Mega MongoDB Database Document

1. Summary

1.1 Document Purpose

This document lists all MongoDB Atlas databases used in Mega’s web applications and explains their purpose, connections, and basic management.

Objectives:

- Provide clear mapping between services and their respective databases
- Enable rapid onboarding of new team members
- Document operational procedures and best practices

Intended Audience:

- Full-Stack Developers
- Technical Leads and Architects
- Database Administrators

1.2 Infrastructure Overview

Mega uses multiple MongoDB projects, with one cluster per project, to keep different business areas separate and easy to manage.

Below is the **infrastructure hierarchy of the Mega We Care website databases**, providing an overview of which websites use which clusters and where their database data is stored.

Infrastructure Hierarchy:

```
Mega's Organization (MongoDB Atlas)
│
├── Project 0 (Corporate + Countries)
│   └── Cluster: megawecare
│       └── Databases for corporate and countries websites
│
├── Mega Micro Sites
│   └── Cluster: MicrositesCluster
│       └── Databases for brands websites
│
├── Mother We Care
│   └── Cluster: motherwecare
│       └── Database for motherwecare website
│
├── Wellness We Care
│   └── Cluster: wellnesswecare
│       └── Database for wellnesswecare website
│
├── Womens We Care
│   └── Cluster: womenswecare
│       └── Database for womenswecare website
```

1.3 Quick Reference Matrix

Project Name	Cluster Name	Cluster Config
Project 0 (Corporate + Country)	megawecare	M20
Mega Micro Sites	MicrositesCluster	**Flex (Shared)**
Mother We Care	motherwecare	Free
Wellness We Care	wellnesswecare	Free
Womens We Care	womenswecare	M10

2. Architecture Overview

2.1 Architecture Principles

Mega’s MongoDB setup uses project-based isolation with one cluster per project. Databases within each cluster are organized based on application or service usage, and cluster sizes are chosen based on workload and future growth.

2.2 Services-to-Database Interaction (High-Level)

At a high level, services interact with MongoDB as follows:

```
Service → Cluster → Target Database → Collections
```

---

3. Infrastructure Inventory

3.1 Complete Database Inventory

Project	Cluster	Database	Purpose
Project 0	megawecare	mega-we-care-corporate	Global corporate website
Project 0	megawecare	mega-we-care-cambodia	Cambodia country website
Project 0	megawecare	mega-we-care-ghana	Ghana country website
Project 0	megawecare	mega-we-care-peru	Peru country website
Project 0	megawecare	mega-we-care-myanmar	Myanmar country website
Project 0	megawecare	mega-we-care-philippines	Philippines country website
Project 0	megawecare	mega-we-care-srilanka	Sri Lanka country website
Project 0	megawecare	mega-we-care-uzbek	Uzbekistan country website
Mega Micro Sites	MicrositesCluster	eugica	Eugica website

Project	Cluster	Database	Purpose
Mega Micro Sites	MicrositesCluster	herbal-meds	Herbal-Meds website
Mega Micro Sites	MicrositesCluster	probiotics	Probiotics website
Mother We Care	motherwecare	mother-we-care-corporate	Mother We Care website
Wellness We Care	wellnesswecare	wellness-we-care-corporate	Wellness We Care website
Womens We Care	womenswecare	womens-we-care-corporate	Womens We Care website

4. Project Details

4.1 Project 0 - Corporate Websites

Project Overview:

Attribute	Value
Project Name	Project 0
Description	Contains databases for corporate global website and country-specific websites
Cluster Name	megawecare
Cluster Status	Operational - No Active Alerts
Cluster Tier	M20

Connection String Template:

```
mongodb+srv://mega:<db_password>@megawecare.vpshqje.mongodb.net/<database_name>?authSource=admin
```

4.2 Mega Micro Sites

Project Overview:

Attribute	Value
Project Name	Mega Micro Sites

Attribute	Value
Description	Databases for brand-specific websites
Cluster Name	MicrositesCluster
Cluster Tier	Flex (Shared)
Cluster Status	Operational (No active alerts)

Connection String Template:

```
mongodb+srv://admin:<db_password>@micrositescluster.ha2m7.mongodb.net/<database_name>?authSource=admin
```

4.3 Mother We Care

Project Overview:

Attribute	Value
Project Name	Mother We Care
Description	Database for mother we care website
Cluster Name	motherwecare
Cluster Tier	Free (M0)
Cluster Status	Operational - No Active Alerts
Database Name	mother-we-care-corporate

Connection String Template:

```
mongodb+srv://<username>:<db_password>@motherwecare.clxtqhn.mongodb.net/<database_name>?authSource=admin
```

4.4 Wellness We Care

Project Overview:

Attribute	Value
Project Name	Wellness We Care
Description	Database for wellness we care website

Attribute	Value
Cluster Name	wellnesswecare
Cluster Tier	Free (M0)
Cluster Status	Operational - No Active Alerts
Database Name	wellness-we-care-corporate

Connection String Template:

```
mongodb+srv://admin:<db_password>@wellnesswecare.0cqgt.mongodb.net/<database_name>?authSource=admin
```

4.5 Womens We Care

Project Overview:

Attribute	Value
Project Name	Womens We Care
Description	Database for womens we care website
Cluster Name	womenswecare
Cluster Tier	M10
Cluster Status	Operational - No Active Alerts
Database Name	womens-we-care-corporate

Connection String Template:

```
mongodb+srv://admin:<db_password>@womenswecare.myxwocu.mongodb.net/<database_name>?authSource=admin
```

---

5. Connection & Configuration

5.1 Connection String Standards

Standard MongoDB Atlas Connection String Format:

```
Production Example:
mongodb+srv://<username>:<password>@<cluster-hostname>/<database>?<options>
```

```
Local example:
mongodb://<username>:<password>@<cluster-hostname>/<database>?<options>
```

Connection String Components:

Component	Description	Example	Notes
`mongodb+srv://`	Protocol	mongodb+srv://	Always use SRV for Atlas (handles DNS resolution)
`mongodb://`	Protocol for local		used for local db connections
`<username>`	Database user	`api_user`, `admin`	Must exist in the target project
`<password>`	User password	`secureP@ss123`	URL-encode special characters
`<cluster-hostname>`	Cluster endpoint	`megawecare.vpshqje.mongodb.net`	Unique per cluster
`<database>`	Default database	`mega-we-care-corporate`	Optional; can be set in application code
`<options>`	Query parameters	`authSource=admin`	gives the admin access to crud operations

5.2 Environment-Specific Configuration

Multi-Environment Strategy:

Environment	Purpose	
Development	Local development, testing	mongodb://:@/?
Production	Live production database	mongodb+srv://:@/?

6. Developer Guide

Prerequisites:

1. **MongoDB Driver:** Install the appropriate driver for your language
2. **Network Access:** Confirm your IP or VPN is whitelisted
3. **Credentials:** Obtain database credentials from your team lead or secret management system
4. **Documentation:** Review this document and the specific project section relevant to your service

Checklist to follow:

- [ ] Install MongoDB driver in your project
- [ ] Obtain connection string for your environment (dev/staging/prod)
- [ ] Store connection string in environment variables, not in code
- [ ] Test connections
- [ ] Install mongodb compass
- [ ] Install cli tools for running mongodb commands
- [ ] Import connections in mongodb compass

Example mongodb compass connections with color codings:

each connections have specified color coding for better understandings and clear differences

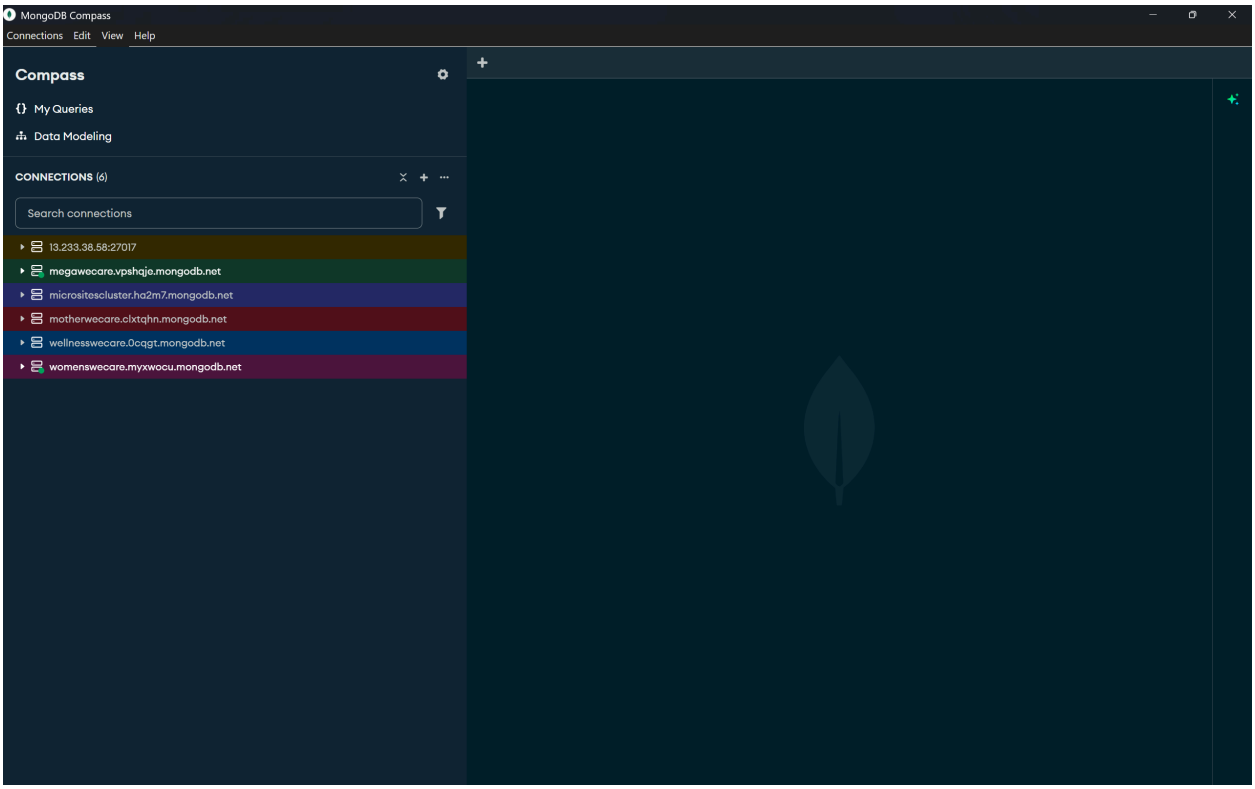

image.png

7. Operational Procedures

7.1 Monitoring & Alerting

- **Cluster Storage Monitoring**
  - Monitor cluster storage growth regularly.
  - If storage usage increases by **more than 500 MB** or approaches allocated limits, **scale the cluster** to avoid performance degradation (e.g., slow queries, increased I/O latency).
- **Deployment Flexibility**
  - Databases are primarily hosted on **MongoDB Atlas**.
  - If Atlas is not used, MongoDB clusters can also be provisioned and managed directly on **AWS** with appropriate backup, monitoring, and security configurations.

7.2 Backup & Recovery

MongoDB Atlas provides **automated backups** for supported clusters, including **Point-in-Time Recovery (PITR)** and **scheduled snapshots** based on the configured backup policy.

Backup retention is **strictly policy-based**:

- Data can be restored **only within the configured PITR window**
- Snapshots are retained **only for the defined retention period**
- Expired backups are **permanently deleted** and cannot be recovered

MongoDB Atlas does **not provide unlimited historical backups** by default. Long-term or compliance-driven retention requires an **explicit Backup Compliance Policy**.

Automated backups and PITR are supported only on M10 and higher clusters.

Shared clusters (M0, M2, M5) offer limited snapshot backups and are **not suitable for production workloads**.

Manual Backup & Restore

The following commands are used for **manual backup and restore operations** during operational or recovery scenarios.

Always verify the **target cluster, database name, and environment** before execution.

---

0. Configure your terminal (one-time setup)

Before running any command below, you must have **MongoDB Database Tools** installed so that `mongodump` and `mongorestore` are available in your terminal.

Step 0.1 — Install MongoDB Database Tools on your machine

- **Official guide (all platforms):** Install MongoDB Database Tools
- Follow the instructions for your OS (Windows, macOS, or Linux) so you can run `mongodump` and `mongorestore` from the command line.

Step 0.2 — Verify installation

Run in your terminal (copy and paste as a single line):

```
mongodump --version
```

If you see a version number, you are ready to use the backup and restore commands below.

---

1. Backup a Single Database

Use case: Create a compressed backup of one database (e.g., before a risky change or for moving data to another environment). The database stays online and unchanged.

Step 1.1 — Replace placeholders in the command with your values: `<USERNAME>`, `<PASSWORD>`, `<HOST>`, `<PORT>`, `<SOURCE_DB>`, `<BACKUP_PATH>`.

Step 1.2 — Run this command as a single line (copy the whole line and paste into your terminal):

```
mongodump --uri="mongodb://<USERNAME>:<PASSWORD>@<HOST>:<PORT
>/<SOURCE_DB>?authSource=admin" --archive="<BACKUP_PATH>/<SOU
RCE_DB>.gzip" --gzip
```

What This Does

Creates a compressed backup of a single database and saves it as one file. The original database remains completely unchanged and available during the backup process.

Parameters Explained

- `uri` — The complete connection string to your MongoDB server
- Format: `mongodb://username:password@host:port/database?authSource=admin`
- `<USERNAME>` — Database user with read permissions
- `<PASSWORD>` — User's password (URL-encode special characters like @, #, %)

- `<HOST>` — Server hostname or IP address (e.g., `localhost`, `192.168.1.100`)
- `<PORT>` — MongoDB port (default is `27017`)
- `<SOURCE_DB>` — The specific database you want to backup
- `?authSource=admin` — Tells MongoDB which database contains the user credentials
- `archive` — Where to save the backup
- `<BACKUP_PATH>` — Full directory path (e.g., `/backups/mongodb/`)
- `<SOURCE_DB>.gzip` — Filename for the backup (include `.gzip` extension)
- Example: `/backups/mongodb/production_db.gzip`
- `gzip` — Compresses the backup file

---

2. Restore Database with Rename

Use case: Restore from a backup file and give the database a new name (e.g., copy production into a staging or dev database on the same server).

Step 2.1 — Replace placeholders: `<USERNAME>`, `<PASSWORD>`, `<HOST>`, `<PORT>`, `<SOURCE_DB>` (name inside the backup), `<TARGET_DB>` (new name on the server), `<BACKUP_PATH>`.

Step 2.2 — Run this command as a single line:

```
mongorestore --noIndexRestore --uri="mongodb://<USERNAME>:<PASSWORD>@<HOST>:<PORT>/?authSource=admin" --nsFrom="<SOURCE_DB>.*" --nsTo="<TARGET_DB>.*" --gzip --archive="<BACKUP_PATH>/<SOURCE_DB>.gzip"
```

What This Does

Restores a database from a backup file and gives it a new name in the process. Perfect for creating copies of production databases in staging or development environments.

Parameters Explained

- `noIndexRestore` — Skip restoring indexes
- Restores only the data and collections, not the indexes
- `uri` — Connection to the target MongoDB server
- Format: `mongodb://username:password@host:port/?authSource=admin`
- `nsFrom` — Source namespace (from the backup file)
- Format: `<SOURCE_DB>.*`
- `<SOURCE_DB>` — The original database name stored in the backup
- `.*` — Wildcard meaning “all collections” in that database
- `nsTo` — Target namespace (where to restore)
- Format: `<TARGET_DB>.*`
- `<TARGET_DB>` — The new database name you want to create
- `.*` — All collections will be restored under this new database name
- `gzip` — Decompress during restore
- `archive` — Path to the backup file
- `<BACKUP_PATH>/<SOURCE_DB>.gzip` — Full path to your backup file
- Example: `/backups/production_db.gzip`

---

3. Restore to MongoDB Atlas

Use case: Restore a backup file into a MongoDB Atlas cluster (e.g., from a local or on-prem backup into Atlas, or to a different Atlas database name).

Step 3.1 — Replace placeholders: `<USERNAME>`, `<PASSWORD>`, `<ATLAS_CLUSTER_URL>` (from Atlas connection string), `<SOURCE_DB>` (name in the backup), `<TARGET_DB>` (database name in Atlas), `<BACKUP_PATH>`.

Step 3.2 — Run this command as a single line:

```
mongorestore --noIndexRestore --uri="mongodb+srv://<USERNAME
>:<PASSWORD>@<ATLAS_CLUSTER_URL>/" --nsFrom="<SOURCE_DB>.*" -
```

```
-nsTo="<TARGET_DB>.*" --gzip --archive="<BACKUP_PATH>/<SOURCE
_DB>.gzip"
```

What This Does

Restores a database backup to a MongoDB Atlas cloud cluster. Atlas is MongoDB's fully managed cloud database service.

Parameters Explained

- `uri` — Atlas connection string
- Format: `mongodb+srv://username:password@cluster-url/`
- `mongodb+srv://` — Special Atlas protocol (different from standard `mongodb://`)
- The `srv` means DNS seedlist connection (Atlas-specific)
- `<USERNAME>:<PASSWORD>` — Atlas database user credentials (created in Database Access)
- `<ATLAS_CLUSTER_URL>` — Your Atlas cluster address
- `nsFrom` **and** `nsTo` — Same as previous section
- `nsFrom="<SOURCE_DB>.*"` — Database name in the backup
- `nsTo="<TARGET_DB>.*"` — Database name in Atlas
- `gzip` **and** `archive` — Same as previous sections

---

4. Cross-Cluster Database Migration

Use case: Move a database from one cluster to another using a backup file (e.g., migrate from on-prem or one Atlas project to a different Atlas cluster). The source cluster is not modified.

Step 4.1 — Replace placeholders: `<USERNAME>`, `<PASSWORD>` for the **destination** cluster, `<TARGET_CLUSTER_URL>` (destination cluster host), `<SOURCE_DB>` (name in the backup), `<TARGET_DB>` (name on destination), `<BACKUP_PATH>`.

Step 4.2 — Run this command as a single line:

```
mongorestore --noIndexRestore --uri="mongodb+srv://<USERNAME
>:<PASSWORD>@<TARGET_CLUSTER_URL>/" --nsFrom="<SOURCE_DB>.*"
--nsTo="<TARGET_DB>.*" --gzip --archive="<BACKUP_PATH>/<SOURC
E_DB>.gzip"
```

What This Does

Migrates a database from one MongoDB cluster to a completely different cluster. The source cluster is never touched or modified—only the backup file is used.

Parameters Explained

- `uri` — Target cluster connection
- `<TARGET_CLUSTER_URL>` — The destination cluster (completely separate from source)
- `<USERNAME>:<PASSWORD>` — Credentials for the destination cluster
- `nsFrom` **and** `nsTo` — Control database naming
- `<SOURCE_DB>` — Database name as it exists in the backup file
- `<TARGET_DB>` — Database name to create on the destination cluster
- `archive` — Local backup file

5. Selective Restore from Multi-Database Backup

Use case: Restore only one database from a backup that contains multiple databases (e.g., you have a full cluster backup but need to restore or copy just one database).

Step 5.1 — Replace placeholders: `<USERNAME>`, `<PASSWORD>`, `<CLUSTER_URL>`, `<SOURCE_DB>` (database to extract from the backup), `<TARGET_DB>` (name for it on the cluster), `<BACKUP_PATH>` (path to the full backup file).

Step 5.2 — Run this command as a single line:

```
mongorestore --noIndexRestore --uri="mongodb+srv://<USERNAME
>:<PASSWORD>@<CLUSTER_URL>/" --nsInclude="<SOURCE_DB>.*" --ns
```

```
From="<SOURCE_DB>.*" --nsTo="<TARGET_DB>.*" --gzip --archive
="<BACKUP_PATH>/full-backup.gzip"
```

What This Does

Extracts and restores only one specific database from a backup file that contains multiple databases. Useful when you have a full cluster backup but only need one database.

Parameters Explained

- `nsInclude` — Filter which databases to restore
- Format: `<SOURCE_DB>.*`
- Only collections matching this pattern will be restored
- `<SOURCE_DB>` — The specific database you want to extract
- `.*` — All collections in that database
- Everything else in the backup is ignored
- `nsFrom` — Source database name (in the backup)
- Must match the `nsInclude` pattern
- Specifies which database to read from the backup
- `nsTo` — Target database name (where to restore)
- Can be the same as source or different
- Allows renaming during selective restore
- Example: `test_db.*` creates `test_db.users`, `test_db.orders`, etc.
- `archive` — Path to the multi-database backup
- `<BACKUP_PATH>/full-backup.gzip` — Your full cluster backup file

---

6. Example commands (Mega context)

The following are real-world examples for Mega databases. Replace `<USERNAME>`, `<PASSWORD>`, `<HOST>`, and `<BACKUP_PATH>` with your credentials and paths. Use these as

reference after reading sections 1–5.

Backup a dev database (e.g., from an EC2/on-prem MongoDB) to a local file:

```
mongodump --uri="mongodb://<USERNAME>:<PASSWORD>@<HOST>:2701
7/womens-we-care-corporate_dev?authSource=admin" --archive="<
BACKUP_PATH>/womens-we-care-corporate.gzip" --gzip
```

Restore that backup on the same server with a different database name (e.g., dev → main):

```
mongorestore --noIndexRestore --uri="mongodb://<USERNAME>:<PA
SSWORD>@<HOST>:27017/?authSource=admin" --nsFrom="womens-we-c
are-corporate_dev.*" --nsTo="womens-we-care-corporate.*" --gz
ip --archive="<BACKUP_PATH>/womens-we-care-corporate.gzip"
```

Restore a backup into MongoDB Atlas (e.g., Womens We Care cluster):

```
mongorestore --noIndexRestore --uri="mongodb+srv://<USERNAME
>:<PASSWORD>@womenswecare.myxwocu.mongodb.net/" --nsFrom="wom
ens-we-care-corporate.*" --nsTo="womens-we-care-corporate.*"
--gzip --archive="<BACKUP_PATH>/megawecarecorporate.gzip"
```

Same database name but different cluster (e.g., restore a country DB from one cluster into another, e.g. Project 0 megawecare):

```
mongorestore --noIndexRestore --uri="mongodb+srv://<USERNAME
>:<PASSWORD>@megawecare.vpshqje.mongodb.net/" --gzip --archiv
e="<BACKUP_PATH>/mega-we-care-philippines.gzip" --nsFrom="meg
awecare.*" --nsTo="mega-we-care-thailand.*"
```

Selective restore: restore only one database from a multi-database backup and give it a new name:

```
mongorestore --noIndexRestore --uri="mongodb+srv://<USERNAME
>:<PASSWORD>@<CLUSTER_URL>/" --nsInclude="womens-we-care-corp
```

```
orate.*" --nsFrom="womens-we-care-corporate.*" --nsTo="mother
-we-care-corporate.*" --gzip --archive="<BACKUP_PATH>/megawec
arecorporate.gzip"
```

Conclusion

This document summarizes MongoDB Atlas architecture and operational practices for Mega and should be updated as changes occur.