**Alumna: Bauque Bernardita**

**Mochila del viajero**

**Práctica individual obligatoria**

**Consigna**

Posicionarse en la rama **Apellido\_Nombre** ya creada y subir a la carpeta **Git\_y\_Github\_Apellido\_Nombre**, también creada en la consigna anterior, un archivo pdf con la siguiente consigna:

* Una **cheatsheet** (recordatorio) de los comandos más utilizados hasta el momento, con una explicación de la sintaxis de cada uno y qué hacen. Tomar como referencia la cheatsheet de la clase de la terminal:

1. **CONFIGURAR HERRAMIENTAS**

|  |  |
| --- | --- |
| **$ git config --global user.name "[name]"** | Configura la información del usuario para todos los repositorios locales  Establece el nombre que desea esté anexado a sus transacciones  de commit |
| **$ git config --global user.email "[email address]"** | Establece el e-mail que desea esté anexado a sus transacciones de commit |
| **$ git config --global color.ui auto** | Habilita la útil colorización del producto de la línea de comando. |

1. **CREAR REPOSITORIOS** (Inicia un nuevo repositorio u obtiene uno de una URL existente)

|  |  |
| --- | --- |
| **$ git init [project-name]** | Crea un nuevo repositorio local con el nombre especificado |
| **$ git clone [url]** | Descarga un proyecto y toda su historia de versión |

1. **EFECTUAR CAMBIOS** (Revisa las ediciones y elabora una transacción de commit)

|  |  |
| --- | --- |
| **$ git status** | Enumera todos los archivos nuevos o modificados que se deben confirmar |
| **$ git add [file]** | Toma una instantánea del archivo para preparar la versión |
| **$ git reset [file]** | Mueve el archivo del área de espera, pero preserva su contenido |
| **$ git diff** | Muestra las diferencias de archivos que no se han enviado aún al  área de espera |
| **$ git diff --staged** | Muestra las diferencias del archivo entre el área de espera y la última  versión del archivo. |
| **$ git commit -m "[descriptive message]"** | Registra las instantáneas del archivo permanentemente en  el historial de version |

1. **CAMBIOS GRUPALES**

|  |  |
| --- | --- |
| **$ git branch** | Nombra una serie de commits y combina esfuerzos ya culminados  Enumera todas las ramas en el repositorio actual |
| **$ git branch [branch-name]** | Crea una nueva rama |
| **$ git checkout [branch-name]** | Cambia a la rama especificada y actualiza el directorio activo |
| **$ git merge [branch]** | Combina el historial de la rama especificada con la rama actual |
| **$ git branch -d [branch-name]** | Borra la rama especificada |

1. **NOMBRES DEL ARCHIVO DE REFACTORIZACIÓN** (Reubica y retira los archivos con versión)

|  |  |
| --- | --- |
| **$ git rm --cached [file]** | Retira el archivo del control de versiones, pero preserva el archivo a  nivel local |
| **$ git rm [file]** | Borra el archivo del directorio activo y pone en el área de espera el  archivo borrado |
| **$ git mv [file-original] [file-renamed]** | Cambia el nombre del archivo y lo prepara para commit |

1. **SUPRIMIR TRACKING (**Excluye los archivos temporales y las rutas)

|  |  |
| --- | --- |
| **$ git ls-files --other --ignored --exclude-standard** | Enumera todos los archivos ignorados en este proyecto |
| **\*.log**  **build/**  **temp-\*** | Un archivo de texto llamado .gitignore suprime la creación accidental  de versiones de archivos y rutas que concuerdan con los patrones  especificados |

1. **GUARDAR FRAGMENTOS** (Almacena y restaura cambios incompletos)

|  |  |
| --- | --- |
| **$ git stash** | Almacena temporalmente todos los archivos tracked modificados |
| **$ git stash list** | Enumera todos los sets de cambios en guardado rápido |
| **$ git stash pop** | Restaura los archivos guardados más recientemente |
| **$ git stash drop** | Elimina el set de cambios en guardado rápido más reciente |

1. **REPASAR HISTORIAL (**Navega e inspecciona la evolución de los archivos de proyecto)

|  |  |
| --- | --- |
| **$ git log** | Enumera el historial de la versión para la rama actual |
| **$ git log --follow [file]** | Enumera el historial de versión para el archivo, incluidos los cambios  de nombre |
| **$ git diff [first-branch]...[second-branch]** | Muestra las diferencias de contenido entre dos ramas |
| **$ git show [commit]** | Produce metadatos y cambios de contenido del commit especificado |

1. **REHACER COMMITS (**Borra errores y elabora historial de reemplazo)

|  |  |
| --- | --- |
| **$ git reset [commit]** | Deshace todos los commits después de [commit], preservando los  cambios localmente |
| **$ git reset --hard [commit]** | Desecha todo el historial y regresa al commit especificado |

1. **SINCRONIZAR CAMBIOS (**Registrar un marcador de repositorio e intercambiar historial de versión)

|  |  |
| --- | --- |
| **$ git fetch [bookmark]** | Descarga todo el historial del marcador del repositorio |
| **$ git merge [bookmark]/[branch]** | Combina la rama del marcador con la rama local actual |
| **$ git push [alias] [branch]** | Carga todos los commits de la rama local al GitHub |
| **$ git pull** | Descarga el historial del marcador e incorpora cambios |