Entrega continua de soluciones InterSystems utilizando GitLab - Parte II: Flujo de trabajo con GitLab

Artículo
Ricardo Paiva · Oct 18, 2019 Lectura de 9 min

Entrega continua de soluciones InterSystems utilizando GitLab - Parte II: Flujo de trabajo con GitLab

¡Hola Comunidad!

En esta serie de artículos, me gustaría presentar y discutir varios métodos posibles para el desarrollo de software con las tecnologías de InterSystems y GitLab. Trataré temas como:

- Git 101
- El flujo de Git (proceso de desarrollo)
- La instalación de GitLab
- **Flujo de trabajo con GitLab**
- **Entrega continua**
- Instalación y configuración de GitLab
- Integración Continua/Entrega Continua (IC/DC) de GitLab

En el artículo anterior, cubrimos lo básico de Git: por qué es importante contar con un entendimiento de alto nivel de los conceptos de Git para el moderno desarrollo de software y cómo se puede usar Git para desarrollar software. Nuestro foco estuvo en la parte de implementación del desarrollo de software, pero esta parte presenta:

- **GitLab Workflow** - es un proceso completo de ciclo de vida del software, desde la concepción una idea hasta la respuesta del usuario
- **Entrega continua** - es un enfoque que se utiliza en la ingeniería de software, en el cual los equipos producen software mediante ciclos cortos para garantizar que pueda lanzarse de forma confiable y en cualquier momento. Su objetivo es construir, probar y lanzar software más rápido y con mayor frecuencia

Flujo de trabajo con GitLab

GitLab Workflow es una secuencia lógica de acciones posibles a tomar durante todo el ciclo de vida del proceso de desarrollo de un software.

El flujo de trabajo de GitLab, o GitLab Workflow, tiene en cuenta el flujo de GitLab (GitLab Flow), que analizamos en un artículo anterior. Así es como se ve:

1. Idea: cada nueva propuesta comienza con una idea.
2. Incidencia: la forma más efectiva de discutir una idea es crear una incidencia (issue) para la misma. Su equipo y sus colaboradores pueden ayudarle a pulir y mejorar el rastreador de incidencias.
3. Plan: una vez que la discusión llega a un acuerdo, es hora de escribir código. Pero primero necesitaremos priorizar y organizar nuestro flujo de trabajo mediante la asignación de incidencias a hitos y a la lista de incidencias.
4. Escribir código: ahora estamos listos para escribir nuestro código una vez tengamos todo organizado.
5. Confirmar: cuando estemos listo con nuestro borrador, podemos confirmar nuestro código a una rama de función con control de versiones. El flujo de GitLab se explicó en detalle en el artículo anterior.
6. Prueba: ejecutar nuestros scripts usando GitLab CI para compilar y probar nuestra aplicación.

7. Revisión: una vez que nuestro script funcione y nuestras pruebas y compilaciones tengan éxito, estaremos listos para revisar y aprobar nuestro código.
8. Ensayo: es momento de implementar nuestro código en un entorno de ensayo para verificar que todo funcionó como esperábamos o si necesitamos más ajustes.
9. Producción: cuando todo funcione como debería, ¡será el momento de implementarlo a nuestro entorno de producción!
10. Feedback: ahora habrá que volver atrás en busca de las etapas de nuestro trabajo que requieran de mejora**s**.

Como ya dijimos, el proceso en sí no es nuevo (ni exclusivo de GitLab), y se puede lograr con otras herramientas disponibles.

Discutamos varias de estas etapas y lo que implican. También hay documentación disponible.

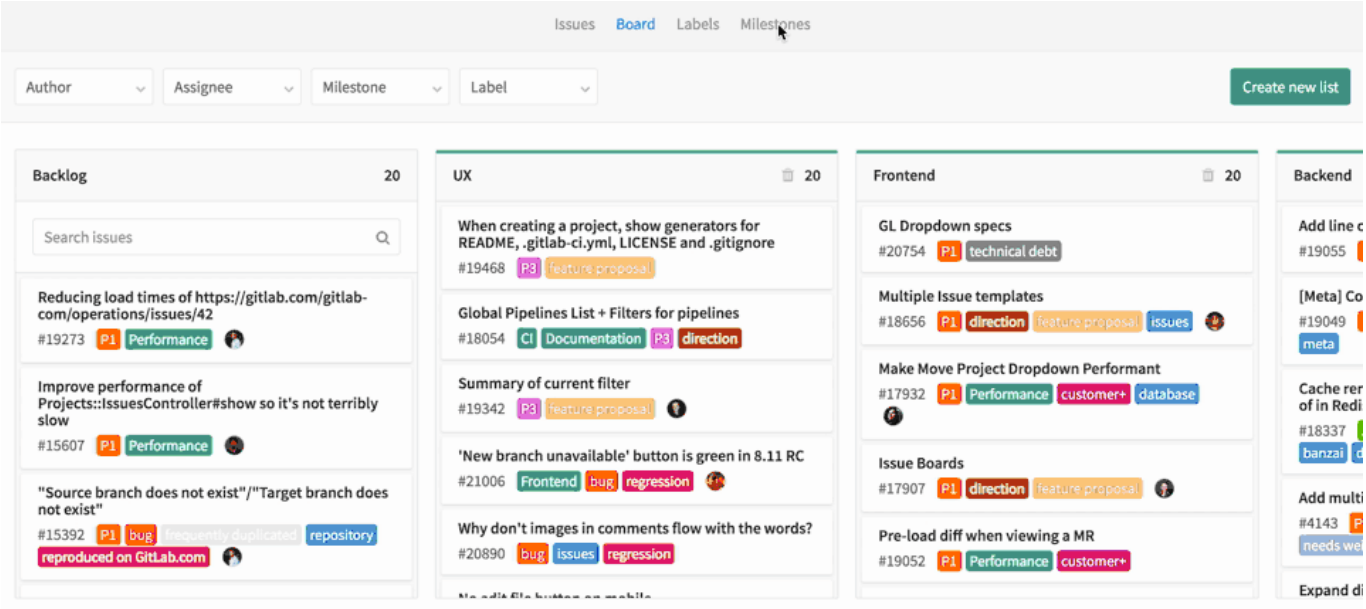
Incidencia y Plan

Las etapas iniciales del flujo de trabajo de GitLab están centradas en una **incidencia**: una función o un error o algún otro tipo de unidad de trabajo semánticamente independiente.

La incidencia tiene varios fines, tales como:

- Gestión: una incidencia tiene una fecha límite, una persona asignada, tiempo invertido y presupuestos, etc. para ayudar a hacer el seguimiento de la resolución de la incidencia.
- Administración: una incidencia es parte de un hito o tabla kanban que nos permite rastrear el software a medida que avanza de versión a versión.
- Desarrollo: una incidencia tiene una discusión y confirmaciones asociadas.

La etapa de planificación nos permite agrupar incidencias por sus prioridades, hitos y tablas kanban, y obtener un resumen de todo esto.



El desarrollo se analizó en la parte anterior: simplemente siga cualquier flujo Git que desee. Luego de desarrollar nuestra nueva función y fusionarla con la maestra, ¿qué sigue?

Entrega Continua

Entrega Continua es un enfoque que se utiliza en la ingeniería de software, en el cual los equipos producen software mediante ciclos cortos para garantizar que pueda lanzarse de forma confiable y en cualquier momento. Su objetivo es construir, probar y lanzar software más rápido y con mayor frecuencia. Este enfoque ayuda a reducir costes, tiempos y riesgos de las entregas de cambios, ya que permite realizar actualizaciones más incrementales de aplicaciones en producción. Para el suministro continuo, es importante contar con un proceso de despliegue lineal y repetible.

Entrega Continua en GitLab

En GitLab, la configuración de la entrega continua se define por repositorio, como un archivo de configuración YAML.

- La configuración del suministro continuo es una serie de **etapas** consecutivas.
- Cada etapa tiene uno o más **scripts** que se ejecutan en paralelo.

Un script define una acción y las condiciones que deberían cumplirse para ejecutarlo:

- Qué hacer (¿ejecutar un comando de SO?, ¿ejecutar un contenedor?)
- Cuándo ejecutar el script:
    - ¿Qué lo dispara (confirmar en una rama específica)?
    - ¿Lo ejecutamos si las etapas anteriores fallaron?
- ¿Ejecutarlo de forma manual o automática?
- ¿En qué entorno ejecutar el script?
- ¿Qué herramientas guardar después de ejecutar los scripts (se cargan desde el entorno a GitLab para acceder más fácilmente)?

Entorno - es un contenedor o servidor configurado en el que puede ejecutar sus scripts.

Runners (ejecutores): ejecutan scripts en entornos específicos. Están conectados a su GitLab y ejecutan scripts cuando es necesario.

Un runner también puede implementarse en un servidor, un contenedor o hasta en su máquina local.

¿Cómo sucede la entrega continua?

1. Una nueva versión confirmada se carga al repositorio.
2. GitLab verifica la configuración de suministro continuo.
3. La configuración de suministro continuo contiene todos los scripts posibles para todos los casos, por lo que se filtran en un conjunto de scripts que deberían ejecutarse para esta confirmación específica (por ejemplo, una confirmación a una rama maestra dispara únicamente acciones relacionadas con una rama maestra). Este conjunto se denomina pipeline**.**
4. El pipeline se ejecuta en un entorno objetivo y los resultados de la ejecución se guardan y muestran en GitLab.

Por ejemplo, este es un pipeline que se ejecuta después de una confirmación a una rama maestra: 

Consiste de cuatro etapas que se ejecutan de forma consecutiva

1. En la etapa de carga, el código se carga a un servidor
2. En la etapa de prueba, se ejecutan pruebas unitarias
3. La etapa de empaquetado consiste de dos scripts que se ejecutan en paralelo:
    - Compilar cliente

- Exportar código de servidor (principalmente con fines informativos)

4. En la etapa de implementación, el cliente compilado se traslada al directorio del servidor web

Como podemos ver, cada script se ejecutó de forma exitosa. Si uno de los scripts falla, de forma predeterminada los siguientes scripts no se ejecutarán (pero podemos cambiar este comportamiento):

Si abrimos el script, podemos ver el registro y determinar por qué falló:

```
Running with gitlab-runner 10.4.0 (857480b6)
 on test runner (ab34a8c5)
Using Shell executor...
Running on gitlab-test...
Fetching changes...
Removing diff.xml
Removing full.xml
Removing index.html
Removing tests.html
HEAD is now at a5bf3e8 Merge branch '4-versiya-1-0' into 'master'
From http://gitlab.eduard.win/test/testProject
 * [new branch] 5-versiya-1-1 -> origin/5-versiya-1-1
 a5bf3e8..442a4db master -> origin/master
 d28295a..42a10aa preprod -> origin/preprod
 3ac4b21..7edf7f4 prod -> origin/prod
Checking out 442a4db1 as master...
Skipping Git submodules setup
$ csession ensemble "##class(isc.git.GitLab).loadDiff()"
[2018-03-06 13:58:19.188] Importing dir /home/gitlab-
runner/builds/ab34a8c5/0/test/testProject/
[2018-03-06 13:58:19.188] Loading diff between a5bf3e8596d842c5cc3da7819409ed81e62c31
e3 and 442a4db170aa58f2129e5889a4bb79261aa0cad0
[2018-03-06 13:58:19.192] Variable modified
var=$lb("MyApp/Info.cls")
Load started on 03/06/2018 13:58:19
Loading file /home/gitlab-
runner/builds/ab34a8c5/0/test/testProject/MyApp/Info.cls as udl
Load finished successfully.
[2018-03-06 13:58:19.241] Variable items
var="MyApp.Info.cls"
var("MyApp.Info.cls")=""
Compilation started on 03/06/2018 13:58:19 with qualifiers 'cuk /checkuptodate=expand
edonly'
Compiling class MyApp.Info
Compiling routine MyApp.Info.1
ERROR: MyApp.Info.cls(version+2) #1003: Expected space : '}' : Offset:14 [zversion+1^
MyApp.Info.1]
 TEXT:  quit, "1.0" }
Detected 1 errors during compilation in 0.010s.
[2018-03-06 13:58:19.252] ERROR #5475: Error compiling routine: MyApp.Info.1. Errors:
 ERROR: MyApp.Info.cls(version+2) #1003: Expected space : '}' : Offset:14 [zversion+1
^MyApp.Info.1]
```

```
 > ERROR #5030: An error occurred while compiling class 'MyApp.Info'
ERROR: Job failed: exit status 1
```

Un error de compilación hizo que nuestro script fallara.

Conclusión

- GitLab soporta todas las etapas principales del desarrollo de software.
- La entrega continua puede ayudarle a automatizar tareas de compilado, prueba e implementación de su software

Enlaces

- Parte I: Git
- Introducción al flujo de trabajo de GitLab
- Documentación sobre Integración Continua/Entrega Continua de GitLab
- El flujo con GitLab
- El código para este artículo

¿Qué sigue?

En el siguiente artículo, veremos cómo:

- Instalar GitLab.
- Conectarlo a varios entornos con productos InsterSystems instalados.
- Crear una configuración de entrega continua.

Analicemos cómo debería funcionar nuestra entrega continua.

Antes que nada, necesitaremos varios entornos y sus ramas correspondientes. El código va en esta rama y se entrega al entorno objetivo:

Entorno	Rama	Entrega	Quién puede confirmar	Quién puede fusionar
Prueba	maestra	Automática	Desarrolladores Dueños	Desarrolladores Dueños
Preproducción	preproducción	Automática	Nadie	Dueños
Producción	producción	Semi-automática (presione un botón para entregar)	Nadie	Dueños

Y, a modo de ejemplo, desarrollaremos una nueva funcionalidad con el flujo de trabajo de GitLab y la entregaremos mediante GitLab CD.

1. La función se desarrolla en una rama de función.
2. La rama de la función pasa por una revisión y se fusiona con la rama maestra.
3. Después de un tiempo (una vez que se fusionaron varias funciones) la rama maestra se fusiona con la preproducción
4. Poco después (pruebas de usuario, etc.), la preproducción se fusiona con la producción

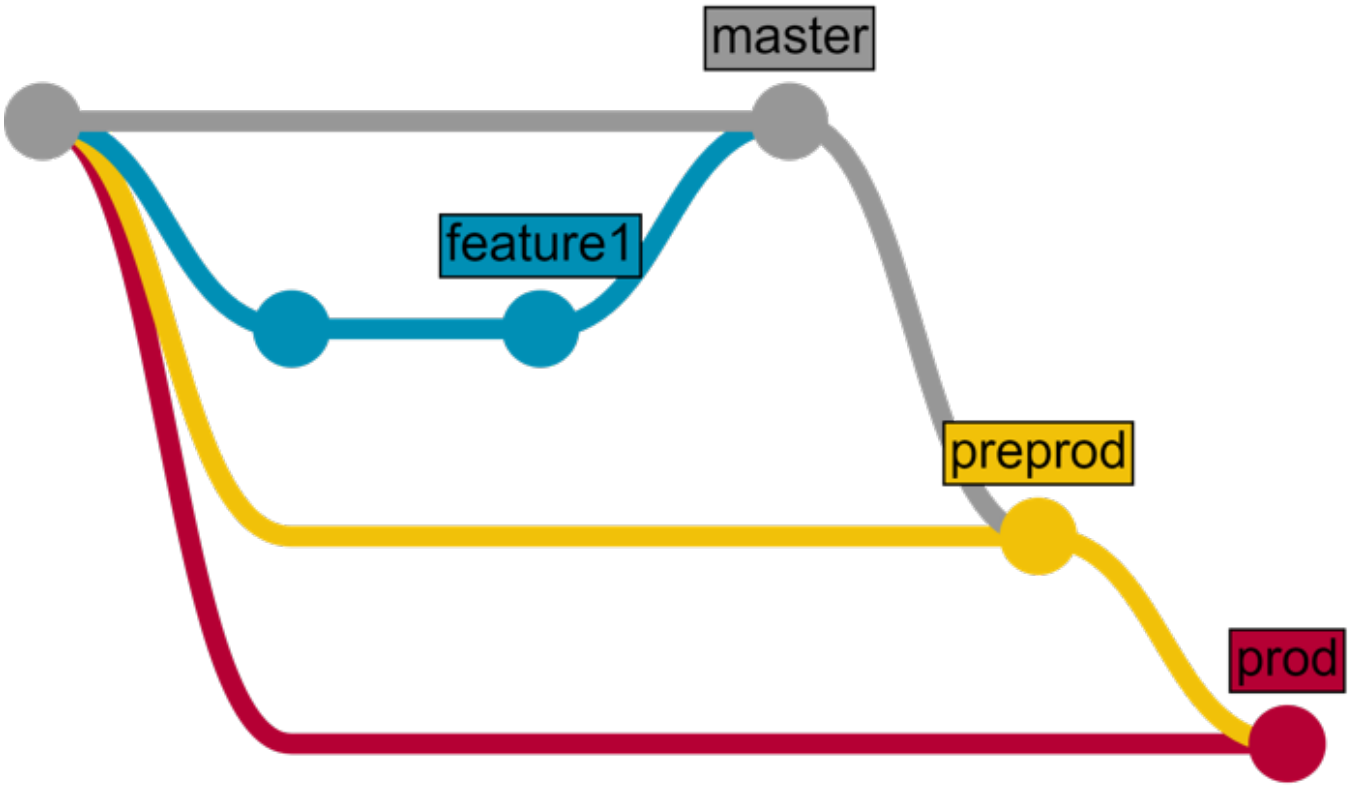
Así es como se vería:

1. Desarrollo y pruebas
    - El desarrollador confirma el código de la nueva función en una rama de función separada

- Luego de que la función se vuelve estable, el desarrollador fusiona nuestra rama de función con la rama maestra
- El código de la rama maestra se entrega al entorno de Prueba, donde se carga y se prueba

2. Entrega al entorno de preproducción
   - El desarrollador crea una solicitud de fusión desde la rama maestra a la rama de preproducción
   - El dueño del repositorio, luego de cierto tiempo, aprueba la solicitud de fusión
   - El código de la rama de preproducción se entrega al entorno de preproducción
3. Entrega al entorno de producción
   - El desarrollador crea una solicitud de fusión desde la rama de preproducción a la rama de producción
   - El dueño del repositorio, luego de cierto tiempo, aprueba la solicitud de fusión
   - El dueño del repositorio presiona el botón "Implementar"
   - El código de la rama de producción se entrega al entorno de producción

Lo mismo, en forma gráfica:



#Change Management #Administración del sistema #Contenedorización #Despliegue #Docker #Git #Integración continua #Principiante #Caché

10 2 0 0 263

Mensajes relacionados

- Entrega continua de soluciones InterSystems utilizando GitLab - Índice
- Entrega continua de soluciones InterSystems utilizando GitLab - Parte I: Git
- Entrega continua de soluciones InterSystems utilizando GitLab - Parte II: Flujo de trabajo con GitLab

Log in or sign up to continue
Añade la respuesta

URL de fuente: https://es.community.intersystems.com/post/entrega-continua-de-soluciones-intersystems-

utilizando-gitlab-parte-ii-flujo-de-trabajo-con