Pipeline como código

Tabla de contenido

Pipeline como código

Introducción

Pipeline as Code describe un conjunto de características que permiten a los usuarios de Jenkins definir procesos de trabajo canalizados con código, almacenados y versionados en un repositorio de origen. Estas características permiten a Jenkins descubrir, gestionar y ejecutar trabajos para múltiples repositorios y ramas de origen, eliminando la necesidad de creación y gestión de trabajos manuales.

Para utilizar *Pipeline como código* , los proyectos deben contener un archivo denominado `Jenkinsfile`en la raíz del repositorio, que contiene un "script de canalización".

Además, uno de los trabajos de habilitación debe configurarse en Jenkins:

- *Tubería multibranco* : crear múltiples ramas de un *solo* repositorio de forma automática
- *Carpetas de organización* : escanee una **organización de GitHub** o un **equipo de Bitbucket** para descubrir los repositorios de una organización, creando automáticamente trabajos de *interconexión de multibranquios* administrados para ellos.

Fundamentalmente, los repositorios de una organización se pueden ver como una jerarquía, donde cada repositorio puede tener elementos secundarios de ramas y solicitudes de extracción.

Ejemplo de estructura de repositorio

```
+--- GitHub Organization
    +--- Project 1
        +--- master
        +--- feature-branch-a
        +--- feature-branch-b
    +--- Project 2
        +--- master
        +--- pull-request-1
        +--- etc...

Prior to _Multibranch Pipeline_ jobs and _Organization Folders_,
link:https://wiki.jenkins-ci.org/display/JENKINS/CloudBees+Folders+Plugin[Folders]
could be used to create this hierarchy in Jenkins by organizing repositories
into folders containing jobs for each individual branch.
```

Las carpetas multinivel y las *carpetas de organización* eliminan el proceso manual mediante la detección de sucursales y repositorios, respectivamente, y la creación de carpetas apropiadas con trabajos en Jenkins automáticamente.

El Jenkinsfile

La presencia de `Jenkinsfile`en la raíz de un repositorio lo hace elegible para que Jenkins administre y ejecute automáticamente trabajos basados en ramas de repositorio.

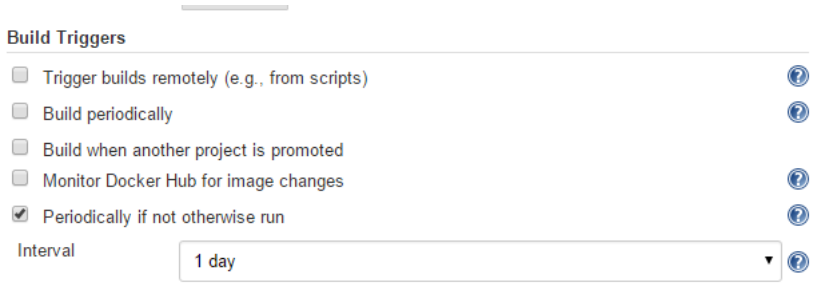
El `Jenkinsfile`debería contener una secuencia de comandos Pipeline, especificando los pasos para ejecutar el trabajo. El script tiene todo el poder de Pipeline disponible, desde algo tan simple como invocar un constructor de Maven, hasta una serie de pasos interdependientes, que han coordinado la ejecución paralela con las fases de implementación y validación.

Una manera simple de comenzar con Pipeline es usar el *generador de fragmentos* disponible en la pantalla de configuración para un trabajo de Jenkins *Pipeline* . Con el *generador de fragmentos* , puede crear un script de canalización a través de las listas desplegables en otros trabajos de Jenkins.

Cálculo de carpetas

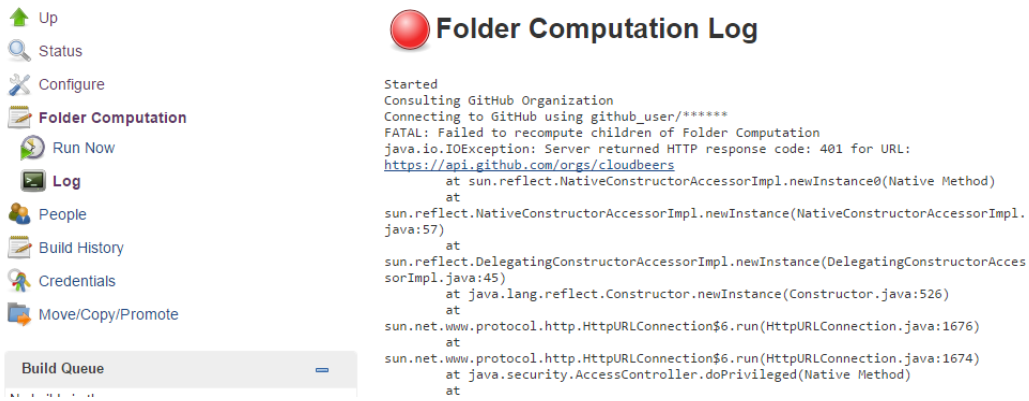
Los proyectos de *líneas múltiples* y las *carpetas de organización* extienden la funcionalidad de la carpeta existente al presentar carpetas *calculadas* . Las carpetas calculadas ejecutan automáticamente un proceso para administrar los contenidos de la carpeta. Este cálculo, en el caso de los proyectos *Multibranch Pipeline* , crea elementos secundarios para cada rama elegible dentro del niño. Para *Carpetas de organización* , el cálculo rellena elementos secundarios para repositorios como *Tuberías multibranquiales* individuales .

El cálculo de carpetas puede realizarse automáticamente a través de devoluciones de llamadas webhook, ya que las sucursales y repositorios se crean o eliminan. El cálculo también puede ser desencadenado por el *Disparador de compilación* definido en la configuración, que ejecutará automáticamente una tarea de cálculo después de un período de inactividad (esto se ejecuta de manera predeterminada después de un día).



La información sobre la última ejecución del cálculo de la carpeta está disponible en la sección **Computación de** la **carpeta** .

El registro del último intento de calcular la carpeta está disponible desde esta página. Si el cálculo de la carpeta no da como resultado un conjunto esperado de repositorios, el registro puede tener información útil para diagnosticar el problema.



Configuración

Tanto los proyectos de *oleoductos multinivel* como las *carpetas de organización* tienen opciones de configuración para permitir la selección precisa de repositorios. Estas características también permiten la selección de dos tipos de credenciales para usar al conectarse a los sistemas remotos:

- *escanear* credenciales, que se utilizan para acceder a las API de GitHub o Bitbucket
- credenciales de *pago* , que se utilizan cuando el repositorio se clona desde el sistema remoto; puede ser útil elegir una clave SSH o *"- anónimo -"* , que usa las credenciales predeterminadas configuradas para el usuario del sistema operativo

Si está utilizando una *Organización GitHub* , debe crear un token de acceso a GitHub para evitar almacenar su contraseña en Jenkins y evitar problemas al usar la API de GitHub. Al usar un token de acceso de GitHub, debe usar un nombre de *usuario* estándar *con* credenciales de *contraseña* , donde el nombre de usuario es el mismo que el nombre de usuario de GitHub y la contraseña es su token de acceso.

Proyectos de oleoductos multibranquios

Los proyectos de *oleoductos multibranquios* son una de las características de habilitación fundamentales para *Pipeline as Code* . Los cambios en el procedimiento de compilación o implementación pueden evolucionar con los requisitos del proyecto y el trabajo siempre refleja el estado actual del proyecto. También le permite configurar diferentes trabajos para diferentes ramas del mismo proyecto, o renunciar a un trabajo si corresponde. El `Jenkinsfile`en el directorio raíz de una bifurcación o solicitud de extracción identifica un proyecto multibranquio.

Los proyectos de *Tuberías múltiples* exponen el nombre de la sucursal que se está construyendo con la `BRANCH_NAME`variable de entorno y proporcionan un `checkout scm` comando Tubería especial , que garantiza la comprobación de la confirmación específica que originó el Jenkinsfile. Si el Jenkinsfile necesita verificar el repositorio por alguna razón, asegúrese de usarlo `checkout scm`, ya que también cuenta con repositorios de origen alternativos para manejar cosas como solicitudes de extracción.

Para crear una *interconexión de líneas múltiples* , vaya a: *Nuevo elemento → Interconexión multinivel* . Configure la fuente de SCM según corresponda. Hay opciones para muchos tipos diferentes de repositorios y servicios, incluidos Git, Mercurial, Bitbucket y GitHub. Si usa GitHub, por ejemplo, haga clic en **Agregar fuente** , seleccione GitHub y configure el propietario apropiado, las credenciales de exploración y el repositorio.

Otras opciones disponibles para los proyectos de *Multibranch Pipeline* son:

- **Punto final de la API** : un **punto final** alternativo de la API para usar un GitHub Enterprise autohospedado
- **Credenciales de pago: credenciales** alternativas para usar al verificar el código (clonación)
- **Incluir ramas** : una expresión regular para especificar ramas para incluir
- **Excluir ramas** : una expresión regular para especificar ramas para excluir; tenga en cuenta que esto tendrá prioridad sobre incluye
- **Estrategia de propiedad** : si es necesario, defina propiedades personalizadas para cada rama

Después de configurar estos elementos y guardar la configuración, Jenkins escaneará automáticamente el repositorio e importará las ramas apropiadas.

Carpetas de organización

Las carpetas de organización ofrecen una forma conveniente de permitir que Jenkins administre automáticamente qué repositorios se incluyen automáticamente en Jenkins. Particularmente, si su organización utiliza las *Organizaciones GitHub* o los *Equipos Bitbucket* , cada vez que un desarrollador crea un nuevo repositorio con una `Jenkinsfile`, Jenkins lo detectará automáticamente y creará un proyecto *Multibranch Pipeline* para él. Esto alivia la necesidad de administradores o desarrolladores de crear proyectos manualmente para estos nuevos repositorios.

Para crear una *carpeta de organización* en Jenkins, vaya a: **Nuevo elemento → Organización GitHub** o **Nuevo elemento → Equipo de Bitbucket** y siga los pasos de configuración para cada elemento, asegurándose de especificar *Credenciales de escaneo* apropiadas y un **propietario** específico para la Organización GitHub o el nombre del Equipo Bitbucket , respectivamente.

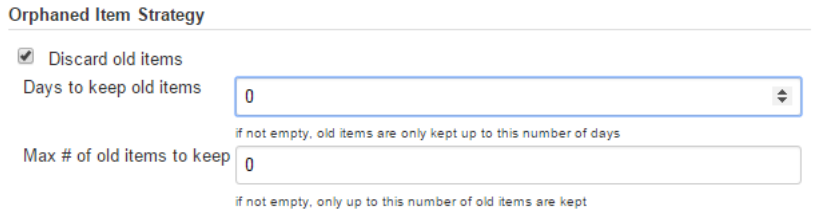
Otras opciones disponibles son:

- **Patrón de nombre de repositorio** : una expresión regular para especificar qué repositorios se **incluyen**
- **Punto final de la API** : un **punto final** alternativo de la API para usar un GitHub Enterprise autohospedado
- **Credenciales de pago: credenciales** alternativas para usar al verificar el código (clonación)

Después de configurar estos elementos y guardar la configuración, Jenkins escaneará automáticamente la organización e importará los repositorios apropiados y las ramas resultantes.

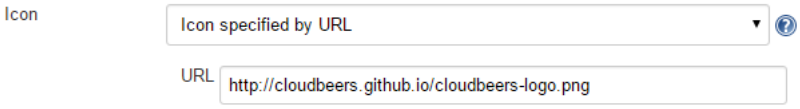
Estrategia de elementos huérfanos

Las carpetas calculadas pueden eliminar elementos inmediatamente o dejarlos en función de una estrategia de retención deseada. De forma predeterminada, los elementos se eliminarán tan pronto como el cálculo de la carpeta determine que ya no están presentes. Si su organización requiere que estos elementos permanezcan disponibles durante un período de tiempo más largo, simplemente configure la Estrategia de elementos huérfanos de manera adecuada. Puede ser útil mantener los elementos para examinar los resultados de compilación de una rama después de haberlos eliminado, por ejemplo.



Icono y vista de estrategia

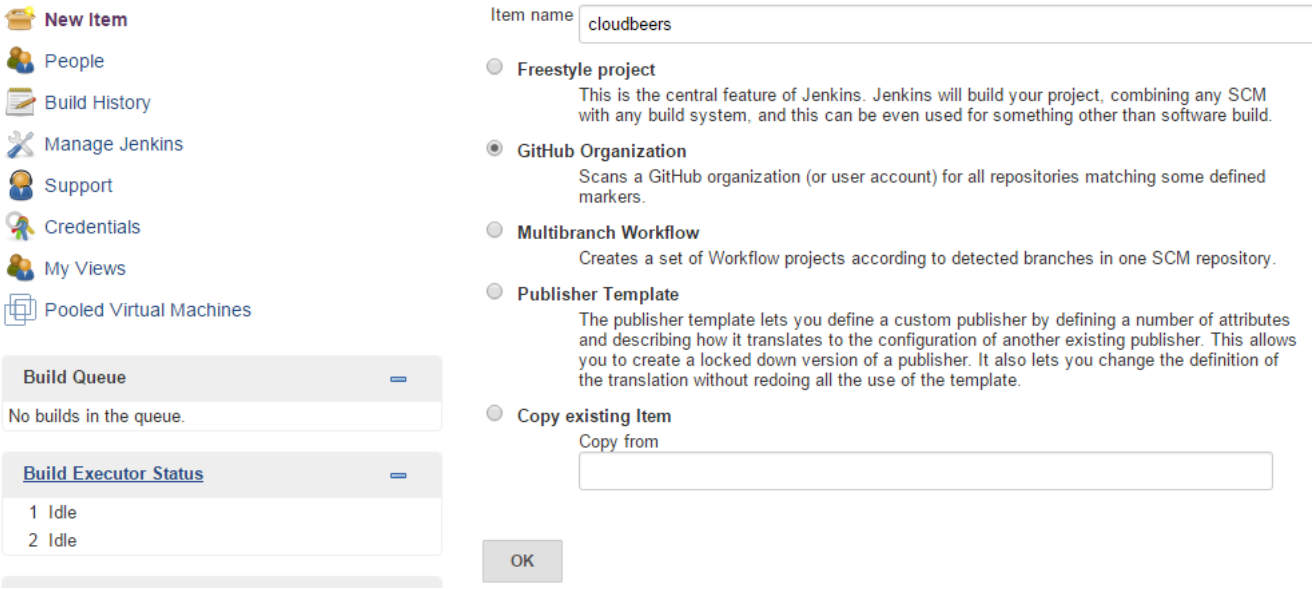
También puede configurar un icono para usar en la visualización de carpetas. Por ejemplo, podría ser útil mostrar un estado agregado de las compilaciones secundarias. Alternativamente, puede hacer referencia al mismo ícono que usa en su cuenta de organización de GitHub.



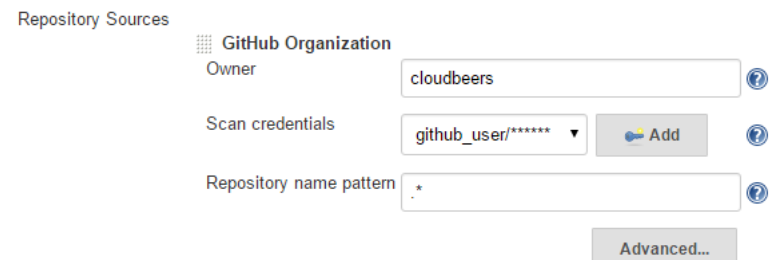
Ejemplo

Para demostrar el uso de una carpeta de organización para administrar repositorios, utilizaremos la organización ficticia: CloudBeers, Inc.

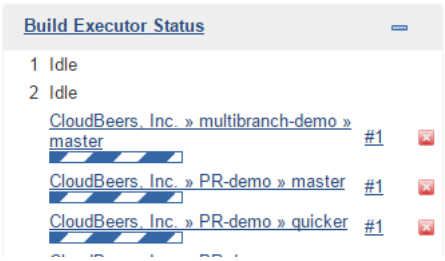
Ir a **Nuevo Artículo** . Introduzca *cloudbeers* para el nombre del elemento. Seleccione **Organización GitHub** y haga **clic** en **Aceptar** .



Opcionalmente, ingrese un mejor nombre descriptivo para la *Descripción* , como *CloudBeers GitHub* . En la sección *Fuentes* del *repositorio* , complete la sección de "Organización GitHub". Asegúrese de que el **propietario** coincida exactamente con el nombre de la Organización GitHub, en nuestro caso debe ser: *cloudbeers* . De manera predeterminada, es el mismo valor que se ingresó para el nombre del elemento en el primer paso. A continuación, seleccione o agregue nuevas "Escanear credenciales": ingresaremos nuestro nombre de usuario de GitHub y token de acceso como contraseña.



Después de guardar, se ejecutará "Computación de carpetas" para buscar repositorios elegibles, seguidos por compilaciones de múltiples filas.



Actualice la página después de que se ejecute el trabajo para asegurarse de que la vista de los repositorios se haya actualizado.

A este punto, ha terminado con la configuración básica del proyecto y ahora puede explorar sus repositorios importados. También puede investigar los resultados de las tareas que se ejecutan como parte del *cálculo* inicial de la *carpeta* .

Entrega continua con Pipeline

Introducción

La entrega continua permite a las organizaciones entregar software con menor riesgo. La ruta hacia la entrega continua comienza modelando la canalización de entrega de software utilizada dentro de la organización y luego centrándose en la automatización de todo. La retroalimentación temprana y directa, habilitada por la automatización de la tubería, permite la entrega de software más rápidamente que los métodos tradicionales de entrega.

Jenkins es la navaja suiza en la cadena de herramientas de entrega de software. Los desarrolladores y el personal de operaciones (DevOps) tienen diferentes mentalidades y utilizan diferentes herramientas para realizar sus respectivos trabajos. Dado que Jenkins se integra con una gran variedad de conjuntos de herramientas, sirve como punto de intersección entre los equipos de desarrollo y operaciones.

Muchas organizaciones han estado orquestando oleoductos con plugins Jenkins existentes desde hace varios años. A medida que aumentan su sofisticación de automatización y su propia experiencia en Jenkins, las organizaciones inevitablemente quieren ir más allá de los simples canales y crear flujos complejos específicos para su proceso de entrega.

Estos usuarios de Jenkins requieren una característica que trate las tuberías complejas como un objeto de primera clase, y así se desarrolló el plugin Pipeline .

Requisitos previos

La entrega continua es un proceso, en lugar de una herramienta, y requiere una mentalidad y una cultura que deben impregnarse de arriba hacia abajo dentro de una organización. Una vez que la organización ha aceptado la filosofía, la siguiente y más difícil es mapear el flujo de software a medida que avanza desde el desarrollo hasta la producción.

La raíz de tal canalización siempre será una herramienta de orquestación como Jenkins, pero hay algunos requisitos clave que una parte tan integral de la canalización debe satisfacer antes de que se le puedan asignar procesos críticos para la empresa:

- **Recuperación ante desastres de tiempo de inactividad cero o bajo** : un compromiso, al igual que un héroe mítico, enfrenta desafíos más difíciles y más largos a medida que avanza por la tubería. No es inusual ver ejecuciones de tuberías que duran días. Un hardware o una falla de Jenkins en el sexto día de una tubería de siete días tiene serias consecuencias para la entrega a tiempo de un producto.
- **La auditoría se ejecuta y la capacidad de depuración** : a los administradores les gusta ver el flujo de ejecución exacto a través de la canalización, para que puedan depurar fácilmente los problemas.

Para garantizar que una herramienta pueda escalar con una organización y automatizar adecuadamente las tuberías de entrega existentes sin cambiarlas, la herramienta también debe ser compatible con:

- **Tuberías complejas** : las **tuberías de** entrega suelen ser más complejas que los ejemplos canónicos (proceso lineal: Dev → Prueba → Implementar, con un par de operaciones en cada etapa). Los administradores de compilación necesitan construcciones que ayuden a paralelizar partes del flujo, ejecutar bucles, realizar reintentos, etc. Dicho de otra manera, los administradores de compilación quieren que las construcciones de programación definan las tuberías.

- **Intervenciones manuales** : los oleoductos cruzan los límites intraorganizacionales que requieren transferencias e intervenciones manuales. Los gerentes de compilación buscan capacidades tales como detener una tubería para que un ser humano intervenga y tomar decisiones manuales.

El complemento Pipeline permite a los usuarios crear una tubería de ese tipo a través de un nuevo tipo de trabajo llamado Pipeline. La definición de flujo se captura en una secuencia de comandos Groovy, agregando así capacidades de flujo de control como bucles, bifurcaciones y reintentos. Pipeline permite etapas con la opción de establecer concurrencias, evitando que múltiples construcciones de la misma interconexión intenten acceder al mismo recurso al mismo tiempo.

Conceptos

Tipo de trabajo de canalización

Solo hay un trabajo para capturar todo el proceso de entrega de software en una organización. Por supuesto, aún puede conectar dos tipos de trabajos Pipeline si lo desea. Un tipo de trabajo Pipeline utiliza una DSL basada en Groovy para definiciones de trabajo. El DSL ofrece la ventaja de definir trabajos de forma programática:

```
node('linux'){
  git url: 'https://github.com/jglick/simple-maven-project-with-tests.git'
  def mvnHome = tool 'M3'
  env.PATH = "${mvnHome}/bin:${env.PATH}"
  sh 'mvn -B clean verify'
}
```

Etapas

Los límites intraorganizacionales (o conceptuales) se capturan a través de una primitiva llamada "etapas". Una tubería de implementación consta de varias etapas, donde cada etapa posterior se basa en la anterior. La idea es gastar la menor cantidad posible de recursos al principio del proceso y encontrar problemas obvios, en lugar de gastar una gran cantidad de recursos informáticos para algo que finalmente se descubre que está roto.

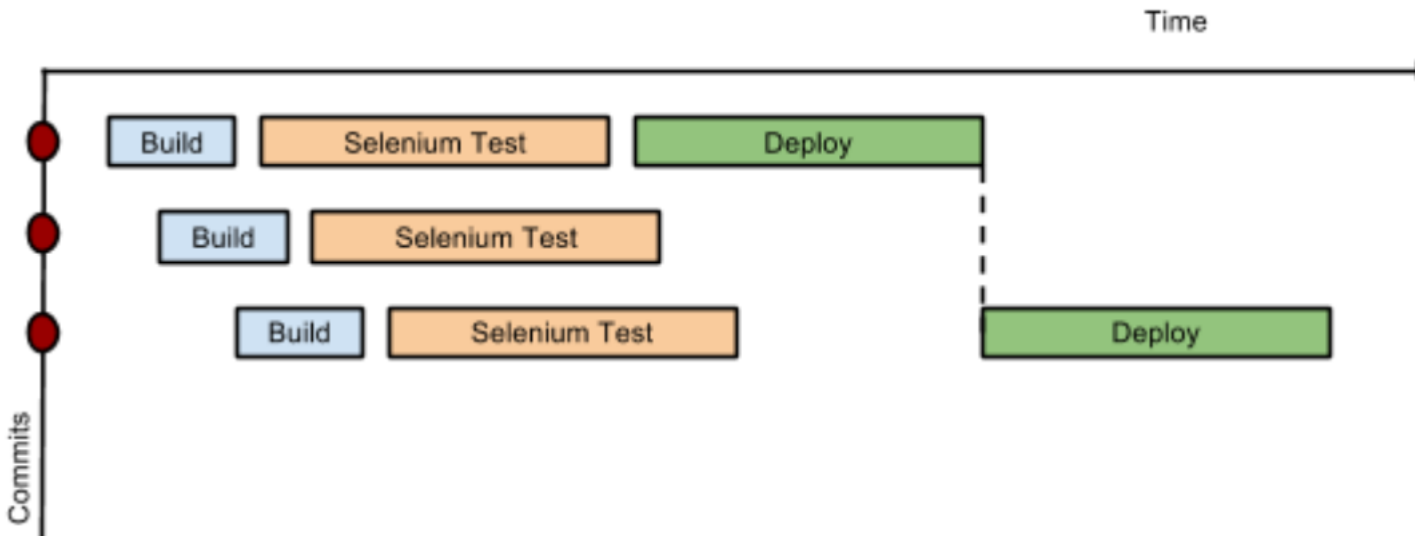


Figura 1. Simultaneidad de la etapa de estrangulación con Pipeline

Considere una tubería simple con tres etapas. Una implementación ingenua de esta canalización puede activar secuencialmente cada etapa en cada confirmación. Por lo tanto, el paso de implementación se desencadena inmediatamente después de que se completen los pasos de prueba de Selenium. Sin embargo, esto significaría que la implementación desde el compromiso dos anula la última implementación en movimiento desde la confirmación uno. El enfoque correcto es que los commits dos y tres esperen a que finalice la implementación de commit one, consolidar todos los cambios que han ocurrido desde commit uno y desencadenar la implementación. Si hay un problema, los desarrolladores pueden averiguar fácilmente si el problema se introdujo en commit dos o commit tres.

Pipeline proporciona esta funcionalidad mejorando la primitiva etapa. Por ejemplo, una etapa puede tener un nivel de concurrencia de una definida para indicar que en cualquier punto solo un hilo debe estar ejecutándose a través del escenario. Esto logra el estado deseado de ejecución de una implementación tan rápido como debería ejecutarse.

```
stage name: 'Production', concurrency: 1
node {
    unarchive mapping: ['target/x.war' : 'x.war']
    deploy 'target/x.war', 'production'
    echo 'Deployed to http://localhost:8888/production/'
}
```

Puertas y aprobaciones

La entrega continua significa tener binarios en un estado listo para lanzar, mientras que la implementación continua significa empujar los binarios a la producción, o implementaciones automatizadas. Aunque el despliegue continuo es un término atractivo y un estado deseado, en realidad las organizaciones todavía quieren que un ser humano otorgue la aprobación final antes de que los bits se envíen a la producción. Esto se captura a través de la primitiva "entrada" en Pipeline. El paso de entrada puede esperar indefinidamente para que intervenga un humano.

```
input message: "Does http://localhost:8888/staging/ look good?"
```

Despliegue de artefactos a puesta en escena / producción

La implementación de binarios es la última milla en una tubería. Los numerosos servidores empleados dentro de la organización y disponibles en el mercado dificultan el uso de un paso de implementación uniforme. Hoy en día, se resuelven mediante productos de implementación de terceros cuyo trabajo es centrarse en el despliegue de una pila particular a un centro de datos. Los equipos también pueden escribir sus propias extensiones para enganchar en el tipo de trabajo Pipeline y facilitar la implementación.

Mientras tanto, los creadores de trabajos pueden escribir una antigua función simple de Groovy para definir cualquier paso personalizado que pueda implementar (o anular la implementación) de artefactos de la producción.

```
def deploy(war, id) {
    sh "cp ${war} /tmp/webapps/${id}.war"
}
```

Flujos reiniciables

Todas las tuberías son reanudables, por lo que si Jenkins necesita reiniciarse mientras se está ejecutando un flujo, debe reanudarse en el mismo punto en su ejecución después de que Jenkins vuelva a iniciar. De manera similar, si un flujo se ejecuta en un largo paso sh o bat cuando un agente se desconecta inesperadamente, no se debe perder ningún progreso cuando se restaura la conectividad.

Hay algunos casos en que una compilación de flujo habrá realizado una gran cantidad de trabajo y se habrá llevado a cabo un error transitorio: uno que no refleje las entradas a esta compilación, como los cambios en el código fuente. Por ejemplo, después de completar una compilación y prueba extensas de un componente de software, la implementación final en un servidor puede fallar debido a problemas de red.

Vista de escenario de tubería

Cuando tiene líneas de compilación complejas, es útil ver el progreso de cada etapa y ver dónde se producen fallas de compilación en la interconexión. Esto puede permitir a los usuarios depurar qué pruebas están fallando en qué etapa o si hay otros problemas en su canalización. Muchas organizaciones también desean que sus canalizaciones sean fáciles de usar para los no desarrolladores sin tener que desarrollar una interfaz de usuario interna, lo que puede ser un esfuerzo de desarrollo prolongado y prolongado.

La función Pipeline Stage View ofrece una visualización extendida del historial de compilación de Pipeline en la página de índice de un proyecto de flujo. Esta visualización también incluye métricas útiles como el tiempo de ejecución promedio por etapa y por construcción, y una interfaz fácil de usar para interactuar con los pasos de entrada.

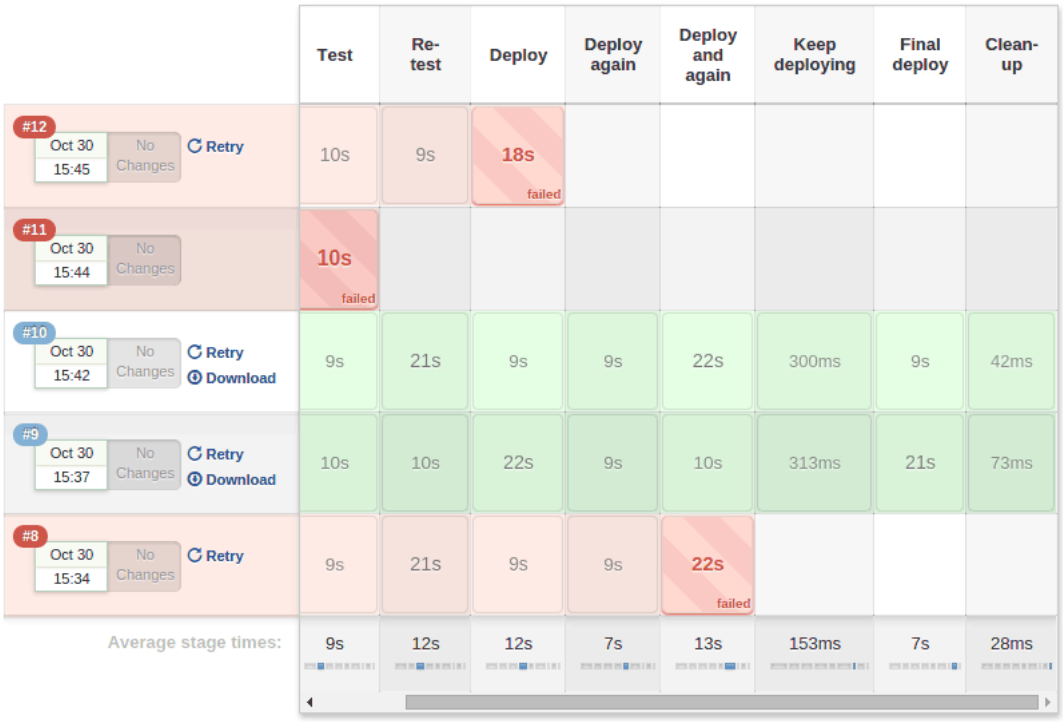


Figura 2. Complemento de Pipeline Stage View

El único requisito previo para este complemento es una interconexión con etapas definidas en el flujo. Puede haber tantas etapas como desee y pueden estar en una secuencia lineal, y los nombres de las etapas se mostrarán como columnas en la interfaz de Stage View.

Trazabilidad de artefactos y con huellas dactilares

La rastreabilidad es importante para los equipos de DevOps que necesitan poder rastrear el código desde el compromiso hasta la implementación. Permite el análisis de impacto al mostrar las relaciones entre los artefactos y permite la visibilidad en todo el ciclo de vida de un artefacto, desde su repositorio de código hasta donde el artefacto finalmente se implementa en la producción.

Jenkins y Pipeline soportan el seguimiento de las versiones de los artefactos mediante el uso de huellas dactilares de archivos, lo que permite a los usuarios rastrear qué construcciones en sentido descendente usan cualquier artefacto dado. Para tomar huellas digitales con Pipeline, simplemente agregue un argumento "huella digital: verdadero" a cualquier paso de archivo de artefactos. Por ejemplo:

```
archiveArtifacts artifacts: '**', fingerprint: true
```

archivará cualquier artefacto WAR creado en Pipeline y los tomará como huellas digitales para su trazabilidad. Este registro de seguimiento de este artefacto y una lista de todos los artefactos de huellas dactilares en una construcción estarán disponibles en el menú de la izquierda de Jenkins:

Para encontrar dónde se usa e implementa un artefacto, simplemente siga el enlace "más detalles" a través del nombre del artefacto y vea las entradas para el artefacto en su lista de "Uso".

Figura 3. Huella dactilar de una GUERRA

Para obtener más información, visite la documentación de huellas digitales para obtener más información sobre cómo funcionan las huellas dactilares.

Mejora esta página | Historial de la página

Recursos

- Eventos
- Documentación
- Blog

Soluciones

- Androide
- C / C ++
- Estibador
- Incrustado
- GitHub
- Java
- PHP
- Entrega continua
- Pitón
- Rubí

Proyecto

- Seguimiento de problemas
- Wiki
- GitHub
- Jenkins en Jenkins

Comunidad

- Lista de correo de usuarios
- Lista de correo de desarrolladores
- Gorjeo
- Reddit
- Mercancías