■ GUÍA COMPLETA

Claude Code · VS Code · GitHub · Vercel · Supabase

Cómo subir tu proyecto de Claude Code a producción

GitHub | VS Code | Vercel | Supabase | Claude Code

Esta guía te lleva de cero al deploy: instala las herramientas, conecta tu repositorio y publica tu app con base de datos en la nube.

1

REQUISITOS E INSTALACIÓN

Todo lo que necesitas antes de empezar

¿Qué necesitas instalar?

Antes de conectar Claude Code con GitHub, Vercel y Supabase, asegúrate de tener las siguientes herramientas instaladas en tu computadora:

Node.js v18+	Runtime para ejecutar Claude Code y proyectos JS/TS	https://nodejs.org
Git	Control de versiones para sincronizar con GitHub	https://git-scm.com
Visual Studio Code	Editor de código recomendado	https://code.visualstudio.com
Claude Code CLI	La herramienta principal	npm install -g @anthropic-ai/claude-code

Paso 1 — Instalar Node.js

Descarga el instalador LTS de **nodejs.org** y sigue el asistente. Verifica la instalación abriendo una terminal:

```
node --version # Debe mostrar v18.x.x o superior
npm --version # Debe mostrar 9.x.x o superior
```

Paso 2 — Instalar Git

En **Windows**: descarga Git for Windows desde git-scm.com.
En **Mac**: ejecuta `xcode-select --install` o instala con Homebrew.
En **Linux**: usa tu gestor de paquetes.

```
git --version # Verifica la instalación
```

Configura tu identidad (solo la primera vez):

```
git config --global user.name "Tu Nombre"
git config --global user.email "tu@email.com"
```

Paso 3 — Instalar Visual Studio Code

Descarga VS Code desde **code.visualstudio.com**. Durante la instalación en Windows, marca la opción *'Add to PATH'* para poder abrirlo desde la terminal.

Extensiones recomendadas para VS Code:

- **GitHub Pull Requests** — Gestión de pull requests desde VS Code
- **GitLens** — Visualización avanzada del historial Git
- **Supabase** — Autocompletado y snippets para Supabase
- **ESLint / Prettier** — Formato y calidad de código
- **Thunder Client** — Cliente HTTP para probar APIs

Paso 4 — Instalar Claude Code

Claude Code se instala como paquete global de npm. Abre una terminal y ejecuta:

```
npm install -g @anthropic-ai/claude-code
```

Luego autentica Claude Code con tu API key de Anthropic:

```
claude # Inicia la configuración interactiva
```

> ■ **Importante**
> Necesitas una API Key de Anthropic. Créala en console.anthropic.com → API Keys.

Verifica que todo esté funcionando abriendo VS Code desde la terminal:

```
code . # Abre el directorio actual en VS Code
```

2 CONECTAR VS CODE A GITHUB

Autenticación, repositorio y primer push

Paso 1 — Crear una cuenta en GitHub

Si aún no tienes cuenta, ve a **github.com** y regístrate gratis. Con la cuenta gratuita tienes repositorios privados y públicos ilimitados.

Paso 2 — Autenticar VS Code con GitHub

VS Code tiene integración nativa con GitHub. Sigue estos pasos:

1	**Abre VS Code y presiona:** `Ctrl+Shift+P (Windows/Linux) o Cmd+Shift+P (Mac)`
2	**Escribe y selecciona:** `GitHub: Sign in`
3	**Se abrirá el navegador:** `Autoriza VS Code haciendo clic en 'Authorize Visual-Studio-Code'`
4	**Regresa a VS Code:** `La autenticación se completará automáticamente`

Paso 3 — Crear el repositorio para tu proyecto

Tienes dos opciones: crear el repo desde GitHub.com o directamente desde VS Code.

Opción A — Desde GitHub.com (recomendado):

- Ve a **github.com/new**
- Pon un nombre al repositorio (ej: `mi-proyecto-claude`)
- Elige **Privado** o **Público** según prefieras
- NO marques "Initialize with README" si ya tienes archivos locales
- Haz clic en **Create repository**

Paso 4 — Conectar tu proyecto local con GitHub

Abre la terminal en VS Code (**Ctrl+**`) y navega a la carpeta de tu proyecto:

```
cd ruta/a/tu/proyecto
```

Inicializa Git y conecta con el repositorio remoto:

Descripción	Comando
Inicializa Git en el proyecto	`git init`
Agrega todos los archivos	`git add .`
Crea el primer commit	`git commit -m "Primer commit desde Claude Code"`
Renombra la rama principal	`git branch -M main`

Conecta con GitHub (reemplaza URL)	`git remote add origin https://github.com/TU_USUARIO/mi-proyecto-claude.git`
Sube el código a GitHub	`git push -u origin main`

Paso 5 — Flujo diario con Git en VS Code

Una vez conectado, puedes usar la interfaz gráfica de VS Code (panel de Control de Código Fuente, ícono de rama en la barra lateral izquierda) o la terminal:

Descripción	Comando
Ver estado de cambios	`git status`
Agregar cambios al stage	`git add .`
Hacer commit	`git commit -m "Descripción del cambio"`
Subir a GitHub	`git push`
Bajar cambios remotos	`git pull`

> ■ **Tip de flujo**
> En VS Code, haz clic en el ícono de la rama (barra lateral) para ver cambios, hacer commits y push con clics, sin usar la terminal.

3 CONFIGURAR SUPABASE

Base de datos PostgreSQL en la nube — gratis

¿Qué es Supabase?

Supabase es una alternativa open-source a Firebase que ofrece base de datos PostgreSQL, autenticación, almacenamiento de archivos y APIs en tiempo real. El plan gratuito incluye 500 MB de base de datos, suficiente para la mayoría de proyectos.

Paso 1 — Crear proyecto en Supabase

- Ve a **supabase.com** y crea una cuenta gratuita
- Haz clic en **New Project**
- Elige un nombre para tu proyecto y una región cercana (ej: South America)
- Crea una contraseña segura para la base de datos (guárdala, la necesitarás)
- Espera 2-3 minutos mientras Supabase configura la base de datos

Paso 2 — Obtener las credenciales de conexión

Una vez creado el proyecto, ve a **Settings** → **API** para obtener:

Credencial	Descripción	Variable de entorno
Project URL	La URL base de tu proyecto Supabase	`NEXT_PUBLIC_SUPABASE_URL`
anon public	Clave pública para el cliente (seguro usarla en frontend)	`NEXT_PUBLIC_SUPABASE_ANON_KEY`
service_role	Clave privada para operaciones del servidor	`SUPABASE_SERVICE_ROLE_KEY`

> ■ **¡Nunca compartas service_role!**
>
> La clave service_role tiene acceso total. Úsala solo en el servidor, nunca en código del frontend ni en repositorios públicos.

Paso 3 — Instalar el cliente de Supabase en tu proyecto

En la terminal de VS Code, dentro de tu proyecto, instala el paquete oficial:

```
npm install @supabase/supabase-js
```

Crea el cliente de Supabase en tu código:

Crea el archivo `lib/supabase.js`:

```
import { createClient } from '@supabase/supabase-js'

const supabaseUrl = process.env.NEXT_PUBLIC_SUPABASE_URL
const supabaseKey = process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY

export const supabase = createClient(supabaseUrl, supabaseKey)
```

Paso 4 — Crear tablas en Supabase

Ve a **Table Editor** en el panel de Supabase para crear tablas con interfaz gráfica, o usa el **SQL Editor** para mayor control:

```
-- Ejemplo: tabla de usuarios
CREATE TABLE profiles (
id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
email TEXT UNIQUE NOT NULL,
name TEXT,
created_at TIMESTAMPTZ DEFAULT NOW()
);
```

4

VARIABLES DE ENTORNO

Conexión segura entre tu app, Supabase y Vercel

Paso 1 — Crear el archivo .env.local

En la raíz de tu proyecto (junto a package.json), crea el archivo `.env.local` con las credenciales de Supabase:

```
# .env.local — NO subir a GitHub
NEXT_PUBLIC_SUPABASE_URL=https://tuproyecto.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJhbGciOiJIUzI1NiIs...
SUPABASE_SERVICE_ROLE_KEY=eyJhbGciOiJIUzI1NiIs...
```

Paso 2 — Agregar .env.local al .gitignore

Es CRÍTICO que este archivo nunca se suba a GitHub. Abre o crea el archivo `.gitignore` y asegúrate que contenga:

```
# Entornos
.env
.env.local
.env.*.local

# Dependencias
node_modules/
.next/
```

✓ **Verificación rápida**

Ejecuta 'git status' y asegúrate de que .env.local NO aparezca en la lista de archivos a subir.

5 DEPLOY EN VERCEL

CI/CD automático desde GitHub — sin configuración compleja

¿Por qué Vercel?

Vercel es la plataforma de deploy más simple para proyectos Next.js, React y Vite. Cada push a GitHub despliega automáticamente tu app. El plan gratuito incluye dominio .vercel.app, SSL y deployments ilimitados.

Paso 1 — Crear cuenta en Vercel

- Ve a **vercel.com** y haz clic en **Sign Up**
- Selecciona **Continue with GitHub** para vincular ambas cuentas
- Autoriza a Vercel a acceder a tus repositorios

Paso 2 — Importar tu proyecto de GitHub

- En el dashboard de Vercel, haz clic en **Add New** → **Project**
- Busca y selecciona tu repositorio de GitHub
- Vercel detecta automáticamente el framework (Next.js, Vite, etc.)
- Revisa la configuración de build — generalmente no necesita cambios

Paso 3 — Configurar las variables de entorno en Vercel

Antes de hacer el primer deploy, agrega las variables de entorno de Supabase en el panel de Vercel (no las del .env.local, ese es solo local):

Variable	Dónde obtenerla	Entorno
`NEXT_PUBLIC_SUPABASE_URL`	Supabase → Settings → API → Project URL	Production, Preview, Development
`NEXT_PUBLIC_SUPABASE_ANON_KEY`	Supabase → Settings → API → anon public	Production, Preview, Development
`SUPABASE_SERVICE_ROLE_KEY`	Supabase → Settings → API → service_role secret	Production

Paso 4 — Hacer el primer deploy

- Haz clic en **Deploy** en Vercel
- Espera 1-3 minutos mientras construye y despliega tu app
- Recibirás una URL como: `tu-proyecto.vercel.app`
- Cada nuevo `git push` a main desplegará automáticamente

> ■ **Deploy automático activado**
>
> A partir de ahora, cada 'git push origin main' activará un nuevo deploy. Vercel también crea previews para cada Pull Request de GitHub.

Paso 5 — Configurar dominio personalizado (opcional)

Si tienes un dominio propio, en Vercel ve a **Settings** → **Domains** y agrega tu dominio. Vercel te dará instrucciones para configurar los registros DNS.

6 FLUJO DE TRABAJO COMPLETO

Del código a producción en minutos

El ciclo completo de desarrollo

Una vez configurado todo, este es el flujo que seguirás cada vez que quieras agregar funcionalidades o hacer cambios en tu proyecto:

1 Trabajar con Claude Code

```
claude
→ Pide a Claude que agregue funcionalidades
→ Claude modifica los archivos directamente
```

2 Probar localmente

```
npm run dev
→ Abre http://localhost:3000
→ Verifica que todo funciona
```

3 Subir a GitHub

```
git add .
git commit -m "Nueva funcionalidad"
git push
```

4 Vercel despliega automáticamente

```
→ Vercel detecta el push
→ Construye y despliega en ~2 min
→ App actualizada en producción
```

Comandos de referencia rápida

Descripción	Comando
Iniciar Claude Code	`claude`
Desarrollar localmente	`npm run dev`
Build de producción	`npm run build`
Ver el log del último deploy Vercel	`vercel logs`
Ver el estado de Git	`git status`
Deshacer el último commit (local)	`git reset --soft HEAD~1`
Ver historial de commits	`git log --oneline`

7 SOLUCIÓN DE ERRORES COMUNES

Los problemas más frecuentes y cómo resolverlos

Error: 'git push' rechazado

Si ves 'rejected — Updates were rejected because the remote contains work'

- ■ Ejecuta `git pull origin main --rebase` primero
- ■ Resuelve conflictos si los hay
- ■ Luego ejecuta `git push` de nuevo

Variables de entorno no encontradas en Vercel

Tu app funciona local pero falla en producción (undefined URL o Key)

- ■ Ve a Vercel → tu proyecto → Settings → Environment Variables
- ■ Verifica que las variables estén configuradas para 'Production'
- ■ Haz un nuevo deploy: Vercel → Deployments → Redeploy

Error de CORS con Supabase

'Access to fetch blocked by CORS policy'

- ■ Ve a Supabase → Settings → API → CORS Settings
- ■ Agrega tu dominio de Vercel: `https://tu-proyecto.vercel.app`
- ■ Agrega también `http://localhost:3000` para desarrollo

Build falla en Vercel

El deploy falla con error de TypeScript o módulo no encontrado

- ■ Prueba el build localmente: `npm run build`
- ■ Lee el error completo en Vercel → Deployments → ver logs
- ■ Arregla el error localmente y vuelve a hacer push

Claude Code no responde o da error de API

'Authentication error' o 'API key invalid'

- ■ Verifica tu API key en `console.anthropic.com`
- ■ Reconfigura ejecutando `claude` y siguiendo las instrucciones
- ■ Verifica que tengas créditos disponibles en tu cuenta Anthropic

---

¡Tu proyecto está en producción! ■

Siguiendo esta guía has configurado un flujo profesional completo: Claude Code genera el código, Git lo versiona, GitHub lo almacena, Supabase provee la base de datos y Vercel lo despliega automáticamente. Este es exactamente el stack que usan equipos profesionales de desarrollo.

■	Node.js, Git, VS Code y Claude Code instalados
■	VS Code autenticado con GitHub
■	Repositorio creado y primer push realizado

■	Proyecto en Supabase con credenciales guardadas
■	Variables de entorno en .env.local y en Vercel
■	Deploy automático activo en Vercel
■	URL de producción funcionando