TUTORIAL · SEPTEMBER 14, 2026

Docker für Delphi-Entwickler

Ein fünfteiliger Kurs über Docker für Delphi-Entwickler — von der Installation von Docker Desktop und einem PostgreSQL-Container über Images, Tags und Compose bis zur dreistufigen Isolationsarchitektur, abgeschlossen mit einer vollständigen FireDAC-Kommandozeilenanwendung zum Bauen und Ausführen.

By Dr. Holger Flick



Sie brauchen ein echtes Backend zum Entwickeln — etwa eine Postgres-Datenbank — auf Ihrem Windows-Rechner, und zwar jetzt. Also suchen Sie das Installationsprogramm. Es registriert einen Windows-Dienst, ändert Ihren `PATH`, legt an drei Stellen Konfigurationsdateien ab und verlangt ein Superuser-Passwort, das Sie bis Donnerstag vergessen haben. Es funktioniert — bis ein Projekt Version 15 braucht und ein anderes Version 17 und sich zwei Dienste um denselben Port streiten. Und die Deinstallation hinterlässt verwaiste Dienste, übrig gebliebene Verzeichnisse und eine Registry, die Sie lieber nicht anfassen.

Jeder erfahrene Desktop-Entwickler kennt genau dieses Unbehagen. Dieses Tutorial ist der Ausweg, ausgeliefert als fünfteilige Artikelserie: Ein Befehl gibt Ihnen eine echte laufende Datenbank, ein weiterer lässt sie spurlos verschwinden — und dieselbe Idee skaliert bis zur vollständigen Architektur Web ’ Backend ’ Datenbank, die hinter praktisch jedem produktiven System steht, das Sie je benutzt haben.

Kein Kubernetes, keine DevOps-Liturgie, kein YAML um seiner selbst willen — nur die Denkmodelle, die Container für jemanden einleuchtend machen, der ohnehin in Klassen und Objekten denkt, mit echtem Delphi- und FireDAC-Code auf dem Weg. Lesen Sie die Teile der Reihe nach: Jeder setzt genau dort an, wo der vorherige aufgehört hat. Und bleiben Sie für den sechsten Eintrag — ein vollständiges Praxisbeispiel, das alle fünf Teile in einem einzigen baubaren Projekt zusammenführt.

Teil 1 — Was ein Container wirklich ist

Docker Desktop installieren und den ersten Container starten (1/5)

Drei Wörter stehen zwischen Ihnen und allem Weiteren: **Image**, **Container**, **Registry**. Dieser Teil bringt Docker Desktop auf Windows, startet `hello-world` und liefert Ihnen dann die eine Analogie, die den Rest der Serie leicht macht — eine Delphi-Analogie, die Sie täglich verwenden, mit einer Klasse, einem Objekt und einem Konstruktor. Auch die Lizenzfrage wird ehrlich beantwortet, mit klar benannten Open-Source-Alternativen statt Schweigen.

Teil 2 — Ein echter PostgreSQL-Server, und FireDAC spricht mit ihm

PostgreSQL im Container, erreicht mit Delphi FireDAC (2/5)

Jetzt wird das Versprechen eingelöst: ein produktionstauglicher PostgreSQL-Server, gestartet mit einem einzigen Befehl, ohne Installer, ohne Dienst im Windows-Autostart — und ein echtes Delphi-Konsolenprogramm, das mit FireDAC eine Tabelle anlegt, Zeilen einfügt und sie wieder aus dem Container ausliest. Zwischen Ihnen und dieser Ausgabe liegt genau ein ehrlicher Haken, eine Abhängigkeit, die der Container Ihnen nicht geben kann, und sie stoppt die meisten Menschen beim ersten Versuch. Hier bekommt sie einen eigenen Abschnitt.

Teil 3 — Woher dieses Image eigentlich kam

Docker-Images, Tags und Repositories erklärt (3/5)

Sie tippen `postgres`, und Docker liefert einen Datenbankserver. Woher kam er? Dieser Teil entschlüsselt `docker.io/library/postgres:16` Stück für Stück, bis Sie eine Image-Referenz lesen wie einen Dateipfad — Registry, Repository, Tag. Dazu: warum der zweite Pull fast sofort fertig ist, wie Sie ein vertrauenswürdiges Image von einem beliebigen unterscheiden, und das am weitesten verbreitete Missverständnis in ganz Docker — was der Tag `latest` eben **nicht** bedeutet.

Teil 4 — Ihr ganzer Stack in einer Datei

Docker Compose: Ihr gesamter Stack in einer Datei (4/5)

Eine echte Anwendung ist nie ein einzelner Container. Jeden Dienst von Hand zu starten heißt, sich pro Dienst einen Absatz voller `docker run`-Schalter zu merken und sie auf jeder Maschine in der richtigen Reihenfolge auszuführen — ein vergessener Schalter, und das Backend findet die Datenbank nicht. Compose beseitigt diese ganze Problemklasse mit einer lesbaren Datei, die in Ihrem Repository liegt. Nebenbei liefert dieser Teil die zwei Antworten, die sich die Serie bewusst aufgespart hatte: wie Dienste einander ohne eine einzige IP-Adresse finden, und wie Ihre Daten einen Neustart überleben.

Teil 5 — Die Architektur, die Sie wirklich ausliefern können

Die Datenbank isolieren: Web ’ Backend ’ Postgres in Docker (5/5)

Der Stack aus Teil 4 lief, er war aufgeräumt — und er hatte einen stillen Fehler, der absichtlich drin geblieben war: Die Datenbank war vom Host aus erreichbar. Bequem, und genau die Form, die Sie niemals ausliefern dürfen. Das Finale baut die dreistufige Schichtung, die hinter praktisch jeder produktiven Webanwendung steht: ein Web-Client, ein Backend, das als Einziges die Zugangsdaten hält, und Postgres in einem privaten Netzwerk ganz ohne öffentliche Tür. Das Backend erscheint in beiden Varianten, die es wert sind — TMS XData in Delphi und Express in TypeScript —, damit Sie nach Passung entscheiden statt nach Mode. Und die Isolation wird bewiesen, nicht behauptet.

Das Praxisbeispiel — eine vollständige FireDAC-Anwendung von Anfang bis Ende

Delphi und PostgreSQL von Grund auf: eine FireDAC-CLI in vier Units

Die meisten Datenbankbeispiele hören genau dort auf, wo es interessant wird: ein Formular, eine hingelegte `TFDConnection`, ein hartcodiertes Passwort, ein Screenshot. Dieses nicht. Es ist die ganze Serie in einem baubaren Projekt — eine `compose.yaml`, die den Server startet, und eine Delphi-Kommandozeilenanwendung in vier Quelldateien, die ihre eigene Datenbank anlegt, zwei Tabellen aufbaut, Zeilen innerhalb einer Transaktion einfügt, sie per Join wieder ausliest und einen formatierten Bericht ausgibt. Jede Datei steht vollständig da, jeder Befehl lässt sich in ein Windows-Terminal einfügen, und gebaut wird von der Kommandozeile, ohne die Maus anzufassen. Enthalten ist außerdem eine Meinung, über die sich streiten lässt: ob eine Anwendung ihre eigene Datenbank überhaupt anlegen sollte.

Was Sie danach können

- Einen echten PostgreSQL-Server mit einem Befehl auf Ihrer Maschine starten — und ihn ebenso sauber wieder entfernen, ohne Rückstände.
- Jede Image-Referenz auf einen Blick lesen, Versionen bewusst statt zufällig festlegen und wissen, warum `latest` in allem, worauf Sie sich verlassen, eine Falle ist.
- Einen kompletten Mehr-Dienste-Stack in einer `compose.yaml` beschreiben, die sich auf jedem Laptop und jedem Server identisch verhält.
- Web, Backend und Datenbank so anordnen, dass die wertvollste Schicht tatsächlich nur über Code erreichbar ist, den Sie selbst geschrieben haben.
- Delphi und FireDAC mit einer containerisierten Datenbank verbinden, ohne an der einen Abhängigkeit zu scheitern, an der alle scheitern.

> Ein Container ist ein Objekt, ein Image seine Klasse und `docker run` der Konstruktor. Sobald das einrastet, hört Docker auf, ein Rätsel zu sein, und wird zu einem Werkzeug, nach dem Sie ohne Nachdenken greifen.

INDEPENDENT · AD-FREE · READER-SUPPORTED

Free to read. Not free to make.

Every tutorial, article, and line of example code on Holger's Code is published without a paywall and without ads. This page is about what keeps it that way.

Professional work, published free

The content here is held to the same standard as professional client work: every tutorial is built as a real project, tested against current versions, documented step by step, and revisited when the frameworks move on. A single in-depth tutorial routinely takes days — and that's before the bills behind it: servers, storage, build infrastructure, licenses, and the hardware everything is verified on.

If an article or tutorial here has saved you an afternoon of debugging, taught you a technique, or helped you ship, it created real value for you. Supporting it isn't charity — it's paying for professional work you found useful, at whatever price you choose. And if you can't, keep reading anyway. That's what the free part means.

Ways to support

KO-FI

Buy me a coffee

A one-time tip — no account, no subscription, any amount.

BOOKS & COURSES

Buy the paid work

The paid tier of everything published free here — and you get something lasting in return.

SPREAD THE WORD

Share an article

Send a tutorial to a colleague, cite an article, subscribe to the RSS feed.

Nothing here goes behind a paywall, whether you chip in or not. But if something on this site earned its keep today, this is where to say so.

Support on Ko-fi —