Repositorio de Código Fuente

En el mundo del desarrollo de software, la gestión eficaz del código fuente es crucial para el éxito de cualquier proyecto. Un repositorio de código fuente centralizado proporciona un lugar seguro para almacenar, organizar y colaborar en el desarrollo del código. Gitlab, una plataforma de gestión de repositorios Git líder, ofrece una amplia gama de funciones para facilitar la colaboración y el control de versiones del código.

En esta sección, se presenta la definición de un repositorio de código fuente para la empresa ADA, utilizando Gitlab y el flujo de trabajo Gitflow.

Gitlab: La Plataforma Elegida

Gitlab se ha convertido en la plataforma de gestión de repositorios Git preferida por muchas empresas debido a su conjunto completo de funciones, escalabilidad y facilidad de uso. Ofrece una interfaz web intuitiva, herramientas integradas para la gestión de tareas, seguimiento de errores y CI/CD (Integración Continua y Entrega Continua).

Flujo de Trabajo Gitflow: Estructurando el Desarrollo

El flujo de trabajo Gitflow es un modelo de ramificación ampliamente utilizado para organizar el desarrollo de software en Git. Proporciona una estructura clara para crear, fusionar y eliminar ramas, asegurando que los cambios de código estén bien definidos y sean rastreables.

Beneficios de la Implementación

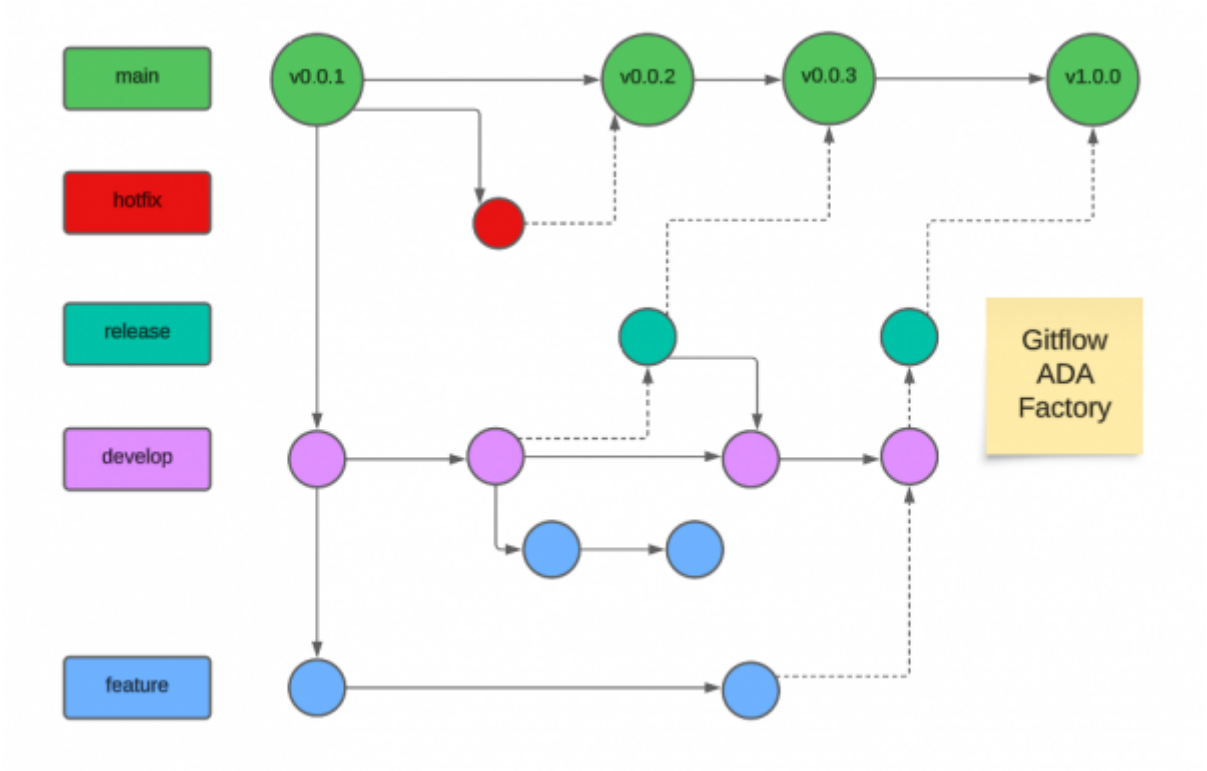
La implementación de un repositorio de código fuente en Gitlab con el flujo de trabajo Gitflow para la empresa ADA ofrece numerosos beneficios:

- **Gestión Centralizada del Código**: Un único repositorio centralizado almacena todo el código fuente del proyecto, facilitando el acceso y la colaboración entre los desarrolladores.
- **Control de Versiones Eficaz**: Gitlab proporciona un historial completo de todos los cambios realizados en el código, permitiendo a los desarrolladores retroceder a versiones anteriores si es necesario.
- **Flujo de Trabajo Estructurado**: El flujo de trabajo Gitflow guía a los desarrolladores en el proceso de desarrollo, asegurando que los cambios se realicen de manera organizada y eficiente.
- **Mejora de la Colaboración**: Gitlab facilita la colaboración entre desarrolladores, permitiendo la revisión y aprobación del código antes de su fusión en la rama principal.
- **Mayor Visibilidad y Transparencia**: El repositorio centralizado proporciona una visión clara del estado del proyecto y el progreso del desarrollo a todos los interesados.
- **Integración con CI/CD**: Gitlab se integra perfectamente con herramientas de CI/CD, permitiendo la automatización de pruebas, implementación y despliegue del código.

Flujo de trabajo - GitFlow ADA

El flujo de trabajo Gitflow es un modelo de ramificación que ayuda a los equipos a gestionar el desarrollo y la colaboración de código de manera eficaz. Proporciona un enfoque estructurado para crear, fusionar y eliminar ramas, asegurando que los cambios de código estén bien definidos y sean rastreables.

A continuación se describen los elementos que hacen parte de ese flujo y como se utilizará en la compañía siguiendo el estandar recomendado. Según el siguiente gráfico:



Ramas

El flujo de trabajo Gitflow utiliza varias ramas para organizar y gestionar el desarrollo del código:

- **master**: La rama principal que representa la versión estable y lanzada del código. Esta rama solo debe actualizarse con cambios de código probados y aprobados.
- **develop**: La rama donde se lleva a cabo el desarrollo activo. Las características se ramifican desde aquí y eventualmente se fusionan de nuevo en esta rama una vez que se completan y se prueban a fondo.
- **feature**: Ramas creadas desde develop para trabajar en características específicas o correcciones de errores. Estas ramas deben tener nombres descriptivos que indiquen claramente el propósito del trabajo que se está realizando.
- **release**: Una rama creada desde develop para preparar una nueva versión. Esta rama se utiliza para probar y finalizar la versión antes de fusionarla en master.
- **hotfix**: Una rama creada directamente desde master para abordar errores o problemas críticos en el código lanzado. Estas ramas deben crearse y fusionarse lo antes posible para minimizar el impacto del problema en los usuarios.

Definición de ramas para la fábrica

La empresa por varios años ha utilizado un repositorio svn para la gestión del código fuente de los desarrollos. Teniendo presente ese enfoque, el personal actual y el flujo de los procesos actuales de implementación de cambios, mejoras o nuevos desarrollos se hace esta definición de ramas en la nueva migración.

- **master**: Esta rama será la utilizada para almacenar el código fuente estable. Es similar a la rama tags del anterior modelo svn utilizado en la compañía para las aplicaciones powerbuilder.
- **develop**: En esta rama se acumulan las validaciones de parte del equipo de QA para posteriormente liberar una versión. Es similar a la rama pre del anterior modelo svn utilizado en la compañía.
- **feature**: En esta rama se realizará la migración de las funcionalidades. Es similar a la rama branches del anterior modelo svn utilizado en la compañía. Sin embargo tambien será utilizada para validar la solucion antes de integrarla a develop.
- **release**: Esta rama será la utilizada para las liberaciones en el ambiente de producción. Es similar a la rama tags del anterior modelo svn utilizado en la compañía, pero su duración es corta ya que una vez se integren los cambios a master y develop será eliminada.
- **hotfix**: Rama utilizada para solución de errores en producción es similar al branches del modelo anterior svn utilizado en la compañía pero su duración es corta ya que una vez se integren los cambios a master y develop será eliminada.

Roles y Responsabilidades

El flujo de trabajo Gitflow define roles y responsabilidades para los miembros del equipo:

- **Mantenedores**: Responsables de gestionar la rama master, asegurando su estabilidad y calidad. Revisan las solicitudes de extracción de la rama release antes de fusionarlas en master.
- **Desarrolladores**: Trabajan en características y correcciones de errores, creando ramas de develop y fusionándolas de nuevo después de su finalización. Deben seguir los estándares de codificación y garantizar la calidad de su código.
- **Testers**: Prueban los cambios de código en las ramas feature y release antes de fusionarlos en develop o master. Deben identificar y reportar cualquier error o problema encontrado durante las pruebas.

Definición de Roles y Responsabilidades

Al igual que las ramas del repositorio anterior, la empresa por varios años ha utilizado una estructura simple de gestion de ramas la cual es branches para desarrolladores, trunk para qa y tags para liberaciones en produccion. Teniendo presente ese enfoque, el personal actual y el flujo de los procesos actuales de implementación de cambios, mejoras o nuevos desarrollos se hace esta definición de roles en la nueva migración.

- **Mantenedores**: Responsables de gestionar la rama master, asegurando su estabilidad y calidad. Revisan las solicitudes de extracción de la rama release antes de fusionarlas en master. Este rol puede ser asumido por el líder técnico del módulo o caracteristica que se va a integrar.

- **Desarrolladores**: Trabajan en características y correcciones de errores, creando ramas de develop y fusionándolas de nuevo después de su finalización. Deben seguir los estándares de codificación y garantizar la calidad de su código. Este rol es asignado a los desarrolladores que realizarán la migración.
- **Testers**: Prueban los cambios de código en las ramas feature y release antes de fusionarlos en develop o master. Deben identificar y reportar cualquier error o problema encontrado durante las pruebas. Este rol es asumido por el equipo de QA.

Procesos

El flujo de trabajo Gitflow implica procesos específicos para gestionar los cambios de código:

Consideraciones previas

- Antes de iniciar el flujo la rama develop debe estar sincornizada con la rama master.
- Todo nombre de repositorio debe estar escrito en minusculas y solo se permitirá el caracter '-' para separar nombres.

Creación de una Rama Feature:

- a. Un desarrollador crea una nueva rama desde develop utilizando un nombre descriptivo, como feature/nueva-característica.
- b. El trabajo en la característica se realiza en esta rama. Los desarrolladores deben seguir los estándares de codificación y realizar pruebas unitarias de su código a medida que trabajan.

Finalización de una Rama Feature:

- a. Una vez que la característica está completa, el desarrollador ejecuta pruebas para asegurarse de que funciona como se espera. Esto incluye tanto pruebas unitarias como pruebas de integración.
- b. Envía los cambios al repositorio remoto.
- c. Crea una solicitud de extracción para fusionar la rama feature en develop. La solicitud de extracción debe incluir una descripción clara de los cambios realizados y cualquier documentación o resultado de prueba relevante.

Revisión y Fusión de una Rama Feature:

- a. Otros desarrolladores revisan la solicitud de extracción, asegurando la calidad del código y el cumplimiento de los estándares. Deben dejar comentarios y sugerencias según sea necesario.
- b. Si no se encuentran problemas, la solicitud de extracción se fusiona en develop.
- c. La rama feature se elimina una vez fusionada.

Creación de una Rama Release:

- a. Una vez que un conjunto de características está listo para su lanzamiento, se crea una rama release desde develop. Esta rama debe etiquetarse con un número de versión, como v1.0.0.
- b. La rama release se actualiza con cualquier cambio o corrección de errores adicional que sea necesario para la versión.

Prueba y Finalización del Release:

- a. La rama release se prueba a fondo para garantizar la estabilidad y la funcionalidad del release. Esto incluye pruebas manuales, pruebas automatizadas y pruebas de rendimiento.
- b. Se realizan las correcciones de errores necesarias y se fusionan en la rama release.

Fusión de la Rama Release:

- a. Una vez finalizado el release, se fusiona en master. Esto solo debe hacerse después de que se complete toda la prueba

Plantilla CI

```
# This file is a template, and might need editing before it works on your
project.
# To contribute improvements to CI/CD templates, please follow the
Development guide at:
# https://docs.gitlab.com/ee/development/cicd/templates.html
# This specific template is located at:
#
https://gitlab.com/gitlab-org/gitlab/-/blob/master/lib/gitlab/ci/templates/G
etting-Started.gitlab-ci.yml

# This is a sample GitLab CI/CD configuration file that should run without
any modifications.
# It demonstrates a basic 3 stage CI/CD pipeline. Instead of real tests or
scripts,
# it uses echo commands to simulate the pipeline execution.
#
# A pipeline is composed of independent jobs that run scripts, grouped into
stages.
# Stages run in sequential order, but jobs within stages run in parallel.
#
# For more information, see:
https://docs.gitlab.com/ee/ci/yaml/index.html#stages
image: maven:3.9.6-amazoncorretto-21  # Ajustar la etiqueta de la imagen si
es necesario
variables:
  MAVEN_CONFIG: 'settings.xml'
stages:          # List of stages for jobs, and their order of execution
```

```
  - build
  - test
  - deploy

build-job:       # This job runs in the build stage, which runs first.
  stage: build
  script:
    - echo "Compiling the code..."
    - mvn clean package -Dsettings.security=none # Ajustar el comando para
Gradle
    - echo "Compile complete."

unit-test-job:   # This job runs in the test stage.
  stage: test    # Etapa opcional para pruebas unitarias
  script:
    - echo "Running unit tests..."
    - mvn test
    - echo "Test Complete"

deploy-job:      # This job runs in the deploy stage.
  stage: deploy  # Etapa opcional para implementación (reemplazar con su
estrategia de implementación)
  script:
    - echo "Deploying application..."
    - scp target/*.jar gestion@10.1.140.21:/opt/ada/deploy/cicd  #
Reemplazar con sus detalles de implementación
    - echo "Application successfully deployed."
```

Flujo Git Flow

Conclusión

La adopción de Gitlab como plataforma de gestión de repositorios Git y el flujo de trabajo Gitflow para la empresa ADA proporcionará una base sólida para el desarrollo de software eficiente, colaborativo y escalable. Esta combinación permitirá a ADA gestionar su código fuente de manera efectiva, mejorar la colaboración entre equipos y acelerar el desarrollo de productos.

←Volver atrás