DZone

Estrategias Git para el desarrollo de software: parte 1

por Dharmendra Rathor · 14 y 18 de agosto · Zona DevOps · Tutorial

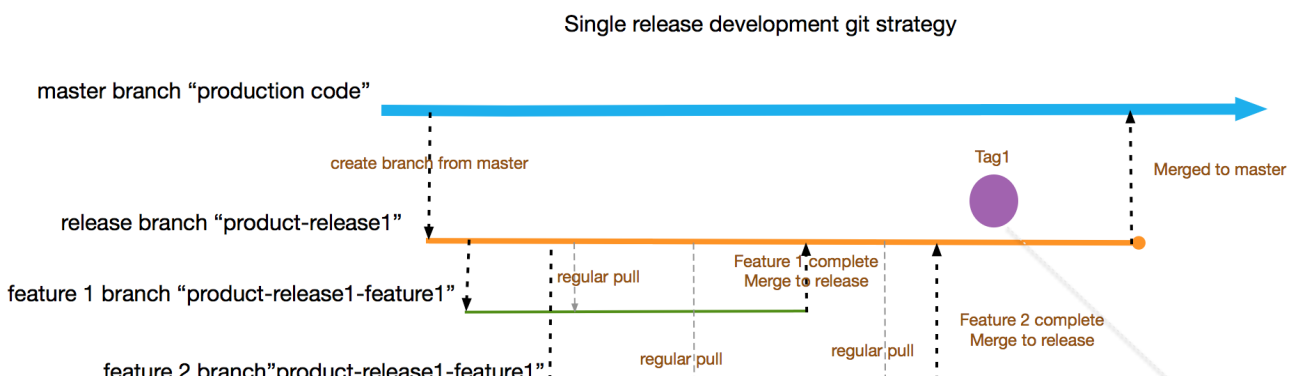
Git es un sistema de control de versiones para rastrear cambios en archivos y coordinar el trabajo en esos archivos entre varias personas. Se utiliza principalmente para la gestión del código fuente en el desarrollo de software. Es un sistema de control de revisión distribuido y es muy útil para admitir flujos de trabajo de desarrollo de software.

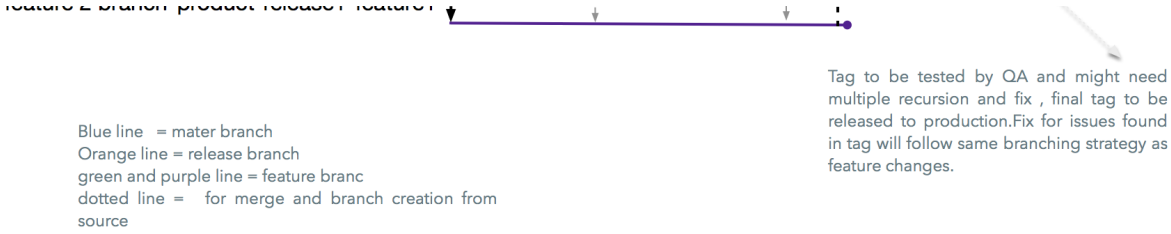
El directorio Git en cada máquina es un repositorio completo que tiene capacidades completas de seguimiento de versiones e independiente del acceso a la red. Puede mantener sucursales, realizar fusiones y continuar con el desarrollo incluso cuando no esté conectado a la red. Para mí, tener un repositorio completo en mi máquina y facilidad de uso (crear una rama, fusionar ramas y mantener flujos de trabajo de ramificación) son las mayores ventajas de Git. Es un software gratuito y de código abierto distribuido bajo los términos de GNU.

Intentaré cubrir dos estrategias comunes que se pueden utilizar para apoyar el desarrollo de software en dos artículos. En este artículo, cubriré la estrategia Git de desarrollo de lanzamiento único. Un equipo de software que trabaja en una versión a la vez generalmente usa esta estrategia.

Estrategia de desarrollo de lanzamiento único

Un equipo de software que trabaja en una versión a la vez sigue los siguientes pasos. Intentaré agregar los pasos del comando git / git UI para realizar cada paso. El siguiente diagrama muestra los detalles de la estrategia:





Cree una rama de lanzamiento a partir de la rama de código de producción "maestro".

```
git checkout master
git checkout -b product-release1
```

El maestro tiene un código de producción y "product-release1" es la nueva rama de lanzamiento.

Cada miembro de desarrollo crea una rama de características a partir de la rama de lanzamiento y trabaja en el desarrollo de características en esa rama.

```
git checkout lanzamiento de producto1
git checkout -b product-release1-feature1
```

Cada miembro de desarrollo debe trabajar en la rama de características "product-release1-feature1".

Todos deberían tomar las últimas actualizaciones de la rama de lanzamiento diariamente usando los comandos `merge` o `rebase` Git.

Supongamos que está trabajando en "product-release1-feature1" y debe extraer los últimos cambios del formulario "product-release1". Esto garantizará que cualquier cambio que se haya movido a la rama de lanzamiento esté disponible en la rama de característica local. Esto también evita conflictos o códigos duplicados al fusionar la rama de la función con la rama de la versión en el momento de finalización de la función.

```
git merge product-release1
o
git rebase product-release1
```

Nota: Sugeriría usar la `merge` opción hasta que tenga muy claro `rebase` .

Antes de realizar este paso, solo debe estar en su rama de características. Use el "`git branch`" comando para verificar el nombre de la rama antes de hacer una fusión.

Al completar una función, genere una solicitud de extracción para fusionar el código con la rama de lanzamiento. Este paso se puede hacer en la interfaz de usuario de GitHub.

Puede crear y asignar solicitudes de extracción a un revisor utilizando la interfaz de usuario de git. Consulte este enlace para ver los pasos en la interfaz de usuario.

Se realizará una revisión del código en la solicitud de extracción y se realizarán cambios si es necesario. Si todo está bien, la solicitud de extracción será aprobada por el revisor. Consulte este enlace para ver los pasos en la interfaz de usuario.

Cree una etiqueta desde la rama de lanzamiento al equipo de prueba para la validación de la función. Tenemos que proporcionar una etiqueta al equipo de prueba una vez que se completen las funciones

relacionadas que se probarán.

```
git tag -a product-release1_tag1 -m  "product release1 tag1"
```

Esto debe hacerse en la interfaz de usuario de Git .

Una vez completadas todas las funciones planificadas en el lanzamiento, la etiqueta final se puede proporcionar al equipo de prueba para su validación.

Una vez que se completen las pruebas, se realizará una versión de producción de software a partir de la etiqueta final aprobada por QA.

Después del lanzamiento de producción, combine el código de ramificación de lanzamiento con el maestro para que el maestro tenga el código de producción. Este paso se puede hacer usando la interfaz de usuario de Git elevando una fusión de la rama de lanzamiento al maestro.

A continuación, elimine la rama de la característica al completar la característica y elimine la rama de la versión al completar la versión.

```
git branch -d product-release1-feature1
git branch -d product-release1
```

Cualquier corrección de lanzamiento de posproducción se puede hacer en una etiqueta. Recuerde mover el código al maestro y publicar cualquier versión de revisión de producción.

```
git checkout -b   product-release1_prodfixticketNumber product-release1_tag1
```

Este comando crea una nueva rama, "product-release1_prodfixticketNumber", a partir de la etiqueta de lanzamiento de producción "product-release1_tag1".

Cree la próxima rama de lanzamiento del maestro, ya que el maestro tiene el código de producción, y siga el mismo enfoque para el desarrollo que se describe anteriormente.

En este enfoque, la rama maestra tiene el código de producción y los lanzamientos se realizan desde una etiqueta. Cada miembro de desarrollo puede cortar una rama de la rama de lanzamiento, trabajar en una función de forma independiente y, una vez completada la función, puede generar una solicitud de extracción de fusión y obtener su aprobación. Mientras realiza el desarrollo, cada miembro de desarrollo debe volver a crear el código de la rama de lanzamiento para que tenga el código más reciente.

Este enfoque funciona bien para proyectos en los que un equipo está trabajando en un solo lanzamiento a la vez. La mayoría de los equipos están trabajando en varias versiones a la vez, por lo que deben mantener activas más de una rama de versiones a la vez.

Lea el siguiente artículo para conocer un posible enfoque para manejar proyectos que trabajan en varias versiones en paralelo.

Referencias

Comandos útiles de git:

- https://github.com/DharmendraRathor/gitCommands

- https://gist.github.com/DharmendraRathor

Herramienta de escritorio de GitHub: https://desktop.github.com/

¿Te gusta este artículo? Leer más de DZone

Trabajando con ramas de funciones Git [Video]

Los 10 mejores sistemas de control de versiones

Deshaciendo cosas en Git

Tarjeta DZone gratis
Patrones de diseño para entrega continua

Temas: GIT FLOW, GIT, TUTORIAL, DEVOPS, DESARROLLO DE SOFTWARE, CONTROL DE VERSIONES, CÓDIGO ABIERTO