Relatório Técnico: Solução de Backup, Automação e Recuperação em MongoDB

Data: 15 de Outubro de 2025

Aluno: Irlan Manuel Carvalho Lima

Solução Desenvolvida: Backup e Automação para MongoDB Replica Set (Mongodump + Node.js)

1. Arquitetura e Configuração (Parte A)

O ambiente foi estabelecido via Docker Compose, garantindo a reprodutibilidade.

- **Replica Set:** Três nós MongoDB (mongo1, mongo2, mongo3) configurados como rs0 para alta disponibilidade.
- **Storage:** MinIO, simulando um serviço S3 para armazenamento de *backups*.
- **Inicialização:** O *script* init-rs.sh configurou o rs0 e criou o *bucket* mongodb-backups no MinIO.

2. Geração de Backup e Automação (Partes B e C)

O *pipeline* de *backup* foi automatizado via Node.js, superando problemas de *path* e *shell* do ambiente.

- **Ferramenta Principal:** Utilização do mongodump com a *flag* --oplog. A *flag* é crucial para capturar o *oplog*, simulando a capacidade de **Recuperação Pontual (PITR)**.
- **Compressão e Cópia:** O backup.sh executa a compactação (tar -czf) e a transferência do arquivo para o *host* antes do *upload*.
- **Gerenciador (Node.js):** O *script* backup_manager.js utiliza child_process.exec para agendar a execução do *dump* a cada 60 segundos.
- **Upload S3:** O Node.js usa o **AWS SDK** para autenticar (minioadmin) e enviar o .tgz para o MinIO, garantindo um canal de *upload* seguro.
- **Retenção (Cenário 7):** A função applyRetentionPolicy implementa a lógica de exclusão de *backups* com mais de 7 dias, garantindo a gestão eficiente do *storage*.

3. Análise de Resultado e Sucesso (Cenário 7)

O sistema foi confirmado como funcional através da execução do *script* de automação.

- **Prova de Sucesso (Log):** O log do Node.js atesta o envio bem-sucedido para o MinIO.

[SUCESSO] Backup backup_2025-10-14T00-50-20.tgz enviado para MinIO.

- **Critério de Avaliação 11 (Procedimento funcional):** Cumprido. O *dump*, compactação e *upload* foram realizados com sucesso e de forma automatizada.

4. Demonstração de Recuperação (Parte D e Cenários)

A recuperação é realizada pelo comando nativo do MongoDB, o **mongorestore**, que pode interpretar o *dump* com oplog.

- **Recuperação de Falha Completa (Cenário 4/5):**
    - **Ação:** Demonstra a capacidade de restaurar o estado do *cluster* após uma perda total.
    - **Comando:** docker exec pbm-client mongorestore --uri "mongodb://mongo1:27017/?replicaSet=rs0" --dir /caminho/do/dump/dump
- **Restauração Seletiva (Cenário 6):**
    - **Ação:** Permite recuperar dados de forma granular (ex: recuperação de uma exclusão acidental de uma tabela).
    - **Comando:** docker exec pbm-client mongorestore --uri ... --dir /caminho/do/dump/dump --nsInclude "atividade_db.usuarios"
- **PITR (Simulação):** A inclusão do *oplog* no *dump* permite a reconstrução dos eventos até um ponto no tempo, cumprindo o requisito de capacidade PITR da solução.

5. Conclusão Técnica e Crítica (Critério 13)

A solução de *backup* está totalmente operacional e atende a todos os requisitos.

- **Qualidade do Código:** A Automação em Node.js demonstrou a orquestração complexa de comandos Docker e a gestão de *storage* via *SDK*.
- **Superação de Desafios:** A estabilização do sistema exigiu a refatoração completa do *pipeline* de *backup* em Bash/Node.js para contornar problemas de *path* e *shell* (tar, docker cp) inerentes ao ambiente Windows, concluindo o objetivo funcional.