# Dropship-Fast — Full Project Files

Ce document contient **tous les fichiers** prêts à coller dans un repo GitHub et déployer sur Vercel. Copie chaque fichier dans l’arborescence indiquée sur ton ordinateur.

IMPORTANT : les clés secrètes doivent être renseignées dans .env.local (ne jamais les committer).

## Arborescence recommandée

dropship-fast/  
├─ package.json  
├─ next.config.js  
├─ .env.example  
├─ README.md  
├─ public/  
│ └─ sample-images/  
│ ├─ watch.jpg  
│ ├─ backpack.jpg  
│ └─ lamp.jpg  
├─ styles/  
│ └─ globals.css  
├─ pages/  
│ ├─ \_app.jsx  
│ ├─ index.jsx  
│ ├─ cart.jsx  
│ ├─ product/[id].jsx  
│ └─ api/  
│ ├─ suppliers.js  
│ ├─ order.js  
│ └─ webhooks/  
│ └─ stripe.js  
├─ components/  
│ └─ ProductCard.jsx  
└─ lib/  
 ├─ shipping.js  
 ├─ cj.js  
 ├─ stripe.js  
 └─ easypost.js

### package.json

{  
 "name": "dropship-fast",  
 "version": "1.0.0",  
 "private": true,  
 "scripts": {  
 "dev": "next dev -p 3000",  
 "build": "next build",  
 "start": "next start"  
 },  
 "dependencies": {  
 "axios": "^1.4.0",  
 "next": "13.5.0",  
 "react": "18.2.0",  
 "react-dom": "18.2.0",  
 "swr": "^2.2.0",  
 "stripe": "^12.11.0"  
 }  
}

### next.config.js

/\*\* @type {import('next').NextConfig} \*/  
const nextConfig = { reactStrictMode: true }  
module.exports = nextConfig

### .env.example

# Fill these into .env.local for development or in Vercel env vars for production  
NEXT\_PUBLIC\_BASE\_URL=http://localhost:3000  
  
# Stripe  
STRIPE\_SECRET\_KEY=sk\_test\_xxx  
STRIPE\_WEBHOOK\_SECRET=whsec\_xxx  
  
# CJdropshipping (demo)  
CJ\_CLIENT\_ID=your\_cj\_client\_id  
CJ\_CLIENT\_SECRET=your\_cj\_client\_secret  
CJ\_BASE\_URL=https://developers.cjdropshipping.com/api  
  
# EasyPost  
EASYPOST\_API\_KEY=epk\_test\_xxx  
  
# Email (SendGrid / Resend)  
EMAIL\_API\_KEY=your\_email\_api\_key  
EMAIL\_FROM=contact@yoursite.com  
  
# Business  
BUSINESS\_NAME=Ton Entreprise  
BUSINESS\_VAT\_NUMBER=FRXX999999999  
  
# Security  
JWT\_SECRET=some\_random\_secret

### README.md

(Contenu abrégé — voir le README complet dans le document créé précédemment)

# Dropship-Fast  
Demo Next.js dropshipping starter optimisé pour livraison rapide en France.  
  
## Installation  
1. Copier les fichiers  
2. `npm install`  
3. `cp .env.example .env.local` puis remplir les clés  
4. `npm run dev`  
  
## Déploiement  
- Pousser sur GitHub  
- Importer le repo dans Vercel  
- Configurer les variables d'environnement dans Vercel  
- Déployer

### styles/globals.css

html,body,#\_\_next{height:100%}  
body{font-family:Inter, ui-sans-serif, system-ui, -apple-system, 'Segoe UI', Roboto, 'Helvetica Neue', Arial; margin:0; background:#f7fafc}  
img{max-width:100%;}  
.container{max-width:1100px;margin:0 auto;padding:20px}

### pages/\_app.jsx

import '../styles/globals.css'  
export default function App({ Component, pageProps }) {  
 return <Component {...pageProps} />  
}

### pages/index.jsx

import useSWR from 'swr'  
import axios from 'axios'  
import ProductCard from '../components/ProductCard'  
  
const fetcher = (url) => axios.get(url).then(r => r.data)  
  
export default function Home() {  
 const { data, error } = useSWR('/api/suppliers', fetcher)  
 if (error) return <div className="container">Erreur de chargement</div>  
 if (!data) return <div className="container">Chargement...</div>  
  
 return (  
 <div className="container">  
 <h1>Produits — Dropship‑Fast (démo)</h1>  
 <div style={{display:'grid',gridTemplateColumns:'repeat(auto-fit,minmax(260px,1fr))',gap:16,marginTop:16}}>  
 {data.products.map(p => <ProductCard key={p.id} product={p} />)}  
 </div>  
 </div>  
 )  
}

### components/ProductCard.jsx

import Link from 'next/link'  
export default function ProductCard({ product }) {  
 return (  
 <article style={{background:'#fff',padding:12,borderRadius:8,boxShadow:'0 1px 4px rgba(0,0,0,0.08)'}}>  
 <img src={product.img} alt={product.title} style={{height:190,objectFit:'cover',borderRadius:6}} />  
 <h3 style={{marginTop:10}}>{product.title}</h3>  
 <p style={{color:'#555'}}>{product.price} €</p>  
 <div style={{marginTop:8}}>  
 <Link href={`/product/${product.id}`}><a style={{textDecoration:'underline'}}>Voir le produit</a></Link>  
 </div>  
 </article>  
 )  
}

### pages/product/[id].jsx

import { useRouter } from 'next/router'  
import useSWR from 'swr'  
import axios from 'axios'  
  
const fetcher = (url) => axios.get(url).then(r => r.data)  
  
export default function ProductPage() {  
 const router = useRouter()  
 const { id } = router.query  
 const { data } = useSWR('/api/suppliers', fetcher)  
 if (!data) return <div className="container">Chargement...</div>  
 const product = data.products.find(p => p.id === id)  
 if (!product) return <div className="container">Produit introuvable</div>  
  
 return (  
 <div className="container">  
 <img src={product.img} alt={product.title} style={{width:'100%',height:360,objectFit:'cover',borderRadius:8}} />  
 <h1 style={{marginTop:12}}>{product.title}</h1>  
 <p style={{color:'#333'}}>{product.price} €</p>  
 <h3>Fournisseurs (priorité EU pour livraison rapide)</h3>  
 <ul>  
 {product.suppliers.map(s => (  
 <li key={s.id}><a href={s.link} target="\_blank" rel="noreferrer">{s.name}</a> — {s.country} — délai indicatif: {s.leadDays}j</li>  
 ))}  
 </ul>  
 <div style={{marginTop:12}}>  
 <button onClick={() => alert('Simulation: ajout au panier')} style={{padding:'10px 16px',background:'#0066ff',color:'#fff',border:'none',borderRadius:6}}>Ajouter au panier</button>  
 </div>  
 </div>  
 )  
}

### pages/cart.jsx

import { useState } from 'react'  
  
export default function Cart() {  
 const [customer, setCustomer] = useState({ name: '', email: '', country: 'FR', address: '' })  
 const [loading, setLoading] = useState(false)  
  
 async function placeOrder() {  
 setLoading(true)  
 const cart = JSON.parse(localStorage.getItem('cart') || '[]')  
 const res = await fetch('/api/order', { method: 'POST', headers: {'Content-Type':'application/json'}, body: JSON.stringify({ cart, customer }) })  
 const data = await res.json()  
 alert('Commande simulée: ' + JSON.stringify(data, null, 2))  
 setLoading(false)  
 }  
  
 return (  
 <div className="container">  
 <h1>Panier (démo)</h1>  
 <div>  
 <label>Nom<input value={customer.name} onChange={e => setCustomer({...customer,name:e.target.value})} /></label>  
 <label>Email<input value={customer.email} onChange={e => setCustomer({...customer,email:e.target.value})} /></label>  
 <label>Pays<select value={customer.country} onChange={e => setCustomer({...customer,country:e.target.value})}><option value="FR">France</option><option value="DE">Allemagne</option></select></label>  
 <div style={{marginTop:12}}>  
 <button onClick={placeOrder} disabled={loading} style={{padding:'10px 14px',background:'#16a34a',color:'#fff',border:'none',borderRadius:6}}>{loading ? 'Traitement...' : 'Passer commande (simulation)'}</button>  
 </div>  
 </div>  
 </div>  
 )  
}

### pages/api/suppliers.js

export default function handler(req, res) {  
 const products = [  
 {  
 id: 'p-watch-01',  
 title: 'Montre minimaliste',  
 price: 49.9,  
 img: '/sample-images/watch.jpg',  
 suppliers: [  
 { id: 'bigbuy-001', name: 'BigBuy (ES)', country: 'ES', leadDays: 1, price: 28, stock: 120, link: 'https://www.bigbuy.eu' },  
 { id: 'cj-eu-01', name: 'CJdropshipping (FR warehouse)', country: 'FR', leadDays: 2, price: 25, stock: 300, link: 'https://fr.cjdropshipping.com' },  
 { id: 'cn-std', name: 'CN Supplier', country: 'CN', leadDays: 8, price: 12, stock: 1000, link: 'https://www.aliexpress.com' },  
 ]  
 },  
 {  
 id: 'p-backpack-01',  
 title: 'Sac à dos urbain',  
 price: 69.9,  
 img: '/sample-images/backpack.jpg',  
 suppliers: [  
 { id: 'bb-002', name: 'BigBuy (ES)', country: 'ES', leadDays: 1, price: 40, stock: 60, link: 'https://www.bigbuy.eu' },  
 { id: 'spocket-eu-01', name: 'Spocket EU (DE)', country: 'DE', leadDays: 3, price: 42, stock: 30, link: 'https://www.spocket.co' },  
 { id: 'cn-std2', name: 'CN Supplier', country: 'CN', leadDays: 10, price: 22, stock: 800, link: 'https://www.aliexpress.com' },  
 ]  
 },  
 {  
 id: 'p-lamp-01',  
 title: 'Lampe LED tactile',  
 price: 34.9,  
 img: '/sample-images/lamp.jpg',  
 suppliers: [  
 { id: 'printful-eu', name: 'Printful EU', country: 'LV', leadDays: 2, price: 20, stock: 50, link: 'https://www.printful.com' },  
 { id: 'cj-cn', name: 'CJ CN', country: 'CN', leadDays: 8, price: 10, stock: 1000, link: 'https://www.cjdropshipping.com' },  
 ]  
 }  
 ]  
 res.status(200).json({ products })  
}

### lib/shipping.js

export function estimateDeliveryDays(supplier, destCountry = 'FR') {  
 if (supplier.country === destCountry) return supplier.leadDays + 1  
 const eu = ['FR','DE','ES','IT','NL','BE','LU','PT','LV']  
 if (eu.includes(supplier.country) && eu.includes(destCountry)) return supplier.leadDays + 2  
 return supplier.leadDays + 7  
}  
  
export function pickBestSupplier(productSuppliers, destCountry='FR', preferFast=true) {  
 const available = productSuppliers.filter(s => s.stock > 0)  
 if (available.length === 0) return null  
 const scored = available.map(s => ({ s, eta: estimateDeliveryDays(s, destCountry), cost: s.price }))  
 if (preferFast) scored.sort((a,b) => a.eta - b.eta || a.cost - b.cost)  
 else scored.sort((a,b) => a.cost - b.cost || a.eta - b.eta)  
 return scored[0]  
}

### lib/cj.js

import axios from 'axios'  
const BASE = process.env.CJ\_BASE\_URL || 'https://developers.cjdropshipping.com/api'  
const CLIENT\_ID = process.env.CJ\_CLIENT\_ID  
const CLIENT\_SECRET = process.env.CJ\_CLIENT\_SECRET  
let tokenCache = { token: null, expiresAt: 0 }  
  
async function getToken() {  
 const now = Date.now()  
 if (tokenCache.token && tokenCache.expiresAt > now + 60000) return tokenCache.token  
 const resp = await axios.post(`${BASE}/oauth/token`, { client\_id: CLIENT\_ID, client\_secret: CLIENT\_SECRET, grant\_type: 'client\_credentials' })  
 tokenCache.token = resp.data.access\_token  
 tokenCache.expiresAt = now + (resp.data.expires\_in || 3600) \* 1000  
 return tokenCache.token  
}  
  
export async function cjCreateOrder(payload) {  
 const token = await getToken()  
 const r = await axios.post(`${BASE}/orders/createOrder`, payload, { headers: { Authorization: `Bearer ${token}` } })  
 return r.data  
}  
  
export async function cjGetProduct(sku) {  
 const token = await getToken()  
 const r = await axios.get(`${BASE}/products/${sku}`, { headers: { Authorization: `Bearer ${token}` } })  
 return r.data  
}

### lib/stripe.js

import Stripe from 'stripe'  
const stripe = new Stripe(process.env.STRIPE\_SECRET\_KEY)  
  
export async function createStripeSession({ line\_items, success\_url, cancel\_url, metadata }) {  
 const session = await stripe.checkout.sessions.create({ payment\_method\_types: ['card'], line\_items, mode: 'payment', success\_url, cancel\_url, metadata, locale: 'fr' })  
 return session  
}  
  
export async function verifyStripeWebhookSignature(req) {  
 const sig = req.headers['stripe-signature']  
 const buf = await buffer(req)  
 try {  
 const event = stripe.webhooks.constructEvent(buf, sig, process.env.STRIPE\_WEBHOOK\_SECRET)  
 return { ok: true, event }  
 } catch (err) {  
 return { ok: false, error: err }  
 }  
}  
  
async function buffer(req) {  
 return new Promise((resolve, reject) => {  
 const chunks = []  
 req.on('data', (c) => chunks.push(c))  
 req.on('end', () => resolve(Buffer.concat(chunks)))  
 req.on('error', reject)  
 })  
}

### lib/easypost.js

import axios from 'axios'  
const BASE = 'https://api.easypost.com/v2'  
const KEY = process.env.EASYPOST\_API\_KEY  
  
export async function createShipmentAndBuyLabel({ to\_address, from\_address, parcel }) {  
 const auth = { username: KEY, password: '' }  
 const shipResp = await axios.post(`${BASE}/shipments`, { shipment: { to\_address, from\_address, parcel } }, { auth })  
 const shipment = shipResp.data  
 const rates = shipment.rates || []  
 if (rates.length === 0) return shipment  
 const chosen = rates.sort((a,b) => parseFloat(a.rate) - parseFloat(b.rate))[0]  
 const buyResp = await axios.post(`${BASE}/shipments/${shipment.id}/buy`, { rate: chosen }, { auth })  
 return buyResp.data  
}

### pages/api/order.js

import axios from 'axios'  
import { createStripeSession } from '../../lib/stripe'  
import { pickBestSupplier } from '../../lib/shipping'  
  
export default async function handler(req, res) {  
 if (req.method !== 'POST') return res.status(405).end()  
 const { cart, customer } = req.body  
 if (!cart || cart.length === 0) return res.status(400).json({ error: 'cart empty' })  
  
 const { data } = await axios.get(`${process.env.NEXT\_PUBLIC\_BASE\_URL || ''}/api/suppliers`)  
 const products = data.products  
  
 const assignments = cart.map(item => {  
 const prod = products.find(p => p.id === item.id)  
 const best = pickBestSupplier(prod.suppliers, customer.country || 'FR', true)  
 return { item, product: prod, chosen: best }  
 })  
  
 const internalOrder = { id: 'INT-' + Date.now(), customer, assignments, status: 'pending\_payment' }  
  
 const line\_items = cart.map(it => ({ price\_data: { currency: 'eur', product\_data: { name: it.title }, unit\_amount: Math.round(it.price \* 100) }, quantity: it.qty }))  
 const session = await createStripeSession({ line\_items, success\_url: `${process.env.NEXT\_PUBLIC\_BASE\_URL}/success?session\_id={CHECKOUT\_SESSION\_ID}`, cancel\_url: `${process.env.NEXT\_PUBLIC\_BASE\_URL}/cart`, metadata: { internalOrderId: internalOrder.id } })  
  
 // TODO: save internalOrder + session.id in DB  
  
 return res.status(200).json({ checkoutUrl: session.url, internalOrder })  
}

### pages/api/webhooks/stripe.js

import { verifyStripeWebhookSignature } from '../../../lib/stripe'  
  
export const config = { api: { bodyParser: false } }  
  
export default async function handler(req, res) {  
 const { ok, event, error } = await verifyStripeWebhookSignature(req)  
 if (!ok) return res.status(400).send(`Webhook Error: ${error.message}`)  
  
 if (event.type === 'checkout.session.completed') {  
 const session = event.data.object  
 const internalOrderId = session.metadata?.internalOrderId  
 console.log('Payment received for internalOrder', internalOrderId)  
 // TODO: 1) Mark order paid in DB  
 // 2) For each assignment, call supplier API (CJ / BigBuy / Printful) to create supplier order  
 // 3) If supplier provides tracking, store and email customer  
 }  
  
 res.status(200).end()  
}

### public/sample-images

Place any three images named watch.jpg, backpack.jpg, lamp.jpg in /public/sample-images/ or modifiez les URLs dans pages/api/suppliers.js pour pointer vers des images externes.

## Instructions pour créer le repo GitHub & déployer

1. git init → git add . → git commit -m "Initial commit"
2. Crée un repo sur GitHub et pousse :

git remote add origin https://github.com/TonUser/dropship-fast.git  
git branch -M main  
git push -u origin main

1. Sur Vercel : *New Project* → Import GitHub repo → Configure env vars (colle le contenu de .env.example dans Vercel) → Deploy.
2. Pour Stripe Webhooks : configure l’URL https://<ton-domaine>/api/webhooks/stripe et coller STRIPE\_WEBHOOK\_SECRET dans Vercel.

## Tests locaux utiles

* npm run dev puis visiter http://localhost:3000.
* Dans /cart, tester Passer commande (simulation) pour recevoir checkoutUrl (Stripe doit être configuré pour que la session soit créée).
* Utilise Stripe CLI pour forwarder webhooks en local si tu veux tester webhooks.

Si tu veux, je peux maintenant : 1. **Générer un ZIP** du projet (je fournis ici les contenus ; tu peux copier-coller).  
2. **Te fournir les commandes exactes** pour créer le repo GitHub (personnalisé avec ton pseudo).  
3. **Configurer pour toi** (je ne peux pas pousser directement vers ton GitHub sans tes accès) — je peux te dire pas à pas comment faire ou te fournir un script gh pour automatiser.

Que souhaites‑tu que je fasse **tout de suite** ?