Deploying a Persistent and Secure Keycloak Cluster with Docker Compose + Nginx + SSL

```
# □□Deploying a Persistent and Secure Keycloak Cluster with Docker Compose + Nginx + SSL

Keycloak is a powerful open-source identity and access management solution. In this guide,
we'll set up a **persistent**, **SSL-secured**, and **Nginx-proxied** Keycloak deployment
using **Docker Compose** — perfect for production or local testing.

---

## □□Project Overview

We'll be running:
- PostgreSQL 15.5 (persistent data)
- Keycloak 26.0.0
- Nginx reverse proxy with Let's Encrypt SSL

Domain used: `keycloak.exceltoquiz.com`
Ports used: `7081` (Keycloak), `5432` (Postgres)

---

## □□□Folder Structure

```
keycloak-cluster/
├── docker-compose.yml
├── postgres_data/       # Persistent Postgres data
└── nginx.conf           # Nginx reverse proxy config
```

```

□□Docker Compose Configuration

Below is the working `docker-compose.yml` file:

```yaml
version: "3.9"

services:
  postgres:
    container_name: keycloak_db
    image: postgres:15.5
    restart: always
    healthcheck:
      test: ["CMD", "pg_isready", "-q", "-d", "keycloak", "-U", "postgres"]
      timeout: 30s
      interval: 10s
      retries: 5
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: keycloak
    volumes:
      - ./postgres_data:/var/lib/postgresql/data
    networks:
      - keycloak_net
    ports:
      - "5432:5432"

  kc1:
    container_name: keycloak_1
    image: quay.io/keycloak/keycloak:26.0.0
    command: ["start-dev"]
    restart: always
    depends_on:
      - postgres
    environment:
      KC_DB: postgres
      KC_DB_URL: jdbc:postgresql://postgres/keycloak
      KC_DB_USERNAME: postgres
```

```
      KC_DB_PASSWORD: postgres

      KC_HTTP_ENABLED: "true"
      KC_PROXY_HEADERS: xforwarded
      KC_HOSTNAME: keycloak.exceltoquiz.com
      KC_HOSTNAME_URL: https://keycloak.exceltoquiz.com
      KC_HOSTNAME_ADMIN_URL: https://keycloak.exceltoquiz.com
      KC_BOOTSTRAP_ADMIN_PASSWORD: admin
      KC_BOOTSTRAP_ADMIN_USERNAME: admin
    networks:
      - keycloak_net
    ports:
      - "7081:8080"

networks:
  keycloak_net:
    driver: bridge
```

□□Notes:
- Persistent Postgres data stored in `./postgres_data`
- Fixed permissions (if needed) using:
  ```bash
  sudo chown -R 999:999 postgres_data
  ```
- Keycloak runs in `start-dev` mode (suitable for initial setup)

□□Nginx HTTPS Reverse Proxy

The following **Nginx configuration** handles HTTPS termination via Certbot-managed certificates:

```nginx
upstream keycloak_cluster {
    server 127.0.0.1:7081;
}

server {
```

```
    server_name keycloak.exceltoquiz.com;

    location / {
        proxy_pass http://keycloak_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 https;
        proxy_set_header X-Forwarded-Port 443;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/keycloak.exceltoquiz.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/keycloak.exceltoquiz.com/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

server {
    if ($host = keycloak.exceltoquiz.com) {
        return 301 https://$host$request_uri;
    }

    listen 80;
    server_name keycloak.exceltoquiz.com;
    return 404;
}
```

□□Enable SSL with Certbot

Install Certbot and obtain the certificate:

```bash
sudo mkdir -p /var/www/certbot
sudo certbot certonly --webroot -w /var/www/certbot -d keycloak.exceltoquiz.com
```

Reload Nginx after Certbot setup:

```bash
sudo systemctl reload nginx
```

□□Useful Commands

Command	Description
`docker compose up -d`	Start all containers
`docker compose logs -f`	View logs
`docker exec -it keycloak_db psql -U postgres -l`	List databases
`docker exec -it keycloak_db psql -U postgres -c "CREATE DATABASE keycloak;"`	Create Keycloak DB manually
`sudo chown -R 999:999 postgres_data`	Fix folder permission for Postgres persistence

□□Result

You now have:
- Persistent Postgres data stored locally
- Secure Keycloak running at **https://keycloak.exceltoquiz.com**
- SSL managed automatically by Certbot
- Production-ready reverse proxy with Nginx

□□**Tip:** You can easily scale Keycloak horizontally by adding another service block (e.g., `kc2`) and including it in the `upstream` block.

Author: Md Golam Kibria (GK)
Tags: `Keycloak`, `Docker`, `Nginx`, `SSL`, `PostgreSQL`, `DevOps`

Revision #2
Created 3 November 2025 13:27:23 by Admin
Updated 3 November 2025 13:30:47 by Admin