# DOCMOST — PostgreSQL / pgAdmin BACKUP & RESTORE CHEAT SHEET

Self-hosted GCP DockerLab • Goal: recover pages + attachments after a VM failure

## 0. KNOW WHAT YOU ARE BACKING UP

| Component | Protects | Backup |
|---|---|---|
| PostgreSQL | Pages, spaces, users, permissions, page data/history | **pgAdmin → Backup → Custom (.backup)** |
| Docmost storage | Uploaded files / attachments / images | Docker volume → .tar.gz |
| .env + compose | APP_SECRET, DB settings, URLs, configuration | Secure copy; encrypt |
| Redis | Supporting/cache state | Not the primary recovery target |

## 1. IDENTIFY THE REAL DB / VOLUMES

```
docker compose ps
cat docker-compose.yml
docker volume ls | grep docmost
docker volume inspect <DOCMOST_STORAGE_VOLUME>
```

Confirm DB name/user/password from compose or DATABASE_URL. Typical: database=**docmost**, port=**5432**. Do not change production settings.

## 2. CONNECT pgAdmin → POSTGRESQL

pgAdmin → Servers → Register → Server. General: **Docmost PostgreSQL**. Connection: Host=**db** (if pgAdmin is on the same Docker network), Port=**5432**, Maintenance DB=**docmost**, Username=**docmost**, Password=**your DB password**. **Do not use localhost** from inside the pgAdmin container.

```
SELECT current_database(); SELECT COUNT(*) FROM pages;
```

## 3. BACK UP POSTGRESQL IN pgAdmin

Servers → Docmost PostgreSQL → Databases → **docmost** → right-click → **Backup…**. General: Format=**Custom**; Filename=**/backup/docmost_YYYY-MM-DD.backup**. Data/Objects: keep **Pre-data + Data + Post-data**. Click **Backup**.

**IMPORTANT:** If pgAdmin runs in Docker, /backup is inside the container unless it is bind-mounted. Mount it to a GCP host folder so the file is downloadable.

```
mkdir -p ~/docmost-backups # pgAdmin volume mapping example: - ~/docmost-backups:/backup
```

## 4. VERIFY THE DB BACKUP

```
ls -lh ~/docmost-backups/ pg_restore --list ~/docmost-backups/docmost_YYYY-MM-DD.backup
```

A successful pgAdmin job + a non-zero archive + a populated pg_restore listing = much stronger verification.

## 5. BACK UP DOCMOST STORAGE (ATTACHMENTS)

```
docker volume ls | grep docmost mkdir -p ~/docmost-backups docker run --rm \ -v <DOCMOST_STORAGE_VOLUME>:/source:ro \ -v
~/docmost-backups:/backup alpine \ tar czf /backup/docmost-storage-YYYY-MM-DD.tar.gz -C /source .
```

Now you should have BOTH: **docmost_YYYY-MM-DD.backup** and **docmost-storage-YYYY-MM-DD.tar.gz**.

## 6. TEST RESTORE — NEVER TEST FIRST ON PRODUCTION

pgAdmin → Databases → Create database: **docmost_restore_test**. Right-click it → **Restore…**. Select the Custom .backup file. Keep **Pre-data + Data + Post-data**. Do not need “Create database” when restoring into the already-created test DB. Click Restore.

```
SELECT COUNT(*) FROM pages; SELECT id, title FROM pages ORDER BY created_at DESC LIMIT 20;
```

Compare page count/titles with production. A real restore test proves the backup is usable.

## 7. DISASTER-RECOVERY RESTORE ORDER

**NEW GCP VM → Docker/Compose → PostgreSQL → restore .backup → restore /app/data/storage → preserve original .env / APP_SECRET → start Docmost → verify pages + attachments.**

## 8. BACKUP PACKAGE / BEST PRACTICE

```
~/docmost-backups/ 2026-08-31/ docmost_2026-08-31.backup docmost-storage-2026-08-31.tar.gz docker-compose.yml .env # ENCRYPT / PROTECT
```

Keep multiple generations (e.g., weekly). Store at least one copy off the GCP VM. Do not rely on a single “latest” backup.

## 9. QUICK RULE

**PostgreSQL = your Docmost knowledge base. Storage volume = your attachments. .env/compose = the recipe to rebuild it.**