

Lleva un control de versiones que guarda el historial, trabaja de manera descentralizada por lo que en equipos de trabajo grandes todos tiene acceso a una copia del proyecto. Las ventajas es que almacena, guarda el historial, flexibilidad en el equipo de trabajo, hacer Rollback que es retroceder a versiones anteriores para recuperar contenido que anteriormente trabajaba.

Instalar GIT

git config –h 🡪 Ve todas las configuraciones de git

Cmnder Console ya lo traé instalado por default

git --version

git update

git update-git-for-windows (reiniciar Cmder)

git config --global user.name "mi\_Git"

git config --global user.email [kenth1977@gmail.com](mailto:kenth1977@gmail.com)

git config --global core.autocrlf true (Compatibilidad Linux | windows)

Cmder

dir o ls 🡨 Ver directorios

pwd 🡨 muestra en que directorio me encuentro

cd 🡨 Change Directory

cd.. 🡨 Retorceder un directorio y cd .. con espacio en linux

cd\ 🡨 Ir a la Raíz (Windows)

md 🡨 Make Directory y mkdir en linux

El flujo de trabajo es la forma como se va guardando los archivos antes de llegar al servidor final.

Computadora 🡨 Control sobre los archivos donde se crea se borra

Stage 🡨 Es la parte donde se elige que si y que no pasar al servidor

Commit 🡨 Es donde se comenta y se prepara el archivo para subir al servidor

Server 🡨 Es el repositorio final donde se almacena el o los archivos

Dentro de la carpeta del proyecto Flask que se va a crear o se ha creado inicializar un repositorio git

git init 🡨 Se crea un archivo oculto dentro de la carpeta del proyecto.

ls -a 🡨 Muestra files o carpetas ocultas

git status 🡨 Muestra en color rojo los archivos que se prepararo para subir al servidor

git add . 🡨 Agrega todos los archivos git add pepino.\* agrega solo lo mencionado

y git status para ver que lo rojo ahora está en verde listo para el siguiente paso.

git commit -m " mensaje representativo de lo que se va a subir " 🡨 revisar con git status

rm 🡨 Borra un archivo antes de ser subido al stage = rm archivo.pepino

y luego realizar un commit con el cambio

mv 🡨 Cambia el nombre de la etiqueta al nuevo archivo pepino1 a pepino

y para subirlo al stage se escribe git pepino1 pepino sustituyendo pepino1 y con git commit “renombrado pepino1 a pepino”

git mv pepino1.txt pepino.tx 🡨 cambia el nombre y lo sube al stage de una vez

git restore --staged pepino.txt 🡨 Saca u omite un archivo desde el stage

git restore pepino.txt 🡨Restaura un archivo que fue sacado del stage

.gitignore 🡨 Ignora los archivos o folders que se coloquen dentro de el y de una vez crear un commit para agregarlo al server git commit –m " xxxxxx"

git status -s 🡨 muestra solo la información modificada pero que ya se encuentran dentro del stage y el archivo o folder modificado muestra una **M** antes de esos folders o archivos modificado y los que son antecedidos con un **??** es que aún no han sido ingresados al stage

git diff 🡨Muestra el contenido del archivo modificado pero que no están en el staged

git diff --staged 🡨Muestra el contenido de un archivo modificado dentro del staged

git log --oneline 🡨 Muestra el status e historial de modificaciones y es antecedido con un número id

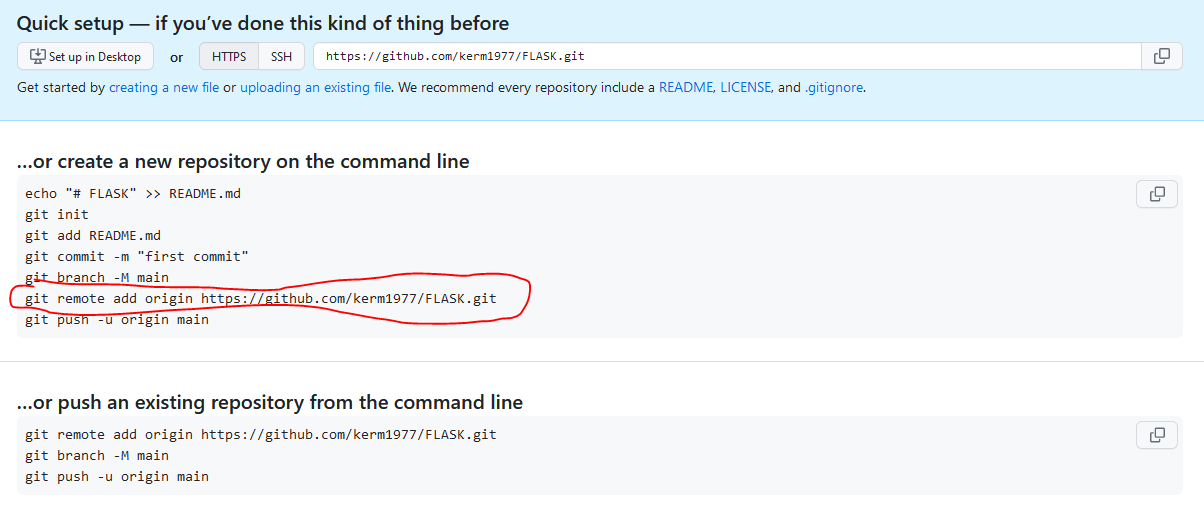
git branch 🡨 Indica sobre que rama se está donde main es la principal

git checkout –b pepino 🡨 Crea una rama que se llama pepino y todos los cambios realizados en esta rama solo afecta a la rama pepino

git checkout main 🡨 Se pasa a la rama principal para hacer un merge

git merge pepino 🡨 Traé todo el contenido de rama pepino

cat pepino.txt 🡨 Consulta el contenido de un archivo y al hacer cat se ve reflejado el contenido

Ir al contenido de github crear una cuenta y nuevo repositorio copiar el remote

Y en la consola pegar la ruta y luego push el contenido para subirlo al servidor.

git branch -M main

git remote add origin https://github.com/kerm1977/FLASK.git

git push -u origin main 🡨 Luego de conectar github solo se debe seguir enviando el con

este comando

Para traer o clonar contenido desde github a mi PC

git clone --bare <https://github.com/kerm1977/FLASK.gi>

Crear una carpeta llamarla ***djangoApp*** y dentro de esta ***inicializar git*** para luego ***instalar el env*** dentro de esta y consultar la versión para ver que todo está bien con ***python -m django --version***

Un MVC 🡪 **M**odelo **V**ista **C**ontrolador

**M**odelo 🡪 Obtiene información de una DB

**V**ista || **T**emplate 🡪 Template Muestra la información al usuario

**C**ontrolador || **V**ista 🡪 Vista Gestiona todas las comunicaciones

md dangoapp #Crea la carpeta de proyecto

cd djangoapp #Entra al directorio creado

git init #Inicializar git

python get-pip.py ##SOLO EN CASO DE QUE NO RECONOCIERA COMANDOS PIP y pip-V para consultar##

pip --version # Ver la versión de pip

python -m venv env #Crea un entorno virtual

cd env // cd scripts // activate //cd.. cd.. #Ingresa y activa al entorno virtual

pip install Django o también pip install Django==4.2.1 #Instala django

python.exe -m pip install --upgrade my\_proyect #Actualiza los repos

python -m django --version #Consulta la version instalada de django

pip install psycopg2 #Instala compatibilidad Con la DB postgrades

pip list #Ve la lista de dependencias

pip freeze > requirements.txt #Crea una lista con los requerimientos de flask

y se hace cada vez que se instala una nueva librería

django-admin startproject my\_proyect