• ENTORNO DE DESARROLLO

Entorno de desarrollo en Windows con WSL2 + Ubuntu

De una instalación limpia de Windows al entorno completo: terminal moderna, shell configurada, editor conectado y agentes de IA en capas. Comando por comando.

01 Instalar WSL2 con Ubuntu

Abrí **PowerShell como administrador** (botón derecho en el menú Inicio → Terminal (Administrador)) y ejecutá:

```
wsl --install -d Ubuntu
```

Windows descarga e instala Ubuntu. **Reiniciá la PC cuando te lo pida.**

02 Primer arranque de Ubuntu

Después del reinicio, abrí **Ubuntu** desde el menú Inicio (aparece como una app más, con el ícono naranja). La primera vez tarda unos minutos en terminar de instalarse.

Te va a pedir crear un **usuario y contraseña de Linux**, independiente del de Windows. Anotá la contraseña: la vas a necesitar cada vez que uses `sudo`. Al escribirla no vas a ver asteriscos ni nada, es normal.

> Cuando aparezca el prompt `usuario@equipo:~$`, ya estás dentro de Ubuntu. Todos los comandos que siguen se escriben ahí, no en PowerShell.

03 Comandos base para moverte

Antes de instalar nada, estos son los comandos que vas a usar todo el tiempo:

`pwd`	mostrar en qué carpeta estás parado
`ls -la`	listar archivos, incluidos los ocultos
`cd carpeta`	entrar a una carpeta
`cd ..`	subir un nivel
`cd ~`	ir a tu carpeta personal (home)
`mkdir proyecto`	crear un directorio
`rm -rf carpeta`	borrar carpeta y contenido (no hay papelera)
`cat archivo`	ver el contenido de un archivo
`nano archivo`	editar un archivo en la terminal
`grep "texto" *`	buscar texto dentro de archivos
`sudo comando`	ejecutar como administrador

> **Importante:** guardá siempre tus proyectos dentro de `~/` (el home de Ubuntu), no en `/mnt/c/`. Cruzar el filesystem de Windows penaliza fuerte la velocidad de npm, pip y los watchers de hot reload.

04 Preparar Ubuntu

```
sudo apt update              # actualizar lista de paquetes
sudo apt upgrade -y          # actualizar lo instalado
sudo apt install -y curl git build-essential procps file
```

05 Terminal — Warp (recomendada)

Descargá Warp desde `warp.dev` e instalalo **en Windows** (no dentro de Ubuntu). Al abrirlo, elegí Ubuntu como shell por defecto desde el selector de perfiles.

Ventaja sobre la terminal común: agrupa cada comando y su salida en bloques separados (copiables y buscables), y trae IA nativa que entiende tu directorio, historial y errores. Alternativa gratuita sin IA: Windows Terminal, que ya viene instalado.

06 Zsh + Oh My Zsh

```
sudo apt install -y zsh
chsh -s $(which zsh)
```

Cerrá la terminal por completo y volvé a abrirla para que tome el cambio. Verificá con `echo $SHELL` — debe decir `/usr/bin/zsh`.

```
sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

07 Plugins de Zsh

```
git clone https://github.com/zsh-users/zsh-autosuggestions ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-autosuggestions
git clone https://github.com/zsh-users/zsh-syntax-highlighting.git ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-syntax-highlighting
```

Abrí la config con `nano ~/.zshrc`, buscá la línea `plugins=(git)` y dejala así:

```
plugins=(git zsh-autosuggestions zsh-syntax-highlighting)
```

Guardá con **Ctrl+O** → Enter, salí con **Ctrl+X**, y recargá con `source ~/.zshrc`

08 Powerlevel10k

```
git clone --depth=1 https://github.com/romkatv/powerlevel10k.git ${ZSH_CUSTOM:-$HOME/.oh-my-zsh/custom}/themes/powerlevel10k
sed -i 's/^ZSH_THEME=.*/ZSH_THEME="powerlevel10k\/powerlevel10k"/' ~/.zshrc
source ~/.zshrc
```

Se dispara un asistente visual. Si no aparece: `p10k configure`

> Instalá la fuente **MesloLGS NF** en Windows (no en Ubuntu) y seleccionala en la configuración de tu terminal, o los íconos del prompt se van a ver como cuadrados.

09 VS Code + extensión WSL

Instalá VS Code en Windows desde `code.visualstudio.com`. Después, dentro de VS Code, andá a Extensiones (Ctrl+Shift+X) y buscá **WSL** — la de Microsoft, con el ícono del pingüino. Instalala.

Ahora, desde tu terminal de Ubuntu, parado en cualquier proyecto:

```
code .          # abre VS Code conectado a WSL2
```

> Abajo a la izquierda de VS Code va a aparecer un indicador verde que dice **"WSL: Ubuntu"**. Eso confirma que está leyendo el filesystem de Linux. Su terminal integrada es la misma terminal de Ubuntu. Cursor y Antigravity, al ser forks de VS Code, funcionan igual con esta extensión.

10 Git y claves SSH

```
git config --global user.name "Tu Nombre"
git config --global user.email "tu@email.com"
ssh-keygen -t ed25519 -C "tu@email.com"     # Enter en todo
cat ~/.ssh/id_ed25519.pub                    # copiar y pegar en GitHub
```

Pegá esa clave en GitHub → Settings → SSH and GPG keys → New SSH key. Probá la conexión con `ssh -T git@github.com`

11 El circuito de trabajo

```
git clone git@github.com:usuario/proyecto.git
cd proyecto
code .
# ... editás ...
git status
git add .
git commit -m "feat: descripción del cambio"
git push
```

12 Homebrew

Gestor de paquetes con formulas versionadas y verificables — la base para instalar el resto de las herramientas.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
echo 'eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"' >> ~/.zshrc
eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"
brew --version
```

> Nunca uses `sudo brew install`. Homebrew se usa siempre como tu usuario normal.

13 Node.js y npm

```
sudo apt install -y nodejs npm
node --version && npm --version
```

14 Claude Code

```
curl -fsSL https://claude.ai/install.sh | bash
claude --version
claude doctor            # chequeo de salud completo
```

Para usarlo, entrá a cualquier proyecto y escribí `claude`. El agente arranca con el contexto de esa carpeta.

> No tengas dos instalaciones a la vez. Si ya instalaste por npm antes: `npm uninstall -g @anthropic-ai/claude-code`

15 OpenCode

Alternativa open source. Se le puede conectar Codex, Kimi u Ollama local como modelo, así el flujo no se frena si se acaban los tokens.

```
curl -fsSL https://opencode.ai/install | bash
opencode --version
```

16 Gentle AI + Engram + GGA

Ecosistema de Alan Buscaglia (Gentleman Programming). Engram da contexto persistente entre sesiones; GGA automatiza hooks de git.

```
brew tap Gentleman-Programming/homebrew-tap
brew trust --formula gentleman-programming/tap/gentle-ai
brew install gentleman-programming/tap/gentle-ai
brew trust --formula gentleman-programming/tap/engram
brew install gentleman-programming/tap/engram
brew trust --formula gentleman-programming/tap/gga
brew install gentleman-programming/tap/gga
```

17 Resolver el alias gga

El plugin `git` de Oh My Zsh define un alias `gga` que pisa el comando real:

```
unalias gga 2>/dev/null
```

Agregá esa misma línea **al final** de `~/.zshrc` para que no vuelva en cada terminal nueva.

```
gentle-ai install --dry-run     # simular primero
gentle-ai install
gentle-ai doctor                 # objetivo: Status: healthy
```

18 GGA en cada proyecto

```
cd ~/tu-proyecto
gga init
gga install
```

19 Ollama (opcional — IA local)

Modelos corriendo en tu propia máquina, sin costo de API ni dependencia de internet.

```
curl -fsSL https://ollama.com/install.sh | sh
ollama pull llama3
ollama run llama3
```

20 Verificación final

```
which claude opencode gentle-ai gga engram brew node npm zsh
claude doctor
gentle-ai doctor
```

· Diagnóstico rápido

"untrusted tap" en Homebrew: `brew trust --tap gentleman-programming/tap`

gga sigue siendo alias de git: confirmá con `type -a gga` y que `unalias gga` esté al final del .zshrc

code . no abre nada: falta la extensión WSL en VS Code (la del pingüino)

Íconos rotos en el prompt: falta MesloLGS NF instalada y seleccionada en la terminal de Windows

npm install muy lento: el proyecto está en /mnt/c/ — movelo a ~/

★ Recomendación final

No hace falta que sigas esta guía solo. Podés **pegarle este mismo documento a Claude** (o al agente que uses) y pedirle que te acompañe paso a paso: te va guiando comando por comando, y si algo falla, le pegás el error y te dice exactamente qué hacer.

Un prompt que funciona bien para arrancar:

```
"Te paso una guía de instalación de entorno de desarrollo.
Acompañame paso a paso, de a un comando por vez.
Esperá que te confirme que funcionó antes de pasar al siguiente.
Si te paso un error, ayudame a resolverlo antes de seguir."
```

Es la forma más rápida de montar todo esto sin frustrarte en el camino — y de paso, es exactamente el tipo de tarea para la que estos agentes son mejores.