Si trabajamos en Windows y queremos utilizar asistentes de programación con inteligencia artificial directamente desde la terminal, podemos instalar herramientas como OpenCode, Claude Code, OpenAI Codex, Google Antigravity y Kimi Code. Estas aplicaciones no se limitan a responder preguntas. Dependiendo de los permisos que les concedamos, pueden **leer el código del proyecto, modificar archivos, ejecutar comandos, instalar dependencias, iniciar pruebas y acceder a otros recursos del sistema**.

Preparar Windows

Para realizar las instalaciones utilizaremos PowerShell. Algunas herramientas incluyen su propio instalador, mientras que otras se distribuyen mediante npm y necesitan que Node.js esté instalado. También instalaremos Git para Windows. Git resulta necesario para trabajar correctamente con repositorios y Kimi Code utiliza el entorno Git Bash incluido en este paquete.

Abrimos PowerShell como administrador y ejecutamos:

```
winget install --id Git.Git --exact --source winget
winget install --id OpenJS.NodeJS.LTS --exact --source winget
```

Cuando terminen las instalaciones, cerramos todas las ventanas de PowerShell y abrimos una nueva para que Windows vuelva a cargar la variable de entorno PATH. Comprobamos que Git, Node.js y npm estén disponibles:

```
git --version
node --version
npm.cmd --version
```

Utilizamos npm.cmd en lugar de npm para evitar que una política restrictiva de PowerShell bloquee la ejecución del archivo npm.ps1. Podemos aplicar el mismo criterio a las aplicaciones instaladas mediante npm si PowerShell intenta ejecutar sus correspondientes archivos con extensión ps1.

OpenCode

OpenCode es un agente de programación de código abierto que puede trabajar con diferentes proveedores y modelos de inteligencia artificial. Aunque su documentación recomienda utilizar WSL para obtener la máxima compatibilidad en Windows, también podemos instalarlo y ejecutarlo de forma nativa mediante npm.

Abrimos PowerShell como usuario normal y ejecutamos:

```
npm.cmd install -g opencode-ai
```

Después de instalarlo, nos situamos en la carpeta del proyecto y lo iniciamos:

```
Set-Location "C:\Ruta\Al\Proyecto"
opencode.cmd
```

Si nuestra configuración de PowerShell permite ejecutar el lanzador correspondiente, también podemos iniciarlo sin indicar la extensión:

```
opencode
```

Durante la primera ejecución podemos utilizar el comando interno de conexión para seleccionar el proveedor de inteligencia artificial y configurar su autenticación:

```
/connect
```

Una vez conectado el proveedor, podemos inicializar OpenCode dentro del proyecto:

```
/init
```

Este proceso analiza la estructura del proyecto y crea un archivo AGENTS.md con información que ayudará al agente a comprender el repositorio.

Claude Code

Claude Code dispone de un instalador nativo para Windows, por lo que ya no necesitamos instalarlo mediante npm. Podemos ejecutar el instalador desde un PowerShell abierto como usuario normal, sin permisos de administrador:

```
irm https://claude.ai/install.ps1 | iex
```

Cuando termine la instalación, comprobamos la versión instalada:

```
claude --version
```

También podemos ejecutar el diagnóstico incorporado para comprobar la instalación, la configuración y el estado de las actualizaciones:

```
claude doctor
```

Para trabajar con un proyecto, nos situamos en su carpeta y ejecutamos Claude Code:

```
Set-Location "C:\Ruta\Al\Proyecto"
claude
```

Durante el primer inicio completamos el proceso de autenticación mediante el navegador. Si tenemos Git para Windows instalado, Claude Code puede utilizar Git Bash para ejecutar comandos. Si no lo tenemos, utiliza PowerShell como intérprete de comandos. Las instalaciones realizadas mediante el instalador nativo comprueban e instalan automáticamente las actualizaciones disponibles. También podemos forzar manualmente la actualización:

```
claude update
```

OpenAI Codex CLI

Codex CLI es el agente de programación de OpenAI para trabajar desde la terminal. Actualmente dispone de un instalador independiente para Windows, por lo que no necesitamos instalar previamente Node.js ni utilizar npm exclusivamente para Codex. Abrimos PowerShell como usuario normal y ejecutamos el instalador oficial:

```
powershell -ExecutionPolicy Bypass -Command "irm https://chatgpt.com/codex/install.ps1 | iex"
```

Cerramos PowerShell, abrimos una nueva ventana y comprobamos la instalación:

```
codex --version
```

Para iniciar Codex dentro de un proyecto:

```
Set-Location "C:\Ruta\Al\Proyecto"
codex
```

Durante la primera ejecución podemos autenticarnos utilizando nuestra cuenta de ChatGPT o configurar otro método de autenticación compatible, como una clave de la API de OpenAI. Si preferimos instalar Codex mediante npm, también podemos hacerlo con:

```
npm.cmd install -g @openai/codex
```

En ese caso, si PowerShell bloquea el lanzador con extensión ps1, podemos iniciar la aplicación mediante:

```
codex.cmd
```

Google Antigravity CLI

Antigravity CLI es el agente de programación para terminal que Google utiliza como sucesor de Gemini CLI para las cuentas individuales. Se instala de forma nativa en Windows y su ejecutable utiliza el nombre agy. Abrimos PowerShell como usuario normal y ejecutamos:

```
irm https://antigravity.google/cli/install.ps1 | iex
```

El instalador descarga el ejecutable y lo añade al PATH de nuestro usuario. Cuando termine, cerramos PowerShell y abrimos una nueva ventana. Nos situamos en la carpeta del proyecto e iniciamos Antigravity:

```
Set-Location "C:\Ruta\Al\Proyecto"
agy
```

Durante el primer inicio se abrirá el navegador para completar la autenticación. Las credenciales de sesión se almacenan mediante el administrador de credenciales seguro de Windows. Si ya utilizábamos Gemini CLI, el instalador de Antigravity puede detectar su directorio de configuración local y migrar la configuración existente.

Kimi Code CLI

Kimi Code CLI dispone de un instalador específico para PowerShell. No debemos utilizar en Windows los comandos apt, bash o el instalador install.sh destinado a Linux. **Antes de iniciar Kimi Code debemos tener Git para Windows instalado**, ya que utiliza el entorno Git Bash incluido en este paquete para ejecutar comandos.

Abrimos PowerShell como usuario normal y ejecutamos:

```
irm https://code.kimi.com/kimi-code/install.ps1 | iex
```

Cuando termine la instalación, cerramos PowerShell y abrimos una nueva ventana. Después, nos situamos en el proyecto e iniciamos Kimi Code:

```
Set-Location "C:\Ruta\Al\Proyecto"
kimi
```

Durante el primer inicio utilizamos el siguiente comando interno para configurar la autenticación:

```
/login
```

Podemos autenticarnos mediante Kimi Code o utilizar una clave de la plataforma de Moonshot AI. Para actualizar posteriormente la aplicación podemos ejecutar:

```
kimi upgrade
```