

# Docker Infrastructure Documentation

---

# Overview

---

This document describes the enhanced Docker infrastructure setup for the PowerShell Script Analysis Platform, including connection pooling, high availability, and automated backups.

# Architecture Components

---

## 1. PgBouncer Connection Pooling

**Purpose:** Manages PostgreSQL connections efficiently to prevent connection exhaustion and improve performance.

**Configuration:** - Port: 6432 - Pool Mode: Transaction - Max Client Connections: 1000 - Default Pool Size: 25 connections per database - Reserve Pool: 5 connections

**Configuration Files:** - `/docker/pgbouncer/pgbouncer.ini` - Main configuration - `/docker/pgbouncer/userlist.txt` - User authentication

**Benefits:** - Reduces database connection overhead - Prevents connection exhaustion - Improves application scalability - Faster connection establishment

**Usage:**

```
# Connect via pgBouncer instead of direct PostgreSQL
DB_HOST=pgbouncer
DB_PORT=6432
```

## 2. Redis Cluster with Sentinel

**Purpose:** Provides high availability for caching and session storage with automatic failover.

**Architecture:** - 1 Redis Master (port 6379) - 2 Redis Replicas (ports 6380, 6381) - 3 Redis Sentinels (ports 26379, 26380, 26381)

**Features:** - Automatic failover detection - Master election on failure - Data replication across replicas - Quorum-based decision making (2/3 sentinels)

**Configuration Files:** - `/docker/redis/redis-master.conf` - Master configuration - `/docker/redis/redis-replica.conf` - Replica configuration - `/docker/redis/sentinel.conf` - Sentinel monitoring configuration

**Failover Process:** 1. Sentinel detects master failure (5 second timeout) 2. Quorum agreement among sentinels (2 out of 3) 3. Automatic promotion of replica to master 4. Client reconnection to new master

### 3. Backup Automation Service

**Purpose:** Automated backup, retention, and disaster recovery for PostgreSQL and Redis.

**Features:** - Scheduled full and incremental backups - Automated cleanup based on retention policy - Optional S3 cloud storage integration - Health monitoring and alerting - Backup verification and metadata tracking

**Backup Schedules** (configurable via environment variables): - PostgreSQL Full Backup: Daily at 2 AM - PostgreSQL Incremental: Every 6 hours - Redis Snapshot: Every 4 hours - Cleanup Old Backups: Daily at 3 AM - Health Checks: Every 5 minutes

**Backup Scripts:** - `postgres-backup.sh` - PostgreSQL backup (full/incremental) - `redis-backup.sh` - Redis snapshot backup - `cleanup-backups.sh` - Remove old backups based on retention policy - `health-check.sh` - Monitor service health - `restore-postgres.sh` - Restore PostgreSQL from backup - `restore-redis.sh` - Restore Redis from backup

**Backup Location:** - Local: `/backups/postgres` and `/backups/redis` - Cloud (optional): S3 bucket (configurable)

# Services Overview

Service	Port(s)	Purpose	Health Check
frontend	3000	React UI	HTTP GET /
backend	4000	API Server	HTTP GET /health
ai-service	8000	AI Analysis	TCP
postgres	5432	Database	pg_isready
pgbouncer	6432	Connection Pool	pg_isready
redis-master	6379	Cache Master	PING
redis-replica-1	6380	Cache Replica	PING
redis-replica-2	6381	Cache Replica	PING
redis-sentinel-1	26379	Failover Monitor	TCP
redis-sentinel-2	26380	Failover Monitor	TCP
redis-sentinel-3	26381	Failover Monitor	TCP
backup-service	-	Backup Automation	Internal

# Network Configuration

---

**Network:** psscript-network - Type: Bridge - Subnet: 172.25.0.0/16 - DNS:

Automatic service discovery by name

# Volume Management

---

**Persistent Volumes:** - postgres\_data - PostgreSQL data directory -  
redis\_master\_data - Redis master data - redis\_replica\_1\_data - Redis  
replica 1 data - redis\_replica\_2\_data - Redis replica 2 data -  
redis\_sentinel\_1\_data - Sentinel 1 configuration - redis\_sentinel\_2\_data  
- Sentinel 2 configuration - redis\_sentinel\_3\_data - Sentinel 3 configuration

**Bind Mounts:** - ./backups - Backup storage directory - ./docker/pgbouncer -  
PgBouncer configuration - ./docker/redis - Redis configuration -  
./docker/postgres - PostgreSQL configuration

# Getting Started

---

## Initial Setup

### 1. Copy environment file:

```
cp .env.example .env  
# Edit .env with your configuration
```

### 1. Create required directories:

```
mkdir -p backups/postgres backups/redis backups/logs
```

### 1. Start all services:

```
docker-compose up -d
```

### 1. Verify services are running:

```
docker-compose ps
```

## Service Management

### Start all services:

```
docker-compose up -d
```

### Stop all services:

```
docker-compose down
```

### View logs:



```
# All services
docker-compose logs -f

# Specific service
docker-compose logs -f backend
docker-compose logs -f redis-master
docker-compose logs -f pgbouncer
```

#### Restart a service:

```
docker-compose restart backend
```

#### Scale replica count (if needed):

```
docker-compose up -d --scale redis-replica=3
```

### Monitoring

#### Check PgBouncer status:

```
docker-compose exec pgbouncer psql -h localhost -p 6432 -U postgres
docker-compose exec pgbouncer psql -h localhost -p 6432 -U postgres
```

#### Check Redis Sentinel status:

```
docker-compose exec redis-sentinel-1 redis-cli -p 26379 SENTINEL
docker-compose exec redis-sentinel-1 redis-cli -p 26379 SENTINEL
```

#### Check Redis replication:

```
docker-compose exec redis-master redis-cli INFO replication
docker-compose exec redis-replica-1 redis-cli INFO replication
```

#### View backup status:

```
# Check backup logs  
tail -f backups/logs/backup.log
```

```
# View backup inventory  
ls -lh backups/postgres/  
ls -lh backups/redis/
```

```
# Check health status  
tail -f backups/logs/health.log
```

# Backup and Recovery

---

## Manual Backups

### PostgreSQL Full Backup:

```
docker-compose exec backup-service /scripts/postgres-backup.sh full
```

### PostgreSQL Incremental Backup:

```
docker-compose exec backup-service /scripts/postgres-backup.sh incremental
```

### Redis Backup:

```
docker-compose exec backup-service /scripts/redis-backup.sh
```

## Restore Operations

### Restore PostgreSQL:

```
# List available backups
docker-compose exec backup-service ls -lh /backups/postgres/

# Restore from backup
docker-compose exec backup-service /scripts/restore-postgres.sh /backups/postgres/backup-2024-10-27-15-30-00.tar.gz
```

### Restore Redis:

```
# List available backups
docker-compose exec backup-service ls -lh /backups/redis/

# Restore from backup
docker-compose exec backup-service /scripts/restore-redis.sh /backups/redis/backup-2024-10-27-15-30-00.tar.gz
```

## S3 Cloud Backup Configuration

Add to your `.env` file:

```
BACKUP_S3_BUCKET=your-bucket-name  
BACKUP_S3_REGION=us-east-1  
AWS_ACCESS_KEY_ID=your-access-key  
AWS_SECRET_ACCESS_KEY=your-secret-key
```

Backups will automatically upload to S3 after local backup completes.

# Disaster Recovery

---

## Complete System Recovery

### 1. Stop all services:

```
docker-compose down
```

### 1. Restore PostgreSQL:

```
docker-compose up -d postgres pgbouncer backup-service  
docker-compose exec backup-service /scripts/restore-postgres.sh /backups
```

### 1. Restore Redis:

```
docker-compose up -d redis-master  
docker-compose exec backup-service /scripts/restore-redis.sh /backups
```

### 1. Start remaining services:

```
docker-compose up -d
```

## Testing Failover

### Test Redis Failover:

```
# Stop master
docker-compose stop redis-master

# Watch sentinel logs
docker-compose logs -f redis-sentinel-1

# Verify new master
docker-compose exec redis-sentinel-1 redis-cli -p 26379 SENTINEL r

# Restart original master (becomes replica)
docker-compose start redis-master
```

# Performance Tuning

---

## PgBouncer Optimization

Edit `/docker/pgbouncer/pgbouncer.ini`:

```
# Increase pool sizes for high-traffic applications
default_pool_size = 50
max_client_conn = 2000

# Adjust timeouts
query_wait_timeout = 60
```

## PostgreSQL Optimization

Edit `/docker/postgres/postgresql.conf`:

```
# Increase memory for better performance
shared_buffers = 512MB
effective_cache_size = 2GB
work_mem = 32MB
```

## Redis Optimization

Edit `/docker/redis/redis-master.conf`:

```
# Increase memory limit
maxmemory 1gb

# Adjust eviction policy
maxmemory-policy allkeys-lfu
```

# Troubleshooting

---

## PgBouncer Connection Issues

```
# Check PgBouncer logs
docker-compose logs pgbouncer

# Test direct connection
docker-compose exec pgbouncer psql -h postgres -p 5432 -U postgres

# Check pool status
docker-compose exec pgbouncer psql -h localhost -p 6432 -U postgres
```

## Redis Failover Not Working

```
# Check sentinel logs
docker-compose logs redis-sentinel-1

# Verify sentinel configuration
docker-compose exec redis-sentinel-1 cat /usr/local/etc/redis/sentinel.conf

# Check master status
docker-compose exec redis-sentinel-1 redis-cli -p 26379 SENTINEL get-master-addr-by-name
```



## Backup Failures

```
# Check backup service logs
docker-compose logs backup-service
tail -f backups/logs/backup.log

# Verify disk space
df -h

# Test manual backup
docker-compose exec backup-service /scripts/postgres-backup.sh fu
```

# Security Considerations

---

1. **Change default passwords** in production:
2. PostgreSQL password
3. Redis password (uncomment in redis configs)
4. PgBouncer user password
5. **Enable SSL/TLS:**
6. Configure PostgreSQL SSL
7. Configure Redis TLS
8. Use encrypted connections in pgBouncer
9. **Restrict network access:**
10. Use firewall rules
11. Limit exposed ports
12. Use VPN for remote access
13. **Encrypt backups:**
14. Enable S3 encryption
15. Use encrypted volumes
16. Secure backup transfer channels

# Maintenance

---

## Regular Maintenance Tasks

### 1. Monitor disk usage:

```
docker-compose exec backup-service df -h /backups
```

### 1. Check service health:

```
docker-compose ps  
docker-compose exec backup-service /scripts/health-check.sh
```

### 1. Review logs:

```
docker-compose logs --tail=100 postgres  
docker-compose logs --tail=100 redis-master
```

### 1. Update Docker images:

```
docker-compose pull  
docker-compose up -d
```

### 1. Vacuum PostgreSQL:

```
docker-compose exec postgres vacuumdb -U postgres -d psscript --a
```

## Support and Resources

---

- Docker Compose Documentation: <https://docs.docker.com/compose/>
- PgBouncer Documentation: <https://www.pgbouncer.org/>
- Redis Sentinel Documentation: <https://redis.io/topics/sentinel>
- PostgreSQL Documentation: <https://www.postgresql.org/docs/>

Generated 2026-01-13 06:26 UTC