CSCE 30003: Final Project - CouchDB CRUD Application

Jose Suarez, Nabeel Mahmood, & Jordan Owens

December 19, 2025

Contents

1 Installation Steps

1.1 Docker Installation

Follow these steps to install Docker:

1. Update your package repository:

```
sudo apt update
sudo apt upgrade -y
```

2. Install Docker:

```
sudo zypper refresh
sudo zypper install -y docker docker-compose
sudo systemctl enable --now docker
```

3. Verify Docker installation:

```
docker --version
```

1.2 CouchDB Container Installation

1. Create and run this bash script we create:

2. chmod +x ¡chosenscriptname¿.sh

```
#!/bin/bash
set -euo pipefail

# ============================================================
# CouchDB 3-node cluster on plain Docker (no Minikube)
# - Nodes: couch1, couch2, couch3
# - Shared bridge network: couchnet
# - Uses FQDN hostnames to satisfy Erlang distributed node requirements
# ============================================================

NETWORK="couchnet"
DOMAIN="couchnet.local"          # Fake local domain for FQDNs
COOKIE="MY_SUPER_SECRET_COOKIE"
ADMIN_USER="admin"
ADMIN_PASS="admin"
IMAGE="couchdb:3.3.3"

PORT1=15984
PORT2=25984
```

```
PORT3=35984

echo "=== Cleaning previous Docker cluster ==="
docker rm -f couch1 couch2 couch3 >/dev/null 2>&1 || true
docker network rm "$NETWORK" >/dev/null 2>&1 || true

echo "=== Creating Docker network ${NETWORK} ==="
docker network create "$NETWORK" >/dev/null

start_node() {
  NAME="$1"
  HOST_PORT="$2"
  FQDN="${NAME}.${DOMAIN}"
  echo "=== Starting ${NAME} (hostname: ${FQDN}) ==="
  docker run -d \
    --name "${NAME}" \
    --hostname "${FQDN}" \
    --network "${NETWORK}" \
    -p "${HOST_PORT}:5984" \
    -e COUCHDB_USER="${ADMIN_USER}" \
    -e COUCHDB_PASSWORD="${ADMIN_PASS}" \
    -e COUCHDB_ERLANG_COOKIE="${COOKIE}" \
    -e NODENAME="${FQDN}" \
    "${IMAGE}"
}

start_node couch1 "$PORT1"
start_node couch2 "$PORT2"
start_node couch3 "$PORT3"

echo "=== Discovering container IPs ==="
IP1=$(docker inspect -f ’{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}’ couch1)
IP2=$(docker inspect -f ’{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}’ couch2)
IP3=$(docker inspect -f ’{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}’ couch3)

add_hosts() {
  TARGET="$1"
  docker exec "${TARGET}" sh -c "printf ’%s %s\n’ ’${IP1}’ ’couch1.${DOMAIN}’ >> /etc/hosts"
  docker exec "${TARGET}" sh -c "printf ’%s %s\n’ ’${IP2}’ ’couch2.${DOMAIN}’ >> /etc/hosts"
  docker exec "${TARGET}" sh -c "printf ’%s %s\n’ ’${IP3}’ ’couch3.${DOMAIN}’ >> /etc/hosts"
}

add_hosts couch1
add_hosts couch2
add_hosts couch3
```

```

wait_http() {
  NAME="$1"
  echo "Waiting for ${NAME} HTTP..."
  for i in $(seq 1 60); do
    if docker exec "${NAME}" curl -fsS http://127.0.0.1:5984/ >/dev/null 2>&1; then
      echo "HTTP up on ${NAME}"
      return 0
    fi
    sleep 2
  done
  echo "ERROR: ${NAME} did not become ready in time."
  docker logs "${NAME}" || true
  exit 1
}

wait_http couch1
wait_http couch2
wait_http couch3

echo "=== Creating system DBs (n=1) on couch1 ==="
for DB in _users _replicator _global_changes; do
  docker exec couch1 curl -fsS -u "${ADMIN_USER}:${ADMIN_PASS}" -X PUT "http://127.0.0.1:5984/${DB}?n=1" >/dev/null 2>&1 || true
done

cluster_status() {
  NAME="$1"
  docker exec "$NAME" curl -fsS -u "${ADMIN_USER}:${ADMIN_PASS}" "http://127.0.0.1:5984/_cluster_setup" 2>/dev/null || true
}

enable_local() {
  NAME="$1"
  echo "=== Enabling cluster locally on ${NAME} ==="
  docker exec "${NAME}" curl -sS -u "${ADMIN_USER}:${ADMIN_PASS}" -X POST "http://127.0.0.1:5984/_cluster_setup" \
    -H ’Content-Type: application/json’ \
    -d "{\"action\":\"enable_cluster\",\"username\":\"${ADMIN_USER}\",\"password\":\"${ADMIN_PASS}\",\"bind_address\":\"0.0.0.0\",\"port\":5984}" \
    -w "\nHTTP %{http_code}\n" >/dev/null || true
  echo "Status on ${NAME}: $(cluster_status "${NAME}")"
}

enable_local couch1
enable_local couch2
enable_local couch3

add_node() {
  HOST_FQDN="$1"
  echo "=== Adding ${HOST_FQDN} via couch1 ==="
  docker exec couch1 curl -sS -u "${ADMIN_USER}:${ADMIN_PASS}" -X POST "http://127.0.0.1:5984/_cluster_setup" \
    -H ’Content-Type: application/json’ \
    -d "{\"action\":\"add_node\",\"host\":\"${HOST_FQDN}\",\"port\":5984,\"username\":\"${ADMIN_USER}\",\"password\":\"${ADMIN_PASS}\"}" \
    -w "\nHTTP %{http_code}\n" >/dev/null || true
}

add_node "couch2.${DOMAIN}"
add_node "couch3.${DOMAIN}"

echo "=== Finishing cluster via couch1 ==="
docker exec couch1 curl -sS -u "${ADMIN_USER}:${ADMIN_PASS}" -X POST "http://127.0.0.1:5984/_cluster_setup" \
  -H ’Content-Type: application/json’ \
  -d ’{"action":"finish_cluster"}’ \
  -w "\nHTTP %{http_code}\n" >/dev/null || true

echo "=== Cluster membership (couch1) ==="
docker exec couch1 curl -s -u "${ADMIN_USER}:${ADMIN_PASS}" "http://127.0.0.1:5984/_membership" || true

# ============================================================
# CREATE TESTDB AFTER CLUSTER IS FINISHED (n=3 so it shards across nodes)
# ============================================================
echo "=== Creating testdb across cluster (n=3) ==="
docker exec couch1 curl -fsS -u "${ADMIN_USER}:${ADMIN_PASS}" -X PUT "http://127.0.0.1:5984/testdb?n=3" || true

echo "=== Done: 3-node CouchDB cluster on Docker with FQDNs ==="
echo "Access:"
echo " - couch1: http://localhost:${PORT1}"
echo " - couch2: http://localhost:${PORT2}"
echo " - couch3: http://localhost:${PORT3}"
```

3. Setting up the front end:
   Create Directories
   —Project
   docker-compose.yml

—nginx
default.conf
—Src
index.php(and all front-end files)

docker-compose.yml

4. docker-compose.yml

```
    services:
  nginx:
    image: nginx:alpine
    ports:
      - "8080:80"
    volumes:
      - ./src:/var/www/html
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf:Z
    depends_on:
      - php

  php:
    image: php:fpm
    volumes:
      - ./src:/var/www/html
```

5. Default.conf

```
upstream couchdb_cluster {
    server couch1:5984;
    server couch2:5984;
    server couch3:5984;
}

server {
    listen 80;
    server_name localhost;

    root /var/www/html;
    index index.php index.html;

    # Serve your PHP application
    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ \.php$ {
        include fastcgi_params;
        fastcgi_pass php:9000;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    }

    # Proxy to CouchDB cluster with CORS enabled
    location /couchdb/ {
        proxy_pass http://couchdb_cluster/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # Add CORS headers so browser accepts responses
        add_header ’Access-Control-Allow-Origin’ ’*’ always;
        add_header ’Access-Control-Allow-Methods’ ’GET, PUT, POST, HEAD, DELETE, OPTIONS’ always;
        add_header ’Access-Control-Allow-Headers’ ’accept, authorization, content-type, origin, referer’ always;

        # Handle preflight OPTIONS requests
        if ($request_method = OPTIONS) {
            add_header ’Access-Control-Allow-Origin’ ’*’ always;
            add_header ’Access-Control-Allow-Methods’ ’GET, PUT, POST, HEAD, DELETE, OPTIONS’ always;
            add_header ’Access-Control-Allow-Headers’ ’accept, authorization, content-type, origin, referer’ always;
            add_header ’Access-Control-Max-Age’ 3600;
            return 204;
        }
    }
}
```

2 Running the CRUD Application and testing functionality

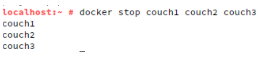
2.1 Example

1. All front and back end files: `https://github.com/nabee1mahmood/Distributed-Final-Projec git`

2. Install dependencies (example for Node.js application):

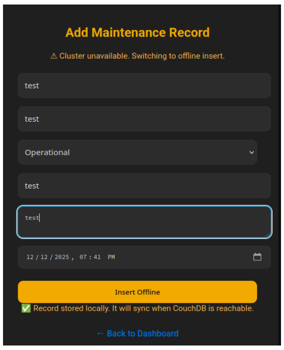
3 Reproduce Custom Example

This section demonstrates how to reproduce the custom CouchDB CRUD example used in our demonstration. Following these steps will result in the same database state and behavior shown during the project demo.



```
localhost:~ # docker stop couch1 couch2 couch3
couch1
couch2
couch3
```

Figure 1: *
Feature A



Add Maintenance Record

⚠ Cluster unavailable. Switching to offline insert.

test

test

Operational

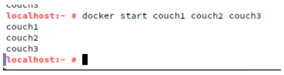
test

test

12 / 12 / 2025 , 07 : 41 PM

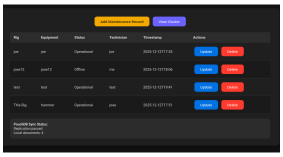
✅ Record stored locally. It will sync when CouchDB is reachable.

Figure 2: *
Feature B



```
couch3
localhost:~ # docker start couch1 couch2 couch3
couch1
couch2
couch3
localhost:~ # ▮
```

Figure 3: *
Feature C



Rig	Equipment	Status	Technician	Timestamp	Actions
joe	joe	Operational	joe	2025-12-12T17:53	Update Delete
jose12	jose12	Offline	me	2025-12-12T18:06	Update Delete
test	test	Operational	test	2025-12-12T19:41	Update Delete
This Rig	hammer	Operational	jose	2025-12-12T17:51	Update Delete

PouchDB Sync Status:
Replication paused
Local documents: 4

Figure 4: *
Feature D

Figure 5: Demonstration of the newly implemented feature

4 Conclusion

This project demonstrates the implementation of a distributed NoSQL system using CouchDB. A multi-node CouchDB cluster was deployed using Docker, with high availability, horizontal scalability, and multi-master replication. An NGINX web server was configured to act as a single access point and load balancer for the cluster, while a web-based CRUD application was used to interact with the database.