A WORKSHOP WORKBOOK

ZeroDB Local & Desktop

Run the full ZeroDB stack on your machine

ZERODB LOCAL — AVAILABLE · V0.3.0

ZERODB DESKTOP — IN DEVELOPMENT

Develop against ZeroDB offline. Local runs the whole stack — vectors, tables, files, memory and events — on your machine, mirroring the cloud API on `localhost`, then syncs up when you are ready. Desktop will wrap it in a one-click app.

The local workflow

Four steps, and the last one is optional until the day you need it. Everything before it happens with the network off.

01 — Install

Docker Compose today, or `pip install zerodb-local`.

02 — Init

`zerodb init` checks Docker, scans ports, and pulls the embedding model.

03 — Develop

Point your app at `http://localhost:8000` and use the same ZeroDB API.

04 — Sync

Link a cloud project and push your work up — or pull the cloud down.

The golden rule: if the base URL is the only thing that changes, nothing else can break.

Local exposes the same API surface as ZeroDB Cloud. Keep that one value in config and the same code runs on your laptop, in CI, and in production.

Why run ZeroDB locally

Three reasons, and they compound: the second is what makes the first safe, and the third is what makes both permanent.

NO NETWORK

Offline-first

Build and test on a plane, in a locked-down environment, or on a Friday afternoon — with no cloud spend and no rate limits.

NO REWRITE

API parity

Local exposes the same API surface as ZeroDB Cloud on `localhost`, so code written against Local runs unchanged against the cloud.

ON YOUR TERMS

Sync when ready

Push local work up to a cloud project, or pull it down, with the sync commands — running on ZeroDB's bundle sync.

SAME DATA MODEL, DIFFERENT MACHINE

Vectors	Tables	Files	Memory	Events
Workbook 02	Workbook 01	Object storage	Workbook 06	Streaming

It is the same data model you already know — just running on your laptop.

What is inside ZeroDB Local V0.3.0

Not a mock and not a shim — the real engines behind ZeroDB, running as local services on your machine.

COMPONENT	TECHNOLOGY	PORT	PURPOSE
API Server	FastAPI	`8000`	REST API mirroring the cloud — docs at `/docs`, health at `/health`
Database	PostgreSQL 16 + pgvector	`5432`	Tables and vectors
Vector search	Qdrant	`6333`	Semantic search — dashboard at `:6333/dashboard`
Object storage	MinIO	`9000` `9001`	S3-compatible files — console on 9001
Event streaming	RedPanda	`9092` `9644`	Kafka-compatible events — console on 9644
Embeddings	sentence-transformers `BAAI/bge-small-en-v1.5` · 384-dim	`8001`	Local embedding generation
Dashboard	React 18 + Next.js 14	`3000`	The web UI you open after setup

> Seven services, one command. The ports matter mainly when something else on your machine already claimed one — which is exactly what the wizard checks next.

Install and run

Two paths to the same stack. Take Compose if you want every service explicit; take pip if you want one command and a Python environment you already have.

OPTION A — DOCKER COMPOSE

TERMINAL

```
cd zerodb-local
cp .env.local.example .env.local
docker-compose up -d

# API:        http://localhost:8000   (docs at /docs, health at /health)
# Dashboard:  http://localhost:3000
```

OPTION B — PIP

TERMINAL

```
pip install zerodb-local==0.3.0        # Python 3.9–3.13
zerodb serve --host 0.0.0.0 --port 8765
```

NO KEY? IT PROVISIONS ONE

Running `zerodb serve` without a cloud API key auto-provisions a temporary cloud project and prints a `claim_url` — the same Instant DB flow as the cloud, so a local start and a cloud start look identical.

The setup wizard `zerodb init`

A guided run, about sixty seconds, that finds the problems before your first request does.

01	**System check**	OS and architecture, RAM, disk, internet reachability.
02	**Docker verification**	Docker installed and the daemon running — with an install link if it is not.
03	**Port scan**	Checks 5432, 6333, 8000, 8001, 9000, 9092 and 3000; offers to free or remap conflicts.
04	**Model download**	Pulls the local embedding model — `BAAI/bge-small-en-v1.5`, about 133 MB.
05	**Service startup**	Compose up with per-service health checks, then opens the dashboard on `:3000`.

WHEN SOMETHING IS OFF

```
zerodb doctor          # full diagnostics
zerodb doctor --fix    # apply fixes automatically
```

Write once, run local or cloud

Projects, Vectors, Tables and NoSQL, Files, Memory and Events all behave the same in both places. Here is the entire diff between environments.

LOCAL

```
Base URL:
  http://localhost:8000
```

CLOUD

```
Base URL:
  https://api.ainative.studio
```

THAT IS THE WHOLE DIFFERENCE

Paths, payloads, response shapes and error codes are the same. Read the base URL from an environment variable and the promotion from laptop to production is a config change, not a migration.

Interactive docs, locally

The full OpenAPI browser is served at `http://localhost:8000/docs` — the same reference, offline.

Health before you debug

`/health` answers first. If it is unhappy, the problem is a service, not your code.

Sync to the cloud

Link a cloud project and move data in either direction — after you have looked at the diff.

Local		Cloud
localhost:8000	`sync apply →` `← cloud pull`	api.ainative.studio

THE SYNC COMMANDS

```
zerodb cloud login                # authenticate
zerodb cloud link <project_id>    # link local ↔ an existing cloud project
zerodb sync plan                  # preview the diff before syncing
zerodb sync apply                 # push local → cloud
zerodb cloud pull                 # pull cloud → local
zerodb cloud status               # show sync status
```

CONFLICT POLICY — SET IN `.env.local`

`newest-wins`	`local-wins`	`cloud-wins`	`manual`
latest timestamp	your machine rules	the server rules	you decide each

> Sync runs on ZeroDB's bundle sync underneath — see the Platform workbook's Cloud Sync section for the API it calls.

ZeroDB Desktop IN DEVELOPMENT

A planned native app that bundles the local server, orchestrator and dashboard into a single one-click install — no manual Docker steps. It is not shipping yet; today, ZeroDB Local runs via Docker Compose or pip.

PLANNED TARGETS		
macOS	**Windows**	**Linux**
`.dmg`	`.exe`	`AppImage · .deb · .rpm`

WHAT IT WILL SHOW

Service dashboard

Green or red health for every service, plus resource usage at a glance.

One-click actions

New Project, View Metrics, Sync to Cloud — without touching a terminal.

Logs and tray controls

Per-service logs, and start/stop for the whole stack from the system tray.

Under a minute

Docker images pre-bundled, so there is no `docker pull` during setup.

Today: use Docker Compose or pip. Everything in this workbook other than this page describes software you can run right now.

Patterns and anti-patterns

PATTERNS

Develop local, ship cloud

The same API surface means zero rewrite when you deploy.

Run the wizard first

`zerodb init` catches Docker and port issues before they bite.

Preview before you sync

`zerodb sync plan` shows the diff. Apply only when it reads right.

Change the default creds

Local ships with dev credentials — never reuse them anywhere real.

ANTI-PATTERNS

Hardcoding localhost

Read the base URL from config so the same code hits the cloud later.

Treating Local as production

It is a development and offline environment, not a hosted deployment.

Ignoring port conflicts

If 8000 or 5432 is already taken, let the wizard remap it.

Assuming Desktop is available

It is in development. Use Docker Compose or pip today.

The same ZeroDB, on your laptop. Build offline, sync when you are ready, ship to the cloud unchanged.

Run it, then connect it

One command gets the stack up. The interesting part is that nothing you write against it has to change when you move.

01 Run it now

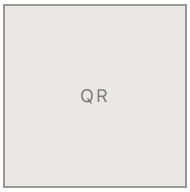
`pip install zerodb-local` or `docker-compose up -d`, then open `localhost:3000`.

02 Connect to the cloud

`zerodb cloud login && zerodb cloud link <project_id>`, then preview with `sync plan`.

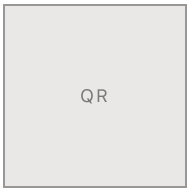
03 Learn more

Local docs and architecture at `docs.ainative.studio` and in the repo under `docs/zerodb-local`.



Install ZeroDB Local

Compose file, pip package, quickstart.



Read the architecture

Services, ports and the sync design.