LiteLLM SQLAlchemy POC - Setup and Testing Runbook

Version: 1.0
Date: 2025-09-26
Environment: POC/Development

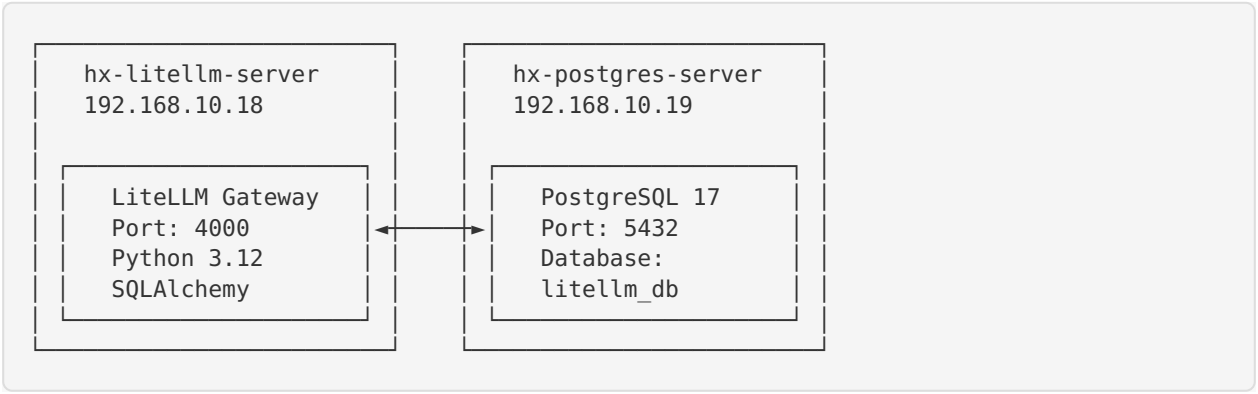
Overview

This runbook provides step-by-step instructions for setting up and testing the LiteLLM Gateway with PostgreSQL backend using SQLAlchemy. The POC demonstrates replacing Prisma with SQLAlchemy + PostgreSQL for request/response logging.

Prerequisites

- Two Ubuntu 24.04 VMs on `192.168.10.0/24` network
- Network connectivity between VMs
- Root/sudo access on both VMs
- Internet connectivity for package installation

Target Architecture



Part 1: PostgreSQL Server Setup (hx-postgres-server - 192.168.10.19)

Step 1.1: Install PostgreSQL 17

```
# Update system packages
sudo apt update && sudo apt upgrade -y

# Install required packages
sudo apt install -y wget ca-certificates

# Add PostgreSQL official APT repository
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key ad
d -
echo "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -cs)-pgdg main" | sud
o tee /etc/apt/sources.list.d/pgdg.list

# Update package list and install PostgreSQL 17
sudo apt update
sudo apt install -y postgresql-17 postgresql-client-17 postgresql-contrib-17

# Verify installation
sudo systemctl status postgresql
psql --version
```

Step 1.2: Configure PostgreSQL

```
# Switch to postgres user
sudo -i -u postgres

# Configure PostgreSQL settings
psql <<'SQL'
-- Enable SCRAM-SHA-256 authentication
ALTER SYSTEM SET password_encryption = 'scram-sha-256';

-- Create database
CREATE DATABASE litellm_db WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'C'
    LC_CTYPE = 'C'
    TEMPLATE = template0;

-- Create user if not exists
DO $$
BEGIN
    IF NOT EXISTS (SELECT FROM pg_roles WHERE rolname = 'litellm_user') THEN
        CREATE ROLE litellm_user LOGIN;
    END IF;
END$$;

-- Set strong password (replace with actual strong password)
ALTER ROLE litellm_user WITH PASSWORD 'LiteLLM_P0C_2025!SecurePass';

-- Grant permissions
GRANT CONNECT ON DATABASE litellm_db TO litellm_user;
GRANT CREATE ON DATABASE litellm_db TO litellm_user;

-- Connect to litellm_db and grant schema permissions
\c litellm_db
GRANT CREATE ON SCHEMA public TO litellm_user;
GRANT USAGE ON SCHEMA public TO litellm_user;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO litellm_user;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO litellm_user;

-- Set default privileges for future objects
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL ON TABLES TO litellm_user;
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL ON SEQUENCES TO litellm_user;

-- Verify setup
\l
\du
SQL

# Exit postgres user
exit
```

Step 1.3: Configure Network Access

```
# Configure PostgreSQL to listen on all addresses
sudo sed -i "s/#listen_addresses = 'localhost'/listen_addresses = '*'/" /etc/postgr-
esql/17/main/postgresql.conf

# Add client authentication rule for LiteLLM server
echo "host    litellm_db    litellm_user    192.168.10.18/32    scram-sha-256" | sudo
tee -a /etc/postgresql/17/main/pg_hba.conf

# Restart PostgreSQL to apply changes
sudo systemctl restart postgresql

# Verify PostgreSQL is running and listening
sudo systemctl status postgresql
sudo netstat -tlnp | grep 5432
```

Step 1.4: Test Local Connection

```
# Test connection as litellm_user
psql "host=localhost dbname=litellm_db user=litellm_user" -c "SELECT version();"

# Test connection from external IP (simulate remote connection)
psql "host=192.168.10.19 dbname=litellm_db user=litellm_user sslmode=disable" -c "SE-
LECT current_database(), current_user;"
```

Part 2: LiteLLM Gateway Server Setup (hx-litellm-server - 192.168.10.18)

Step 2.1: System Preparation

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install Python 3.12 and development tools
sudo apt install -y python3.12 python3.12-venv python3.12-dev python3-pip
sudo apt install -y build-essential libpq-dev curl wget git

# Create litellm user and directories
sudo useradd -r -m -s /bin/bash litellm
sudo mkdir -p /opt/litellm /etc/litellm /var/log/litellm
sudo chown litellm:litellm /opt/litellm /var/log/litellm
sudo chown root:litellm /etc/litellm
sudo chmod 750 /etc/litellm
```

Step 2.2: Python Environment Setup

```
# Switch to litellm user
sudo -u litellm -i

# Create Python virtual environment
cd /opt/litellm
python3.12 -m venv venv
source venv/bin/activate

# Upgrade pip and install required packages
pip install --upgrade pip setuptools wheel

# Install LiteLLM and dependencies
pip install litellm[proxy]
pip install sqlalchemy psycopg2-binary
pip install python-dotenv pydantic

# Verify installations
pip list | grep -E "(litellm|sqlalchemy|psycopg2)"
python -c "import litellm, sqlalchemy, psycopg2; print('All imports successful')"

# Exit litellm user
exit
```

Step 2.3: Database Connectivity Test

```
# Test PostgreSQL connection from LiteLLM server
psql "host=192.168.10.19 dbname=litellm_db user=litellm_user sslmode=disable" -c "SE-
LECT version();"

# If connection fails, check network connectivity
ping -c 3 192.168.10.19
telnet 192.168.10.19 5432
```

Step 2.4: Deploy POC Files

```
# Copy POC files to appropriate locations
sudo cp ./poc-files/db_init.py /opt/litellm/
sudo cp ./poc-files/config.yaml /etc/litellm/

# Update file ownership
sudo chown litellm:litellm /opt/litellm/db_init.py
sudo chown root:litellm /etc/litellm/config.yaml
sudo chmod 640 /etc/litellm/config.yaml
sudo chmod 755 /opt/litellm/db_init.py
```

Step 2.5: Configure Environment Variables

```
# Create environment file
sudo tee /etc/litellm/litellm.env > /dev/null <<'EOF'
# LiteLLM Environment Configuration
LITELLM_DB_URL=postgresql+psycopg2://litellm_user:LiteLLM_P0C_2025!
SecurePass@192.168.10.19/litellm_db
PYTHONUNBUFFERED=1
LITELLM_LOG_LEVEL=INFO

# OpenAI API Key (replace with actual key)
OPENAI_API_KEY=REDACTED_MODEL_API_KEY

# LiteLLM Master Key (replace with secure key)
LITELLM_MASTER_KEY=REDACTED_MASTER_KEY
EOF

# Set proper permissions
sudo chown root:litellm /etc/litellm/litellm.env
sudo chmod 640 /etc/litellm/litellm.env
```

Step 2.6: Initialize Database Schema

```
# Run database initialization as litellm user
sudo -u litellm -i
cd /opt/litellm
source venv/bin/activate

# Set environment variable
export LITELLM_DB_URL="postgresql+psycopg2://litellm_user:LiteLLM_P0C_2025!Secure-
Pass@192.168.10.19/litellm_db"

# Run database initialization
python db_init.py

# Verify tables were created
psql "$LITELLM_DB_URL" -c "\dt"
psql "$LITELLM_DB_URL" -c "\d requests"
psql "$LITELLM_DB_URL" -c "\d responses"

# Exit litellm user
exit
```

Step 2.7: Create Systemd Service

```
# Create systemd service file
sudo tee /etc/systemd/system/litellm-gateway.service > /dev/null <<'EOF'
[Unit]
Description=LiteLLM Gateway with PostgreSQL Backend
After=network-online.target
Wants=network-online.target
Requires=network.target

[Service]
Type=simple
User=litellm
Group=litellm
WorkingDirectory=/opt/litellm
Environment="PYTHONUNBUFFERED=1"
EnvironmentFile=/etc/litellm/litellm.env
ExecStart=/opt/litellm/venv/bin/litellm --config /etc/litellm/config.yaml
ExecReload=/bin/kill -HUP $MAINPID
Restart=on-failure
RestartSec=5
TimeoutStartSec=300
TimeoutStopSec=30

# Security settings
NoNewPrivileges=true
PrivateTmp=true
ProtectSystem=strict
ProtectHome=true
ReadWritePaths=/var/log/litellm /tmp

# Resource limits
LimitNOFILE=65536
LimitNPROC=4096

[Install]
WantedBy=multi-user.target
EOF

# Reload systemd and enable service
sudo systemctl daemon-reload
sudo systemctl enable litellm-gateway
```

Part 3: Service Startup and Validation

Step 3.1: Start LiteLLM Gateway

```
# Start the service
sudo systemctl start litellm-gateway

# Check service status
sudo systemctl status litellm-gateway

# Monitor logs for startup
sudo journalctl -u litellm-gateway -f

# Check if service is listening on port 4000
sudo netstat -tlnp | grep 4000
curl -s http://192.168.10.18:4000/health
```

Step 3.2: Verify Database Connectivity

```
# Check logs for database connection messages
sudo journalctl -u litellm-gateway -n 100 | grep -i -E "(database|connection|postgr-
esql)"

# Verify database connection from application perspective
sudo -u litellm -i
cd /opt/litellm
source venv/bin/activate
export LITELLM_DB_URL="postgresql+psycopg2://litellm_user:LiteLLM_P0C_2025!Secure-
Pass@192.168.10.19/litellm_db"
python -c "
from sqlalchemy import create_engine, text
engine = create_engine('$LITELLM_DB_URL')
with engine.connect() as conn:
    result = conn.execute(text('SELECT current_database(), current_user, version()'))
    print('Database connection successful:', result.fetchone())
"
exit
```

Part 4: API Testing and Validation

Step 4.1: Basic Health Check

```
# Test health endpoint
curl -s http://192.168.10.18:4000/health | jq .

# Test metrics endpoint (if enabled)
curl -s http://192.168.10.18:4000/metrics
```

Step 4.2: API Request Testing

```
# Test chat completions endpoint
curl -X POST http://192.168.10.18:4000/v1/chat/completions \
  -H 'Content-Type: application/json' \
  -H 'Authorization: Bearer TEST_KEY' \
  -d '{
    "model": "gpt-4o-mini",
    "messages": [
      {
        "role": "user",
        "content": "Hello! This is a test message for the LiteLLM SQLAlchemy POC."
      }
    ],
    "max_tokens": 100,
    "temperature": 0.7
  }' | jq .

# Test with different API key
curl -X POST http://192.168.10.18:4000/v1/chat/completions \
  -H 'Content-Type: application/json' \
  -H 'Authorization: Bearer POC_VALIDATION_KEY' \
  -d '{
    "model": "gpt-4o-mini",
    "messages": [
      {
        "role": "user",
        "content": "What is SQLAlchemy and how does it compare to Prisma?"
      }
    ],
    "max_tokens": 150
  }' | jq .
```

Step 4.3: Database Logging Verification

```
# Check if requests are being logged to database
psql "postgresql+psycopg2://litellm_user:LiteLLM_P0C_2025!SecurePass@192.168.10.19/li-
tellm_db" <<'SQL'
-- Check requests table
SELECT
    id,
    created_at,
    request_id,
    route,
    model,
    status_code,
    user_id,
    api_key
FROM requests
ORDER BY created_at DESC
LIMIT 5;

-- Check responses table
SELECT
    r.id,
    r.created_at,
    r.latency_ms,
    r.prompt_tokens,
    r.completion_tokens,
    r.total_tokens,
    req.request_id,
    req.model
FROM responses r
JOIN requests req ON r.request_id_fk = req.id
ORDER BY r.created_at DESC
LIMIT 5;

-- Get summary statistics
SELECT
    COUNT(*) as total_requests,
    COUNT(DISTINCT model) as unique_models,
    AVG(CASE WHEN status_code = 200 THEN 1.0 ELSE 0.0 END) * 100 as success_rate
FROM requests;
SQL
```

Step 4.4: Error Testing

```
# Test with invalid API key
curl -X POST http://192.168.10.18:4000/v1/chat/completions \
  -H 'Content-Type: application/json' \
  -H 'Authorization: Bearer INVALID_KEY' \
  -d '{
    "model": "gpt-4o-mini",
    "messages": [{"role": "user", "content": "Test"}]
  }'

# Test with invalid model
curl -X POST http://192.168.10.18:4000/v1/chat/completions \
  -H 'Content-Type: application/json' \
  -H 'Authorization: Bearer TEST_KEY' \
  -d '{
    "model": "invalid-model",
    "messages": [{"role": "user", "content": "Test"}]
  }'

# Verify error requests are also logged
psql "postgresql+psycopg2://litellm_user:LiteLLM_P0C_2025!SecurePass@192.168.10.19/li-
tellm_db" -c "
SELECT request_id, route, model, status_code, created_at
FROM requests
WHERE status_code != 200
ORDER BY created_at DESC
LIMIT 3;"
```

Part 5: Performance and Load Testing

Step 5.1: Concurrent Request Testing

```
# Install Apache Bench for load testing
sudo apt install -y apache2-utils

# Create test payload file
cat > /tmp/test_payload.json <<'EOF'
{
  "model": "gpt-4o-mini",
  "messages": [
    {
      "role": "user",
      "content": "This is a load test message. Please respond briefly."
    }
  ],
  "max_tokens": 50
}
EOF

# Run concurrent requests test
ab -n 10 -c 2 -T 'application/json' -H 'Authorization: Bearer TEST_KEY' -p /tmp/
test_payload.json http://192.168.10.18:4000/v1/chat/completions

# Monitor database during load test
psql "postgresql+psycopg2://litellm_user:LiteLLM_P0C_2025!SecurePass@192.168.10.19/li-
tellm_db" -c "
SELECT
    COUNT(*) as total_requests,
    MIN(created_at) as first_request,
    MAX(created_at) as last_request,
    COUNT(CASE WHEN status_code = 200 THEN 1 END) as successful_requests
FROM requests
WHERE created_at > NOW() - INTERVAL '5 minutes';"
```

Step 5.2: Database Performance Check

```
# Check database connection pool status
sudo journalctl -u litellm-gateway -n 50 | grep -i pool

# Monitor PostgreSQL activity
psql "postgresql+psycopg2://litellm_user:LiteLLM_P0C_2025!SecurePass@192.168.10.19/li-
tellm_db" -c "
SELECT
    datname,
    numbackends,
    xact_commit,
    xact_rollback,
    blks_read,
    blks_hit,
    tup_returned,
    tup_fetched,
    tup_inserted,
    tup_updated
FROM pg_stat_database
WHERE datname = 'litellm_db';"
```

Part 6: Troubleshooting Guide

Common Issues and Solutions

Issue 1: Service Won't Start

```
# Check service status and logs
sudo systemctl status litellm-gateway
sudo journalctl -u litellm-gateway -n 50

# Common causes:
# 1. Database connection issues
# 2. Missing environment variables
# 3. Port already in use
# 4. Permission issues

# Check port availability
sudo netstat -tlnp | grep 4000

# Check file permissions
ls -la /etc/litellm/
ls -la /opt/litellm/
```

Issue 2: Database Connection Failures

```
# Test direct database connection
psql "host=192.168.10.19 dbname=litellm_db user=litellm_user sslmode=disable" -c "SE-
LECT 1;"

# Check PostgreSQL logs
sudo tail -f /var/log/postgresql/postgresql-17-main.log

# Verify pg_hba.conf configuration
sudo cat /etc/postgresql/17/main/pg_hba.conf | grep litellm

# Check network connectivity
ping 192.168.10.19
telnet 192.168.10.19 5432
```

Issue 3: API Requests Failing

```
# Check LiteLLM logs
sudo journalctl -u litellm-gateway -f

# Verify API key configuration
grep -A 5 "static_api_keys" /etc/litellm/config.yaml

# Test with curl verbose mode
curl -v -X POST http://192.168.10.18:4000/v1/chat/completions \
  -H 'Content-Type: application/json' \
  -H 'Authorization: Bearer TEST_KEY' \
  -d '{"model":"gpt-4o-mini","messages":[{"role":"user","content":"test"}]}'
```

Issue 4: Database Logging Not Working

```
# Check if tables exist
psql "postgresql+psycopg2://litellm_user:LiteLLM_P0C_2025!SecurePass@192.168.10.19/litellm_db" -c "\dt"

# Verify table structure
psql "postgresql+psycopg2://litellm_user:LiteLLM_P0C_2025!SecurePass@192.168.10.19/litellm_db" -c "\d requests"

# Check LiteLLM configuration
grep -A 10 "telemetry" /etc/litellm/config.yaml
grep -A 10 "database" /etc/litellm/config.yaml

# Re-run database initialization
sudo -u litellm -i
cd /opt/litellm
source venv/bin/activate
export LITELLM_DB_URL="postgresql+psycopg2://litellm_user:LiteLLM_P0C_2025!SecurePass@192.168.10.19/litellm_db"
python db_init.py
exit
```

Part 7: Monitoring and Maintenance

Log Monitoring

```
# Monitor LiteLLM Gateway logs
sudo journalctl -u litellm-gateway -f

# Monitor PostgreSQL logs
sudo tail -f /var/log/postgresql/postgresql-17-main.log

# Check disk space
df -h
du -sh /var/log/litellm/
du -sh /var/log/postgresql/
```

Database Maintenance

```
# Connect to database for maintenance
psql "postgresql+psycopg2://litellm_user:LiteLLM_P0C_2025!SecurePass@192.168.10.19/li-
tellm_db"

-- Check table sizes
SELECT
    schemaname,
    tablename,
    attname,
    n_distinct,
    correlation
FROM pg_stats
WHERE schemaname = 'public';

-- Analyze tables for performance
ANALYZE requests;
ANALYZE responses;

-- Check for long-running queries
SELECT
    pid,
    now() - pg_stat_activity.query_start AS duration,
    query
FROM pg_stat_activity
WHERE (now() - pg_stat_activity.query_start) > interval '5 minutes';
```

Backup Procedures

```
# Create database backup
pg_dump "postgresql://litellm_user:LiteLLM_P0C_2025!SecurePass@192.168.10.19/li-
tellm_db" > litellm_backup_$(date +%Y%m%d_%H%M%S).sql

# Create configuration backup
sudo tar -czf litellm_config_backup_$(date +%Y%m%d_%H%M%S).tar.gz /etc/litellm/ /opt/
litellm/db_init.py
```

Part 8: Cleanup Procedures

Stop Services

```
# Stop LiteLLM Gateway
sudo systemctl stop litellm-gateway
sudo systemctl disable litellm-gateway

# Remove systemd service
sudo rm /etc/systemd/system/litellm-gateway.service
sudo systemctl daemon-reload
```

Remove Installation

```
# Remove LiteLLM installation
sudo rm -rf /opt/litellm
sudo rm -rf /etc/litellm
sudo rm -rf /var/log/litellm
sudo userdel -r litellm

# Remove PostgreSQL (if needed)
sudo systemctl stop postgresql
sudo apt remove --purge postgresql-17 postgresql-client-17 postgresql-contrib-17
sudo rm -rf /etc/postgresql/17/
sudo rm -rf /var/lib/postgresql/17/
```

Success Criteria Checklist

- [ ] PostgreSQL 17 installed and configured
- [ ] Database `litellm_db` created with proper user permissions
- [ ] Network connectivity established between servers
- [ ] LiteLLM Gateway service starts successfully
- [ ] Database schema initialized with `requests` and `responses` tables
- [ ] API requests return HTTP 200 with valid responses
- [ ] Request/response data logged to PostgreSQL database
- [ ] Error handling works correctly
- [ ] Service runs as systemd unit
- [ ] All configuration files properly secured
- [ ] Documentation complete and accurate

Next Steps for Production

1. **Security Hardening**
   - Implement TLS/SSL encryption
   - Configure firewall rules
   - Use secrets management system
   - Enable audit logging

2. **High Availability**
   - Set up PostgreSQL replication
   - Implement load balancing
   - Configure backup and recovery procedures

3. **Monitoring**
   - Set up Prometheus/Grafana monitoring
   - Configure alerting rules
   - Implement health checks

4. **Performance Optimization**
   - Tune PostgreSQL configuration
   - Optimize connection pooling
   - Implement caching strategies

End of Runbook