PostgreSQL Optimization Guide for Lemmy

Quick reference for optimizing PostgreSQL performance on Lemmy instances.

1. Find Your Config File

```
sudo -u postgres psql -c "SHOW config_file;"
```

2. Check Current Settings

```
SELECT name, setting, unit
FROM pg_settings
WHERE name IN ('shared_buffers', 'work_mem', 'effective_cache_size', 'maintenance
```

3. Memory Settings

Edit `postgresql.conf` or use `ALTER SYSTEM`:

```
-- For 8GB RAM system:
ALTER SYSTEM SET shared_buffers = '2GB';           -- 25% of RAM
ALTER SYSTEM SET effective_cache_size = '6GB';     -- 75% of RAM
ALTER SYSTEM SET maintenance_work_mem = '512MB';
ALTER SYSTEM SET work_mem = '16MB';

-- For 16GB RAM system:
ALTER SYSTEM SET shared_buffers = '4GB';
ALTER SYSTEM SET effective_cache_size = '12GB';
ALTER SYSTEM SET maintenance_work_mem = '1GB';
ALTER SYSTEM SET work_mem = '32MB';
```

4. Autovacuum Tuning (Most Important)

```
ALTER SYSTEM SET autovacuum = on;
ALTER SYSTEM SET autovacuum_naptime = '30s';
ALTER SYSTEM SET autovacuum_max_workers = 4;
ALTER SYSTEM SET autovacuum_vacuum_threshold = 50;
```

```
ALTER SYSTEM SET autovacuum_vacuum_scale_factor = 0.05;
ALTER SYSTEM SET autovacuum_analyze_threshold = 50;
ALTER SYSTEM SET autovacuum_analyze_scale_factor = 0.025;
ALTER SYSTEM SET autovacuum_vacuum_cost_delay = '2ms';
ALTER SYSTEM SET autovacuum_vacuum_cost_limit = 1000;
ALTER SYSTEM SET autovacuum_work_mem = '512MB';
```

5. WAL Settings

```
ALTER SYSTEM SET max_wal_size = '2GB';
ALTER SYSTEM SET min_wal_size = '1GB';
ALTER SYSTEM SET checkpoint_completion_target = 0.9;
```

6. Apply Changes

```
# Reload config (for most settings)
sudo -u postgres psql -c "SELECT pg_reload_conf();"

# Restart required for some settings (shared_buffers, max_wal_size)
sudo systemctl restart postgresql
```

7. Run Manual VACUUM

```
-- Connect to Lemmy database
\c lemmy

-- Vacuum all tables
VACUUM ANALYZE;

-- Or target the activity table specifically
VACUUM ANALYZE activity;
```

8. Monitor Performance

```
-- Check autovacuum activity
SELECT schemaname, relname, n_dead_tup, n_live_tup,
       round(100.0 * n_dead_tup / NULLIF(n_live_tup + n_dead_tup, 0), 2) AS dead_
       last_vacuum, last_autovacuum
FROM pg_stat_user_tables
WHERE n_dead_tup > 0
```

```
ORDER BY n_dead_tup DESC LIMIT 10;

-- Check table sizes
SELECT schemaname, tablename,
       pg_size_pretty(pg_total_relation_size(schemaname||'.'||tablename)) AS size
FROM pg_tables
WHERE schemaname NOT IN ('pg_catalog', 'information_schema')
ORDER BY pg_total_relation_size(schemaname||'.'||tablename) DESC LIMIT 10;
```

Docker Compose Method

Add to your `docker-compose.yml`:

```
postgres:
  image: postgres:15
  command:
    - "postgres"
    - "-c"
    - "shared_buffers=2GB"
    - "-c"
    - "effective_cache_size=6GB"
    - "-c"
    - "maintenance_work_mem=512MB"
    - "-c"
    - "autovacuum_max_workers=4"
    - "-c"
    - "autovacuum_vacuum_scale_factor=0.05"
```

Quick Config Generator

Use PGTune for customized settings based on your hardware.

Priority Order

1. Autovacuum tuning (biggest impact)
2. Memory settings
3. Manual VACUUM ANALYZE
4. Monitor for a few days
5. Fine-tune based on results

Note: Adjust RAM values based on your system. These examples assume dedicated database servers.