Overview

A polished Next.js app on your VPS

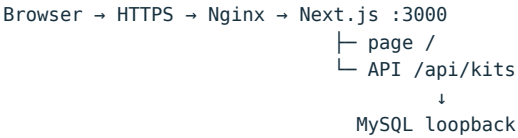
Build the interface, connect MySQL and deploy with HTTPS.
One Next.js server handles pages and API routes.

CodeKits • Original catalogue template + free Magic UI component

Architecture

Next.js joins UI and server code

React components describe the interface. Next.js adds routing, server rendering and server endpoints.



Browser-side React still runs on the user’s device; it is not inside the server process.

Choose a template

Free components versus paid templates

This kit uses an original CodeKits layout and the free Border Beam component. Magic UI Pro templates are separate products.

```
Start from the included project
OR create a fresh Next.js App Router project
Use TypeScript and Tailwind CSS
Keep the template’s licence and attribution
Review dependencies before installing components
```

Do not redistribute a purchased template unless its licence permits it.

Project setup

Install and inspect the project

Use Node.js 24 and the included lockfile. Understand the file structure before changing it.

```
npm ci

app/page.tsx       → server-rendered catalogue
app/api/kits/route.ts → public JSON endpoint
components/hero.tsx → interactive decoration
components/border-beam.tsx → Magic UI
lib/db.ts          → server-only MySQL queries
```

The kit has a root layout, metadata, error states and responsive styling.

Magic UI

Add a component from the registry

For a fresh project, follow Magic UI’s installation instructions and install the component through the shadcn registry.

```
# Fresh, compatible Next.js + Tailwind project only:
npx shadcn@latest add https://magicui.design/r/border-beam.json

# The downloadable kit already includes the component.
# Use npm ci rather than installing it again.
```

Inspect what the installer changes. Imports and Tailwind version must match your project.

Magic UI

Use a small client boundary

Border Beam uses animation. Keep that client behavior in Hero and leave database code on the server.

```
"use client";
import { BorderBeam } from "./border-beam";

export function Hero() {
 return <section className="relative overflow-hidden rounded-3xl">
  <h1>A polished interface. A real database.</h1>
  <BorderBeam duration={9} size={160} />
 </section>;
}
```

This excerpt omits padding and Reduce Motion handling; the full Hero includes both.

Magic UI

Use motion as decoration

The headline and links must work without animation. The kit checks the user’s reduced-motion preference.

```
const reduceMotion = useReducedMotion();

<div aria-hidden="true">
 {reduceMotion === false && <BorderBeam />}
</div>
```

Add other components one at a time. Measure performance instead of animating everything.

Database

Create a local development database

For local development, create a dedicated MySQL database and user with your own password, then seed the table.

```
CREATE DATABASE nextcatalog;
CREATE USER 'nextcatalog'@'127.0.0.1'
 IDENTIFIED BY 'REPLACE_WITH_A_RANDOM_PASSWORD';
GRANT SELECT ON nextcatalog.* TO 'nextcatalog'@'127.0.0.1';
-- Import database/schema.sql as your local admin account.
```

The Ubuntu setup script does this with a generated secret. Never reuse the placeholder.

Database

Keep secrets server-side

Copy the example environment file locally and enter the real credentials. Use a root-only file for VPS runtime secrets.

```
cp .env.example .env.local

DB_HOST=127.0.0.1
DB_USER=nextcatalog
DB_NAME=nextcatalog
DB_PASSWORD=your-generated-secret
```

Never prefix a password with NEXT_PUBLIC_. Never commit .env.local.

Database

Query through a server-only module

The shared module creates a small connection pool and exposes a query function. A server-only import guards the boundary.

```
import "server-only";

export async function getKits() {
 const [rows] = await pool().execute(
  "SELECT id,title,category,summary,body FROM kits LIMIT 100"
 );
 return rows;
}
```

Full code uses environment variables, a reusable pool and explicit result types.

Render the page

Fetch on the server

The page awaits database data directly. It does not need to call its own API over HTTP.

```
export const dynamic = "force-dynamic";
export const runtime = "nodejs";

export default async function Page() {
 const kits = await getKits();
 return <main>{kits.map(kit =>
  <article key={kit.id}>{kit.title}</article>
 )}</main>;
}
```

Full code adds error and empty states. Dynamic rendering keeps DB work out of static build generation.

API routes

Expose JSON when another client needs it

A route handler can serve the same public catalogue to Swift, React Native or another client.

```
export async function GET() {
 try {
  return Response.json({ kits: await getKits() });
 } catch {
  return Response.json({ error: "Catalogue unavailable" },
                       { status: 503 });
 }
}
```

This is public data only. Add authentication and authorization before exposing private records.

Local check

Build before deploying

Run locally against your database, check the page and JSON endpoint, then compile the production app.

```
npm run dev
# Open http://localhost:3000
# Open http://localhost:3000/api/kits

npm run build
npm run typecheck
```

A successful build does not prove that DNS, database permissions or TLS work.

Prepare Ubuntu

Set up a fresh server and SSH user

Use Ubuntu 24.04 LTS, a sudo deployment user, SSH key access and an A record for web.example.com.

```
# Initial provider/root session:
adduser deploy
usermod -aG sudo deploy
# Install your public SSH key for deploy and test a second login.
# DNS: A record web → VPS IPv4
```

Keep the original SSH session open until the new login works. Add AAAA only with working IPv6.

Prepare Ubuntu

Install packages and firewall rules

Allow SSH before enabling the firewall. Only Nginx receives public application traffic.

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y nginx mysql-server ufw curl \
 ca-certificates xz-utils openssl rsync snapd
sudo ufw allow OpenSSH
sudo ufw allow "Nginx Full"
sudo ufw enable
```

OpenSSH assumes port 22. Mirror rules in your provider firewall. Leave 3000 and 3306 private.

Upload

Copy source, not build-machine dependencies

Use rsync to exclude dependencies, build outputs and local secrets. Install and build on the VPS.

```
# Mac/Linux, inside project/:
rsync -av --exclude=node_modules --exclude=.next \
 --exclude=".env*" ./ deploy@VPS_IP:~/nextcatalog/
# VPS:
cd ~/nextcatalog
bash scripts/install-node24.sh
bash scripts/init-database.sh
```

The database script is for first setup only. It creates /etc/nextcatalog.env.

Run Next.js

Build and install the service

The helper installs locked dependencies, builds the app, copies it to a dedicated directory and starts systemd.

```
bash scripts/deploy.sh
sudo systemctl status nextcatalog
curl -f http://127.0.0.1:3000/api/kits
sudo journalctl -u nextcatalog -n 50 --no-pager
```

The simple deployment stops the old process during file replacement, so expect brief downtime.

Run Next.js

Use the production server

next start serves the production build. Do not run the development server as your production service.

```
WorkingDirectory=/srv/nextcatalog
EnvironmentFile=/etc/nextcatalog.env
ExecStart=/opt/node24/bin/node \
 node_modules/next/dist/bin/next start \
 -H 127.0.0.1 -p 3000
Restart=on-failure
```

Shown wrapped for teaching; the supplied systemd file has the exact valid ExecStart line.

Nginx and HTTPS

Proxy the entire application

Unlike a static React build, this app needs the Next.js server for its dynamic page and database-backed API.

```
location / {
 proxy_pass http://127.0.0.1:3000;
 proxy_set_header Host $host;
 proxy_set_header X-Forwarded-Proto $scheme;
 proxy_buffering off;
}
```

Do not use static export for this server-backed example.

Nginx and HTTPS

Install the site configuration

Replace web.example.com in the supplied file. Test configuration before reloading Nginx.

```
nano deploy/nginx.conf
sudo cp deploy/nginx.conf /etc/nginx/sites-available/nextcatalog
sudo ln -s /etc/nginx/sites-available/nextcatalog \
 /etc/nginx/sites-enabled/nextcatalog
sudo nginx -t
sudo systemctl reload nginx
```

Create the symlink once. Visit the HTTP host and confirm DNS before Certbot.

Nginx and HTTPS

Enable HTTPS and check renewal

Use Certbot after the public HTTP site works. Verify HTTPS from outside the server.

```
sudo snap install --classic certbot
sudo /snap/bin/certbot --nginx -d web.example.com
sudo /snap/bin/certbot renew --dry-run
curl -f https://web.example.com/api/kits
```

Avoid mixing apt and snap Certbot installs. Follow the official instructions for your existing setup.

Operations

Update, back up and diagnose

Review changes, build and deploy them, then check the public route. Back up MySQL before schema changes.

```
sudo journalctl -u nextcatalog -n 80 --no-pager
sudo nginx -t
sudo systemctl status nextcatalog mysql nginx
# Make a protected backup, then copy it off-server:
sudo mysqldump --single-transaction --no-tablespaces \
 nextcatalog > nextcatalog-backup.sql
```

Set umask 077 before creating backup files; restrict access and test restores in a separate database.

Production scope

Add features deliberately

The starter is a public catalogue. Login, payments, admin writes and file uploads require their own server-side controls.

```
Before launch:
• Verify database permissions and HTTPS
• Check mobile, keyboard and Reduce Motion
• Add monitoring and tested backups
• Review dependency updates
• Plan versioned releases and rollback
```

Do not treat polished UI as proof of authentication or production readiness.

Finish

Your design. Your server.

Customise the template and add focused features.
Explore more Next.js and app tutorials at CodeKits.

codekits.shop • Source, code sheets and deployment files included

References and validation

Silent illustrated walkthrough, not a live browser or VPS recording. Includes an original template with the free MIT-licensed Magic UI Border Beam; no paid Pro template is bundled. Production build and TypeScript checks pass; live MySQL and VPS/TLS still require verification.

Next.js self-hosting

Next.js deployment

Magic UI installation

Border Beam

Magic UI MIT licence

Certbot