Pulse Deployment Guide

This guide covers deploying Pulse in various environments: local development, Docker, and Kubernetes.

Table of Contents

Local Development

Prerequisites

- Node.js 22.x
- pnpm 9.x
- MySQL 8.0
- Git

Setup Steps

1. **Clone repository** `bash git clone https://github.com/yourusername/pulse.git cd pulse`

2. **Install dependencies** `bash pnpm install`

3. **Configure environment** `bash cp .env.example .env.local # Edit .env.local with your settings`

4. **Initialize database** `bash pnpm db:push`

5. **Start development server** `bash pnpm dev`

6. **Access application**

7. Frontend: http://localhost:3000
8. API: http://localhost:3000/api/trpc

Development Commands

```
# Build for production
pnpm build

# Start production server
pnpm start

# Run linter
pnpm lint

# Type checking
pnpm type-check

# Database migrations
pnpm db:push
pnpm db:generate
```

Docker Deployment

Single Container Deployment

1. **Build Docker image** `bash docker build -f infrastructure/docker/Dockerfile -t pulse:latest .`

2. **Run container** `bash docker run -d \ --name pulse \ -p 3000:3000 \ -e DATABASE_URL=mysql://user:pass@db:3306/pulse \ -e NODE_ENV=production \ -e JWT_SECRET=your-secret \ -e VITE_APP_ID=your-app-id \ pulse:latest`

3. **Check logs** `bash docker logs -f pulse`

Docker Compose Deployment

1. **Navigate to Docker directory** `bash cd infrastructure/docker`

2. **Create environment file** `bash cat > .env << EOF NODE_ENV=production DB_ROOT_PASSWORD=rootpassword DB_NAME=pulse DB_USER=pulse DB_PASSWORD=pulsepassword GRAFANA_PASSWORD=admin EOF`

3. **Start services** `bash docker-compose up -d`

4. **Verify services** `bash docker-compose ps`

5. **Access services**

6. Application: http://localhost:3000
7. Prometheus: http://localhost:9090

8. Grafana: http://localhost:3001

9. **Stop services** `bash docker-compose down`

Docker Compose with Volumes

For persistent data:

```yaml
volumes:
  db_data:
    driver: local
  prometheus_data:
    driver: local
  grafana_data:
    driver: local
```

Kubernetes Deployment

Prerequisites

- Kubernetes cluster 1.24+
- kubectl configured
- Docker image in registry

- Helm (optional)

Step-by-Step Deployment

1. **Create namespace** `bash kubectl create namespace pulse`

2. **Create secrets** `bash kubectl create secret generic pulse-secrets \ --from-literal=database-url=mysql://user:pass@db:3306/pulse \ --from-literal=jwt-secret=your-secret-key \ --from-literal=vite-app-id=your-app-id \ --from-literal=oauth-server-url=https://oauth.example.com \ --from-literal=vite-oauth-portal-url=https://portal.example.com \ --from-literal=owner-open-id=your-owner-id \ --from-literal=forge-api-url=https://api.example.com \ --from-literal=forge-api-key=your-api-key \ -n pulse`

3. **Create ConfigMap** `bash kubectl create configmap pulse-config \ --from-literal=owner-name="Your Name" \ --from-literal=app-title="Pulse" \ --from-literal=app-logo="https://example.com/logo.png" \ -n pulse`

4. **Create service account** `bash kubectl create serviceaccount pulse-app -n pulse`

5. **Apply deployment** `bash kubectl apply -f infrastructure/kubernetes/deployment.yaml -n pulse`

6. **Verify deployment** `bash kubectl get pods -n pulse kubectl get svc -n pulse kubectl get hpa -n pulse`

7. **Check pod logs** `bash kubectl logs -f deployment/pulse-app -n pulse`

Accessing the Application

```
 # Port forward to local machine
kubectl port-forward svc/pulse-app 3000:80 -n pulse

# Access at http://localhost:3000
```

Scaling

```
 # Manual scaling
kubectl scale deployment pulse-app --replicas=5 -n pulse

# Check HPA status
kubectl get hpa -n pulse

# Watch HPA scaling
kubectl get hpa -n pulse --watch
```

Updating Deployment

```
 # Update image
kubectl set image deployment/pulse-app \
  pulse=myregistry/pulse:v1.1.0 \
  -n pulse

# Rollout status
kubectl rollout status deployment/pulse-app -n pulse

# Rollback if needed
kubectl rollout undo deployment/pulse-app -n pulse
```

Cloud Providers

AWS ECS

1. **Create ECR repository** `bash aws ecr create-repository --repository-name pulse`

2. **Push image** `bash docker tag pulse:latest <account>.dkr.ecr.<region>.amazonaws.com/pulse:latest docker push <account>.dkr.ecr.<region>.amazonaws.com/pulse:latest`

3. **Create ECS task definition** `json { "family": "pulse", "containerDefinitions": [ { "name": "pulse", "image": "<account>.dkr.ecr.<region>.amazonaws.com/pulse:latest", "portMappings": [ { "containerPort": 3000, "hostPort": 3000, "protocol": "tcp" } ], "environment": [ { "name": "NODE_ENV", "value": "production" } ] } ] }`

4. **Create ECS service** `bash aws ecs create-service \ --cluster pulse-cluster \ --service-name pulse-service \ --task-definition pulse \ --desired-count 3`

Google Cloud Run

1. **Build and push image** `bash gcloud builds submit --tag gcr.io/PROJECT_ID/pulse`

2. **Deploy to Cloud Run** `bash gcloud run deploy pulse \ --image gcr.io/PROJECT_ID/pulse \ --platform managed \ --region us-central1 \ --memory 512Mi \ --cpu 1 \ --set-env-vars DATABASE_URL=mysql://... \ --allow-unauthenticated`

Azure Container Instances

1. **Push to Azure Container Registry** `bash az acr build --registry myregistry --image pulse:latest .`

2. **Deploy container** `bash az container create \ --resource-group mygroup \ --name pulse \ --image myregistry.azurecr.io/pulse:latest \ --cpu 1 \ --memory 1 \ --ports 3000 \ --environment-variables \ NODE_ENV=production \ DATABASE_URL=mysql://...`

Production Checklist

Pre-Deployment

- [ ] All tests passing
- [ ] Code review completed
- [ ] Security scan passed
- [ ] Database backup created
- [ ] Secrets configured securely
- [ ] SSL/TLS certificates ready
- [ ] Monitoring configured

- [ ] Logging configured
- [ ] Backup plan documented
- [ ] Rollback plan documented

Deployment

- [ ] Environment variables set correctly
- [ ] Database migrations applied
- [ ] Application health checks passing
- [ ] Load balancer configured
- [ ] DNS records updated
- [ ] SSL/TLS enabled
- [ ] Monitoring alerts active
- [ ] Logging aggregation working

Post-Deployment

- [ ] Application responding to requests
- [ ] Database connectivity verified
- [ ] Monitoring metrics flowing
- [ ] Alerts tested
- [ ] Performance baseline established
- [ ] User access verified
- [ ] Backup verification completed
- [ ] Documentation updated

Monitoring & Maintenance

Health Checks

```bash
# Check application health
curl http://localhost:3000/health

# Check database connectivity
kubectl exec -it pod/pulse-app -n pulse -- \
  mysql -h db -u pulse -p -e "SELECT 1"

# Check Prometheus metrics
curl http://localhost:9090/api/v1/targets
```

Logs

```bash
# Docker logs
docker logs -f pulse

# Kubernetes logs
kubectl logs -f deployment/pulse-app -n pulse

# Follow logs with timestamps
kubectl logs -f deployment/pulse-app -n pulse --timestamps=true
```

Metrics

Access Grafana at http://localhost:3001 (default: admin/admin)

Key dashboards: - System Overview - Application Metrics - Monitoring Health - Database Performance

Backup & Recovery

```bash
# Backup database
mysqldump -u pulse -p pulse > backup.sql

# Restore database
mysql -u pulse -p pulse < backup.sql

# Backup Kubernetes secrets
kubectl get secrets -n pulse -o yaml > secrets-backup.yaml

# Restore Kubernetes secrets
kubectl apply -f secrets-backup.yaml
```

Updates

1. **Test in staging** `bash git checkout develop pnpm install pnpm build docker build -t pulse:staging .`

2. **Deploy to staging** `bash docker-compose -f docker-compose.staging.yml up -d`

3. **Verify in staging**

4. Run smoke tests
5. Check functionality

6. Monitor performance

7. **Deploy to production** `bash git checkout main git merge develop docker build -t pulse:latest . docker push myregistry/pulse:latest kubectl set image deployment/pulse-app pulse=myregistry/pulse:latest -n pulse`

Troubleshooting

Pod CrashLoopBackOff

```
# Check pod logs
kubectl logs <pod-name> -n pulse

# Describe pod
kubectl describe pod <pod-name> -n pulse

# Check resource limits
kubectl top pod -n pulse
```

Database Connection Issues

```
# Test connectivity
kubectl exec -it pod/pulse-app -n pulse -- \
  nc -zv db 3306

# Check environment variables
kubectl exec -it pod/pulse-app -n pulse -- \
  env | grep DATABASE
```

High Memory Usage

```bash
# Check memory usage
kubectl top pod -n pulse

# Increase memory limit
kubectl set resources deployment pulse-app \
  --limits=memory=2Gi -n pulse
```

Version: 1.0.0
Last Updated: 2024