1. Introducción

El presente documento es una introducción a los comandos básicos de Git que se utilizarán a lo largo del curso para subir los reportes de prácticas y tener un control de las versiones de los archivos que conformen el repositorio de cada alumno. Si se desea profundizar acerca del uso de esta herramienta, se recomienda consultar la documentación oficial.

En un ***sistema de control de versiones*** se registran los cambios de uno o múltiples archivos a lo largo del tiempo, esto con la finalidad de tener un control sobre los cambios que se llevan a cabo en un proyecto. Principalmente, existen tres tipos: local, centralizado o distribuido.

2. Git

Es un sistema de control de versiones distribuido creado por *Linus Torvalds* en 2005. En este sistema, al no depender de un servidor local, cada desarrollador debe de clonar una copia del repositorio completo en sus máquinas por lo que la mayoría de las operaciones se realizarán de manera rápida al ejecutarse de manera local. Para la comunicación con el servidor central se utilizan los protocolos HTTPS, SSH o Git Protocol, en el mercado existen varios servidores en línea que ofrecen el almacenamiento en la nube de repositorios remotos [1].

Debido al uso de GitHub Classroom en el desarrollo del curso, se utilizará GitHub como servidor central para almacenar los repositorios remotos de las entregas de reportes, por lo que primero se debe contar con una cuenta gratuita en esta plataforma. Una vez que se tiene la cuenta, es posible instalar ***git*** en la computadora local del desarrollador si el siguiente comando no sale exitoso:

```
git --version
```

Código fuente 1: Consulta de versión de git instalado

2.1. Instalación de Git

- Para Windows:

  I. Descargar el instalador oficial de Git desde su sitio oficial: https://git-scm.com/download/win
  II. Iniciar el instalador y dejar todas las configuraciones por default.

  NOTA: Para el caso de Windows y que el alumno desee las ***décimas extras*** que se otorgarán por el uso de esta herramienta, es obligatorio sólo utilizar el ambiente precargado *MinGW64*, que nos permite operar en Windows con los mismos comandos y herramientas disponibles en Linux. Para acceder a éste, abrir el Explorador de archivos y dar clic derecho sobre la carpeta de trabajo como se muestra en la siguiente figura 1:

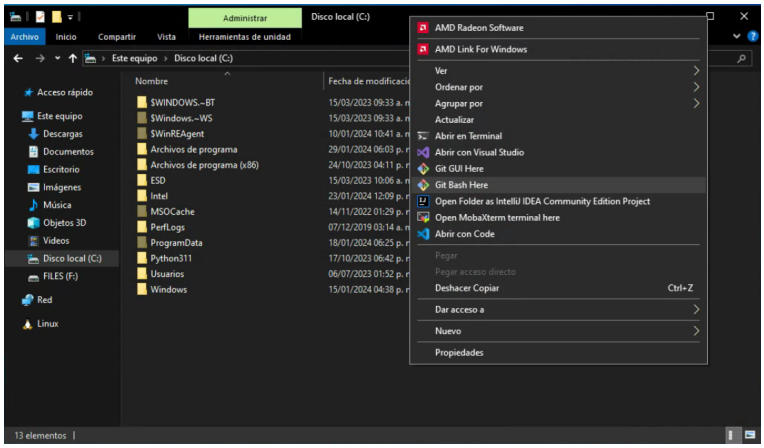

Figura 1: Acceso a Git Bash

Dar clic en `Git Bash Here` para acceder al entorno de *MinGW64* como el que se muestra en la figura 2:

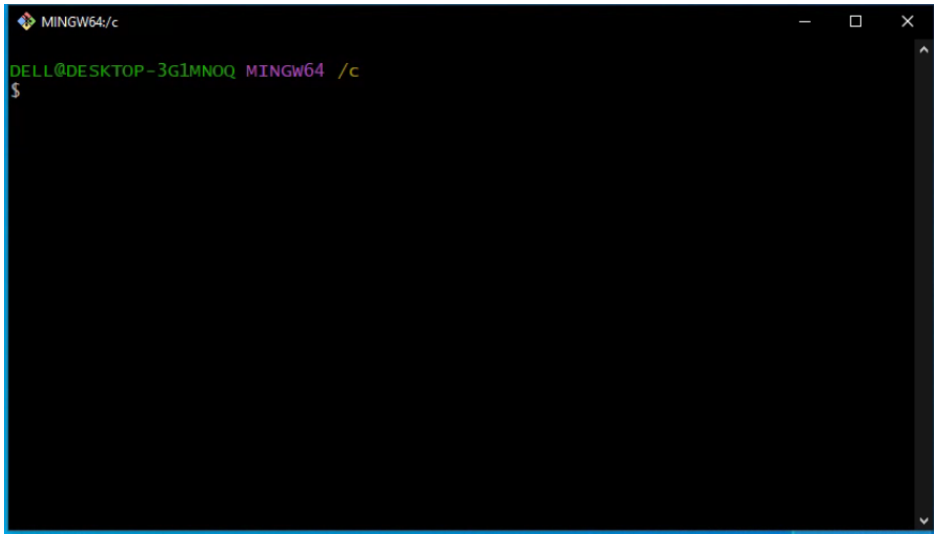

Figura 2: Ambiente MinGW64

- Para Ubuntu (Debian):

  I. Ejecutar el siguiente comando en la terminal:

```
$ sudo apt install git-all
```

Código fuente 2: Instalación de git (Debian)

- Para CentOS (Fedora):

  I. Ejecutar el siguiente comando en la terminal:

```
$ sudo yum install git-all
```

Código fuente 3: Instalación de git (Fedora)

Para instalaciones en otros sistemas operativos se puede consultar la documentación oficial.

2.2. Configuración

1. Verificar que la instalación de git se realizó de manera correcta en nuestro entorno de trabajo:

```
$ git --version
```

Código fuente 4: Versión de git instalada

2. Establecer el nombre de usuario que tienes registrado en GitHub:

```
$ git config --global user.name "<username>"
```

Código fuente 5: Configurar nombre del usuario de GitHub

3. Configurar el correo electrónico que tienes registrado en GitHub:

```
$ git config --global user.email <email>
```

Código fuente 6: Configurar correo electrónico de GitHub

4. Comprobar que la configuración se realizó de manera correcta:

```
$ git config --list
```

Código fuente 7: Configuración de git

3. Flujo de trabajo con Git

A continuación se describirán los pasos para trabajar el repositorio creado por GitHub Classroom para subir reportes y demás material que sea solicitado para el desarrollo de la prácticas:

3.1. Importando el repositorio

Para iniciar a trabajar, se requiere copiar el repositorio remoto donde se tiene los archivos iniciales para trabajar con la práctica. Para realizar esta acción, es necesario utilizar un protocolo de comunicación con GitHub desde nuestra computadora local para subir los cambios realizados, para este documento se detallan los protocolos más utilizados en la industria: HTTPS y SSH.

3.1.1. Clonación por SSH

Se recomienda ampliamente utilizar esta opción para tener un canal de comunicación directo y seguro con el servidor de GitHub. Solo realizar estas configuraciones cada vez que se desee verificar un nuevo dispositivo de confianza y vincular a la cuenta de GitHub:

I) Generar una llave con el correo electrónico registrado en GitHub:

```
$ ssh-keygen -t rsa -C "<email>" -b 4096
```

Código fuente 8: Generación de llave para comunicación por SSH

Nos solicitará la ubicación para generar el archivo con la llave SSH, se recomienda dejar la configuración por defecto para la conexión con el repositorio remoto en GitHub presionando la tecla *Enter*. Por otra parte, nos pregunta por una frase -*Enter passphrase (empty for no passphrase):*- la cual se utilizará para generar y autenticar la llave SSH (se recomienda ampliamente no dejar la frase en blanco). Una vez introducida la frase de confirmación, se genera el siguiente mensaje (figura 3) que nos indica que la llave fue creada correctamente:

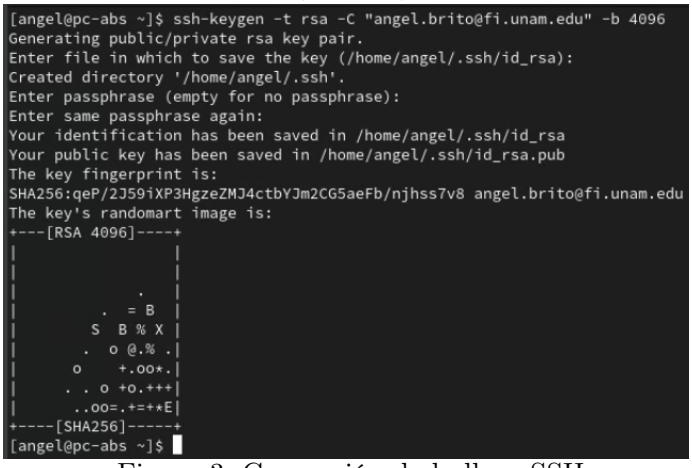

Figura 3: Generación de la llave SSH

II) Copiar el resultado del siguiente comando que nos muestra la llave pública generada:

```
$ cat ~/.ssh/id_rsa.pub
```

Código fuente 9: Llave pública SSH generada

III) Pegar la llave copiada en la sección *Key* de `Add new SSH Key` en tu perfil de GitHub que se accede desde el siguiente link: https://github.com/settings/ssh/new. Cabe resaltar que el contenido de la llave inicia con la cadena `ssh-rsa` y termina con el correo electrónico ingresado en el paso anterior.

IV) Ingresar un nombre por la cual vas a identificar este dispositivo en la sección *Title* y dar clic en "Add SSH key".

Para la clonación por HTTPS no se requieren realizar configuraciones previas, por lo que los siguientes pasos aplican para cualquier protocolo de comunicación elegido para conectarse con el servidor de GitHub.

3.1.2. Clonación del repositorio remoto

1. Acceder a GitHub y entrar al repositorio que se desea copiar a la computadora local.
2. Dar clic en la opción <> `Code` y copiar la URL de la sección correspondiente del protocolo a utilizar (HTTPS o SSH) como se muestra en la siguiente figura 4:

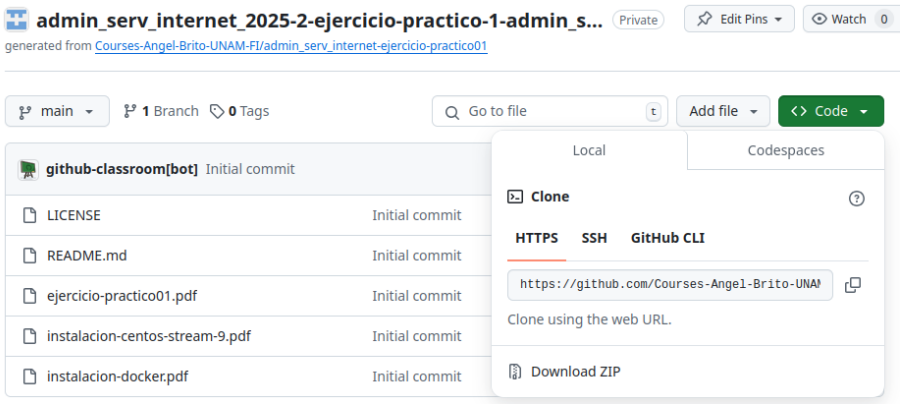

Figura 4: Opciones de clonación en GitHub

3. Clonar el repositorio en la computadora local, con el siguiente comando:

```
$ git clone <url-repositorio>
```

Código fuente 10: Clonación del repositorio

Con este comando se crea una carpeta llamada `practica<XX>-<username_GitHub>` en donde se ubicará el Directorio de trabajo con todos los archivos que tenga el repositorio remoto (se colocan en disco para que se puedan usar o modificar) y una base de datos comprimida en el subdirectorio *.git* para realizar el seguimiento de los cambios de manera local.

3.2. Trabajando en el directorio de trabajo

Una vez que se han realizado cambios en el Directorio de trabajo es importante indicarle a Git que se desean confirmar dichas modificaciones:

1. Ingresar al directorio de trabajo:

```
$ cd <ruta_absoluta>/practica<XX>-<username_GitHub>
```

Código fuente 11: Ingresando al directorio de trabajo

2. Obtener el estatus del repositorio:

```
$ git status
```

Código fuente 12: Estatus del repositorio

Este comando nos muestra el estado del directorio de trabajo (*Working Directory*), el cual, si se encuentra algún cambio, nos indica en que estado se encuentran los archivos modificados/agregados:

- **Sin rastrear (*untracked files*)**: archivos que se requieren agregar a git para darle seguimiento.
- **Modificados (*modified*)**: archivos modificados pero que todavía no se han confirmado en la base de datos local (subdirectorio *.git*).
- **Preparados (*staged*)**: archivos modificados en su versión actual que están listos para ser agregados en la próxima confirmación (*commit*) a la base de datos local.

3. Agregar el archivo modificado al área de preparación (se puede utilizar el comodín **.** para agregar todos los archivos cambiados):

```
$ git add <nombre_archivo>
```

Código fuente 13: Agregar archivos al Área de preparación

La área de preparación o zona de ensayo (*Staging Area*) almacena la información acerca de lo que va a ir en la próxima confirmación al repositorio local.

4. Confirmar que el o los archivo(s) en el área de preparación se pueden agregar al repositorio local:

```
$ git commit -m "<mensaje_descriptivo_cambios>"
```

Código fuente 14: Confirmación de cambios

3.3. Trabajando con el repositorio remoto

Una vez que se han realizado todos los cambios en el repositorio local, se pueden subir estas modificaciones al repositorio remoto con el siguiente comando:

```
$ git push origin main
```

Código fuente 15: Agregar cambios del repositorio local al remoto

Para traer los cambios en el repositorio remoto al local se ejecuta el siguiente comando:

```
$ git pull origin main
```

Código fuente 16: Descargar versión actual del repositorio remoto en el local

Referencias

[1] B. S. Scott Chacon, *Pro Git*. Apress, 2024.

[2] D. Barajas y M. Suárez., «Control de versiones con GIT,» en *Programa de Becas de Formación en TIC.*, Dirección General de Cómputo y de Tecnologías de Información y Comunicación, UNAM, ed., 2022.

[3] K. Sulaimon, «Introduction to Git and GitHub,» en *Google*, Coursera, ed., 2023.