# Mustard Seed Bank — Full Auto Repo (Deploy + Status + Reset + Backup/Restore)

This repo is designed for **minimum technical knowledge** deployment. It includes: - Dockerized Flask backend + Postgres + Nginx - Automatic Let’s Encrypt HTTPS with **auto-renewal** - Seed script for demo members - **Status check script** (./status.sh) - **Reset script** (./reset.sh) to safely wipe and rebuild - **Backup & Restore scripts** (./backup.sh + ./restore.sh) for safe data handling

## ✅ Deployment Steps (Start to Finish)

1. **Prepare GitHub repository (optional but recommended)**
   * Go to <https://github.com/new>
   * Create a new **empty** repository called mustard-seed-bank
   * Or use ./init\_github.sh to auto-create and push.
2. **Clone the repo on your server**

* git clone https://github.com/YOUR\_USERNAME/mustard-seed-bank.git  
  cd mustard-seed-bank

1. **Edit environment file**

* cp .env.example .env  
  nano .env # update DOMAIN=yourdomain.com and EMAIL=your@email.com
* Required fields:
  + DOMAIN: the real domain pointing to your server (set DNS A record).
  + EMAIL: email for Let’s Encrypt notifications.

1. **Run setup**

* chmod +x setup.sh seed.sh status.sh reset.sh backup.sh restore.sh  
  ./setup.sh
  + Builds Docker containers
  + Starts Postgres + backend + nginx
  + Seeds members
  + Requests HTTPS certificate
  + Sets up cron auto-renew

1. **Access your site**
   * Visit: https://yourdomain.com
   * Certificates auto-renew daily at 3 AM.
2. **Check status anytime**

* ./status.sh
* Shows:
  + Docker container status
  + Database connection check
  + SSL certificate expiry date

1. **Reset everything if needed**

* ./reset.sh
* Wipes database & certs, then rebuilds everything.

1. **Backup data**

* ./backup.sh
  + Creates backups/mustard\_backup\_YYYYMMDD.sql.gz
  + Includes Postgres DB + certs snapshot

1. **Restore data**

* ./restore.sh backups/mustard\_backup\_YYYYMMDD.sql.gz
* Restores DB from a backup file.

## 🔧 Key Repo Files

### .env.example

POSTGRES\_USER=msb\_user  
POSTGRES\_PASSWORD=msb\_pass  
POSTGRES\_DB=mustard  
POSTGRES\_PORT=5432  
BACKEND\_PORT=5000  
NGINX\_PORT=80  
DOMAIN=yourdomain.com  
EMAIL=you@example.com  
SECRET\_KEY=change\_this\_to\_a\_random\_value

### backup.sh

#!/usr/bin/env bash  
set -euo pipefail  
  
mkdir -p backups  
DATE=$(date +%Y%m%d)  
FILE="backups/mustard\_backup\_${DATE}.sql.gz"  
  
export $(grep -v '^#' .env | xargs)  
  
echo "📦 Dumping Postgres database..."  
docker-compose exec -T db pg\_dump -U "$POSTGRES\_USER" "$POSTGRES\_DB" | gzip > "$FILE"  
  
echo "🔐 Copying SSL certs snapshot..."  
tar -czf "backups/certs\_${DATE}.tar.gz" -C certbot/conf .  
  
echo "✅ Backup complete: $FILE and backups/certs\_${DATE}.tar.gz"

### restore.sh

#!/usr/bin/env bash  
set -euo pipefail  
  
if [ $# -lt 1 ]; then  
 echo "Usage: ./restore.sh <backup\_file.sql.gz>"  
 exit 1  
fi  
  
BACKUP\_FILE=$1  
  
if [ ! -f "$BACKUP\_FILE" ]; then  
 echo "❌ Backup file not found: $BACKUP\_FILE"  
 exit 1  
fi  
  
export $(grep -v '^#' .env | xargs)  
  
echo "⚠️ Restoring database from $BACKUP\_FILE ..."  
gunzip -c "$BACKUP\_FILE" | docker-compose exec -T db psql -U "$POSTGRES\_USER" "$POSTGRES\_DB"  
  
echo "✅ Restore complete."

## 🚀 Quick Recap

1. Clone repo → 2. Edit .env → 3. Run ./setup.sh → 4. Visit site → 5. Use ./status.sh → 6. Use ./reset.sh for rebuild → 7. Use ./backup.sh & ./restore.sh for safe data handling.

That’s it ✅ — deployment, reset, health check, and backups all simplified for non-technical users.