Glossary

Colored Contents	Represents
BLACK	Normal DB Activities (Managed Services)
RED	New Implementations (Professional Services)

Support Type	Related Tasks
Full Support	Covers all **BLACK** + **RED** Colored tasks mentioned in this document. **NOTE**: All **RED** colored contents will be executed in Serial manner (one task at a time). If any multiple **RED** colored tasks needed to be executed in parallel will be carried over with additional costs based on the workload proposed.
Normal Support	Covers ONLY BLACK colored tasks mentioned in this document

Equivalent Work Experience

Tasks Level	
Level-0	Freshers
Level-1	1-2 years exp
Level-2	3-5 years exp
Level-3	5+ years exp
Level-4	8+ Technology & Leadership exp
ITIL	Process Expertise

MONGODB Database Project & Support Services

Unlike the traditional RDBMS (ie: Oracle, Sql Server), MongoDB is the NO-SQL Unstructured Database. So, it would be strongly recommended to have MongoDB expertise to handle it from scratch. We provide following structured services:

Database Health Check (level-1)

1. Daily check the status of mongodb process is running in prod/dev/qa server
2. Review and analyze system logs for any error messages or warnings that might indicate potential problems.
3. If you're using a replica set for high availability, we check the status of each member, and ensure that they are all in sync and have the same data.
4. Database Space check
5. OS Space check
6. Proactive basic Performance check
7. Sharding Status: For sharded clusters, verify the status and health of the shards and the balancer to ensure data distribution is even and efficient.
8. Hardware Resources: Monitor the server's hardware resources, including CPU usage, memory consumption, and disk I/O. Ensure that the hardware can handle the current workload and accommodate future growth.

Database Design, Installation & Configuration. (Level-3)

1. MongoDB installation with source binary files/tar file on different OS linux, centos, Ubuntu.
2. Design OS requirements: Choosing & configuring the appropriate CPU, Memory, storage and ancillary software
3. Design Database Structures: Creating scalable database architectures that allows for expansion
4. Conducting design reviews
5. Designing database for high-speed, high-volume transactions
6. Installation and configuration of database from source or Packages on all types Operating Systems.
7. Preparing documentation
8. Installation & configuration of all types Database client side & server tools for both windows & Linux. (ie: mongo compass etc.)
9. Schema Design – working with Application/Development Team.
10. Version Check: Verify that you are running the latest stable version of MongoDB. Upgrading to the latest version may include bug fixes, performance improvements, and security enhancements.
11. Resource Balancing: Ensure that multiple MongoDB instances running on the same server are appropriately configured to prevent resource contention.
12. Memory Management: Configure MongoDB's memory-related settings, such as the WiredTiger cache size and the operating system's page cache, to optimize read and write performance.
13. Choose hardware with sufficient CPU, memory, and storage resources to handle the database workload.

Backup & Disaster Recovery Management (Level-2)

Backup your data:

Regularly backing up your data will ensure that you have access to your information even if a cyberattack or system failure occurs.

1. **Implement optimal backup and recovery strategy:** Configure new backups
2. **Regular Backups:** Schedule regular backups to capture the latest changes in the database. The frequency of backups should align with your application's Recovery Point Objective (RPO) to minimize data loss in the event of a disaster.
3. Re-Run failed backups
4. Ensure, all backups are stored in a safer remote location to restore consistency.
5. Implementing best practices for backups, replication, and health checks

Disaster Recovery Planning:

1. **Data Archival:** Consider long-term data archival for compliance or historical purposes, ensuring data retention even in the case of accidental data deletion.
2. **Recovery Time Objective (RTO):** Define the acceptable downtime for your application and design your disaster recovery plan to meet this RTO.

Backup Verification and Testing:

1. **Backup Integrity:** Periodically verify the integrity of backups to ensure they are valid and restorable. Use checksums and test restores to validate backup data.

2. **Backup and Restore:** Regularly test your backup and restore processes to ensure data recoverability in case of failures.

3. **Disaster Recovery Drills:** Conduct disaster recovery drills to test the recovery process and identify any potential issues before a real disaster occurs.

Backup Types:

- **mongodump** – A utility included with MongoDB that can dump an entire database or query result, creating a snapshot of a database. Great for small deployments but struggles with larger databases.

- **filesystem snapshots** – Use tools like LinuxLVM to take snapshots of the file system. A reliable way to create large backups.

- **MongoDB Management Service (MMS)** – Managed online backup service that continuously streams MongoDB oplog data to MMS to create backups. Takes snapshots every 6 hours with 24 hours retention.
- **Backup Consistency:** Full backups and incremental backups, to balance data protection and storage efficiency.
- **Snapshot-Based Backups:** For sharded clusters and replica sets, use storage system snapshots (e.g., LVM snapshots, EBS snapshots) to create consistent backups without impacting database performance

Standby Database Management (Database Replication) & HA (Level-2 & 3)

1. **Replication setup:** Architecting, Installation & Configuration of a new standby database.
2. **Replica Set Deployment:** Plan and execute a replica set deployment for high availability and data redundancy.
3. **Oplog Size:** The oplog (operation log) is essential for replication. Ensure that the oplog is large enough to accommodate the typical workload and provides enough history for your recovery needs.
4. Replication lags: Re-sync if in case of any gaps
5. Designing DR (Disaster Recovery)/COB (Continuity of Business)
6. Failover/switchback

7. Perform DR drill activity every 6 month DC/DR activity.
8. Node failures
9. Configuring arbiter nodes.
10. Tuning the replica vote values.
11. Query routing.
12. Enable Read Preferences
13. **MongoDB Cluster**

- **Replica Sets:** A MongoDB replica set is a collection of one or more servers that contain an exact copy of the data. While it is possible to have one or two nodes, three is the recommended minimum. A primary node handles read and write operations for your application, while two secondary nodes hold a replica of the data.

- **Sharded Clusters:** A sharded cluster is a method of horizontally scaling your data by distributing it across multiple replica sets. The client sends a request to the router (mongos) whenever a read or write operation is performed on a collection. The router will then use the configuration server to determine which shard the data is stored in. It then sends the requests to the appropriate cluster.

Database User and Security Management (Level-1 & 2)

1. To create database, user and restricted privileges.
2. User Management in existing setup.
3. **Security Settings:** Verify that proper security measures are in place, such as authentication, authorization, and SSL encryption to protect sensitive data.

4. **Encrypt your data at rest & SSL:** WiredTiger is a storage engine that provides native encryption at rest so that your data can't be read by an individual unless they have a decryption key to translate the protected data into something readable.

5. Enable access controls and Use Role-based access control (and the principle of least privilege): Role-Based Access Control (RBAC) is included with MongoDB by default since version 2. With RBAC you can control what actions a user has permission to take and which resources they can access. There are five different database roles included by default:

   - **read** – Read-only access.
   - **readWrite** – Permission to read and edit data.
   - **dbAdmin** – Permission to perform administrative tasks like indexing.
   - **db0wner** – Permission to perform any administrative action on the databases (combines readWrite, dbAdmin, and userAdmin).
   - **userAdmin** – Permission to create and modify roles and users.

6. **Compliance Auditing:** Auditing user actions taken within databases is critical for identifying malicious activity and troubleshooting.

7. User Authentication: User authentication is a must-have for restricting who can connect to the database and verifying the identity of a client.

   - Salted Challenge Response Authentication Mechanism (SCRAM)
   - x.509 Certificate Authentication
   - Lightweight Directory Access Protocol (LDAP) Proxy Authentication
   - Kerberos authentication

8. **Encrypt network traffic:** To protect your data, encrypt network traffic to and from the database by configuring Transport Layer Security (TLS) and Secure Sockets Layer (SSL). You can deploy TLS/SSL by setting net.ssl.mode to requireSSL, which will instruct the system to only permit TLS/SSL connections.

9. Inbound/Outbound whitelisting:

10. Define custom roles for more fine-grained access control when needed.

11. Create a dedicated administrative user with the necessary privileges to manage the database and its users.

12. Enforce strong password policies, including minimum password length, complexity requirements, and password expiration.

13. Configure IP whitelisting to restrict access to the database only from known and trusted IP addresses.

14. Enable auditing to log user authentication and authorization activities.

15. If using MongoDB Atlas (MongoDB's cloud-based database service), leverage built-in security features like database-level access controls and network peering to enhance security.

16. Periodically review user access and roles to ensure that only necessary permissions are granted and revoke any unnecessary privileges.

Database Capacity Planning & Space Management (Level-3)

1. **Capacity Planning:** Designing database storage requirements by considering immediate future growth.

2. Performing periodic capacity reviews (daily, Weekly, Monthly, and Quarterly based on customer requirements & send database growth report regularly.

3. **Data Size and Growth:** Keep track of the database size and its growth rate to anticipate storage needs and avoid potential space-related issues.

4. Add additional space by taking approval from management.
5. Proactive housekeeping: Log rotation / maintenance (mongos, mongod, config etc)

6. Monitor disk usage regularly to identify trends and potential space constraints.
7. Set up alerts for disk space utilization to proactively address any space-related issues.
8. Utilize compression techniques (e.g., WiredTiger compression) to reduce storage footprint and improve performance.
9. For data with a defined time-to-live (TTL), use TTL indexes to automatically delete documents after a certain period, freeing up space.
10. Periodically run the compact command to reclaim disk space by removing deleted records and freeing up space in data files.
11. Use the repairDatabase command to defragment data files and reclaim disk space.
12. Consider archiving old or infrequently accessed data to a separate storage system.
13. Regularly review and purge unnecessary or obsolete data from the database.

Performance Tuning (level 2 & 3)

1. Identifiy the slow queries.
2. Database profiling, Locks, No of connections,index usage patterns,CPU Usage, Memory Usage
3. Database profiling, locks, memory usage,no of connections, page fault.
4. Max connections monitoring and clean up sleep connections.
5. Analyzing database logs for erros and performing routine table and index maintenance tasks.
6. Resolving table and index level database corruption.
7. Opt for SSD storage for better I/O performance.
8. Configure the WiredTiger cache size to fit the working set in memory to minimize disk I/O.
9. Carefully choose the appropriate write concern to balance data durability and write performance.
10. Use connection pooling to efficiently manage and reuse database connections, reducing connection overhead

11. Separate MongoDB instances from other resource-intensive applications to avoid contention for CPU and memory.

12. **Storage Engine Performance:** MongoDB offers different storage engines like WiredTiger and MMAPv1. Evaluate the performance of your chosen storage engine and consider switching if it's not optimal for your workload.

13. **Indexes:** Review the indexes on your collections to ensure they are appropriate for the most common queries. Unnecessary indexes can slow down write operations, and missing indexes can impact read performance.

Query Tuning:-

1. **Analyze the query performance in mongodb:** (using Build-in profiling or Monitoring Tools) The cursor.explain ("executionStats") and the db.collection.explain("executionStats") methods provide statistics about the performance of a query. These statistics can be useful in measuring if and how a query uses an index. See db.collection.explain() for details.

2. **Query Optimization:** Create an Index to Support Read Operations
   Limit the Number of Query Results to Reduce Network Demand - limit() method
   Use $hint to Select a Particular Index - hint() method
   Use the Increment Operator to Perform Operations Server-Side - Use $inc operator

3. Identify frequently executed queries and create appropriate indexes to support them

4. Use compound indexes for queries that have multiple fields in the filter criteria or sort operations.

5. Analyze query patterns and adjust the data model and indexes to better suit the application's needs.

6. **Operating System Tuning:** Optimize the operating system settings for MongoDB deployment, including file descriptor limits, ulimit settings, and kernel parameters.

7. **Linus (OS) level best practices:**

- Set **ulimit** - 64,000 “max user processes” and 64,000 “open files”
  /etc/security/limits.d/mongod.conf

- **Virtual Memory** - The “dirty_ratio” is the percentage of total system memory that can hold dirty pages
  /etc/sysctl.conf
  vm.dirty_ratio = 15

  vm.dirty_background_ratio = 5

- **Swappiness -** “Swappiness” is a Linux kernel setting that influences the behavior of the Virtual Memory manager when it needs to allocate a swap, ranging from 0-100.
  /etc/sysctl.conf
  vm.swappiness = 1

- **Read-Ahead -** Read-ahead is a per-block device performance tuning in Linux that causes data ahead of a requested block on disk to be read and then cached into the filesystem cache.
  /etc/udev/rules.d/60-sda.rules

  # set deadline scheduler and 16kb read-ahead for /dev/sda

  ACTION=="add|change", KERNEL=="sda", ATTR{queue/scheduler}="deadline", ATTR{bdi/read_ahead_kb}="16"

- **Filesystem and Options -** It is recommended that MongoDB uses only the ext4 or XFS filesystems for on-disk database data

- **Network Stack -** MongoDB, limit a typical host with 1000mbps network interfaces

  /etc/sysctl.conf

  net.core.somaxconn = 4096

```
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_keepalive_intvl = 30
net.ipv4.tcp_keepalive_time = 120
net.ipv4.tcp_max_syn_backlog = 4096
net.core.somaxconn = 4096
```

Database Patch & Upgrade (Level-2)

1. Upgrade MongoDB different version on linux/centos/Ubuntu & Java Version
2. Work closely with the application team to understand what changes are coming in a build/patch/hotfix.
3. Some upgrades may have specific prerequisites or requirements. Make sure to fulfill them before starting the upgrade process.
4. Ensure that the patches or upgrades are compatible with your current hardware, operating system, and other software dependencies.
5. Patching or upgrading may require downtime / no-downtime, so plan the maintenance window accordingly to minimize the impact on users and applications.
6. Verify that the applications using the database will continue to work as expected with the new version. Check for any changes in behavior that might affect the application.
7. After applying patches or upgrades, benchmark and test the database performance to ensure it meets expected levels.
8. Have a rollback plan in place in case the patching or upgrade process encounters critical issues. This should include a step-by-step procedure to revert to the previous version.
9. Security fixes

Database Refresh (Level-2)

1. Database refresh / collection level refresh activities
2. Logical method
3. From 3rd party TAPE & Backup Storage

Database Migration (Level-3)

1. Straight & Cross Platform Database migrations.
2. Homogeneous & Heterogeneous
3. On-Perm to cloud & vice versa: Migrate onprem MongoDB to AWS Ec2/RDS.
4. Between Datacenters
5. MongoDB Atlas

Vendor & Licensing Management (Level-2)

1. Utilizing our vendor relationships, we increase turn-around speed and get licensing information quickly.
2. We navigate the licensing maze, providing you with all the licensing options available.
3. We regularly work with software vendors to get the "best pricing" for our clients.

Database Continuous Monitoring: - (Level-1)

1. Implementation of Customized Monitoring using Shell & PowerShell.
2. Implement any open-source monitoring tools like Grafana or PMM.

6. Monitoring at server, Database, Collection Level and various monitoring tools related to MongoDB.
7. Splunk: Monitor the logs and raise on-call alerts to DBAs at the time of an incident.
8. Proactive checks based on the monitoring alerts help DBAs to keep the system healthy.

Database Process Implementation, Documentation & Orientation (ITIL)

- Introducing best-fit processes for database administration using ITIL global standards.
- Deriving & implementing new processes & periodically reviewing existing processes, in Incident Management, Problem Management, Change Management & Release management.
- Can Manage ITIL portfolio by supplying, Incident Manager, Problem Manager, Change Manager, Release Manager.
- Documenting the SOP (Standard Operating Procedures) for end-to-end database administration portfolio for your various environments Prod, QA & Development.
- Also we educate your IT / Non-IT staffs for the Process & SOP structured.
- **Compliance and Best Practices:** Check if your MongoDB deployment follows recommended best practices and complies with any relevant industry or organizational standards.
-

Database Status Reporting to IT Management (Level 1)

1. **Backup Status Report:** (Daily/Weekly/Monthly/Quarterly) backup status reports for end-to-end database environment.
2. **Daily Status Check Report:** Services Availability Status, Storage Status & Standby Sync status, HA Availability Status.
3. **Performance Status Report:** Periodic database Locking, Blocking, Inactive sessions status.
4. **Capacity Growth Report:** Periodic database storage growth (Daily/Weekly/Monthly/Quarterly) for Production databases.
5. **Others:** Other Database related report as the customers customized requirements.

Cloud 'Database as Service' support (Level-2 & 3)

End to End Database Consultation, Administration & Management in all Cloud Technologies (as well as in Hybrid Model)

We are expertise in supporting below cloud environments

Cloud We Support

Azure aws ORACLE CLOUD Google Cloud

Also we are in Technical Partnership with **SpeedCloud** cloud company and provide special discounted rates for our customers on our Techno-partner **SpeedCloud** cloud platform.

Our Database Support in cloud includes: -

- **Rehost patterns** (from on-perm database to self managed* in Amazon EC2, Azure VM, Compute Engine in GCC etc.)
- **Re-platform patterns** (from on-perm database to cloud DBaaS*)
- **Re-architect patterns** (from on-perm one database Technology to other open-source or cloud-native database Technologies)
- Setup monitoring of DBs using cloud native tools. (Eg: Cloud watch)
- Setup and manage DB in muti, hybrid cloud environment like AWS, AZURE & GCC along with On-Perm

- **Database Consulting in Cloud Infra**
    - Start with a comprehensive plan and a governance framework
    - Run the right database in the right cloud
    - Use data services that support multi-cloud environments
    - Exploit managed database services, or DBaaS
    - Consider database portability across multiple clouds
    - Optimize data access for applications and end users
    - Connect cloud networks to reduce data latency

- **Cloud Security**
    - Define standards, security, and compliance policies. Cloud database vendors rarely enforce more than the most obvious weaknesses in the out-of-the-box installations of their platforms
    - Run vulnerability assessments. Since databases are often an organization's largest repository of sensitive information, they should be evaluated to not only search for potential vulnerabilities but also to ensure they fulfill any relevant regulatory compliance requirements
    - Understand user privilege and access. As people change roles or leave an organization, user privileges are often not kept up-to-date, and, as a result, organizations lack a full understanding of who has access to sensitive data.
    - Use data analytics to mitigate risks. Remediating high-risk vulnerabilities and misconfigurations within your databases not only reduces your risk of compromise, but it also narrows the scope of any required compensating controls you might need, such as exploit monitoring.
    - Respond to policy violations in real time. For vulnerabilities that cannot be remediated or patched in a timely manner, real-time database activity monitoring (DAM) can be an appropriate compensating control.
    - Database encryption
    - Data Masking
    - Multi-Factor Authentication (MFA)

- **Cloud Database Migration**
  - Migrate your existing database platform from On-premises datacenter to any cloud Technologies includes (AWS, Azure, Google Cloud, Oracle Cloud & other 3rd party cloud services) & Vice versa, Cloud to On-premises datacenter
  - New database design and implementation in Cloud platform.
  - Cloud database integration with other cloud services as per your business functional requirements.

- **Cloud Database Performance Tuning**

DBaaS* (Database as a service), Managed database service that are fully managed by the vendor, which could be a cloud platform provider or another database vendor that runs its cloud DBMS on a platform provider's infrastructure

Self-managed database*: This is an infrastructure as a service (IaaS) environment, in which the database runs in a virtual machine on a system operated by a cloud provider