

MINI PACK

Survive Your First Traffic Spike

Keep your Supabase + Vercel app up when launch day brings a crowd.

For Supabase + Vercel · with Claude Code or Cursor

9 prompts · 4 sections

Welcome! How to Use This Pack

Your app is fast when you and a few friends use it. Then a launch post takes off, lots of people arrive at once, and pages start timing out. The logs fill with database connection errors, and the quick fix your AI suggests ("just add caching") can make things worse, like showing one person's page to someone else.

Here's the good news. Vercel's own page on load testing says the bottleneck is likely to be the database or the outside services your code calls. That means a few known problems: too many database connections, slow queries with no index, and requests that wait forever on another service. Supabase and Vercel both document how to handle these. This pack walks you through it: finding what will break first, fixing your database connections, speeding up slow queries, caching only what's safe to share, and running a careful test before launch day.

The prompts are written for apps that use Supabase for the database and Vercel for hosting, whichever tool you built them with. Many of the ideas also apply to other Postgres databases and hosts.

Supabase and Vercel change things often. Every setting and limit mentioned here comes from their official docs as of October 2026. If a menu, port, or limit looks different for you, check the current docs. This pack can't promise your app will survive any amount of traffic. It helps you find and fix the usual weak spots before your users do.

How each prompt is laid out

- **Title:** what the prompt is.
- **Use this when...:** one or two sentences on what the prompt is for and when to reach for it.
- **Fill in:** the blanks to replace before you paste.
- **The prompt:** the box you copy.
- **Tips to customize:** optional ways to change it.

Anything in **[BRACKETS IN CAPS]** is a blank to fill in.

Works best with: Claude Code or Cursor, opened on your app's code, so the AI can read how your app talks to the database. You'll also use the Supabase dashboard and your Vercel project's logs. Prompt 1.2 only needs the errors you paste, so you can also run it in any AI chat.

The words, in one line each. A **connection** is one open line between your app's server code and the database, and the database only allows a limited number at once. A **pooler** sits in between and shares a small number of real connections among many requests. An **index** is like the index at the back of a book: it lets the database find rows without reading the whole table. A **cache** saves a response so the next visitor gets it without redoing the work. A **load test** sends lots of fake visitors to your app on purpose, to see where it slows down.

Quick Start: Find Your Problem

- **Launching soon and want to know what will break?** Prompt 1.1.
- **Site down or timing out right now?** Prompt 1.2.
- **Seeing "too many connections" or "remaining connection slots" errors?** Prompt 2.1.
- **Connections stay used up even when traffic is low?** Prompt 2.2.
- **Some pages or actions are slow even with few users?** Prompt 3.1.
- **Know which query is slow and need the right index?** Prompt 3.2.
- **Thinking about adding caching?** Prompt 4.1.
- **A payment, email, or AI service makes your pages hang?** Prompt 4.2.
- **Want to test with lots of fake visitors before launch?** Prompt 4.3.

Golden rule: Find the real bottleneck before you add caching, and never cache anything that changes depending on who's logged in.

What's Inside

1. Find What Breaks First (2 prompts)
2. Fix Your Database Connections (2 prompts)
3. Make Slow Queries Fast (2 prompts)
4. Cache, Time Out, and Test Safely (3 prompts)

SECTION 1

Find What Breaks First

Look for the weak spots now, while it's quiet.

1.1 What Breaks First When the Crowd Shows Up?

Use this when you have a launch, post, or event coming and want to know what will fail first under a lot of visitors. It's a read-only review of how your app talks to the database and outside services, with the riskiest problems listed first.

Fill in: [WHAT MY APP DOES] (e.g. "a waitlist page plus a dashboard where people save recipes"), [MY STACK] (e.g. "Next.js on Vercel, Supabase for login and database"), [EXPECTED VISITORS] (e.g. "maybe 2,000 people in the first hour", or "not sure"), [BUSIEST PAGES] (e.g. "the home page and the signup form").

```
Don't change any files yet. I'm launching soon and want to find what will break first if a lot of people use my app at once. I'm a beginner, so explain in plain words.
```

```
What my app does: [WHAT MY APP DOES]
```

```
My stack: [MY STACK]
```

```
Expected visitors: [EXPECTED VISITORS]
```

```
Busiest pages: [BUSIEST PAGES]
```

```
Read the project and check:
```

1. How my code reaches the database. Is it the supabase-js client (which goes through Supabase's Data API), a Postgres driver or ORM with a connection string (like Prisma, Drizzle, Postgres.js, or pg), or both? For each connection string, which host and port does it use? Don't print any passwords.
2. Where database clients get created. Is a new client or pool created inside a request handler instead of once at the top of the file?
3. Pool size settings. Are any left at the library default, which is usually set for a long-running server, not serverless functions?
4. Queries inside loops (one query per item in a list), and pages that load whole tables with no limit or pagination.
5. Columns my queries and Row Level Security policies filter on, and whether they have an index.
6. Calls to outside services (payments, email, AI, other APIs) that have no timeout.
7. Pages that are the same for every visitor but are rebuilt on every request.

```
For each check, say "OK", "Risk", or "Not sure", with the file path and one line on why. Then list the risks from most to least likely to take the site down during a spike, with a short fix for each and which prompt type would fix it
```

(connections, slow queries, caching, or timeouts). If you're not sure how Supabase or Vercel behaves today, tell me to check their docs instead of guessing.

 **Tips to customize:** Run this in plan mode in Cursor or Claude Code, so the AI explains before it edits anything. Supabase's connection guide says the supabase-js client uses its Data API, which has its own built-in connection pooler, so the pooler steps in Prompt 2.1 matter most if your server code also uses a Postgres driver or ORM. Supabase's production checklist also points to its Performance Advisor, which flags issues like missing indexes. If you expect a lot of signups, our Signup Emails pack (Prompt 2.2) covers Supabase's auth email rate limit, and our Ship It Safe pack (Prompt 5.1) covers rate limits for abuse.

1.2 It's Down Right Now: Calm Triage

Use this when your site is slow, timing out, or showing errors because of a sudden rush of visitors. It's for finding the cause from the errors you can see, and getting the site back up before you make bigger changes.

Fill in: [WHAT I'M SEEING] (e.g. "pages spin, then show a 504 error"), [ERRORS FROM MY LOGS] (paste the latest lines from Vercel's logs and any Supabase errors, with passwords and keys removed), [WHEN IT STARTED] (e.g. "about 20 minutes after my post went up"), [WHAT I CHANGED RECENTLY] (e.g. "nothing", or "added a new dashboard page yesterday").

My app is struggling under a sudden traffic spike. Help me find the cause and get it back up. Don't change any code until I approve a plan. Keep it calm and step by step.

What I'm seeing: [WHAT I'M SEEING]
Errors from my logs: [ERRORS FROM MY LOGS]
When it started: [WHEN IT STARTED]
What I changed recently: [WHAT I CHANGED RECENTLY]

Some error messages I know about:

- "FATAL: remaining connection slots are reserved for non-replication superuser connections" from Postgres means the database has run out of connections.
- A 504 error with FUNCTION_INVOCATION_TIMEOUT on Vercel means a function ran longer than its maximum duration.
- Supabase's docs say "Connection refused" is usually an IP ban rather than an outage.

Please:

1. Tell me which of these is most likely happening: out of database connections, slow queries piling up, an outside service hanging, or something else. Quote the exact log line that points to it.
2. Tell me where to look to confirm it, step by step: the Database Connections page in my Supabase dashboard, Supabase's reports, and my Vercel project's logs.
3. Give me the fastest safe way to get the site working again, and say what it

costs or risks.

4. Then list the proper fixes for after the rush, in order.

If a message isn't one you recognize, say so and tell me to search Supabase's or Vercel's troubleshooting docs instead of guessing.

 **Tips to customize:** Supabase's docs list three ways out of the connection-slots error: move to a larger compute size, set your clients to use fewer connections, or raise the database's connection limit yourself (they list things to consider before you do). The Database Connections page in your Supabase dashboard shows what each connection is doing, what's blocking what, and lets you end a stuck session. If you're on a Supabase Team or Enterprise plan and expect a big launch, Supabase's production checklist says to contact support beforehand so they can keep an eye on your project. Once the site is back, use Prompt 2.1 or 3.1 for the real fix.

SECTION 2

Fix Your Database Connections

When every request opens its own connection, the database runs out. Here's how to share them.

2.1 Switch My Server Code to the Pooler the Right Way

Use this when you see "too many connections" errors, or your server code connects to Postgres directly from Vercel functions. It's for moving to Supabase's pooler with the settings serverless code needs, so you don't trade one error for another.

Fill in: [MY DATABASE LIBRARY] (e.g. "Prisma", "Drizzle with Postgres.js", "pg", or "not sure"), [WHERE MY SERVER CODE RUNS] (e.g. "Next.js API routes and server actions on Vercel"), [HOW I RUN MIGRATIONS] (e.g. "prisma migrate deploy", "supabase db push", or "not sure").

My server code connects to my Supabase Postgres database from serverless functions on Vercel, and I want to set up connection pooling the right way. Show me the plan before you change anything.

My database library: [MY DATABASE LIBRARY]
Where my server code runs: [WHERE MY SERVER CODE RUNS]
How I run migrations: [HOW I RUN MIGRATIONS]

What I know from Supabase's docs:

- For serverless or edge functions, Supabase recommends the shared pooler in transaction mode. That string comes from the Connect button in the dashboard, under "Transaction pooler", and uses port 6543.
- Transaction mode doesn't support prepared statements, so they must be turned off: prepare: false for Postgres.js and Drizzle, or pgbouncer=true on the connection string for Prisma.
- The database client should be created once at the top of the file (module scope), not inside each request.
- Library defaults are too high for serverless. Postgres.js, for example, defaults to 10 connections per warm instance.
- Migrations and other long single sessions should use the direct connection (port 5432), or session mode (also port 5432) on networks that only support IPv4.

What I know from Vercel's docs:

- With Fluid compute, one function instance can handle several requests at once and shares the same pool.
- Vercel suggests creating the pool once, registering it with attachDatabasePool from @vercel/functions, and using a short idle timeout.

Please:

1. Find every place that creates a database client, and tell me which connection string each one uses (by environment variable name, never the value).

2. Move request-time code to the transaction pooler string, keep migrations on the right connection, and turn off prepared statements for my library.

3. Supabase's guide starts serverless pools at 1 connection per instance, while Vercel's Fluid compute guide warns that a maximum of 1 can hurt concurrency. Explain the tradeoff for my setup and suggest a small starting number.

4. List the environment variables I need to add or change in Vercel and anywhere else I run code, with what each is for.

5. Tell me how to check that it worked.

If my library needs another setting for transaction mode, tell me to check Supabase's guide for that library.

 **Tips to customize:** Supabase's docs say to take the host, port, and username from the string you copy, not type them yourself. Pooled connections use a different username (`postgres`. plus your project ref), and the pooler host can't be worked out from your region. Supabase also notes that Postgres.js sends several queries at once by default, which can cause hanging queries or mixed-up results in transaction mode, so read its Postgres.js guide if you use that library. To see how many connections your compute size allows, check the Compute and Disk page in Supabase's docs. Our Ship It Live pack (Prompt 1.1) covers env var changes that don't seem to apply on Vercel.

2.2 Find Who's Hogging My Connections

Use this when your connections stay used up even when few people are on the app, or queries hang and then time out. It's for finding the sessions that hold connections or locks, and the code that leaves them open.

Fill in: [WHAT I'M SEEING] (e.g. "the Database Connections page shows lots of 'idle in transaction'", or "queries hang for a minute and fail"), [WHAT I PASTED] (leave blank the first time, then paste the results of the query the AI gives you, or describe what the Database Connections page shows).

Don't change any files yet. My Supabase database connections stay used up, or some queries hang. Help me find out what's holding them, step by step. I'm a beginner.

What I'm seeing: [WHAT I'M SEEING]

What I pasted: [WHAT I PASTED]

What I know from Supabase's docs:

- `pg_stat_activity` shows live connections, and its `username` column shows which service owns each one (for example, `authenticator` is the Data API, `supabase_auth_admin` is Auth, and `postgres` is the dashboard and outside tools like Prisma).
- "idle in transaction" means a transaction was opened and never committed or rolled back. It holds locks until it closes.
- `pg_cancel_backend(pid)` cancels a running query but keeps the connection.
- `pg_terminate_backend(pid)` closes the whole connection, and it's the only way to end an idle-in-transaction session.

Please:

1. Give me a read-only SQL query I can run in the Supabase SQL editor to see connections grouped by role and state, plus any sessions that are blocked and what is blocking them.
2. When I paste the results, explain them in plain words and say which ones look like a problem.
3. Then search my code for the likely cause: transactions that can end without commit or rollback (for example, an early return or an error inside the transaction), clients that never get released back to the pool, or a new pool created on every request.
4. Suggest a fix for each cause, and tell me whether it's safe to cancel or end any session right now, and which function to use.



Tips to customize: You can see the same information without writing SQL on the Database Connections page in your Supabase dashboard, which shows session states and blocking chains and lets you end a stuck session. Ending a stuck session is a quick fix, but if the code keeps leaving transactions open it will happen again, so fix the code too. If the problem only shows up sometimes, our Fix My Bugs pack (Prompt 3.1) helps with bugs that come and go.

SECTION 3

Make Slow Queries Fast

One missing index can make a page slow for everyone at once.

3.1 Find My Slowest Queries

Use this when pages or actions feel slow, or your database works hard even with few users. It's for finding the queries that take the most total time, using the tools Supabase already gives you.

Fill in: [SLOW PARTS OF MY APP] (e.g. "the dashboard takes 4 seconds to load", or "everything feels a bit slow"), [WHAT I PASTED] (paste rows from the Query Performance report or the SQL results the AI asks for, or leave blank for now).

Don't change any files yet. I want to find the slowest and busiest queries in my Supabase database and understand them. I'm a beginner.

Slow parts of my app: [SLOW PARTS OF MY APP]

What I pasted: [WHAT I PASTED]

What I know from Supabase's docs:

- The Query Performance report in the Supabase dashboard lists slow queries.
- The `pg_stat_statements` extension tracks how often each query runs and how long it takes (`calls`, `mean_exec_time`, `total_exec_time`). Its numbers add up from the last time the stats were reset.
- The Supabase CLI has inspection commands like `"supabase inspect db outliers"`, `"long-running-queries"`, and `"seq-scans"` (after `"supabase link"`).

Please:

1. Tell me how to check whether `pg_stat_statements` is turned on, and how to turn it on from the dashboard if it isn't.
2. Give me a read-only SQL query that lists the top 10 queries by total time, with calls and average time.
3. When I paste the results, explain the top 3 in plain words: which table they read, roughly what they're doing, and why they might be slow.
4. Match each one to the code in my project that runs it (file path), including queries made through `supabase-js` or my ORM.
5. For each one, say whether the likely fix is an index, a smaller query (fewer columns or rows, or pagination), or running it less often. Don't write the fix yet.

 **Tips to customize:** Sort by total time first. A fairly fast query that runs on every page view can add up to more database time than one slow report you open once a day. Supabase's docs say a long-running session or a high total time is something to investigate, not by itself a reason to cancel

queries or reset the stats. Run this again after your fixes and compare, ideally after resetting the stats so the numbers start fresh. When you know which query to fix, use Prompt 3.2.

3.2 Add the Right Indexes Safely

Use this when you've found a slow query and want to add an index for it without slowing down the rest of your app. It's for checking that the index really helps, adding it safely, and speeding up Row Level Security policies too.

Fill in: [THE SLOW QUERY] (paste it from the Query Performance report or `pg_stat_statements`), [THE TABLE AND ROUGH SIZE] (e.g. "recipes, about 200,000 rows"), [HOW I CHANGE MY DATABASE] (e.g. "Supabase migrations in my repo", "the SQL editor", or "Lovable makes the changes").

I want to speed up a slow query in my Supabase database with the right index. Show me the plan before you change anything.

The slow query: [THE SLOW QUERY]

The table and rough size: [THE TABLE AND ROUGH SIZE]

How I change my database: [HOW I CHANGE MY DATABASE]

What I know from Supabase's and Postgres's docs:

- Indexes usually help on columns in where clauses, join columns, and order by columns. A primary key is already indexed.
- Supabase's `index_advisor` extension suggests indexes for a query, but only single-column B-tree indexes. It's also in the dashboard: open the Query Performance report, pick a query, then the indexes tab.
- "explain" shows how Postgres runs a query. A "Seq Scan" on a big table often means a missing index. If an index doesn't lower the cost, remove it.
- Too many indexes slow down writes, so only add ones that help real queries.
- A normal "create index" blocks writes to the table while it builds. "create index concurrently" doesn't block writes but takes longer, and it can't run inside a transaction block.
- For Row Level Security, Supabase recommends an index on every column a policy filters on (like `user_id`), and writing `auth.uid()` as `(select auth.uid())` inside policies.

Please:

1. Explain the query plan in plain words, and tell me how to run "explain" on it myself.
2. Suggest the index or indexes that would help, and why. Use `index_advisor`'s idea as a starting point if it fits.
3. Check the RLS policies on this table for unindexed columns and for `auth.uid()` calls that aren't wrapped in a select.
4. Write the change in the way I change my database. If it uses "concurrently", tell me whether my migration tool runs it inside a transaction, which would make it fail.
5. Tell me how to confirm afterward that the query got faster.

💡 **Tips to customize:** On a small table, Postgres may still choose to read the whole table even with an index, because it decides that's quicker. That's normal. Supabase's docs suggest running `analyze` on a table now and then so Postgres has up-to-date stats about it. If Lovable manages your database, read our Graduate From Lovable pack (Prompt 4.3) before changing it from outside Lovable. Our Ship It Safe pack (Prompts 2.1 and 2.2) covers writing RLS policies that keep data safe. This prompt is only about making them fast.

SECTION 4

Cache, Time Out, and Test Safely

Share only what's safe to share, stop waiting forever, and practice before the real crowd comes.

4.1 Cache Without Leaking One User's Data

Use this when you want to add caching, or an AI already added it and you're not sure it's safe. It's for caching pages everyone sees the same, while making sure no one can ever get another person's cached page.

Fill in: [PAGES I WANT TO CACHE] (e.g. "home page, pricing, public recipe pages"), [PAGES THAT ARE PER USER] (e.g. "dashboard, settings, anything after login"), [MY FRAMEWORK] (e.g. "Next.js App Router", or "Vite React app with API routes on Vercel").

Don't change any files yet. I want to add caching to my app on Vercel, and I must never show one user's data to another user. Review my caching and make a safe plan. I'm a beginner.

Pages I want to cache: [PAGES I WANT TO CACHE]

Pages that are per user: [PAGES THAT ARE PER USER]

My framework: [MY FRAMEWORK]

What I know from Vercel's docs:

- Vercel's CDN caches a function's response only when it sends a Cache-Control header with s-maxage (for example, "public, s-maxage=60, stale-while-revalidate=300").
- It won't cache a response that sets a cookie (Set-Cookie), a request with an Authorization header, or a response marked private, no-cache, or no-store.
- A request that only carries a login cookie isn't on that list, so a per-user page that sends s-maxage could still be cached and shown to others.
- The x-vercel-cache response header shows whether a response came from the cache.

What I know from Next.js's docs:

- Functions marked "use cache" can't read cookies() or headers(). Values passed in as arguments, or used from outside the function, become part of the cache key.

Please:

1. Find every place that sets Cache-Control, s-maxage, revalidate, "use cache", unstable_cache, or any other cache, in code and in vercel.json or next.config.
2. For each one, say "safe" (same for everyone), "leak risk" (depends on who's logged in but could be shared), or "not sure", with the file path and why.
3. Make sure every per-user page and API route sends Cache-Control: private, no-store, and that cached functions never fetch the current user's data unless the user's ID is part of the cache key.
4. Suggest caching for the public pages I listed, with how long, and how updates

will show up.

5. Give me a 3-step test with two different accounts and a logged-out browser to prove no one sees someone else's data.

 **Tips to customize:** Vercel's docs also say it won't cache a response whose Vary header names Cookie, because nearly every visitor would get their own copy. Check what you're caching with your browser's developer tools: open the Network tab, click a request, and look for `x-vercel-cache` in the response headers. Caching helps the pages everyone sees, but it won't fix a slow query on a logged-in page, so use Section 3 for that. Our Ship It Safe pack (Prompt 3.2) has a full test for whether User A can see User B's data.

4.2 Timeouts and Safe Retries for Outside Services

Use this when a payment, email, AI, or other outside service makes your pages hang when it's slow. It's for adding timeouts so one slow service can't tie up your app, and retries that never charge or email anyone twice.

Fill in: [OUTSIDE SERVICES I CALL] (e.g. "Stripe, Resend for email, OpenAI"), [WHAT HAPPENS WHEN ONE IS SLOW] (e.g. "the checkout button spins forever", or "not sure").

Add timeouts and safe retries to the calls my app makes to outside services. Show me the plan before you change anything.

Outside services I call: [OUTSIDE SERVICES I CALL]

What happens when one is slow: [WHAT HAPPENS WHEN ONE IS SLOW]

What I know:

- fetch can take a timeout signal with `AbortSignal.timeout(milliseconds)`, which stops the request after that long.
- On Vercel, a function that runs past its maximum duration is stopped with a 504 error (`FUNCTION_INVOCATION_TIMEOUT`). In Next.js App Router, a route can set its own limit with `"export const maxDuration"`.
- On Vercel, waiting for another service doesn't count as active CPU time, but the function's memory time keeps counting while it runs.

Please:

1. List every call to an outside service in my code, with the file path, and whether it has a timeout. Include calls made through an official library, and tell me whether that library has its own timeout or retry setting.
2. Add a sensible timeout to each one, explaining why you chose that number.
3. Add retries only where repeating the call is safe, like reading data. For anything that charges money, sends an email, or creates something, don't retry unless the provider supports an idempotency key or the code can check first whether it already happened.
4. When you retry, wait longer between each try, stop after a few tries, and respect a `Retry-After` header on 429 or 503 responses.
5. Show the user a clear, friendly message when a service fails, instead of an

endless spinner.
6. Set a sensible `maxDuration` on the routes that call slow services.

 **Tips to customize:** Check each provider's own docs for its timeout and retry settings before changing defaults. Some official libraries may already retry for you, and adding your own retries on top can multiply the calls. If AI calls are your slowest part, our Ship It Safe pack (Prompt 5.2) covers spend caps so a spike doesn't turn into a surprise bill. For payment webhooks that arrive twice, our Get Paid pack (Prompt 2.3) covers handling duplicate events.

4.3 A Load Test I'm Allowed to Run

Use this when you want to see how your app handles lots of visitors before launch, without breaking any rules. It's for a small, staged test on your own app, using a copy of your database instead of the real one.

Fill in: [MY APP'S KEY ACTIONS] (e.g. "open home page, sign in, load dashboard, save a recipe"), [WHERE I'LL RUN THE TEST] (e.g. "a production build on my computer", or "a Vercel preview, my plan allows load testing"), [MY TEST DATABASE] (e.g. "a separate Supabase project with fake data"), [MY VERCEL PLAN] (e.g. "Hobby", "Pro", or "Enterprise").

Help me plan and write a small, safe load test for my own app using k6. I'm a beginner. Don't run anything for me.

My app's key actions: [MY APP'S KEY ACTIONS]
Where I'll run the test: [WHERE I'LL RUN THE TEST]
My test database: [MY TEST DATABASE]
My Vercel plan: [MY VERCEL PLAN]

Rules I must follow:

- Only test my own app, never someone else's site or a service I don't control.
- Vercel's load testing policy says load testing is only permitted on Enterprise plans, and unannounced tests may get the testing IP addresses blocked. Unless my plan allows it, I'll test a production build running on my own computer instead of my Vercel deployment.
- Supabase's production checklist recommends load testing, preferably on a staging environment. I'll point the test at a separate test project with fake data, never my real users' data.
- Don't send test traffic to real payment, email, or AI services. Use test mode or fake them.

Please:

1. Tell me how to run a production build of my app locally, connected to my test database.
2. Write a k6 script that walks through my key actions with fake test accounts, starting with a smoke test (very few virtual users) and then a slow ramp up and back down, not an instant jump.
3. Add thresholds so the test fails if more than 1% of requests error or responses

get too slow, and explain how to pick those numbers.

4. Tell me what to watch while it runs: k6's results, my Supabase project's Database Connections page and reports, and my app's logs.
5. Explain how to read the results and which prompt type (connections, slow queries, caching, or timeouts) fits each kind of failure.
6. Remind me how to stop the test right away if something goes wrong.

 **Tips to customize:** k6 installs on Mac, Windows, and Linux, and you start a test with `k6 run script.js`. Grafana's k6 docs suggest starting with a smoke test before any bigger test. A test on your own computer won't behave exactly like Vercel's servers, but Vercel's own load testing article says the bottleneck is usually your database or the APIs your code calls, which this still tests. Read Vercel's and Supabase's current policies before testing anything they host. Our Ship It Live pack (Prompt 1.2) covers keeping preview and production settings apart.

Bonus: Magic Phrases for Traffic Spikes

- "Don't change any files yet. Find the bottleneck first."
- "Show me the connection string by environment variable name, never the value."
- "Does this depend on who's logged in? Then don't cache it."
- "Prove it with a query plan or a measurement, not a guess."
- "Add a timeout, and only retry what's safe to repeat."
- "If you're not sure how Supabase or Vercel works now, tell me to check the docs."

Survive Your First Traffic Spike

That's the whole pack: **9 prompts across 4 sections**. Find what breaks first, fix your database connections, make slow queries fast, and cache, time out, and test safely, so your first big day is a good one.

Happy building!