@push.rocks/smartmongo

A module for creating and managing a local MongoDB instance for testing purposes.

- readme.md for @push.rocks/smartmongo
- changelog.md for @push.rocks/smartmongo

readme.md for @push.rocks/smartmongo

A MongoDB memory server toolkit for testing and development — spin up real MongoDB replica sets on the fly with zero configuration. □□

Install

```
pnpm add -D @push.rocks/smartmongo
# or
npm install @push.rocks/smartmongo --save-dev
```

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

What It Does

`@push.rocks/smartmongo` wraps mongodb-memory-server to give you a **real MongoDB replica set** running entirely in memory. It downloads and manages the MongoDB binary for you — just `createAndStart()` and you're good to go.

Perfect for:

- ⚡ **Integration tests** that need 100% MongoDB compatibility
- □□ **CI/CD pipelines** where you can't install MongoDB system-wide
- □□ **Development environments** that need a disposable database
- □□ **Data dump/export** with built-in support for dumping collections to disk

> **Looking for a lightweight, pure-TypeScript alternative?** Check out `@push.rocks/smartdb` — a wire-protocol-compatible MongoDB server with zero binary dependencies, instant startup, and file-based persistence.

Quick Start

```
import { SmartMongo } from '@push.rocks/smartmongo';

// Start a MongoDB replica set (downloads binary automatically on first run)
const mongo = await SmartMongo.createAndStart();

// Get connection details for your app or ORM
const descriptor = await mongo.getMongoDescriptor();
console.log(descriptor.mongoDbUrl);
// => mongodb://127.0.0.1:xxxxx/?replicaSet=testset

// Use with any MongoDB client
import { MongoClient } from 'mongodb';
const client = new MongoClient(descriptor.mongoDbUrl);
await client.connect();

const db = client.db(descriptor.mongoDbName);
await db.collection('users').insertOne({ name: 'Alice', role: 'admin' });

const user = await db.collection('users').findOne({ name: 'Alice' });
console.log(user); // { _id: ObjectId(...), name: 'Alice', role: 'admin' }

// Clean up
await client.close();
await mongo.stop();
```

API

`SmartMongo.createAndStart(replCount?: number)`

Static factory method that creates and starts a SmartMongo instance.

```
// Single replica (default)
const mongo = await SmartMongo.createAndStart();

// Multi-replica for testing replication scenarios
const mongo = await SmartMongo.createAndStart(3);
```

`getMongoDescriptor()`

Returns an `IMongoDescriptor` with the connection URL and database name, compatible with `@push.rocks/smartdata` and other push.rocks modules.

```
const descriptor = await mongo.getMongoDescriptor();
// {
//   mongoDbName: 'smartmongo_testdatabase',
//   mongoDbUrl: 'mongodb://127.0.0.1:xxxxx/?replicaSet=testset'
// }
```

`stop()`

Stops the replica set and cleans up all resources (temporary files, processes).

```
await mongo.stop();
```

`stopAndDumpToDir(dir, nameFunction?, emptyDir?)`

Stops the replica set **and** dumps all collections to a directory on disk before cleanup. Useful for debugging or archiving test data.

```
// Dump all collections with default naming
await mongo.stopAndDumpToDir('./test-output');
```

```
// With custom file naming
await mongo.stopAndDumpToDir('./test-output', (doc) => `${doc.collection}-${doc._id}.bson`);

// Keep existing files in the directory (don't empty it first)
await mongo.stopAndDumpToDir('./test-output', undefined, false);
```

`readyPromise`

A promise that resolves when the replica set is fully started and ready to accept connections.

```
const mongo = new SmartMongo();
mongo.start(2); // non-blocking
await mongo.readyPromise; // wait for startup
```

Testing Examples

With @git.zone/tstest (tapbundle)

```
import { expect, tap } from '@git.zone/tstest/tapbundle';
import { SmartMongo } from '@push.rocks/smartmongo';
import { MongoClient } from 'mongodb';

let mongo: SmartMongo;
let client: MongoClient;

tap.test('setup', async () => {
  mongo = await SmartMongo.createAndStart();
  const { mongoDbUrl, mongoDbName } = await mongo.getMongoDescriptor();
  client = new MongoClient(mongoDbUrl);
  await client.connect();
});

tap.test('should insert and query documents', async () => {
  const col = client.db('test').collection('items');
  await col.insertOne({ name: 'Widget', price: 9.99 });
```

```
  const item = await col.findOne({ name: 'Widget' });
  expect(item?.price).toEqual(9.99);
});

tap.test('teardown', async () => {
  await client.close();
  await mongo.stop();
});

export default tap.start();
```

With @push.rocks/smartdata

```
import { SmartMongo } from '@push.rocks/smartmongo';
import { SmartdataDb } from '@push.rocks/smartdata';

const mongo = await SmartMongo.createAndStart();
const descriptor = await mongo.getMongoDescriptor();

const db = new SmartdataDb(descriptor);
await db.init();

// Use smartdata models against the memory server...

await db.close();
await mongo.stop();
```

License and Legal Information

This repository contains open-source code licensed under the MIT License. A copy of the license can be found in the LICENSE file.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH or third parties, and are not included within the scope of the MIT license granted herein.

Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

Company Information

Task Venture Capital GmbH Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

changelog.md for @push.rocks/smartmongo

2026-03-26 - 7.0.0 - BREAKING CHANGE(api)

reduce the package to the SmartMongo memory replica set API and remove bundled TsmDB and LocalTsmDb modules

- remove exports and source code for TsmDB, LocalTsmDb, and related utilities, handlers, storage adapters, and tests
- flatten SmartMongo into the main package entrypoint and keep mongodb-memory-server-based replica set startup as the core feature
- trim dependencies and documentation to focus the package on in-memory MongoDB replica sets for testing and development

2026-03-26 - 5.1.1 - fix(build)

migrate smartconfig metadata and refresh build dependencies

- replace npmextra.json with .smartconfig.json and update packaged config file inclusion
- switch the build script to tsbuild tsfolders
- update core build and runtime dependencies and add explicit TypeScript type annotations to satisfy newer tooling

2026-02-03 - 5.1.0 - feat(localtsmdb)

export ILocalTsmDbConnectionInfo and expand LocalTsmDb/TsmDB documentation and examples

- Exported new type ILocalTsmDbConnectionInfo from ts_local (ts/index.ts)
- Added LocalTsmDb configuration example, methods table, and ConnectionInfo interface to README
- Documented Unix socket vs TCP connection modes and updated usage examples (TCP and socket examples)
- Expanded TsmDB docs: additional server properties, aggregation stages, regex examples, index operations, database ops, checksums, and wire protocol commands
- Updated architecture notes to include Unix socket support and new engine components (QueryEngine, UpdateEngine, AggregationEngine)

2026-02-03 - 5.0.0 - BREAKING CHANGE(localtsmdb)

add Unix socket support and change LocalTsmDb API to return connection info instead of a MongoClient

- LocalTsmDb.start() now returns ILocalTsmDbConnectionInfo { socketPath, connectionUri } instead of a connected MongoClient
- Removed internal MongoClient management: consumers must create/connect/close their own MongoClient using the returned connectionUri (close client before calling db.stop())
- Added ILocalTsmDbConnectionInfo type and getConnectionInfo() (replaces getClient())
- TsmdbServer: added socketPath option to listen on Unix sockets, cleans up stale socket files on start/stop, and encodes socket paths in getConnectionUri()
- LocalTsmDb can auto-generate socket paths in the OS temp dir; LocalTsmDb no longer depends on the mongodb package internally (lightweight Unix socket wrapper)
- Updated docs and tests to use MongoClient externally and to demonstrate socketPath/connectionUri workflow
- ts_local plugins no longer export net (net usage moved to server implementation)

2026-02-03 - 4.3.0 - feat(docs)

add LocalTsmDb documentation and examples; update README code samples and imports; correct examples and variable names; update package author

- Introduce LocalTsmDb: zero-config local database with automatic persistence, auto port discovery, and pre-connected client (added Quick Start, API, Features, and testing examples).
- Expand comparison table to include LocalTsmDb alongside SmartMongo and TsmDB.

- Update README examples: new LocalTsmDb usage, reorder options (LocalTsmDb, TsmDB, SmartMongo), rename test DB variable (db -> testDb), and adjust test snippets for Jest/Mocha and tap.
- Adjust code snippets and API notes: switch some example imports to use tsmdb, replace FileStorageAdapter references, change planner.createPlan to await planner.plan, and use wal.getEntriesAfter(...) without awaiting.
- Update package.json author from 'Lossless GmbH' to 'Task Venture Capital GmbH'.

2026-02-03 - 4.2.1 - fix(package.json)

replace main and typings with exports field pointing to ./dist_ts/index.js

- Added package.json exports field mapping "." to ./dist_ts/index.js to declare the package entrypoint.
- Removed main (dist_ts/index.js) and typings (dist_ts/index.d.ts) entries.
- Note: switching to exports improves Node resolution but removing the typings entry may affect TypeScript consumers expecting index.d.ts.

2026-02-01 - 4.2.0 - feat(tsmdb)

implement TsmDB Mongo-wire-compatible server, add storage/engine modules and reorganize exports

- Add full TsmDB implementation under ts/ts_tsmdb: wire protocol, server, command router, handlers, engines (Query, Update, Aggregation, Index, Transaction, Session), storage adapters (Memory, File), OpLog, WAL, utils and types.
- Remove legacy ts/tsmdb implementation and replace with new ts_tsmdb module exports.
- Introduce ts/ts_mongotools module and move SmartMongo class there; update top-level exports in ts/index.ts to export SmartMongo, tsmdb (from ts_tsmdb) and LocalTsmDb.
- Add LocalTsmDb convenience class (ts/ts_local) to start a file-backed TsmDB and return a connected MongoClient.
- Refactor plugin imports into per-module plugins files and add utilities (checksum, persistence, query planner, index engine).

2026-02-01 - 4.1.1 - fix(tsmdb)

add comprehensive unit tests for tsmdb components: checksum, query planner, index engine, session, and WAL

- Add new tests: test.tsmdb.checksum.ts — CRC32 and document checksum utilities (add/verify/remove)
- Add new tests: test.tsmdb.queryplanner.ts — QueryPlanner plans, index usage, selectivity, explain output, and edge cases
- Add new tests: test.tsmdb.indexengine.ts — Index creation, unique/sparse options, candidate selection, and constraints
- Add new tests: test.tsmdb.session.ts — Session lifecycle, touch/refresh/close, extractSessionId handling
- Add new tests: test.tsmdb.wal.ts — WAL initialization, LSN increments, logging/recovery for inserts/updates/deletes, binary and nested data handling
- Tests only — no production API changes; increases test coverage
- Recommend patch bump from 4.1.0 to 4.1.1

2026-02-01 - 4.1.0 - feat(readme)

expand README with storage integrity, WAL, query planner, session & transaction docs; update test script to enable verbose logging and increase timeout

- Updated npm test script to run tstest with --verbose, --logfile and --timeout 60 to improve test output and avoid timeouts.
- Extensive README additions: file storage adapter examples with checksum options, write-ahead logging (WAL) usage and recovery, query planner examples, index and query execution details, session and transaction examples and features.
- Wire protocol / features table updated to include Transactions and Sessions and added admin commands (dbStats, collStats).
- Architecture diagram and component list updated to include QueryPlanner, SessionEngine, TransactionEngine and WAL; storage layer annotated with checksums and WAL.
- Minor example import tweak: MongoClient import now includes Db type in test examples.

2026-02-01 - 4.0.0 - BREAKING CHANGE(storage,engine,server)

add session & transaction management, index/query planner, WAL and checksum support; integrate index-accelerated queries and update storage API (findByIds) to enable index optimizations

- Add SessionEngine with session lifecycle, auto-abort of transactions on expiry and session tracking in CommandRouter and AdminHandler.
- Introduce TransactionEngine integrations in CommandRouter and AdminHandler; handlers now support start/commit/abort transaction workflows.
- Add IndexEngine enhancements including a simple B-tree and hash map optimizations; integrate index usage into Find/Count/Insert/Update/Delete handlers for index-accelerated queries and index maintenance on mutations.
- Add QueryPlanner to choose IXSCAN vs COLLSCAN and provide explain plans.
- Add WAL (write-ahead log) for durability, with LSNs, checkpoints and recovery APIs.
- Add checksum utilities and FileStorageAdapter support for checksums (enableChecksums/strictChecksums), with verification on read and optional strict failure behavior.
- IStorageAdapter interface changed to include findByIds; MemoryStorageAdapter and FileStorageAdapter implement findByIds to support index lookups.
- Exported API additions: WAL, QueryPlanner, SessionEngine, checksum utilities; CommandRouter now caches IndexEngines and exposes transaction/session engines.
- Breaking change: the IStorageAdapter interface change requires third-party storage adapters to implement the new findByIds method.

2026-02-01 - 3.0.0 - BREAKING CHANGE(tsmdb)

rename CongoDB to TsmDB and relocate/rename wire-protocol server implementation and public exports

- Project refactor renames the in-memory wire-protocol server from CongoDB -> TsmDB (identifiers, files and namespaces changed).
- ts/index.ts now exports tsmdb instead of congodb (public API change; consumers must update imports).
- All congodb sources under ts/congodb were removed and equivalent implementations added under ts/tsmdb (errors, engines, storage adapters, server, handlers, WireProtocol, types).
- Readme and usage examples updated to reference TsmDB/tsmdb and example code updated accordingly.
- Tests renamed/updated from test.congodb.ts -> test.tsmdb.ts to exercise the new tsmdb export and server.

2026-01-31 - 2.2.0 - feat(readme)

update README with expanded documentation covering CongoDB and SmartMongo, installation, quick start examples, architecture, usage examples, and legal/company information

- Completely expanded README: added detailed overview for SmartMongo and new CongoDB (wire-protocol server)
- Added Quick Start examples for both SmartMongo and CongoDB (TypeScript/ESM snippets)
- Included installation instructions for npm and pnpm and issue reporting/security guidance
- Added architecture diagram, example tests, and storage/engine descriptions
- Clarified license, trademark, and company contact information
- Large non-functional documentation-only change (+398 -44)

2026-01-31 - 2.1.0 - feat(congodb)

implement CongoDB MongoDB wire-protocol compatible in-memory server and APIs

- Add full congodb module: CongoServer, WireProtocol, CommandRouter and handlers (Hello, Insert, Find, Update, Delete, Aggregate, Index, Admin).
- Implement query/update/aggregation/index/transaction engines (QueryEngine, UpdateEngine, AggregationEngine, IndexEngine, TransactionEngine) and OpLog for change stream support.
- Add storage adapters: in-memory (MemoryStorageAdapter) and file-backed (FileStorageAdapter) with persistence and oplog support.
- Introduce types/interfaces and rich error classes (CongoErrors) plus congodb.plugins re-exports (bson, mingo, smartfs, smartpath, smartrx).
- Add many server-side utilities: IndexEngine, Aggregation helpers ($lookup, $graphLookup, $merge, $facet, $unionWith), cursor management and command routing.
- Add integration tests for CongoDB using official mongodb MongoClient (test/test.congodb.ts) and update unit test entry (test/test.ts) to use tstest tapbundle.
- Export congodb from ts/index.ts and update package.json: bump devDependencies, add runtime deps (mongodb, bson, mingo, mingo), add new @push.rocks/* deps and dev tool versions.
- Update readme.hints.md with CongoDB architecture, usage examples and supported commands.
- Update npmextra.json metadata and release/registry config and reorganize tsdoc mappings.

2025-11-17 - 2.0.14 - fix(smartmongo.plugins)

Use default import for mongodb-memory-server (Deno compatibility), update hints and bump package version to 2.0.13

- Replace namespace import with default import for mongodb-memory-server to ensure compatibility with Deno (ts/smartmongo.plugins.ts).
- Add readme.hints.md documenting the Deno compatibility change and the reason for using a default import.
- Bump package.json version to 2.0.13.
- Note: ts/00_commitinfo_data.ts still lists version 2.0.12 and may need to be updated to match package.json.

2025-04-06 - 2.0.12 - fix(ci/config)

Update CI workflow environment variables, refine package metadata, and improve configuration settings

- Updated workflow YAML files to use new IMAGE and npmci package names
- Adjusted package.json homepage, added bugs field and pnpm overrides
- Minor formatting improvements in readme.md and .gitignore
- Enhanced tsconfig with baseUrl and paths for improved module resolution

2025-04-06 - 2.0.11 - fix(dependencies)

Update dependency names and versions in CI workflows and package configuration

- Rename devDependency packages from '@gitzone/' *to* '*@git.zone/*' for consistency
- Bump '@types/node' from '^20.4.8' to '^22.14.0'
- Upgrade 'mongodb-memory-server' from '^8.14.0' to '^10.1.4'
- Add 'packageManager' field in package.json
- Introduce pnpm-workspace.yaml with 'onlyBuiltDependencies' configuration

2024-05-29 - 2.0.10 - misc

Various updates to project configuration and documentation.

- update description

- update tsconfig
- update npmextra.json: githost (applied on three occasions)

2023-08-08 - 2.0.9 - core

Core fix.

- fix(core): update

2023-08-08 - 2.0.8 - core

Core fix.

- fix(core): update

2023-08-08 - 2.0.7 - core & org

Combined changes for core stability and organization improvements.

- fix(core): update
- switch to new org scheme (recorded twice)

2022-06-08 - 2.0.6 - core

Core fix.

- fix(core): update

2022-06-08 - 2.0.5 - core

Core fix.

- fix(core): update

2022-06-06 - 2.0.4 - core

Core fix.

- fix(core): update

2022-06-06 - 2.0.3 - core

Core fix.

- fix(core): update

2022-06-03 - 2.0.2 - core

Core fix.

- fix(core): update

2022-05-19 - 2.0.1 - core

Core fix.

- fix(core): update

2022-05-18 - 2.0.0 - core

Core fix.

- fix(core): update

2022-05-17 - 1.0.9 - core

Breaking change for module format.

- BREAKING CHANGE(core): switch to esm

2022-05-17 - 1.0.8 - core

Core fix.

- fix(core): update

2021-12-21 - 1.0.7 - core

Core fix.

- fix(core): update

2021-12-20 - 1.0.6 - core

Core fix.

- fix(core): update

2021-12-20 - 1.0.5 - core

Core fix.

- fix(core): update

2021-12-20 - 1.0.4 - core

Core fix.

- fix(core): update

2021-12-20 - 1.0.3 - core

Core fix.

- fix(core): update

2021-12-20 - 1.0.2 - no notable changes

These version bumps did not include additional modifications.

- version update only

2021-12-20 - 1.0.1 - core

Core fix.

- fix(core): update