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.

Revision #3
Created 2026-03-28 11:11:33 UTC by foss.global Team
Updated 2026-03-28 12:18:24 UTC by foss.global Team