2024

Instituto Tecnologico Las Americas

❖ **Eddy Disla 2023-0212.**

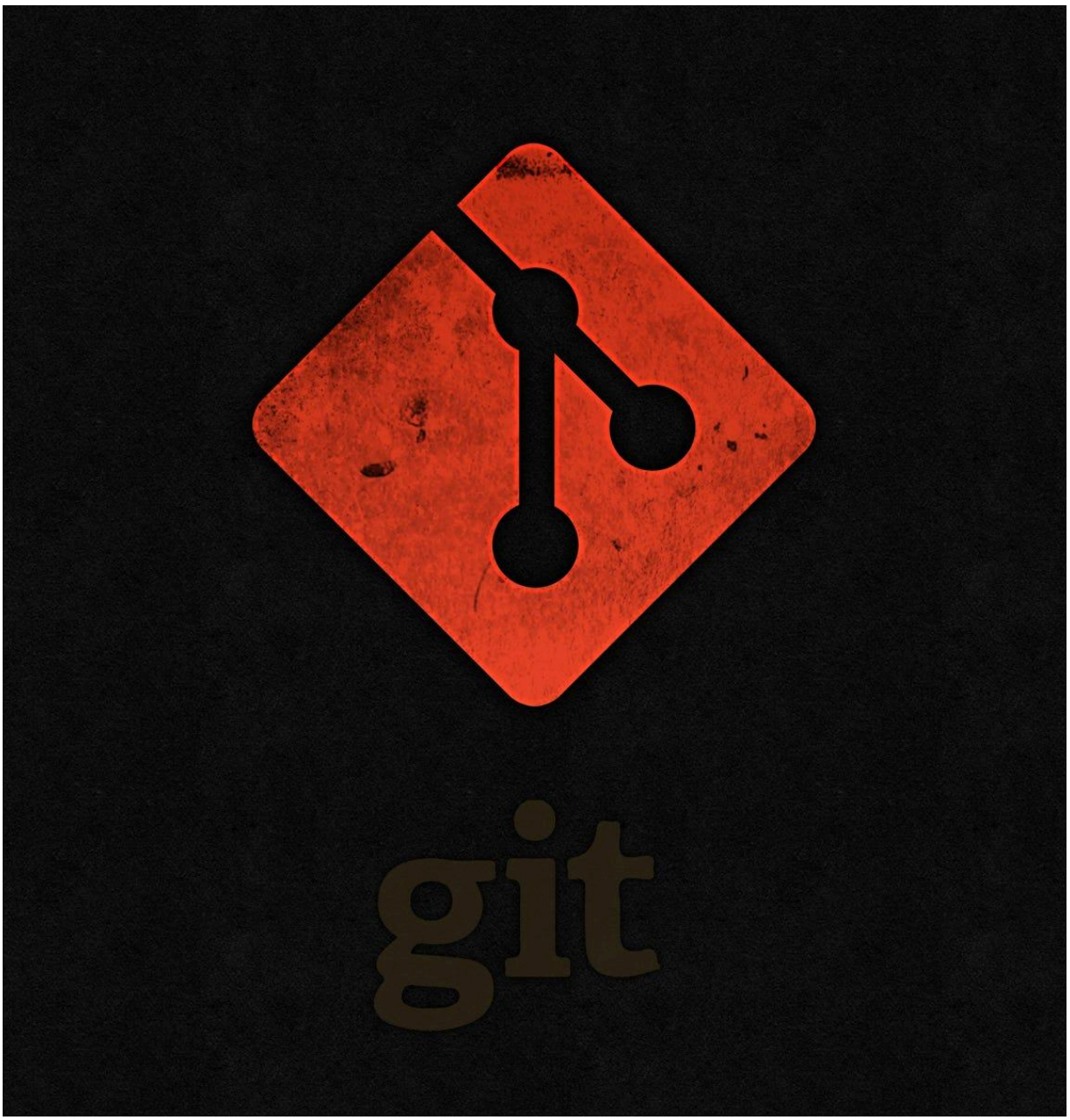
Programación III

Kelyn Tejada Belliard

Tarea 3

Fecha de Entrega: 22/11/2024

"Git: Controla el cambio, lidera el desarrollo."



Índice de Contenido

Introducción:

En el mundo del desarrollo de software, la colaboración eficiente y el manejo adecuado de los cambios en el código son fundamentales. Git, un sistema de control de versiones distribuido creado por Linus Torvalds en 2005, se ha convertido en una herramienta imprescindible para gestionar proyectos de cualquier escala. Gracias a Git, los desarrolladores pueden rastrear cambios, trabajar en equipo y garantizar la integridad del código a lo largo del tiempo.

El uso de Git no solo implica conocer sus comandos esenciales, como git init, git add, o git push, sino también entender conceptos clave como las ramas, que permiten trabajar en nuevas características de forma paralela, y estrategias avanzadas como Git Flow y Trunk Based Development (TBD). Estas metodologías ofrecen enfoques distintos para gestionar los flujos de trabajo y lanzar productos de manera organizada.

Mientras que Git Flow estructura el desarrollo en ramas especializadas para características, versiones y correcciones, TBD simplifica el proceso mediante una integración continua directa en una única rama principal. Cada enfoque responde a diferentes necesidades, dependiendo del tamaño del equipo, la frecuencia de lanzamientos y los objetivos del proyecto.

Este documento explora conceptos básicos y avanzados de Git, los comandos esenciales para dominarlo, y las estrategias de flujo de trabajo más utilizadas, proporcionando una base sólida para entender y aplicar las mejores prácticas en el desarrollo de software.

1-Desarrolla el siguiente Cuestionario

1- ¿Qué es Git?
2- ¿Para qué funciona el comando Git init?
3-¿Qué es una rama?
4- ¿Cómo saber en cual rama estoy?
5- ¿Quién creó git?
6- ¿Cuáles son los comandos más esenciales de Git?
7- ¿Qué es git Flow?
8- ¿Qué es trunk based development ?

– Desarrollo –

¿Qué es Git?

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o front end como Cogito o StGIT. Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena. Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de programación del núcleo Linux.

El mantenimiento del software Git está actualmente (2009) supervisado por Junio Hamano, quien recibe contribuciones al código de alrededor de 280 programadores. En cuanto a derechos de autor Git es un software libre distribuible bajo los términos de la versión 2 de la Licencia Pública General de GNU.

¿Para qué funciona el comando Git init?

El comando git init crea un nuevo repositorio de Git. Puede utilizarse para convertir un proyecto existente y sin versión en un repositorio de Git, o para inicializar un nuevo repositorio vacío. La mayoría de los demás comandos de Git no se encuentran disponibles fuera de un repositorio inicializado, por lo que este suele ser el primer comando que se ejecuta en un proyecto nuevo.

Al ejecutar git init, se crea un subdirectorio de .git en el directorio de trabajo actual, que contiene todos los metadatos de Git necesarios para el nuevo repositorio. Estos metadatos incluyen subdirectorios de objetos, referencias y archivos de plantilla. También se genera un archivo HEAD que apunta a la confirmación actualmente extraída.

Aparte del directorio de .git, en el directorio raíz del proyecto, se conserva un proyecto existente sin modificar (a diferencia de SVN, Git no requiere un subdirectorio de .git en cada subdirectorio).

De manera predeterminada, git init inicializará la configuración de Git en la ruta del subdirectorio de .git. Puedes cambiar y personalizar dicha ruta si quieres que se encuentre en otro sitio. Asimismo, puedes establecer la variable de entorno $GIT_DIR en una ruta personalizada para que git init inicialice ahí los archivos de configuración de Git. También puedes utilizar el argumento --separate-git-dir para conseguir el mismo resultado. Lo que se suele hacer con un subdirectorio de .git independiente es mantener los archivos ocultos de la configuración del sistema (.bashrc, .vimrc, etc.) en el directorio principal, mientras que la carpeta de .git se conserva en otro lugar.

¿Qué es una rama?

Las ramas son una de las principales utilidades que disponemos en Git para llevar un mejor control del código. Se trata de una bifurcación del estado del código que crea un nuevo camino de cara a la evolución del código, en paralelo a otras ramas que se puedan generar.

Las ramas nos pueden servir para muchos casos de uso. Por ejemplo, tras la elección del hosting para crear una página web o desarrollar una tienda online, para la creación de una funcionalidad que queramos integrar en un programa y para la cual no queremos que la rama principal se vea afectada. Esta función experimental se puede realizar en una rama independiente, de modo que, aunque tardemos varios días o semanas en terminarla, no afecte a la producción del código que tenemos en la rama principal y que permanecerá estable.

El trabajo con ramas resulta muy cómodo en el desarrollo de proyectos, porque es posible que todas las ramas creadas evolucionen al mismo tiempo, pudiendo el desarrollador pasar de una rama a otra en cualquier momento según las necesidades del proyecto. Si en un momento dado el trabajo con una rama nos ha resultado interesante, útil y se encuentra estable, entonces podemos fusionar ramas para incorporar las modificaciones del proyecto en su rama principal.

¿Cómo saber en cual rama estoy?

En todo proyecto creado con Git podemos ver la rama en la que estamos actualmente situados. Para ello, utilizamos el comando:

```
git branch
```

Si no has hecho ninguna rama todavía (o si no te has movido de rama), Git nos indicará que estamos en la rama master o principal.

```
$ git branch
 * master
```

También disponemos del comando show-branch que nos muestra un resumen de las ramas de un proyecto y sus últimas modificaciones (o commits).

```
git show-branch
```

O también puedes seguir los siguientes pasos sin repetir y sin dejar espacios entre ellos:

1. Abre la terminal.

2. Navega hasta el directorio raíz de tu repositorio utilizando el comando «cd» seguido de la ruta del directorio.

3. Ejecuta el comando «git branch» para ver la lista de ramas existentes en el repositorio.

4. La rama activa o actual se mostrará resaltada con un asterisco (*) al lado del nombre de la rama.

5. Identifica la rama activa seleccionada.

Recuerda que Git es una herramienta poderosa para el control de versiones, y trabajar con ramas te permite organizar tu trabajo de manera efectiva y colaborar con otros desarrolladores.

¿Quién creó git?

Linus Torvalds es un programador finlandés que es considerado uno de los padres fundadores del software libre. En 1991, Torvalds comenzó a desarrollar el kernel de Linux, el núcleo del sistema operativo GNU/Linux. Para poder gestionar las versiones del código fuente de Linux, Torvalds utilizó el sistema de control de versiones BitKeeper. Sin embargo, en 2005, los desarrolladores de BitKeeper decidieron restringir el uso de su software para proyectos comerciales.

Esto llevó a Torvalds a crear su propio sistema de control de versiones, que llamó Git. Git se basó en los conceptos de control de versiones distribuidos, que permitían a los desarrolladores trabajar en el mismo proyecto desde diferentes ubicaciones.

Git se lanzó por primera vez en 2005. En los primeros años, Git fue utilizado principalmente por desarrolladores de Linux. Sin embargo, a medida que Git se hizo más conocido, comenzó a ser adoptado por otros proyectos de software libre y comercial.

El auge de Git se produjo en los años 2010. En esta década, el desarrollo de software se volvió cada vez más colaborativo y distribuido. Git se adaptó perfectamente a este nuevo paradigma, convirtiéndose en el sistema de control de versiones más popular del mundo.

¿Cuáles son los comandos más esenciales de Git?

Inicialización y configuración:

- git init: Inicializa un nuevo repositorio.
- git config: Configura opciones como el nombre y correo del usuario.

Estado y visualización:

- git status: Muestra el estado del repositorio.
- git log: Muestra el historial de confirmaciones.

Trabajar con ramas:

- git branch: Crea, lista o elimina ramas.
- git checkout: Cambia de rama.
- git merge: Fusiona ramas.

Seguimiento y confirmación de cambios:

- git add: Añade archivos al área de preparación.
- git commit: Confirma los cambios al repositorio.
- git diff: Muestra diferencias entre versiones de archivos.

Colaboración remota:

- git clone: Clona un repositorio remoto.
- git pull: Descarga y fusiona cambios desde un repositorio remoto.
- git push: Envía cambios al repositorio remoto.

¿Qué es git Flow?

Gitflow es un modelo alternativo de creación de ramas en Git en el que se utilizan ramas de función y varias ramas principales. Fue Vincent Driessen en nvie quien lo publicó por primera vez y quien lo popularizó. En comparación con el desarrollo basado en troncos, Gitflow tiene diversas ramas de más duración y mayores confirmaciones. Según este modelo, los desarrolladores crean una rama de función y retrasan su fusión con la rama principal del tronco hasta que la función está completa. Estas ramas de función de larga duración requieren más colaboración para la fusión y tienen mayor riesgo de desviarse de la rama troncal. También pueden introducir actualizaciones conflictivas.

Gitflow puede utilizarse en proyectos que tienen un ciclo de publicación programado, así como para la práctica recomendada de DevOps de entrega continua. Este flujo de trabajo no añade ningún concepto o comando nuevo, aparte de los que se necesitan para el flujo de trabajo de ramas de función. Lo que hace es asignar funciones muy específicas a las distintas ramas y definir cómo y cuándo deben estas interactuar. Además de las ramas de función, utiliza ramas individuales para preparar, mantener y registrar publicaciones. Por supuesto, también puedes aprovechar todas las ventajas que aporta el flujo de trabajo de ramas de función: solicitudes de incorporación de cambios, experimentos aislados y una colaboración más eficaz.

¿Qué es trunk based development ?

El desarrollo basado en troncos es una práctica de gestión de control de versiones en la que los desarrolladores fusionan actualizaciones pequeñas y frecuentes en un "tronco" principal o rama principal. Es una práctica común entre los equipos de

DevOps y forma parte del ciclo de vida de DevOps, ya que agiliza las fases de fusión e integración. De hecho, el desarrollo basado en troncos es una práctica obligatoria de CI/CD. Los desarrolladores pueden crear ramas de corta duración con algunas confirmaciones pequeñas en comparación con otras estrategias de ramificación de características de larga duración. A medida que crece la complejidad de la base de código y el tamaño del equipo, el desarrollo basado en troncos ayuda a mantener el flujo de versiones de producción.

En los primeros días del desarrollo de software, los programadores no podían permitirse el lujo de los modernos sistemas de control de versiones. Más bien, desarrollaron dos versiones de su software al mismo tiempo como un medio para rastrear los cambios y revertirlos si era necesario. Con el tiempo, este proceso resultó ser laborioso, costoso e ineficiente.

A medida que los sistemas de control de versiones maduraron, surgieron varios estilos de desarrollo, lo que permitió a los programadores encontrar errores más fácilmente, codificar en paralelo con sus colegas y acelerar la cadencia de lanzamiento. Hoy en día, la mayoría de los programadores aprovechan uno de los dos modelos de desarrollo para ofrecer software de calidad: Gitflow y el desarrollo basado en troncos.

Conclusión

Git es mucho más que una herramienta; es un pilar esencial en el desarrollo moderno de software, permitiendo una colaboración eficiente y un control preciso sobre los cambios en el código. Desde comandos básicos como git init hasta estrategias avanzadas como Git Flow y Trunk Based Development (TBD), Git ofrece una flexibilidad que se adapta a las necesidades de cualquier equipo, ya sea pequeño o grande.

Por un lado, Git Flow proporciona un enfoque estructurado para gestionar ramas y garantizar que el desarrollo, las versiones y las correcciones se organicen meticulosamente. Por otro lado, TBD fomenta la integración continua, ideal para equipos que priorizan ciclos de entrega rápidos y simplificados. Ambas estrategias destacan la versatilidad de Git como una herramienta para impulsar la productividad y garantizar la calidad en proyectos de software.

Para finalizar, comprender Git y aplicar las estrategias adecuadas no solo mejora el flujo de trabajo, sino que también permite a los equipos mantener un enfoque ágil y escalable, factores críticos en el éxito de cualquier proyecto en la actualidad. La clave está en elegir las prácticas que mejor se adapten a las metas del equipo y el entorno en el que se desarrolla.

Bibliografía

Pregunta 1:
Wikipedia. (s. f.). Git. Wikipedia. Recuperado de Git - Wikipedia, la enciclopedia libre

---

Pregunta 2:
Atlassian. (s. f.). git init | Atlassian Git Tutorial. Recuperado de git init | Atlassian Git Tutorial

---

Pregunta 3:
Arsys. (s. f.). Ramas en Git: qué son y para qué sirven. Blog de Arsys. Recuperado de Ramas en Git: qué son y para qué sirven | Blog de Arsys

---

Pregunta 4:

1. Arsys. (s. f.). Ramas en Git: qué son y para qué sirven. Blog de Arsys. Recuperado de Ramas en Git: qué son y para qué sirven | Blog de Arsys
2. Como Reclamar. (s. f.). ¿Cómo saber en qué rama estoy en Git? Guía completa en español. Recuperado de ¿Cómo saber en qué rama estoy en Git? Guía completa en español - Como Reclamar

---

Pregunta 5:
Leninmhs. (s. f.). La historia de Git: el sistema de control de versiones que cambió el mundo de la programación. Recuperado de La historia de Git: el sistema de control de versiones que cambió el mundo de la programación | Leninmhs

---

Pregunta 6:
(s. f.). Comandos de GIT Básicos - Guía Completa. Recuperado de Comandos de GIT Básicos - Guía Completa

---

Pregunta 7:
Atlassian. (s. f.). Flujo de trabajo de Gitflow | Atlassian Git Tutorial. Recuperado de Flujo de trabajo de Gitflow | Atlassian Git Tutorial

---

Pregunta 8:
Atlassian. (s. f.). Desarrollo basado en troncales. Recuperado de Desarrollo basado en troncales | Atlassian