• ENTORNO DE DESARROLLO

Entorno de desarrollo en Ubuntu nativo

Desde la ISO (o una máquina virtual) hasta el entorno completo con agentes de IA. El mismo stack que uso todos los días, comando por comando.

00 Opcional — probarlo primero en una VM

Si no querés instalar Ubuntu directo en tu disco, probalo aislado en VirtualBox. Todo lo que sigue funciona igual dentro de la VM.

1. Descargá VirtualBox desde `virtualbox.org` e instalalo.
2. Descargá la ISO de Ubuntu Desktop desde `ubuntu.com/download/desktop` (~5 GB).
3. En VirtualBox: botón *Nueva* → nombre "Ubuntu Dev", tipo Linux, versión Ubuntu (64-bit).
4. Asignale mínimo **4 GB de RAM** (8 GB si podés) y **25 GB de disco**.
5. Antes de arrancar: *Configuración* → *Almacenamiento* → click en el ícono del CD vacío → seleccioná la ISO.
6. *Iniciar* → seguí el instalador gráfico de Ubuntu.

> Después de instalar, apagá la VM, andá a Almacenamiento y quitá la ISO del lector virtual — si no, vuelve a arrancar el instalador cada vez.

01 Comandos base para moverte

`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

02 Preparar Ubuntu

```
sudo apt update
sudo apt upgrade -y
sudo apt install -y curl git build-essential procps file
```

Verificá que quedó todo con `git --version` y `gcc --version`

03 Zsh + Oh My Zsh

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

Cerrá la terminal por completo y volvé a abrirla. Verificá con `echo $SHELL`.

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

04 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
```

En `nano ~/.zshrc`, dejá la línea de plugins así:

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

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

05 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
```

Si no arranca el asistente solo: `p10k configure`. Instalá la Nerd Font **MesloLGS NF** y seleccionala en tu terminal.

06 Resolver el alias gga antes de tiempo

El plugin git de Oh My Zsh define un alias `gga` que después va a chocar con la herramienta real. Agregá esta línea **al final** de `~/.zshrc` desde ahora:

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

07 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"
cat ~/.ssh/id_ed25519.pub
```

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

08 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
```

09 VS Code

Instalá desde `code.visualstudio.com` (.deb) o vía snap:

```
sudo snap install code --classic
```

En Linux nativo no hace falta la extensión WSL — el filesystem ya es nativo. Desde cualquier proyecto: `code .`

> Cursor y Antigravity son forks de VS Code: se instalan aparte pero mantienen el mismo comportamiento y extensiones.

10 Homebrew (Linuxbrew)

```
sudo apt install -y build-essential procps curl file git
/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`.

11 Node.js, npm y Claude Code

```
sudo apt install -y nodejs npm
curl -fsSL https://claude.ai/install.sh | bash
claude --version && claude doctor
```

Para usarlo: entrá a un proyecto y escribí `claude`.

12 Go + OpenCode

```
brew install go
go version
brew install anomalyco/tap/opencode
opencode --version
```

A OpenCode se le puede conectar Codex, Kimi u Ollama local como modelo alternativo.

13 Gentle AI + Engram + GGA

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

```
brew trust --tap gentleman-programming/tap
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
type -a gga && unalias gga 2>/dev/null && source ~/.zshrc
```

14 Instalar y verificar Gentle AI

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

15 GGA en cada proyecto

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

16 Ollama (opcional — IA local)

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

17 Verificación final

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

· Diagnóstico rápido

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

gga sigue como alias: `type -a gga` y verificá el unalias al final del .zshrc

gentle-ai doctor sin agentes: corré `gentle-ai install`

Engram MCP no configurado: `gentle-ai sync` y volvé a correr doctor

OpenCode no toma el plugin: cerralo del todo y reabrilo

★ 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.