

Tutorial Git Completo: De Casa a Trabajo

Guía Paso a Paso con Explicación Detallada de Cada Comando

TU SITUACIÓN ACTUAL:

- Usuario: `gustavomanfre`
- Carpeta local: `/home/gustavo/Documentos/repositoy-node.js`
- Repositorio GitHub: `https://github.com/gustavomanfre/repository-node.js`
- Ya tienes el Personal Access Token funcionando 

PARTE 1: ENTENDER QUÉ ES CADA COMANDO

Comandos de Navegación

```
bash
cd ./Documentos
```

¿Qué hace? `cd` = "change directory" (cambiar directorio) **Explicación:** Te mueve desde tu carpeta actual (`~` = `/home/gustavo`) a la carpeta `Documentos` **Resultado:** Ahora estás en `/home/gustavo/Documentos`

```
bash
cd ./repositoy-node.js
```

¿Qué hace? Te mueve a la carpeta del proyecto El `./` significa "desde la carpeta actual" **Resultado:** Ahora estás en `/home/gustavo/Documentos/repositoy-node.js`

Comandos Básicos de Git

```
bash
git init
```

¿Qué hace? Inicializa un repositorio Git en la carpeta actual **Crea:** Una carpeta oculta `.git` que contiene toda la información de control de versiones **Cuándo usarlo:** Solo la primera vez en una carpeta nueva

```
bash
git add .
```

Desglosando el comando:

- `git` = programa de control de versiones
- `add` = agregar archivos al "área de staging" (zona de preparación)
- `.` = punto significa "todos los archivos y carpetas de aquí"

¿Qué hace? Prepara TODOS los archivos para ser confirmados (committed) **Alternativas:**

- `git add archivo.txt` = solo un archivo específico
- `git add *.js` = solo archivos JavaScript
- `git add carpeta/` = solo una carpeta específica

```
bash
git status
```

¿Qué hace? Te muestra el estado actual del repositorio **Te dice:**

- Qué archivos están preparados para commit (en verde)
- Qué archivos han cambiado pero no están preparados (en rojo)
- En qué rama estás trabajando

```
bash
git commit -m "mensaje"
```

Desglosando:

- `commit` = confirmar/guardar los cambios
- `-m` = flag para agregar un mensaje
- `"mensaje"` = descripción de qué cambios hiciste

¿Qué hace? Crea una "foto" permanente de tus archivos con un mensaje descriptivo

```
bash
git remote add origin URL
```

Desglosando:

- `remote` = repositorio remoto (en internet)
- `add` = agregar
- `origin` = nombre estándar para el repositorio principal
- `URL` = dirección de tu repositorio en GitHub

¿Qué hace? Conecta tu carpeta local con tu repositorio en GitHub

```
bash
git push -u origin main
```

Desglosando:

- `push` = empujar/subir cambios
- `-u` = establece la conexión permanente (upstream)
- `origin` = nombre del repositorio remoto
- `main` = nombre de la rama principal

¿Qué hace? Sube todos tus commits locales a GitHub

```
bash
git pull
```

¿Qué hace? Descarga y combina cambios desde GitHub a tu carpeta local **Cuándo usarlo:** Antes de empezar a trabajar, para tener la versión más reciente

PARTE 2: CONFIGURACIÓN EN PC DE CASA (TU SITUACIÓN ACTUAL)

Paso 1: Verificar tu ubicación actual

```
bash
pwd
```

¿Qué hace? "Print Working Directory" - te dice dónde estás exactamente **Deberías ver:** `/home/gustavo/Documentos/repositoy-node.js`

Paso 2: Ver qué archivos tienes


```
bash
ls -la
```

Desglosando:

- `ls` = listar archivos
- `-l` = formato largo (con permisos, tamaños, fechas)
- `-a` = mostrar archivos ocultos (incluye `.git`)

Paso 3: Verificar si Git ya está inicializado

```
bash
git status
```

Si ves: "On branch main" o "On branch master" → Ya tienes Git inicializado  **Si ves:** "not a git repository" → Necesitas hacer `git init`

Paso 4: Verificar conexión remota

```
bash
git remote -v
```

¿Qué hace? Muestra qué repositorios remotos tienes configurados **Deberías ver:**

```
origin https://github.com/gustavomanfre/repository-node.js (fetch)
origin https://github.com/gustavomanfre/repository-node.js (push)
```

Paso 5: Configurar conexión remota (si no existe)

```
bash
git remote add origin https://github.com/gustavomanfre/repository-node.js.git
```

Nota: Si ya existe, verás "error: remoto origin ya existe" - ¡está bien!

Paso 6: Verificar qué archivos vas a subir

```
bash

# Ver tamaño total
du -sh .

# Ver archivos más grandes
find . -type f -size +10M -exec ls -lh {} \;
```

¿Por qué importante? GitHub tiene límite de 100MB por archivo y repositorios muy grandes dan problemas

PARTE 3: CREAR .gitignore PARA EVITAR ARCHIVOS PESADOS

¿Qué es .gitignore?

Un archivo que le dice a Git qué archivos/carpetas NO subir a GitHub

Crear .gitignore específico para Node.js

```
bash
nano .gitignore
```

¿Qué hace **nano**? Editor de texto simple en terminal

Contenido a pegar en .gitignore:

```
# Dependencias de Node.js
node_modules/
npm-debug.log*
package-lock.json

# Archivos grandes
*.zip
*.rar
*.tar.gz
*.7z

# Videos e imágenes pesadas
*.mp4
*.avi
*.mov
*.mkv
*.iso

# Archivos temporales
*.tmp
*.temp
*~
.DS_Store
Thumbs.db

# Logs
logs/
*.log

# Archivos de entorno
.env
.env.local
```

Guardar en nano: **Ctrl + X** → **Y** → **Enter**

Agregar .gitignore al repositorio

```
bash
```

```
git add .gitignore
```

```
git commit -m "Agregar .gitignore para excluir archivos pesados"
```

PARTE 4: SUBIR TUS CARPETAS A GITHUB

Paso 1: Agregar archivos selectivamente (Recomendado)

```
bash
```

```
# Solo archivos de código
```

```
git add "*.js" "*.html" "*.css" "*.json" "*.md"
```

```
# O si quieres todo (después de crear .gitignore)
```

```
git add .
```

Paso 2: Verificar qué se va a subir

```
bash
```

```
git status
```

Lee cuidadosamente: Los archivos en verde son los que se subirán

Paso 3: Ver tamaño aproximado

```
bash
```

```
git diff --cached --stat
```

¿Qué hace? Muestra estadísticas de lo que vas a subir

Paso 4: Hacer commit

```
bash
```

```
git commit -m "Subir proyecto Node.js desde casa - $(date)"
```

El `$(date)` agrega la fecha automáticamente al mensaje

Paso 5: Subir a GitHub

```
bash
```

```
git push -u origin main
```

Si te pide credenciales:

- **Username:** `gustavomanfre`
 - **Password:** TU PERSONAL ACCESS TOKEN (no tu contraseña normal)
-



PARTE 5: DESCARGAR EN PC DEL TRABAJO

Paso 1: Verificar Git instalado

```
bash  
  
git --version
```

Si no está: `sudo apt install git`

Paso 2: Configurar Git (solo primera vez)

```
bash  
  
git config --global user.name "Gustavo Manfredi"  
git config --global user.email "tu-email@gmail.com"
```



Importante: Usa el MISMO email de GitHub

Paso 3: Clonar repositorio

```
bash  
  
cd ~/Documentos  
git clone https://github.com/gustavomanfre/repository-node.js.git
```

¿Qué hace `clone`? Descarga TODO el repositorio y crea la carpeta automáticamente

Paso 4: Verificar descarga

```
bash  
  
cd repository-node.js  
ls -la  
git status
```



PARTE 6: FLUJO DE TRABAJO DIARIO



EN CASA - Cuando hagas cambios:

bash

1. Ir a la carpeta del proyecto

`cd ~/Documentos/repositoy-node.js`

2. Ver qué cambió

`git status`

3. Agregar cambios

`git add .`

O específicos: git add archivo-modificado.js

4. Confirmar cambios

`git commit -m "Descripción clara de los cambios"`

5. Subir a GitHub

`git push`



EN EL TRABAJO - Para obtener cambios:

bash

1. Ir a la carpeta del proyecto

`cd ~/Documentos/repository-node.js`

2. Descargar últimos cambios

`git pull`

3. Verificar que todo está actualizado

`git status`



EN EL TRABAJO - Si haces cambios:

bash

1. Después de trabajar

`git add .`

`git commit -m "Cambios hechos en el trabajo"`

`git push`

2. Antes de irte, asegúrate que todo está subido

`git status`



PARTE 7: COMANDOS DE DIAGNÓSTICO Y SOLUCIÓN DE PROBLEMAS

Ver historial de cambios


```
bash

git log --oneline
```

¿Qué muestra? Lista de todos los commits con sus mensajes

Ver diferencias

```
bash

# Ver qué cambió antes de hacer commit
git diff

# Ver diferencias entre commits
git diff HEAD~1 HEAD
```

Deshacer cambios

```
bash

# Deshacer cambios no guardados
git checkout -- archivo.txt

# Quitar archivo del área de staging
git reset archivo.txt

# Volver al último commit (CUIDADO: borra cambios)
git reset --hard HEAD
```

Problemas de conexión

```
bash

# Ver configuración remota
git remote -v

# Cambiar URL remota si es necesario
git remote set-url origin https://github.com/gustavomanfre/repository-node.js.git
```

Verificar configuración

```
bash

git config --global --list
```



Tu secuencia exacta desde donde estás:

bash

Verificar ubicación (deberías estar aquí)

pwd

Resultado esperado: /home/gustavo/Documentos/repositoy-node.js

Ver estado actual

git status

Ver archivos

ls -la

Ver tamaño de la carpeta

du -sh .

Si es muy grande, crear .gitignore primero

nano .gitignore

(Pegar el contenido que te dí arriba)

Agregar .gitignore

git add .gitignore

git commit -m "Agregar .gitignore"

Agregar resto de archivos

git add .

Verificar qué se va a subir

git status

git diff --cached --stat

Hacer commit

git commit -m "Proyecto Node.js completo desde casa"

Verificar conexión remota

git remote -v

Si no está conectado:

git remote add origin https://github.com/gustavomanfre/repository-node.js.git

Subir a GitHub

git push -u origin main

PARTE 9: PROBLEMAS COMUNES Y SOLUCIONES

Error: "Authentication failed"

Solución:

- Username: `gustavomanfre`
- Password: Tu Personal Access Token (no tu contraseña de GitHub)

Error: "Repository not found"

Verificar:

```
bash  
  
git remote -v
```

Debe mostrar: `https://github.com/gustavomanfre/repository-node.js.git`

Error: "File too large"

Solución:

```
bash  
  
# Encontrar archivos grandes  
find . -size +100M  
  
# Agregar al .gitignore o usar Git LFS
```

Error: "Cannot push to non-bare repository"

Solución:

```
bash  
  
git push -f origin main
```

⚠ **CUIDADO:** `-f` fuerza la subida, úsalo solo si sabes lo que haces

Conflictos al hacer git pull

Solución:

bash

Ver archivos en conflicto

`git status`

Editar archivos manualmente para resolver conflictos

Buscar líneas con <<<<<<< y >>>>>>>

Después de resolver:

`git add` archivo-resuelto.txt

`git commit -m "Resolver conflicto en archivo-resuelto.txt"`

✓ PARTE 10: LISTA DE VERIFICACIÓN FINAL

Configuración inicial (solo una vez):

- ☐ Git instalado: `git --version`
- ☐ Usuario configurado: `git config --global user.name "Gustavo Manfredi"`
- ☐ Email configurado: `git config --global user.email "tu-email"`
- ☐ Personal Access Token creado en GitHub
- ☐ Repositorio creado en GitHub: `https://github.com/gustavomanfre/repository-node.js`

En casa (cada vez que hagas cambios):

- ☐ `cd ~/Documentos/repository-node.js`
- ☐ `git status` (ver qué cambió)
- ☐ `git add .` (agregar cambios)
- ☐ `git commit -m "mensaje descriptivo"`
- ☐ `git push` (subir a GitHub)

En el trabajo (primera vez):

- ☐ `cd ~/Documentos`
- ☐ `git clone https://github.com/gustavomanfre/repository-node.js.git`
- ☐ `cd repository-node.js`
- ☐ `ls -la` (verificar que todo se descargó)

En el trabajo (diariamente):

- ☐ `cd ~/Documentos/repository-node.js`
- ☐ `git pull` (descargar cambios de casa)
- ☐ Trabajar normalmente
- ☐ Si haces cambios: `git add .`, `git commit -m "..."`, `git push`



bash

NAVEGACIÓN

`cd ~/Documentos/repositoy-node.js` *# Ir a tu proyecto*

`pwd` *# ¿Dónde estoy?*

`ls -la` *# ¿Qué archivos hay?*

INFORMACIÓN

`git status` *# ¿Qué está pasando?*

`git log --oneline` *# ¿Qué cambios he hecho?*

`git remote -v` *# ¿A dónde está conectado?*

TRABAJO DIARIO

`git add .` *# Preparar cambios*

`git commit -m "mensaje"` *# Guardar cambios*

`git push` *# Subir a GitHub*

`git pull` *# Descargar de GitHub*

DIAGNÓSTICO






`du -sh .` *# ¿Qué tan grande es?*

`git diff` *# ¿Qué cambió?*

`git config --global --list` *# ¿Cómo estoy configurado?*

¡FELICIDADES!

Ahora tienes una comprensión completa de:

-  Qué hace cada comando de Git
-  Cómo conectar tu carpeta local con GitHub
-  Cómo mantener sincronizadas casa y trabajo
-  Cómo solucionar problemas comunes
-  Comandos específicos para tu situación

¡Practica estos comandos hasta que se vuelvan automáticos!