







# readme.md for @push.rocks/mongodump

A powerful MongoDB backup and restore tool 

[npm](#) [version](#) [license](#) [TypeScript](#) [ES](#) [Module](#)

## What it does

`@push.rocks/mongodump` is your go-to solution for creating and managing MongoDB database dumps. Whether you're backing up critical production data, migrating between environments, or implementing disaster recovery strategies, this tool has got your back. It provides a clean, TypeScript-based API for:

-  **Full database backups** - Export entire MongoDB databases
-  **Collection-level dumps** - Selectively backup specific collections
-  **Custom naming** - Transform document names during export
-  **Archive support** - Create compressed tar archives of your dumps
-  **Async/await patterns** - Modern promise-based workflow
-  **TypeScript first** - Full type safety and IntelliSense support

## Installation

```
# Using npm
npm install @push.rocks/mongodump --save

# Using pnpm (recommended)
pnpm add @push.rocks/mongodump

# Using yarn
yarn add @push.rocks/mongodump
```

# Quick Start ☐☐

Get up and running in seconds:

```
import { MongoDump } from '@push.rocks/mongodump';
import type { IMongoDescriptor } from '@tsclass/tsclass';

// Define your MongoDB connection
const mongoDescriptor: IMongoDescriptor = {
  mongoDbName: 'my-database',
  mongoDbUser: 'admin',
  mongoDbPass: 'secure-password',
  mongoDbUrl: 'mongodb+srv://cluster.mongodb.net',
};

// Create a dump of your entire database
const mongoDump = new MongoDump();
const target = await mongoDump.addMongoTargetByMongoDescriptor(mongoDescriptor);
await target.dumpAllCollectionsToDir('./backups/my-backup');

// Done! Your backup is ready ☐☐
await mongoDump.stop();
```

## Detailed Usage ☐☐

### Setting Up MongoDB Connection

The module uses MongoDB descriptors to establish connections. This approach provides flexibility and security:

```
import { IMongoDescriptor } from '@tsclass/tsclass';
import { MongoDump, MongoDumpTarget } from '@push.rocks/mongodump';

const mongoDescriptor: IMongoDescriptor = {
  mongoDbName: 'production_db',
  mongoDbUser: process.env.MONGO_USER,
```

```
mongoDbPass: process.env.MONGO_PASS,  
mongoDbUrl: process.env.MONGO_URL,  
};
```

# Working with MongoDump

The `MongoDump` class is your main entry point for managing database dumps:

```
const mongoDump = new MongoDump();  
  
// Add a MongoDB target  
const dumpTarget = await mongoDump.addMongoTargetByMongoDescriptor(mongoDescriptor);  
  
// The target is now ready for dump operations
```

## Dumping Collections

### Dump All Collections to Directory

Perfect for complete database backups:

```
// Basic dump - uses document _id as filename  
await dumpTarget.dumpAllCollectionsToDir('./backups/full-backup');  
  
// Custom naming - use any document field for filenames  
await dumpTarget.dumpAllCollectionsToDir(  
  './backups/named-backup',  
  (doc) => `${doc.username}_${doc.timestamp}`,  
  true // Clean directory before dumping  
);
```

### Dump Specific Collection

When you need granular control:

```
// Get available collections  
const collections = await dumpTarget.getCollections();  
console.log('Available collections:', collections);
```

```
// Dump a specific collection
const userCollection = collections.find(c => c.collectionName === 'users');
await dumpTarget.dumpCollectionToDir(
  userCollection,
  './backups/users',
  (doc) => doc.email.replace('@', '_at_') // Custom naming
);
```

## Archive Support

Create compressed archives for efficient storage and transfer:

```
// Dump to tar archive file
await dumpTarget.dumpCollectionToTarArchiveFile(
  userCollection,
  './backups/users.tar.gz'
);

// Or work with streams for advanced scenarios
const archiveStream = await dumpTarget.dumpAllCollectionsToTarArchiveStream();
// Pipe to S3, network storage, etc.
```

## Advanced Patterns

### Scheduled Backups

Implement automated backup strategies:

```
import { CronJob } from 'cron';

const backupJob = new CronJob('0 0 * * *', async () => {
  const timestamp = new Date().toISOString().split('T')[0];
  const mongoDump = new MongoDump();
  const target = await mongoDump.addMongoTargetByMongoDescriptor(mongoDescriptor);

  await target.dumpAllCollectionsToDir(
    `./backups/daily/${timestamp}`,
```

```
    null,  
    true  
  );  
  
  await mongoDump.stop();  
  console.log(` Daily backup completed: ${timestamp}`);  
});  
  
backupJob.start();
```

## Multiple Database Targets

Handle multiple databases simultaneously:

```
const mongoDump = new MongoDump();  
  
// Add multiple targets  
const prodTarget = await mongoDump.addMongoTargetByMongoDescriptor(prodDescriptor);  
const stagingTarget = await mongoDump.addMongoTargetByMongoDescriptor(stagingDescriptor);  
  
// Dump both in parallel  
await Promise.all([  
  prodTarget.dumpAllCollectionsToDir('./backups/production'),  
  stagingTarget.dumpAllCollectionsToDir('./backups/staging')  
]);  
  
await mongoDump.stop();
```

## Error Handling

Implement robust error handling:

```
try {  
  const mongoDump = new MongoDump();  
  const target = await mongoDump.addMongoTargetByMongoDescriptor(mongoDescriptor);  
  
  await target.dumpAllCollectionsToDir('./backups/safe-backup');  
  console.log(' Backup successful');  
  
} catch (error) {
```

```
console.error(' Backup failed:', error);
// Implement notification/retry logic

} finally {
  await mongoDump.stop();
}
```

## Clean Shutdown

Always ensure proper cleanup of database connections:

```
// Closes all open MongoDB connections
await mongoDump.stop();
```

# API Reference

## MongoDump

The main class for managing dump operations.

### Methods:

- `addMongoTargetByMongoDescriptor(descriptor: IMongoDescriptor)`: Create a new dump target
- `stop()`: Close all database connections

## MongoDumpTarget

Handles operations for a specific MongoDB database.

### Methods:

- `getCollections()`: List all collections in the database
- `dumpCollectionToDir(collection, directory, nameTransform?)`: Dump single collection
- `dumpAllCollectionsToDir(directory, nameTransform?, cleanDir?)`: Dump all collections
- `dumpCollectionToTarArchiveFile(collection, filepath)`: Create tar archive
- `dumpAllCollectionsToTarArchiveStream()`: Get tar archive stream

# Best Practices

1. **Environment Variables:** Store credentials in environment variables, never hardcode them
2. **Error Handling:** Always wrap dump operations in try-catch blocks
3. **Connection Cleanup:** Always call `stop()` when done
4. **Backup Verification:** Periodically verify your backups can be restored
5. **Storage Rotation:** Implement backup rotation to manage disk space
6. **Monitoring:** Add logging and monitoring for production backup jobs

## Why Choose

### @push.rocks/mongodump?

- **Type Safety:** Full TypeScript support with comprehensive type definitions
- **Modern Async:** Built on promises and async/await patterns
- **Flexible Export:** Multiple export formats and naming options
- **Production Ready:** Battle-tested in production environments
- **Clean API:** Intuitive, well-documented interface
- **Active Maintenance:** Regular updates and security patches

## License and Legal Information

This repository contains open-source code that is licensed under the MIT License. A copy of the MIT License can be found in the [license](#) file within this repository.

**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 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, and any usage must be approved in writing by Task Venture Capital GmbH.

# Company Information

Task Venture Capital GmbH

Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at [hello@task.vc](mailto: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 11:09:56 UTC by foss.global Team

Updated 2026-03-28 12:16:33 UTC by foss.global Team