PowerChat Plus - Advanced Multi-Instance Deployment Guide

Deploy multiple isolated PowerChat Plus instances using the pre-compiled production build with automatic port management and conflict resolution.

Features

- **Multi-Instance Support**: Deploy unlimited isolated instances on the same server
- **Automatic Port Detection**: Intelligent port assignment (9000, 9001, 9002...) with conflict resolution
- **Production Ready**: Uses PostgreSQL 16.1 and Node.js 20.19.1 in optimized containers
- **Secure by Default**: Auto-generated secrets and secure configurations
- **Instance Isolation**: Separate networks, volumes, and containers per instance
- **Easy Management**: Comprehensive instance management tools

Prerequisites

- **Docker**: 20.10.0 or higher
- **Docker Compose**: 2.0.0 or higher
- **Node.js**: 18.0.0 or higher (for dependency installation)
- **npm**: Latest version
- **OpenSSL**: For secure secret generation
- **System Requirements**: 2GB RAM per instance, 5GB storage

Quick Start

1. Deploy Your First Instance

```
# Navigate to the share directory
cd share/

# Make scripts executable
chmod +x multi-instance-deploy.sh manage-instances.sh

# Deploy a new instance (interactive)
./multi-instance-deploy.sh
```

2. Follow Interactive Setup

The script will guide you through:

1. **Dependency Installation**: Automatic `npm install --production`
2. **Instance Configuration**: Name, database, company details
3. **Admin Setup**: Email, username, password (auto-generated)
4. **Port Assignment**: Automatic detection of available ports
5. **Deployment**: Docker build and container startup
6. **Database Migration**: Automatic schema setup

3. Access Your Instance

After deployment:

- **Application**: http://localhost:9000 (or assigned port)
- **Admin Panel**: http://localhost:9000/admin
- **Database**: localhost:5432 (or assigned port)

Instance Management

List All Instances

```
# Show all deployed instances
./multi-instance-deploy.sh list

# Or use the management utility
./manage-instances.sh list
```

Manage Individual Instances

```
# Start an instance
./manage-instances.sh start my-company

# Stop an instance
./manage-instances.sh stop my-company

# Restart an instance
./manage-instances.sh restart my-company

# View logs
./manage-instances.sh logs my-company

# Show detailed info
./manage-instances.sh info my-company

# Create backup
./manage-instances.sh backup my-company
```

Bulk Operations

```
# Start all instances
./manage-instances.sh start-all

# Stop all instances
./manage-instances.sh stop-all

# Show status of all instances
./multi-instance-deploy.sh status
```

Directory Structure

```
share/
├── multi-instance-deploy.sh      # Main deployment script
├── manage-instances.sh            # Instance management utility
├── Dockerfile.production          # Production container (auto-created)
├── init-db.sql                   # Database initialization
├── dist/                          # Pre-compiled application
├── migrations/                    # Database migrations
├── scripts/                       # Utility scripts
├── instances/                     # Instance configurations
│   ├── company1/
│   │   ├── .env                   # Instance environment
│   │   ├── docker-compose.yml     # Instance services
│   │   └── manage.sh              # Instance management
│   └── company2/
│       ├── .env
│       ├── docker-compose.yml
│       └── manage.sh
└── backups/                       # Instance backups
    ├── company1/
    └── company2/
```

Security Features

Automatic Secret Generation

Each instance gets unique, secure secrets:

- **SESSION_SECRET**: 32-character random string
- **ENCRYPTION_KEY**: 32-byte hex key
- **Database passwords**: 16-character random passwords

Instance Isolation

- **Network isolation**: Each instance has its own Docker network
- **Volume isolation**: Separate data volumes per instance
- **Container isolation**: Unique container names and configurations
- **Port isolation**: Automatic port conflict detection

Production Security

- **Non-root containers**: Applications run as unprivileged users
- **Health checks**: Automatic container health monitoring
- **Optimized databases**: PostgreSQL 16.1 with performance tuning
- **Secure defaults**: Production-ready security configurations

Port Management

Automatic Port Assignment

- **Application ports**: Starting from 9000 (9000, 9001, 9002...)
- **Database ports**: Starting from 5432 (5432, 5433, 5434...)
- **Conflict detection**: Checks system and Docker port usage
- **Range limits**: Configurable port ranges with safety limits

Port Validation

The script automatically:

1. Scans for available ports in the specified range
2. Checks both system and Docker port usage
3. Suggests next available ports
4. Validates port availability before deployment

Monitoring & Backup

Health Monitoring

Each instance includes:

- **Application health**: HTTP endpoint monitoring
- **Database health**: PostgreSQL connection checks
- **Container health**: Docker health check integration
- **Resource monitoring**: CPU and memory usage tracking

Backup System

```
# Create backup for specific instance
./manage-instances.sh backup my-company

# Backup includes:
# - Database dump (SQL format)
# - Application volumes (uploads, media)
# - Configuration files (.env, docker-compose.yml)
```

Troubleshooting

Common Issues

Dependencies fail to install:

```
# Clear npm cache and retry
npm cache clean --force
./multi-instance-deploy.sh
```

Port conflicts:

```
# Script automatically detects and suggests alternatives
# Check current usage: netstat -tuln | grep :9000
```

Container won't start:

```
# Check logs
./manage-instances.sh logs instance-name

# Check status
./manage-instances.sh status instance-name
```

Database connection issues:

```
# Check database health
docker-compose -f instances/company1/docker-compose.yml \
  exec postgres-company1 pg_isready -U powerchat
```

Performance Optimization

Resource Limits:

Edit `instances/INSTANCE_NAME/docker-compose.yml`:

```
services:
  app-instance:
    deploy:
      resources:
        limits:
          memory: 512M
          cpus: '0.5'
```

Advanced Configuration

Environment Customization

Each instance's `.env` file can be customized:

```
# Edit instance configuration
nano instances/company1/.env

# Restart to apply changes
./manage-instances.sh restart company1
```

Database Tuning

PostgreSQL is pre-optimized with:

- Connection pooling (100 max connections)
- Memory optimization (256MB shared buffers)
- Performance tuning for PowerChat Plus workloads

Support

Command Reference

```
# Deployment
./multi-instance-deploy.sh              # Deploy new instance
./multi-instance-deploy.sh list         # List instances
./multi-instance-deploy.sh help         # Show help

# Management
./manage-instances.sh list              # List with details
./manage-instances.sh info INSTANCE     # Show instance info
./manage-instances.sh backup INSTANCE   # Create backup
```

Getting Help

1. **Check logs** for error messages
2. **Verify configuration** in instance `.env` files
3. **Test connectivity** between services
4. **Review documentation** for specific issues

Whatsapp 923059002132