

春景智能 QMES SaaS 平台 — 部署与运维手册

文档版本: V1.0

编写日期: 2025-07-17

部署方式: Docker Compose

服务器要求: Linux (Ubuntu 22.04 LTS+) / macOS / Windows WSL2

目录

- [1. 第一性原理设计](#)
- [2. 环境要求](#)
- [3. 快速部署指南](#)
- [4. 详细部署步骤](#)
- [5. Docker Compose 服务说明](#)
- [6. 配置说明](#)
- [7. 数据库运维](#)
- [8. Redis运维](#)
- [9. 监控与告警](#)
- [10. 备份与恢复](#)
- [11. 版本升级与回滚](#)
- [12. 常见故障处理](#)
- [13. 运维检查清单](#)

1. 第一性原理设计

部署运维的核心命题

"运维操作能帮平台运营团队做什么决策？"

运维环节	决策价值	设计原则
一键部署	"新租户入驻能在5分钟内完成吗？"	Docker Compose 标准化部署
日志监控	"系统现在健康吗？有没有异常请求？"	集中日志 + 实时告警

运维环节	决策价值	设计原则
备份恢复	"数据丢了能多久恢复？"	自动每日备份 + 15分钟RPO
水平扩展	"用户量翻倍了，系统扛得住吗？"	无状态应用水平扩展
版本升级	"新功能上线能零停机吗？"	蓝绿部署/滚动更新

运维SLA目标

指标	目标
系统可用性	99.9%（年停机≤8.7小时）
故障恢复时间(MTTR)	≤30分钟
数据恢复点(RPO)	≤15分钟
新租户开通时间	≤5分钟

--

2. 环境要求

2.1 服务器最低配置

环境	CPU	内存	磁盘	带宽
开发	2核	4GB	20GB SSD	—
测试	4核	8GB	50GB SSD	100Mbps
生产(起步)	4核	16GB	200GB SSD	200Mbps
生产(扩展)	8核	32GB	500GB SSD	500Mbps

2.2 软件依赖

软件	版本	安装方式
Docker	24+	docker.io
Docker Compose	2.20+	docker-compose-plugin
PHP	8.2+	容器内
Node.js	18+	用于前端构建
Git	2.x	—

2.3 端口规划

端口	服务	说明
80	Nginx	HTTP
443	Nginx	HTTPS
5432	PostgreSQL	可对外(需白名单)
6379	Redis	不对外
8080	应用	开发环境HTTP

3. 快速部署指南

3.1 一键部署（新服务器）

```
# 1. 安装 Docker 和 Docker Compose
curl -fsSL https://get.docker.com | bash
sudo usermod -aG docker $USER
# 重新登录使组生效

# 2. 克隆代码
git clone https://gitea.qmes.cn/qmes/qmes-saas.git /opt/qmes-saas
cd /opt/qmes-saas

# 3. 配置环境变量
cp .env.example .env
# 编辑 .env 配置数据库密码、JWT密钥等

# 4. 启动所有服务
docker compose up -d

# 5. 安装PHP依赖
docker compose exec php-fpm composer install --no-dev

# 6. 运行数据库迁移
docker compose exec php-fpm php artisan migrate --force

# 7. 初始化种子数据
docker compose exec php-fpm php artisan db:seed --force

# 8. 生成JWT密钥
docker compose exec php-fpm php artisan jwt:secret

# 9. 构建前端
cd frontend && npm install && npm run build && cd ..

# 10. 重启Nginx加载前端
```

```
docker compose restart nginx
```

```
# ✅ 部署完成! 访问 http://localhost:8080
```

3.2 快速检查

```
# 检查所有容器运行状态
```

```
docker compose ps
```

```
# 检查应用日志
```

```
docker compose logs php-fpm --tail=50
```

```
# 检查API是否正常
```

```
curl http://localhost:8080/api/plans
```

```
# 健康检查
```

```
docker compose exec php-fpm php artisan health:check
```

4. 详细部署步骤

4.1 Docker 环境准备

```
# Ubuntu 22.04 安装 Docker
```

```
sudo apt-get update
```

```
sudo apt-get install -y ca-certificates curl gnupg
```

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyr
```

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://
```

```
sudo apt-get update
```

```
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-com
```

4.2 环境配置文件

.env 关键配置项:

```
# 应用配置
```

```
APP_NAME=QMES-SaaS
```

```
APP_ENV=production
```

```
APP_DEBUG=false
```

```
APP_URL=https://qmes.cn
```

```
# 数据库配置
```

```
DB_CONNECTION=pgsql
```

```
DB_HOST=postgres
```

```
DB_PORT=5432
```

```
DB_DATABASE=qmes_saas
```

```
DB_USERNAME=qmes
DB_PASSWORD=<强密码：推荐32位随机>

# Redis配置
REDIS_HOST=redis
REDIS_PORT=6379
REDIS_PASSWORD=<可选>

# JWT配置
JWT_SECRET=<由 artisan jwt:secret 生成>
JWT_TTL=3600
JWT_REFRESH_TTL=20160

# Session/缓存
CACHE_DRIVER=redis
SESSION_DRIVER=redis
QUEUE_CONNECTION=redis

# 邮件
MAIL_MAILER=smtp
MAIL_HOST=smtp.qmes.cn
MAIL_PORT=587
MAIL_USERNAME=noreply@qmes.cn
MAIL_PASSWORD=<邮箱密码>

# 对象存储（S3兼容）
FILESYSTEM_DISK=s3
AWS_ACCESS_KEY_ID=<access_key>
AWS_SECRET_ACCESS_KEY=<secret_key>
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=qmes-files
AWS_ENDPOINT=https://s3.qmes.cn

# AI API配置（默认，可在租户设置中覆盖）
AI_VISION_API_URL=https://ai-vision.qmes.cn/api
AI_VISION_API_KEY=<ai_api_key>
```

4.3 容器构建

```
# PHP-FPM Dockerfile (docker/php.dockerfile)
FROM php:8.2-fpm-alpine

RUN apk add --no-cache postgresql-dev \
    && docker-php-ext-configure pgsql --with-pgsql \
    && docker-php-ext-install pdo_pgsql bcmath pcntl

COPY --from=composer:latest /usr/bin/composer /usr/bin/composer

WORKDIR /var/www/html
```

4.4 Nginx配置

```
# docker/nginx.conf
server {
    listen 80;
    server_name localhost;

    # API后端
    location /api {
        proxy_pass http://php-fpm:9000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # 大JSON请求体
        client_max_body_size 50M;
    }

    # 前端静态文件
    root /var/www/html/frontend/dist;
    index index.html;

    location / {
        try_files $uri $uri/ /index.html;
    }
}
```

4.5 Horizon队列配置

```
// config/horizon.php
'environments' => [
    'production' => [
        'supervisor-1' => [
            'connection' => 'redis',
            'queue' => ['high', 'default', 'low'],
            'balance' => 'auto',
            'minProcesses' => 3,
            'maxProcesses' => 10,
            'tries' => 3,
        ],
        'supervisor-ai' => [
            'connection' => 'redis',
            'queue' => ['ai-vision'],
            'balance' => 'auto',
            'minProcesses' => 1,
            'maxProcesses' => 5,
            'tries' => 1,
            'timeout' => 120,
        ],
    ],
]
```

```
1,
1,
```

5. Docker Compose 服务说明

5.1 服务概览

服务名	镜像	依赖	健康检查	用途
nginx	nginx:1.25-alpine	php-fpm	端口80	反向代理
php-fpm	自定义构建	postgres, redis	-	Laravel应用
postgres	postgres:16-alpine	-	pg_isready	数据库
redis	redis:7.2-alpine	-	redis-cli ping	缓存/队列
horizon	自定义构建	postgres, redis	-	队列处理

5.2 容器资源限制

```
services:
  php-fpm:
    deploy:
      resources:
        limits:
          cpus: '2'
          memory: 2G
        reservations:
          cpus: '1'
          memory: 1G

  postgres:
    deploy:
      resources:
        limits:
          cpus: '4'
          memory: 4G
```

5.3 数据持久化 (Volumes)

```
volumes:
  postgres_data:
    driver: local
    driver_opts:
      type: none
      device: /data/qmes/postgres # 挂载到宿主机
      o: bind
```

```
redis_data:
  driver: local
  driver_opts:
    type: none
    device: /data/qmes/redis
    o: bind
```

6. 配置说明

6.1 后端配置 (config/)

配置文件	关键配置	说明
app.php	name, url, timezone, locale	应用基本配置
database.php	pgsql连接/redis连接	数据库连接池
cache.php	redis驱动, prefix	缓存配置
jwt.php	secret, ttl, refresh_ttl	JWT认证
sanctum.php	stateful_domains	SPA认证
horizon.php	queue配置	队列worker
filesystems.php	s3/local disks	文件存储
mail.php	smtp配置	邮件服务
services.php	ai_vision, wecom	第三方服务
tenant.php	cookie_domain, plans	租户特有配置

6.2 租户配置

每个租户可独立配置：

配置项	来源	说明
AI API URL	tenants.ai_api_url	视觉检测AI地址
AI API Key	tenants.ai_api_key	视觉检测密钥
套餐限制	tenants.plan, max_users, max_workshops	功能/资源限制
子域名	tenants.domain	独立访问地址

7. 数据库运维

7.1 日常维护

```
# 数据库全量备份
docker compose exec postgres pg_dump -U qmes qmes_saas > /backup/qmes_$(date +%Y%m%d_%H%M%S).

# 数据库恢复
docker compose exec -T postgres psql -U qmes qmes_saas < /backup/qmes_20250717_120000.sql

# 重建索引（低峰期执行）
docker compose exec postgres psql -U qmes -d qmes_saas -c "REINDEX DATABASE qmes_saas;"

# 清理死元组
docker compose exec postgres psql -U qmes -d qmes_saas -c "VACUUM ANALYZE;"
```

7.2 迁移管理

```
# 执行迁移
php artisan migrate

# 回滚迁移
php artisan migrate:rollback
php artisan migrate:rollback --step=3

# 查看迁移状态
php artisan migrate:status

# 生成迁移
php artisan make:migration add_xxx_to_xxx_table
```

7.3 大表分区维护

对于 device_data 表（10亿+行）：

```
-- 创建新分区（每月1日执行）
CREATE TABLE device_data_202508 PARTITION OF device_data
    FOR VALUES FROM ('2025-08-01') TO ('2025-09-01');

-- 查询分区大小
SELECT
    schemaname, tablename, pg_size_pretty(pg_total_relation_size(schemaname||'.'||tablename))
FROM pg_tables
WHERE tablename LIKE 'device_data_%'
ORDER BY tablename;
```

8. Redis运维

8.1 缓存管理

```
# 清除租户缓存
docker compose exec redis redis-cli KEYS "qmes_cache_:tenant_{id}:*" | xargs redis-cli DEL

# 清除全部缓存
php artisan cache:clear

# 查看key数量
docker compose exec redis redis-cli DBSIZE

# 查看内存使用
docker compose exec redis redis-cli INFO memory
```

8.2 队列监控

```
# 查看Horizon状态
php artisan horizon:status

# 暂停队列(维护时)
php artisan horizon:pause

# 恢复队列
php artisan horizon:continue

# 查看队列长度
docker compose exec redis redis-cli LLEN queues:default
docker compose exec redis redis-cli LLEN queues:high
```

9. 监控与告警

9.1 内置监控

监控项	接口/命令	告警阈值
API可用性	GET /api/plans	连续3次失败
数据库连接	DB::connection()->getPdo()	连接失败
队列堆积	horizon:status	>1000 pending
磁盘使用	df -h	>85%
内存使用	free -m	>90%

9.2 日志管理

```
# 应用日志
docker compose logs php-fpm --tail=100

# Nginx访问日志
docker compose logs nginx --tail=100

# 操作日志（数据库）
SELECT * FROM oper_logs
WHERE created_at >= NOW() - INTERVAL '1 hour'
ORDER BY id DESC;

# 日志轮转
# 在 docker-compose.yml 中配置日志驱动
logging:
  driver: "json-file"
  options:
    max-size: "10m"
    max-file: "3"
```

9.3 系统健康检查

```
// OperationController@systemHealth
public function systemHealth(): JsonResponse
{
    try {
        DB::connection()->getPdo();
        $dbStatus = '正常';
    } catch (\Exception $e) {
        $dbStatus = '异常';
    }

    return success([
        'db_status' => $dbStatus,
        'avg_response_time' => $avgResponseTime,
        'online_rate' => $onlineRate,
        'total_tenants' => $totalTenants,
        'active_tenants' => $activeTenants,
    ]);
}
```

9.4 告警推送（运维团队）

告警事件	推送方式	接收人
服务不可用	企微机器人	运维团队
数据库死锁	企微+邮件	DBA

告警事件	推送方式	接收人
磁盘不足	企微	运维
队列堆积	企微	开发
异常错误激增	企微	开发

10. 备份与恢复

10.1 自动备份策略

```
#!/bin/bash
# /opt/qmes/scripts/backup.sh

# 每日凌晨3点执行
BACKUP_DIR="/backup/qmes"
DATE=$(date +%Y%m%d_%H%M%S)

# PostgreSQL全量备份
docker compose exec -T postgres \
    pg_dump -U qmes -Fc qmes_saas > ${BACKUP_DIR}/db_${DATE}.dump

# 保留最近30天备份
find ${BACKUP_DIR} -name "db_*.dump" -mtime +30 -delete

# 备份上传至S3
aws s3 cp ${BACKUP_DIR}/db_${DATE}.dump s3://qmes-backup/db/
```

Crontab配置:

```
0 3 * * * /opt/qmes/scripts/backup.sh > /var/log/qmes-backup.log 2>&1
```

10.2 恢复流程

```
# 1. 停止应用
docker compose stop php-fpm horizon

# 2. 重建数据库
docker compose exec postgres psql -U qmes -c "DROP DATABASE IF EXISTS qmes_saas;"
docker compose exec postgres psql -U qmes -c "CREATE DATABASE qmes_saas OWNER qmes;"

# 3. 恢复备份
docker compose exec -T postgres pg_restore -U qmes -d qmes_saas /backup/db_20250717_030000.du

# 4. 恢复应用
docker compose start php-fpm horizon
```

```
# 5. 验证
curl http://localhost:8080/api/plans
```

10.3 恢复演练计划

频率	内容	预期RTO
每月	从备份恢复测试数据库	≤30分钟
每季度	完整灾难恢复演练	≤2小时

11. 版本升级与回滚

11.1 升级流程

```
# 1. 备份（必须）
bash /opt/qmes/scripts/backup.sh

# 2. 拉取新代码
cd /opt/qmes-saas
git pull origin main

# 3. 更新PHP依赖
docker compose exec php-fpm composer install --no-dev --optimize-autoloader

# 4. 执行迁移
docker compose exec php-fpm php artisan migrate --force

# 5. 清除缓存
docker compose exec php-fpm php artisan optimize:clear

# 6. 重新构建前端
cd frontend
npm install && npm run build
cd ..

# 7. 重启服务
docker compose restart php-fpm horizon

# 8. 验证
curl http://localhost:8080/api/plans
```

11.2 回滚流程

```
# 1. 回滚代码
cd /opt/qmes-saas
git reset --hard <上一版本commit>
```

```
# 2. 回滚迁移
docker compose exec php-fpm php artisan migrate:rollback --force

# 3. 恢复备份 (如需)
# 执行恢复流程

# 4. 重启
docker compose restart php-fpm horizon nginx
```

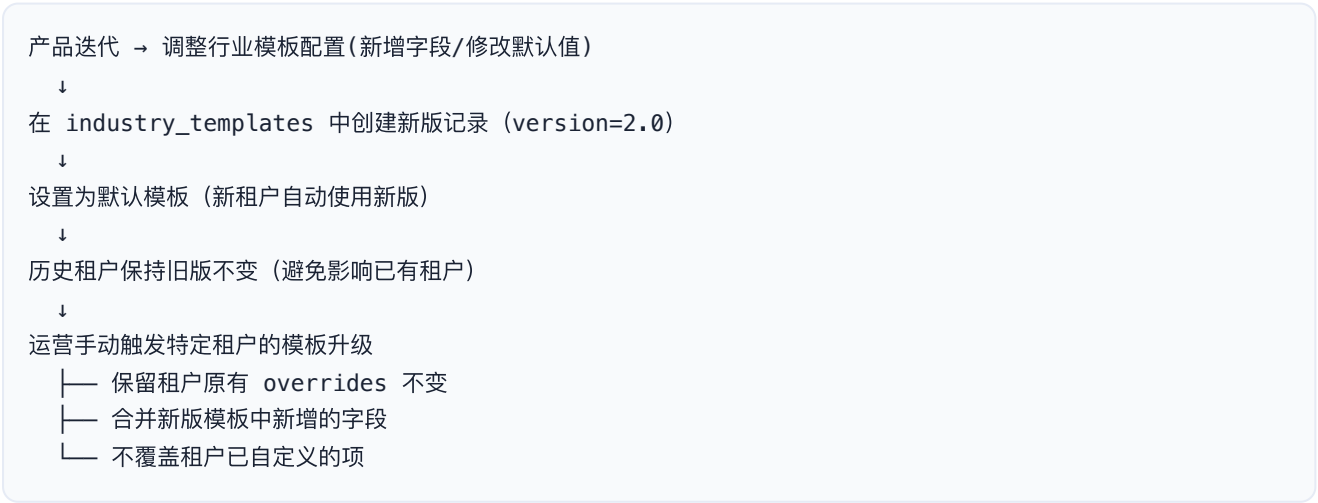
11.3 版本发布记录

版本	日期	变更内容	数据库迁移
v1.0.0	2025-07-01	初始发布: QMES/QMS/设备数采/ESOP/安灯	0001-0028
v1.1.0	2025-08-15	AI视觉+联动版	0029-0032
v1.2.0	2025-10-01	老板推送+自定义字段+品质预警	0033-0036
v2.0.0	2026-01-15	自动排程+PDA+触摸屏	0037-0045
v2.1.0	2026-03-01	条码标签管理+消息通知预警中心+追溯链全景图	0046-0050
v2.2.0	2026-05-01	报表BI看板引擎+模具刀具管理+文档管理DMS	0051-0055
v3.0.0	2026-06-30	开放API+小程序+ERP对接	0056+

11.4 行业模板升级策略

行业模板配置系统引入了JSON配置驱动模板版本管理，模板升级策略与常规版本升级不同，需要特别注意配置合并的兼容性。

11.4.1 模板升级流程



11.4.2 升级脚本示例

```
// artisan industry:upgrade-template --industry=injection_molding --from=1.0 --to=2.0
public function handle()
{
    $industry = Industry::where('code', $this->option('industry'))->firstOrFail();
    $newTemplate = IndustryTemplate::where('industry_id', $industry->id)
        ->where('version', $this->option('to'))
        ->where('is_default', true)
        ->firstOrFail();
    $oldTemplate = IndustryTemplate::where('industry_id', $industry->id)
        ->where('version', $this->option('from'))
        ->firstOrFail();

    // 查询所有使用旧模板的租户
    $settings = TenantIndustrySetting::where('template_id', $oldTemplate->id)->get();

    $bar = $this->output->createProgressBar($settings->count());
    foreach ($settings as $setting) {
        // 1. 计算旧模板与租户overrides的实际差异
        $oldConfig = $oldTemplate->config;
        $userOverrides = $setting->overrides;

        // 2. 切换到新模板
        $setting->template_id = $newTemplate->id;

        // 3. 将旧模板中租户未覆盖的字段标记为"继承模板"
        //     新模板中新增的字段自动生效
        //     租户原来覆盖的值保留不变

        $setting->save();
        $bar->advance();
    }

    $this->info("已升级 {$settings->count()} 个租户");
}
```

11.4.3 管理后台操作

操作	说明	影响范围
创建新版模板	在原有基础上修改JSON配置	无（仅新增记录）
设为默认	新租户使用新版模板	不自动影响历史租户
批量升级租户	选择指定行业/版本，批量迁移	选中的历史租户
单个租户升级	在租户设置页面手动触发	单个租户
预览升级效果	升级前查看合并后的配置对比	预览只读，不执行

11.4.4 升级注意事项

注意事项	说明
兼容性保证	JSON新增字段不影响旧系统，向后兼容
覆盖保留	租户已有的overrides在升级后必须保留
先预览后执行	批量升级前应先预览效果，确认无冲突
回滚方案	upgrade脚本支持 --dry-run 参数预览不执行
通知租户	模板升级后建议通过消息中心通知租户管理员
灰度发布	建议先对少量测试租户执行升级，确认无误后再全量

11.4.5 升级后的验证

```
# 1. 检查升级后的租户配置
docker compose exec php-fpm php artisan industry:check-config --tenant=123

# 2. 对比新旧配置差异
docker compose exec php-fpm php artisan industry:diff-template --industry=injection_molding -

# 3. 验证租户设置完整性
docker compose exec php-fpm php artisan industry:validate-settings

# 4. 缓存刷新（使新配置立即生效）
docker compose exec php-fpm php artisan cache:clear --tags=industry_config
```

12. 55张迁移表部署说明

12.1 数据库表总览

QMES SaaS平台共包含55张业务表，分布在14个产品模块中：

模块	表名	说明	迁移编号
1. 平台基础	tenants	租户表	0001
	users	用户表	0002
	roles	角色表	0003
	role_has_permissions	角色权限关联	0004
	permissions	权限表	0005
	menus	菜单表	0006

模块	表名	说明	迁移编号
	depts	部门表	0007
	plans	套餐表	0008
	tenant_plans	租户套餐记录	0009
	oper_logs	操作日志	0010
	custom_fields	自定义字段定义	0011
	custom_field_values	自定义字段值	0012
2. QMES生产	products	产品表	0013
	product_routings	工艺路线表	0014
	work_orders	工单表	0015
	work_reports	报工记录表	0016
	production_schedules	生产排程表	0017
	defect_codes	缺陷码表	0018
3. WMS仓储	inventories	库存表	0019
	inventory_records	出入库记录表	0020
	warehouses	仓库表	0021
	locations	库位表	0022
4. QMS质量	inspection_records	检验记录表	0023
	inspection_templates	检验模板表	0024
	inspection_items	检验项目表	0025
	ncr_records	NCR不合格品记录	0026
5. 设备数采	devices	设备表	0027
	device_points	设备点位表	0028
	device_data	设备遥测数据表(分区)	0029
	device_alerts	设备告警表	0030
	gateways	网关表	0031
6. ESOP	sop_documents	SOP文档表	0032
	sop_versions	SOP版本历史	0033

模块	表名	说明	迁移编号
7. 安灯	andon_calls	安灯呼叫表	0034
8. AI视觉	ai_vision_tasks	AI检测任务表	0035
	ai_models	AI模型表	0036
9. 条码标签	barcode_rules	条码规则表	0037
	barcode_records	条码记录表	0038
	label_templates	标签模板表	0039
	barcode_scan_logs	扫码日志表	0040
10. 消息通知	message_channels	消息通道表	0041
	message_records	消息记录表	0042
	notification_rules	通知规则表	0043
	message_templates	消息模板表	0044
	alert_configs	预警配置表	0045
11. BI看板	bi_data_sources	BI数据源表	0046
	bi_charts	BI图表配置表	0047
	bi_dashboards	BI看板表	0048
	bi_dashboard_charts	看板-图表关联表	0049
	bi_dashboard_shares	看板分享表	0050
12. 追溯链	trace_chain_records	追溯链记录表	0051
13. 模具刀具	mold_tools	模具刀具档案表	0052
	mold_tool_lends	模具刀具领用表	0053
	mold_tool_maintains	模具刀具保养表	0054
14. 文档DMS	documents	文档表	0055
	document_categories	文档目录表	0056
	document_versions	文档版本表	0057
	document_approvals	文档审批表	0058
	document_permissions	文档权限表	0059

12.2 迁移执行策略

```
# 全量迁移（生产环境）
php artisan migrate --force

# 增量迁移（已有环境升级）
php artisan migrate --force --step

# 按模块迁移（选择性部署）
php artisan migrate --path=/database/migrations/2025_01_01_000001_create_tenants_table.php
php artisan migrate --path=/database/migrations/2025_*_barcode*

# 迁移前备份（必须）
pg_dump -U qmes qmes_saas > /backup/pre_migration_$(date +%Y%m%d).sql
```

12.3 大表说明

表名	预计行数	分区策略	索引策略
device_data	10亿+	按月分区(device_data_YYYYMM)	(device_id, point_id, ts)复合索引
work_reports	1亿+	按年分区	(tenant_id, work_order_id)
message_records	5000万+	按月分区	(tenant_id, event_type, created_at)
barcode_scan_logs	5000万+	按月分区	(tenant_id, barcode)
trace_chain_records	1000万+	按年分区	(tenant_id, trace_id)

13. BI缓存配置

13.1 缓存策略

BI看板数据聚合查询使用Redis缓存，降低数据库压力：

缓存类型	Key模式	TTL	说明
图表数据	bi:chart:{chartId}:{period}:{date}	300s(5分钟)	图表聚合数据
数据源元数据	bi:datasource:{id}	3600s(1小时)	数据源字段定义
图表配置	bi:chart:{id}:config	3600s(1小时)	图表配置JSON
看板数据	bi:dashboard:{id}:data	300s(5分钟)	看板全部图表数据

13.2 缓存配置

```
// config/cache.php - BI专用缓存
'bi' => [
    'driver' => 'redis',
    'connection' => 'cache',
    'prefix' => 'qmes_bi_',
],

// 图表数据缓存实现
public function getCachedChartData(int $chartId, array $params): array
{
    $cacheKey = 'bi:chart:' . $chartId . ':' . ($params['period'] ?? 'day')
        . ':' . ($params['end_date'] ?? date('Y-m-d'));

    return Cache::store('bi')->remember($cacheKey, 300, function () use ($chartId, $params) {
        return $this->queryChartData($chartId, $params);
    });
}

// 缓存清除
public function refreshChartCache(int $chartId): void
{
    Cache::store('bi')->tags(['bi:chart:' . $chartId])>flush();
}
```

13.3 缓存预热

```
# 定时任务：每小时预热热门看板
php artisan bi:cache-warmup --dashboard=1,2,3

# 手动刷新
php artisan bi:cache-refresh --all
```

14. 消息队列配置

14.1 队列拓扑

QMES消息队列结构（基于Redis）

└─ high	（高优先级：设备告警、安灯呼叫、推送）	3 workers
└─ default	（默认：Webhook推送、消息通知）	5 workers
└─ low	（低优先级：消息日志、统计）	2 workers
└─ bi	（BI数据聚合、缓存刷新）	2 workers
└─ documents	（文档处理：转PDF、水印、图片压缩）	2 workers

14.2 Horizon配置

```
// config/horizon.php - 完整队列配置
'environments' => [
    'production' => [
        'supervisor-1' => [
            'connection' => 'redis',
            'queue' => ['high'],
            'balance' => 'auto',
            'minProcesses' => 2,
            'maxProcesses' => 5,
            'tries' => 3,
            'timeout' => 60,
        ],
        'supervisor-2' => [
            'connection' => 'redis',
            'queue' => ['default'],
            'balance' => 'auto',
            'minProcesses' => 3,
            'maxProcesses' => 8,
            'tries' => 3,
            'timeout' => 60,
        ],
        'supervisor-3' => [
            'connection' => 'redis',
            'queue' => ['low'],
            'balance' => 'auto',
            'minProcesses' => 1,
            'maxProcesses' => 3,
            'tries' => 1,
            'timeout' => 120,
        ],
        'supervisor-bi' => [
            'connection' => 'redis',
            'queue' => ['bi'],
            'balance' => 'auto',
            'minProcesses' => 1,
            'maxProcesses' => 3,
            'tries' => 1,
            'timeout' => 300,
        ],
        'supervisor-documents' => [
            'connection' => 'redis',
            'queue' => ['documents'],
            'balance' => 'auto',
            'minProcesses' => 1,
            'maxProcesses' => 3,
            'tries' => 2,
            'timeout' => 120,
        ],
    ],
],
```

14.3 队列监控

```
# 查看队列长度
redis-cli LLEN queues:high
redis-cli LLEN queues:default
redis-cli LLEN queues:bi
redis-cli LLEN queues:documents

# Horizon状态
php artisan horizon:status
php artisan horizon:metrics

# 失败任务
php artisan queue:failed
```

15. 14个产品Docker编排说明

15.1 Docker Compose 完整配置

```
version: '3.8'

services:
  # — 基础服务 —
  nginx:
    image: nginx:1.25-alpine
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./docker/nginx.conf:/etc/nginx/conf.d/default.conf
      - ./frontend/dist:/var/www/html/frontend/dist
      - ./storage:/var/www/html/storage
    depends_on:
      - php-fpm
    networks:
      - qmes-network
    healthcheck:
      test: ["CMD", "nginx", "-t"]
      interval: 30s
      timeout: 5s
      retries: 3

  php-fpm:
    build:
      context: ./docker
      dockerfile: php.dockerfile
    volumes:
      - ./var/www/html
      - ./storage:/var/www/html/storage
```

```
environment:
  - APP_ENV=production
  - DB_HOST=postgres
  - REDIS_HOST=redis
depends_on:
  postgres:
    condition: service_healthy
  redis:
    condition: service_healthy
networks:
  - qmes-network
deploy:
  resources:
    limits:
      cpus: '2'
      memory: 2G
    reservations:
      cpus: '1'
      memory: 1G

postgres:
  image: postgres:16-alpine
  environment:
    POSTGRES_DB: qmes_saas
    POSTGRES_USER: qmes
    POSTGRES_PASSWORD: ${DB_PASSWORD}
  volumes:
    - postgres_data:/var/lib/postgresql/data
    - ./docker/postgres-init:/docker-entrypoint-initdb.d
  ports:
    - "5432:5432"
  networks:
    - qmes-network
  healthcheck:
    test: ["CMD-SHELL", "pg_isready -U qmes"]
    interval: 10s
    timeout: 5s
    retries: 5
  deploy:
    resources:
      limits:
        cpus: '4'
        memory: 4G

redis:
  image: redis:7.2-alpine
  command: redis-server --requirepass ${REDIS_PASSWORD} --maxmemory 2gb --maxmemory-policy
  volumes:
    - redis_data:/data
  networks:
    - qmes-network
  healthcheck:
    test: ["CMD", "redis-cli", "ping"]
```

```
    interval: 10s
    timeout: 3s
    retries: 5

# — 队列服务 —
horizon:
  build:
    context: ./docker
    dockerfile: php.dockerfile
  command: php artisan horizon
  volumes:
    - ./var/www/html
  environment:
    - APP_ENV=production
    - DB_HOST=postgres
    - REDIS_HOST=redis
  depends_on:
    postgres:
      condition: service_healthy
    redis:
      condition: service_healthy
  networks:
    - qmes-network
  deploy:
    resources:
      limits:
        cpus: '2'
        memory: 2G

# — 定时任务 —
scheduler:
  build:
    context: ./docker
    dockerfile: php.dockerfile
  command: php artisan schedule:work
  volumes:
    - ./var/www/html
  environment:
    - APP_ENV=production
    - DB_HOST=postgres
    - REDIS_HOST=redis
  depends_on:
    - horizon
  networks:
    - qmes-network

# — 消息推送服务(可选扩展) —
webhook-worker:
  build:
    context: ./docker
    dockerfile: php.dockerfile
  command: php artisan queue:work redis --queue=high
  volumes:
```

```

    - ./var/www/html
environment:
    - APP_ENV=production
    - DB_HOST=postgres
    - REDIS_HOST=redis
depends_on:
    - redis
networks:
    - qmes-network
deploy:
    replicas: 2 # 消息推送可扩展

# — BI缓存刷新服务(可选扩展) —
bi-worker:
    build:
        context: ./docker
        dockerfile: php.dockerfile
    command: php artisan queue:work redis --queue=bi
    volumes:
        - ./var/www/html
    environment:
        - APP_ENV=production
        - DB_HOST=postgres
        - REDIS_HOST=redis
    depends_on:
        - redis
    networks:
        - qmes-network
    deploy:
        replicas: 1

networks:
    qmes-network:
        driver: bridge

volumes:
    postgres_data:
        driver: local
        driver_opts:
            type: none
            device: /data/qmes/postgres
            o: bind
    redis_data:
        driver: local
        driver_opts:
            type: none
            device: /data/qmes/redis
            o: bind

```

15.2 14个产品功能与服务映射

#	产品	功能组件	依赖服务	数据库表数
1	QMES生产管理	工单/排程/产品/BOM/甘特图	postgres, redis	6
2	WMS仓储	库存/出入库/仓库/库位	postgres	4
3	QMS质量管理	检验/NCR/趋势/预警	postgres, redis	4
4	设备数采看板	设备/网关/数据/告警	postgres, redis	5
5	ESOP作业指导书	SOP文档/版本/工位绑定	postgres	2
6	安灯呼叫系统	呼叫/响应/超时升级	postgres, redis	1
7	工厂AI应用	视觉检测/模型管理/批量	postgres, redis	2
8	全链路联动版	追溯/综合看板/端到端流程	postgres	1
9	条码标签管理	规则/标签模板/打印/扫码	postgres	4
10	消息通知预警中心	通道/规则/消息/预警	postgres, redis	5
11	报表BI看板引擎	数据源/图表/看板/分享	postgres, redis	5
12	追溯链全景图	链路查询/图谱/分析/导出	postgres	1
13	模具刀具管理	档案/领用/保养/维修/报废	postgres	3
14	文档管理DMS	目录/文档/版本/审批/权限	postgres	5

15.3 按需部署

可根据租户购买的产品套餐选择性部署数据库迁移和启用功能：

```
# 仅部署QMES+QMS+设备数采(基础套餐)
php artisan migrate --path=/database/migrations/0001_0001_* # 平台基础
php artisan migrate --path=/database/migrations/0013_0018_* # QMES
php artisan migrate --path=/database/migrations/0023_0026_* # QMS
php artisan migrate --path=/database/migrations/0027_0031_* # 设备数采

# 部署全部55张表
php artisan migrate --force

# 选择性部署新模块
php artisan migrate --path=/database/migrations/0037_0040_* # 条码标签
php artisan migrate --path=/database/migrations/0041_0045_* # 消息通知
php artisan migrate --path=/database/migrations/0046_0050_* # BI看板
php artisan migrate --path=/database/migrations/0051_0055_* # 模具+文档
```

12.1 数据库相关

故障现象	可能原因	解决方案
连接池耗尽	连接未释放/连接数配置过小	1) 增加 max_connections 2) 启用 PgBouncer
慢查询	缺少索引/全表扫描	1) EXPLAIN ANALYZE 分析 2) 添加索引
死锁	并发事务冲突	1) 检查代码事务顺序 2) 设置 lock_timeout
磁盘IOPS高	未分区/未VACUUM	1) 对device_data分区 2) 定期VACUUM

12.2 Redis相关

故障现象	可能原因	解决方案
OOM (内存不足)	maxmemory配置不够	1) 增大maxmemory 2) 调整淘汰策略
连接超时	连接池满/网络延迟	1) 增加maxclients 2) 检查网络
缓存雪崩	大量key同时过期	1) 增加随机过期时间 2) 本地缓存兜底

12.3 应用相关

故障现象	可能原因	解决方案
502 Bad Gateway	PHP-FPM进程挂掉	1) docker compose restart php-fpm 2) 检查日志
API响应慢	第三方API超时	1) 设置超时时间 2) 异步处理
文件上传失败	磁盘满/权限问题	1) df -h 检查 2) chmod 检查
推送消息失败	企微Webhook配置错误	1) 检查webhook URL 2) 测试推送

12.4 常见问题快速解决

```
# 重启全部服务
docker compose restart

# 仅重启PHP-FPM
docker compose restart php-fpm

# 清空Redis (慎用!)
docker compose exec redis redis-cli FLUSHALL

# 重新构建容器
```

```
docker compose build --no-cache php-fpm
docker compose up -d

# 查看容器资源占用
docker stats

# 查看磁盘占用
du -sh /var/lib/docker/volumes/
```

13. 运维检查清单

13.1 每日检查

- [] Docker容器状态: `docker compose ps` 全部 Running
- [] 数据库连接: `php artisan health:check`
- [] Horizon队列: `php artisan horizon:status` Running
- [] 最新操作日志: 无异常错误
- [] 磁盘使用率: <80%

13.2 每周检查

- [] 慢查询分析: 检查 PostgreSQL 慢查询日志
- [] 备份验证: 从备份恢复测试数据库
- [] 版本更新检查: `git log` 查看是否有新提交
- [] 错误日志分析: 统计本周5xx错误

13.3 每月检查

- [] 数据库VACUUM: `VACUUM ANALYZE`
- [] 分区表维护: 创建新的 `device_data` 分区
- [] 日志轮转: 检查日志文件大小
- [] 备份恢复演练: 完整执行恢复流程
- [] 性能基准测试: 对比上月响应时间

13.4 部署检查清单

- [] .env 配置正确 (APP_DEBUG=false)
- [] HTTPS已配置
- [] JWT_SECRET已生成
- [] 数据库迁移完成
- [] 种子数据已导入
- [] 前端已构建并部署

- ☐ Horizon队列正在运行
- ☐ Nginx配置正确
- ☐ 防火墙端口已开放
- ☐ 监控告警已配置

文档结束