Nombre:
Pedro Rosario

Matrícula:
2019-8594

Materia:
Programación lll

Maestro:
Kelyn tejada

Programacion 3

Tarea GIT
Calificación: la que indica la plataforma.

Punto1:
Desarrolla el siguiente Cuestionario

¿Qué es Git?

Git es un sistema de control de versiones distribuido, lo que significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan la copia del repositorio con la del servidor. Este paradigma es distinto del control de versiones centralizado, donde los clientes deben sincronizar el código con un servidor antes de crear nuevas versiones.

La flexibilidad y popularidad de Git lo convierten en una excelente opción para cualquier equipo. Muchos desarrolladores y graduados universitarios ya saben cómo usar Git. La comunidad de usuarios de Git ha creado recursos para entrenar a los desarrolladores y la popularidad de Git facilita recibir ayuda cuando se necesita. Casi todos los entornos de desarrollo tienen compatibilidad con Git y las herramientas de línea de comandos de Git implementadas en todos los sistemas operativos principales.

2. ¿Cuál es el propósito del comando git init en Git?

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.

3. ¿Qué representa una rama en Git y cómo se utiliza?

Una rama (o branch) en Git representa una línea de desarrollo independiente dentro de un repositorio. Las ramas permiten a los desarrolladores trabajar en diferentes características, correcciones de errores o experimentos de manera aislada del resto del proyecto. Esto facilita la colaboración y el desarrollo paralelo sin interferir con el código principal.

Cómo se utilizan las ramas en Git:

Creación de una nueva rama:

Para crear una nueva rama, se utiliza el comando git branch nombre_de_la_rama. Esto crea una nueva rama a partir del punto en el que se encuentra actualmente tu rama activa (normalmente main o master).

Cambio a una rama diferente:

- Para cambiar a una rama existente, se utiliza el comando git checkout nombre_de_la_rama. A partir de Git 2.23, se recomienda usar git switch nombre_de_la_rama para mayor claridad.

Trabajo en una rama:

Una vez que estás en una rama, todos los cambios que hagas se registrarán en esa rama. Puedes añadir, modificar y eliminar archivos, y luego hacer commits para registrar estos cambios.

Fusión de ramas:

Cuando una rama contiene cambios que deseas integrar en otra rama (por ejemplo, la rama principal), puedes usar git merge nombre de la rama mientras estás en la rama de destino. Esto combina los cambios de la rama especificada en la rama actual.

Eliminación de una rama:

Una vez que una rama ya no es necesaria (por ejemplo, después de que sus cambios se han fusionado en la rama principal), puedes eliminarla usando git branch d nombre de la rama

4. ¿Cómo puedo determinar en qué rama estoy actualmente en Git?

1. Usando git branch:

El comando git branch muestra una lista de todas las ramas locales en tu repositorio y destaca la rama en la que estás actualmente con un asterisco (*) al principio de su nombre.

Codigo

```
git Branch
```

2. Usando git status:

El comando git status proporciona información sobre el estado actual de tu repositorio, incluyendo la rama en la que te encuentras actualmente.

Codigo:

```
git status
```

La salida incluirá una línea similar a:
Codigo

```
On branch nombre_de_la_rama
```

3. Usando git symbolic-ref:

El comando git symbolic-ref puede usarse para mostrar la referencia simbólica de la rama actual.

Codigo:

```
git symbolic-ref --short HEAD
```

4. Consultando el prompt de la línea de comandos:
Muchos entornos de línea de comandos, como bash, zsh, y otros, pueden configurarse para mostrar la rama actual en el prompt. Por ejemplo, en bash, puedes modificar tu archivo. bashrc o .bash_profile para incluir algo como:

Codigo

```
parse_git_branch() {
  git branch 2>/dev/null | sed -n '/\* /s///p'
}

PS1="\u@\h
\[\033[32m\]\w\[\033[33m\]\$(parse_git_branch)\[\033[00m\] $ "
```

Esto hará que el prompt muestre la rama actual cuando te encuentres en un repositorio de Git.

5. ¿Quién es la persona responsable de la creación de Git y cuándo fue desarrollado?

Git fue creado por Linus Torvalds en el año 2005.

El desarrollo de Git comenzó cuando el equipo de desarrollo del kernel de Linux dejó de usar el sistema de control de versiones propietario BitKeeper, debido a un cambio en los términos de uso del software. Torvalds decidió entonces crear un nuevo sistema de control de versiones que cumpliera con sus necesidades y las del equipo de desarrollo del kernel de Linux, priorizando velocidad, integridad de los datos y soporte para el desarrollo distribuido. Git fue desarrollado con estas metas en mente y rápidamente ganó popularidad en la comunidad de desarrollo de software.

6. ¿Cuáles son algunos de los comandos esenciales de Git y para qué se utilizan?

Algunos de los comandos esenciales de Git son:

- git add: Agrega archivos al índice de Git para ser versionados.
- git commit: Crea un nuevo commit con los cambios añadidos al índice.
- git push: Envía los cambios locales al repositorio remoto.
- git pull: Descarga los cambios del repositorio remoto y los fusiona con la rama local.
- git branch: Crea, elimina o cambia a una rama diferente.

7. ¿Puedes mencionar algunos de los repositorios de Git más reconocidos y utilizados en la actualidad?

Algunos de los repositorios de Git más reconocidos y utilizados en la actualidad son:

1. GitHub
2. GitLab
3. Bitbucket

Punto 2:
Desarrolle un ejercicio práctico en Azure Devops o GitHub con las siguientes características
Link con respuestas:

● Utiliza OneDrive paa la entrega de la teoría o sube el documento al repositorio del proyecto.
● Comparte el documento y asegúrate de configurarlo como público o proporciona acceso al profesor y al monitor. (Si das los accesos no se corregirá)

Link del proyecto en GitHub o Azure Devops:
Tareas o requerimientos (Seguir los pasos indicados):
Crea issues para cada tarea.
En el ejemplo explicado verá que la estructura que debe tener el proyecto será la siguientes:

Branch- main -> Es la rama principal o rama productiva.
QA - > Esta es la rama utilizada para el entorno de prueba o entorno de QA donde los equipos de calidad son los responsables de hacer toda la prueba necesaria del codigoen este ambiente.