Desarrolla el siguiente Cuestionario

1. ¿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.

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

Crear 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.

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

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. En este artículo, repasamos para qué sirven las ramas de Git y cómo podemos trabajar con ellas en un proyecto.

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.

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

Utilizando el comando *show-branch* que nos muestra un resumen de las ramas de un proyecto y sus últimas modificaciones (o commits).

git show-branch

5. ¿Quién es la persona responsable de la creación de Git y cuándo fue desarrollado? Linus Torvalds en abril de 2005 cuando el sistema propietario de control de versiones usado en el desarrollo del kernel de Linux desde 2002, BitKeeper revoco la licencia gratuita usada para el desarrollo del Linux. El titular del copyright de Bitkepper Larry McVoy, afirmó que Andrew Tridgell había creado Source Puller utilizando ingeniería inversa a partir de los protocolos de BitKeeper.

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

1. Git clone

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.

Hay un par de formas de descargar el código fuente, pero principalmente yo prefiero **clonar de la forma con https**:

git clone <https://link-con-nombre-del-repositorio>

2. Git branch

Las ramas (branch) son altamente importantes en el mundo de Git. Usando ramas, varios desarrolladores pueden trabajar en paralelo en el mismo proyecto simultáneamente. Podemos usar el comando git branch para crearlas, listarlas y eliminarlas.

Creando una nueva rama:

git branch <nombre-de-la-rama>

3. Git checkout

Este es también uno de los comandos más utilizados en Git. Para trabajar en una rama, primero tienes que cambiarte a ella. Usaremos **git checkout** principalmente para cambiarte de una rama a otra. También lo podemos usar para chequear archivos y commits.

git checkout <nombre-de-la-rama>

Hay algunos pasos que debes seguir para cambiarte exitosamente entre ramas:

- Los cambios en tu rama actual tienen que ser confirmados o almacenados en el guardado rápido (stash) antes de que cambies de rama.
- La rama a la que te quieras cambiar debe existir en local.

4. Git status

El comando de git status nos da toda la información necesaria sobre la rama actual.

git status

Podemos encontrar información como:

- Si la rama actual está actualizada
- Si hay algo para confirmar, enviar o recibir (pull).
- Si hay archivos en preparación (staged), sin preparación(unstaged) o que no están recibiendo seguimiento (untracked)
- Si hay archivos creados, modificados o eliminados

5. Git add

Cuando creamos, modificamos o eliminamos un archivo, estos cambios suceden en local y no se incluirán en el siguiente commit (a menos que cambiemos la configuración).

Necesitamos usar el comando git add para incluir los cambios del o de los archivos en tu siguiente commit.

Añadir un único archivo:

git add <archivo>

Añadir todo de una vez:

git add -A

6. Git commit

Este sea quizás el comando más utilizado de Git. Una vez que se llega a cierto punto en el desarrollo, queremos guardar nuestros cambios (quizás después de una tarea o asunto específico).

Git commit es como establecer un punto de control en el proceso de desarrollo al cual puedes volver más tarde si es necesario.

También necesitamos escribir un mensaje corto para explicar qué hemos desarrollado o modificado en el código fuente.

git commit -m "mensaje de confirmación"

7. Git push

Después de haber confirmado tus cambios, el siguiente paso que quieres dar es enviar tus cambios al servidor remoto. Git push envía tus commits al repositorio remoto.

git push <nombre-remoto> <nombre-de-tu-rama>

De todas formas, si tu rama ha sido creada recientemente, puede que tengas que cargar y subir tu rama con el siguiente comando:

git push --set-upstream <nombre-remoto> <nombre-de-tu-rama>

or

git push -u origin <nombre-de-tu-rama>

8. Git pull

El comando **git pull** se utiliza para recibir actualizaciones del repositorio remoto. Este comando es una combinación del **git fetch** y del **git merge** lo cual significa que cundo usemos el git pull recogeremos actualizaciones del repositorio remoto (git fetch) e inmediatamente aplicamos estos últimos cambios en local (git merge).

git pull <nombre-remoto>

Esta operación puede generar conflictos que tengamos que resolver manualmente.

9. Git revert

A veces, necesitaremos deshacer los cambios que hemos hecho. Hay varias maneras para deshacer nuestros cambios en local y/o en remoto (dependiendo de lo que necesitemos), pero necesitaremos utilizar cuidadosamente estos comandos para evitar borrados no deseados.

Una manera segura para deshacer nuestras commits es utilizar **git revert**. Para ver nuestro historial de commits, primero necesitamos utilizar el **git log--oneline:**

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

TensorFlow: Un repositorio de Google para el desarrollo del popular framework de aprendizaje automático y redes neuronales.

React: Un repositorio de Facebook para el desarrollo de la biblioteca de JavaScript React, utilizada ampliamente para construir interfaces de usuario.

Linux Kernel: El repositorio oficial del kernel de Linux, mantenido por Linus Torvalds y otros desarrolladores.

VS Code: El repositorio de Microsoft para Visual Studio Code, un editor de código fuente gratuito y de código abierto.

Bootstrap: Un repositorio para el framework front-end de código abierto Bootstrap, utilizado para diseñar sitios web y aplicaciones web.