION		
☆	GIT_EVENT_OWNER_NAME		
☆	GIT_EVENT_TIMESTAMP		
☆	GIT_REPO_NAME		
☆	GIT_SERVER_URL		

Configure la canalización para probar el código:

- Agregue y configure una tarea de CI.
- Incluya los pasos para ejecutar `mvn test` en el código.

- Ejecute herramientas de compilación de posprocesamiento, como JUnit, JaCoCo, FindBugs y Checkstyle para identificar cualquier problema después de que se ejecute la tarea.

The screenshot shows the configuration for a task named 'Unit-Test'. The task type is 'CI'. The steps section contains a terminal window with the following commands:

```
1 cd demo-project
2 mvn test
```

Below the steps, there is a section for 'Artefakte b...' (Artifacts) with a 'Verarbeiten' (Process) button. A table lists the tools and their paths:

Tool	Path
JUnit	/demo-project
JaCoCo	/demo-project
Checkstyle	/demo-project
FindBugs	/demo-project

At the bottom, there are links for 'status' and 'exports'.

Configure la canalización para compilar el código:

- Agregue y configure una tarea de CI.
- Incluya los pasos para ejecutar `mvn clean install` en el código.
- Incluya la ubicación y el nombre del archivo JAR para que conserve el artefacto.

The screenshot shows the configuration for a task named 'Build-App'. The task type is 'CI'. The steps section contains a terminal window with the following commands:

```
1 cd demo-project
2 mvn clean install -DskipTests
```


Below the steps, there is a section for 'Preservar ar...' (Preserve artifacts) with a 'Procesar' (Process) button. A table lists the tools and their paths:


Tool	Path
JUnit	/demo-project
JaCoCo	/demo-project
Checkstyle	/demo-project
FindBugs	/demo-project


At the bottom, there are links for 'status' and 'exports'.

Configure la canalización para publicar la imagen en el host de Docker:

- Agregue y configure una tarea de CI.
- Agregue los pasos que confirmarán, exportarán, compilarán e insertarán la imagen.
- Agregue la clave de exportación de `IMAGE` para la siguiente tarea que se va a consumir.


Tarea: **Build-Image** Notificaciones Revertir **VALIDAR TAREA** 

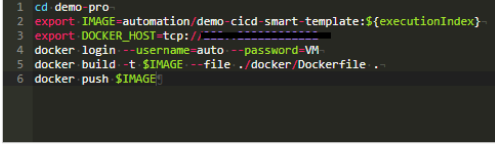
Nombre de la tar...  * Build-Image



Tipo * CI 

Continuar en cas... ☐


Ejecutar tarea ☒ Siempre ☐ Con condición

Integración continua 

Pasos * 

Preservar artefa...  

Exportar IMAGE

Procesar **AGREGAR** 

Parámetros de salida

status **exports**

Guarde la canalización después de configurar el área de trabajo, los parámetros de entrada, las tareas de prueba y las tareas de compilación.

Cómo habilitar y ejecutar la canalización

Después de configurar la canalización con fases y tareas, puede guardar y habilitar la canalización.

A continuación, espere a que la canalización se ejecute y finalice y compruebe que se completó correctamente. Si se produjeron errores, corrijalos y vuelva a ejecutarla.

Después de que la canalización se complete correctamente, estos son algunos de los aspectos que debe confirmar:

- Examine la ejecución de la canalización y vea los resultados de los pasos de las tareas.
- En el área de trabajo de la ejecución de la canalización, busque los detalles del contenedor y el repositorio de Git clonado.
- En el área de trabajo, consulte los resultados de las herramientas de posprocesamiento y compruebe si hay errores, cobertura de código y problemas de estilo.
- Confirme que el artefacto se conserva. Confirme también que la imagen se exportó con el nombre y el valor IMAGE.
- Vaya al repositorio de Docker y compruebe que la canalización publicó el contenedor.

Para ver un ejemplo detallado que muestre cómo vRealize Automation Code Stream integra continuamente el código, consulte [Cómo integrar de forma continua código de un repositorio de GitLab o GitHub en la canalización en vRealize Automation Code Stream](#).

Planificar una reversión en vRealize Automation Code Stream

En caso de que se produzca un error de ejecución de la canalización, se puede utilizar la reversión para que el entorno vuelva a un estado estable anterior. Para usar la reversión, debe planificar un flujo de reversión y comprender cómo implementarlo.

Un flujo de reversión indica los pasos necesarios para revertir un error en la implementación. El flujo adopta la forma de una canalización de reversión donde se incluyen una o varias tareas secuenciales, las cuales varían según el tipo de implementación que se ejecutó y en la que se produjo un error. Por ejemplo, la implementación y la reversión de una aplicación tradicional son diferentes de la implementación y la reversión de una aplicación de contenedor.

Para regresar a un buen estado de implementación, una canalización de reversión normalmente incluye tareas para:

- Limpiar los estados o los entornos
- Ejecutar un script especificado por el usuario para revertir los cambios
- Implementar una revisión anterior de una implementación

Para agregar la reversión a una canalización de implementación existente, debe adjuntar la canalización de reversión a las tareas o etapas de la canalización de implementación que desee revertir antes de ejecutar la canalización de implementación.

Cómo configurar la reversión

Para configurar la reversión en la implementación, debe realizar lo siguiente:

- Cree una canalización de implementación.
- Identifique los puntos de error potenciales en la canalización de implementación que activarán la reversión para poder adjuntar la canalización de reversión. Por ejemplo, puede adjuntar la canalización de reversión a una condición o un tipo de tarea de sondeo en la canalización de implementación para comprobar si una tarea anterior se completó correctamente. Para obtener información sobre las tareas de condición, consulte [Cómo utilizar enlaces de variables en una tarea de condición para ejecutar o detener una canalización en vRealize Automation Code Stream](#).
- Determine el alcance del error que activará la canalización de reversión, por ejemplo, un error de tarea o de etapa. También puede adjuntar la reversión a una etapa.
- Decida qué tarea o tareas de reversión se deben ejecutar en caso de que se produzca un error. Creará la canalización de reversión con esas tareas.

Puede crear manualmente una canalización de reversión o vRealize Automation Code Stream puede crear una por usted.

- Con un lienzo en blanco, puede crear manualmente una canalización de reversión que siga un flujo en paralelo a una canalización de implementación existente. A continuación, debe adjuntar la canalización de reversión a una o varias tareas de la canalización de implementación que activa la reversión al producirse un error.
- Con una plantilla inteligente, puede configurar una canalización de implementación con la acción de reversión. A continuación, vRealize Automation Code Stream crea automáticamente una o varias canalizaciones de reversión predeterminadas con tareas predefinidas que revierten la implementación en caso de error.

Para obtener un ejemplo detallado sobre la forma de configurar una canalización de entrega continua con la reversión mediante una plantilla inteligente, consulte [Cómo revertir una implementación en vRealize Automation Code Stream](#).

Qué sucede si la canalización de implementación tiene varias tareas o etapas con reversión

Si tiene varias tareas o tareas y etapas con la reversión agregada, tenga en cuenta que la secuencia de reversión varía.

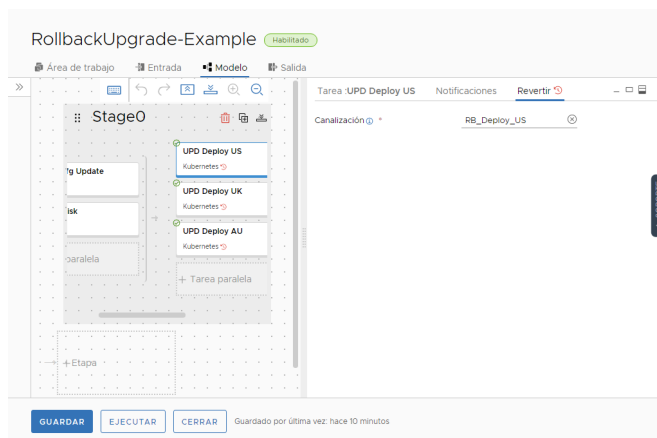
Tabla 4-2. Determinación de la secuencia de reversión

Si agrega la reversión a...	Cuándo se realiza la reversión...
Tareas paralelas	Si se produce un error en una de las tareas paralelas, la reversión de la tarea se realiza después de que todas las tareas paralelas se hayan completado correctamente o con errores. La reversión no se realiza inmediatamente después de que se produce un error en la tarea.
Tanto la tarea dentro de una etapa como la etapa	Si se produce un error en una tarea, se ejecuta la reversión de la tarea. Si la tarea se encuentra en un grupo de tareas paralelas, la reversión de la tarea se ejecuta después de que todas las tareas paralelas se hayan completado correctamente o con errores. La reversión de la etapa se ejecuta después de que la tarea se completa correctamente o con errores.

Considere una canalización que tiene:

- Una etapa de producción con reversión
- Un grupo de tareas paralelas, cada tarea con su propia reversión

La tarea con el nombre **UPD Deploy US** tiene la canalización de reversión **RB_Deploy_US**. Si se produce un error en **UPD Deploy US**, la reversión sigue el flujo definido en la canalización **RB_Deploy_US**.



Si se produce un error en **UPD Deploy US**, la canalización **RB_Deploy_US** se ejecuta después de que **UPD Deploy UK** y **UPD Deploy AU** se hayan completado correctamente o con errores. La reversión no se realiza inmediatamente después de que se produce un error en **UPD Deploy US**. Debido a que la etapa de producción también tiene reversión, después de que se ejecuta la canalización **RB_Deploy_US**, se ejecuta la canalización de reversión de la etapa.

Ejemplos de casos prácticos comunes para vRealize Automation Code Stream

5

Utilice vRealize Automation Code Stream para modelar y admitir el ciclo de vida de publicación de DevOps, y para probar y publicar continuamente las aplicaciones.

Ya ha configurado todo lo que necesita para poder usar vRealize Automation Code Stream. Consulte [Capítulo 2 Configuración de vRealize Automation Code Stream para modelar el proceso de lanzamiento](#).

Ahora puede crear las canalizaciones que automatizan la compilación y la prueba del código del desarrollador antes de publicarlo en producción. Puede hacer que vRealize Automation Code Stream implemente aplicaciones tradicionales o basadas en contenedores.

Tabla 5-1. Usar vRealize Automation Code Stream en el ciclo de vida de DevOps

Uso de funciones...	Ejemplos de lo que puede hacer...
Usar la capacidad de compilación nativa en vRealize Automation Code Stream	<p>Cree las canalizaciones de CICD, CI y CD que integrarán, aislarán y distribuirán continuamente el código.</p> <ul style="list-style-type: none">■ Utilice una plantilla inteligente para crear una canalización.■ Agregue manualmente etapas y tareas a una canalización.
Publicar las aplicaciones y automatizar las publicaciones	<p>Integre y publique las aplicaciones de varias maneras.</p> <ul style="list-style-type: none">■ Integre continuamente el código de GitHub o GitLab en la canalización.■ Automatice la implementación de la aplicación mediante un blueprint de YAML.■ Automatice la implementación de la aplicación en un clúster de Kubernetes.■ Publique la aplicación en una implementación azul-verde.■ Integre vRealize Automation Code Stream con sus propias herramientas de compilación, prueba e implementación.■ Utilice una API de REST que integre vRealize Automation Code Stream con otras aplicaciones.
Realizar un seguimiento de tendencias, métricas y KPI	<p>Cree paneles de control personalizados y obtenga información sobre el rendimiento de sus canalizaciones.</p>
Solucionar problemas	<p>Cuando se produce un error en una ejecución de canalización, haga que vRealize Automation Code Stream cree un ticket de JIRA.</p>

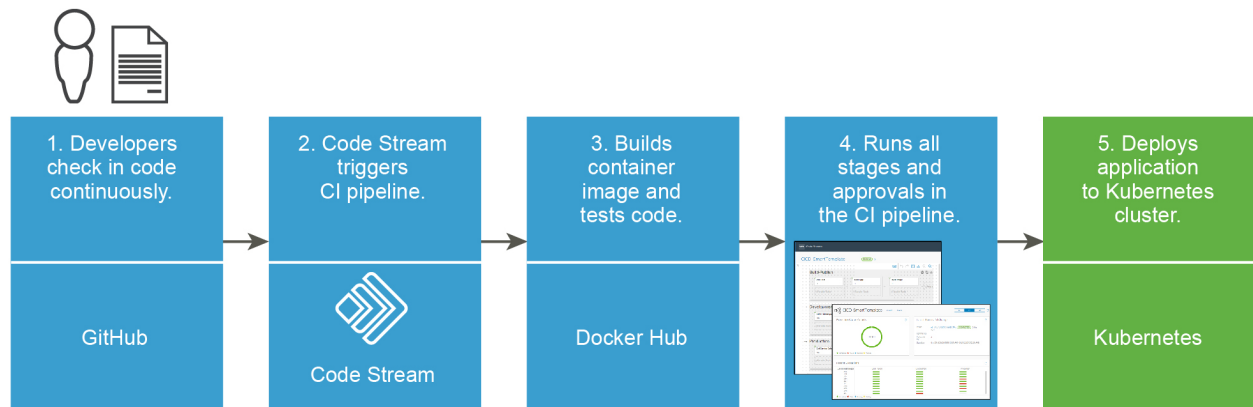
Este capítulo incluye los siguientes temas:

- [Cómo integrar de forma continua código de un repositorio de GitLab o GitHub en la canalización en vRealize Automation Code Stream](#)

- Cómo automatizar la publicación de una aplicación implementada a partir de un blueprint de YAML en vRealize Automation Code Stream
- Cómo automatizar el lanzamiento de una aplicación en vRealize Automation Code Stream en un clúster de Kubernetes
- Cómo implementar la aplicación en vRealize Automation Code Stream en una implementación azul-verde
- Cómo se integran las herramientas personalizadas de compilación, pruebas e implementación con vRealize Automation Code Stream
- Cómo usar una API de REST para integrar vRealize Automation Code Stream con otras aplicaciones

Cómo integrar de forma continua código de un repositorio de GitLab o GitHub en la canalización en vRealize Automation Code Stream

Como desarrollador, deberá integrar de forma continua el código de un repositorio de GitHub o GitLab Enterprise. Cada vez que los desarrolladores actualizan el código y confirman los cambios en el repositorio, vRealize Automation Code Stream puede estar atento a esos cambios y activar la canalización.



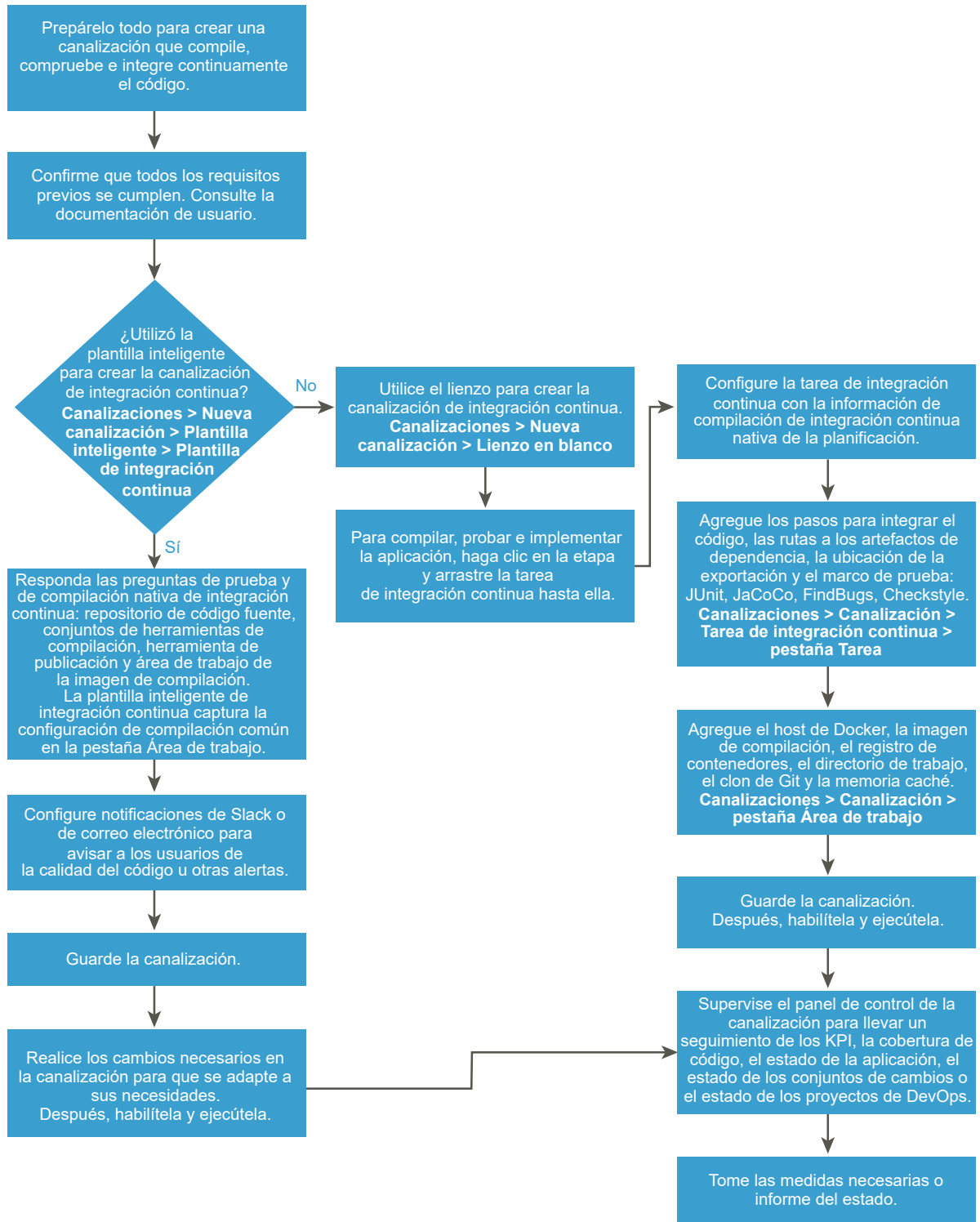
Para que vRealize Automation Code Stream active la canalización cuando se produzcan cambios en el código, se utilizará el activador de Git. vRealize Automation Code Stream activará la canalización cada vez que confirme cambios en el código.

Para compilar el código, se utilizará un host de Docker. Como herramientas de marco de pruebas se utilizará JUnit y JaCoCo, las cuales ejecutan pruebas de unidad y cobertura de código, y que se incluirán en la canalización.

A continuación, utilizará la plantilla inteligente de CI para crear una canalización de CI que compile, pruebe e implemente el código en el clúster de Kubernetes del equipo del proyecto en AWS. Usará una memoria caché para almacenar los artefactos de dependencia de código de la tarea de CI, lo que ahorrará tiempo en las compilaciones de código.

En la tarea de canalización que compila y prueba el código, se incluirán varios pasos de integración continua. Estos pasos residirán en el mismo directorio de trabajo en el que se clona el código fuente cuando se activa la canalización.

Para implementar el código en el clúster de Kubernetes, deberá utilizar una tarea de Kubernetes en la canalización. Habilitará y ejecutará la canalización. A continuación, deberá realizar un cambio en el código del repositorio y observar el activador de la canalización. Posteriormente, deberá supervisar las tendencias de la canalización después de que se ejecute la canalización y presentar informes sobre estas, y deberá utilizar los paneles de control.



En este ejemplo, utilizará la plantilla inteligente de CI para crear una canalización de CI, de modo que pueda integrar el código en la canalización de manera continua.

De manera opcional, puede crear manualmente la canalización, y agregarle etapas y tareas. Para obtener más información sobre la planificación de una compilación de integración continua y la creación manual de la canalización, consulte [Planificar una compilación nativa de CICD en vRealize Automation Code Stream antes de agregar tareas manualmente](#).

Requisitos previos

- Planifique una compilación de integración continua. Consulte [Planificar una compilación nativa de CI en vRealize Automation Code Stream antes de usar la plantilla inteligente](#) y la sección sobre la planificación de la etapa de integración continua (Continuous Integration, CI).
- Asegúrese de que dispone de un repositorio de código fuente de GitLab. Para obtener ayuda, consulte al administrador de vRealize Automation Code Stream.
- Agregue un endpoint de Git. Para ver un ejemplo, consulte [Cómo usar el activador de Git en vRealize Automation Code Stream para ejecutar una canalización](#).
- Para que vRealize Automation Code Stream esté atento a los cambios en el repositorio de GitHub o GitLab y active una canalización cuando se produzcan, agregue un webhook. Para ver un ejemplo, consulte [Cómo usar el activador de Git en vRealize Automation Code Stream para ejecutar una canalización](#).
- Agregue un endpoint de host de Docker, el cual crea un contenedor para la tarea de CI que varias tareas de CI pueden usar. Para obtener más información sobre los endpoints, consulte [Qué son los endpoints en vRealize Automation Code Stream](#).
- Obtenga la URL de la imagen, el host de compilación y la URL de la imagen de compilación. Para obtener ayuda, consulte al administrador de vRealize Automation Code Stream.
- Asegúrese de que utiliza JUnit y JaCoCo como las herramientas de marco de pruebas.
- Configure una instancia externa para la compilación de CI: Jenkins, TFS o Bamboo. El complemento de Kubernetes implementará el código. Para obtener ayuda, consulte al administrador de vRealize Automation Code Stream.

Procedimiento

- 1 Asegúrese de cumplir los requisitos previos.
- 2 Para crear la canalización mediante la plantilla inteligente, abra la plantilla inteligente de CI y rellene el formulario.
 - a Haga clic en **Canalizaciones > Nueva canalización > Plantilla inteligente > Integración continua**.
 - b Responda las preguntas de la plantilla sobre el repositorio de código fuente, los conjuntos de herramientas de compilación, la herramienta de publicación y el área de trabajo de la imagen de compilación.
 - c Agregue notificaciones de Slack o de correo electrónico para el equipo.
 - d Para que la plantilla inteligente cree la canalización, haga clic en **Crear**.

- e Para realizar cambios adicionales en la canalización, haga clic en **Editar**, realice los cambios y haga clic en **Guardar**.
 - f Habilite la canalización y ejecútela.
- 3 Para crear la canalización de forma manual, agregue etapas y tareas al lienzo, y prepare la información de compilación nativa de CI para configurar la tarea de integración continua.
- a Haga clic en **Canalizaciones > Nueva canalización > Lienzo en blanco**.
 - b Haga clic en la etapa y, a continuación, arrastre las diversas tareas de CI desde el panel de navegación hasta la etapa.
 - c Para configurar la tarea de CI, haga clic en ella y, a continuación, haga clic en la pestaña **Tarea**.
 - d Agregue los pasos que integran el código de forma continua.
 - e Incluya las rutas de los artefactos de dependencia.
 - f Agregue la ubicación de exportación.
 - g Agregue las herramientas de marco de pruebas que utilizará.
 - h Agregue el host de Docker y la imagen de compilación.
 - i Agregue el registro de contenedor, el directorio de trabajo y la memoria caché.
 - j Guarde la canalización y, a continuación, habilítela.
- 4 Realice un cambio en el código del repositorio de GitHub o GitLab.
- El activador Git activará la canalización, la cual comenzará a ejecutarse.
- 5 Para comprobar que el cambio de código activó la canalización, haga clic en **Activadores > Git > Actividad**.

- 6 Para ver la ejecución de la canalización, haga clic en **Ejecuciones** y compruebe que los pasos crearon y exportaron la imagen de compilación.

The screenshot displays the vRealize Automation Code Stream interface. On the left is a navigation sidebar with options: Dashboards, Executions, User Operations, Pipelines, Manage (with sub-options: Endpoints, Variables, Triggers), and Triggers (with sub-options: Gerrit, Git). The main panel shows the execution details for 'CICD-SmartTemplate #51', which is in a 'COMPLETED' state. A progress bar at the top indicates the execution flow: 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 successful Docker login and build steps. The Exports section shows the 'IMAGE' variable with the value 'automation/cicd-smart-template:51'.

- 7 Para supervisar el panel de control de la canalización a fin de realizar un seguimiento de los KPI y las tendencias, haga clic en **Paneles de control > Paneles de control de canalizaciones**.

Resultados

Enhorabuena. Ha creado una canalización que integra continuamente el código de un repositorio de GitHub o GitLab en la canalización e implementa la imagen de compilación.

Pasos siguientes

Para obtener más información, consulte [Más recursos para desarrolladores y administradores de vRealize Automation Code Stream](#).

Cómo automatizar la publicación de una aplicación implementada a partir de un blueprint de YAML en vRealize Automation Code Stream

Como desarrollador, necesita una canalización que recupere un blueprint de automatización desde una instancia de GitHub local cada vez que confirme un cambio. La canalización es necesaria para implementar una aplicación de WordPress para Amazon Web Services (AWS) EC2 o un centro de datos. vRealize Automation Code Stream llama al blueprint desde la canalización y automatiza la integración continua y distribución continua (Continuous Integration and Continuous Delivery, CICD) de ese blueprint para implementar la aplicación.

Para crear y activar la canalización, necesitará un blueprint.

Para el valor de **Origen de blueprint** en la tarea del blueprint, puede seleccionar cualquiera de las siguientes opciones:

- **Blueprints de Cloud Assembly** como el control de origen. En este caso, no necesita un repositorio de GitLab o GitHub.
- **Control de origen** si utiliza GitLab o GitHub para el control de origen. En este caso, debe tener un webhook de Git y activar la canalización a través del webhook.

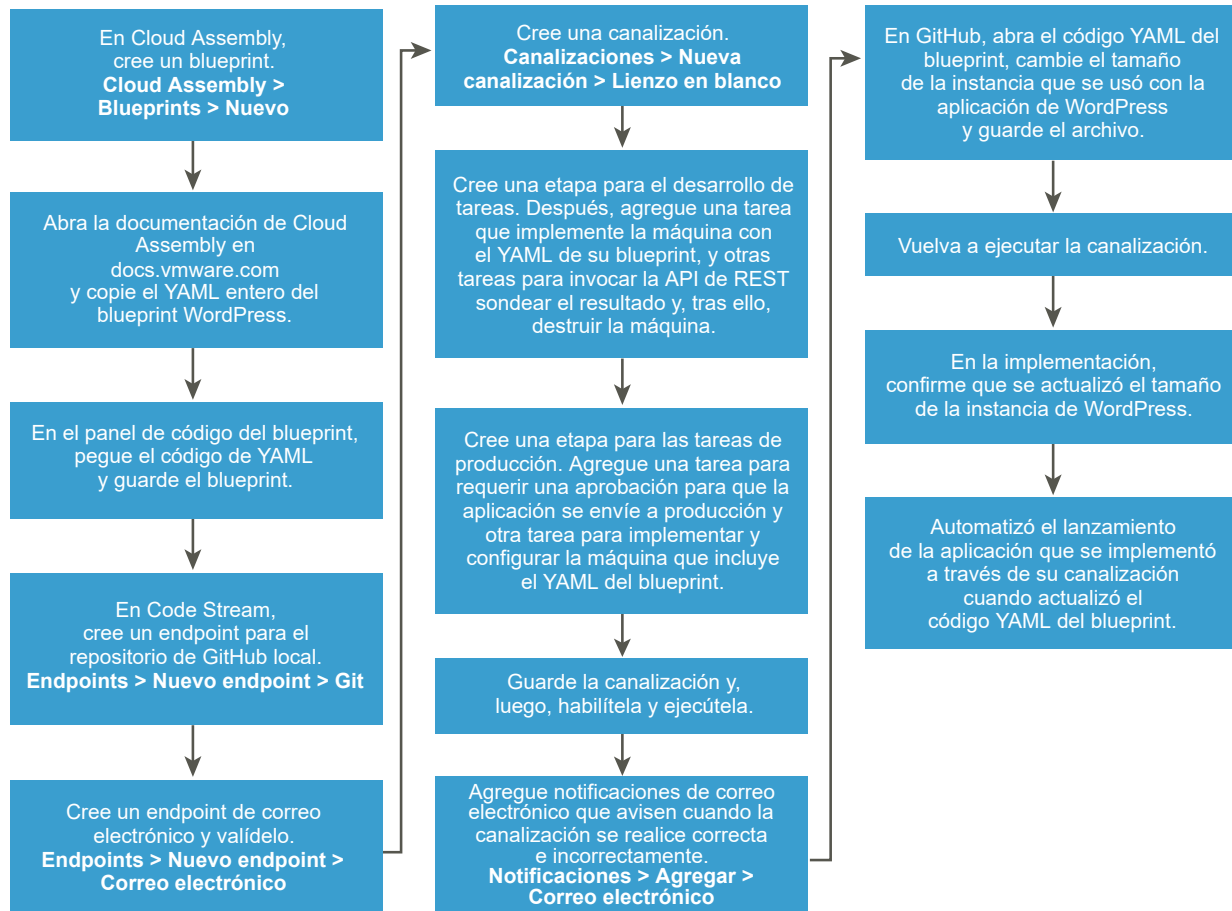
Si tiene un blueprint de YAML en su repositorio de GitHub y desea utilizar ese blueprint en la canalización, debe hacer lo siguiente:

- 1 En vRealize Automation Cloud Assembly, inserte el blueprint en el repositorio de GitHub.
- 2 En vRealize Automation Code Stream, cree un endpoint de Git. A continuación, cree un webhook de Git que use el endpoint de Git y la canalización.
- 3 Para activar la canalización, actualice cualquier archivo del repositorio de GitHub y confirme el cambio.

Si no tiene un blueprint de YAML en el repositorio de GitHub y desea utilizar un blueprint del control de origen, utilice este procedimiento para obtener información sobre cómo hacerlo. Le muestra cómo crear un blueprint para una aplicación de WordPress y activarlo desde un repositorio de GitHub local. Siempre que se realiza un cambio en el blueprint de YAML, la canalización se activa y automatiza la versión de la aplicación.

- En vRealize Automation Cloud Assembly, debe agregar una cuenta de nube, una zona de nube y crear el blueprint.
- En vRealize Automation Code Stream, debe agregar un endpoint para el repositorio de GitHub local que aloja el blueprint. A continuación, debe agregar el blueprint a la canalización.

Este ejemplo de caso práctico muestra cómo utilizar un blueprint desde un repositorio de GitHub local.

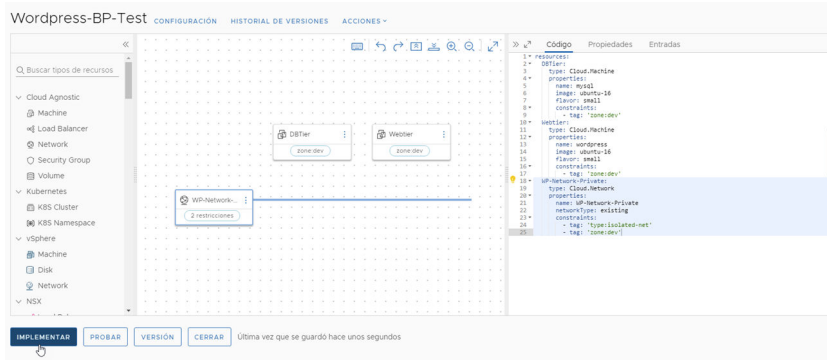


Requisitos previos

- Agregue una cuenta de nube y una zona de nube en la infraestructura de vRealize Automation Cloud Assembly. Consulte la documentación de vRealize Automation Cloud Assembly.
- Para crear el blueprint con el procedimiento que se explica a continuación, copie el código de YAML de WordPress en el portapapeles. En la documentación de vRealize Automation Cloud Assembly, consulte el código YAML del blueprint en el caso de uso de WordPress.
- Agregue el código de YAML de la aplicación de WordPress a su instancia de GitHub.
- Agregue un webhook para el activador de Git de manera que la canalización pueda extraer el código de YAML cada vez que confirme cambios en él. En vRealize Automation Code Stream, haga clic en **Activadores > Git > Webhooks para Git**.
- Para trabajar con una tarea de blueprint, debe tener cualquiera de las funciones de vRealize Automation Cloud Assembly.

Procedimiento

- 1 En vRealize Automation Cloud Assembly, siga los siguientes pasos.
 - a Haga clic en **Blueprints** y, a continuación, cree un blueprint y una implementación para la aplicación de WordPress.
 - b Pegue en el blueprint el código de YAML de WordPress que copió en el portapapeles e impleméntelo.



- 2 En vRealize Automation Code Stream, cree endpoints.
 - a Cree un endpoint de Git para el repositorio de GitHub local en el que resida el archivo de YAML.
 - b Agregue un endpoint de correo electrónico para notificar a los usuarios sobre el estado de la canalización cuando se ejecuta.

Nuevo endpoint

Proyecto * Codestream @

Tipo * Email

Nombre * Valor

Descripción

Marcar como restrin... ☐ no restringido

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

[CREAR VARIABLE](#)

[CREAR](#) [VALIDAR](#) [CANCELAR](#)

- 3 Cree una canalización y agregue notificaciones para informar de si la canalización se ejecutó correctamente o con errores.

Notification

Type

Event

Email server ⓘ *

Send Email

To ⓘ \$ *

Subject \$ *

Body ⓘ \$ *

☒ Email ☐ Ticket ☐ Webhook

☒ On Pipeline Completion ☐ On Pipeline Waiting ☐ On Pipeline Failure
☐ On Pipeline Cancellation ☐ On Pipeline Started

--Select Email server-- ▾

Email IDs of recipients

Email Subject

1

CANCEL

SAVE

- 4 Agregue una etapa para el desarrollo y agregue tareas de blueprint, REST y POLL.
 - a Agregue una tarea de blueprint que implemente la máquina y configúrela para utilizar el YAML del blueprint para la aplicación de 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 Agregue una tarea de REST que invoque la API de REST y ejecute una prueba en la máquina que implementó.
 - c Agregue una tarea de POLL que establezca la API de REST de sondeo y realice un sondeo para obtener el resultado de la prueba en la máquina.
 - d Agregue una tarea de blueprint que destruya la máquina para liberar recursos.

- 5 Agregue una etapa para la producción e incluya tareas de autorización e implementación.
 - a Agregue una tarea de operación de usuario para solicitar autorización para transferir la aplicación de WordPress a producción.
 - b Agregue una tarea de blueprint para implementar la máquina y configurarla con el YAML del blueprint para la aplicación de WordPress.

Al seleccionar **Crear**, el nombre de la implementación debe ser único. Si deja el nombre en blanco, vRealize Automation Code Stream le asignará un nombre aleatorio único.

Esto es lo que debe saber si selecciona **Reversión** en su propio caso práctico: si escoge la acción **Reversión** e introduce un valor para **Versión de reversión**, la versión debe tener el formato **n-x**. Por ejemplo, **n-1**, **n-2**, **n-3**, etc. Si crea y actualiza la implementación en cualquier ubicación que no sea vRealize Automation Code Stream, no se permitirá la reversión.

Al iniciar sesión en vRealize Automation Code Stream, obtiene un token de usuario, que es válido durante 30 minutos. En canalizaciones de larga ejecución, cuando la tarea anterior a la de blueprint tarda 30 minutos o más en ejecutarse, el token de usuario caduca. Como resultado, se produce un error en la tarea de blueprint.

Para asegurarse de que la canalización se pueda ejecutar durante más de 30 minutos, puede introducir un token de API opcional. Cuando vRealize Automation Code Stream invoca al blueprint, el token de API persiste y la tarea de blueprint sigue usando el token de API.

Cuando se utiliza el token de API como una variable, está cifrado. De lo contrario, se utilizará como texto sin formato.

The screenshot shows the configuration page for a task named 'Deploy BP'. The interface includes tabs for 'Task: Deploy BP', 'Notifications', and 'Rollback'. A 'VALIDATE TASK' button is in the top right. The configuration fields are as follows:

- Task name:** Deploy BP
- Type:** Blueprint (dropdown)
- Continue on failure:** ☐
- Execute task:** ☒ Always ☐ On condition
- Deployment Task:**
 - Action:** ☒ Create ☐ Update ☐ Delete ☐ Rollback
 - API token:** API token (text field)
 - Deployment Name:** Enter deployment name (text field)
 - Blueprint source:** ☒ Cloud Assembly Blueprints ☐ Source Control
 - Blueprint:** --Select Blueprint-- (dropdown)
 - Version:** --Select Blueprint Version-- (dropdown)
- Output Parameters:** status (button)

A vertical 'SUPPORT' button is visible on the right side of the form.

- 6 Ejecute la canalización.

- 7 En GitHub, modifique el tipo de la instancia del servidor de WordPress de `small` a `medium`.

Cuando se confirman cambios, la canalización se activa. Extrae el código actualizado del repositorio de GitHub y compila la aplicación.

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

- 8 Vuelva a ejecutar la canalización y compruebe que finalizó correctamente y que cambió el tipo de la instancia de WordPress de pequeña a mediana.

Resultados

Enhorabuena. Se automatizó la publicación de la aplicación implementada a partir de un blueprint de YAML.

Pasos siguientes

Para obtener más información sobre cómo puede usar vRealize Automation Code Stream, consulte [Capítulo 5 Ejemplos de casos prácticos comunes para vRealize Automation Code Stream](#).

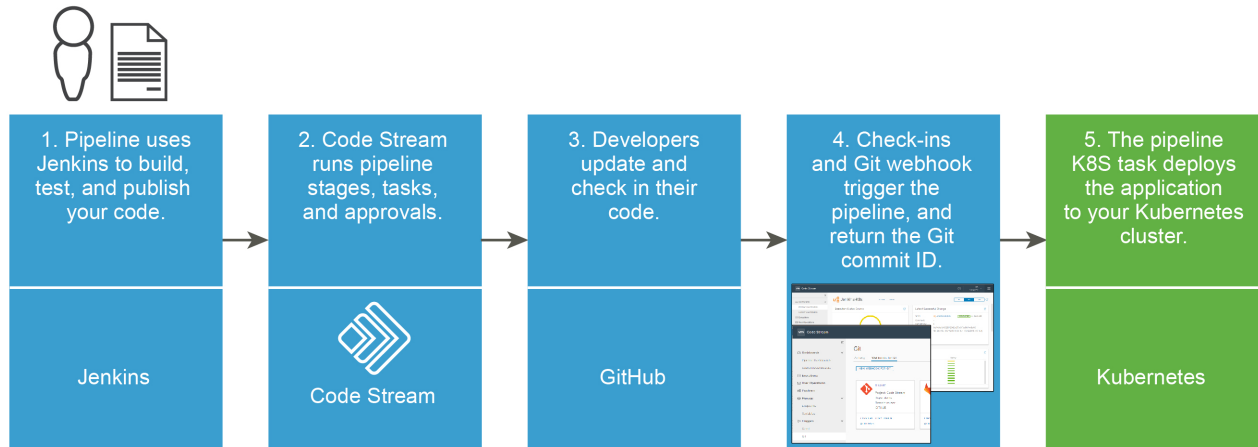
Para obtener más referencias, consulte [Más recursos para desarrolladores y administradores de vRealize Automation Code Stream](#).

Cómo automatizar el lanzamiento de una aplicación en vRealize Automation Code Stream en un clúster de Kubernetes

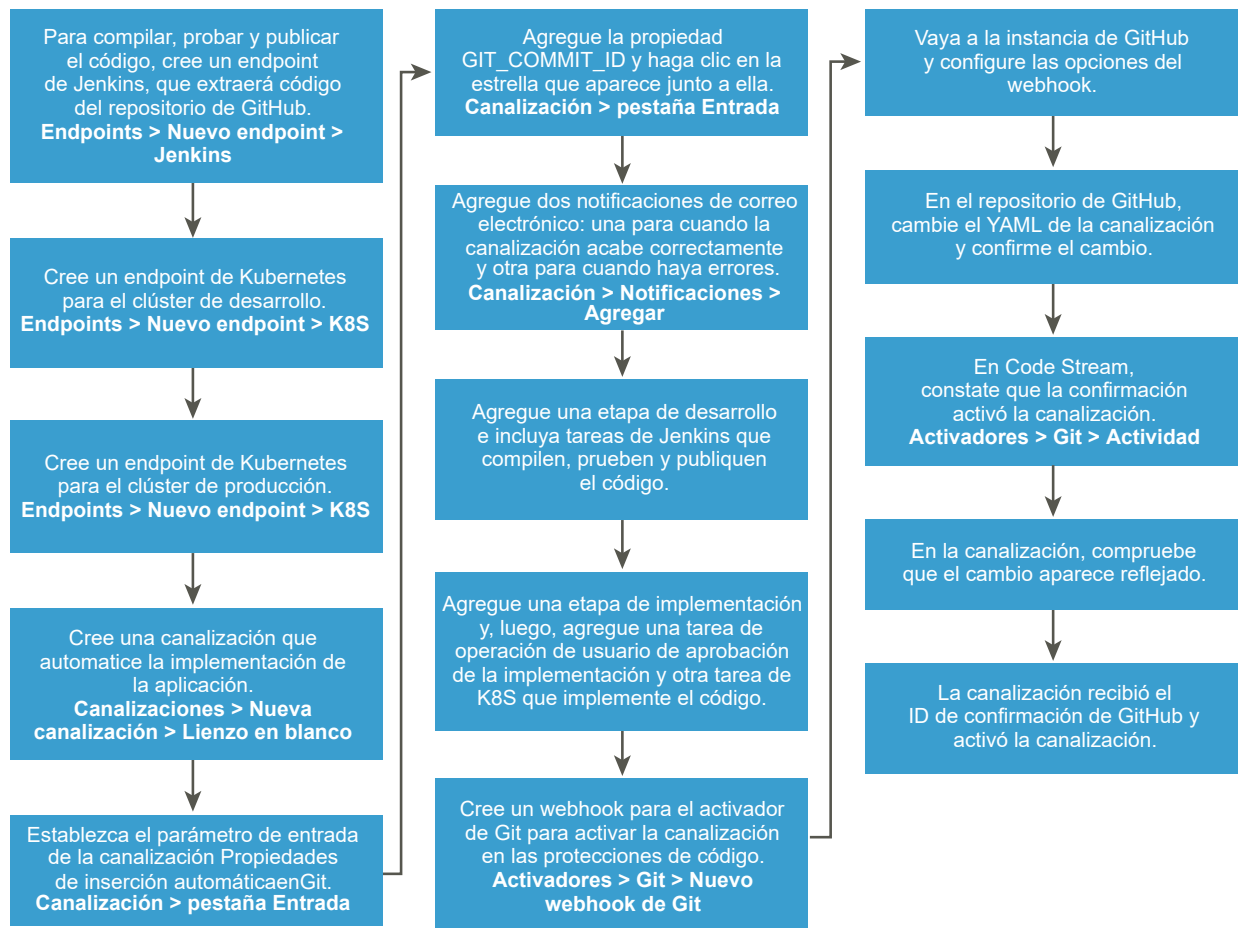
Como desarrollador o administrador de DevOps, puede usar vRealize Automation Code Stream y VMware Cloud PKS para automatizar la implementación de sus aplicaciones de software en un clúster de Kubernetes. En este caso práctico se mencionan otros métodos que se pueden utilizar para automatizar el lanzamiento de aplicaciones.

En este caso práctico, creará una canalización compuesta por dos etapas y utilizará Jenkins para compilar e implementar la aplicación.

- La primera etapa se centra en el desarrollo. En ella, se usa Jenkins para extraer el código de una rama del repositorio de GitHub y, a continuación, este se compila, prueba y publica.
- La segunda etapa se centra en la implementación. En ella, se ejecuta una tarea de operación de usuario que requiere la aprobación de los usuarios principales para que la canalización pueda implementar la aplicación en el clúster de Kubernetes.



Las herramientas de desarrollo, las instancias de implementación y el archivo YAML de la canalización deben estar disponibles para que dicha canalización pueda compilar, probar, publicar e implementar la aplicación. La canalización implementará la aplicación en las instancias de desarrollo y de producción de los clústeres de Kubernetes en AWS.



Estos son otros métodos que automatizan el lanzamiento de la aplicación:

- En lugar de usar Jenkins para compilar la aplicación, se puede utilizar la capacidad de compilación nativa de vRealize Automation Code Stream y un host de compilación de Docker.
- En lugar de implementar la aplicación en un clúster de Kubernetes, se puede implementar en un clúster de Amazon Web Services (AWS).

Para obtener más información sobre cómo usar la capacidad de compilación nativa de vRealize Automation Code Stream y un host de Docker, consulte:

- [Planificar una compilación nativa de CICD en vRealize Automation Code Streamantes de usar la plantilla inteligente](#)
- [Planificar una compilación nativa de CICD en vRealize Automation Code Stream antes de agregar tareas manualmente](#)

Requisitos previos

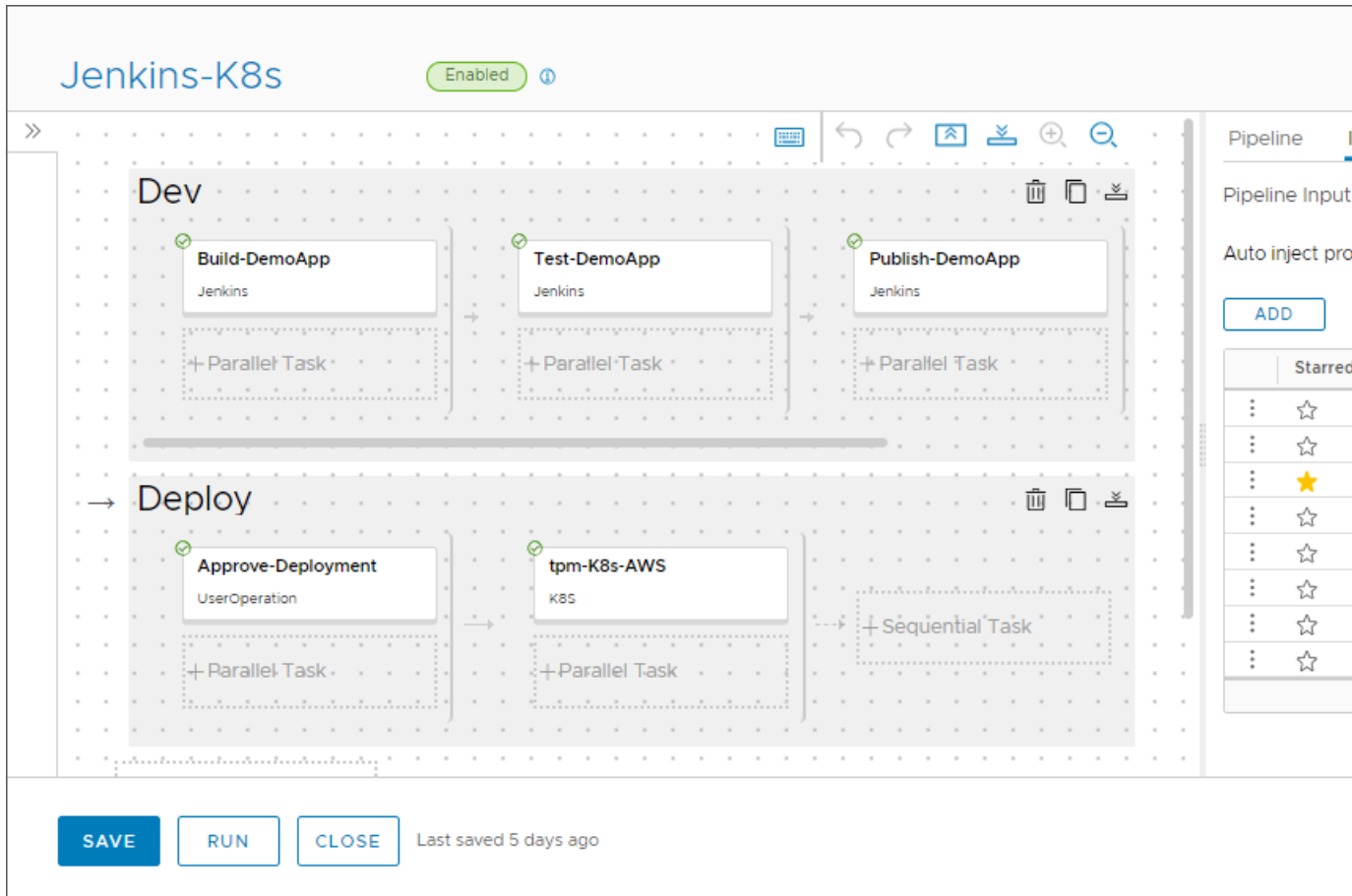
- Compruebe que el código de la aplicación que se va a implementar está en un repositorio de GitHub en funcionamiento.
- Compruebe que tiene una instancia de Jenkins en funcionamiento.
- Compruebe que tiene un servidor de correo electrónico en funcionamiento.
- En vRealize Automation Code Stream, cree un endpoint de correo electrónico que se conecte al servidor de correo electrónico.
- Configure dos clústeres de Kubernetes en Amazon Web Services (AWS), uno de desarrollo y otro de producción, donde la canalización va a implementar la aplicación.
- Compruebe que el repositorio de GitHub contiene el código YAML de la canalización y, opcionalmente, un archivo YAML donde se definen los metadatos y las especificaciones del entorno.

Procedimiento

- 1 En vRealize Automation Code Stream, haga clic en **Endpoints > Nuevo endpoint** y cree un endpoint de Jenkins, que utilizará en la canalización para extraer el código del repositorio de GitHub.
- 2 Para crear endpoints de Kubernetes, haga clic en **Nuevo endpoint**.
 - a Cree un endpoint para el clúster de Kubernetes de desarrollo.
 - b Cree un endpoint para el clúster de Kubernetes de producción.

- 3 Cree una canalización que implemente un contenedor de la aplicación (como WordPress) en el clúster de Kubernetes de desarrollo y establezca las propiedades de entrada de la canalización.
 - a Para que la canalización pueda reconocer una confirmación de código en GitHub que active la canalización, haga clic en la pestaña **Entrada** de la canalización y seleccione **Propiedades de inserción automática**.
 - b Agregue la propiedad **GIT_COMMIT_ID** y haga clic en la estrella que la acompaña.

Cuando la canalización se ejecute, dicha ejecución mostrará el identificador de confirmación que el activador de Git devuelve.



- 4 Agregue notificaciones para enviar un correo electrónico cuando la canalización realice la operación correcta o incorrectamente.
 - a En la canalización, haga clic en la pestaña **Notificaciones** y, a continuación, en **Agregar**.
 - b Para agregar una notificación de correo electrónico cuando la canalización termine de ejecutarse, seleccione **Correo electrónico** y seleccione **Al finalizar la canalización**. Luego, seleccione el servidor de correo electrónico, introduzca las direcciones de correo electrónico y haga clic en **Guardar**.
 - c Para agregar otra notificación de correo electrónico, esta vez para cuando se produzca un error en la canalización, seleccione **En caso de error de canalización** y haga clic en **Guardar**.

Notification

Type

☒ Email ☐ Ticket ☐ Webhook

Event

☒ On Pipeline Completion ☐ On Pipeline Waiting ☐ On Pipeline Failure
☐ On Pipeline Cancellation ☐ On Pipeline Started

Email server ⓘ *

--Select Email server-- ▾

Send Email

To ⓘ \$ *

Email IDs of recipients

Subject \$ *

Email Subject

Body ⓘ \$ *

1

CANCEL

SAVE

- 5 Agregue una etapa de desarrollo a la canalización y agregue tareas que compilen, prueben y publiquen la aplicación. Después, valide cada tarea.
 - a Para compilar la aplicación, agregue una tarea de Jenkins que use el endpoint de Jenkins y ejecute un trabajo de compilación desde el servidor de Jenkins. A continuación, para que la canalización extraiga el código, introduzca la rama de Git con este formato: `{input.GIT_BRANCH_NAME}`
 - b Para probar la aplicación, agregue una tarea de Jenkins que use el mismo endpoint de Jenkins y ejecute un trabajo de prueba desde el servidor de Jenkins. A continuación, introduzca la misma rama de Git.
 - c Para publicar la aplicación, agregue una tarea de Jenkins que use el mismo endpoint de Jenkins y ejecute un trabajo de publicación desde el servidor de Jenkins. A continuación, introduzca la misma rama de Git.

The screenshot displays the Jenkins-K8s interface with a pipeline configuration for a stage named 'Dev'. The pipeline is titled 'Jenkins-K8s' and is marked as 'Enabled'. The 'Dev' stage contains three Jenkins tasks: 'Build-DemoApp', 'Test-DemoApp', and 'Publish-DemoApp'. Each task is configured with a Jenkins endpoint and a job name. Below the 'Dev' stage, there is a 'Deploy' task and a '+ Stage' button. The right sidebar shows the configuration for the selected task, 'Task : Build-DemoApp', including fields for 'Task name', 'Type', 'Continue On Failure', 'Execute Task', 'Jenkins Job', 'Endpoint', 'Job', 'branchName', and 'Output Parameters'. The 'Output Parameters' section shows 'status', 'job', and 'jobid'.

Jenkins-K8s Enabled

Dev

- Build-DemoApp** (Jenkins)
- Test-DemoApp** (Jenkins)
- Publish-DemoApp** (Jenkins)

Below the 'Dev' stage, there is a **Deploy** task and a **+ Stage** button.

Task : Build-DemoApp

- Task name
- Type
- Continue On Failure
- Execute Task
- Jenkins Job
- Endpoint
- Job
- branchName
- Output Parameters: status, job, jobid

SAVE **RUN** **CLOSE** Last saved 5 days ago

- 6 Agregue una etapa de implementación a la canalización y, a continuación, agregue una tarea que requiera aprobación para la implementación de la aplicación y otra tarea que implemente la aplicación en el clúster de Kubernetes. Después, valide cada tarea.
 - a Para requerir una aprobación de la implementación de la aplicación, agregue una tarea de operación de usuario, agregue las direcciones de correo electrónico de los usuarios que deben aprobarla y escriba un mensaje. Luego, habilite **Enviar correo electrónico**.
 - b Para implementar la aplicación, agregue una tarea de Kubernetes. A continuación, en las propiedades de la tarea de Kubernetes, seleccione el clúster de Kubernetes de desarrollo, seleccione la acción **Crear** y, después, seleccione el origen de carga útil **Definición local**. Tras ello, seleccione el archivo YAML local.
- 7 Agregue un webhook de Git que permita a vRealize Automation Code Stream utilizar el activador de Git, que activa la canalización cuando los desarrolladores confirman sus códigos.

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 field)
- 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 probar la canalización, vaya al repositorio de GitHub, actualice el archivo YAML de la aplicación y confirme el cambio.
 - a En vRealize Automation Code Stream, compruebe que la confirmación aparece.
 - a Haga clic en **Activadores > Git > Actividad**.
 - b Busque el activador de la canalización.

- c Haga clic en **Paneles de control > Paneles de control de canalizaciones**.
- d En el panel de control de la canalización, busque GIT_COMMIT_ID en el área de último cambio realizado correctamente.

9 Repase el código de la canalización y confirme que el cambio aparece reflejado.

Resultados

Enhorabuena. Automatizó la implementación de la aplicación de software en el clúster de Kubernetes.

Ejemplo: YAML de canalización de ejemplo que implementa una aplicación en un clúster de Kubernetes

El YAML del tipo de canalización utilizado en este ejemplo es similar al siguiente código:

```
apiVersion: v1
kind: Namespace
metadata:
  name: ${input.GIT_BRANCH_NAME}
  namespace: ${input.GIT_BRANCH_NAME}
---
apiVersion: v1
data:
  .dockercfg:
    eyJzeWlwG9ueS10YW5nbyliZXRhMi5qZnJvZy5pbyI6eyJlc2VybmFtZSI6InRhbmdvLWJldGEyIiwicGFzc3dvcmQI Oi
    JhRGstcmVOLWlUQi1IejciLCJlbWFPbCI6InRhbmdvLWJldGEyQHZtd2FyZS5jb20iLCJhdXRoIjoizEdGdVoyOHRZbVYw
    WVRJNllVUnJWWEpsVGkxdFZFSXRTSG8zIn19
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

```

Pasos siguientes

Para implementar la aplicación de software en el clúster de Kubernetes de producción, vuelva a realizar los pasos y seleccione el clúster de producción.

Para obtener más información sobre cómo integrar vRealize Automation Code Stream con Jenkins, consulte [Cómo se integra vRealize Automation Code Stream con Jenkins](#).

Cómo implementar la aplicación en vRealize Automation Code Stream en una implementación azul-verde

El modelo de implementación azul-verde utiliza dos hosts de Docker que implementa y configura de manera idéntica en un clúster de Kubernetes. Con el modelo de implementación azul-verde, se reduce el periodo de inactividad que puede producirse en el entorno cuando las canalizaciones de vRealize Automation Code Stream implementan las aplicaciones.

Cada una de las instancias de azul y verde del modelo de implementación tiene un objetivo diferente. Solo una instancia a la vez acepta el tráfico activo que implementa la aplicación y cada instancia acepta ese tráfico en momentos específicos. La instancia azul recibe la primera versión de la aplicación y la instancia verde recibe la segunda.

El equilibrador de carga del entorno azul-verde determina la ruta que toma el tráfico activo al implementar la aplicación. Al utilizar el modelo azul-verde, el entorno permanece operativo, los usuarios no experimentan ningún periodo de inactividad y la canalización integra e implementa la aplicación en el entorno de producción de forma continua.

La canalización que crea en vRealize Automation Code Stream representa el modelo de implementación azul-verde en dos etapas. Una etapa es para el desarrollo y la otra etapa para la producción.

Tabla 5-2. Tareas de la etapa de desarrollo para una implementación azul-verde

Tipo de tarea	Tarea
Kubernetes	Cree un espacio de nombres para la implementación azul-verde.
Kubernetes	Cree una clave secreta para Docker Hub.
Kubernetes	Cree el servicio utilizado para implementar la aplicación.
Kubernetes	Cree la implementación azul.
Sondeo	Compruebe la implementación azul.
Kubernetes	Elimine el espacio de nombres.

Tabla 5-3. Tareas de la etapa de producción para una implementación azul-verde

Tipo de tarea	Tarea
Kubernetes	La instancia verde obtiene los detalles del servicio de la azul.
Kubernetes	Obtenga los detalles del conjunto de réplicas verde.
Kubernetes	Cree la implementación verde y use la clave secreta para extraer la imagen del contenedor.
Kubernetes	Actualice el servicio.
Sondeo	Compruebe que la implementación se haya completado correctamente en la URL de producción.
Kubernetes	Finalice la implementación azul.
Kubernetes	Elimine la implementación azul.

Para implementar la aplicación en su propio modelo de implementación azul-verde, cree una canalización en vRealize Automation Code Stream que incluya dos etapas. La primera etapa incluye las tareas azules que implementan la aplicación en la instancia azul, mientras que la segunda incluye tareas verdes que implementan la aplicación en la instancia verde.

Puede crear la canalización mediante la plantilla inteligente de CICD. La plantilla crea automáticamente las etapas y las tareas de la canalización e incluye las selecciones de implementación.

Si crea canalización de forma manual, debe planificar las etapas de la canalización. Para ver un ejemplo, consulte [Planificar una compilación nativa de CICD en vRealize Automation Code Stream antes de agregar tareas manualmente](#).

En este ejemplo, se utiliza la plantilla inteligente de CICD para crear la canalización azul-verde.

Requisitos previos

- Compruebe que puede acceder a un clúster de Kubernetes operativo en AWS.
- Asegúrese de que ha configurado un entorno de implementación azul-verde y que las instancias de azul y verde son idénticas.
- Cree un endpoint de Kubernetes en vRealize Automation Code Stream que implemente la imagen de la aplicación en el clúster de Kubernetes en AWS.
- Familiarícese con la plantilla inteligente de integración continua y distribución continua (Continuous Integration and Continuous Delivery, CICD). Consulte [Planificar una compilación nativa de CICD en vRealize Automation Code Streamantes de usar la plantilla inteligente](#).

Procedimiento


- 1 Haga clic en **Canalizaciones > Nueva canalización > Plantillas inteligentes > Plantilla de CI/CD**.
- 2 Introduzca la información de la parte de CI de la plantilla inteligente de CICD y haga clic en **Siguiente**.


Para obtener ayuda, consulte [Planificar una compilación nativa de CICD en vRealize Automation Code Streamantes de usar la plantilla inteligente](#).
- 3 Complete la parte de CD de la plantilla inteligente.
 - a Seleccione los entornos para la implementación de la aplicación. Por ejemplo, **Dev** y **Prod**.
 - b Seleccione el servicio que utilizará la canalización para la implementación.

- c En el área Implementación, seleccione el endpoint del clúster para los entornos Dev y Prod.
- d Para el modelo de implementación de producción, seleccione **Azul-verde** y haga clic en **Crear**.

Plantilla inteligente: CI/CD

Paso 2 de 2

Entorno  ☒ Desarrollo ☒ Producción

Archivos YAML de Kube...  SELECCIONAR PROCESAR

Archivos procesados: codestream.yaml

Seleccionar servicio

Nombre de la implementación	Servicio	Espacio de nombres	Imagen
codestream-demo	codestream-demo	bgreen1	4

1 servicios

Implementación

Entorno	Endpoint del clúster	Espacio de nombres
Desarrollo	1030Endpoint-Kubernetes	bgreen1-670443
Producción	1030Endpoint-Kubernetes	bgreen1

Modelo de implementac... ☐ Canary ☐ Actualización gradual ☒ azul-verde

Revertir ☐

URL de comprobación d...

CREAR ATRÁS CANCELAR

Resultados

Enhorabuena. Ha utilizado la plantilla inteligente para crear una canalización que implementa la aplicación en las instancias azul-verde en el clúster de producción de Kubernetes en AWS.

Ejemplo: Ejemplo de código de YAML para algunas tareas de implementación azul-verde

El código de YAML que aparece en las tareas de canalización de Kubernetes para la implementación azul-verde puede ser similar a los siguientes ejemplos. Después de que la plantilla inteligente cree la canalización, puede modificar las tareas según sea necesario para su propia implementación.

Código de YAML para crear un espacio de nombres de ejemplo:

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

Código de YAML para crear un servicio de ejemplo:

```
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 de YAML para crear una implementación de ejemplo:

```
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
```

Pasos siguientes

Para obtener más información sobre cómo puede usar vRealize Automation Code Stream, consulte [Capítulo 5 Ejemplos de casos prácticos comunes para vRealize Automation Code Stream](#).

Para revertir una implementación, consulte [Cómo revertir una implementación en vRealize Automation Code Stream](#).

Para obtener más referencias, consulte [Más recursos para desarrolladores y administradores de vRealize Automation Code Stream](#).

Cómo se integran las herramientas personalizadas de compilación, pruebas e implementación con vRealize Automation Code Stream

Como administrador o desarrollador de DevOps, puede crear scripts personalizados que amplíen la capacidad de vRealize Automation Code Stream. Con un script, puede integrar vRealize Automation Code Stream con sus propias API y herramientas de integración continua (Continuous Integration, CI) y distribución continua (Continuous Delivery, CD) que compilan, prueban e implementan las aplicaciones. Los scripts personalizados son especialmente útiles si no se exponen las API de aplicaciones de manera pública.

El script personalizado puede hacer casi todo lo necesario para integrarse con las herramientas de compilación, pruebas e implementación. Por ejemplo, puede utilizarse con el área de trabajo de la canalización para brindar soporte a tareas de CI que compilan y prueban la aplicación, así como a tareas de CD que implementan la aplicación. Puede enviar un mensaje a Slack cuando finaliza una canalización, etc.

Escriba el script personalizado en uno de los lenguajes compatibles. En el script, incluya la lógica empresarial y defina las entradas y las salidas. Los tipos de salida pueden ser números, cadenas, texto y contraseñas. Puede crear varias versiones de un script personalizado con diferentes entradas, salidas y lógicas empresariales.

Haga que la canalización ejecute una versión del script en una tarea personalizada. Los scripts que cree residirán en la instancia de vRealize Automation Code Stream.

Cuando una canalización utiliza una integración personalizada y se intenta eliminarla, aparece un mensaje de error que indica que no se puede eliminar.

Al eliminar una integración personalizada, se eliminan todas las versiones de su script personalizado. Si tiene una canalización existente con una tarea personalizada que utiliza cualquier versión del script, se producirá un error en esa canalización. Para asegurarse de que no se produzcan errores en las canalizaciones existentes, puede descartar y retirar la versión del script que ya no desee utilizar. Si ninguna canalización utiliza esa versión, puede eliminarla.

Tabla 5-4. Qué hacer después de escribir el script personalizado

Qué hacer...	Más información sobre esta acción...
Agregue una tarea personalizada a la canalización.	<p>La tarea personalizada:</p> <ul style="list-style-type: none"> ■ Se ejecuta en el mismo contenedor que otras tareas de CI de la canalización. ■ Incluye variables de entrada y de salida que el script rellena antes de que la canalización ejecute la tarea personalizada. ■ Admite varios tipos de datos y varios tipos de metadatos que define como entradas y salidas en el script.
Seleccione el script en la tarea personalizada.	Declare las propiedades de entrada y de salida en el script.
Guarde la canalización y, a continuación, habilítela y ejecútela.	Cuando se ejecuta la canalización, la tarea personalizada llama a la versión del script especificado y ejecuta la lógica empresarial que contiene, lo que integra la herramienta de compilación, pruebas e implementación con vRealize Automation Code Stream.
Una vez que se ejecute la canalización, observe las ejecuciones.	Compruebe que la canalización ha producido los resultados esperados.

En este ejemplo, se crea una integración personalizada que conecta vRealize Automation Code Stream con la instancia de Slack y publica un mensaje en un canal de Slack.

Requisitos previos

- Para escribir un script personalizado, asegúrese de tener uno de estos lenguajes: Python 2, Python 3, Node.js o cualquiera de estos lenguajes de shell: Bash, sh o zsh.
- Genere una imagen de contenedor mediante el tiempo de ejecución de Node.js o Python instalado.

Procedimiento

1 Cree la integración personalizada.

- Haga clic en **Integraciones personalizadas > Nuevo** e introduzca un nombre pertinente.
- Seleccione el entorno de tiempo de ejecución que prefiera.
- Haga clic en **Crear**.

Se abrirá el script mostrando el código, que incluye el entorno de tiempo de ejecución requerido. Por ejemplo, `runtime: "nodejs"`. El script debe incluir el tiempo de ejecución que la imagen de Builder usa, de forma que la tarea personalizada que agregue a la canalización se realice correctamente cuando la canalización se ejecute. De lo contrario, se producirá un error en la tarea personalizada.

Las principales áreas de los YAML de integración personalizada incluyen el tiempo de ejecución, el código, las propiedades de entrada y las propiedades de salida. Este procedimiento explica diversos tipos y sintaxis.

Claves de YAML de integración personalizada	Descripción
runtime	Entorno de tiempo de ejecución de la tarea en el que vRealize Automation Code Stream ejecuta el código, que puede ser una de estas cadenas sin distinción entre mayúsculas y minúsculas: <ul style="list-style-type: none"> ■ nodejs ■ python2 ■ python3 ■ shell Si no se proporciona nada, se asume el valor predeterminado de shell.
code	Lógica empresarial personalizada que se ejecutará como parte de la tarea personalizada.
inputProperties	Matriz de propiedades de entrada que se capturarán como parte de la configuración de la tarea personalizada. Por lo general, estas propiedades se utilizan en el código.
outputProperties	Matriz de propiedades de salida que puede exportar de la tarea personalizada para propagarla a la canalización.

- 2 Declare las propiedades de entrada en el script mediante los tipos de datos y los metadatos disponibles.

Las propiedades de entrada se transfieren como contexto al script en la sección `code`: del YAML.

Claves de entrada de YAML de tareas personalizadas	Descripción	Obligatorio
type	Tipos de entrada para representar: <ul style="list-style-type: none"> ■ Texto ■ Área de texto ■ Número ■ Casilla de verificación ■ Contraseña ■ Selección 	Sí
name	Nombre o cadena de entrada a la tarea personalizada, que se inyecta en el código YAML de integración personalizada. Debe ser único para cada propiedad de entrada definida para una integración personalizada.	Sí
title	Etiqueta de la cadena de texto de la propiedad de entrada de la tarea personalizada en el lienzo del modelo de canalización. Si se deja en blanco, se utiliza name de forma predeterminada.	No
required	Determina si un usuario debe introducir la propiedad de entrada al configurar la tarea personalizada. Establézcalo en true o false. Cuando el valor es true, si un usuario no proporciona un valor al configurar la tarea personalizada en el lienzo de canalización, el estado de la tarea permanece como sin configurar.	No

Claves de entrada de YAML de tareas personalizadas	Descripción	Obligatorio
placeholder	Texto predeterminado para el área de entrada de propiedades de entrada cuando no hay ningún valor. Se asigna al atributo del marcador de posición HTML. Solo se admite para ciertos tipos de propiedad de entrada.	No
defaultValue	Valor predeterminado que rellena el área de entrada de la propiedad de entrada cuando la tarea personalizada se representa en la página del modelo de canalización.	No
bindable	Determina si la propiedad de entrada acepta variables de signo de dólar al modelar la tarea personalizada en el lienzo de canalización. Agrega el indicador \$ junto al título. Solo se admite para ciertos tipos de propiedad de entrada.	No
labelMessage	Cadena que actúa como información sobre herramientas de ayuda para los usuarios. Agrega un icono de información sobre herramientas i junto al título de entrada.	No
enum	<p>Toma una matriz de valores que muestra las opciones de la propiedad de entrada seleccionada. Solo se admite para ciertos tipos de propiedad de entrada.</p> <p>Cuando un usuario selecciona una opción y la guarda para la tarea personalizada, el valor de inputProperty corresponde a este valor y aparece en el modelado de tareas personalizado.</p> <p>Por ejemplo, el valor 2015.</p> <ul style="list-style-type: none"> ■ 2015 ■ 2016 ■ 2017 ■ 2018 ■ 2019 ■ 2020 	No
options	<p>Toma una matriz de objetos usando optionKey y optionValue.</p> <ul style="list-style-type: none"> ■ optionKey. Valor propagado a la sección de código de la tarea. ■ optionValue. Cadena que muestra la opción en la interfaz de usuario. <p>Solo se admite para ciertos tipos de propiedad de entrada.</p> <p>Opciones:</p> <p>optionKey: key1. Cuando se selecciona y se guarda para la tarea personalizada, el valor de inputProperty correspondería a key1 en la sección de código.</p> <p>optionValue: 'Etiqueta para 1'. Valor para mostrar de key1 en la interfaz de usuario. No aparece en ningún otro lugar para la tarea personalizada.</p> <p>optionKey: key2</p> <p>optionValue: 'Etiqueta para 2'</p> <p>optionKey: key3</p> <p>optionValue: 'Etiqueta para 3'</p>	No

Claves de entrada de YAML de tareas personalizadas	Descripción	Obligatorio
minimum	Acepta un número que actúa como el valor mínimo que es válido para esta propiedad de entrada. Solo se admite para la propiedad de entrada del tipo de número.	No
maximum	Acepta un número que actúa como el valor máximo que es válido para esta propiedad de entrada. Solo se admite para la propiedad de entrada del tipo de número.	No

Tabla 5-5. Tipos de datos y metadatos admitidos para scripts personalizados

Tipos de datos admitidos	Metadatos admitidos para la entrada
<ul style="list-style-type: none"> ■ String ■ Text ■ List: como una lista de cualquier tipo ■ Map: como map[string]any ■ Secure: se representa como un cuadro de texto de contraseña, cifrado cuando se guarda la tarea personalizada ■ Number ■ Boolean: se muestra como cuadros de texto ■ URL: igual que una cadena, con validación adicional ■ Botón de selección o de radio 	<ul style="list-style-type: none"> ■ type: un tipo como String Text... ■ default: valor predeterminado ■ options: lista o asignación de opciones que se usará con un botón de selección o de radio ■ min: valor o tamaño mínimo ■ max: valor o tamaño máximo ■ title: nombre detallado del cuadro de texto ■ placeholder: marcador de posición de la interfaz de usuario ■ description: se convierte en información sobre herramientas

Por ejemplo:

```
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 las propiedades de salida del script.

El script captura las propiedades de salida de la sección `code`: de la lógica empresarial del script, donde declara el contexto de la salida.

Cuando se ejecuta la canalización, puede introducir el código de respuesta para la salida de la tarea. Por ejemplo, **200**.

Las claves que vRealize Automation Code Stream admite para cada **outputProperty**.

key	Descripción
type	Actualmente incluye un solo valor de label .
name	Clave que emite el bloque de código de YAML de integración personalizada.
title	Etiqueta en la interfaz de usuario que muestra outputProperty .

Por ejemplo:

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

- 4 Para interactuar con la entrada y la salida del script personalizado, obtenga una propiedad de entrada o establezca una propiedad de salida mediante **context**.

Para una propiedad de entrada: `(context.getInput("key"))`

Para una propiedad de salida: `(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 En la sección `code:`, declare toda la lógica empresarial de la integración personalizada.

Por ejemplo, con el entorno de tiempo de ejecución de 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 de crear una versión del script de integración personalizado y publicarla, descargue el archivo de contexto para Python o Node.js y pruebe la lógica empresarial que incluyó en el script.
 - a Coloque el cursor en el script y haga clic en el botón de archivo de contexto situado en la parte superior del lienzo. Por ejemplo, si el script se encuentra en Python, haga clic en **CONTEXT.PY**.
 - b Modifique el archivo y guárdelo.
 - c En el sistema de desarrollo, ejecute y pruebe su script personalizado con la ayuda del archivo de contexto.
- 7 Aplique una versión al script de integración personalizado.
 - a Haga clic en **Versión**.
 - b Introduzca la información de la versión.

- c Haga clic en **Versión de publicación** para poder seleccionar el script en la tarea personalizada.
- d Para crear la versión, haga clic en **Crear**.

Creando versión

Versión * 1.0

Descripción New

Registro de cambios New for 1.0

Publicar versión ☒

- 8 Para guardar el script, haga clic en **Guardar**.
- 9 En la canalización, configure el área de trabajo.
 - a Haga clic en la pestaña **Área de trabajo**.
 - b Seleccione el host de Docker y la dirección URL de la imagen de Builder.

Demo-customTask-nodejs Habilitado

Área de trabajo Entrada Modelo Salida

Proporcione detalles sobre el contenedor y el host para ejecutar tareas de integración continua.

Host * Docker-saas

Dirección URL de la i... node.lasest

Registro de imágenes --Seleccionar endpoint de registro de imágenes--

Directorio de trabajo

Memoria caché

Límite de CPU *

Límite de memoria *

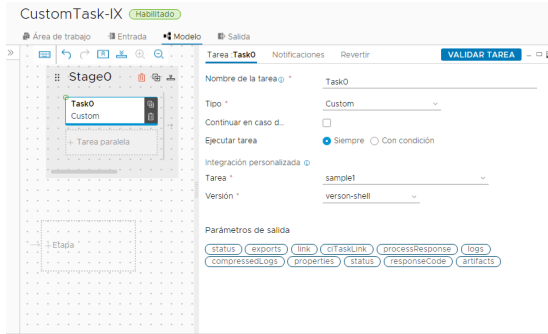
Clon de Git ☐

Si esta canalización se vincula a Git a través de un webhook, la canalización se activará en los eventos de Git. Para las tareas de CI, el repositorio de Git vinculado (detalles de los parámetros de webhook de Git) se clona automáticamente en el área de trabajo.

- 10 Agregue una tarea personalizada a la canalización y configúrela.
 - a Haga clic en la pestaña **Modelo**.
 - b Agregue una tarea, seleccione el tipo como **Personalizado** e introduzca un nombre pertinente.

- c Seleccione el script de integración personalizado y la versión.
- d Para mostrar un mensaje personalizado en Slack, introduce el texto del mensaje.

Cualquier texto que introduzca reemplaza el valor `defaultValue` en el script de integración personalizado. Por ejemplo:

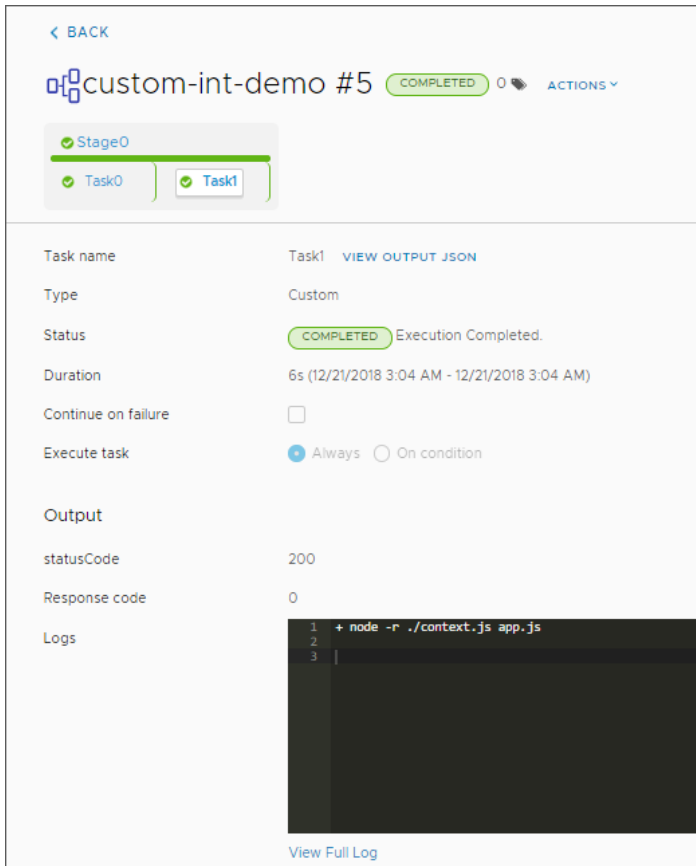


- 11 Guarde y habilite la canalización.
 - a Haga clic en **Guardar**.
 - b En la pestaña Canalización, haga clic en **Habilitar canalización** para que el círculo se mueva hacia la derecha.
- 12 Ejecute la canalización.
 - a Haga clic en **Ejecutar**.
 - b Observe la ejecución de la canalización.

- c Confirme que la salida incluya el código de estado, el código de respuesta, el estado y la salida declarada previstos.

Ha definido **statusCode** como una propiedad de salida. Por ejemplo, si **statusCode** tiene un valor de 200 puede indicar una publicación de Slack correcta, y **responseCode** con un valor de 0 puede indicar que el script se ha completado correctamente sin errores.

- d Para confirmar la salida en los registros de ejecución, haga clic en **Ejecuciones**, haga clic en el vínculo a la canalización, haga clic en la tarea y observe los datos registrados. Por ejemplo:



- 13 Si se produce un error, solucione el problema y vuelva a ejecutar la canalización.

Por ejemplo, si falta un archivo o un módulo en la imagen base, debe crear otra imagen base que incluya el archivo que falta. A continuación, proporcione el archivo de Docker e inserte la imagen a través de la canalización.

Resultados

Enhorabuena. Se creó un script de integración personalizada que conecta vRealize Automation Code Stream con la instancia de Slack y publica un mensaje en un canal de Slack.

Pasos siguientes

Siga creando integraciones personalizadas para permitir el uso de tareas personalizadas en las canalizaciones, de modo que pueda ampliar la capacidad de vRealize Automation Code Stream en la automatización del ciclo de vida de publicación de software.

Cómo usar una API de REST para integrar vRealize Automation Code Stream con otras aplicaciones

vRealize Automation Code Stream proporciona un complemento de REST que le permite integrar vRealize Automation Code Stream con otras aplicaciones que utilizan una API de REST para que pueda desarrollar y distribuir continuamente aplicaciones de software que deben interactuar entre sí. El complemento de REST invoca una API, que envía y recibe información entre vRealize Automation Code Stream y otra aplicación.

Con el complemento de REST, puede:

- Integrar sistemas externos basados en API de REST en una canalización de vRealize Automation Code Stream.
- Integrar una canalización de vRealize Automation Code Stream como parte del flujo de sistemas externos.

El complemento de REST funciona con cualquier API de REST y admite los métodos GET, POST, PUT, PATCH y DELETE para enviar o recibir información entre vRealize Automation Code Stream y otras aplicaciones.

Tabla 5-6. Preparar una canalización para comunicarse a través de la API de REST

Qué hacer...	Qué sucede...
Agregue una tarea de REST a la canalización.	La tarea de REST comunica información entre aplicaciones y puede proporcionar información de estado para una tarea posterior en la etapa de la canalización.
En la tarea de REST, seleccione la acción REST e incluya la URL.	La tarea de la canalización llama a la URL cuando se ejecuta la canalización. Para las acciones POST, PUT y PATCH, debe incluir una carga útil. En la carga útil, puede enlazar las propiedades de la canalización y de la tarea cuando se ejecuta la canalización.
Observe este ejemplo.	Ejemplo de uso del complemento de REST: Puede agregar una tarea de REST para crear una etiqueta en una confirmación de Git para una compilación y hacer que la tarea envíe una solicitud para obtener el identificador de inserción del repositorio. La tarea puede enviar una carga útil al repositorio y crear una etiqueta para la compilación, y el repositorio puede devolver la respuesta con la etiqueta.

Al igual que si usa el complemento de REST para invocar una API, puede incluir una tarea de sondeo en la canalización para invocar una API de REST y sondearla hasta que finalice y la tarea de la canalización cumpla los criterios de salida.

También puede usar las API de REST para importar y exportar una canalización, y utilizar los scripts de ejemplo para ejecutar una canalización.

Este procedimiento obtiene una dirección URL simple.

Procedimiento

- 1 Para crear una canalización, haga clic en **Canalizaciones > Nueva canalización > Lienzo en blanco**.
- 2 En la etapa de la canalización, haga clic en **+ Tarea secuencial**.
- 3 En el panel de tareas, agregue la tarea de REST:
 - a Introduzca un nombre para la tarea.
 - b En el menú desplegable Tipo, seleccione **REST**.
 - c En el área Solicitud de REST, seleccione **GET**.

Para que la tarea de REST solicite datos de otra aplicación, seleccione el método GET.
Para enviar datos a otra aplicación, seleccione el método POST.

- d Introduzca la URL que identifica el endpoint de la API de REST. Por ejemplo, `https://www.google.com`.

Para que una tarea de REST importe datos desde otra aplicación, puede incluir la variable de carga útil. Por ejemplo, para una acción de importación, puede introducir `${Stage0.export.responseBody}`. Si el tamaño de los datos de respuesta supera los 5 MB, es posible que se produzca un error en la tarea de REST.

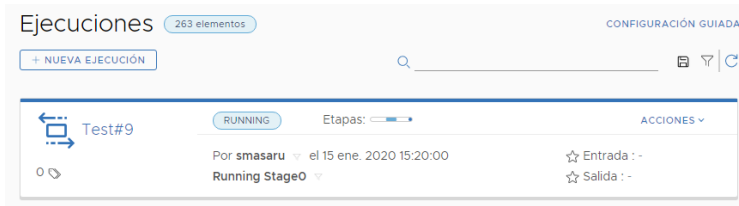
- e Para proporcionar autorización para la tarea, haga clic en **Agregar encabezados** e introduzca una clave de encabezado y un valor.

- 4 Para guardar la canalización, haga clic en **Guardar**.
- 5 En la pestaña Canalización, haga clic en **Habilitar canalización**.

- 6 Haga clic en **Guardar** y, a continuación, haga clic en **Cerrar**.

7 Haga clic en **Ejecutar**.

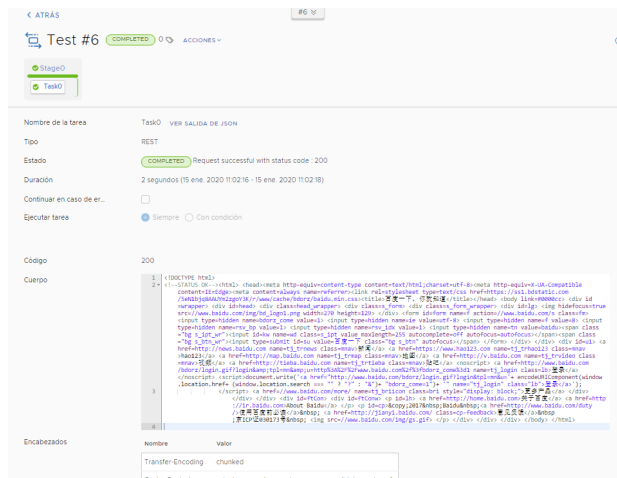
8 Para ver la ejecución de la canalización, haga clic en **Ejecuciones**.



9 Para comprobar que el complemento de REST devuelve la información esperada, examine la ejecución de la canalización y los resultados de la tarea.

- Una vez que se complete la canalización, para confirmar que la otra aplicación devolvió los datos solicitados, haga clic en el vínculo a la ejecución de la canalización.
- Haga clic en la tarea de REST de la canalización.
- En la ejecución de la canalización, haga clic en la tarea, examine los detalles de la tarea y compruebe que el complemento de REST devolvió los resultados esperados.

Los detalles de la tarea muestran el código de respuesta, el cuerpo, las claves de encabezado y los valores.



10 Para ver la salida de JSON, haga clic en **VER SALIDA DE JSON**.

```

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_zRnp3usxttMpd7YiqRUOSfMkTC7cTERbd
    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/googlelog/1x/googlelog_standard_color_128dp.png\" itemprop=\"image\"><title>Google</title><script
nonce=\"aWwMw/ydugKGr9CHU6QQGz==\"><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
,430,2487,850,525,22,599,5,2,2,1963,528,3,1959,105,465,556,905,1378,966,942,108,334,130,1190,154,306,8
,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

Enhorabuena. Ha configurado una tarea de REST que ha invocado una API de REST y enviado información entre vRealize Automation Code Stream y otra aplicación mediante el complemento de REST.

Pasos siguientes

Siga utilizando tareas de REST en las canalizaciones para ejecutar comandos e integrar vRealize Automation Code Stream con otras aplicaciones para poder desarrollar y distribuir aplicaciones de software. Considere la posibilidad de usar tareas de sondeo para sondear la API hasta que finalice y la tarea de canalización cumpla los criterios de salida.

Conectar vRealize Automation Code Stream a endpoints

6

vRealize Automation Code Stream se integra con herramientas de desarrollo a través de complementos. Entre los complementos compatibles se encuentran Jenkins, Bamboo, Artifactory, vRealize Operations, Pivotal Cloud Foundry, Bugzilla, Team Foundation Server, Git y muchos otros.

También puede desarrollar sus propios complementos para integrar vRealize Automation Code Stream con otras aplicaciones de desarrollo.

Para integrar vRealize Automation Code Stream con JIRA, no se necesita un complemento externo, ya que la funcionalidad de creación de tickets JIRA está integrada en vRealize Automation Code Stream como un tipo de notificación. Para crear tickets JIRA sobre el estado de la canalización, debe agregar un endpoint de JIRA.

Este capítulo incluye los siguientes temas:

- [Qué son los endpoints en vRealize Automation Code Stream](#)
- [Cómo se integra vRealize Automation Code Stream con Jenkins](#)
- [Cómo se puede integrar vRealize Automation Code Stream con Git](#)
- [Cómo se integra vRealize Automation Code Stream con Gerrit](#)
- [Cómo se integra vRealize Automation Code Stream con vRealize Orchestrator](#)

Qué son los endpoints en vRealize Automation Code Stream

Un endpoint es una instancia de una aplicación de DevOps que se conecta a vRealize Automation Code Stream para proporcionar datos para que se ejecuten las canalizaciones, como un origen de datos, un repositorio o un sistema de notificaciones.

Su función en vRealize Automation Code Stream determina cómo utiliza los endpoints.

- Los administradores y los desarrolladores pueden crear, actualizar, eliminar y ver endpoints.
- Los administradores pueden marcar un endpoint como restringido y ejecutar canalizaciones que usen endpoints restringidos.
- Los usuarios que tienen la función de visor pueden ver endpoints, pero no pueden crearlos, actualizarlos ni eliminarlos.

Para obtener más información, consulte [Cómo se administran las autorizaciones y el acceso de los usuarios en vRealize Automation Code Stream](#).

Para conectar vRealize Automation Code Stream a un endpoint, debe agregar una tarea en la canalización y configurarla para que se comuniquen con el endpoint. Para comprobar que vRealize Automation Code Stream se pueda conectar al endpoint, haga clic en **Validar**. A continuación, cuando ejecute la canalización, la tarea de la canalización se conecta al endpoint para ejecutar la tarea.

Tabla 6-1. Endpoints compatibles con vRealize Automation Code Stream

Endpoint	Para qué sirve	Versiones compatibles	Requisitos
Bamboo	Crea planes de compilación.	6.9.*	
Docker	Las compilaciones nativas pueden utilizar hosts de Docker para la implementación.		Cuando una canalización incluye una imagen de Docker Hub, debe asegurarse de que la imagen tenga una cURL integrada antes de ejecutar la canalización. Cuando se ejecuta la canalización, vRealize Automation Code Stream descarga un archivo binario que utiliza la cURL para ejecutar comandos.
Registro de Docker	Registra imágenes de contenedor para que un host de compilación de Docker pueda extraer imágenes.	2.7.1	
Gerrit	Se conecta a un servidor de Gerrit para revisiones y activaciones	2.14.*	
Git	Activa canalizaciones cuando los desarrolladores actualizan el código y lo insertan en el repositorio.	Git Hub Enterprise 2.1.8 Git Lab Enterprise 11.9.12-ee	
Jenkins	Compila artefactos de código.	1.6.* y 2.*	
Jira	Crea un ticket de JIRA cuando se produce un error en una tarea de canalización.	8.3.*	
Kubernetes	Automatiza los pasos para implementar, ampliar y administrar aplicaciones en contenedores.	1.9.*	
PowerShell	Crea tareas que ejecuten scripts de PowerShell en máquinas con Windows o Linux.	4 y 5	
SSH	Crea tareas que ejecuten scripts SSH en máquinas Windows o Linux.	7.0	

Tabla 6-1. Endpoints compatibles con vRealize Automation Code Stream (continuación)

Endpoint	Para qué sirve	Versiones compatibles	Requisitos
TFS, Team Foundation Server	Administra el código fuente, las compilaciones automatizadas, las pruebas y las actividades relacionadas.	2015 y 2017	
vRealize Orchestrator	Organiza y automatiza los flujos de trabajo del proceso de compilación.	7.* y 8.*	

Ejemplo de código de YAML para un endpoint de GitHub

Este ejemplo de código de YAML define un endpoint de GitHub al que se puede hacer referencia en una tarea de 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
---
```

Cómo se integra vRealize Automation Code Stream con Jenkins

vRealize Automation Code Stream proporciona un complemento de Jenkins, que activa trabajos de Jenkins que compilan y prueban el código fuente. El complemento de Jenkins ejecuta casos de prueba y puede utilizar scripts personalizados.

Para ejecutar un trabajo de Jenkins en la canalización, utilice un servidor de Jenkins y agregue el endpoint de Jenkins en vRealize Automation Code Stream. A continuación, cree una canalización y agréguele una tarea de Jenkins.

Requisitos previos

- Configure un servidor de Jenkins que ejecute la versión 1.561 o posterior.

- Compruebe si es un miembro de un proyecto en vRealize Automation Code Stream. Si no lo es, solicite al administrador de vRealize Automation Code Stream que lo agregue como miembro de un proyecto. Consulte [Cómo agregar un proyecto en vRealize Automation Code Stream](#).
- Compruebe que haya un trabajo en el servidor de Jenkins para que la tarea de canalización pueda ejecutarlo.

Procedimiento

- 1 Agregue y valide un endpoint de Jenkins.
 - a Haga clic en **Endpoints > Nuevo endpoint**.
 - b Seleccione un proyecto y, como tipo de endpoint, seleccione **Jenkins**. A continuación, escriba un nombre y una descripción.
 - c Si este endpoint es un componente fundamental para la empresa en la infraestructura, habilite la opción **Marcar como restringido**.
 - d Introduzca la URL del servidor de Jenkins.

- e Escriba el nombre de usuario y la contraseña para iniciar sesión en el servidor de Jenkins. A continuación, introduzca la información restante.

Tabla 6-2. Información restante para el endpoint de Jenkins

Entrada de endpoint	Descripción
Ruta de la carpeta	<p>Ruta de la carpeta que agrupa los trabajos. Jenkins puede ejecutar todos los trabajos de la carpeta. Puede crear subcarpetas. Por ejemplo:</p> <ul style="list-style-type: none"> ■ <code>folder_1</code> puede incluir <code>job_1</code> ■ <code>folder_1</code> puede incluir <code>folder_2</code>, que a su vez puede incluir <code>job_2</code> <p>Al crear un endpoint para <code>folder_1</code>, la ruta de la carpeta es <code>job/folder_1</code> y el endpoint solo muestra <code>job_1</code>.</p> <p>Para obtener la lista de trabajos de la carpeta secundaria denominada <code>folder_2</code>, debe crear otro endpoint que use la ruta de carpeta como <code>/job/folder_1/job/folder_2/</code>.</p>
URL	URL del host del servidor de Jenkins. Introduzca la URL con el formato <code>protocol://host:port</code> . Por ejemplo: <code>http://192.10.121.13:8080</code>
Intervalo de sondeo	Duración del intervalo en el que vRealize Automation Code Stream sondea el servidor de Jenkins en busca de actualizaciones.
Número de reintentos de solicitud	Número de reintentos de la solicitud de compilación programada para el servidor de Jenkins.
Tiempo de espera entre reintentos	Número de segundos de espera antes de volver a intentar la solicitud de compilación para el servidor de Jenkins.

- f Haga clic en **Validar** y compruebe que el endpoint se conecta a vRealize Automation Code Stream. Si no se conecta, corrija los errores y, a continuación, haga clic en **Guardar**.

- 2 Para compilar el código, cree una canalización y agregue una tarea que use el endpoint de Jenkins.
 - a Haga clic en **Canalizaciones > Nueva canalización > Lienzo en blanco**.
 - b Haga clic en la etapa predeterminada.
 - c En el área Tarea, introduzca un nombre para la tarea.
 - d Seleccione el tipo de tarea como **Jenkins**.
 - e Seleccione el endpoint de Jenkins que creó.

- f En el menú desplegable, seleccione un trabajo del servidor de Jenkins que ejecutará la canalización.
- g Introduzca los parámetros del trabajo.
- h Introduzca el token de autenticación del trabajo de Jenkins.

The screenshot displays the 'Build and Deploy' task configuration in the vRealize Automation Code Stream interface. The task is currently 'Enabled'.

Task Configuration:

- Task name:** Build
- Type:** Jenkins
- Continue On Failure:** ☐
- Execute Task:** ☒ Always ☐ On Condition
- Jenkins Endpoint:** aa
- Job:** add_numbers
- Num1:** 22
- Num2:** 22
- Token:** (empty field)

Output Parameters: status, job, jobId, jobResults, jobUrl

Visual Editor: The left pane shows a visual workflow editor with a stage named 'Stage0' containing two tasks: 'Build' and 'Test', both using the 'Jenkins' provider. Below 'Stage0' is a 'Parallel Task' block and a '+ Stage' button to add a new stage.

Footer: At the bottom, there are buttons for 'SAVE', 'RUN', and 'CLOSE'. A status message indicates 'Last saved a month ago'. A chat icon is located in the bottom right corner.

3 Habilite y ejecute la canalización, y vea cómo se ejecuta.

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 Consulte los detalles y el estado de la ejecución en el panel de control de la canalización.

Puede identificar cualquier error y su causa. También puede ver tendencias sobre la duración de la ejecución de las canalizaciones, las canalizaciones completadas y los errores.

Build and Deploy ACCIONES

Ejecuciones recientes

Ejecución #/Etapas	Estado	Stage0
#15	COMPLETED	
#14	FAILED	
#13	COMPLETED	
#12	COMPLETED	
#11	COMPLETED	
#10	COMPLETED	
#9	COMPLETED	
#8	COMPLETED	
#7	COMPLETED	

COMPLETED FAILED RUNNING WAITING CANCELED ROLLBACK_COMPLETED ROLLBACK_FAILED

Detalles de ejecución

Ejecución#	Estado	Mensaje de estado	Duración	Actualizado
#15	COMPLETED	Execution Completed.	13 segundos	15 ene. 20 14:35:32
#14	FAILED	Stage0.Build: Unable to execute request : www.baidu232.com: Name or service not known	13 segundos	15 ene. 20 14:35:19
#13	COMPLETED	Execution Completed.	13 segundos	15 ene. 20 14:34:14
#12	COMPLETED	Execution Completed.	13 segundos	15 ene. 20 14:34:06
#11	COMPLETED	Execution Completed.	13 segundos	15 ene. 20 14:33:57
#10	COMPLETED	Execution Completed.	12 segundos	15 ene. 20

Resultados

Enhorabuena. Ha integrado vRealize Automation Code Stream con Jenkins tras agregar un endpoint, crear una canalización y configurar una tarea de Jenkins que compila el código.

Ejemplo: Ejemplo de YAML para una tarea de compilación de Jenkins

Para el tipo de tarea de compilación de Jenkins utilizada en este ejemplo, el YAML se asemeja al siguiente código, con las notificaciones activadas:

```
test:
  type: Jenkins
  endpoints:
    jenkinsServer: jenkins
  input:
    job: Add two numbers
  parameters:
    Num1: '23'
    Num2: '23'
```

Pasos siguientes

Revise las demás secciones para obtener más información. Consulte [Capítulo 6 Conectar vRealize Automation Code Stream a endpoints](#).

Cómo se puede integrar vRealize Automation Code Stream con Git

vRealize Automation Code Stream proporciona una manera de activar una canalización si se produce un cambio de código en el repositorio de GitHub, GitLab o Bitbucket. El activador de Git utiliza un endpoint de Git en la rama del repositorio que se desea supervisar. vRealize Automation Code Stream se conecta con el endpoint de Git a través de un webhook.

Para definir un endpoint de Git en vRealize Automation Code Stream, seleccione un proyecto e introduzca la rama del repositorio de Git donde se encuentra el endpoint. El proyecto agrupa la canalización con el endpoint y otros objetos relacionados. Cuando se elige el proyecto en la definición de webhook, se seleccionan el endpoint y la canalización que se desean activar.

Nota Si define un webhook con el endpoint y posteriormente edita el endpoint, no podrá cambiar los detalles del endpoint en el webhook. Para cambiar los detalles del endpoint, debe eliminar y redefinir el webhook con el endpoint. Consulte [Cómo usar el activador de Git en vRealize Automation Code Stream para ejecutar una canalización](#).

Requisitos previos

- Compruebe que puede acceder el repositorio de GitHub, GitLab o Bitbucket al que planea conectarse.

- Compruebe si es un miembro de un proyecto en vRealize Automation Code Stream. Si no lo es, solicite al administrador de vRealize Automation Code Stream que lo agregue como miembro de un proyecto. Consulte [Cómo agregar un proyecto en vRealize Automation Code Stream](#).

Procedimiento

1 Defina un endpoint de Git.

- Haga clic en **Endpoints > Nuevo endpoint**.
- Seleccione un proyecto y seleccione **Git** para el tipo de endpoint. A continuación, escriba un nombre y una descripción.
- Si este endpoint es un componente fundamental para la empresa en la infraestructura, habilite la opción **Marcar como restringido**.
- Seleccione uno de los tipos de servidor Git compatibles.
- Introduzca la URL del repositorio con la puerta de enlace de API del servidor de en la ruta de acceso. Por ejemplo, introduzca **`https://api.github.com/vmware-example/repo-example`**.
- Introduzca la rama del repositorio donde se encuentra el endpoint.
- Seleccione el tipo de autenticación e introduzca el nombre de usuario para GitHub, GitLab o BitBucket. A continuación, introduzca la contraseña, el token privado o la clave privada que van con el nombre de usuario.
 - Contraseña. La contraseña proporciona acceso total al repositorio. También puede crear una variable para la contraseña.

Utilice variables secretas para ocultar y cifrar información confidencial. Utilice variables restringidas para las cadenas, las contraseñas y las URL que deben estar ocultas y cifradas, así como para restringir el uso en ejecuciones. Por ejemplo, utilice una variable secreta para una contraseña o una URL. Puede usar variables de tipo secreto y restringido en cualquier tipo de tarea de la canalización.
 - Token privado. Este token es específico de Git y proporciona acceso a una acción específica. Consulte https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html. También puede crear una variable para el token privado.
 - Clave privada. Esta clave SSH es una clave privada que proporciona acceso a un repositorio específico. Cuando se produce un evento de Git, vRealize Automation Code Stream utiliza esta clave para clonar un repositorio. Consulte <https://help.github.com/articles/reviewing-your-ssh-keys/>.

- Haga clic en **Validar** y compruebe que el endpoint se conecte con vRealize Automation Code Stream.

Si no se conecta, corrija los errores y haga clic en **Crear**.

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

Pasos siguientes

Para obtener más información, revise las demás secciones. Consulte [Cómo usar el activador de Git en vRealize Automation Code Stream para ejecutar una canalización](#).

Cómo se integra vRealize Automation Code Stream con Gerrit

vRealize Automation Code Stream proporciona una forma de activar una canalización si se produce una revisión de código en el proyecto de Gerrit. La definición del activador de Gerrit incluye el proyecto de Gerrit y las canalizaciones que se ejecutarán para diferentes tipos de eventos.

El activador de Gerrit utiliza un agente de escucha de Gerrit en el servidor de Gerrit que desea supervisar. Para definir un endpoint de Gerrit en vRealize Automation Code Stream, seleccione un proyecto e introduzca la URL del servidor de Gerrit. A continuación, especifique el endpoint cuando cree un agente de escucha de Gerrit en ese servidor.

Requisitos previos

- Compruebe que puede acceder al servidor de Gerrit al que planea conectarse.
- Compruebe si es un miembro de un proyecto en vRealize Automation Code Stream. Si no lo es, solicite al administrador de vRealize Automation Code Stream que lo agregue como miembro de un proyecto. Consulte [Cómo agregar un proyecto en vRealize Automation Code Stream](#).

Procedimiento

1 Defina un endpoint de Gerrit.

- a Haga clic en **Configurar > Endpoints** y haga clic en **Nuevo endpoint**.
- b Seleccione un proyecto y, para el tipo de endpoint, seleccione Gerrit. A continuación, escriba un nombre y una descripción.
- c Si este endpoint es un componente fundamental para la empresa en la infraestructura, habilite la opción **Marcar como restringido**.
- d Introduzca la URL del servidor de Gerrit.

Puede proporcionar un número de puerto con la URL o dejar el valor en blanco para utilizar el puerto predeterminado.

- e Introduzca el nombre de usuario y la contraseña del servidor de Gerrit.

Si desea cifrar la contraseña, haga clic en **Crear variable** y seleccione el tipo:

- **Secreto**. La contraseña la resuelve un usuario con cualquier función en el momento de la ejecución.
- **Restringido**. La contraseña la resuelve un usuario con la función de administrador en el momento de la ejecución.

Para valor, introduzca la contraseña que desea que sea segura, como la contraseña de un servidor de Jenkins.

- f Para la clave privada, introduzca la clave SSH utilizada para acceder al servidor de Gerrit de forma segura.

Esta es la clave privada RSA que se encuentra en el directorio .ssh.

- g (opcional) Si la clave privada tiene asociada una frase de contraseña, introduzca la frase de contraseña.

Si desea que se cifre la frase de contraseña, haga clic en **Crear variable** y seleccione el tipo:

- Secreto. La frase de contraseña la resuelve un usuario con cualquier función en el momento de la ejecución.
- Restringido. La frase de contraseña la resuelve un usuario con la función de administrador en el momento de la ejecución.

En valor, introduzca la frase de contraseña que desea que sea segura, como la frase de contraseña de un servidor SSH.

- 2 Haga clic en **Validar** y compruebe que el endpoint de Gerrit en vRealize Automation Code Stream se conecte al servidor de Gerrit.

Si no se conecta, corrija los errores y, a continuación, intente volver a validarlo.

Nuevo endpoint

Proyecto *	test
Tipo *	Gerrit
Nombre *	Gerrit-Demo-Endpoint
Descripción	<input type="text"/>
Marcar como restringido	<input type="checkbox"/> no restringido
Proxy de nube *	Valor predeterminado
URL *	http://example-gerrit.mycompany.com:8080
Username *	CS_user
Password * CREAR VARIABLE
Private Key *	-----BEGIN RSA PRIVATE KEY----- Proc-Type: 4,ENCRYPTED DEK-Info: AES-128-CBC,FOOCE0B6526AF67DC77ADC09620BF92
Pass Phrase ⓘ CREAR VARIABLE

[CREAR](#)
[VALIDAR](#)
[CANCELAR](#)

- 3 Haga clic en **Crear**.

Pasos siguientes

Para obtener más información, revise las demás secciones. Consulte [Cómo usar el activador de Gerrit en vRealize Automation Code Stream para ejecutar una canalización](#).

Cómo se integra vRealize Automation Code Stream con vRealize Orchestrator

vRealize Automation Code Stream se puede integrar con vRealize Orchestrator (vRO) para ampliar su capacidad mediante la ejecución de flujos de trabajo de vRO. vRealize Orchestrator incluye

muchos flujos de trabajo predefinidos que se pueden integrar con herramientas de terceros. Estos flujos de trabajo ayudan a automatizar y administrar los procesos de DevOps, automatizar operaciones en bloque, y más.

Por ejemplo, puede utilizar un flujo de trabajo en una tarea de vRO en la canalización para habilitar un usuario, quitar un usuario, mover máquinas virtuales, realizar integraciones con marcos de pruebas para probar el código mientras se ejecuta la canalización y mucho más. Puede buscar ejemplos de código para flujos de trabajo de vRealize Orchestrator en code.vmware.com.

Con un flujo de trabajo de vRealize Orchestrator, la canalización puede ejecutar una acción a medida que compila, prueba e implementa la aplicación. Puede incluir flujos de trabajo predefinidos en la canalización, o bien crear y utilizar flujos de trabajo personalizados. Cada flujo de trabajo incluye entradas, tareas y salidas.

Para ejecutar un flujo de trabajo de vRO en la canalización, el flujo de trabajo debe aparecer en la lista de flujos de trabajo disponibles de la tarea de vRO que incluya en la canalización.

Antes de que el flujo de trabajo pueda aparecer en la tarea de vRO en la canalización, un administrador debe realizar los siguientes pasos en vRealize Orchestrator:

- 1 Aplicar la etiqueta CODESTREAM al flujo de trabajo de vRO.
- 2 Marcar el flujo de trabajo de vRO como global.

Requisitos previos

- Compruebe que puede acceder a una instancia local de vRealize Orchestrator como administrador. Para obtener ayuda, consulte a su propio administrador y la [documentación de vRealize Orchestrator](#).
- Compruebe si es un miembro de un proyecto en vRealize Automation Code Stream. Si no lo es, solicite al administrador de vRealize Automation Code Stream que lo agregue como miembro de un proyecto. Consulte [Cómo agregar un proyecto en vRealize Automation Code Stream](#).
- En vRealize Automation Code Stream, cree una canalización y agregue una etapa.

Procedimiento

- 1 Como administrador, prepare un flujo de trabajo de vRealize Orchestrator para que la canalización se ejecute.
 - a En vRealize Orchestrator, busque el flujo de trabajo que necesita utilizar en la canalización, como un flujo de trabajo para habilitar un usuario.
Si necesita un flujo de trabajo que no existe, puede crearlo.
 - b En la barra de búsqueda, introduzca **Flujo de trabajo de etiqueta** para buscar el flujo de trabajo con el nombre `Flujo de trabajo de etiqueta`.
 - c En la tarjeta llamada `Flujo de trabajo de etiqueta`, haga clic en **Ejecutar**, que muestra el área de configuración.

- d En el área de texto `Flujo de trabajo etiquetado`, introduzca el nombre del flujo de trabajo que se utilizará en la canalización de vRealize Automation Code Stream y, a continuación, selecciónelo en la lista.
- e En las áreas de texto `Etiqueta` y `Valor`, introduzca `CODESTREAM` en mayúsculas.
- f Haga clic en la casilla de verificación **Etiqueta global**.
- g Haga clic en **Ejecutar**, que asocia la etiqueta con el nombre `CODESTREAM` al flujo de trabajo que debe seleccionar en la canalización de vRealize Automation Code Stream.
- h En el panel de navegación, haga clic en **Flujos de trabajo** y confirme que la etiqueta con el nombre `CODESTREAM` aparezca en la tarjeta de flujo de trabajo que ejecutará la canalización.

Después de iniciar sesión en vRealize Automation Code Stream y agregar una tarea de vRO a la canalización, el flujo de trabajo etiquetado aparece en la lista de flujos de trabajo.

2 En vRealize Automation Code Stream, cree un endpoint para la instancia de vRealize Orchestrator.

- a Haga clic en **Endpoints > Nuevo endpoint**.
- b Seleccione un proyecto.
- c Introduzca un nombre pertinente.
- d Introduzca la URL del endpoint de vRealize Orchestrator.

Utilice este formato: `https://cava-n-01-234.eng.vmware.com:8281`
No utilice el formato: `https://cava-n-01-234.eng.vmware.com:8281/vco/api`
- e Haga clic en **Aceptar certificado** en caso de que la URL que introdujo necesite un certificado.
- f Introduzca el nombre de usuario y la contraseña del servidor de vRealize Orchestrator.

3 Prepare la canalización para ejecutar la tarea de vRO.

- a Agregue una tarea de vRO a la etapa de la canalización.
- b Introduzca un nombre pertinente.
- c En el área `Propiedades de flujo de trabajo`, seleccione el endpoint de vRealize Orchestrator.
- d Seleccione el flujo de trabajo etiquetado como `CODESTREAM` en vRealize Orchestrator.

Si selecciona un flujo de trabajo personalizado que creó, es posible que deba introducir los valores de los parámetros de entrada.

e En **Ejecutar tarea**, haga clic en **Con condición**.

f Introduzca las condiciones aplicables cuando se ejecuta la canalización.

Cuándo ejecutar la canalización...	Seleccionar condiciones...
<p>Con condición</p>	<p>Ejecuta la tarea de canalización solo si la condición definida se evalúa como true. Si la condición es false, se omite la tarea.</p> <p>La tarea de vRO permite incluir una expresión booleana, que usa los siguientes operandos y operadores.</p> <ul style="list-style-type: none"> ■ Variables de canalización, como <code>\${pipeline.variableName}</code>. Utilice únicamente llaves al introducir variables. ■ Variables de salida de tarea, como <code>\${Stage1.task1.machines[0].value.hostIp[0]}</code>. ■ Variables de enlace de canalización predeterminadas, como <code>\${releasePipelineName}</code>. ■ Valores booleanos que no distinguen entre mayúsculas y minúsculas, como <code>true</code>, <code>false</code>, <code>'true'</code> y <code>'false'</code>. ■ Valores enteros o decimales sin comillas. ■ Valores de cadena utilizados con comillas simples o dobles, como <code>"test"</code> y <code>'test'</code>. ■ Tipos de valores de cadena y numéricos, como <code>==</code> <code>Equals</code> y <code>!=</code> <code>Not Equals</code>. ■ Operadores relacionales, como <code>></code>, <code>>=</code>, <code><</code> y <code><=</code>. ■ Lógica booleana, como <code>&&</code> y <code> </code>. ■ Operadores aritméticos, como <code>+</code>, <code>-</code>, <code>*</code> y <code>/</code>. ■ Expresiones anidadas que utilizan corchetes. ■ Las cadenas que incluyen el valor literal <code>ABCD</code> se evalúan como <code>false</code>, por lo que se omite la tarea. ■ No se admiten operadores unarios. <p>Una condición de ejemplo podría ser <code>\${Stage1.task1.output} == "Passed" \${pipeline.variableName} == 39</code></p>
<p>Siempre</p>	<p>Si selecciona Siempre, la canalización ejecuta la tarea sin condiciones.</p>

g Introduzca un mensaje de saludo.

h Haga clic en **Validar tarea** y corrija los errores que se produzcan.

4 Guarde, habilite y ejecute la canalización.

5 Una vez que se ejecute la canalización, examine los resultados.

- a Haga clic en **Ejecuciones**.
- b Haga clic en la canalización.
- c Haga clic en la tarea.
- d Examine los resultados, el valor de entrada y las propiedades.

Puede identificar el identificador de ejecución del flujo de trabajo, quién respondió a la tarea y cuándo, así como cualquier comentario que haya incluido.

Resultados

Enhorabuena. Ha etiquetado un flujo de trabajo de vRealize Orchestrator para su uso en vRealize Automation Code Stream y agregado una tarea de vRO en la canalización de vRealize Automation Code Stream para que ejecute un flujo de trabajo que automatiza una acción en su entorno de DevOps.

Ejemplo: Formato de salida de la tarea de vRO

El formato de salida de una tarea de vRO es similar a este ejemplo.

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

Pasos siguientes

Siga incluyendo tareas de flujo de trabajo de vRO en las canalizaciones para poder automatizar tareas en los entornos de desarrollo, pruebas y producción.

Activar canalizaciones en vRealize Automation Code Stream

7

Puede hacer que vRealize Automation Code Stream desencadene una canalización cuando se produzcan determinados eventos.

Por ejemplo, puede:

- Utilizar el activador de Docker para ejecutar una canalización cuando se cree o se actualice un artefacto.
- Utilizar el activador de Git para activar una canalización cuando los desarrolladores actualicen el código.
- Utilizar el activador de Gerrit para activar una canalización cuando los desarrolladores revisen el código.
- Utilizar el comando `curl` para que Jenkins active la canalización después de que se complete una compilación.

Este capítulo incluye los siguientes temas:

- [Cómo usar el activador de Docker en vRealize Automation Code Stream para ejecutar una canalización de entrega continua](#)
- [Cómo usar el activador de Git en vRealize Automation Code Stream para ejecutar una canalización](#)
- [Cómo usar el activador de Gerrit en vRealize Automation Code Stream para ejecutar una canalización](#)

Cómo usar el activador de Docker en vRealize Automation Code Stream para ejecutar una canalización de entrega continua

Como administrador o desarrollador de DevOps, puede utilizar el activador de Docker en vRealize Automation Code Stream. El activador de Docker ejecuta una canalización de entrega continua (Continuous Delivery, CD) independiente cada vez que se crea o se actualiza un artefacto de compilación. El activador de Docker ejecuta la canalización de entrega continua, que envía el artefacto nuevo o actualizado como una imagen de contenedor a un repositorio de Docker Hub. La canalización de entrega continua se puede ejecutar como parte de las compilaciones automatizadas.

Por ejemplo, para implementar constantemente la imagen de contenedor actualizada a través de la canalización de entrega continua, use el activador de Docker. Cuando la imagen del contenedor se carga en el registro de Docker, el webhook de Docker Hub notifica a vRealize Automation Code Stream que la imagen cambió. Esta notificación activa la canalización de entrega continua para que se ejecute con la imagen de contenedor actualizada y carga la imagen en el repositorio de Docker Hub.

Para usar el activador de Docker, debe realizar varios pasos en vRealize Automation Code Stream.

Tabla 7-1. Cómo usar el activador de Docker

Qué hacer...	Más información sobre esta acción...
Crear un endpoint de registro de Docker	<p>Para que vRealize Automation Code Stream active la canalización, debe tener un endpoint de registro de Docker. Si el endpoint no existe, puede seleccionar una opción que lo cree cuando se agregue el webhook relativo al activador de Docker.</p> <p>El endpoint del registro de Docker incluye la URL que lleva al repositorio de Docker Hub.</p>
Agregar parámetros de entrada a la canalización que insertan automáticamente parámetros de Docker cuando la canalización se ejecuta	<p>Puede insertar parámetros de Docker automáticamente en la canalización. Los parámetros pueden incluir el nombre del propietario del evento de Docker, la imagen, el nombre del repositorio, el espacio de nombres del repositorio y la etiqueta.</p> <p>En la canalización de entrega continua se incluyen los parámetros de entrada que el webhook de Docker envía a la canalización antes de que esta se active.</p>
Crear un webhook de Docker	<p>Al crear el webhook de Docker en vRealize Automation Code Stream, también se crea el webhook correspondiente en Docker Hub. El webhook de Docker en vRealize Automation Code Stream se conecta a Docker Hub a través de la URL incluida en el webhook.</p> <p>Los webhooks se comunican entre sí y activan la canalización cuando un artefacto se crea o actualiza en Docker Hub.</p> <p>Si actualiza o elimina el webhook de Docker en vRealize Automation Code Stream, el webhook de Docker Hub también se actualizará o eliminará.</p>
Agregar y configurar una tarea de Kubernetes en la canalización	<p>Cuando se crea o se actualiza un artefacto en el repositorio de Docker Hub, la canalización se activa. A continuación, implementa el artefacto a través de la canalización en el host de Docker en el clúster de Kubernetes.</p>
Incluir una definición de YAML local en la tarea	<p>La definición de YAML que se usa en la tarea de implementación incluye la imagen de contenedor de Docker y cualquier clave secreta necesaria para extraer la imagen del repositorio para implementarla.</p>

Cuando un artefacto se crea o actualiza en el repositorio de Docker Hub, el webhook de Docker Hub avisa al webhook en vRealize Automation Code Stream, lo que activa la canalización. Se producen las siguientes acciones:

- 1 Docker Hub envía una solicitud POST a la URL del webhook.
- 2 vRealize Automation Code Stream ejecuta el activador de Docker.
- 3 El activador de Docker inicia la canalización de entrega continua.
- 4 La canalización de entrega continua inserta el artefacto en el repositorio de Docker Hub.
- 5 vRealize Automation Code Stream activa su webhook de Docker, lo que ejecuta una canalización de entrega continua que implementa el artefacto en el host de Docker.

En este ejemplo, se crea un endpoint de Docker y un webhook de Docker en vRealize Automation Code Stream que implementan la aplicación en el clúster de Kubernetes de desarrollo. Los pasos incluyen el código de ejemplo de la carga útil que Docker envía a la URL en el webhook, el código de API que este utiliza y el código de autenticación con el token seguro.

Requisitos previos

- Compruebe que existe una canalización de entrega continua en la instancia de vRealize Automation Code Stream. Asimismo, compruebe que incluye una o varias tareas de Kubernetes que implementan la aplicación. Consulte [Capítulo 4 Planificar la compilación, la integración y la distribución nativas del código en vRealize Automation Code Stream](#).
- Confirme que puede acceder a un clúster de Kubernetes existente donde la canalización de entrega continua puede implementar la aplicación para desarrollarla.
- Compruebe si es un miembro de un proyecto en vRealize Automation Code Stream. Si no lo es, solicite al administrador de vRealize Automation Code Stream que lo agregue como miembro de un proyecto. Consulte [Cómo agregar un proyecto en vRealize Automation Code Stream](#).

Procedimiento

- 1 Cree un endpoint de registro de Docker.
 - a Haga clic en **Endpoints**.
 - b Haga clic en **Nuevo endpoint**.
 - c Introduzca un nombre pertinente.
 - d Seleccione el tipo de servidor **Docker Hub**.

- e Introduzca la URL que lleva al repositorio de Docker Hub.
- f Introduzca el nombre y la contraseña que se utilizan para acceder al repositorio.

- 2 En la canalización de entrega continua, configure las propiedades de entrada para que se inserten parámetros de Docker automáticamente cuando la canalización se ejecute.

Destacado	Nombre
☆	DOCKER_REGISTRY_EVENT_OWNER_NAME
☆	DOCKER_REGISTRY_IMAGE
☆	DOCKER_REGISTRY_REPO_NAME
☆	DOCKER_REGISTRY_REPO_NAMESPACE
☆	DOCKER_REGISTRY_TAG

- 3 Cree un webhook de Docker.
 - a Haga clic en **Activadores > Docker**.
 - b Haga clic en **Nuevo webhook de Docker**.
 - c Seleccione un proyecto.
 - d Introduzca un nombre pertinente.
 - e Seleccione el endpoint del registro de Docker.

Si el endpoint no existe aún, haga clic en **Crear endpoint** y créelo.
 - f Seleccione la canalización con los parámetros de Docker insertados para que se active el webhook. Consulte [Paso 2](#).

Si la canalización se configuró con parámetros de entrada personalizados agregados, la lista Parámetros de entrada muestra parámetros y valores. Puede introducir valores para los parámetros de entrada que se transferirán a la canalización con el evento activador. También puede dejar los valores en blanco o utilizar los valores predeterminados si se definieron.

Para obtener más información sobre los parámetros de la pestaña Entrada, consulte [Planificar una compilación nativa de CI/CD en vRealize Automation Code Stream antes de agregar tareas manualmente](#).

- g Introduzca el token de API.

El token de API de CSP lo autentica para las conexiones de API externas con vRealize Automation Code Stream. Para obtener el token de API:

- 1 Haga clic en **Generar token**.
- 2 Introduzca la dirección de correo electrónico asociada con el nombre de usuario y la contraseña y haga clic en **Generar**.

El token que se genera es válido durante seis meses. También se conoce como token de actualización.

- Para mantener el token como una variable para un uso futuro, haga clic en **Crear variable**, introduzca un nombre para la variable y haga clic en **Guardar**.
- Para conservar el token como valor de texto para usarlo en el futuro, haga clic en **Copiar** y pegue el token en un archivo de texto para guardarlo de forma local.

Puede elegir entre crear una variable y almacenar el token en un archivo de texto para utilizarlo en el futuro.

- 3 Haga clic en **Cerrar**.

- h Introduzca la imagen de compilación.

- i Introduzca una etiqueta.

The screenshot shows the 'Docker' configuration page with the 'Webhooks para Docker' tab selected. The form contains the following fields and values:

- URL de webhook: [https://\[redacted\]/codestream/api/registry-webhook-listeners/1c9b3ae4-3...](https://[redacted]/codestream/api/registry-webhook-listeners/1c9b3ae4-3...)
- Proyecto: test
- Nombre: sm-1-Docker-WH
- Descripción: Docker webhook trigger for sm-1
- Registro de Docker: Docker-Register-Endpoint
- Canalización: sm-1
- Token de API: [redacted] with a 'GENERAR TOKEN' button
- Imagen: imagen
- Etiqueta: Etiquetas

At the bottom of the form are two buttons: 'GUARDAR' and 'CANCELAR'.

- j Haga clic en **Guardar**.

La tarjeta del webhook aparece con el webhook de Docker habilitado. Si desea realizar una inserción ficticia en el repositorio de Docker Hub sin activar el webhook de Docker ni ejecutar una canalización, haga clic en **Deshabilitar**.

- 4 En la canalización de entrega continua, configure la tarea de implementación de Kubernetes.
 - a En las propiedades de la tarea de Kubernetes, seleccione el clúster de Kubernetes de desarrollo.
 - b Seleccione la acción **Crear**.

- c Seleccione la **Definición local** del origen de la carga útil.
- d Tras ello, seleccione el archivo YAML local.

Por ejemplo, Docker Hub puede publicar esta definición de YAML local como la carga útil en la URL del 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#\nVOLUME\u0009\u0009[/var/cache/apt-cacher-ng]\nRUN\u0009\u0009apt-get update ; apt-get install -yq apt-cacher-ng\n#\nEXPOSE\n\u0009\u00093142\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"
  }
}
```

La API que crea el webhook en Docker Hub

utiliza este formato: `https://cloud.docker.com/v2/repositories/%3CUSERNAME%3E/%3CREPOSITORY%3E/webhook_pipeline/`

El cuerpo del código JSON se parece a este:

```
{
  "name": "demo_webhook",
}
```

```
"webhooks": [
{
"name": "demo_webhook",
"hook_url": "http://www.google.com"
}
]
```

Para recibir eventos del servidor de Docker Hub, el esquema de autenticación del webhook de Docker que se crea en vRealize Automation Code Stream utiliza un mecanismo de autenticación de lista de permitidos con un token de cadena aleatorio del webhook. Este mecanismo filtra los eventos en función del token seguro, que puede anexar a `hook_url`.

vRealize Automation Code Stream puede comprobar cualquier solicitud procedente del servidor de Docker Hub usando el token seguro configurado. Por ejemplo: `hook_url = IP:Port/pipelines/api/docker-hub-webhooks?secureToken = ""`

- 5 Cree un artefacto de Docker en el repositorio de Docker Hub o actualice uno ya existente.
- 6 Para confirmar que se produjo la activación y ver la actividad en el webhook de Docker, haga clic en **Activadores > Docker > Actividad**.

Docker

GUIDED SETUP

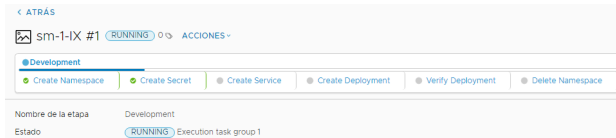
Activity
Webhooks for Docker

Commit Time	Webhook	Image	Tag	Owner	Repository	Pipeline	Execution Status
01/09/2019 10:59 AM	dt1l-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	dt1l-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 Haga clic en **Ejecuciones** y lleve un seguimiento de la canalización mientras se ejecuta.



- 8 Haga clic en la etapa en ejecución y vea las tareas a medida que la canalización se ejecuta.



Resultados

Enhorabuena. Configure el activador de Docker para ejecutar la canalización de entrega continua de forma continua. Ahora la canalización puede cargar artefactos de Docker nuevos y actualizados en el repositorio de Docker Hub.

Pasos siguientes

Compruebe que el artefacto nuevo o actualizado se implementó en el host de Docker, en el clúster de Kubernetes de desarrollo.

Cómo usar el activador de Git en vRealize Automation Code Stream para ejecutar una canalización

Como desarrollador o administrador de DevOps, puede utilizar el activador de Git para integrar vRealize Automation Code Stream con el ciclo de vida de Git. Cuando se realiza un cambio de código en GitHub, GitLab o Bitbucket Enterprise, el evento se comunica con vRealize Automation Code Stream mediante un webhook y activa una canalización para que se ejecute.

Al agregar el webhook de Git en vRealize Automation Code Stream, también se crea un webhook en el repositorio de GitHub, GitLab o Bitbucket. Si actualiza o elimina el webhook más adelante, el webhook de GitHub, GitLab o Bitbucket también se actualizará o eliminará.

La definición de webhook debe incluir un endpoint de Git en la rama del repositorio que desea supervisar. vRealize Automation Code Stream utiliza el endpoint de Git para crear el webhook. Si el endpoint no existe, puede crearlo cuando agregue el webhook. En este ejemplo, se supone que tiene un endpoint de Git predefinido en GitHub.

En este ejemplo, se muestra cómo usar el activador de Git con un repositorio de GitHub, pero los requisitos previos incluyen preparaciones necesarias si se utiliza otro tipo de servidor de Git.

Requisitos previos

- Compruebe si es un miembro de un proyecto en vRealize Automation Code Stream. Si no lo es, solicite al administrador de vRealize Automation Code Stream que lo agregue como miembro de un proyecto. Consulte [Cómo agregar un proyecto en vRealize Automation Code Stream](#).
- Asegúrese de que dispone de un endpoint de Git en la rama de GitHub que desea supervisar. Consulte [Cómo se puede integrar vRealize Automation Code Stream con Git](#).
- Compruebe que tenga derechos para crear un webhook en el repositorio de Git.
- Si va a configurar un webhook en GitLab, cambie la configuración de red predeterminada en GitLab Enterprise para habilitar las solicitudes salientes y permitir la creación de webhooks locales.

Nota Este cambio solo es necesario para GitLab Enterprise. Esta configuración no se aplica a GitHub ni a Bitbucket.

- Inicio sesión en la instancia de GitLab Enterprise como administrador.
 - Vaya a la configuración de red con una URL como, `http://{gitlab-server}/admin/application_settings/network`.
 - Expanda **Solicitudes salientes** y haga clic en:
 - Permitir solicitudes a la red local desde enlaces web y servicios.
 - Permitir solicitudes a la red local desde el enlace del sistema.
- Para las canalizaciones que desee activar, compruebe que estén establecidas las propiedades de entrada para insertar parámetros de Git cuando se ejecute la canalización.

Build and Deploy Habilitado

Área de trabajo **Entrada** Modelo Salida

Parámetros de entrada ⓘ

Parámetros de inserció... ☐ Gerrit ☒ Git ☐ Docker ☐ Ninguno

[AGREGAR](#) [AGREGAR O ELIMINAR PARÁMETROS INSERTADOS](#)

Destacado ⓘ	Nombre
⋮ ☆	GIT_BRANCH_NAME
⋮ ☆	GIT_CHANGE_SUBJECT
⋮ ☆	GIT_COMMIT_ID
⋮ ☆	GIT_EVENT_DESCRIPTION
⋮ ☆	GIT_EVENT_OWNER_NAME
⋮ ☆	GIT_EVENT_TIMESTAMP
⋮ ☆	GIT_REPO_NAME
⋮ ☆	GIT_SERVER_URL

Para obtener información sobre los parámetros de entrada, consulte [Planificar una compilación nativa de CI/CD en vRealize Automation Code Stream antes de agregar tareas manualmente](#).

Procedimiento

- 1 En vRealize Automation Code Stream, haga clic en **Activadores > Git**.
- 2 Haga clic en la pestaña **Webhooks para Git** y, a continuación, haga clic en **Nuevo webhook para Git**.

- a Seleccione un proyecto.
- b Introduzca un nombre descriptivo y una descripción para el webhook.
- c Seleccione un endpoint de Git que esté configurado para la rama que desea supervisar.

Cuando crea el webhook, la definición de este incluye los detalles actuales del endpoint.

- Si más adelante cambia el tipo de Git, el tipo de servidor de Git o la URL del repositorio de Git en el endpoint, el webhook ya no podrá activar una canalización porque intentará acceder al repositorio de Git con los detalles originales del endpoint. Debe eliminar el webhook y volver a crearlo con el endpoint.
- Si más adelante cambia el tipo de autenticación, el nombre de usuario o la contraseña del endpoint, el webhook seguirá funcionando.

Consulte [Cómo se puede integrar vRealize Automation Code Stream con Git](#).

- d (opcional) Introduzca la rama que desea que el webhook supervise.

Si se deja sin especificar, el webhook supervisa la rama configurada para el endpoint de Git.

- e (opcional) Genere un token secreto para el webhook.

Si se utiliza vRealize Automation Code Stream, genera un token de cadena aleatorio para el webhook. A continuación, cuando el webhook recibe datos de eventos de Git, envía los datos con el token secreto. vRealize Automation Code Stream usa la información para determinar si las llamadas proceden del origen esperado, como la instancia, el repositorio y la rama de GitHub configurados. El token secreto proporciona una capa adicional de seguridad que se utiliza para comprobar que los datos de eventos de Git provienen del origen correcto.

f (opcional) Proporcione inclusiones o exclusiones de archivos como condiciones para el activador.

- Las inclusiones de archivos se proporcionan para que, cuando cualquiera de los archivos de una confirmación coincida con los archivos especificados en las expresiones regulares o rutas de inclusión, se activen las canalizaciones. Con una expresión regular especificada, vRealize Automation Code Stream solo activa las canalizaciones con nombres de archivo en el conjunto de cambios que coinciden con la expresión proporcionada. El filtro de expresiones regulares es útil cuando se configura un activador para varias canalizaciones en un solo repositorio.
- Las exclusiones de archivos se proporcionan para que, cuando todos los archivos de una confirmación coincidan con los archivos especificados en las expresiones regulares o las rutas de exclusión, no se activen las canalizaciones.
- Cuando se activa la prioridad de exclusión, garantiza que las canalizaciones no se activen aunque alguno de los archivos de una confirmación coincida con los archivos especificados en las expresiones regulares o las rutas de exclusión. De forma predeterminada, esta opción está desactivada.

Si se cumplen tanto las condiciones de inclusión como las de exclusión, no se activan las canalizaciones.

En el siguiente ejemplo, tanto las inclusiones como las exclusiones de archivos son condiciones para el activador.

The screenshot shows a configuration panel for file inclusion and exclusion. It has a title 'Archivo' with a help icon. Below it, there are two sections: 'Inclusiones' and 'Exclusiones'. Each section has a dropdown menu for the file type (PLAIN or REGEX) and a text input field for the file path or regex. To the right of each input field are minus and plus icons. At the bottom, there is a toggle switch for 'Priorizar exclusión' which is currently turned off.

Section	File Type	File Path / Regex	Actions
Inclusiones	PLAIN	runtime/src/main/a.java	-
	REGEX	[[a-z A-Z]+[/]] [a-z A-Z]]+	- +
Exclusiones	PLAIN	runtime/pom.xml	-
	PLAIN	runtime/demo.yaml	- +

Priorizar exclusión: ☐

- Para las inclusiones de archivos, una confirmación con cualquier cambio en `runtime/src/main/a.java` o cualquier archivo Java activará las canalizaciones definidas en la configuración de eventos.
 - Para las exclusiones de archivos, una confirmación con cambios solo en ambos archivos no activará las canalizaciones configuradas en las configuraciones de eventos.
- g Para el evento de Git, seleccione una solicitud **Push** o **Pull**.

h Introduzca el token de API.

El token de API de CSP lo autentica para las conexiones de API externas con vRealize Automation Code Stream. Para obtener el token de API:

- 1 Haga clic en **Generar token**.
- 2 Introduzca la dirección de correo electrónico asociada con el nombre de usuario y la contraseña y haga clic en **Generar**.

El token que se genera es válido durante seis meses. También se conoce como token de actualización.

- Para mantener el token como una variable para un uso futuro, haga clic en **Crear variable**, introduzca un nombre para la variable y haga clic en **Guardar**.
- Para conservar el token como valor de texto para usarlo en el futuro, haga clic en **Copiar** y pegue el token en un archivo de texto para guardarlo de forma local.

Puede elegir entre crear una variable y almacenar el token en un archivo de texto para utilizarlo en el futuro.

- 3 Haga clic en **Cerrar**.

i Seleccione la canalización para que se active el webhook.

Si la canalización se configuró con parámetros de entrada personalizados agregados, la lista Parámetros de entrada muestra parámetros y valores. Puede introducir valores para los parámetros de entrada que se transferirán a la canalización con el evento activador. También puede dejar los valores en blanco o utilizar los valores predeterminados si se definieron.

Para obtener información sobre los parámetros de entrada de inserción automática de los activadores de Git, consulte [Requisitos previos](#).

j Haga clic en **Crear**.

El webhook aparece como una nueva tarjeta.

3 Haga clic en la tarjeta del webhook.

Cuando el formulario de datos de webhook vuelve a aparecer, verá una URL de webhook agregada a la parte superior del formulario. El webhook de Git se conectará al repositorio de GitHub a través de la URL del 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

Endpoint `DemoApp-Git`

Branch ⓘ `master`

Secret token ⓘ * `GYH0cBWZx4dUn47Y/KA8H/BOKts=` GENERATE

File ⓘ

Inclusions	Value	
--Select--		+
Exclusions	Value	+

Prioritize Exclusion ☐

Trigger

For Git ☒ PUSH ☐ PULL REQUEST

API token *

.....
✖
CREATE VARIABLE
GENERATE TOKEN

Pipeline * `CICD-2` ⓘ

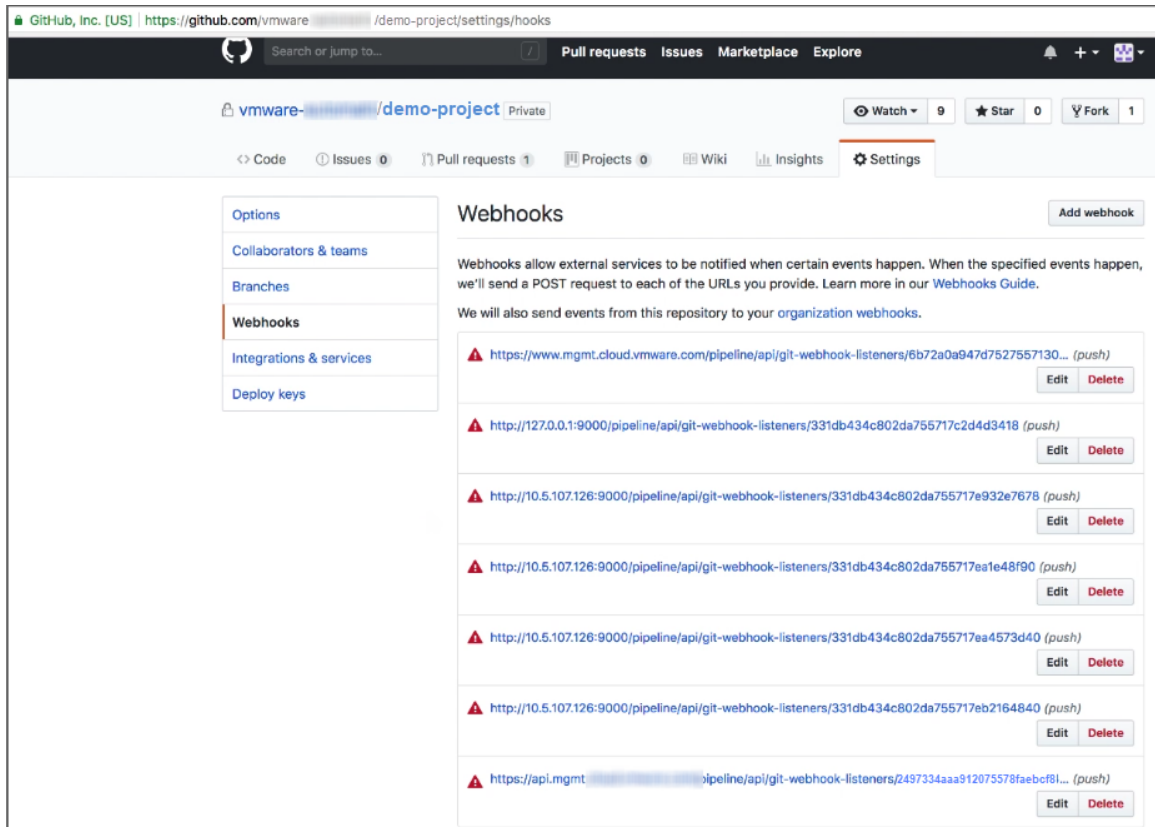
Comments

Execution trigger delay ⓘ `1` ⓘ

SAVE CANCEL

- 4 En una nueva ventana del explorador, abra el repositorio de GitHub que se conectó a través del webhook.
 - a Para ver el webhook que agregó en vRealize Automation Code Stream, haga clic en la pestaña **Configuración** y seleccione **Webhooks**.

En la parte inferior de la lista de webhooks, verá la misma URL de webhook.



- b Para realizar un cambio de código, haga clic en la pestaña **Código** y seleccione un archivo para editar en la rama que desea modificar. Confirme el cambio.
 - c Para comprobar que la URL del webhook funcione, haga clic en la pestaña **Configuración** y seleccione **Webhooks** nuevamente.

En la parte inferior de la lista de webhooks, se mostrará una marca de verificación verde junto a la URL del webhook.



- 5 Vuelva a vRealize Automation Code Stream para ver la actividad en el webhook de Git. Haga clic en **Activadores > Git > Actividad**.

En Estado de ejecución, compruebe que la ejecución de la canalización se haya iniciado.

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 Haga clic en **Ejecuciones** para realizar un seguimiento de la canalización mientras se ejecuta.

Puede hacer clic en el botón de actualización para ver la ejecución de la canalización.

Ejecuciones 18 elementos

+ NUEVA EJECUCIÓN

CONFIGURACIÓN GUIADA

CICD-2#1

RUNNING

Etapas: 1 2 3

ACCIONES

Por el 17 ene. 2020 11:48:51

Running Stage0

Entrada : adc63c0058eac16cecefe973b...

Salida : -

Resultados

Enhorabuena. Utilizó correctamente un activador de Git para ejecutar una canalización.

Cómo usar el activador de Gerrit en vRealize Automation Code Stream para ejecutar una canalización

Como desarrollador o administrador de DevOps, puede utilizar el activador de Gerrit para integrar vRealize Automation Code Stream con el ciclo de vida de revisión de código de Gerrit. El evento acciona una canalización para que se ejecute al crear un conjunto de revisiones, publicar borradores, combinar cambios de código en el proyecto de Gerrit o insertar cambios directamente en la rama de Git.

Al agregar el activador para Gerrit, seleccionará un agente de escucha de Gerrit y un proyecto de Gerrit en el servidor, y configurará los eventos de Gerrit. En este ejemplo, se configura un agente de escucha de Gerrit en primer lugar y, a continuación, se utiliza ese agente en un activador de Gerrit con dos eventos en tres canalizaciones diferentes.

Requisitos previos

- Compruebe si es un miembro de un proyecto en vRealize Automation Code Stream. Si no lo es, solicite al administrador de vRealize Automation Code Stream que lo agregue como miembro de un proyecto. Consulte [Cómo agregar un proyecto en vRealize Automation Code Stream](#).
- Asegúrese de que tiene un endpoint de Gerrit configurado en vRealize Automation Code Stream. Consulte [Cómo se integra vRealize Automation Code Stream con Gerrit](#).

- Para que se activen las canalizaciones, compruebe que estén establecidas las propiedades de entrada para insertar parámetros de Gerrit cuando se ejecute la canalización.

Build and Deploy Habilitado

Área de trabajo **Entrada** Modelo Salida

Parámetros de entrada ⓘ

Parámetros de inser... ☒ Gerrit ☐ Git ☐ Docker ☐ Ninguno

AGREGAR **AGREGAR O ELIMINAR PARÁMETROS INSERTADOS**

Destacado ⓘ	Nombre
☆	GERRIT_BRANCH
☆	GERRIT_CHANGE_COMMIT_MESSAGE
☆	GERRIT_CHANGE_FILELIST
☆	GERRIT_CHANGE_ID
☆	GERRIT_CHANGE_NUMBER
☆	GERRIT_CHANGE_OWNER
☆	GERRIT_CHANGE_OWNER_EMAIL
☆	GERRIT_CHANGE_OWNER_NAME
☆	GERRIT_CHANGE_OWNER_USERNAME
☆	GERRIT_CHANGE_SUBJECT

Para obtener información sobre los parámetros de entrada, consulte [Planificar una compilación nativa de CI/CD en vRealize Automation Code Stream antes de agregar tareas manualmente](#).

Procedimiento

- 1 En vRealize Automation Code Stream, haga clic en **Activadores > Gerrit**.
- 2 (opcional) Haga clic en la pestaña **Agentes de escucha** y, a continuación, haga clic en **Nuevo agente de escucha**.

Nota Omite este paso si el agente de escucha de Gerrit que tiene pensado usar para el activador de Gerrit ya está definido.

- a Seleccione un proyecto.
- b Introduzca un nombre para el agente de escucha de Gerrit.
- c Seleccione un endpoint de Gerrit.

d Introduzca el token de API.

El token de API de CSP lo autentica para las conexiones de API externas con vRealize Automation Code Stream. Para obtener el token de API:

- 1 Haga clic en **Generar token**.
- 2 Introduzca la dirección de correo electrónico asociada con el nombre de usuario y la contraseña y haga clic en **Generar**.

El token que se genera es válido durante seis meses. También se conoce como token de actualización.

- Para mantener el token como una variable para un uso futuro, haga clic en **Crear variable**, introduzca un nombre para la variable y haga clic en **Guardar**.
- Para conservar el token como valor de texto para usarlo en el futuro, haga clic en **Copiar** y pegue el token en un archivo de texto para guardarlo de forma local.

Puede elegir entre crear una variable y almacenar el token en un archivo de texto para utilizarlo en el futuro.

3 Haga clic en **Cerrar**.

Si creó una variable, el token de API muestra el nombre de la variable que introdujo. Si ha copiado el token, el token de API muestra el token enmascarado.

e Para validar los detalles del endpoint y el token, haga clic en **Validar**.

El token caduca tras 90 días.

f Haga clic en **Crear**.g En la tarjeta del agente de escucha, haga clic en **Conectar**.

El agente de escucha inicia la supervisión de toda la actividad en el servidor de Gerrit y escucha los activadores habilitados en ese servidor. Para dejar de escuchar un activador en ese servidor, desactive el activador.

Nota Para actualizar un endpoint de Gerrit que está conectado a un agente de escucha, se debe desconectar antes de actualizar el endpoint.

- Haga clic en **Configurar > Activadores > Gerrit**.
- Haga clic en la pestaña **Agentes de escucha**.
- Haga clic en **Desconectar** en el agente de escucha que está conectado al endpoint que desea actualizar.

3 Haga clic en la pestaña **Activadores** y, a continuación, haga clic en **Nuevo activador de Gerrit**.

4 Seleccione un proyecto del servidor de Gerrit.

5 Introduzca un nombre.

El nombre del activador de Gerrit debe ser exclusivo.

6 Seleccione un agente de escucha de Gerrit configurado.

vRealize Automation Code Stream utiliza la selección de agente de escucha de Gerrit para proporcionar una lista de proyectos de Gerrit que están disponibles en el servidor.

7 Seleccione un proyecto del servidor de Gerrit.

8 Introduzca la rama en el repositorio que se va a supervisar.

9 (opcional) Proporcione inclusiones o exclusiones de archivos como condiciones para el activador.

- Proporcione inclusiones de archivos para que se activen las canalizaciones. Cuando cualquiera de los archivos de una confirmación coincida con los archivos especificados en las rutas de inclusión o las expresiones regulares, se activarán las canalizaciones. Con una expresión regular especificada, vRealize Automation Code Stream solo activa canalizaciones con nombres de archivo en el conjunto de cambios que coinciden con la expresión proporcionada. El filtro de expresiones regulares es útil cuando se configura un activador para varias canalizaciones en un solo repositorio.
- Proporcione exclusiones de archivos para impedir que se activen las canalizaciones. Cuando todos los archivos de una confirmación coinciden con los archivos especificados en las rutas de exclusión o en las expresiones regulares, no se activarán las canalizaciones.
- Al activar Priorizar exclusión, garantiza que no se activan las canalizaciones. Las canalizaciones no se activarán aunque alguno de los archivos de una confirmación coincida con los archivos especificados en las rutas de exclusión o en las expresiones regulares. De forma predeterminada, esta opción está desactivada.

Si se cumplen tanto las condiciones de inclusión como las de exclusión, no se activan las canalizaciones.

En el siguiente ejemplo, tanto las inclusiones como las exclusiones de archivos son condiciones para el activador.

The screenshot shows a configuration window titled 'Archivo' with a sub-section 'Inclusiones' and 'Exclusiones'. Under 'Inclusiones', there are two entries: one with 'PLAIN' type and path 'runtime/src/main/a.java', and another with 'REGEX' type and path '([a-z A-Z]+/[a-z A-Z])+'. Under 'Exclusiones', there are two entries: one with 'PLAIN' type and path 'runtime/pom.xml', and another with 'PLAIN' type and path 'runtime/demo.yaml'. At the bottom, there is a toggle switch for 'Priorizar exclusión' which is currently turned off.

- Para las inclusiones de archivos, una confirmación con cualquier cambio en `runtime/src/main/a.java` o cualquier archivo Java activará las canalizaciones configuradas en la configuración de eventos.

- Para las exclusiones de archivos, una confirmación con cambios solo en ambos archivos no activará las canalizaciones configuradas en las configuraciones de eventos.

10 Haga clic en **Nueva configuración**.

- Para un evento de Gerrit, seleccione **Conjunto de revisiones creado**, **Borrador publicado** o **Cambio combinado**. O bien, para una inserción directa en Git que omita Gerrit, seleccione **Inserción de Git directa**.
- Seleccione la canalización que desea activar.

Si la canalización se configuró con parámetros de entrada personalizados agregados, la lista Parámetros de entrada muestra parámetros y valores. Puede introducir valores para los parámetros de entrada que se transferirán a la canalización con el evento activador. Opcionalmente, puede dejar los valores en blanco o utilizar los valores predeterminados.

Nota Si se van a definir valores predeterminados:

- Cualquier valor que introduzca para los parámetros de entrada sobrescribirán los valores predeterminados que se hayan definido en el modelo de canalización.
 - Los valores predeterminados que se utilicen para configurar el activador no se actualizarán si se cambian los valores de los parámetros en el modelo de canalización.
-

Para obtener información sobre los parámetros de entrada de inserción automática de los activadores de Gerrit, consulte [Requisitos previos](#).

- Para las opciones **Conjunto de revisiones creado**, **Borrador publicado** y **Cambio combinado**, algunas acciones aparecen con etiquetas de forma predeterminada. Puede cambiar la etiqueta o añadir comentarios. A continuación, cuando se ejecuta la canalización, la etiqueta o el comentario aparecen en la pestaña Actividad como la **Acción realizada** para la canalización.
- Haga clic en **Guardar**.

Para agregar varios eventos de activador en diversas canalizaciones, vuelva a hacer clic en **Nueva configuración**.

En el siguiente ejemplo, puede ver eventos de tres canalizaciones:

- Si se produce un evento de **Cambio combinado** en el proyecto de Gerrit, se activará la canalización **Gerrit-Pipeline**.
- Si se produce un evento de **Conjunto de revisiones creado** en el proyecto de Gerrit, se activarán las canalizaciones **Gerrit-Trigger-Pipeline** y **Gerrit-Demo-Pipeline**.

Gerrit CONFIGURACIÓN GUIADA

Actividad **Activadores** Agentes de escucha

Proyecto * test! ⓘ

Nombre * Gerrit-Demo-Trigger

Agente de escucha de ... * Gerrit-Demo-Listener v

Proyecto de Gerrit * --Seleccionar proyecto de Gerrit-- v

Rama * master

Archivo ⓘ

Inclusiones --Seleccionar-- v Valor ⓘ

Exclusiones --Seleccionar-- v Valor ⓘ

Priorizar exclusión ☐

[+ NUEVA CONFIGURACIÓN](#)

Tipo de evento	Canalización	Etiqueta
⋮ Change Merged	Gerrit-Pipeline	
⋮ Patchset Created	Gerrit-Trigger-Pipeline	Verified
⋮ Patchset Created	Gerrit-Demo-Pipeline	Verified

3 elementos

11 Haga clic en **Crear**.

El activador de Gerrit aparece como una tarjeta nueva en la pestaña **Activadores** y está establecido en **Deshabilitado** de forma predeterminada.

12 En la tarjeta del activador, haga clic en **Habilitar**.

Cuando el activador está habilitado, utiliza el agente de escucha de Gerrit para empezar a supervisar los eventos que se producen en la rama del proyecto de Gerrit.

Cuando se crea el activador, se incluye el repositorio donde se confirma el código. Si desea crear un activador con las mismas condiciones de inclusión o exclusión del archivo, pero con otro repositorio, puede hacer clic en **Acciones > Clonar** en la tarjeta del activador. A continuación, haga clic en **Abrir** en el nuevo activador y cambie los parámetros.

Resultados

Enhorabuena. Configuró correctamente un activador de Gerrit con dos eventos en tres canalizaciones diferentes.

Pasos siguientes

Después de confirmar un cambio de código en el proyecto de Gerrit, consulte la pestaña **Actividad** del evento de Gerrit en vRealize Automation Code Stream. Compruebe que la lista de actividades incluya las entradas correspondientes a cada ejecución de la canalización configurada en el activador. Cuando se produzca un evento, solo se ejecutarán las canalizaciones del activador de Gerrit que estén relacionadas con el tipo de evento en cuestión. En este ejemplo, si se crea un conjunto de revisiones, solo se ejecutarán las canalizaciones **Gerrit-Trigger-Pipeline** y **Gerrit-Demo-Pipeline**.

La información de las columnas de la pestaña **Actividad** describe cada evento del activador de Gerrit. Puede elegir las columnas que se van a mostrar.

- Si el activador era una inserción de Git directa, las columnas **Cambiar asunto** y **Ejecución** están vacías.
- La columna **Activador de Gerrit** muestra el activador que creó el evento.

- El agente de escucha está desactivado de forma predeterminada. Si se selecciona, se muestra el agente de escucha de Gerrit que recibió el evento. Un agente de escucha puede asociarse con varios activadores.
- La opción Tipo de activador está desactivada de forma predeterminada. Al seleccionarla, muestra si el activador se activó de forma manual o 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	Drivank sphyank@vm	test1
⋮	Nov 12, 2019, 12:50:04 PM	19570 /6	1111Dummy	Gerrit-Pipeline #2	WAITING	Stage0.Task0: Execution Waiting for User Action.		Drivank sphyank@vm	test1
⋮			1111Dummy	Gerrit-Demo-Pipeline #1	FAILED	Stage0.Task0: User Operation request has been rejected by fritz.	Verified -1	Drivank sphyank@vm	test1
⋮			1111Dummy	Gerrit-Trigger-Pipeline #1	WAITING	Stage0.Task0: Execution Waiting for User Action.		Drivank sphyank@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 la actividad de una ejecución completada o con errores, haga clic en los tres puntos situados a la izquierda de cualquier entrada de la pantalla Actividad.

- Si la canalización no se ejecuta debido a un error en el modelo de canalización u otro problema, corrija el error y seleccione **Volver a ejecutar** para volver a ejecutarla.
- Si la canalización no se ejecuta debido a un problema de conectividad de red o de otro tipo, seleccione **Reanudar** para reiniciar la misma ejecución de la canalización. Si lo hace de esta manera, ahorrará tiempo de ejecución.
- Utilice **Ver ejecución** para ir a la pantalla Ejecución. Consulte [Cómo ejecutar una canalización y ver los resultados](#).

- Utilice **Eliminar** para eliminar la entrada de la pantalla Actividad.

Si un evento de Gerrit no activa una canalización, puede hacer clic en el botón Activar manualmente, e introducir el nombre del activador de Gerrit y el identificador de cambio.

Supervisar canalizaciones en vRealize Automation Code Stream



Como desarrollador o administrador de DevOps, puede usar paneles de control de vRealize Automation Code Stream para supervisar las tendencias y los resultados de una ejecución de canalización. Puede utilizar los paneles de control de canalizaciones predeterminados para supervisar una sola canalización o crear paneles de control personalizados para supervisar varias canalizaciones.

Descripción de los paneles de control de canalizaciones

Un panel de control de canalización es una vista de los resultados de una canalización específica que se ejecutó, como las tendencias, los errores principales y los cambios realizados correctamente. vRealize Automation Code Stream crea el panel de control de canalización al crear una canalización.

Descripción de los paneles de control personalizados

Un panel de control personalizado es una vista de los resultados de una o varias canalizaciones que se ejecutaron. Cree el panel de control personalizado y agregue widgets para mostrar los resultados que desea ver. Por ejemplo, puede crear un panel de control de todo el proyecto con KPI y métricas recopilados de varias canalizaciones. Si se indica un error o una advertencia de ejecución, puede usar el panel de control para solucionar el error.

Este capítulo incluye los siguientes temas:

- [Cómo realizar un seguimiento de los indicadores clave de rendimiento de la canalización en vRealize Automation Code Stream](#)

Cómo realizar un seguimiento de los indicadores clave de rendimiento de la canalización en vRealize Automation Code Stream

Como desarrollador o administrador de DevOps, necesita conocer el rendimiento de las canalizaciones en vRealize Automation Code Stream. Debe saber si las canalizaciones logran publicar eficazmente el código a través de las etapas de desarrollo, pruebas y producción.

Para obtener información detallada, puede usar el panel de control predeterminado para la canalización o un panel de control personalizado.

- Las métricas de canalización incluyen estadísticas, como los tiempos medios, que se encuentran disponibles en el panel de control de la canalización.
- Para ver las métricas de varias canalizaciones, utilice los paneles de control personalizados.

Puede hacer que vRealize Automation Code Stream mida los tiempos medios de recuperación, distribución o error de una canalización a través del tiempo, y mostrar las tendencias de esos tiempos medios.

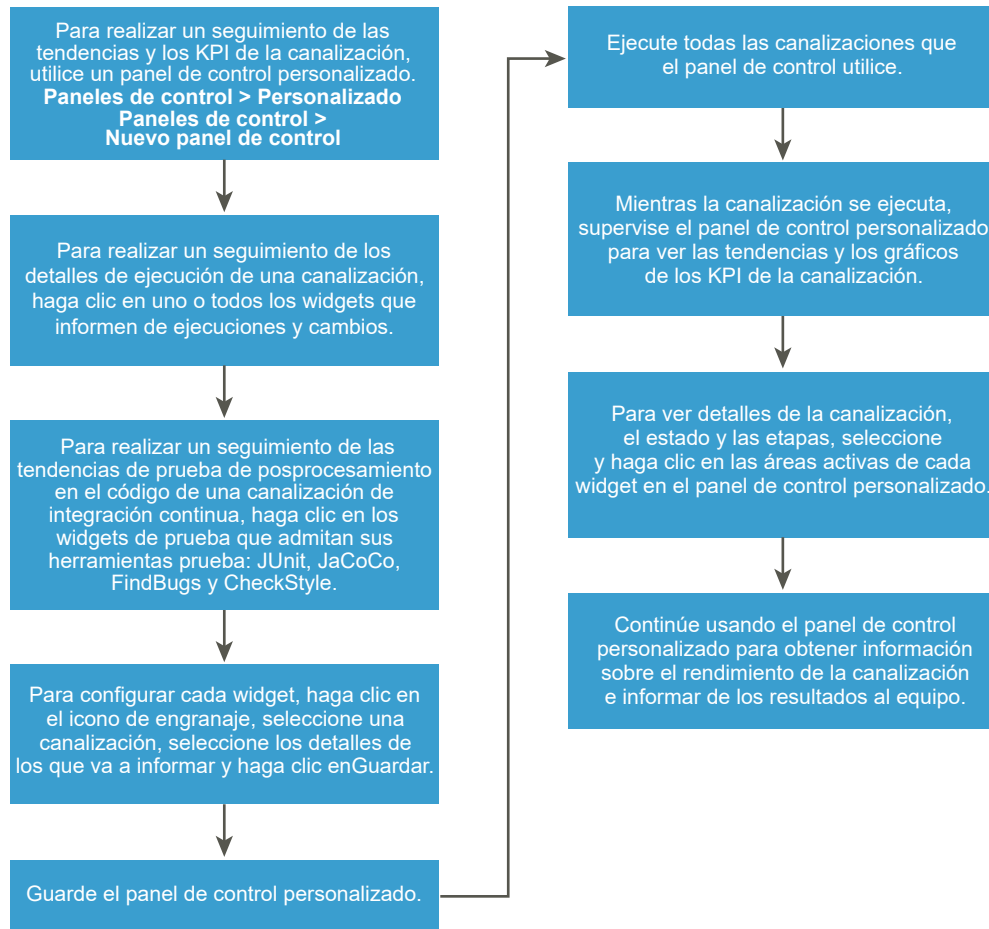
Tabla 8-1. Medición de tiempos medios

Qué se mide...	Qué significa...
CI promedio	Tiempo promedio transcurrido en la fase de integración continua, el cual se mide por el tiempo en el tipo de tarea de CI.
Tiempo medio para distribuir una canalización	Tiempo promedio que requiere la canalización para distribuir una actualización después de que se activa.
Tiempo medio entre implementaciones correctas	Tiempo entre implementaciones correctas, lo que indica la frecuencia con la que se actualiza un entorno de producción.
Tiempo medio para generar un error en una canalización	Tiempo que demora una canalización en generar un error después de que se activa.
Tiempo medio para recuperarse de un error de canalización	Tiempo promedio para distribuir una canalización correcta después de que se produce un error en la canalización. Se mide el tiempo entre el error de un criterio de compilación o prueba y la siguiente compilación que produce una ejecución correcta de la canalización, promediado en una semana o un mes.

También puede hacer que vRealize Automation Code Stream muestre las principales tareas y etapas con errores en una canalización. Esta medición notifica la cantidad y el porcentaje de errores de los entornos de desarrollo y posdesarrollo de cada canalización y proyecto, promediados en una semana o un mes. Puede ver los errores principales para solucionar problemas en el proceso de automatización de la publicación.

Por ejemplo, puede configurar la pantalla para una duración determinada (por ejemplo, los últimos siete días) y anotar las principales tareas con errores durante ese período. Si realiza un cambio en el entorno o la canalización y vuelve a ejecutar la canalización, revise las principales tareas con errores de un período más extenso (por ejemplo, los últimos 14 días); es posible que las tareas principales con errores hayan cambiado. Con ese resultado, sabrá que el cambio en el proceso de automatización de la publicación mejoró la tasa de éxito de ejecución de la canalización.

Para realizar un seguimiento de las tendencias y los indicadores clave de rendimiento de las canalizaciones mediante un panel de control personalizado, agregue widgets al panel de control y configúrelos para informar sobre las canalizaciones.



Requisitos previos

- Asegúrese de tener una o más canalizaciones. En la interfaz de usuario, haga clic en **Canalizaciones**.
- Para las canalizaciones que desea supervisar, compruebe que se hayan ejecutado correctamente. Haga clic en **Ejecuciones**.

Procedimiento

- 1 Para crear un panel de control personalizado, haga clic en **Paneles de control > Paneles de control personalizados > Nuevo panel de control**.

- 2 Para personalizar el panel de control de modo que informe sobre tendencias e indicadores clave de rendimiento específicos para la canalización, haga clic en un widget.

Por ejemplo, para mostrar detalles sobre el estado de la canalización, las etapas, las tareas, la duración de la ejecución y quién la ejecutó, haga clic en el widget **Detalles de ejecución**. Opcionalmente, para una canalización de integración continua (Continuous Integration, CI), puede realizar un seguimiento de las tendencias del posprocesamiento mediante los widgets para JUnit, JaCoCo, FindBugs y CheckStyle.

IX KPIS ACCIONES ▾

Detalles de ejecución

Ejecución# ▾	Estado ▾	Mensaje de estado ▾	Todas las tareas	TaskID (StageID)	Duración ▾
⋮ #22	WAITING	StageID.TaskID: Execution Waiting for User Action.	■	■	2 horas, 48 minutos, 39 segundos
⋮ #21	COMPLETED	Execution Completed.	■	■	17 segundos

Elementos por página 10 1-10 de 22 elementos

- 3 Configure cada widget que agregue.
 - a En el widget, haga clic en el icono de engranaje.
 - b Seleccione una canalización, establezca las opciones disponibles y seleccione las columnas que desea mostrar.
 - c Para guardar la configuración del widget, haga clic en **Guardar**.
 - d Para guardar el panel de control personalizado, haga clic en **Guardar** y, a continuación, haga clic en **Cerrar**.
- 4 Para mostrar más información sobre la canalización, haga clic en las áreas activas de los widgets.

Por ejemplo, en el widget **Detalles de ejecución**, haga clic en una entrada de la columna Estado para mostrar más información sobre la ejecución de la canalización. Asimismo, en el widget **Último cambio correcto**, puede mostrar un resumen de la etapa y la tarea de la canalización, para lo que debe hacer clic en el vínculo activo.

Resultados

Enhorabuena. Se creó un panel de control personalizado que supervisa las tendencias y los KPI de las canalizaciones.

Pasos siguientes

Siga supervisando el rendimiento de las canalizaciones en vRealize Automation Code Stream y comparta los resultados con el administrador y los equipos para seguir mejorando el proceso para publicar las aplicaciones.

Más información sobre Code Stream

9

Hay muchas maneras de que los desarrolladores y administradores de DevOps obtengan más información sobre vRealize Automation Code Stream y lo que pueden hacer por los usuarios.

Utilice esta documentación para obtener más información sobre las canalizaciones y sus ejecuciones, cómo agregar endpoints, cómo agregar proyectos y muchos más.

Conozca los permisos que las funciones proporcionan. Aprenda a usar recursos restringidos y a exigir aprobaciones en las canalizaciones. Consulte [Cómo se administran las autorizaciones y el acceso de los usuarios en vRealize Automation Code Stream](#).

Aprecie el valor de las búsquedas detectando dónde se encuentra un trabajo o componente específico dentro de una canalización, ejecución o endpoint.

Este capítulo incluye los siguientes temas:

- [Qué es la función de búsqueda en vRealize Automation Code Stream](#)
- [Más recursos para desarrolladores y administradores de vRealize Automation Code Stream](#)

Qué es la función de búsqueda en vRealize Automation Code Stream

Utilice la función de búsqueda para saber dónde se encuentran elementos específicos u otros componentes. Por ejemplo, es posible que desee buscar canalizaciones activadas o desactivadas, ya que, si una canalización está desactivada, no se puede ejecutar.

¿Qué se puede buscar?

Puede realizar búsquedas en:

- Proyectos
- Endpoints
- Canalizaciones
- Ejecuciones
- Paneles de control de canalizaciones y paneles de control personalizados
- Servidores y activadores de Gerrit

- Webhooks de Git
- Webhooks de Docker

Puede realizar búsquedas de filtro basadas en columnas en las siguientes áreas:

- Operaciones de usuario
- Variables
- Activadores de Gerrit, Git y Docker

Puede realizar búsquedas de filtro basadas en cuadrículas en la página de actividad de cualquier activador.

Cómo funcionan las búsquedas

Los criterios de búsqueda varían según la página en la que se encuentre. Cada página tiene distintos criterios de búsqueda.

Dónde buscar	Criterios que se usarán en la búsqueda
Paneles de control de canalizaciones	Proyecto, Nombre, Descripción, Etiquetas, Vínculo
Paneles de control personalizados	Proyecto, Nombre, Descripción, Vínculo (UUID de un elemento en el panel de control)
Ejecuciones	Nombre, Comentarios, Motivo, Etiquetas, Índice, Estado, Proyecto, Mostrar, Ejecutado por, Ejecutado por mí, Vínculo (UUID de la ejecución) y Parámetros de entrada, Parámetros de salida o Mensaje de estado con este formato: <key>:<value>
Canalizaciones	Nombre, Descripción, Estado, Etiquetas, Creado por, Creado por mí, Actualizado por, Actualizado por mí, Proyecto
Proyectos	Nombre, Descripción
Endpoints	Nombre, Descripción, Tipo, Actualizado por, Proyecto
Activadores de Gerrit	Nombre, Estado, Proyecto
Servidores de Gerrit	Nombre, URL del servidor, Proyecto
Webhooks de Git	Nombre, Tipo de servidor, Repositorio, Rama, Proyecto

Donde:

- Vínculo es el UUID de una canalización, ejecución o widget en un panel de control.
- Algunos ejemplos de Parámetro de entrada, Parámetro de salida y Mensaje de estado, y su notación incluyen:
 - Notación: `input.<inputKey>:<inputValue>`

Ejemplo: `input.GERRIT_CHANGE_OWNER_EMAIL:joe_user`

- Notación: `output.<outputKey>:<outputValue>`

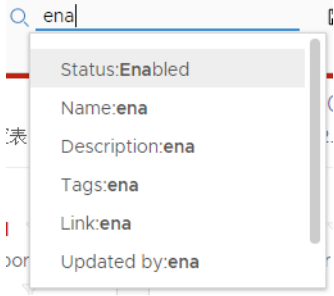

Ejemplo: **`output.BuildNo:29`**

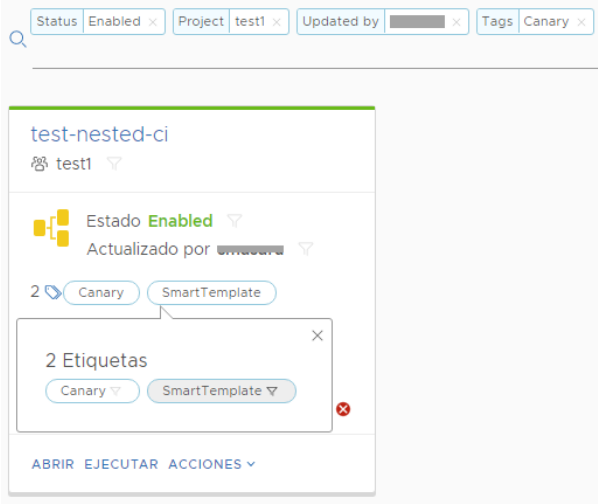
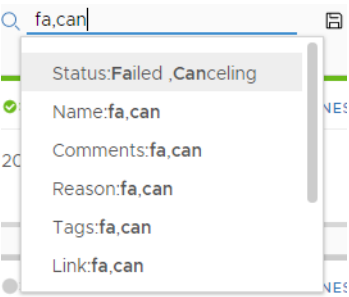
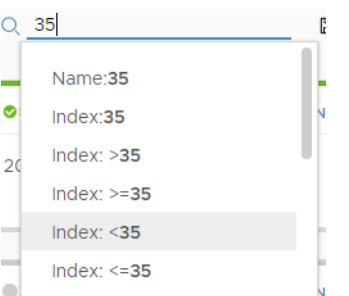
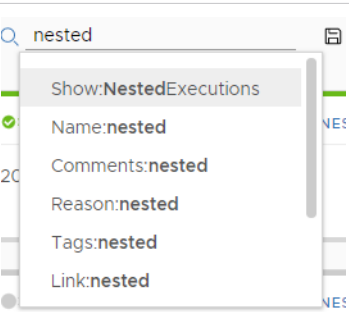
- Notación: `statusMessage:<value>`

Ejemplo: **`statusMessage:Execution failed`**

- Estado depende de la página de búsqueda.
 - En el caso de las ejecuciones, los valores posibles son completado, error, error de reversión o cancelado.
 - Para las canalizaciones, los valores de estado posibles son: habilitado, deshabilitado o publicado.
 - En el caso de los activadores, los valores posibles son habilitado o deshabilitado.
- Las opciones Ejecutado, Creado o Actualizado por mí hacen referencia a uno mismo, el usuario que inició sesión.

La búsqueda aparece en la parte superior derecha de cada página válida. Al empezar a escribir en el cuadro de búsqueda en blanco, vRealize Automation Code Stream conoce el contexto de la página y sugiere opciones de búsqueda.

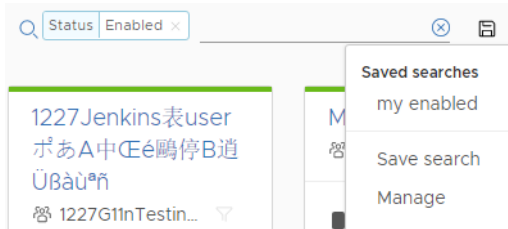
Métodos que se pueden usar para buscar	Cómo especificarlos
<p>Escribir una parte del parámetro de búsqueda.</p> <p>Por ejemplo, escriba ena para agregar un filtro de estado que enumere todas las canalizaciones habilitadas.</p>	 <p>The screenshot shows a search input field with the text 'ena'. A dropdown menu is open, displaying several suggestions: 'Status:Enabled', 'Name:ena', 'Description:ena', 'Tags:ena', 'Link:ena', and 'Updated by:ena'.</p>
<p>Agregar un filtro para reducir el número de elementos encontrados.</p> <p>Por ejemplo, escriba Tes para agregar un filtro de nombre. El filtro funciona como un operador AND con el filtro Estado:deshabilitado para mostrar solo las canalizaciones desactivadas con Tes en el nombre.</p>	 <p>The screenshot shows a search bar with the text 'Tes'. Above the search bar, there are two filter buttons: 'Status' and 'Disabled'. Below the search bar, a list of search results is displayed, including 'Name:Tes', 'Description:Tes', 'Tags:Tes', 'Link:Tes', 'Project:218 test result', and 'Updated by:Tes'.</p>

Métodos que se pueden usar para buscar	Cómo especificarlos
<p>Haga clic en el icono de filtro de las propiedades de una canalización o ejecución para reducir la cantidad de elementos que se muestran.</p> <ul style="list-style-type: none"> ■ Para las canalizaciones, cada opción Estado, Etiquetas, Proyecto y Actualizado por tiene un icono de filtro. ■ Para las ejecuciones, cada opción Etiquetas, Ejecutado por y Mensaje de estado tiene un icono de filtro. <p>Por ejemplo, en la tarjeta de canalización, haga clic en el icono para agregar el filtro de la etiqueta Plantilla inteligente a los filtros existentes para: Estado:habilitado, Proyecto:prueba, Actualizado por:usuario y Etiquetas:Canary.</p>	
<p>Usar una coma como separador para incluir todos los elementos en dos estados de ejecución.</p> <p>Por ejemplo, escriba fa,can para crear un filtro de estado que funcione como operador OR para enumerar todas las ejecuciones con error o canceladas.</p>	
<p>Escribir un número para incluir todos los elementos dentro de un rango de índices.</p> <p>Por ejemplo, escriba 35 y seleccione < para enumerar todas las ejecuciones con un número de índice inferior a 35.</p>	
<p>Las canalizaciones que se modelan como tareas se convierten en ejecuciones anidadas y no aparecen junto con las demás ejecuciones de forma predeterminada.</p> <p>Para mostrar las ejecuciones anidadas, escriba nested y seleccione el filtro Mostrar.</p>	

Cómo guardar una búsqueda favorita

Las búsquedas favoritas se pueden guardar para usarlas en cada página haciendo clic en el icono de disco situado junto al área de búsqueda.

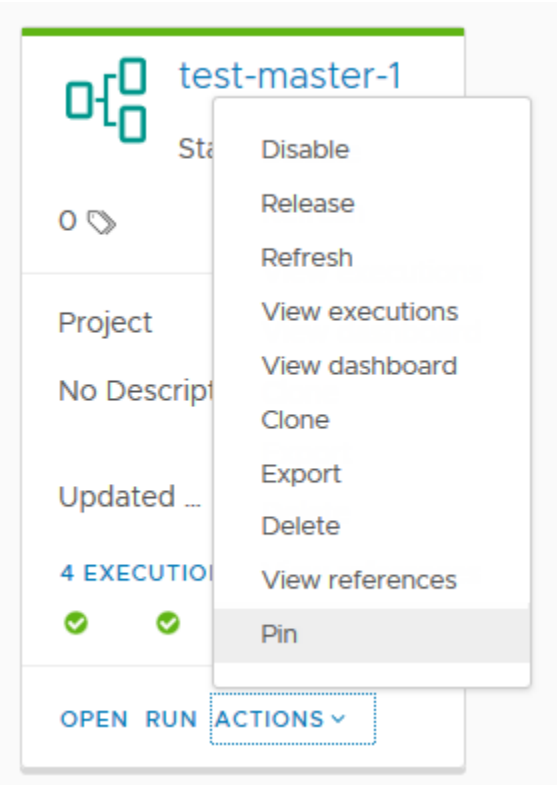
- Para guardar una búsqueda, escriba los parámetros de búsqueda y haga clic en el icono para asignar un nombre a la búsqueda, como **my enabled**.
- Tras guardarla, haga clic en el icono para acceder a ella. También puede seleccionar **Administrar** para cambiar de nombre, eliminar o mover la búsqueda en la lista de búsquedas guardadas.



Las búsquedas están vinculadas a su nombre de usuario y solo aparecen en las páginas relacionadas con la búsqueda. Por ejemplo, si guardó una búsqueda llamada **my enabled** relativa a **Estado:habilitado** en la página de canalizaciones, la búsqueda **my enabled** no estará disponible en la página de activadores de Gerrit, aunque **Estado:habilitado** sea una búsqueda válida para un activador.

¿Se puede guardar una canalización favorita?

Si tiene una canalización o un panel de control favoritos, puede anclarlos para que se muestren siempre en la parte superior de la página de canalizaciones o de paneles de control. En la tarjeta de canalización, haga clic en **Acciones > Anclar**.



Más recursos para desarrolladores y administradores de vRealize Automation Code Stream

Como administrador de DevOps, o como desarrollador, puede obtener más información sobre vRealize Automation Code Stream.

Tabla 9-1. Más recursos para los administradores de DevOps

Para obtener información sobre...	Consulte estos recursos...
<p>Otras formas en que los administradores de DevOps pueden utilizar vRealize Automation Code Stream:</p> <ul style="list-style-type: none">■ Configure las canalizaciones para automatizar las pruebas y la publicación de aplicaciones nativas de nube.■ Automatice y pruebe el código fuente del desarrollador en producción a través de pruebas.■ Configure las canalizaciones para que los desarrolladores prueben los cambios antes de confirmarlos en la rama principal.■ Realice un seguimiento de las métricas clave de la canalización.	<p>vRealize Automation Code Stream</p> <ul style="list-style-type: none">■ Documentación de vRealize Automation■ Sitio web de productos de vRealize Automation <p>Talleres de VMware</p> <ul style="list-style-type: none">■ Utilice la Comunidad de vRealize Automation.■ Use la Zona de aprendizaje de VMware.■ Busque los Blogs de VMware.■ Pruebe los Laboratorios de talleres de VMware.

Tabla 9-2. Más recursos para desarrolladores

Para obtener información sobre...	Consulte estos recursos...
<p>Otras maneras en que los desarrolladores pueden usar vRealize Automation Code Stream:</p> <ul style="list-style-type: none"> ■ Use imágenes públicas y privadas del registro para crear entornos de nuevas aplicaciones o servicios. ■ Configure los entornos de desarrollo para poder crear bifurcaciones desde la última compilación estable. ■ Actualice entornos de desarrollo con los cambios de código y artefactos más recientes. ■ Pruebe los cambios de código no confirmados frente a las últimas compilaciones estables de otros servicios de dependencias. ■ Reciba una notificación cuando un cambio confirmado en una canalización de CICD principal altere otros servicios. 	<p>vRealize Automation Code Stream</p> <ul style="list-style-type: none"> ■ Documentación de vRealize Automation ■ Sitio web de productos de vRealize Automation <p>Talleres de VMware</p> <ul style="list-style-type: none"> ■ Utilice la Comunidad de vRealize Automation. ■ Use la Zona de aprendizaje de VMware. ■ Busque los Blogs de VMware. ■ Pruebe los Laboratorios de talleres de VMware.