Nombres: Inocencio Junior

Apellidos: Avila Gonzalez

Matricula: 2021-0836

Asignatura: Programación III

Profesor: Kelyn Tejada Belliard

Tema: Tarea 3 (Herramientas de administración de fuentes.)

❖ Desarrolle un ejercicio práctico en Azure Devops o GitHub con las siguientes características

Link de GitHub donde se desarrollen las siguientes actividades:

-Crear un proyecto.

-Utilizar la técnica Git Flow en su proyecto.

-Proyecto funcional.

Inocencio-Junior-Avila / Manager-DataBase "github.com"

❖ Desarrolla el siguiente Cuestionario

1-Que es Git?

Git es un software de control de versiones desarrollado por Linus Torvalds que tiene en cuenta la eficiencia, confiabilidad y compatibilidad de las aplicaciones de control de versiones cuando tienen una gran cantidad de archivos de código fuente.

Su propósito es rastrear los cambios en los archivos de la computadora, incluida la coordinación del trabajo que diferentes personas están haciendo en los archivos compartidos en un repositorio de código.

Inicialmente, Git estaba pensado como un motor de bajo nivel en el que otros, como Cogito o StGIT, pudieran escribir la interfaz de usuario o el frontend. Sin embargo, 2Git ha evolucionado desde entonces hasta convertirse en un sistema de control de versiones con todas las funciones.

Ya hay algunos proyectos grandes que utilizan Git, sobre todo el grupo de programación del kernel de Linux.

2-Para que funciona el comando Git init?

El comando git init crea un nuevo repositorio Git. Se puede usar para convertir un proyecto existente sin versionar en un repositorio de Git o para inicializar un nuevo repositorio vacío.

La mayoría de los demás comandos de Git no están disponibles fuera del repositorio inicializado, por lo que suele ser el primer comando que se ejecuta en un nuevo proyecto.

Ejecutar git init crea un subdirectorio .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 extraída actualmente.

Además del directorio .git, en el directorio raíz del proyecto se almacenan los proyectos existentes sin modificar (a diferencia de SVN, Git no requiere un subdirectorio .git en cada subdirectorio).

De forma predeterminada, git init inicializará la configuración de Git en la ruta del subdirectorio .git. Puedes modificar y personalizar esta ruta si quieres que se encuentre en otros sitios. Además, puede configurar la variable de entorno $GIT_DIR en una ruta personalizada para que git init inicialice los archivos de configuración de Git allí. También puede usar el argumento --separate-git-dir para lograr el mismo resultado. Lo que suele hacer con subdirectorios .git separados es almacenar archivos de configuración del sistema ocultos (.bashrc, .vimrc, etc.) en el directorio principal, mientras que la carpeta .git se almacena en otro lugar.

4-Que es una rama?

Una rama de Git es solo un puntero móvil que apunta a una de esas confirmaciones. La rama predeterminada de Git es la rama maestra.

Con el primer compromiso que creamos, esta rama maestra principal se creará apuntando a ese compromiso. En cada compromiso de cambio que hagamos, la rama avanzará automáticamente.

La creación de ramas es una función disponible en la mayoría de los sistemas de control de versiones modernos. La bifurcación a otro sistema de control de versiones puede llevar mucho tiempo y ocupar mucho espacio de almacenamiento. En Git, las ramas son parte del proceso de desarrollo diario. Las ramas de Git son punteros poderosos para instantáneas de sus cambios. Cada vez que desee agregar una nueva función o corregir un error, independientemente del tamaño, genera una nueva rama para adaptarse a estos cambios. Esto dificulta la fusión de código inestable en la base de código principal y le brinda la oportunidad de limpiar su historial futuro antes de fusionarlo en la rama principal.

3-Como saber es que rama estoy?

Suponiendo que el repositorio en el que está trabajando contiene ramas preexistentes, puede cambiar de una de esas ramas a otra con git checkout. Para ver qué ramas están disponibles y cómo se llama la rama actual, ejecute git branch.

```
$>git Branch
Main
another_branch
feature_inprogress_branch
$>git checkout feature_inprogress_branch
```

El ejemplo anterior muestra cómo ver la lista de sucursales disponibles al ejecutar el comando git branch y cómo cambiar a una sucursal específica, en este caso, feature_inprogress_branch.

5-Quien creo git?

Git es un proyecto de código abierto maduro y mantenido activamente desarrollado originalmente en 2005 por Linus Torvalds, famoso creador del kernel del sistema operativo Linux.

6-Cuales son los comandos más esenciales de Git?

- **git init** - inicializa un repositorio.
- **git add** - agrega archivos al index.
- **git commit** - crea un commit, sólo los archivos que estén en el index serán incluídos en el commit.
- **git log** - muestra el historial de commits.
- **git show** - muestra la información de un commit específico incluyendo los cambios que se realizaron en ese commit.
- **git status** - muestra el estado del repositorio.
- **git merge** - fusiona una o más ramas con otra rama activa y crea automáticamente un nuevo commit si no hay conflictos.
- **git push** - envía todos los objetos modificados localmente al repositorio remoto.
- **git remote** - muestra todas las versiones remotas de tu repositorio.
- **Git clone** - es un comando para descargarte el código fuente existente desde un repositorio remoto (como Github, por ejemplo). En otras palabras, Git clone básicamente realiza una copia idéntica de la última versión de un proyecto en un repositorio y la guarda en tu ordenador.

- **git pull** - obtiene los archivos del repositorio remoto y los combinacon el local.

- **git branch** - para listar las ramas existentes, incluyendo las ramas remotas, si se proporciona '-a'. Crea una nueva rama si se proporciona un nombre.

- **git reset HEAD** - remueve uno o más archivos del index.

7-Que es git Flow?

Git Flow es una estrategia creada para mejorar la organización de las sucursales dentro de un repositorio, haciendo más fluido el proceso de nuevas funcionalidades y lanzamientos.

El término es aún reciente. Fue publicado en 2010 por el ingeniero de software holandés Vincent Driessen. En él, detalló cuáles fueron sus pensamientos al crear este modelo para usar Branch.

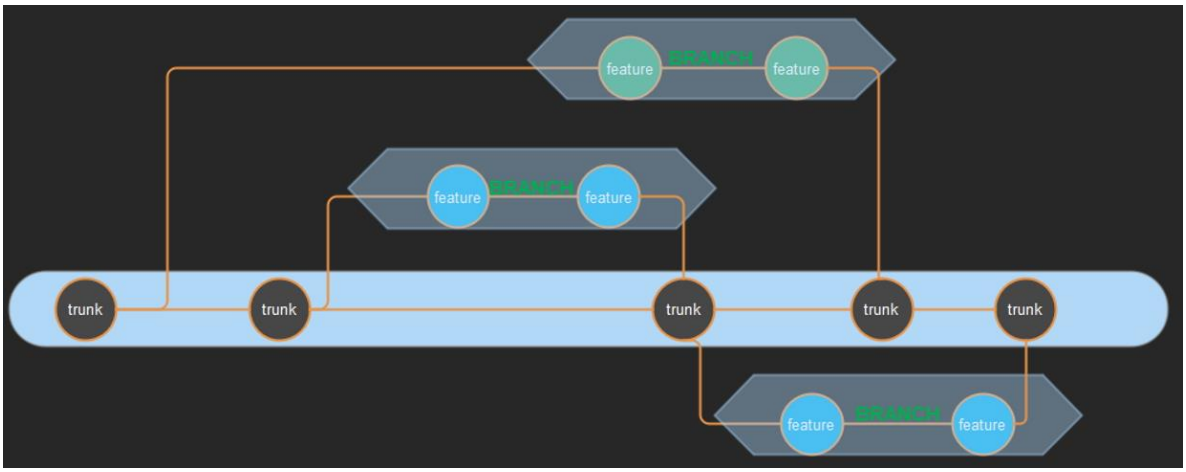
8-Que es trunk based development?

El trunk based development o desarrollo basado en troncales es una técnica de control de versiones en la que los desarrolladores combinan con frecuencia pequeñas actualizaciones en el "troncal" o rama principal.

A medida que se desarrollaron los sistemas de control de versiones, surgieron diferentes estilos de desarrollo que permitieron a los programadores encontrar errores más fácilmente, codificar en paralelo con colegas y acelerar el ritmo de los lanzamientos. Hoy en día, la mayoría de los desarrolladores confían en uno de los dos modelos de desarrollo para ofrecer software de calidad: Git Flow y Trunk Based Development.

Git Flow, que se popularizó por primera vez, es un modelo de desarrollo más rígido en el que solo ciertas personas pueden aprobar

cambios en el código principal. El desarrollo basado en troncos es un modelo más abierto. Todos los desarrolladores tienen acceso al código principal para que los equipos puedan iterar rápidamente y hacer que CI y CD funcionen.



El desarrollo basado en troncos es otro modelo de trabajo, en este caso más efectivo para DevOps, porque te permite acelerar el ciclo.