

# PostgreSQL Yedek & Kurtarma Referansı

pg\_dump, pg\_basebackup, WAL arşivi, pgBackRest, PITR ve felaket senaryoları için production rehberi

PostgreSQL 15-18

Güncelleme: Ekim 2026 · Production-grade



PG



RPO

Ne kadar veri? · kabul edilebilir kayıp



RTO

Ne kadar sürede? · geri dönüş süresi



KURAL

3-2-1-1-0 · 3 kopya, 2 ortam, 1 off-site



+1 / +0

immutable · 0 hatalı restore testi



ALTIN KURAL

Test edilmemiş yedek = yedek değildir

## 01 Temel Kavramlar & Strateji



| KAVRAM             | AÇIKLAMA                                                                                                                                |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Mantıksal          | SQL / nesne seviyesinde döküm. Sürüm & platform bağımsız, seçici restore kolay; büyük veride yavaş restore (indeksler yeniden kurulur). |
| Fiziksel           | Veri dosyalarının blok kopyası. Hızlı restore, TB ölçeğe uygun; aynı ana sürüm + platform gerekir.                                      |
| Snapshot           | LVM / ZFS / EBS / SAN anlık görüntüsü. Crash-consistent; WAL/redo aynı volume'deyse ya da backup modu kullanılırsa güvenli.             |
| Full / Incr / Diff | Tamamı / son yedekten beri / son full'den beri değişenler. Incremental zinciri uzadıkça restore süresi ve risk artar.                   |
| Hot / Warm / Cold  | Okuma+yazma açık / sadece okuma / sunucu kapalı. Production'da hedef: hot.                                                              |
| Sürekli Arşiv      | Base backup + kesintisiz WAL akışı → PITR: herhangi bir ana geri dönüş. RPO'yu saniyelere indirir.                                      |

### YÖNTEM SEÇİMİ

| SENARYO                          | ÖNERİ                             |
|----------------------------------|-----------------------------------|
| < 100 GB, sürüm geçişi, tek şema | pg_dump -Fd -j N                  |
| TB ölçek, kısa RTO               | pgBackRest / pg_basebackup + WAL  |
| Saniyelik RPO                    | Streaming replika + pg_receivewal |
| "Yanlışlıkla sildim" koruması    | PITR + recovery_min_apply_delay   |

### ARAÇ HARİTASI

|           |                                              |
|-----------|----------------------------------------------|
| Mantıksal | pg_dump · pg_dumpall · pg_restore            |
| Fiziksel  | pg_basebackup · pg_combinebackup (17+)       |
| WAL       | archive_command · pg_receivewal · pg_waldump |
| Kurumsal  | pgBackRest · Barman · WAL-G                  |

⚠ Replikasyon yedek değildir: DROP TABLE veya hatalı UPDATE milisaniyeler içinde tüm standby'lara da gider. HA ≠ DR.

## 02 Mantıksal Yedek



| PG_DUMP FORMATI | RESTORE ARACI | PARALEL        | SEÇİCİ |
|-----------------|---------------|----------------|--------|
| -Fp plain       | psql          | x              | x      |
| -Fc custom      | pg_restore    | restore        | ✓      |
| -Fd directory   | pg_restore    | dump + restore | ✓      |
| -Ft tar         | pg_restore    | x              | ✓      |

### PARALEL DÖKÜM & GLOBAL NESNELER

```
# 8 paralel iş, zstd sıkıştırma (PG16+)
$ pg_dump -h db1 -U backup -Fd -j 8 -Z zstd:3 \
  -f /bk/shop_$(date +%F) shop
# Roller, tablespace'ler, parametreler
$ pg_dumpall -h db1 -U postgres --globals-only \
  > /bk/globals_$(date +%F).sql
```

### PARALEL & SEÇİCİ RESTORE

```
$ psql -f globals.sql postgres
$ createdb -O app shop
$ pg_restore -j 8 -d shop --no-owner /bk/shop_dir
# Yalnızca tek tablo:
$ pg_restore -l /bk/shop_dir > list.txt # düzenle
$ pg_restore -L list.txt -d shop /bk/shop_dir
$ pg_restore -t orders -d shop /bk/shop_dir
```

**Tutarlılık** Tek snapshot (REPEATABLE READ) — yazmaları bloklamaz; uzun dump vacuum'u geciktirir.

**Sürüm Geçişi** Her zaman hedef sürümün pg\_dump binary'sini kullan.

⚠ Dikkat: pg\_dump rolleri ve tablespace'leri içermez. Mutlaka pg\_dumpall --globals-only ile birlikte al — yoksa restore'da "role does not exist" hataları gelir.

## 03 Fiziksel Yedek & WAL Arşivi



| PARAMETRE       | DEĞER / NOT                                              |
|-----------------|----------------------------------------------------------|
| wal_level       | replica (varsayılan) — PITR için yeterli                 |
| archive_mode    | on — restart gerekir                                     |
| archive_command | Yalnızca başarıda 0 dönmeli; ya da archive_library (15+) |
| archive_timeout | 60s — düşük trafikte RPO üst sınırı                      |
| summarize_wal   | on — incremental backup için (17+)                       |

### BASE BACKUP + DOĞRULAMA

```
archive_command = 'test ! -f /arch/%f && cp %p /arch/%f'

$ pg_basebackup -h db1 -U repl -D /bk/base_$(date +%F) \
  -Ft -z -Xs -P -c fast
$ pg_verifybackup /bk/base_2026-10-13 # manifest (13+)
```

### NATIVE INCREMENTAL (PG 17+)

```
$ pg_basebackup -D /bk/incr1 \
  --incremental=/bk/full/backup_manifest
# restore öncesi zinciri birleştir:
$ pg_combinebackup /bk/full /bk/incr1 /bk/incr2 -o /restore
```

| ARAÇ       | ÖNE ÇIKAN                                                             |
|------------|-----------------------------------------------------------------------|
| pgBackRest | Full/diff/incr, paralel, sıkıştırma + şifreleme, S3/GCS/Azure, verify |
| Barman     | Merkezi yedek sunucusu, streaming WAL, retention policy               |
| WAL-G      | Bulut odaklı, delta backup, hafif kurulum                             |

⚠ Disk Dolma Riski: archive\_command başarısız oldukça WAL'ler pg\_wal'da birikir → disk dolar → sunucu durur. pg\_stat\_archiver.failed\_count için alarm kur.

04

Point-in-Time Recovery

KRİTİK

ADIM ADIM PITR

1

Sunucuyu durdur: `systemctl stop postgresql`

2

Eski data dizinini **silme, taşı** — `pg_wal` içindeki arşivlenmemiş WAL'ler gerekebilir.

3

Base backup'ı aç (tar ise `base.tar.gz` + `pg_wal.tar.gz`), sahiplik: `postgres`.

4

`postgresql.conf`'a `restore_command` + hedef parametrelerini yaz.

5

`touch $PGDATA/recovery.signal` → başlat, logu izle.

6

Hedefte veriyi kontrol et → `SELECT pg_wal_replay_resume()`; ile promote.

📖

PostgreSQL.conf (pg 12+)

```
restore_command      = 'cp /arch/%f %p'
recovery_target_time  = '2026-10-13 09:59:59+03'
recovery_target_inclusive = off
recovery_target_action = 'pause'      # pause|promote|shutdown
recovery_target_timeline = 'latest'   # varsayılan
```

HEDEF

NE ZAMAN?

recovery\_target\_time

Olay zamanı biliniyorsa (saat dilimini yaz!)

recovery\_target\_xid

Hatalı transaction ID'si biliniyorsa

recovery\_target\_lsn

`pg_waldump` ile bulunan kesin WAL konumu

recovery\_target\_name

`pg_create_restore_point()` ile işaretlenen nokta

recovery\_target

'immediate' — tutarlı olunca hemen dur

⚡ DEPLOY ÖNCESİ İŞARET & PGBACKREST İLE PITR

```
SELECT pg_create_restore_point('before_deploy_v42');
$ pgbackrest --stanza=main --delta --type=time \
  --target="2026-10-13 09:59:59+03" \
  --target-action=promote restore
```

⚠ Dikkat:

Promote sonrası **yeni timeline** başlar. Kurtarılan sunucudan hemen **yeni bir base backup** al; eski zincir yeni timeline'ı kapsamaz.

05

Doğrulama, Test & İzleme

KONTROL

ARAÇ / YÖNTEM

Yedek Bütünlüğü

`pg_verifybackup` (manifest), `pgbackrest` verify

Veri Bütünlüğü

`Data checksums`, `pg_amcheck` (14+), `pg_checksums --check`

Arşiv Sağlığı

`pg_stat_archiver`, `.ready` dosya sayısı

Restore Testi

Otomatik, en az haftalık; ayrı sunucuda geri yükle → sorgu ile doğrula → **RTO'yu ölç** ve kaydet.

💖 WAL ARŞİVİ SAĞLIK KONTROLÜ

```
SELECT archived_count, last_archived_wal,
       now() - last_archived_time AS since_last,
       failed_count, last_failed_wal, last_failed_time
FROM   pg_stat_archiver;
-- Arşivlenmeyi bekleyen WAL sayısı:
SELECT count(*) FROM pg_ls_archive_statusdir()
WHERE  name LIKE '%.ready';
```

🔧 PGBACKREST & BÜTÜNLÜK KONTROLLERİ

```
$ pgbackrest --stanza=main check # arşiv + config testi
$ pgbackrest --stanza=main info  # yedekler & WAL analığı
$ pgbackrest --stanza=main verify
$ pg_amcheck -d shop --heapallindexed -j 4
```

🔴 ALARM KUR

Son başarılı yedeğin yaşı > 26 saat

KRİTİK

`failed_count` artışı · `.ready` birikimi

KRİTİK

`now()` - `last_archived_time` > RPO

KRİTİK

Yedek boyutunda ani ±%30 değişim

ÖNEMLİ

`pg_wal` / yedek disk doluluğu > %80

ÖNEMLİ

📌 PG 18: `initdb` artık data checksum'ları **varsayılan olarak açık** başlatır. Eski kümelerde: `pg_checksums --enable` (sunucu kapalıyken).

06

Felaket Senaryoları & Kontrol Listesi

SENARYO

ÇÖZÜM

Kazara DROP / DELETE

PITR (`recovery_target_xid` / `_time`) veya gecikmeli standby'dan `pg_dump -t`

Disk / sunucu kaybı

Standby promote (`pg_promote()`) → yeni standby: `pgbackrest --type=standby restore`

Fidye yazılımı

Immutable repo (S3 object lock) + `repo-cipher-type` + ayrı kimlik bilgileri

Bozuk sayfa

Checksum / `pg_amcheck` ile tespit → etkilenen nesneyi yedekten geri al

Bölge (region) kaybı

`pgBackRest` çoklu repo (`repo2` başka bölgede) + DR runbook

🔁 GEÇİKMELİ REPLİKA (1 SAAT)

```
# standby: postgresql.conf
recovery_min_apply_delay = '1h'
-- olay anında: replay'i durdur, veriyi kurtar
SELECT pg_wal_replay_pause();
```

☑

RPO / RTO yazılı ve iş birimiyle onaylı

SLA → yöntem seçimi

KRİTİK

☑

WAL arşivi açık ve izleniyor

`archive_mode` · `pg_stat_archiver`

KRİTİK

☑

Otomatik restore testi

haftalık · RTO ölçümü

KRİTİK

☑

Off-site + şifreli kopya

`repo-cipher-type` · S3 object lock

KRİTİK

☑

Retention politikası uygulayanıor

`repo1-retention-full` · `repo1-retention-diff`

ÖNEMLİ

☑

Global nesneler yedekte

`pg_dumpall --globals-only`

ÖNEMLİ

☑

Config dosyaları versiyonlu

`postgresql.conf` · `pg_hba.conf` · `pgbackrest.conf`

ÖNERİLEN

☑

DR runbook güncel ve tatbik edilmiş

yılda 2 tatbikat

ÖNERİLEN

KRİTİK

hemen

ÖNEMLİ

bu çeyrek

ÖNERİLEN

planla